PLATFORM ENGINEERING FOR THE AI-NATIVE ENTERPRISE · PLAYBOOK

The platform, not the model, decides whether enterprise AI delivers.

Three things are true at once in 2026: AI adoption is near universal, most AI projects still fail in production, and the ones that succeed share a mature platform underneath. This playbook makes the case that the foundation decides, maps the five-layer pyramid and the four-pillar control plane, gives you two accelerators on one Azure and GitHub foundation, and turns the strategy into a 90/180/365-day sequence measured in DORA and GitHub AI Credits.

AUTHOR

Paula Silva

ROLE

AI-Native Software Engineer

CONTACT

in/paulanunes

READING TIME

~ 45 minutes, end to end

5	6	4
PYRAMID LAYERS	CHAPTERS	CNCF PILLARS

2.5x

ROI COMPRESSION, PLATFORM-MATURE

CHAPTERS

Chapter 00

Why the foundation decides

The three truths of 2026, the amplification thesis from DORA 2025, the five failure modes when the platform is weak, and the evidence a CTO can take to the board.

The Case →

Chapter 01

The Five-Layer Pyramid

Platform, Context, Cognitive, Intent, and Agentic in strict dependency order, the stacking rule that governs the build, and where GitHub Copilot maps onto each layer.

Architecture →

Chapter 02

The Four Pillars as Control Plane

Golden Paths, Guardrails, Safety Nets, and Manual Review re-expressed to govern agents with the machinery a platform already runs, seen through the Copilot lens and the 2026 Dual Mandate.

Control Plane →

Chapter 03

Two Accelerators, One Foundation

Three Horizons on Red Hat Developer Hub and Open Horizons on Backstage, the Azure and GitHub foundation that does not change, and a one-day method to choose by client archetype.

The Choice →

Chapter 04

The 90/180/365-day sequence

The phased execution plan from raw Azure infrastructure to production agents in a year, and the three non-negotiables that decide whether the sequence lands.

Execution →

Chapter 05

Maturity, measurement, and ROI

The CNCF five-dimension maturity model, the DORA Four Keys plus the 2026 AI extensions and agent health, cost governance in GitHub AI Credits, and a business case defensible at the CFO level.

The Payoff →

READING TIPS

> **KEYBOARD SHORTCUTS**
>
> Press `/` to search. Use `j` and `k` to move between chapters. Press `t` to toggle theme. Press `g` to go to top.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

00 THE CASE

Why the foundation decides: the platform, not the model, sets the ceiling.

Three things are true at once in 2026. AI adoption is nearly universal. Most AI projects still fail in production. And the organizations that succeed share one trait: a mature platform underneath. This chapter argues that the foundation, not the model, decides the outcome. It reads the 2025 and 2026 evidence the way a CTO has to read it, as a spending decision with a measurable downside.

The three truths of 2026

Start with adoption. McKinsey's State of AI 2025 finds that 78% of organizations now use AI in at least one function. In the enterprises I work with across Latin America, that adoption usually has one name: GitHub Copilot, and increasingly Copilot in agent mode running across the whole repository estate. Gartner projects that 40% of enterprise applications will embed task-specific agents by the end of 2026, up from under 5% in 2024. Adoption is settled. The open question is not whether teams use AI. It is what that AI does to the delivery system it lands on.

The second truth is less comfortable. The MIT NANDA study The GenAI Divide, published in 2025 across more than 350 companies, found that 95% of generative AI pilots fail to deliver measurable value. Failure there is defined operationally: the pilot never reached sustained production with attributable business outcomes. The third truth reconciles the first two. The organizations that do reach production are, with striking consistency, the platform-mature ones. NANDA attributes the gap not to weak models but to missing system-level context and feedback loops. That is a platform property, not a model property. The same tool succeeds in one estate and fails in the next, and what changed is the foundation underneath.

The amplification thesis

The mechanism behind those three truths has a name, and the strongest evidence for it is DORA. The 2025 DORA report is built on nearly 5,000 survey responses and more than 100 hours of qualitative research. Its central finding is that AI does not behave as a universal productivity input. It behaves as an amplifier of whatever delivery system it inherits. Teams on loosely coupled architectures with fast feedback loops turn the same AI tools into real throughput. Teams on tightly coupled systems with slow feedback loops watch the added change volume become instability. Same input, opposite outcomes. DORA also reports that 90% of surveyed organizations have adopted at least one internal platform, which moves the conversation from whether to build a platform to how mature yours is.

This is where the GitHub Copilot lens matters. When an enterprise turns on agent mode across thousands of repositories, it amplifies the delivery system those repositories already encode. The platform's job is to feed the agent authoritative context so the amplification runs in the right direction. In practice that means a maintained .github/copilot-instructions.md in each repository, MCP servers that expose the service catalog, policy, and runbooks, and custom agents scoped to sanctioned tasks. It also makes the cost visible. Copilot usage is billed as GitHub AI Credits, where one credit is one US cent, and since June 2026 that line item shows exactly how much context reconstruction the estate is paying for. A weak foundation makes that number grow. A mature one turns it into throughput.

> DEFINITION
>
> DORA 2025 puts it in one line: AI doesn't fix a team; it amplifies what's already there. Read it as a spending rule. AI investment without platform investment is investment in amplifying whatever is already there, in whichever direction it already points.

Five failure modes when the foundation is weak

When the platform is under-built relative to the AI scaled on top of it, five failure modes recur. The first is Triple Debt. AI generates technical debt through unreviewed

code, cognitive debt through knowledge that lives only in prompts and chat threads, and intent debt through goals that are implicit or contradictory. The three accumulate in parallel and none resolves the others. The second is Shadow Platforms. When the central platform is hard to use, each team builds its own: bespoke pipelines, conflicting infrastructure modules, its own MCP servers and custom agents. Gartner's shadow IT data puts that distributed spend at two to three times the cost of a real platform, invisible in any single budget line.

The third is Context Rot. Service catalogs, documentation, and lineage go stale silently, and an agent reasoning over a stale copilot-instructions.md or a stale MCP source returns answers that are plausible and wrong. At agent volume, a small rot rate becomes a large error rate. The fourth is Security Regression. CrowdStrike's 2026 Global Threat Report documents that organizations deploying AI coding assistants before maturing their platform security posture saw a 38% rise in exploitable vulnerabilities in the first twelve months. The regression disappears where the platform enforces supply-chain and identity controls at admission. The fifth is the 100:1 problem. By 2028 the agent-to-human ratio in a typical enterprise is projected to reach roughly 100 to 1. Manual supervision of agent identity, credentials, and audit does not scale to that ratio. Only a platform that treats agents as first-class workloads does.

> DEFINITION
>
> Five failure modes, one cause: AI runs faster than ad-hoc process can govern it. Triple Debt. Shadow Platforms at two to three times the cost. Context Rot. Security Regression. The 100:1 ratio by 2028. Each is preventable by the platform. None is preventable by a better model.

The evidence a CTO can take to the board

For an executive, the decisive numbers are the ones that separate intent from capability. IDC research on agentic AI finds that while roughly half of surveyed organizations report ten or more agents in some form of production, fewer than 7% run even one use case in full production. The willingness is broadly distributed; the capability is concentrated in the organizations that built the platform first. That single

gap is more useful than any productivity headline, because it is observable inside your own estate today. Count how many agent use cases are actually in full production, not in pilot. The answer tells you where your foundation stands.

The upside is documented too. Forrester's Platform Engineering research ties productized internal platforms directly to business outcomes: 77% of companies attribute measurable time-to-market gains to their internal developer platform, and 85% report positive revenue impact. Put the two findings together and the argument closes. The binding constraint on enterprise AI is not which model sits in the Copilot picker. Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro, and the GPT-5.x line are all one dropdown away, and any of them will amplify what it inherits. The constraint is the platform that feeds them context, governs their actions, and pays for their tokens. That is why the foundation decides. Build it first.

References

Primary sources for the three truths, the amplification thesis, the failure modes, and the executive evidence.

- McKinsey, The State of AI 2025, the adoption baseline: 78% of organizations now use AI in at least one function.
- MIT NANDA, The GenAI Divide: State of AI in Business 2025, the 95% pilot failure rate and its diagnosis as a context and feedback problem.
- DORA 2025, Accelerate State of DevOps Report, the amplification finding built on nearly 5,000 responses and over 100 hours of qualitative research.
- Gartner, Over 40% of Agentic AI Projects Will Be Canceled by End of 2027, the cancellation projection driven by cost, unclear value, and weak risk controls.
- IDC, Agentic AI Platforms and Strategies, the readiness gap between agents claimed in production and use cases actually in full production.
- Forrester Wave, Platform Engineering Solutions Q1 2026, internal developer platforms tied to time-to-market and revenue impact.
- CNCF Platforms Whitepaper, the reference definition of the platform capabilities this chapter argues for.

- CrowdStrike, 2026 Global Threat Report, the security regression measured when AI coding assistants land ahead of platform controls.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

01 THE ARCHITECTURE

The Five-Layer Pyramid: five layers in strict dependency order, from platform to agents.

The AI-native enterprise is not one thing. It is a pyramid of five layers, each depending on the one below it. Platform Engineering is the base. Context, Cognitive, Intent, and Agentic stack on top, in that order. The order is not advice, it is a structural law. This chapter maps the five layers, names what each one owns, shows where GitHub Copilot fits, and explains why building agents on a missing foundation produces an amplified mess instead of agents.

The stacking rule: each layer sits in strict dependency order

The AI-native enterprise is a pyramid, not a layer cake. The base layer is broader, costs more, and supports everything above it. Each layer above is narrower and explicitly conditional on the layer below. The construction order is not a preference, it is a structural law: you cannot build layer N+1 faster than layer N. You can build them in parallel, but only if you accept the debt that comes from the gap. A pyramid is assembled bottom-up. You cannot suspend the apex in the air and build the base around it later. The Platform layer is built first because every layer above it needs platform primitives to exist. Remove a stone and you do not get a smaller pyramid, you get a collapse.

The rule is empirically grounded. DORA 2025, built on nearly 5,000 survey responses and more than 100 hours of qualitative research, found that 90% of organizations have adopted at least one internal platform, with a direct correlation between platform quality and what AI delivers. The report distills the mechanism into one line: AI does not fix a team, it amplifies what is already there. That sentence is the pyramid restated. AI consumes the stack bottom-up. A weak base produces a weak

top, regardless of how strong the top looks in isolation. Same AI tools, opposite outcomes, and the platform is the difference.

L1 Platform to L5 Agentic: what each layer does and what it owns

Layer 1, Platform Engineering, is the self-service, policy-governed, observable substrate. It owns the CNCF four pillars: Golden Paths, Guardrails, Safety Nets, and Manual Review. Golden Paths take a developer or an agent from "I want to build X" to a compliant running system without filing a ticket. Guardrails are preventive, enforced at admission rather than detected after deployment. Safety Nets recover from the failures guardrails did not prevent. Manual Review keeps a human in the loop for irreversible actions. The concrete artifacts are a Backstage portal, GitOps, policy-as-code, observability with OpenTelemetry, and workload identity. Layer 2, Context, is codified enterprise knowledge in machine-consumable form: the service catalog, documentation-as-code, lineage, vector stores, and MCP servers. It depends on L1 because the platform owns the identity, observability, and lifecycle primitives that keep context trustworthy.

Layer 3, Cognitive, is the models, embeddings, and evaluation that consume context to produce outputs, mediated through a model gateway the platform provisions and governs. Layer 4, Intent, is the codified goals, specifications, and policies that agents plan against. Intent without context is hallucination; context without intent is description without direction. Layer 5, Agentic, is autonomous, goal-driven agents that combine intent, context, and cognition to take action. This is the visible layer for most AI investment in 2026, and it is the layer most likely to fail when the layers beneath it are missing. An agent is a model plus a harness. The model is the pluggable part. The harness is everything around it: tools, memory, context, validation, and gates. The harness is the four lower layers in operational form, which is why an agent in production is the integrated test of whether the foundation is sound.

The Copilot lens: where GitHub Copilot maps onto the pyramid

GitHub Copilot makes the pyramid concrete. The context layer is where MCP servers and the .github/copilot-instructions.md file live. MCP servers expose enterprise systems (Azure, GitHub, monitoring, ticketing) to Copilot through one standard protocol, so the agent reads enterprise reality instead of guessing it. The copilot-instructions file is repository-scoped context: the conventions, the architecture, and the rules the codebase already encodes, written down where every Copilot session reads them. Both are Layer 2 artifacts. Neither is a model. They are the knowledge the model reasons over, and they are what stops a capable model from producing plausible output that does not match how the codebase actually works.

The cognitive layer is the Copilot model picker. In 2026 the picker offers Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro, and the GPT-5.x family. No single model is best for every task, so the picker is the model gateway made visible: you route a fast, cheap model to a scoped edit and a stronger reasoning model to a hard refactor. Usage is billed in GitHub AI Credits, where one credit is one US cent, on the metering that took effect on June 1, 2026. Copilot agent mode and custom agents are the Layer 5 surface: they combine the instructions (intent and context), the picked model (cognition), and the platform controls to act across the repository. The picker without the instructions and without MCP is cognition with no grounding.

Jumping to L5 without L1 to L4 does not produce agents

The most common failure is to fund the Agentic layer, which is visible and easy to fund, without funding the Platform and Context layers, which are invisible and unglamorous. The result is an agent that works in a demo and fails in production. It has no grounded context, so it hallucinates. It has no governed permissions, so it acts outside its scope. It has no observable trajectories, so no one can debug or certify what it did. Gartner estimates that 40% of agentic AI projects will be cancelled by 2027 on cost, unclear value, and weak risk controls. Structurally, those are agent deployments on unfit foundations: a pyramid missing one or more lower layers.

An organization that jumps to L5 without L1 to L4 in place does not produce agents. It produces an amplified mess. The remediation is never to fix the agent. It is to retrofit the missing layers, in dependency order, which costs more after the agent has been demoed to executives than it would have cost to build correctly the first time. The Copilot version of the same mistake is handing a team the strongest model in the picker with no copilot-instructions file and no MCP servers wired. The model is not the constraint. The context and the platform beneath it are. Start at L1. Commit to the foundation before the agents.

> DEFINITION
>
> The stacking rule in one line: you cannot build layer N+1 faster than layer N, and you can only build them in parallel if you are willing to carry the debt that comes from the gap. Jumping to L5 without L1 to L4 does not produce agents. It produces an amplified mess.

References

Primary sources for the five-layer pyramid, the stacking rule, and the platform and context foundations it rests on.

- CNCF Platforms Whitepaper, the reference definition of the platform layer as a self-service, policy-governed, observable substrate.
- CNCF Platform Engineering Maturity Model, the source for the four pillars that Layer 1 owns: Golden Paths, Guardrails, Safety Nets, and Manual Review.
- DORA 2025, State of AI-assisted Software Development, the empirical basis for amplification: AI amplifies what is already there.
- Backstage, the open-source developer portal that materializes Golden Paths and the service catalog at the base of the pyramid.
- Model Context Protocol Specification, the open standard MCP servers use to expose enterprise systems as agent-consumable context.
- Vishnyakova et al., From Context Engineering to Multi-Agent Architecture, the dependency hierarchy from context to intent that the pyramid's upper layers formalize.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

02 THE CONTROL PLANE

The Four Pillars as Control Plane: how a platform governs agents with the machinery it already runs.

Golden Paths, Guardrails, Safety Nets, and Manual Review were engineered to govern human delivery. In 2026 the same four pillars govern agents. The re-expression is operational, not metaphorical. A platform that codified the four pillars for developers already holds most of the AI control plane. This chapter walks the four pillars as the surface that admits, contains, recovers, and approves every action an agent takes.

Golden Paths and Guardrails: the sanctioned surface and the containment field

An agent is a service. That single equivalence is what lets a platform govern agents with the machinery it already runs. A Golden Path is an opinionated, self-service workflow that carries a developer, or an agent, from "I want to build X" to a compliant, running system in minutes. For agents, the Golden Path becomes the scaffold by which the agent is instantiated. The rule is blunt: if it is not a Golden Path, the agent does not run it. The Golden Path is the sanctioned execution surface, and everything outside it is off the surface.

Guardrails are the other half. They are preventive, not detective. A detective control flags a violation after deployment, which is too slow when agents act faster than any human review loop. A preventive control rejects the violation at admission, before it lands. The same OPA Gatekeeper or Kyverno policy that rejects a non-compliant Kubernetes manifest rejects a non-compliant agent configuration. The same Workload Identity that issues short-lived credentials to a service issues them to an agent. The wrong action is not warned against. It is made impossible.

The agent containment field

On top of the shared guardrails, the platform adds four that are specific to agents. A permission-scope guardrail declares the data an agent can read and the actions it can take, enforced at runtime. An output-filter guardrail redacts and classifies before output reaches the consuming system. A data-residency guardrail routes traffic only to model endpoints in permitted regions. A cost-circuit-breaker guardrail terminates a session when spend crosses a ceiling. Together these four form the agent containment field, enforced at admission and again at runtime.

> DEFINITION
>
> Golden Paths are the only sanctioned way an agent starts. Guardrails are what it must satisfy to keep running. One admits, the other contains. Neither is optional.

Safety Nets and Manual Review: recover the reversible, gate the irreversible

Guardrails prevent. Safety Nets recover from what prevention missed. The most foundational Safety Net is GitOps reconciliation: Argo CD or Flux continuously compares the deployed state to the declared state in Git and corrects drift with no human in the loop. Re-expressed for agents, the same loop becomes decision-level rollback. When an agent output drifts from its declared specification, when cost-per-task climbs, or when success rate drops, the Safety Net rolls the decision back or escalates. Loop detection aborts a session that repeats without progress. An SLO-driven freeze halts an agent population the moment a service objective is breached, the same way a canary halts a bad rollout.

Manual Review is the last pillar and the one that grows in importance as everything else accelerates. Its job is to insert human judgment at the boundaries that warrant it, and nowhere else. Routine changes flow through Guardrails and Safety Nets untouched. Irreversible actions hit an approval gate: a production data migration, a permission-scope expansion, a spend above a threshold. For agents, the critical gate is capability expansion. Before an agent gains a new data source or a new action, a human approves it. This is the AI equivalent of an elevated production deploy, and it is the one gate a mature platform never automates away.

The auditor agent

The friction is intentional, but the evidence gathering does not have to be manual. An auditor agent reads the proposed change, retrieves the service catalog entry, the deployment and incident history, the cost impact, and the policy compliance status, then assembles a structured review packet. The human spends time on the decision, not on collecting the facts. Cognitive load on the reviewer drops. Accountability stays exactly where it was. This is a clean example of AI augmenting the four pillars rather than replacing them.

The Copilot lens: guardrails an agent satisfies at runtime

The four pillars stay abstract until they touch a tool a developer already uses. GitHub Copilot in agent mode is where I ground them. A repository .github/copilot-instructions.md file is a guardrail written as text. It states the conventions, the boundaries, and the sanctioned patterns the agent must honor on every task in that repository. Custom agents narrow the surface further: each one is scoped to a domain, carries its own instructions, and can reach only the tools and context you grant it. That scoping is a permission-scope guardrail expressed in the Copilot layer rather than the cluster layer.

MCP is how the agent reaches context and tools under governance. An MCP server exposes a defined set of resources and actions, and the platform decides which agents connect to which servers. The model itself comes from the Copilot picker, and the picker is its own kind of guardrail: an agent can run only on a model the organization has enabled, which today means Claude Fable 5, Claude Opus 4.8, Claude Sonnet 5, Claude Haiku 4.5, Gemini 3.1 Pro, or a GPT-5.x model. Since June 1, 2026, premium usage is billed in GitHub AI Credits, where one credit equals one US cent. That billing surface is the cost-circuit-breaker guardrail in practice: spend is metered per request, attributable per team, and capped before it runs away.

DEFINITION

In the Copilot lens, .github/copilot-instructions.md is the guardrail, the custom agent is the permission scope, MCP is the governed context boundary, and GitHub AI Credits are the cost circuit breaker. The four pillars are not a diagram. They are configuration.

The 2026 Dual Mandate

The four pillars describe how a platform governs agents. The Dual Mandate describes what a platform team owes the enterprise while it does so. Mandate A is to augment the platform itself with AI for internal velocity. Agents triage alerts, draft runbook fixes, propose infrastructure changes, review policy-as-code, and generate Golden Path invocations in the IDE. The goal is that platform improvements ship faster, and every improvement that lands earlier compounds across every team consuming the platform. The discipline that keeps Mandate A honest is measured adoption. If fewer than a third of the team daily actions actually flow through the agents, the tools are theater, not integration.

Mandate B is to expose AI as platform primitives that application teams consume through Golden Paths for external velocity. Inference through a governed model gateway, vector storage, agent runtime, evaluation, and agent observability all become first-class surfaces, the same way Kubernetes and Postgres became first-class a decade earlier. Azure AI Foundry is the reference gateway in both accelerators. The goal is that application teams ship AI-using products in weeks instead of quarters. A platform that runs only Mandate A produces a faster platform team that does not move the enterprise. A platform that runs only Mandate B becomes an AI buffet with no velocity of its own, and eventually the bottleneck. Mature platforms run both.

DEFINITION

The maturity test is one question in two halves. Can you name one recurring platform-team task an agent has absorbed, and one application team whose every AI primitive is a Golden Path off your platform? Two yeses is a mature platform. One yes is half a platform. Neither answer is optional in 2026.

References

Primary sources for the four pillars, the agent control-plane re-expression, and the Copilot execution surface.

- CNCF Platform Engineering Maturity Model, the reference taxonomy for Golden Paths, Guardrails, Safety Nets, and Manual Review, and the per-pillar maturity levels.
- Backstage Software Templates (Scaffolder), the single developer-facing entry point that makes a Golden Path the sanctioned execution surface.
- Open Policy Agent Gatekeeper, admission-time policy enforcement, the mechanism that turns a guardrail into a compile-time error for infrastructure.
- Argo CD Documentation, GitOps reconciliation, the foundational Safety Net that corrects drift back to declared state.
- GitHub Copilot Documentation, the reference execution surface for agent mode, custom agents, and .github/copilot-instructions.md.
- Microsoft Entra Agent ID, managed agent identity, the primitive that lets the same guardrails govern agents and services uniformly.
- Azure AI Foundry Documentation, the governed model gateway at the center of Mandate B.
- DORA Report 2025, the amplification finding: AI does not fix a team, it amplifies what the platform already is.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

03 THE CHOICE

Two Accelerators, One Foundation.

Both accelerators reach the same place. Three Horizons and Open Horizons build the same AI-Native platform, on the same Azure and GitHub foundation, and both terminate at the same Horizon 3. What differs is the stack in the middle: whether the internal developer platform runs on Red Hat Developer Hub with commercial support, or on Backstage that your own team operates. This chapter lays out both, names the foundation that does not move, and gives you a one-day method to choose.

Three Horizons: Red Hat Developer Hub on Azure Red Hat OpenShift

Three Horizons is the accelerator for clients with a prior Red Hat investment or a commercial-support requirement. The internal developer platform is Red Hat Developer Hub. The cluster is Azure Red Hat OpenShift, with the control plane operated jointly by Microsoft and Red Hat under a joint SLA. CI runs on OpenShift Pipelines, GitOps on OpenShift GitOps, and Red Hat carries enterprise support around the clock. Regulated sectors that require every platform component to carry its own certification, FedRAMP, FIPS, or Common Criteria, tend to land here, because the Red Hat stack components hold those certifications and the open-source equivalents do not.

The Red Hat subscription bundle typically runs 10 to 30 percent of total platform spend, scaling with cluster size, user count, and add-on coverage. That line item buys a support model. The platform team's internal requirement stays operational, running and customizing the platform, rather than deep engineering, because upstream-stack issues route to Red Hat instead of to your own developers. For a bank under BACEN rules or an energy operator with a RHEL fleet at scale, Red Hat Developer Hub on OpenShift is continuity, not overhead.

Open Horizons: Backstage on Azure across three horizons

Open Horizons is a single GitHub template repository that provisions the same platform on Azure using CNCF open-source components. The contents are concrete: 16 Terraform modules for the Azure Kubernetes Service foundation, 22 Golden Path templates spread across the three horizons, 13 MCP server configurations, and 17 GitHub Copilot custom agents wired for the Dual Mandate. Backstage is the internal developer platform, and your own team operates it. A team adopting Open Horizons does not write a platform from scratch. It customizes an opinionated starting point, which is an order-of-magnitude difference in time, cost, and risk.

The GitHub Copilot lens is explicit here. The 17 custom agents run in agent mode; the conventions each agent follows live in .github/copilot-instructions.md; and the 13 MCP server configurations give every agent typed, role-scoped access to Azure, GitHub, Terraform, Kubernetes, and the observability stack. Agent consumption bills as GitHub AI Credits, where one credit is one US cent. The trade against Three Horizons is direct. There is no Red Hat subscription line item, and the internal capability requirement is broader, because upstream issues route to the community and to your own engineers rather than to a vendor SLA.

The foundation that does not change

Under both accelerators the substrate is identical, and it is worth naming precisely because it is what does not move. Azure is the cloud. GitHub is the source of record and the CI surface. One identity plane runs through Microsoft Entra, so people and agents are governed in the same place. One GitHub Copilot model picker exposes the same models to every developer and every agent regardless of the accelerator above it, including Claude Opus 4.8 and Sonnet 5, Gemini 3.1 Pro, the GPT-5.x line, and lighter options such as Haiku 4.5 for cheap, high-volume work. The CNCF four pillars, the H1, H2, and H3 maturity model, and the Dual Mandate are the same on both.

This is why the decision is a stack choice, not a strategy choice. The strategy does not vary: the same five-layer pyramid, the same four pillars, the same Dual Mandate, the same 90 to 180 day path to Horizon 3. What varies is the middle of the stack, the

internal developer platform, the cluster, the CI substrate, the supply-chain tooling, and whether Red Hat subscriptions sit in the contract. Both paths end at the same Azure AI Foundry destination, exposing the same Golden Paths, the same MCP servers, and the same agents.

> DEFINITION
>
> Same destination, same foundation, different stack on top. Azure for cloud, GitHub for source and CI, one Microsoft Entra identity plane for people and agents, one GitHub Copilot model picker for both. The accelerator you pick changes the middle of the stack, never the strategy and never the endpoint.

Choosing by client archetype

Most clients tilt before they analyze, and the tilt is usually right. A digital-native retailer already on Azure Kubernetes Service, with an OSS-first culture and a deliberately small vendor count, tilts to Open Horizons. A BACEN-regulated bank with an on-premise OpenShift estate and a commercial-support clause in procurement tilts to Three Horizons. The tilt comes from procurement profile, engineering capability, regulatory posture, and vendor strategy, not from a feature checklist. Roughly a quarter to a third of enterprises sit in a middle cohort where neither tilt is dominant, and that is where a method matters.

The one-day discovery

For the middle cohort I run a one-day discovery workshop. Two weeks ahead, the client sends an inventory: current Red Hat footprint, current Kubernetes footprint, GitHub investment, engineering team profile, and procurement and regulatory posture. The day is five working sessions: strategy alignment, a walk through the ten-dimension stack comparison, a three-year total cost of ownership model built on the client's own numbers, an honest self-assessment of the team's capacity to operate upstream Backstage and Azure Kubernetes Service, and a recommendation. The output is one page: the tilt across ten dimensions, two TCO numbers, a capability score, and one recommendation the executive sponsor can sign.

The discipline is that the team leaves with a recommendation, not a menu. Telling a client that both are good and asking what they prefer pushes the decision back onto the people least equipped to make it. Cost alone should not decide, because the Red Hat line item is the smaller component and the cost of a misfit accelerator is larger than the subscription it saves. In the rare engagement where the workshop lands genuinely close, the answer is a thirty-day pilot of both with two real teams. Either way the endpoint is the same Horizon 3, so a decision revisited at the end of year one migrates the internal developer platform and the cluster without rebuilding the strategy.

References

Primary sources for the two accelerators, the shared foundation, and the decision framework.

- Red Hat Developer Hub, the enterprise-supported internal developer platform in the Three Horizons stack.
- Azure Red Hat OpenShift, the jointly operated cluster substrate for Three Horizons.
- Backstage, the open-source developer portal your own team operates in Open Horizons.
- Azure Kubernetes Service, the cluster substrate for Open Horizons.
- GitHub Copilot Documentation, agent mode, custom agents, and the shared model picker on both accelerators.
- Model Context Protocol Specification, the open standard behind the 13 MCP server configurations.
- Microsoft Entra, the single identity plane for people and agents under both accelerators.
- The Forrester Wave: Platform Engineering Solutions, Q1 2026, external reference for the internal developer platform decision.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

04 THE EXECUTION

The 90/180/365-day sequence: from raw infrastructure to production agents in a year.

The strategy only matters if it ships. This chapter turns the argument into a calendar: a 90, 180, and 365-day sequence that takes an enterprise from raw Azure infrastructure to production agents inside a year. The sequence respects one rule above all others. You build each horizon on the horizon below it, and you refuse to skip the foundation.

Days 0 to 90: build the H1 foundation

The first ninety days produce a foundation that runs in production, not a demo. You provision the substrate with Terraform: an AKS cluster with Workload Identity, Azure Key Vault for secrets, ACR for images, a zero-trust VNet, and the network security groups that go with it. On top of that you stand up Backstage with the first H1 golden paths, so a developer goes from intent to a compliant service in minutes. Argo CD wires GitOps with environment-specific sync policies. Prometheus and Grafana light up the platform dashboard. Three application teams onboard through the golden paths, and the DORA Four Keys go live for all three.

The target for day ninety is concrete: time-to-first-PR under one day, against a typical baseline of two to four weeks. Two things make that number real. The first is the golden path itself, which encodes the enterprise's current best practice as a versioned template rather than a wiki page. The second is the repository instruction file. A curated .github/copilot-instructions.md tells GitHub Copilot in agent mode how this codebase actually works, so the agent scaffolds inside the golden path instead of guessing. DORA is not a scorecard here. It is the filter through which you watch AI's effect on delivery, so you instrument the keys for measurement, not for ranking.

Months 3 to 6: the H2 enhancement

With the foundation credible, months three to six widen it. The supply chain gets hardened: signed images, SLSA provenance, and admission controllers that reject unsigned workloads. A service mesh (Istio in Open Horizons) handles traffic management, mTLS, and progressive delivery. The service catalog fills out past eighty percent coverage of eligible workloads, and documentation-as-code becomes the org-wide default for every new service. This is where the four CNCF pillars finish getting wired: Golden Paths, Guardrails, Safety Nets, and Manual Review.

The point of H2 is that the same four pillars now serve two populations. Golden paths are the sanctioned execution surface for a human developer and for an agent alike. Guardrails written to prevent human misconfiguration become the containment field for an agent's runtime permissions. Safety nets that roll back a bad deploy become the reconciliation loops that catch prompt injection and cost spikes. By day 180 the target is time-to-first-production for a new microservice under one week, platform adoption above half of eligible workloads, and the DORA keys moving in the right direction. The platform reaches the Scalable level on most CNCF maturity dimensions.

Months 6 to 12: the H3 innovation

The last two quarters put AI workloads into production on the same substrate. The first H3 template ships: a RAG application or a Foundry agent. Multi-agent systems get explored through their own template. Every H3 template inherits H1 and H2 primitives, so a Foundry agent is structurally a microservice plus a model binding plus an evaluation pipeline, not a separate architecture. Agent observability completes: trajectories, cost, output sampling, and evaluation results all surface in one dashboard.

The test of the whole year is here. Agent mode consumes the same golden paths as developers. A custom agent that scaffolds a service runs the identical template a human would run, satisfies the identical guardrails, and passes through the identical manual-review gate. It runs against a model from the Copilot picker, Claude Opus 4.8 for the hard reasoning and Sonnet 5 or Haiku 4.5 for the routine work, and every call

is metered in GitHub AI Credits, where one credit is one cent. MCP servers give those agents governed access to the catalog and the platform's tools. By day 365, time-to-first-production for an AI workload has dropped from the nine to eighteen month baseline to 90 to 180 days, and the agent fleet has grown from one or two agents to ten or twenty in production.

The three non-negotiables

Three commitments separate the enterprises that finish this sequence from the ones that stall in month four.

Treat the platform as a product, not a project

A project has an end date and a project manager. A product has neither; it has a roadmap and a product manager who owns it. The distinction is not cosmetic. The platform's value compounds over years. The Forrester compression is a one-year effect, and the McKinsey ROI multiple is a three-year effect, and both require the platform to exist as a continuous, owned, funded thing.

Refuse to skip H1

The pressure to show H3 in ninety days is real, and it is wrong. Every shortcut past the foundation surfaces later as debt: security regressions, shadow platforms, context rot. The discipline that prevents it is authority. The platform team must be allowed to say no to H3 work that depends on H1 work that has not landed, and leadership must defend that no when the sponsor pushes back.

Ship the first Copilot custom agent early

Even a minimal custom agent that answers how do I do X on our platform pays for itself inside a quarter, because it absorbs the onboarding load that otherwise lands on the platform team. Ship it in week eight to ten, not week thirty. It is the first executable proof that the platform is being built for agents and humans at once, and its trajectory data makes the next version better.

DEFINITION

The sequence is the contract: H1 in ninety days, H2 by day 180, H3 by day 365. Build each horizon on the one below it, treat the platform as a product, and ship the first custom agent early. Skip the foundation and you do not save time; you defer the bill.

References

Primary sources for the phased sequence, the maturity progression, and the delivery metrics behind the ninety-day targets.

- DORA, 2025 Accelerate State of DevOps Report, the delivery metrics: the DORA Four Keys as the filter for AI's effect, with elite performers seeing 20 to 30 percent productivity gains.
- CNCF, Platform Engineering Maturity Model, the Operational to Scalable to Optimizing progression the sequence tracks against.
- Forrester, The Forrester Wave: Platform Engineering Solutions Q1 2026, the platform engineering research: 77 percent of companies attribute measurable time-to-market gains and 85 percent report positive revenue impact to internal developer platforms.
- CNCF, Platform Engineering Whitepaper, the four pillars and golden paths the sequence wires across H1 and H2.
- Backstage, open framework for building developer portals, the developer portal and golden-path templating layer used from day one.
- McKinsey & Company, The State of AI, the multi-year ROI evidence behind treating the platform as a product, not a project.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps

05 THE PAYOFF

Maturity, measurement, and ROI: what good looks like, and how to defend it.

A platform that cannot be measured cannot be defended. This chapter gives the platform team three measurement layers and a business case. The CNCF Platform Engineering Maturity Model places the platform on a Provisional to AI-Native ladder. The DORA Four Keys, extended with 2026 AI metrics and a separate agent health set, surface delivery and agent behavior through four dashboards. Cost governance runs on GitHub AI Credits. It closes with the two field numbers and the external evidence that defend the investment at the CFO level.

The CNCF maturity model: five dimensions, five levels

The CNCF Platform Engineering Maturity Model scores a platform on five independent dimensions. Investment asks how the platform is funded and staffed. Adoption asks how broadly and deeply it is used. Interfaces asks how users interact with it. Operations asks how the platform itself is run. Measurement asks what the platform team measures about itself and its users. The dimensions are independent. A platform can be well funded on Investment and still route developers through wiki pages on Interfaces. Treating maturity as five separate axes is what keeps the score honest.

Each dimension moves through five levels. Provisional is ad hoc and fragmentary. Operational means a dedicated effort exists but is not yet productized. Scalable means users self-serve through a stable interface with documented SLAs. Optimizing means the dimension produces metrics that drive its own improvement. AI-Native means the dimension is consumed by agents at scale, on the same primitives that serve human users. The common mistake is to average the five

dimensions into one flattering number. The honest score is the minimum. A platform that is AI-Native on Adoption but Provisional on Investment is a Provisional platform.

AI-Native Level 5 closes the Dual Mandate

AI-Native is the target that closes the Dual Mandate from chapter 04. At Level 5, Investment carries a dual-mandate budget that funds both AI augmenting the platform team and AI primitives for application teams. Adoption includes agents consuming the same Golden Paths as human developers. Interfaces expose those Golden Paths through MCP servers, so an agent's "deploy a service" call invokes the same template a person would. Operations are agent-operated with human review. Measurement spans DORA, cost, AI ROI, and agent health. Few enterprises were fully AI-Native in early 2026. The realistic target is Optimizing across all dimensions, with AI-Native on at least Investment and Adoption.

> DEFINITION
>
> The honest maturity score is the minimum across the five dimensions, not the average. The weakest dimension determines the platform's effective maturity, because the weakest dimension is the failure mode AI amplifies first.

DORA Four Keys, the AI extensions, and agent health

The DORA Four Keys are still the lens on delivery performance: deployment frequency, lead time for changes, change failure rate, and time to restore service. The 2025 DORA report, built on nearly 5,000 survey responses, states the finding plainly: AI does not fix a team, it amplifies what is already there. The Four Keys are the filter through which that amplification becomes visible. In mature systems AI compounds the keys; in immature systems the same AI regresses them.

In 2026 the platform team extends DORA with four AI-specific families. AI usage covers the number of agents in production, request volume, and error rate. AI quality covers evaluation pass rate and its drift over time, plus the fraction of outputs an LLM judge approves. AI cost covers credit cost per workload, latency, cache hit rate,

and per-model utilization. AI behavior covers permission-scope compliance and audit-trail completeness. These families are most useful per workload, not in aggregate. A single workload regressing on lead time while the aggregate improves is the early warning that the platform is absorbing work it is not yet ready to support.

Agent health is a separate metric set

Agent health is distinct from delivery performance. An agent can deploy often and fail rarely while its cost drifts, its scope expands, and its outputs grow inconsistent. The minimum 2026 set covers identity compliance (every action attributable to a managed identity, target 100%), permission-scope adherence (no successful out-of-scope actions), cost per session (a stable distribution, with sessions above three times the median flagged), loop detection, evaluation pass rate, output quality, error rate, and drift detection. These surface through the four dashboards both accelerators ship: platform, cost, AI usage, and security. Continuous trajectory capture on every session, sampled evaluation, and triggered evaluation on anomalies keep measurement affordable at thousands of sessions a day.

Cost governance in GitHub AI Credits

GitHub Copilot agent usage is metered in GitHub AI Credits. Since June 1 2026, one credit equals one US cent. That single conversion is what makes agent spend legible to finance: every prompt, tool call, and agent session resolves to a credit count, and a credit count resolves to dollars. The cost dashboard rolls those credits up per agent, per workload, and per team. Per-agent attribution is the prerequisite for the AI cost family in the previous section. Without it, agent spend hides inside an aggregate cloud bill and business value stays anecdotal.

Attribution enables control. A cost circuit breaker terminates an agent session when its credit spend crosses a threshold, the runtime complement to the budget ceiling that guardrails enforce before a session starts. Model routing is the other lever. GitHub Copilot exposes a model picker (Claude Fable 5, Opus 4.8, Sonnet 5, Haiku 4.5, Gemini 3.1 Pro, GPT-5.x), so routine agent work runs on Haiku 4.5 while high-judgment work is reserved for Opus 4.8 or Claude Fable 5. Per-model utilization on the AI usage dashboard shows whether that routing is holding, and cache hit rate shows whether repeated context is being paid for twice.

DEFINITION

Per-agent AI Credit attribution plus cost circuit breakers turn agent spend from an uncontrolled cloud line item into a governed, per-session cost. One credit is one US cent; the cost dashboard is where that arithmetic becomes a CFO conversation.

The business case: what good looks like, and how to defend it

Two field numbers anchor what good looks like. The first is golden-path adoption above 60%, the share of new services that start from a Golden Path rather than from scratch. Below that threshold the platform team operates as a service desk; above it, it becomes a force multiplier and shadow services fall away. It is tracked monthly and owned by the platform product manager. The second is a median developer experience gain of roughly 15 points in the two quarters after Golden Paths land, measured against an internal pre-and-post baseline. The larger part of that gain is in cognitive load and feedback loops, not raw speed, and it correlates with retention, not only productivity.

Those two numbers connect to outcomes a CFO recognizes. McKinsey's State of AI 2025, across more than 1,400 organizations, reports that platform-mature organizations reach attributable AI value in about 6 months against about 15 for immature peers, a 2.5x compression, at roughly half the cost per workload. Forrester's platform engineering research finds 77% of companies attributing measurable time-to-market gains to internal developer platforms and 85% reporting positive revenue impact. IDC's data shows the same split in agent deployment, with production-agent rates far higher in platform-mature organizations than in immature ones.

DEFINITION

The business case passes the defensibility test when every number is sourced or directly measurable, the pessimistic scenario still produces positive net value, and the metrics already live in the platform's dashboards. Be conservative in projection and over-deliver in execution.

References

Primary sources for the maturity model, the measurement layers, and the business case in this chapter.

- CNCF Platform Engineering Maturity Model, the five-dimension, five-level model this chapter scores against.
- DORA 2025 Accelerate State of DevOps Report, the Four Keys and the amplification finding, on nearly 5,000 survey responses.
- McKinsey, The State of AI 2025, the 2.5x ROI compression and roughly half the cost per workload for platform-mature organizations.
- Forrester Wave, Platform Engineering Solutions Q1 2026, 77% time-to-market gains and 85% positive revenue impact from internal developer platforms.
- IDC, Agentic AI Platforms and Strategies, the production-agent deployment differential between platform-mature and immature organizations.
- GitHub Copilot Documentation, agent mode, the model picker, and the GitHub AI Credits billing model this chapter measures.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-02

Building the future of software development with AI and Agentic DevOps