THREE HORIZONS, THE RED HAT PATH · PLAYBOOK

From raw Azure to an Agentic DevOps Platform in 90 to 180 days.

Three Horizons is the accelerator for enterprises that want the platform vendor-supported, certified, and jointly owned by Red Hat and Microsoft: Red Hat Developer Hub on Azure Red Hat OpenShift, OpenShift Pipelines and GitOps, Trusted Application Pipeline, and the Microsoft AI foundation underneath. This playbook walks the opinionated stack, the three-horizon sequence from secure foundation to governed agents, and the decision framework that says when this path is the right one.

AUTHOR

Paula Silva

ROLE

AI-Native Software Engineer

CONTACT

in/paulanunes

READING TIME

~ 40 minutes, end to end

3

HORIZONS

6

CHAPTERS

22

GOLDEN PATH TEMPLATES

90-180

DAYS TO FIRST PRODUCTION VALUE

CHAPTERS

Chapter 00

The case

What the accelerator is, the three-part value proposition (speed, commercial coherence, regulatory posture), the reference numbers, and the maturity ladder that names it.

The Case →

Chapter 01

The opinionated stack

RHDH as the IDP, ARO as the jointly operated cluster, Pipelines plus GitOps as delivery, Trusted Application Pipeline as the supply chain, and the three-surface AI-assist layer.

Architecture →

Chapter 02

Horizon 1, the foundation

The secure Azure runtime in 90 days: zero-trust network, no static credentials anywhere, registry and data on private endpoints, tested recovery, and cost governance from day one.

Horizon 1 →

Chapter 03

Horizon 2, the internal product

App-of-apps GitOps, Developer Hub as the front door, the 22-template Golden Path catalog, and the observability stack with more than 30 alerts across five layers.

Horizon 2 →

Chapter 04

Horizon 3, the agentic layer

Azure AI Foundry as Terraform, seven AI Golden Paths, the 17-agent fleet with MCP integrations, and the evaluation gates that make autonomy governable.

Horizon 3 →

Chapter 05

The decision and the deployment

The Four Pillars crosswalk, the joint commercial structure, the client profiles that tilt each way, and the three-phase sequence anchored by the first-value milestone.

The Decision →

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 · 2026-07-03

Building the future of software development with AI and Agentic DevOps

00 THE CASE

One accelerator, one opinionated stack: from raw Azure to an Agentic DevOps Platform.

Three Horizons is a customer-deployed accelerator that materializes an Agentic DevOps Platform on the Red Hat stack running on Azure. It exists because assembling that platform from raw components takes 9 to 18 months, and because a class of enterprises needs the result to be vendor-supported, certified, and auditable from day one. This chapter defines what the accelerator is, states its value proposition, and introduces the maturity ladder that gives it its name.

What Three Horizons is

Three Horizons is a reference architecture and a customer-deployment pattern, not a product SKU. It combines existing Red Hat and Microsoft products into an integrated, opinionated foundation: Red Hat Developer Hub as the Internal Developer Platform, Azure Red Hat OpenShift as the cluster, OpenShift Pipelines and OpenShift GitOps as the delivery substrate, Red Hat Trusted Application Pipeline for supply-chain security, and the Microsoft foundation (Entra ID, Azure AI Foundry, GitHub Enterprise) underneath. The components are independently licensed and supported; the accelerator is the composition.

Physically, the accelerator ships as a single GitHub template repository: more than 120 files and roughly 20,000 lines of curated, documented code, including 16 Terraform modules that cover every Azure infrastructure layer, 22 Golden Path templates for self-service scaffolding, 17 specialized Copilot Chat agents, 13 MCP server integrations, and more than 30 Prometheus alerting rules. The customer clones the template and customizes policy and content. The customer does not assemble integrations.

> DEFINITION
>
> The operating principle of every accelerator in this body of work: the customer customizes, the customer does not assemble. The integration work is the most expensive part of platform construction, and it is exactly the part the accelerator absorbs.

The three-part value proposition

The first component is speed. A green-field platform that would take 9 to 18 months to assemble from raw components compresses to 90 to 180 days, because the accelerator ships the integration patterns, the Golden Path catalog, the supply-chain wiring, the GitOps configuration, and the agent infrastructure pre-integrated. The second component is commercial coherence. Red Hat and Microsoft operate the underlying services as a joint strategic partnership, so the customer has one foundation deployment with two world-class commercial owners and unified support, licensing, and roadmap conversations.

The third component is regulatory and audit posture. The Red Hat stack carries FedRAMP, FIPS, and Common Criteria certifications that some sectors treat as procurement-blocking criteria, and the accelerator ships policy defaults opinionated for Latin American regimes: LGPD in Brazil, BACEN in financial services, ANEEL in energy, ANS in healthcare. The customer inherits this posture instead of negotiating it. For banks, utilities, healthcare and government, this third component is frequently the deciding one.

The impact, by the numbers

The accelerator's reference deployments report four headline effects. Deployment speed: roughly 60% faster releases, because automated Tekton pipelines and GitOps promotion cut release cycles from weeks to hours. Operational efficiency: about 40% less toil, because infrastructure as code, reconciliation, and self-service portals remove manual work from the platform team's week. Developer experience: adoption surveys report satisfaction gains around 3x once Golden Paths and Copilot-assisted workflows land. Compliance: policy-as-code brings every deploy under the same enforced standards.

Treat these numbers the way an engineering leader should: as directional evidence from reference deployments, not as guarantees. The measurable commitments that matter are the ones the platform itself emits once it is running: DORA lead time and change-failure rate for onboarded teams, per-workload cost attribution on the spending dashboards, and the first-value milestone, the week the first team ships through a Golden Path faster than its own prior baseline.

> DEFINITION
>
> Four reference numbers: 60% faster deployments, 40% less operational toil, 3x developer satisfaction, and 100% of deploys under policy-as-code. Verify them in your own estate with DORA metrics and cost attribution, both of which the platform ships by default.

The maturity ladder that names the accelerator

Three Horizons is named after the maturity model it operationalizes. Horizon 1, Optimize Present, is the credible foundation: cluster, network, identity, pipelines, and the security baseline, targeted at the first 90 days. Horizon 2, Emerging Capabilities, turns infrastructure into an internal product: the Developer Hub portal, Golden Paths at scale, observability, and governed self-service, in months three to six. Horizon 3, Transform Future, is the agentic layer: AI Foundry, RAG and multi-agent workloads, and autonomous operations with human review at promotion boundaries, in months six to twelve.

The ladder is a sequencing model: it tells the platform team what to build next and refuses to let the agentic layer land on an unfinished foundation. It is compatible with the CNCF Platform Maturity Model, which is a measurement model: Horizon 1 corresponds roughly to the Operational level, Horizon 2 to Scalable, and Horizon 3 to Optimizing or AI-Native on most dimensions. Chapters 02 through 04 walk each horizon in operational depth, and chapter 05 closes with the decision framework and the deployment sequence.

References

Primary sources for the accelerator definition, the stack, and the maturity model.

- Red Hat Developer Hub documentation, the enterprise Backstage distribution that serves as the accelerator's IDP.
- Azure Red Hat OpenShift documentation, the jointly operated managed OpenShift service the platform runs on.
- CNCF Platforms Whitepaper, the reference definition of platform capabilities the accelerator materializes.
- CNCF Platform Engineering Maturity Model, the measurement model the H1/H2/H3 ladder maps onto.
- DORA 2025, State of AI-assisted Software Development, the delivery-metrics lens used to verify the accelerator's impact claims.
- Red Hat Trusted Application Pipeline, the supply-chain security stack wired in by default.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-03

Building the future of software development with AI and Agentic DevOps

01 ARCHITECTURE

Every component named, every component integrated: the stack at a glance.

The Three Horizons stack is opinionated on purpose. Every layer has a named component, and every component arrives integrated with the others at deploy time. This chapter walks the five load-bearing choices: the Internal Developer Platform, the cluster, the delivery substrate, the supply-chain security stack, and the AI-assist layer, and explains what each choice buys the customer.

Red Hat Developer Hub as the IDP

Red Hat Developer Hub is Red Hat's enterprise-supported distribution of Backstage, and it plays the same role relative to upstream Backstage that Red Hat Enterprise Linux plays relative to upstream Linux: same core, hardened packaging, enterprise support, longer lifecycle. RHDH ships a curated set of roughly 30 to 40 tested plugins, including OpenShift-native plugins that surface Pipelines, GitOps, ACM, and ACS information directly in the catalog, plus 24x7 support with security patching and upgrade assistance. For regulated industries, that support contract is a procurement criterion, not a convenience.

RHDH's Software Templates are the Golden Path surface of the accelerator, organized by the H1/H2/H3 ladder, and its Software Catalog is the registry of every service, component, API, and resource in the enterprise, populated by automated discovery and by declarative YAML entries next to the code. For agent workloads the catalog is the discovery surface: an agent that needs to call a service queries the catalog, finds the entry, retrieves the API spec, and uses it. The catalog is the operational expression of the platform's context layer for service-level information.

Azure Red Hat OpenShift as the cluster

ARO is a managed OpenShift service jointly operated by Microsoft and Red Hat. The control plane, etcd, and API server are provisioned, patched, and monitored by the joint engineering organization under a single SLA, typically 99.95%, and a single bill from Microsoft covers the cluster. That joint operational model is one of the accelerator's differentiating properties: incidents that span the Microsoft and Red Hat boundary are resolved inside one support relationship instead of between two vendors blaming each other.

ARO integrates natively with the Microsoft foundation. Entra ID is the cluster's identity provider with RBAC governed by Entra groups; Workload Identity issues short-lived federated credentials so long-lived service secrets are unnecessary; Azure Monitor ingests cluster telemetry into the unified observability plane; Azure Policy enforces governance at the cluster boundary; and Defender for Cloud provides runtime threat detection. For data-residency sensitive workloads, ARO in Brazil South keeps the cluster, the workloads, and the foundation services fully in-region, which matters operationally under LGPD and BACEN.

> DEFINITION
>
> The cluster choice is a support-model choice. ARO means one SLA, one bill, and one escalation path across Microsoft and Red Hat. Where that single relationship is a procurement requirement, the cluster decision is already made.

OpenShift Pipelines and OpenShift GitOps

OpenShift Pipelines is Red Hat's distribution of Tekton, and its structural property is that pipelines are Kubernetes resources, not external orchestrator state: a pipeline is a YAML object in the cluster API, observable by the same stack that observes workloads. OpenShift GitOps is Red Hat's distribution of Argo CD, and it makes Git the only write surface for the cluster: state is declared in Git and continuously reconciled, so manual drift is detected and reverted, and the cluster self-heals.

The combined pattern is a loop: a developer or agent commits, Pipelines builds and validates, Pipelines updates the GitOps repository with the new image reference, GitOps

reconciles the cluster, and observability closes the loop. Sync policies are environment-specific: dev auto-syncs continuously, staging requires an explicit approval, production requires explicit sync with multiple approvers. That is manual review enforced at the deployment boundary, and every change is auditable in Git history, which satisfies the compliance posture with Git's existing primitives.

> DEFINITION
>
> Two properties do the heavy lifting: production has no direct write surface, everything flows through Git; and every change is auditable in Git history. The most common production-incident pattern, the unauthorized direct change, is structurally prevented.

Trusted Application Pipeline and the supply chain

Red Hat Trusted Application Pipeline is the supply-chain security stack integrated with Pipelines. It generates a complete SBOM at every build step, signs every artifact and attests build provenance in SLSA-compliant form, scans dependencies and produced artifacts for vulnerabilities, and enforces policy that rejects failing artifacts at build time, deploy time, and runtime through admission control. The layered pattern with GitHub is deliberate: GitHub Advanced Security at commit, TAP at build, Advanced Cluster Security at runtime.

This wiring is a direct answer to a measured regression: organizations that deployed AI coding assistants before maturing their platform security posture saw a 38% rise in exploitable vulnerabilities in the first twelve months, per CrowdStrike's 2026 Global Threat Report. TAP's four mechanisms close that gap structurally: vulnerable dependencies rejected at build, unsigned artifacts rejected at admission, provenance mismatches rejected at deploy, and SBOMs continuously verified against what is actually running.

The three-surface AI-assist layer

The accelerator integrates three AI surfaces by audience. GitHub Copilot Enterprise assists developers at the source-code layer: completion, Copilot Chat, and the Copilot coding agent for multi-session work. Red Hat Lightspeed assists operators at the platform

layer: cluster operations, Ansible playbook generation, policy explanation. Microsoft Foundry Agent Service runs the custom agents the customer builds, with Entra Agent ID for identity and Foundry observability for trajectories. Developers meet Copilot in the IDE, operators meet Lightspeed in the console, and production agents run on Foundry.

The design choice worth noticing is that the agent runtime layer does not vary by accelerator: custom agents in Three Horizons run on the Microsoft foundation, the same runtime the Open Horizons variant uses, because Red Hat does not ship a competing agent runtime as of 2026. The differentiation between the two accelerators lives at the IDP and cluster layers. The AI primitive layer, model gateway, agent runtime, evaluation, and observability, is shared foundation.

References

Component documentation for every named element of the stack.

- Red Hat Developer Hub documentation, the curated plugin set, software templates, and support lifecycle.
- Azure Red Hat OpenShift documentation, the joint operational model, SLA, and Azure integrations.
- OpenShift Pipelines documentation, the Tekton-based, Kubernetes-native CI/CD substrate.
- OpenShift GitOps documentation, the Argo CD-based reconciliation model and sync policies.
- Red Hat Trusted Application Pipeline, SBOMs, signing, attestation, and policy enforcement.
- SLSA, Supply-chain Levels for Software Artifacts, the provenance and attestation framework TAP implements.
- Red Hat Lightspeed, the operator-side AI-assist layer across the Red Hat portfolio.
- CrowdStrike, 2026 Global Threat Report, the measured security regression the supply-chain wiring addresses.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-03

Building the future of software development with AI and Agentic DevOps

02 HORIZON 1

Optimize Present: a secure, observable Azure foundation in the first 90 days.

Horizon 1 is the credible foundation. In the first 90 days the accelerator provisions a production-grade Azure environment through 16 Terraform modules: the ARO cluster, zero-trust networking, identity and secrets, registry and databases, disaster recovery, and cost governance, and onboards the first three teams through six H1 Golden Path templates. Everything above this layer assumes it is done right, which is why it comes first and why none of it is optional.

The scope of the first 90 days

Horizon 1 delivers the minimum viable platform: the ARO cluster provisioned with Workload Identity, Key Vault, and Azure Monitor integration; Red Hat Developer Hub deployed, branded, and secured behind GitHub OAuth; OpenShift GitOps with environment-specific sync policies; OpenShift Pipelines; and the initial H1 template catalog loaded. The deployment sequence runs foundation work in weeks zero to four and team onboarding from week four onward, with DORA measurements live for every onboarded team from the start.

The milestone that closes Horizon 1 is observable, not ceremonial: the first application team ships a workload through a Golden Path, and its PR-to-production lead time, measured on the new platform, beats the team's own prior baseline. That typically lands in week six to eight. Once the first team has shipped, the next three onboard with reduced friction because the deployment patterns are validated and the documentation is field-tested.

Zero-trust networking

The network module provisions a segmented virtual network: dedicated subnets for ARO master and worker nodes, private endpoints, and management traffic, with network

security groups restricting inbound and outbound flows at every subnet boundary. All PaaS services, Key Vault, the container registry, and the databases, are reached exclusively through private endpoints resolved by Azure Private DNS zones. No data service has public exposure in any configuration the accelerator ships.

The posture is deliberately preventive rather than detective: the attack surface is reduced before workloads arrive, not monitored into shape afterwards. Egress control, private resolution, and subnet isolation are Terraform-managed, which means the network's security properties are reviewable in a pull request and reproducible across environments, instead of living as console state that drifts.

Identity, secrets, and runtime protection

The security module enforces one rule everywhere: no static credentials. Every service uses a managed identity; GitHub Actions and Kubernetes workloads authenticate through Workload Identity Federation over OIDC, so no long-lived secret exists to leak; and RBAC assignments are least-privilege, scoped to resource groups. Azure Key Vault, with soft delete and purge protection enabled, is the single source of truth for the secrets that must exist at all.

Runtime protection comes from Microsoft Defender for Containers, which watches the ARO cluster for suspicious process activity, privilege escalation, and lateral movement, scans container images for vulnerabilities, and assesses the cluster against CIS benchmarks. Full RBAC integrates Azure AD with OpenShift OAuth, and Security Context Constraints are enforced at the cluster level. The identity model is the part of Horizon 1 that the agentic layer will lean on hardest later: agents get identities the same way services do.

> DEFINITION
>
> One rule, enforced by construction: no passwords, no service principal secrets in code, no long-lived credentials anywhere. Managed identity plus Workload Identity Federation is the default, and the exception list is empty.

Registry and data services

Azure Container Registry runs with geo-replication for multi-region pull latency, built-in vulnerability scanning on push, a private endpoint instead of public registry exposure, and managed-identity pull authentication from ARO, which removes registry credentials as a leak class entirely. The database module provisions PostgreSQL Flexible Server as the Developer Hub backend with automatic backups and production high availability, Redis for session management and caching, and optionally Cosmos DB behind a Terraform flag for globally distributed access.

Every one of these services is reached through private endpoints only. The practical consequence for the platform team is that data-layer compliance conversations get short: there is no public path to enumerate, and the SBOM-to-runtime verification chain from chapter 01 extends down to the images the registry admits.

Resilience, cost, and naming

Disaster recovery is explicit rather than aspirational. Velero backs up cluster state and persistent volumes to Azure Blob Storage on a configurable schedule; storage accounts and databases replicate to a secondary region with geo-redundancy; and recovery point and recovery time objectives are defined per sizing profile, from best-effort development to strict production SLAs. Recovery is a tested property of the platform, not a document.

Cost governance ships in Horizon 1 because retrofitting it later never happens. Azure budgets carry threshold alerts routed to email and Teams, tagging policies are mandatory on every resource, and spending analysis dashboards land in Grafana next to the platform metrics. All resources follow Cloud Adoption Framework naming conventions, so estates stay searchable and auditable across environments and subscriptions.

> DEFINITION
>
> Six H1 Golden Path templates onboard the first teams: basic-cicd, documentation-site, infrastructure-provisioning, new-microservice, security-baseline, and web-application. Each ships with a template.yaml, skeleton code, and documentation.

References

Component documentation for the Horizon 1 foundation.

- Azure Red Hat OpenShift documentation, cluster architecture, private clusters, and MachineSet autoscaling.
- Azure Key Vault documentation, soft delete, purge protection, and RBAC-scoped secret access.
- Microsoft Entra Workload ID documentation, workload identity federation over OIDC for pipelines and workloads.
- Microsoft Defender for Containers, runtime threat detection and CIS benchmark assessment for the cluster.
- Velero documentation, cluster state and persistent volume backup and restore.
- Cloud Adoption Framework naming conventions, the resource naming standard applied by the CAF naming module.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-03

Building the future of software development with AI and Agentic DevOps

03 HORIZON 2

Emerging Capabilities: infrastructure becomes an internal product.

Horizon 2 is where raw infrastructure turns into something developers actually want to use. In months three to six the platform gains its GitOps backbone at scale, its front door in Developer Hub, its full 22-template Golden Path catalog, and the observability stack that makes operations evidence-based. The exit criterion is adoption: at least half of eligible workloads running through the platform, a populated catalog, and TechDocs in organization-wide use.

GitOps at scale: app-of-apps and sync policies

The accelerator configures Argo CD with the app-of-apps pattern: a root application manages all child applications, so the entire platform stack rolls out in a declared order through sync waves. Environment policies encode the review model in configuration rather than in process documents. Dev auto-syncs, so changes deploy immediately; staging is semi-automatic, syncing on approval; production is manual, requiring explicit human approval. Access control runs through SSO with Azure AD via Dex, with platform-admin and developer roles enforced by RBAC.

At Horizon 2 scale the discipline that matters is repository structure. Multi-tenant estates produce GitOps repositories with hundreds or thousands of applications, and the application-to-repository relationship becomes the real operational surface. The accelerator ships opinionated repository conventions for exactly this reason: the pattern survives scale only when the structure is decided before the fiftieth application arrives, not after.

Developer Hub as the front door

Developer Hub is the unified entry point for every developer: one place to discover services, scaffold new applications, browse documentation, and check system health. The

Software Catalog is the centralized registry of every service, API, and component; TechDocs renders version-controlled documentation next to each catalog entity; and GitHub integration provides OAuth authentication plus GitHub App access to repositories and workflows.

Horizon 2 sets explicit adoption targets for the portal because a portal without adoption is furniture. The Service Catalog is expected to reach at least 80% coverage of real services, and TechDocs is expected to be in organization-wide use. Both are measurable from the catalog itself, which keeps the conversation honest: either the estate is discoverable through the portal or it is not.

> DEFINITION
>
> The portal test is coverage, not existence. A catalog at 80% coverage changes how agents and humans discover services. A catalog at 20% coverage teaches everyone to bypass the portal, and the platform loses its front door.

The 22-template Golden Path catalog

Horizon 2 releases the full Golden Path catalog: 22 templates across the three horizons, each shipping a template.yaml, skeleton code, and documentation. The nine H2 templates cover the middle of the estate: api-gateway with rate limiting and authentication policies, api-microservice with OpenAPI and validation, batch-job as a CronJob with retries and alerting, data-pipeline wired to Event Hub, event-driven-microservice with Service Bus or Event Hubs integration, gitops-deployment for rapid onboarding, reusable-workflows as a shared Actions library, ado-to-github-migration for guided platform moves, and a general-purpose microservice template.

The catalog's function is standardization through convenience rather than through mandate. When the sanctioned path is also the fastest path, teams take it voluntarily, and every service scaffolded through a template arrives with CI/CD, security baseline, observability, and catalog registration already wired. The Horizon 2 exit target is platform adoption at 50% or more of eligible workloads, measured, like everything else here, from the platform's own data.

The observability stack

The stack is Prometheus, Grafana, Alertmanager, and Jaeger, deployed and configured by the accelerator. Prometheus runs two replicas with 15-day retention and 50GB of storage; Grafana authenticates through Azure AD SSO and ships three curated dashboards: Platform Overview as the platform team's daily command center, Cost Management with per-team and per-namespace attribution, and Golden Path Application with the four golden signals for every templated service. Alertmanager routes to PagerDuty and Microsoft Teams with silence and inhibit rules, and Jaeger traces requests across services.

More than 30 alerting rules ship by default, covering five layers: infrastructure alerts for node pressure, disk, network, and ARO component health; application alerts for error-rate SLOs, latency thresholds, crash loops, and pending deployments; AI workload alerts for token quota, inference latency, model endpoint availability, and evaluation failures; GitOps alerts for sync failures and out-of-sync applications; and security alerts for Defender findings, policy violations, unusual access, and certificate expiry. The AI alert group exists in Horizon 2 on purpose: the platform is instrumented for agents before the agents arrive.

Secrets, runners, and ingress

Three operational services complete the horizon. The External Secrets Operator continuously synchronizes secrets from Azure Key Vault into Kubernetes Secrets through a ClusterSecretStore, with automatic rotation and zero secrets stored in Git or environment variables. GitHub self-hosted runners on ARO give CI/CD workflows access to private resources, ACR, Key Vault, the cluster itself, without any public exposure; the runners autoscale on queue depth, authenticate through managed identity, and are ephemeral, so every job starts from clean state.

Ingress and TLS are handled by NGINX Ingress Controller with path and host routing, rate limiting, and WAF capability; cert-manager provisions and renews certificates from Let's Encrypt with no manual certificate management; and External DNS creates and updates Azure DNS records as ingress resources appear and change. None of these three services is glamorous, and all three are the difference between a platform that operates itself and one that pages its team.

DEFINITION

The secrets flow in one line: Azure Key Vault is the source of truth, the External Secrets Operator syncs and rotates, pods consume Kubernetes Secrets at runtime. Nothing sensitive ever lives in Git, in environment variables, or in a pipeline definition.

References

Component documentation for the Horizon 2 platform services.

- Argo CD documentation, the app-of-apps pattern, sync waves, and sync policies.
- Backstage documentation, the catalog, TechDocs, and scaffolder foundations under Developer Hub.
- Prometheus documentation, metrics collection and the alerting rule model.
- Grafana documentation, dashboards and Azure AD single sign-on.
- External Secrets Operator documentation, the ClusterSecretStore pattern and secret rotation.
- cert-manager documentation, automated TLS certificate provisioning and renewal.
- Jaeger documentation, distributed tracing across the platform's services.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-03

Building the future of software development with AI and Agentic DevOps

04 HORIZON 3

Transform Future: the agentic layer lands on a finished platform.

Horizon 3 is the reason the other two horizons exist. In months six to twelve the platform gains its AI infrastructure as code, its seven AI Golden Paths, its fleet of 17 specialized Copilot Chat agents, and the evaluation and observability machinery that makes autonomy governable. The exit criterion is one real business process running end to end through an agent, with human review at promotion boundaries and every trajectory, token, and cost accounted for.

Azure AI Foundry, provisioned as code

The AI Foundry module provisions the complete AI infrastructure as Terraform: the Foundry hub, projects, model deployments, and Azure AI Search, reproducible and version-controlled like every other layer of the platform. Reference model deployments include GPT-4o for multimodal work, o3 for harder reasoning, and next-generation previews as they reach the subscription, all exposed through managed online endpoints for inference and through the Foundry gateway for routing, metering, and policy.

The structural property worth internalizing is that AI infrastructure stops being special. Model deployments live in the same pull-request flow as network rules; the vector store is provisioned by the same pipeline as the database; and rollbacks work the same way everywhere. Teams request AI capability through Golden Paths, not through tickets, and the platform meters what they consume per workload from day one.

The seven AI Golden Paths

Horizon 3 ships seven AI-focused templates. rag-application scaffolds a complete retrieval-augmented generation app: document ingestion, vector embeddings in Azure AI Search, and a chat interface, with observability built in. multi-agent-system provides an orchestrator pattern with specialist sub-agents, MCP tool integrations, handoff protocols, and shared

state. mlops-pipeline covers the full ML lifecycle from training through evaluation, registry, deployment, and drift monitoring. foundry-agent scaffolds single-agent applications on Foundry Agent Service with identity and evaluation wired.

The remaining three close the loop around quality and operations: ai-evaluation-pipeline turns model evaluation into a reusable pipeline, copilot-extension scaffolds custom GitHub Copilot extensions backed by Azure AI services, and sre-agent-integration wires AI-assisted incident triage, runbook execution, and root-cause analysis into the platform's operational flow. Together the seven templates mean an application team can stand up a governed AI workload the same afternoon it decides to build one, and every one of those workloads arrives observable and evaluable by construction.

> DEFINITION
>
> Seven templates, one property: an AI workload scaffolded through a Golden Path arrives with identity, observability, evaluation, and cost metering already attached. The team writes the domain logic. The platform supplies everything that makes it production-grade.

The 17-agent fleet and MCP

The accelerator ships 17 specialized Copilot Chat agents that operate directly in VS Code, covering development, deployment, operations, and incident management workflows, backed by 16 operational skills and 13 MCP server integrations that expose platform capabilities as tools. Each agent has a defined role, explicit tool access, and three-tier behavioral boundaries: actions it ALWAYS takes, actions where it must ASK FIRST, and actions it NEVER takes. The boundary model is the governance primitive: autonomy is granted per action class, not per agent.

The fleet is also how the platform team eats its own cooking. The first agents adopted in a Three Horizons deployment are typically platform-internal: incident triage, runbook authoring, policy review, drift remediation. That internal adoption, at least one recurring platform task absorbed by an agent, is half of the maturity test this body of work calls the Dual Mandate, and it is expected to pass by roughly week six of the deployment.

Evaluation gates and agent observability

Evaluation is a deploy gate, not a report. The evaluation pipeline scores model outputs for relevance, coherence, and groundedness against reference datasets, screens for harmful, biased, or policy-violating content, and feeds the results into CI/CD as gates: a failed evaluation blocks the deployment automatically, the same way a failed test does. Promotion of an agent to a wider blast radius follows the same manual-review boundaries as any production change.

Observability extends to everything the agents do: traces of agent trajectories, per-request and per-workload cost, output sampling for quality review, and evaluation results over time, alongside the AI workload alerts from Horizon 2. The AI Usage dashboard becomes a primary KPI surface for the platform. The Horizon 3 exit is concrete: at least one real business process runs end to end through an agent, with human review at promotion boundaries, at 90 to 180 days from the start of the deployment.

> DEFINITION
>
> The governance stance in one sentence: agents are first-class workloads with identity, budget, evaluation gates, and audit, and autonomy is expanded only as the evidence accumulates. That is what makes the 100-to-1 agent-to-human future operable at all.

References

Component documentation for the Horizon 3 agentic layer.

- Azure AI Foundry documentation, the hub, projects, model deployments, and gateway.
- Azure AI Search documentation, the vector retrieval layer under the RAG Golden Path.
- Microsoft Agent Framework 1.0, the framework custom agents build on.
- Model Context Protocol, the tool integration standard behind the 13 MCP servers.
- GitHub Copilot documentation, Copilot Chat, custom agents, and extensions.

- OpenTelemetry GenAI semantic conventions, the telemetry standard for agent trajectories and cost.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-03

Building the future of software development with AI and Agentic DevOps

05 THE DECISION

Choose it for the right reasons, deploy it in the right order.

Three Horizons is not the right answer for every client, and pretending otherwise wastes everyone's year. This closing chapter gives the decision its structure: what the accelerator materializes in governance terms, what it costs commercially and what the joint support buys, which client profiles tilt toward it and away from it, and the three-phase sequence that takes a signed decision to a measured first-value milestone.

What the pillars crosswalk buys

In governance terms, the accelerator materializes the four CNCF pillars with named components. Golden Paths are RHDH Software Templates with generated Tekton CI/CD, Argo CD deployment, and Terraform infrastructure. Guardrails are Advanced Cluster Security policies, admission controllers, Azure Policy at the cloud and cluster boundary, Workload Identity in place of long-lived credentials, and TAP signing at admission. Safety Nets are GitOps reconciliation, the unified observability plane, Velero backup, progressive delivery through the service mesh, and ACS runtime detection. Manual Review is environment-gated sync plus governed agent promotion.

The differentiating property is that these pillars arrive pre-wired at deploy time. The customer does not start from a blank cluster and integrate each pillar; the customer starts from an integrated foundation and customizes policy and content. That absorbed integration work is the structural explanation for the compression from 9 to 18 months down to 90 to 180 days: the slowest part of platform construction is exactly the part the accelerator ships finished.

The commercial structure and joint support

Three commercial line items compose the accelerator. Red Hat subscriptions cover RHDH, ARO, Pipelines, GitOps, Service Mesh, ACM, ACS, TAP, and Lightspeed where licensed.

Azure consumption covers cluster compute and storage plus the foundation services: Key Vault, AI Foundry, Monitor, Entra. GitHub Enterprise covers per-seat licensing with Advanced Security and Copilot Enterprise. Comparing against the open-source variant of this architecture, the Red Hat subscription line is the primary cost differentiator, and the support contract is exactly what that line buys.

The joint support model is the strongest commercial reason clients choose this path. Microsoft and Red Hat operate ARO jointly, with escalation paths between them, so an incident that spans the boundary is engaged through one party and resolved internally, instead of stalling while two vendors blame each other. For some procurement organizations that single relationship is a non-negotiable requirement, and where it is, the decision is effectively made before any technical comparison starts.

> DEFINITION
>
> Read the cost structure as a trade, not a penalty: the Red Hat subscription line buys the joint support contract, the certification posture, and the enterprise lifecycle. Clients who do not need those three things should not pay for them, and chapter logic says so explicitly.

Client profiles: toward, away, and undecided

The profile that tilts toward Three Horizons has at least three of these properties: prior Red Hat investment at material scale, RHEL, OpenShift, or Ansible; regulatory or audit requirements where FedRAMP, FIPS, or Common Criteria certifications are procurement-blocking; a vendor strategy that prefers one strong joint relationship over many loose ones; and concentration in financial services, energy, government, or healthcare. The reference deployments behind this playbook sit in exactly those sectors, and the policy library ships opinionated for LGPD, BACEN, ANEEL, and ANS because of it.

The profile that tilts away is just as clear: no prior Red Hat footprint, a vendor strategy that minimizes vendor count, strong internal upstream Kubernetes expertise, or an OSS-first engineering culture that values transparency and pace over vendor support. Those clients are better served by the open-source Open Horizons variant on AKS and upstream Backstage. When leadership is split or both engineering cultures coexist, run a one-day

comparative discovery instead of arguing: score the estate against these profile properties and let the evidence recommend.

The three-phase deployment sequence

Phase one, weeks zero to four, is foundation: the ARO cluster provisioned with Workload Identity, Key Vault, and Monitor integration; RHDH deployed, branded, and secured behind GitHub OAuth; GitOps with environment-specific sync; Pipelines; and the H1 template catalog loaded. Phase two, weeks four to twelve, is onboarding: the first three application teams arrive through H1 templates, the Service Catalog populates, TechDocs is adopted, ACS policies apply, TAP integrates with Pipelines, and DORA measurement goes live for every onboarded team.

Phase three, months three to twelve, is scaling: H2 templates release and adopt, the cost dashboard goes live, the first Copilot Chat agent deploys, catalog coverage crosses 80%, multi-cluster expands through ACM where applicable, and H3 templates reach early adopters. The milestone that anchors the whole sequence lands in week six to eight: the first team ships through a Golden Path with a lead time faster than its own prior baseline. That number, not a demo, is what expansion decisions should be based on.

> DEFINITION
>
> The first-value milestone is deliberately unimpressive: one team, one workload, one lead-time number better than its own baseline. Everything that scales afterwards is built on that verified fact, which is why the sequence refuses to skip it.

Maturity mapping and the standing test

The H1/H2/H3 ladder is a sequencing model; the CNCF Platform Maturity Model is a measurement model, and the two compose. Horizon 1 corresponds roughly to CNCF Operational on most dimensions, Horizon 2 to Scalable, and Horizon 3 to Optimizing or AI-Native. Used together, the ladder tells the platform team what to build next and the CNCF model tells them how mature the result actually is, which keeps roadmap conversations anchored to assessed state rather than to enthusiasm.

The standing test of a healthy deployment is the Dual Mandate, and in Three Horizons it is tractable by roughly week six: at least one of the platform team's own recurring tasks has been absorbed by an agent, and at least one application team is consuming the model gateway through a Golden Path. When both halves hold, the platform is simultaneously eating its own cooking and serving its customers, and the agentic layer that follows lands on demonstrated capability instead of on a slide.

References

Primary sources for the decision framework, the commercial model, and the maturity mapping.

- Azure Red Hat OpenShift documentation, the joint operation, SLA, and support model the commercial argument rests on.
- CNCF Platform Engineering Maturity Model, the five-dimension measurement model the horizons map onto.
- CNCF Platforms Whitepaper, the pillar definitions behind the crosswalk.
- Red Hat Advanced Cluster Security, the runtime security component of the Guardrails and Safety Nets rows.
- Red Hat Advanced Cluster Manager, multi-cluster policy and governance for the scaling phase.
- DORA 2025 research program, the delivery metrics that anchor the first-value milestone and the expansion decision.

Paula Silva

AI-NATIVE SOFTWARE ENGINEER

in/paulanunes

v1.0.0 · 2026-07-03

Building the future of software development with AI and Agentic DevOps