Deloitte. | aws



INDUSTRIALIZING ENTERPRISE AI APPLICATION DELIVERY

A single Claude Code plug-in that turns a YAML configuration into deployed, observable, secure analyst application–frontend, backend, agents, data, infrastructure–with enterprise governance enforced by the tools.

10X-Rapids is part of Deloitte's RAPID framework for fast software development lifecycle (SDLC), which accelerates secure, governed software delivery.

Authored by **10X-Analyst platform team**

THE INNOVATION BOTTLENECK—AT ENTERPRISE SCALE

How a Claude Code-native, plugin-driven development factory turns months of platform engineering into hours—and why this changes the unit economics of building enterprise AI applications at scale.

Every enterprise has a backlog of analyst-grade artificial intelligence (AI) use cases—claims triage, municipal-bond analysis, clinical-trial review, equity research, cyber investigation, underwriting decision support—that stall before reaching production. The reason is not merit; it is the cost of operationalizing the second, fifth, and fiftieth use case.

The first prototype is built heroically; the second is built differently, by a different team, with different observability, security posture, and deployment topology. Within three or four domains, the platform is no longer a platform—it is a handful of slightly-related codebases nobody can maintain.

The 10X-Analyst platform was built to break that pattern: one repository, one templated lifecycle, one set of opinionated infrastructure modules, and one declarative YAML schema per business domain.

The 10X-Rapids Claude Code plugin is what makes the vision operationally real—encoding the team's hard-won opinions about how to build, test, secure, observe, and ship a 10X-Analyst app into 70+ executable skills, 15 specialist AI agents, and 10 enforceable safety hooks that any developer invokes with a single slash command inside Claude Code on Amazon Bedrock.

THE STACK: CLAUDE CODE ON AMAZON BEDROCK, SUPERCHARGED BY 10X-RAPIDS

Claude Code on Amazon Bedrock

Claude Code is Anthropic's agentic coding tool—a command line interface (CLI) and integrated development environment (IDE)-integrated AI assistant that operates directly inside the developer's environment. It reads project files, writes code, executes commands, runs builds, reads compile errors, fixes them, and iterates autonomously, while a human stays in the loop at every decision gate.

Running Claude Code through Amazon Bedrock is what makes this acceptable for regulated, enterprise-grade work:

CONCERN	*WHAT AMAZON BEDROCK CAN DELIVER*
Data sovereignty	Prompts, code, and proprietary specs never leave the AWS environment; processed inside a virtual private cloud (VPC), governed by AWS Identity and Access Management (IAM), encrypted with AWS Key Management Service (KMS).
Access control	bedrock:InvokeModel scoped per-developer, per-model Amazon Resource Name (ARN), through existing IAM identity.
Cost and observability	Token usage, latency, and cost streamed into CloudWatch alongside other AWS workloads.
Prompt caching	Reduces token cost and latency for agentic sessions that reference the same large codebase repeatedly.
Guardrails	Service quotas and throttling prevent runaway spend.
Deployment portability	Equivalent enterprise controls—System and Organization Controls (SOC) 2, audit logging, System for Cross-domain Identity Management (SCIM) — are available on the Anthropic application programming interface (API) direct path through Claude Enterprise governance.

THE STACK: CLAUDE CODE ON AMAZON BEDROCK, SUPERCHARGED BY 10X-RAPIDS

The 10X-Rapids plugin itself is transport-agnostic. Its skills, specialist agents, and safety hooks operate identically whether Claude Code is invoking the Anthropic API direct or Amazon Bedrock. Client engagements select the transport based on cloud posture, data residency, and procurement vehicle; the brief that follows details the Amazon Bedrock pattern because it is the most frequently requested by enterprise clients standardized on AWS. A developer goes from zero to prototyping by setting CLAUDE_CODE_USE_BEDROCK=1 and an AWS region—full enterprise governance is "on" from day one. A general-purpose AI coding agent is powerful but unopinionated.

THREE-LAYER ENTERPRISE STACK

Each layer adds a new repeatable enterprise property

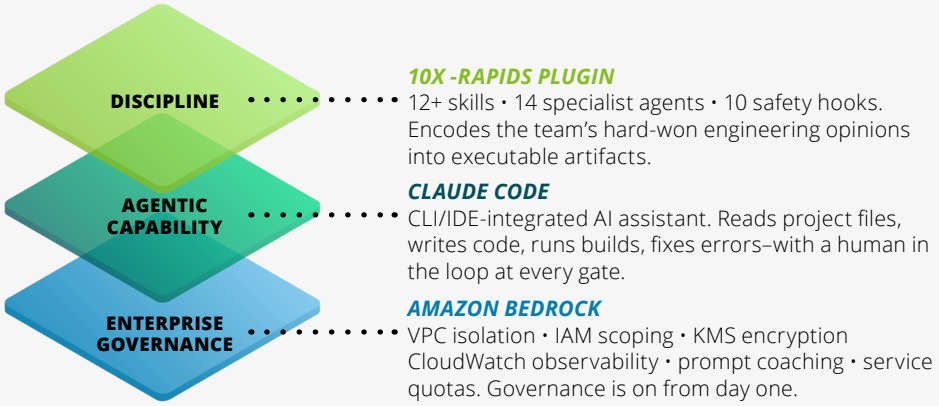


Figure 1. Stack overview
Claude Code agentic capability + Amazon Bedrock governance + 10X-Rapids discipline.

THE 10X-RAPIDS PLUGIN: TURNING A GENERAL AGENT INTO A COMPLETE ENGINEERING TEAM

A regulated enterprise platform cannot afford to be unopinionated. The 10X-Rapids plugin—installed inside Claude Code via the team marketplace (/plugin install 10X-Rapids@10X-marketplace)—is what closes that gap.

What it gives you, out of the box:

CAPABILITY LAYER	CONTENT	VALUE
Skills	70+ executable recipes (/create-app, /create-agent-service, /create-lambda-server, /create-terraform-module, /create-github-workflow, /iam-policy-audit, /generate-runbook, /generate-kt-doc, ...).	Every scaffolding task that used to require tribal knowledge is now a single slash command that produces consistent output.
Specialist agents	15 role-based AI personas: agent-architect, backend-developer, frontend-developer, infra-engineer, data-engineer, langgraph-expert, suitability-architect, full-stack-team, agent-dev-team, review-team, code-reviewer, logic-reviewer, performance-engineer, code-hygiene-auditor, builder.	Multi-step work is delegated to AI specialists that already understand the codebase's patterns—and to a full-stack-team agent that coordinates them in parallel like a tech lead.
Safety hooks	10 shell-based gates wired into Claude Code's PreToolUse/PostToolUse/PreCompact events.	Organizational policy becomes executable—the agent cannot bypass it, even by mistake.
Plugin manifest	A single plugin.json registers the plugin in the 10X-marketplace for one-command install across the team.	New developers are onboarded in seconds, not days.

WHY THIS COMBINATION IS AN ENTERPRISE ARCHITECTURE DECISION, NOT A TOOLING CHOICE

Three things make this stack uniquely fit for Deloitte's 10X-Analyst program:

1. Patterns become enforceable, not aspirational.

Every agent service scaffolded by /create-agent-service ships with mandatory structured JavaScript Object Notation (JSON) logging, a ContextAwareLogFormatter that injects trace IDs and request context into every JSON log line, embedded metric format (EMF) metrics for CloudWatch.

AWS Secrets Manager–resolved configuration, and a non-root container user. A developer cannot forget to add observability—the scaffold ships with it, and the skill's hard rules block the alternative.

2. Policy becomes code.

The plugin ships with 10 shell-based safety hooks wired into Claude Code's PreToolUse, PostToolUse, and PreCompact events. They fire automatically before a file is written or a command is run, and block non-compliant operations—secrets exposure, destructive structured query language (SQL), Terraform applies without a reviewed plan, oversized writes, missing test coverage—before they execute.

The agent cannot route around them, even by mistake. This is what turns organizational policy into executable code.

POLICY-AS-CODE ENFORCEMENT

Shell-based hooks block non-compliant operations before they execute



Figure 2. Policy enforcement—shell-based pre/post-tool hooks that block non-compliant operations before they execute.

10 HOOKS SHIPPED WITH THE PLUGIN
secrets-blocker · terraform-plan-check · pre-migration · large-file-guard · api-breaking-change · dependency-audit · test-coverage-gate · verify-after-edit · commit-format · compaction-guard

3. Specialist agents collapse multi-day team coordination into single-session sprints.

The full-stack-team agent is a delegating tech lead: it breaks a feature into backend, frontend, and infrastructure tasks, spawns backend-developer, frontend-developer, and infra-engineer agents in parallel against a shared API contract, then hands off to review-team (which itself coordinates code-reviewer, logic-reviewer, performance-engineer, and code-hygiene-auditor) for review.

SPECIALIST AGENTS, COORDINATED LIKE A TEAM

Tech-lead persona delegates in parallel, then hands off to a review pipeline

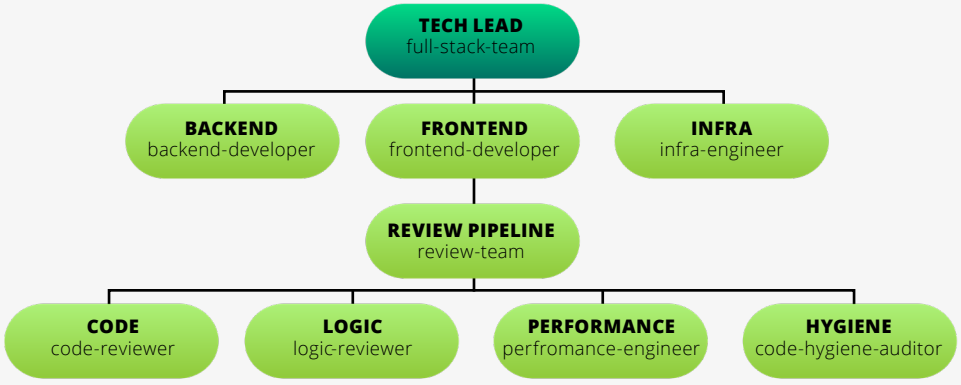


Figure 3. Agent orchestration—a lead persona delegating in parallel to specialist agents that already understand codebase.

ALSO AVAILABLE: SPECIALIST AGENTS LOADED ON DEMAND
agent-architect · langgraph-expert · suitability-architect · data-engineer · agent-dev-team · builder

CASE STUDY: BUILDING A REGULATED SUITABILITY REVIEW WORKBENCH

The problem statement

In regulated insurance, every new annuity-style contract sale must clear a suitability review before issuance. An analyst reads multiple application forms, applies dozens of decision rules covering customer identification, financial position, risk profile, product fit, and state-specific regulations, and produces an audit trail strict enough to survive regulatory examination. Until this implementation, the work was manual per case, and any platform integration to automate it was additional work per engineering team.

The application that delivers it—the suitability review workbench—is the one of the most demanding implementation on the 10X-Analyst platform so far. Multiple evaluation phases grade case data against N rules knowledge base; N rules pre-check layer screens out hard-stop conditions before any large language model (LLM) spend; multiple document types flow through ingestion; and every verdict is traceable back to the rule that produced it.

The knowledge base itself is a structured rules workbook, authored by the domain team and ingested into Amazon Bedrock Knowledge Base (KB)—so updating a rule is a workbook edit followed by a re-ingestion, never a code change.

What follows is the build narrative: the components produced, the plugin skills that produced them, and the specialist agents that participated at every step.

COMPONENT	***PLUGIN SKILLS THAT PRODUCE IT***	***SPECIALIST AGENT IN THE LOOP***
Workbench application shell	/create-app, /configure-app	frontend-developer
Seven-phase [JC1.1] evaluators	/plan-suitability-phase, / create-phase-agent	suitability-architect
LangGraph orchestrator	/create-langgraph-agent, / create-state-graph	langgraph-expert
Synthesis agent	/create-synthesis-agent	agent-architect
Pre-check validation server	/create-precheck-rules, / create-lambda-server	backend-developer
Analyst workbench screens	/create-human-review-screen, /create-screen	frontend-developer
Data layer and ingestion	/design-dynamodb-table, / create-dynamodb-schema, / create-airflow-dag, /create-ingestion-pipeline	data-engineer
API Gateway and infrastructure	/create-terraform-module, / iam-policy-audit	infra-engineer
Output verifier	/evaluate-run-output	suitability-architect

CASE STUDY: BUILDING A REGULATED SUITABILITY REVIEW WORKBENCH

Where the build starts

The session does not start with code. It starts with a conversation. The suitability-architect agent already encodes the N phases, the rule taxonomy, the FOLLOWUP-vs-HARDSTOP decision logic, and the pre-condition matrix for low-touch approval. Pointing the agent at the suitability KB workbook confirms rule counts, phase boundaries, and dependency order. From that point on, every architectural decision is made with a domain specialist in the loop—not as a documentation step, but as a participant in the build. /plan-agent-architecture brings in the agent-architect specialist.

The rules are grouped into N evaluation phases—each a coherent slice of the KB. The resulting design has few phases running in parallel, few sequential because they share intermediate state. The rationale is committed to a Knowledge Exchange (KX) document via /generate-kx-doc so it lives next to the code it describes.

The workbench shell

The analyst workbench is what humans see and interact with—a Next.js application where analysts review AI-flagged cases, walk through the rule-by-rule reasoning, and either approve a case, request additional documentation, or escalate it. It needs three core tabs: AI Review (the queue of cases the AI has processed), Human Validation (the per-case review surface), and Investigate (the audit drill-down). Behind the scenes, the entire shell is produced from declarative YAML—tab definitions, palette, sidebar features—by two plugin skills: /create-app writes the application's registry files, and /configure-app materializes the React surface against them. The shell opens in a browser before a single custom screen exists, confirming that the platform contract holds for one more domain before any custom code is written.

The phase evaluators

The heart of the suitability workflow is N phase agents, each grading a slice of the KB against a single case. The build pattern repeats: /plan-suitability-phase reads the KB with help from the suitability-architect and produces a phase specification in markdown. /create-phase-agent emits the corresponding Python module under py/agents/ai_review_agent/agents/, contributing to phased_suitability_check.py —the single dispatch surface that loads each phase's prompt and rules and produces structured per-rule verdicts.

Each new phase is reviewed against the KB by the suitability-architect before the next one is written, so drift is caught at write time, not at test time.



CASE STUDY: BUILDING A REGULATED SUITABILITY REVIEW WORKBENCH

Orchestration, synthesis, pre-checks

/create-langgraph-agent and /create-state-graph generate orchestrator_agent.py — the LangGraph workflow that wires the phase nodes into the parallel/sequential topology designed earlier. The langgraph-expert agent reviews the graph for race conditions and state-merge correctness.

/create-synthesis-agent produces synthesis_agent.py, the aggregator that turns seven-phase outputs into one of two case decisions—APPROVED or FOLLOWUP—and the analyst-facing markdown summary.

Not every rejection requires an LLM. /create-precheck-rules reads the pre-check rules workbook and emits fourteen deterministic Python rules into the precheck validation server. Cases that fail a pre-check never spend an LLM token.

The analyst workbench surface

/create-human-review-screen builds the case-list, the run-status view, and the per-case review surface, including the SuitabilityReviewDialog component. /create-screen adds the investigation drill-down. The frontend-developer agent reviews each screen for conformance with the shadcn/ui design tokens already established in the platform—the result feels native, not a one-off.

Data, ingestion, infrastructure

/design-dynamodb-table and /create-dynamodb-schema produce the single-table design for cases, run history, and audit events. /create-airflow-dag together with /create-ingestion-pipeline build the document-ingestion path that feeds Bedrock Knowledge Base. /create-terraform-module provisions the suitability-initiate and suitability-review API Gateway modules. /iam-policy-audit scans the generated IAM and confirms no wildcard resources and no inline secrets before anything is applied.

The verifier loop

The final asset is the only one that grades the application's own output. /evaluate-run-output compares an output.json from a real run against the canonical KB. A structural pass canonicalizes rule IDs to absorb naming drift. A deterministic consistency pass flags cross-field contradictions in the run. A semantic pass calls Claude Sonnet on Bedrock to grade each rule's reasoning against the KB's prescribed checks. The output is an analyst-readable Excel report—Executive Summary on top, prioritized findings, and a full rule-level audit underneath. This is the skill the team uses to keep the suitability agents honest as the KB evolves.

What makes the difference

Every component above is traceable in two directions: to a row in the KB workbook on one side, and to a plugin skill on the other. The suitability-architect agent participates in every domain-sensitive step. The agent-architect, langgraph-expert, backend-developer, frontend-developer, data-engineer, and infra-engineer agents participate in every cross-cutting step. No file is hand-written from scratch. No design decision is hidden in a slide deck. Every piece can be re-emitted from the same skill against the latest KB.

THE PROTOTYPE-TO-PRODUCTION BOUNDARY

It is important to be honest with the business about what this stack does and does not do. The 10X-Rapids plugin produces production-grade scaffolds—not finished products. The scaffold ships with the cross-cutting concerns that are universally correct (observability, structured logging, secrets management, IAM scaffolds, CI/CD, tests, Dockerfiles), and the business logic remains the engineer's responsibility. The economics of the work shift accordingly: the boring, error-prone, replicable 80% is done by the scaffold; the high-judgment, domain-specific 20% is where the engineer focuses.

COMPONENT	*WHAT THE PLUGIN ALREADY PROVIDES*	*WHAT THE HUMAN STILL OWNS*
Observability	Structured JSON logging with trace and request context (request_id, session_id, user_id, project_id, app_id, app_name), CloudWatch metrics, and a non-root container user — applied identically across every generated service.	Alarms, dashboards, on-call rotation, business-specific SLOs.
Security	secrets-blocker hook, /iam-policy-audit, /fix-hardcoded-secrets-tf, Secrets Manager wiring, non-root container user, IAM least-privilege scaffolds.	Threat model, pen-test, data classification, compliance attestation (SOC 2/ HIPAA/GDPR/FINRA/state insurance regulations).
Data layer	DynamoDB schema scaffolds, ingestion DAG templates, S3 namespacing under s3://10X-analyst/<org>/<env>/<app>/, Pydantic schemas, Alembic migration skeletons. analyst/<org>/<env>/<app>/, Pydantic schemas, Alembic migration skeletons.	Real-world schema evolution, backup/recovery strategy, data-quality contracts.
Testing	pytest scaffolds + pytest-asyncio + pytest-cov, test-coverage-gate hook.	Domain-specific edge cases, load tests, chaos tests.
Architecture	Reference patterns from powerchat_server and orchestrator_server, LangGraph state machines, model-per-node sizing.	Trade-off decisions, capacity planning, multi-region strategy.
Business logic	None—that's by design.	All of it.



FROM SCAFFOLD TO PRODUCTION

A scaffold is the start of the engineering journey, not the end. Production-grade applications still need real data contracts, security review, business logic, and operational ownership. The same Claude Code Amazon Bedrock 10X-Rapids pattern that produced the scaffold powers every subsequent stage of the lifecycle.

What one plugin install delivers

70+ executable skills across agent engineering, backend, frontend, data, infrastructure, security, code quality, debugging, documentation, and the suitability domain

15 specialist AI agents, each with curated tool access and codebase knowledge

10 policy-as-code safety hooks wired into laude Code's pre/post-tool events

4 analyst applications already running on the platform

10 backend services following identical production patterns

Why this is a stakeholder conversation tool

A scaffolded application is more than a deliverable—it is a way to make abstract architecture decisions tangible. Decision-makers can react to something real rather than something imagined. "What would the Claims Triage analyst experience look like?" becomes a click-through demo with the configured palette and real data shapes. "Can the same platform handle Muni Bonds, Equity Research, Rate Filing, Clinical Trial, Cyber Security?" becomes a directory switch—same UI shell, different YAML—proving the platform thesis live. "How quickly can we onboard a new business domain?" becomes "Watch us scaffold one now."

The economics shift

When the marginal cost of industrializing a new business domain approaches the cost of writing its YAML configuration, the conversation changes. Teams stop asking "which domain should we pick first?" and start asking "why aren't we standing up all of them in parallel?" Time to first scaffolded, observable application compresses from weeks to a single working session. Onboarding a new engineer compresses from days of pair-programming to a plugin install and a README. Consistency of observability and security across applications goes from best-effort to enforced. The marginal cost of the fifth domain looks like the marginal cost of the first.

FROM SCAFFOLD TO PRODUCTION

The phased lifecycle

Every application on the platform follows the same eight-stage path. The plugin contributes at each gate; humans own the judgment calls.

Compliance enforcement is continuous across all stages—the plugin's policy-as-code safety hooks fire on every tool invocation, and every scaffold ships with mandated security posture by default. Stage 4 formalizes the human attestation, risk sign-off, and regulatory approval for the accumulated compliance posture.



STAGE	*WHAT HAPPENS*	*PLUGIN'S ROLE*	*HUMAN'S ROLE*
1. Architect	Pick template, features, runtime, DB.	agent-architect agent; trade-offs and cost/latency estimates.	Approves plan; adjusts for domain constraints.
2. Scaffold	Generate UI, agent, Lambda, Terraform, CI/CD.	/create-app, /create-agent-service, /create-lambda-server, /create-terraform-module, /create-github-workflow.	Reviews every diff at approval gates.
3. Connect data	Replace static JSON with real APIs, Bedrock calls, DynamoDB reads.	/create-fastapi-endpoint, /create-pydantic-model, /create-alembic-migration.	Validates schemas; signs off on contracts.
4. Secure and comply	IAM audit, secrets scan, Terraform review, PII redaction.	/iam-policy-audit, /fix-hardcoded-secrets-tf, /terraform-best-practices, /create-pii-redaction.	Risk sign-off; pen-test; regulatory approval.
5. Test and review	Unit + integration tests, PR review, perf benchmarks.	/create-test-suite, /review-pr, review-team agent, /benchmark-endpoint.	Approves merge; signs off on quality gates.
6. Deploy	CI/CD builds, scans, deploys to AWS.	Generated GitHub Actions workflows, Terraform apply.	Approves production deployment.
7. Operate	Runbooks, dashboards, incident response.	/generate-runbook, /generate-architecture-diagram, /explain-codebase.	Owns on-call and SLOs.
8. Iterate	New features, new personas, new data sources.	Same plugin skills, re-run against updated YAML.	Sets product priorities.

CLOSING: FROM SCAFFOLD TO STRATEGY

This is not an internal productivity story. It is a delivery accelerator for client engagements. A prospect's domain—claims, suitability, equity research, cyber, underwriting—can be produced as a click-through prototype during a sales conversation, not written up as a four-week proposal.

Client onboarding to the 10X-Analyst platform compresses from a multi-quarter platform build into a one-quarter app build, because the platform pieces are already done and the new application is a YAML configuration plus business logic.

A client weighing five candidate AI use cases does not have to pick one—Deloitte can stand up all five so the decision is made against real prototypes, not slideware.

In regulated industries—financial services, health care, government, insurance—the platform delivers the observability and security posture by default. Observability and security posture gaps remain among the most frequently cited barriers to enterprise AI adoption in these sectors—and it is precisely this gap the stack closes. Deloitte's position as a lead collaborator in Anthropic's Claude ecosystem, and as the firm behind the 10X-Analyst featured on Anthropic's Claude for Financial Services page, helps accelerate and uniquely positions our ability to make this pattern repeatable at the speed regulated clients require.

The most expensive AI prototype is the one that succeeded in a demo and then could not be replicated, because the next team had to start from scratch with different opinions about observability, security, infrastructure, and deployment. Most enterprise AI programs stall not at the first prototype but at the third, when leadership realizes the marginal cost of each new domain has not dropped.

> *The question is no longer "Can we build an AI-powered analyst workbench?" It is "How many business domains will we roll out this quarter?"*

The 10X-Rapids plugin attacks that ceiling directly. By encoding hard-won opinions about how to build a 10X-Analyst application into executable skills, specialist AI agents, and enforceable safety hooks, it makes the second, fifth, and fiftieth business domain as economical to roll out as the first.

Claude Code on Amazon Bedrock supplies the raw agentic capability and the enterprise governance. The 10X-Rapids plugin supplies the discipline—the patterns, the guardrails, the role-based agents, and the muscle memory of an engineering organization that has already done this once and now refuses to do it inconsistently ever again.

CONTACT US

Nithin Gopidi
Associate Vice President, AI & Data
Deloitte Consulting LLP
ngopidi@deloitted.com

Kasi Muthu
Global Data & AI Solutions Architect
Amazon Web Services (AWS)
kasimuth@amazon.com

Devyani Prafulla Kumar
AI Architect & Senior Developer
Deloitte Consulting LLP
devykumar@amazon.com

About Deloitte

As used in this publication, "Deloitte" means Deloitte Consulting LLP, a subsidiary of Deloitte LLP. Please see www.deloitte.com/us/about for a detailed description of our legal structure. Certain services may not be available to attest clients under the rules and regulations of public accounting.

This publication contains general information only and Deloitte is not, by means of this publication, rendering accounting, business, financial, investment, legal, tax, or other professional advice or services. This publication is not a substitute for such professional advice or services, nor should it be used as a basis for any decision or action that may affect your business. Before making any decision or taking any action that may affect your business, you should consult a qualified professional advisor.

Deloitte shall not be responsible for any loss sustained by any person who relies on this publication. Copyright © 2026 Deloitte Development LLC. All rights reserved.

About Amazon Web Services (AWS)

Amazon Web Services (AWS) is guided by customer obsession, pace of innovation, commitment to operational excellence, and long-term thinking. By democratizing technology for nearly two decades and making cloud computing and generative AI accessible to organizations of every size and industry, AWS has built one of the fastest-growing enterprise technology businesses in history. Millions of customers trust AWS to accelerate innovation, transform their businesses, and shape the future. With the most compre hensive AI capabilities and global infrastructure footprint, AWS empowers builders to turn big ideas into reality. Learn more at aws.amazon.com