Superblocks

The CIO's AI Security Playbook for the enterprise

Defending the enterprise against vibe coding

Over the past year, our team has undergone some of the most rigorous AI security reviews that any modern software vendor has faced from Fortune 500 enterprises. We can't name the customers. The questions they asked, though, have become the de facto standard for how serious CIOs and Heads of IT should think about AI-era security. This playbook turns those questions into the controls every large company needs before it lets vibe coding into production.

Why vibe coding changes the threat model

Vibe coding pushes software creation out of engineering and into every business function. A finance analyst writes an app that reads from Salesforce, joins it with HR data, and posts to Slack. A sales operator stands up a workflow that queries a data warehouse and emails customers. The code is generated by an AI. The reviewer is the same person who wrote the prompt. The integrations use whatever credentials the builder can get their hands on. None of this hits your traditional SDLC.

The result is a new class of risk. AI agents move faster than your change advisory board. Generated code reaches production data without a human-readable diff. Every prompt is a potential exfiltration vector. Every new integration is a service account you didn't approve. The platforms enabling this work concentrate enormous power in one runtime, and most CIOs are evaluating them with security questionnaires built for SaaS dashboards. Those questionnaires miss the point.

Vibe coding is not a productivity feature. It is a new application surface. The controls below are the minimum bar for trusting that surface in a regulated enterprise.

Seven controls every AI development platform must support

1 — Defensive review of AI-generated code

Every line of AI-generated code is untrusted until it has been reviewed by something other than the AI that wrote it. The code generation agent cannot be the single point of failure before code reaches production data. Wrap every change in defensive layers: static analysis tools that catch unsafe patterns, security agents that scan for secret exposure and risky integrations, and policy checks that block dangerous capabilities by default.

On top of that, demand a full history of every change with the ability to roll back safely, including database migrations. The exact mechanism matters less than the outcome. If the only thing standing between an AI suggestion and your customer data is the human who typed the prompt, you do not have governance.

2 — Runtime sandboxing and per-tenant isolation

AI-generated code must execute in a hardened sandbox from day one, with explicit network egress controls and per-tenant isolation. The platform needs an automated kill switch that disables a tenant or an app when error rates spike, before a runaway agent burns through API quotas, customer records, or downstream systems. Sandboxing adds latency. Accept it. The platforms that skip it are gambling with your blast radius.

3 — Identity-bound integrations, no static keys

Treat every integration like a production service. OAuth, service principals, or mTLS-backed certificates only. No static API keys pasted into a prompt. No popup re-auth flows that train users to click through warnings. Long-lived OAuth tokens with token exchange where the upstream supports it. For internal systems, federate through your identity vault and LDAP. Every action an AI agent takes must trace back to a named human or a documented service identity in your IdP. If you can’t answer "who ran this query," you don’t have governance.

4 — Deployment governance with real kill switches

Pull-based upgrades, not push. Your environment, your cadence. First-time deployment of any app requires admin approval. Subsequent deploys are configurable, and adding a new integration to an existing app re-triggers approval automatically. Sensitive apps gate every deploy. Production data planes sit on a separate network from the editor, with a routing layer that enforces the rule mechanically, not by convention. When a vulnerability is found, you need a single switch that blocks all deploys across the platform until it is fixed.

5 — AI-specific data controls

Every prompt and every read is a potential data egress event. The platform must scan and redact secrets on the way to the model, not just at session start. You need a clear inventory of what leaves your environment for inference, what gets logged, and what can be turned off. New telemetry fields must default off behind a feature flag, and your settings must persist across vendor upgrades without configuration drift. If the vendor cannot tell you exactly what fields move through their telemetry pipeline, assume the worst.

6 — Centralized observability across every AI-built app

You will end up with hundreds of AI-generated apps inside two years. You need a single governance plane that lists every app, every builder, every integration, every prompt, and every API call. Anomaly detection on usage patterns. Per-app, per-user, and per-integration rate limits, because some apps are more equal than others. Audit trails that retain ninety days minimum and export cleanly to your SIEM. Without this layer, vibe coding becomes the new shadow IT, except faster and louder.

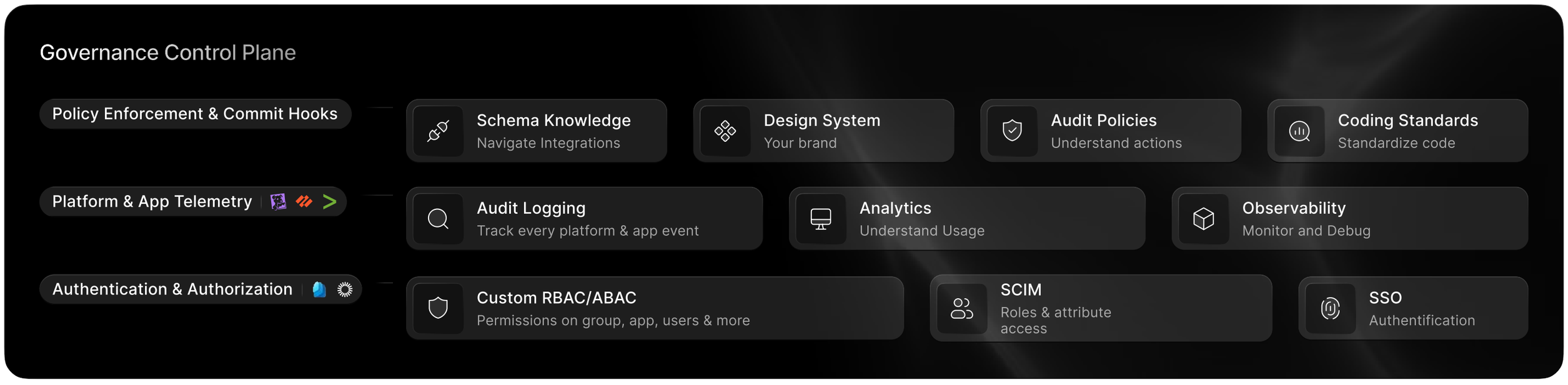
7 — Joint validation and continuous penetration testing

A vendor pen test on the vendor’s SaaS environment does not certify your deployment. Demand a joint penetration test on your actual infrastructure once the platform is live. Demand an equivalency attestation when major features ship. Image scanning on every dependency, signed images verified at deploy, and customer-managed KMS for any data at rest. The vendor should run these scans themselves before you do, and produce findings on their own initiative. If your security team is the one finding the bugs, the vendor is not ready for the enterprise.

Governance as a control plane

> Vibe coding for business teams, with the controls IT needs.

The seven controls above describe what to demand. The governance control plane is how a platform delivers them at scale. Without one, you end up with point solutions glued together: a code scanner here, an audit log there, an identity provider that doesn't talk to the audit system. A real control plane sits between the people building apps and the systems those apps touch. It enforces policy on the way in, watches everything on the way through, and ties every action to a real identity. Three pillars, working together.



Policy enforcement

Policy enforcement decides whether code ships. It runs at commit time, before any deploy, on every AI-generated change. Standards become checks. Reviews become automatic. The code-generation agent never gets the last word.

Security Scanning. Every commit scanned for vulnerabilities, secret exposure, dangerous packages, and policy violations before the change reaches production. Findings block the deploy.

Defensive Agents. Independent AI agents that review what the code-generation agent produced. They look for issues the prompt author would miss, including risky integrations, dangerous capabilities, and prompts that violate platform policy.

Coding Standards. Architecture, patterns, and naming conventions enforced as automated checks across every AI-built app. Style guides stop being advice and start being gates.

Secrets Management. Secrets vaulted centrally and referenced by name. No raw values in prompts, repositories, error messages, or logs. Rotation handled by the platform.

Platform and app telemetry

Telemetry shows what is actually being built. Every action by every AI agent and every builder is logged, indexed, and queryable. The platform you can't see is the platform that becomes shadow IT in eighteen months.

Audit Logs. Every platform event and every app event captured: user, action, resource, timestamp, outcome. Retained for at least ninety days and exportable to your SIEM in a structured format.

Observability. Traces, metrics, and logs available for every app and every workflow. Builders can debug without a ticket. Operations can identify slow integrations and flaky external APIs without guessing.

Monitoring and Proactive Resolution. Automatic alerts on error spikes, integration failures, and unusual usage patterns. The platform disables runaway tenants before they exhaust quotas or surface customer data in the wrong place.

Identity and access

Identity ties every action to a real person or a documented service identity. Nothing happens on the platform without a name attached to it. If you can't answer who built it, who deployed it, or who ran the query, the platform is failing you.

Custom RBAC and ABAC. Granular permissions on apps, integrations, folders, and any other resource. Role-based for the default cases, attribute-based for the edge cases. Permission inheritance so folders cascade.

SSO. Every login federated to your identity provider. No platform-local passwords. MFA enforced by your IdP, not by the vendor.

SCIM. Users and groups provisioned and deprovisioned automatically from your directory. When someone leaves the company, their access disappears in minutes, not in the next quarterly access review.

How the three pillars work together

Policy enforcement decides whether code ships. Telemetry shows what is actually being built. Identity ties every action to a real person. When all three live in one platform, IT supports hundreds of business builders without losing visibility or control. When they live in three separate tools, the gaps between them become the place every incident lives.

The integration layer

> Where governance meets your business systems.

Most security incidents on AI development platforms start with an integration. A static API key in a prompt. A service account with too many permissions. A connection to a production database that nobody approved. The integration layer is the surface where this either happens or doesn't.

A real integration layer is a managed, audited, IT-controlled boundary between every AI-built app and every business system the app touches. It is not a list of connectors. It is a governance surface. Three things have to be true.

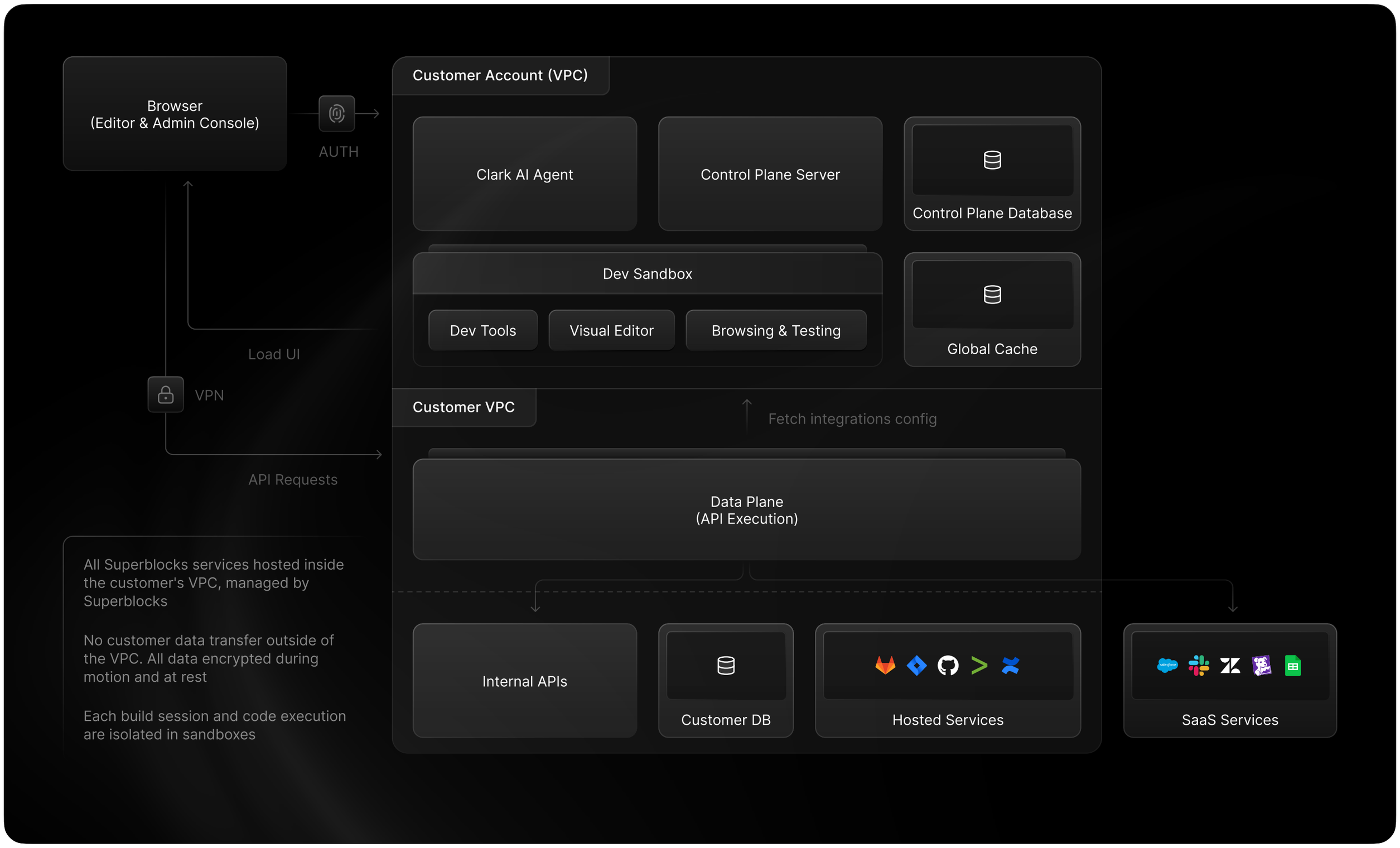
Every connection is identity-bound. OAuth, service principals, or mTLS-backed certificates. No static keys in prompts, no credentials pasted into code. When an AI agent connects to Salesforce, the credential traces back to a named user or a documented service identity in your IdP. Token exchange where the upstream supports it.

Every secret lives in the vault. Apps reference secrets by name. Raw values never appear in prompts, repositories, logs, or error messages. Rotation is automated and centrally managed.

The deployment model

> Where governance meets your business systems.

For most SaaS, deployment is a footnote in the contract. For an AI development platform that touches every business system you own, deployment is the contract. The deployment model decides where your data sits, where AI inference runs, where customer-bound traffic flows, and how much control you actually keep when something goes wrong. Three models matter.



Every integration has its own gates. Per-integration rate limits, per-integration approval flows, per-integration scope. Adding a new integration to an existing app re-triggers approval. Disabling an integration disables it everywhere it is used.

The integration layer is also the place to enforce least privilege at the data level. Row-level access. Column-level masking. Per-app data scopes. The platform should make it easier to grant a business builder narrow access to one specific dataset than to grant them broad access to everything.

Treat the integration layer as part of your security perimeter. If your vendor treats integrations like a config dropdown, walk away.

Cloud

The entire platform runs in the vendor's cloud. Your apps connect outbound to your SaaS systems and databases over the public internet, with IP allowlisting and OAuth-based authentication. Fastest setup. Lowest operational overhead. Best fit when production systems can be reached over public endpoints and data sensitivity is moderate. The model most startups and mid-market teams begin with, and the model most CIOs ask to leave when the platform reaches real production scale.

Hybrid

The data plane runs in your VPC. The control plane and AI inference run in the vendor's cloud. Production data never leaves your network. AI inference works on non-production data only. Best fit when production systems cannot be exposed publicly but non-production environments can safely interact with cloud-hosted services. A practical middle ground for companies with strict data residency rules and a pragmatic stance on dev and staging.

Cloud-Prem

The entire platform runs inside your AWS, GCP, or Azure cloud environment. Control plane, data plane, AI inference, every component. The vendor manages the platform inside your boundary. No customer data ever leaves your environment. Single control plane with multi-region data planes for global enterprises that need data residency by region. This is the model regulated industries, financial services, healthcare, and security-led tech companies ask for by name. It is what serious CIOs say yes to.

Comparison

Model	Where the platform runs	Where data sits	Where AI inference runs	Best for
Cloud	Vendor cloud	Vendor cloud (transiting)	Vendor cloud	Speed of setup, lower data sensitivity
Hybrid	Vendor cloud (control) and your VPC (data)	Your VPC	Vendor cloud, non-production data only	Production data stays in your VPC
Cloud-Prem	Your cloud	Your cloud	Your cloud	Maximum data residency, isolation, and compliance

How to choose

If you are a fast-growing tech company between 500 and 5,000 employees adopting AI for the first time, you probably begin with Cloud or Hybrid. Most of you will end up at Cloud-Prem within eighteen months, for two reasons.

First, your security team gets one look at how much data flows through the platform and demands it. Second, you cross a customer or compliance threshold (SOC 2 Type 2, HIPAA, FedRAMP, a regulated industry contract) that requires data residency by region.

Picking a vendor without a Cloud-Prem path means you will be ripping out the platform exactly when it is most embedded in your business. Pick a vendor whose Cloud-Prem story is already production-ready, not a roadmap promise.

The CIO’s vendor checklist

Print this. Hand it to every AI development platform you evaluate. The right vendor answers each row in writing, with a demo, and without flinching.

Control Area	What to ask the vendor
Defensive review	AI-generated code is treated as untrusted by default. Static analysis, security alerts, code reviews, and design review are entry touchpoints and placed in the development lifecycle to detect problems before code enters a customer-facing production database changes.
Runtime isolation	AI-generated code runs in a hardened sandbox by default. Show me the infrastructure protection layer architecture that governs where code can be deployed and how it can be spike.
Deployment gates	We always work with canary deployments and rich code deployment. We always work with canary deployments to production and which auto-deploy.
Vigilant control	You probably need vendor discovery of service accounts. You may also need a feature flag that defaults off. Settings paired across segments, no config drift.
AI data controls	Show me where teams own environment for inference, where gen1 redacted, and how I turn outlining on for every prompt and every read.
Audit and observability	Does someone own this every day, every bubble, every integration, and every prompt and file platform? Also, platform logs and service provider logs, and dashboard that proves compliance.
Validation	I want code before deployment. I see code everywhere in a full stack, joint pre-test on our infrastructure, not yours. Equivalency attestation about deployment methods, model stability and how a pipeline matches what I approved.

What good looks like

The fast-growing tech companies winning with AI in 2026 are not the ones that banned vibe coding. They are also not the ones that let it run wild. They picked one governed platform, gave their business teams real tools, and wrapped that platform in the seven controls, the governance control plane, and a deployment model their security team approved. Every AI-built app is reviewed defensively. Every integration uses a real identity. Every prompt is observable. Every deploy is gated. Every model interaction is auditable. The whole stack runs in their cloud, on their cadence, under their boundary.

Their security teams stopped saying no, because they finally had something to say yes to. That is the bar. Set it for every vendor that walks through your door. Anything less is a future incident report.