Deploying AI in an on-premise environment

A how-to guide

Written by COSINE

Executive summary

On-premise, once considered outdated, is becoming an attractive option for many organisations when it comes to AI deployment.

Running AI inside your boundary can deliver stronger control, lower cost volatility, and better compliance than cloud deployments. The shift reflects a growing need for secure, predictable, and auditable AI infrastructure in software engineering.

On-premise deployment of AI requires enterprises to make three thoughtful design decisions.

Without cloud elasticity or proprietary APIs, teams must make four core decisions up front:
1. How many GPUs to dedicate;
2. Which open-weight model to run;
3. How much post-training to perform

1. GPU capacity is the main determinant of AI's accuracy and throughput.

Too few GPUs, and accuracy, latency, and throughput degrade sharply. The right investment here sets the upper bound on model quality and responsiveness.

2. Your choice of foundation model then defines the performance baseline.

Open-source models vary widely in performance, licensing terms, and language coverage. Selecting the right foundation determines how far your deployment can go before needing significant fine-tuning.

3. Custom post-training provides a real performance edge for organisations.

Tuning a model on internal or domain-similar codebases transforms generic output into high-fidelity engineering assistance, raising precision and trust while maximising value per GPU hour.

Leaders should deploy now and iterate over time, rather than waiting for perfection.

You don't need to start with a fully post-trained model or a large GPU cluster. Begin with a strong open-source base, deploy it securely inside your boundary, and evolve with post-training over time. The best way to build confidence in AI is to start, and then iterate safely inside your own walls.

About Cosine

Cosine is an agentic AI software engineer for highly secure, on-premise environments, fine-tuned to each customer's codebase.

Cosine provides autonomous, policy aware engineering agents that work through the software delivery pipeline. The agents open evidence rich pull requests, generate and run tests, satisfy security and compliance checks, and attach clear rationale so reviewers can approve with confidence. Cosine deploys inside your boundary and integrates with your existing repositories, continuous integration, and security tooling.

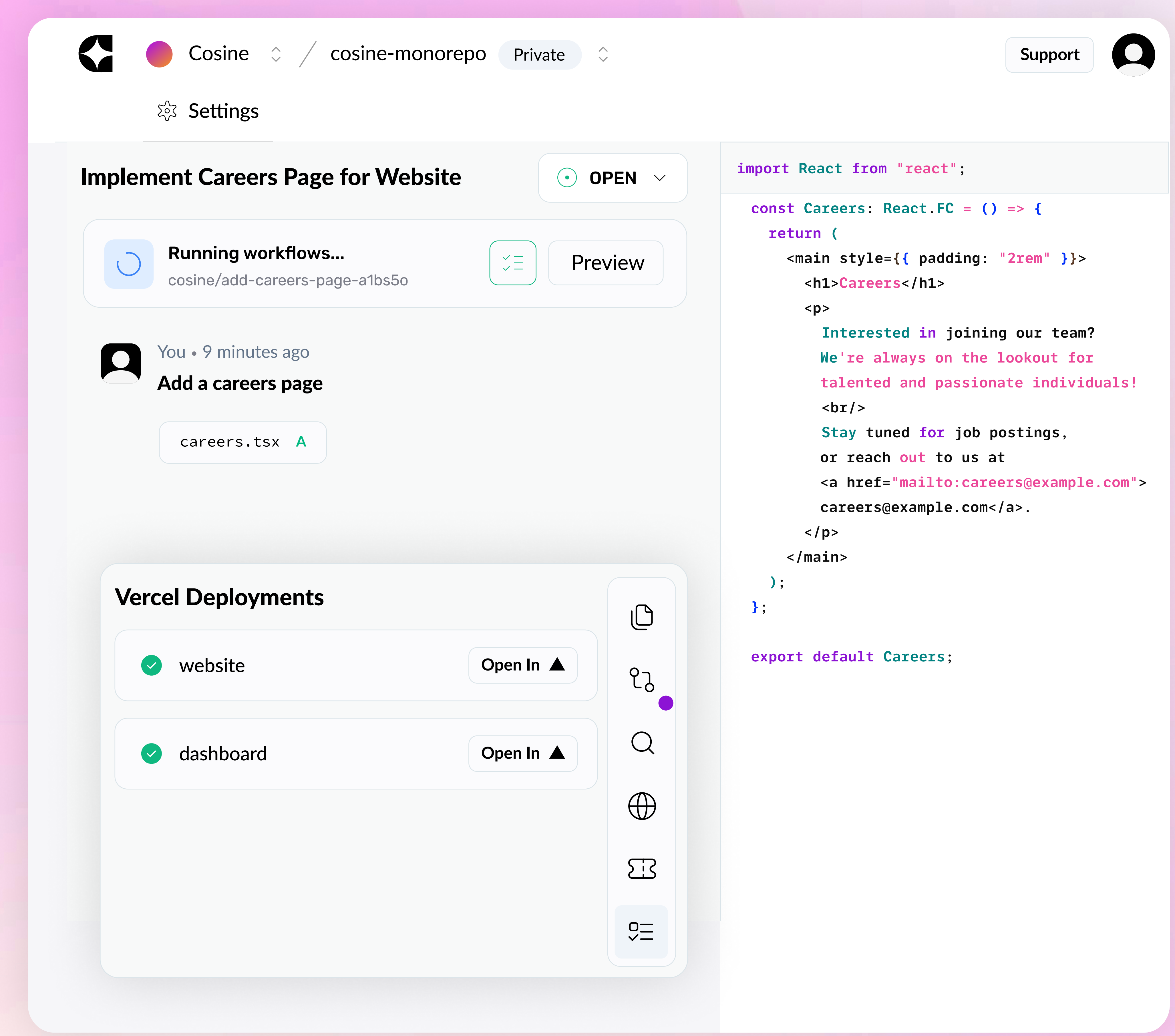
Cosine serves engineering leaders who need measurable throughput in complex or regulated settings. Typical owners include Heads of Engineering, Platform Engineering, and Application Security. Teams use Cosine for high volume flows such as test generation and maintenance, small bug fixes, dependency and lockfile updates, flaky test repair, documentation at scale, and targeted security remediation. The system is asynchronous and queue driven, so it clears backlogs without interrupting developer focus.

Cosine is available for free as a cloud service online, while enterprises can explore virtual private cloud or on premise fully air-gapped deployments. Cosine's vertically-integrated setup ensures no on-premise data egress.

For enterprises, Cosine can custom-train LLMs on specific coding languages and/or internal data. This drives higher accuracy at an efficient level of compute and cost.

Organisations adopt Cosine to increase pull request throughput and merge rate, raise build success, shorten time to remediation, shrink aged technical and security debt, and maintain a complete audit trail with data kept inside their boundary.

Give it a try **Book a demo**



Implement Careers Page for Website

Running workflows...
cosine/add-careers-page-a1bs5o

You • 9 minutes ago
Add a careers page

careers.tsx A

```
import React from "react";

  const Careers: React.FC = () => {
    return (
      <main style={{ padding: "2rem" }}>
        <h1>Careers</h1>
        <p>
          Interested in joining our team?
          We're always on the lookout for
          talented and passionate individuals!
          <br/>
          Stay tuned for job postings,
          or reach out to us at
          <a href="mailto:careers@example.com">
          careers@example.com</a>.
        </p>
      </main>
    );
  };

  export default Careers;
```

The surprising resurgence of on-premise

"Reports of my death are greatly exaggerated"

Mark Twain (supposedly)

For a decade the story was clear: enterprises were progressively shifting towards SaaS and public cloud, and away from on-premise.

In 2025, the narrative has become more interesting. Enterprises are still adding net-new cloud workloads, but a meaningful share of existing work is returning to on-premise.

SaaS is not shrinking overall - spend is still growing. But selective repatriation is happening.

Source: Citrix; Barclays; Flexera

94%

A Citrix survey found that 94% of respondents admitted to having **moved some workloads back from the cloud** in the past three years

42%

Meanwhile, 42% of those surveyed said they are **considering or have already repatriated at least half** of their cloud-based workloads

83%

In a Barclays CIO survey, 83% of CIOs indicated they **intend to move some workloads back** to private or on-premise infrastructure in 2024–25 (not necessarily full reversals)

21%

Flexor's latest State of the Cloud shows 21% of workloads and data **have already been repatriated** to data centres

Cost, security and regulatory factors are tilting the balance in favour of on-prem…

Why is this repatriation taking place? 3 reasons:

1. **Cost control and budget predictability.** After a decade of "pay-as-you-go," many organisations have discovered they are paying too much for tools. Flexera's latest survey estimates that 27% of IaaS/PaaS spend is wasted, and notes that repatriation is frequently mentioned as a cost-optimisation lever. In contrast, on-premise (or private cloud) restores a fixed, knowable compute envelope.

2. **Security, sovereignty, and governance.** Leaders increasingly cite security and control as primary reasons to rebalance. Bringing critical data and code back inside the boundary reduces exposure and makes audit, key management, and RBAC truly first-party. Recent increases in cyberattacks globally (Check Point's Q1'25 view reports a ~47% YoY jump in average weekly attacks per org) have accelerated this trend.

3. **Regulatory pressure and data residency.** As data-use rules tighten, the path of least resistance is to process sensitive workloads where organisations own the rails.

…and especially for AI tools

Meanwhile, AI is accelerating the shift back to on-premise.

Each API call may silently export chunks of context, making it nearly impossible to audit whether data escaped.

Models can memorise or reconstruct proprietary inputs, unintentionally revealing sensitive logic or code.

Behind the scenes, dynamic GPU scaling and "burst" usage can shift costs unpredictably, and prompt caching introduces side-channel leakage risks (research has identified shared model caches across users).

In contrast, on-premise flips the equation: data never leaves your boundary, governance stays internal, and GPU capacity becomes a fixed asset you own and control.

Source: Citrix; Check Point; Flexera

The design burden | Why on-prem AI requires more thought

Shifting AI from cloud to on-premise is inherently more complex, and requires you to make decisions around design, provisioning and governance. When it comes to AI software developer tools, there are three key decisions:

1

What hardware do I deploy?

You'll need to decide how many GPUs to allocate and how to balance them between inference and training workloads. Under-provisioning limits your accuracy and throughput; over-provisioning ties up capital and power.

2

What open-weight AI model do I choose?

Instead of an instant API call to GPT-5 or Claude, you'll need to choose and host an open-weights model, like Llama 3, Mistral, Gemma, DeepSeek, Phi-3, and others. Each of these has different strengths, licenses, and compute footprints. Picking one that fits your stack and GPU budget is critical.

3

What post-training do I carry out?

To reach enterprise-grade precision, many organisations choose to train their chosen model on internal data, domain-specific code, or synthetic analogues. The more specialised and critical the work, the more this matters. This is how you turn a capable base model into a high-fidelity engineering assistant.

1 ✦ Hardware provisioning | Define your ceiling

The first question is what hardware to deploy. With on-premise, each GPU you allocate is deliberate, and in sum they define the performance envelope for everything a model can do: latency, accuracy, and concurrency all depend on it.

Model scale

Language models scale non-linearly with parameter count and compute. Doubling parameters can multiply throughput requirements by 3–5x once you account for context windows and concurrency. A 70B-parameter model therefore typically demands 2–4 A100s (40–80 GB) just to serve interactively, while a 13B model can run comfortably on a single high-end GPU.

Trade-offs

Smaller models are cheaper to serve but less robust to complex prompts and domain shifts. Cosine's internal benchmarks show roughly a 15–20pp drop in task-level accuracy when stepping down from a 70B to a 7B model on real engineering tasks.

Ultimately, your decision on how many GPUs to allocate depends on **two factors:**

1. **How many you can afford**
2. **What tolerance for error you can expect**

Teams running code generation and test-automation workloads often find that **2–4 well-utilised GPUs deliver most of the value, provided fine-tuning closes the precision gap.**

GPU allocation can also be minimised if you build your GPU set-up and agentic workflows efficiently. Specifically, quantisation and low-rank adaptation (LoRA) reduce memory load while inference batching improves utilisation. Some enterprises also pair high-end GPUs for model serving with smaller cards (e.g. L4s) for agent orchestration, optimising cost per request.

To Conclude

The sweet spot is right-sizing compute to the precision and latency that matter for your business, then maximising efficiency through fine-tuning and orchestration.

2 ✦ Model choice | Define your baseline

Picking the right model is a foundational decision: it establishes what "good" looks like before any fine-tuning optimisations.

Here are the trade-offs you'll juggle:

- **Accuracy vs. efficiency / cost:** larger models tend to have better generalisation and subtle reasoning, but cost more to serve.
- **Context window & memory:** models vary in how many tokens they support (important for long codebases or multi-file context).
- **License & usage terms:** "open source" isn't uniform—some models allow full commercial use, while others impose redistributor restrictions (e.g. LLaMA's "open" license isn't fully open by OSI standards)
- **Community & tooling support:** for example LLaMA has a large ecosystem, which helps with integrations and debugging
- **Model source:** many of the highest-quality open source models are Chinese, which some organisations choose not to deploy for security reasons

The table below advises on a realistic model choice for a given level of compute:

# GPUs (H100 or equivalent)	**Recommended base model** (as of January 2026)	**Alternative options** (as of January 2026)
2-7	gpt-oss-120B	Llama 3.3 70B; Qwen 2.5; DeepSeek Coder 33B
8-15	Mistral Large 3; Deepseek V3.x	Qwen-Coder-480B; Mistral-next
16 or more	Kimi K2 thinking	Qwen-Ultra

3 ✦ **Post-training** | Define your edge

Generic, open-weights models are a great start, but post-training is how you turn them into high-fidelity engineering assistants.

Domain-tuning consistently lifts task accuracy and reduces off-target answers on specialised work. For example, BloombergGPT (50B), trained on finance corpora, outperforms similar-size general models on financial tasks while remaining competitive on general benchmarks.

Post-training helps by teaching a model your languages, frameworks, build systems, and policies, so it proposes diffs that match your stack and passes your gates. Multiple studies show that targeted adaptation improves factuality in domain tasks and counters hallucinations compared with base models alone.

Given the complexity and cost of these processes, organisations can adopt an incremental approach where post training initially takes place on domain-similar data and only extends to internal code if necessary - i.e. if you use a very niche coding language or have very specific working norms.

Organisations can choose which data to post-train a model on - these options vary in terms of complexity, cost and likely accuracy:

Least complex but least accurate

Synthetic, domain-similar data (fastest to start; useful to shape style and interfaces)

↓

Moderate complexity, moderate accuracy

Public, language/framework-matched repos and docs (better: closer to your real code paths)

↓

Most complex but most accurate

Internal code, tests, runbooks, policies (maximizes “ready-to-merge with proof”). Pair this with long-context tooling when you need to expose policy packs or larger repo slices during training/eval

Source: 2024 Developer Survey, Stack Overflow

Don't let perfect be the enemy of the good

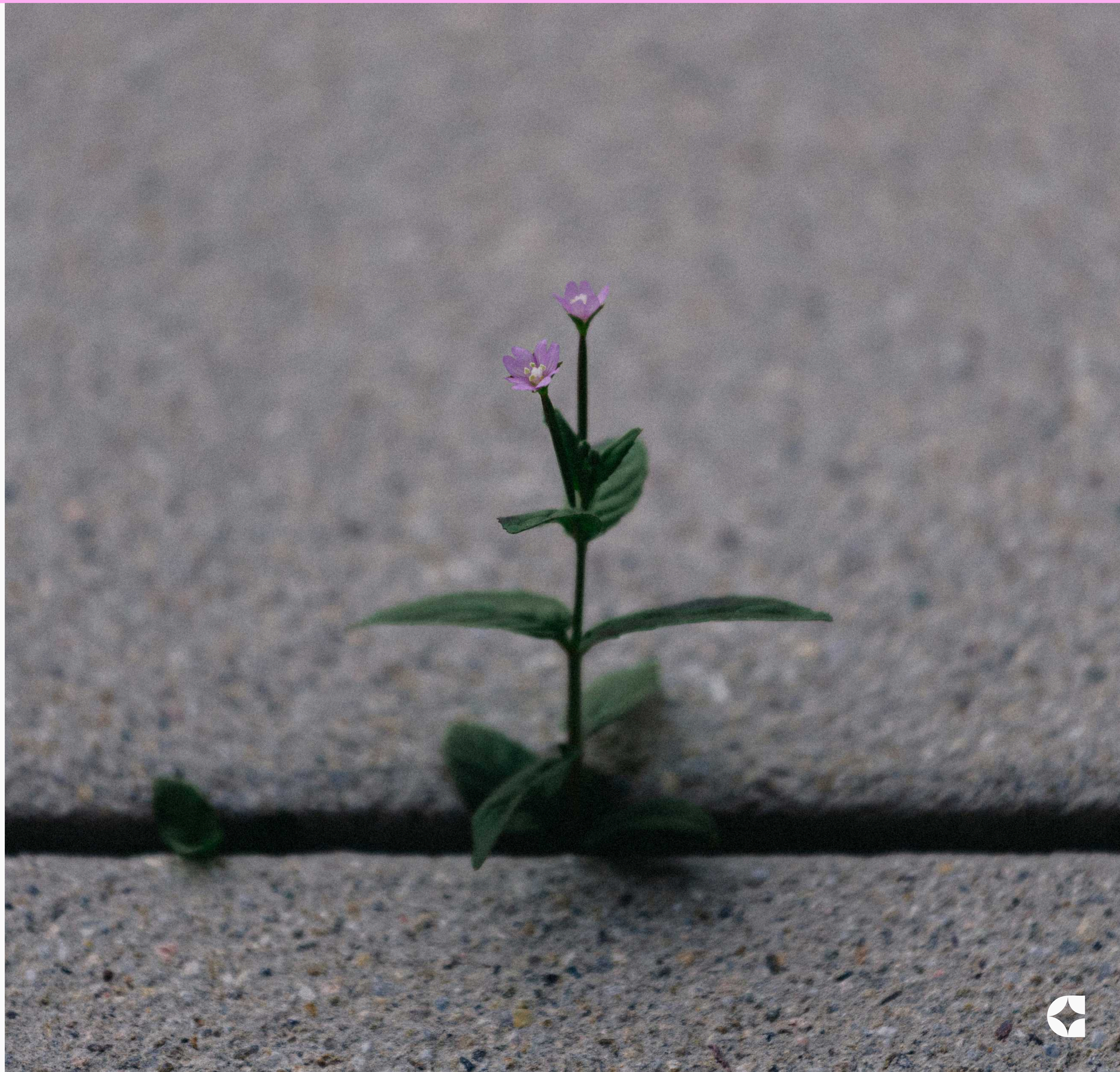
The hardest part of deploying AI on-prem isn't the technology. In practice, teams often delay action chasing the "ideal" setup: perfect data pipelines, fully post-trained models, or a complete GPU cluster. **In practice, those ambitions can stall momentum before value is proven.**

A bias to action is critical. A strong open-source model running securely inside your boundary is already a meaningful step forward. It gives you immediate control over data, cost, and policy, while proving how AI interacts with your codebase and workflows.

Once on-premise is deployed, organisations can then continue to iterate. Once the first deployment is stable, layer on the next improvements: post-training, policy integration, test automation. Each round compounds accuracy and trust without the overhead of a full rebuild.

Orgs can then measure tangible outcomes (merge rates, time-to-merge, CI pass rate) and judge what needs to be ramped up and when.

The enterprises seeing the fastest returns are those who ship a minimal on-prem AI capability early, learn from telemetry, and scale deliberately. Start small, stay secure, and evolve.



✦ What next? | The 12-month plan for tech leaders

Objective: Within 12 months, AI becomes a governed participant in the software delivery pipeline — fully within your boundary, policy-aware by default, and tuned to your stack.

Month 1

Build the foundation

- **Define scope and ownership:** name accountable leads
- **Baseline metrics:** record current pull-request throughput, merge rate, build success, time-to-remediation, and aged issues
- **Set up a pilot environment:** stand up a private or air-gapped cluster; connect repos, CI, and security scanners
- **Start small:** deploy a strong open-weights model (e.g., Mistral 7B or Llama 3 70B) for low-risk flows such as documentation or small bug fixes

Months 2 to 6

Prove and tune

- **Instrument telemetry:** measure latency, GPU utilisation, and quality of AI-touched changes
- **Begin lightweight post-training:** use domain-similar public or synthetic data to close precision gaps
- **Enforce policy-as-code:** require every AI-generated change to ship with tests run, checks green, and a short rationale
- **Governance cadence:** weekly review with security and monthly KPI reports to leadership

Q3-Q4

Expand

- **Add more workflows:** extend to test generation, flaky-test repair, dependency updates, and security remediation
- **Introduce internal post-training:** train adapters on your own code, policies, and design docs
- **Harden infrastructure:** implement automated patching, continuous vulnerability scans, and encrypted model artifact stores
- **Monitor ROI:** expect measurable gains of +15-25% first-pass approvals, −20-30% time-to-merge, and higher build health

Year 2 onwards

Standardise & scale

- **Move from pilot to platform:** offer on-prem AI as an internal service, with defined SLAs and access tiers
- **Full audit integration:** all model inferences and commits logged locally for compliance
- **Cost optimisation:** quantise models, right-size GPU allocation, and automate adapter refreshes
- **Continuous improvement:** establish a retraining cadence using fresh diffs, incidents, and regression tests

Closing thoughts

The cloud made AI accessible, but on-premise will make it accountable. Enterprises are rediscovering the value of sovereignty: knowing exactly where models run, what data they see, and how costs scale.

This report first traced that shift: we see a clear repatriation of workloads from public cloud to private infrastructure.

Then it identified the reasons why: cost volatility, security fatigue, and regulatory gravity. The arrival of generative AI simply sharpened that choice: on-premise brings AI inside the boundary where it can be governed and proven.

Then we outlined the playbook: start by allocating GPUs; then select an open-weights model; and then layer on post-training to deliver precision.

And finally, show a bias to action: deploy early, observe what works (and doesn't), and evolve accordingly.

In time, the organisations that win with AI will be those who learn to use it securely and accurately. On premise provides an excellent avenue.

COSINE