

Aurora Solar grows its IaC coverage from under 5% to over 95% with Spacelift

spacelift

aurora

95%+

Infrastructure managed as code has risen from < 5% to 95% in four years.

185

Each of the 185 stacks running through Spacelift is checked for drift every 30 minutes.

~200

Spacelift handles ~200 Terraform changes a week, about a full-time engineer's worth of work

Summary

Aurora Solar builds the software platform solar companies use to design and sell photovoltaic (PV) systems, with more than 20 million projects designed on it to date. Before adopting the Spacelift platform in 2022, its seven-person platform team applied Terraform or Terragrunt CLI manually, with no CI/CD, no audit trail, and less than 5% of Aurora's infrastructure managed as code. Today, Spacelift runs around 185 stacks covering more than 95% of Aurora's infrastructure, with drift detection and automatic reconciliation on every stack. No engineer holds direct write access to AWS, and lead time for infrastructure changes has plunged from days to hours or even minutes.



No engineer at Aurora has write access to AWS. Every infrastructure change is a change to code, checked by Spacelift before it reaches production.



A shared internal module in Aurora's private Spacelift registry gives every new stack the same policies, contexts, and drift detection by default.



Coding agents like Claude Code now read Aurora's existing Terraform and write changes that match its standards, reviewed under the same policies as any human-authored change.

EA software platform that streamlines solar design, Aurora employs 180 people, 80 of them engineers. Seven of those engineers make up Aurora's platform team, and platform work is only part of what the team does. They support all 80 engineers at Aurora, including themselves: The same team that builds Aurora's infrastructure workflow is one of its main users.

We spoke with Morley Abbott, Senior Engineering Manager at Aurora Solar, about how the team took Aurora's infrastructure from manual and undocumented to governed and largely self-service, using Spacelift.

The challenge for Aurora Solar

Before Spacelift, infrastructure at Aurora didn't run through a pipeline. Most of what was running in AWS had been created by hand in the console rather than defined anywhere as code. Less than 5% of Aurora's infrastructure was managed as code at the time, and the company was still largely running on Heroku.

"That left gaps. There was no audit trail of who applied what, and drift accumulated between what was committed and what was actually running," says Morley Abbott, Senior Engineering Manager at Aurora Solar. It became increasingly clear that the way Aurora was managing infrastructure needed to change. "We needed a developer-friendly tool to roll out standardized IaC across the org," Morley says. Migration planning aligned. When they returned in August 2025, the deal closed in two weeks.

Why Aurora Solar chose Spacelift

Because Aurora had no existing infrastructure as code (IaC) platform to migrate from, adopting Spacelift and adopting IaC were effectively the same project. The team researched the market and evaluated Terraform Cloud, Atlantis, and envO before choosing Spacelift, citing competitive pricing, feature depth, and a strong technical team, including an early meeting with Marcin Wyszynski, one of Spacelift's co-founders. A demo answered Aurora's technical and business questions and gave the team confidence in the decision.

Onboarding was straightforward, and the platform team committed to adopting infrastructure as code properly, as well as the automation to run it. "That took us from less than 5% of our infrastructure managed as code to 95%+," Morley says.

Four years in, hurdles have been rare, and Aurora credits Spacelift's customer support team with resolving the few that came up quickly.

Aurora Solar's Spacelift experience

Aurora manages its infrastructure repositories as a hybrid: a monorepo for shared platform infrastructure, and separate polyrepos for service-specific infrastructure. Every Spacelift stack is defined through Spacelift's Terraform provider, never through the UI. Stack definitions live in the same monorepo as the infrastructure they manage, and each one is instantiated from a shared internal module that Aurora publishes to its own private Spacelift module registry and consumes by version constraint. That module wraps the stack alongside its AWS integration, environment variables, policy attachments, context attachments, and drift detection, so a new stack inherits Aurora's defaults instead of being configured by hand. An administrative stack manages the stacks themselves, which means adding a new stack is a pull request against the monorepo, not a console operation. Aurora runs around 185 stacks this way.

Policy as code

Policy is built into that same shared module, so effectively every stack picks up Aurora's default policy set automatically. The team runs plan, approval, git-push, login, and notification policies, with the Rego source held in the monorepo alongside the stack definitions and covered by an Open Policy Agent (OPA) test harness. The plan policy Aurora relies on most requires infrastructure to use Aurora-owned modules. The git-push policy ignores changes outside a stack's root, so an unrelated commit doesn't trigger a run it shouldn't. The notification policy reads slack:<channel-id> off a label on each stack and posts failures there, so individual teams can opt into their own alerting without the platform team wiring it up by hand.

Contexts

Aurora also runs six contexts, mostly for environment variables: integration credentials for Datadog, Infracost, MongoDB Atlas, and its Kubernetes tooling, with secrets sourced from Doppler and passed as write-only values. One context mounts a file: VPC data written to S3, so any stack that needs network details reads it from a file instead of duplicating the lookup. An org-wide context autoattaches to every stack through the shared module, and service-specific contexts for Kubernetes, Helioscope, and MongoDB Atlas autoattach by label, so a stack opts in by carrying the right label rather than the platform team editing an attachment.

Provisioning infrastructure

To get infrastructure provisioned and configured, developers have two options. Self-service at Aurora runs through Backstage, the company's internal service catalog, rather than through Spacelift's own Blueprints or Kubernetes Operator. A developer creating a standard service uses a Backstage template, which generates a pull request against the Terraform monorepo with the stack definition and supporting configuration already written. An administrative Spacelift stack watches that directory with autodeploy enabled, so merging the PR creates and applies the new stack without anyone on the platform team touching it. For anything that doesn't fit an existing template, the request comes through Aurora's Slack support channel, becomes a Jira ticket, and ends as a Terraform pull request written by the requesting developer, the platform team, or both together. "Engineers have no write access to AWS at all, so a change to infrastructure is a change to code by definition," Morley explains.

Drift detection

Drift detection runs on all approximately 185 stacks, checked every 30 minutes, with automatic reconciliation enabled. It's on by default because it's configured in the shared module every stack is built from, not something an individual stack has to opt into. Aurora deliberately keeps its stacks independent rather than using explicit output dependencies or trigger policies; where one stack needs a value another produces, the team passes it out of band. At Aurora's scale, that has meant no run ever waits on another stack's state, and the coupling that does exist stays explicit and easy to trace.

Agentic AI

AI has become part of that workflow too. "The biggest win has been coding agents such as Claude Code reading our existing Terraform and producing high-quality changes that match our standards," Morley says. Aurora also runs review agents with company-specific context and access to related documentation, so a pull request often already carries a first pass of scrutiny before a human looks at it. None of that changes Aurora's guardrails: Every infrastructure change still runs through Spacelift plans as a deterministic check. Whether or not it's AI-assisted, it still gets reviewed by a human. The exclusion of write access to AWS also extends to agents.

Spacelift's impact on Aurora Solar

Spacelift now runs close to 95% of Aurora's infrastructure across its roughly 185 stacks, up from less than 5% four years ago. Lead time to change (from code committed to code running in production) has fallen from days to hours or even minutes.

Developers on every team at Aurora have made a change to the Terraform repo and seen it deployed after merge. "It isn't a frequent task for most of them, but the automation gives us confidence that it's safe, and it takes the load off our team, so we're not babysitting infrastructure deployments," Morley says.

Gaining the equivalent of a full-time engineer

The time that frees up is equivalent to one FTE. Morley explains: "Spacelift is running around 200 Terraform changes a week for us. Doing that by hand would be a full-time job for at least one engineer." And, although, in principle, Aurora could maintain its current deployment cadence by hand, applying Terraform changes manually would take priority over most of what the team does instead.

"The constraint isn't how many deployments we could support; it's what we'd have to stop doing in order to support them," Morley says.

Four years on, Spacelift has blended into the background for Morley's team, and that's just the way they like it. Morley describes the Spacelift experience as "solid and largely uneventful, which is what you want from infrastructure tooling." He highlights drift detection as the clearest indication of how much Aurora trusts Spacelift. "Drift detection with automatic reconciliation runs on every one of our stacks every thirty minutes, and nobody has asked to turn it off. Spacelift has become part of how we work rather than a tool we think about."

No surprises

His favorite part of working with Spacelift is that nobody needs write access to AWS. "It's partly a security feeling and partly an ease-of-use one. The process that keeps people out of the console is also the easiest way for them to get their work done, so the two don't pull against each other. Mostly it buys us the absence of surprises: every change goes through one stable, guarded path, so we don't expect to ever be caught out by something our infrastructure is doing," he says.

Asked what he'd tell another company considering the move, Abbott returns to the tradeoff Spacelift lets Aurora avoid. "The value of IaC is high enough now that it isn't really optional for a software company. Spacelift raises that value further by adding what raw Terraform doesn't give you: automation, policy enforcement, and auditing. The practical result is that engineers with no write access to AWS at all can still make the infrastructure changes they need, quickly. And we can let them do that because of the guardrails Spacelift gives us," he says.

This could be your story

Empower your platform team with Spacelift.

Liftoff with Spacelift!