

Linus Health unlocks deploy-at-will infrastructure with Spacelift

spacelift

LINUS HEALTH

AWS, Terraform

Stack



Linus Health's deployment times have shrunk from hours of coordination to "deploy-at-will" for standard changes.



A policy-driven model enables developer self-service while preserving governance and consistency.



Spacelift's Terraform provider enables automatic state file creation.

Summary

Boston-based digital health company Linus Health needed an infrastructure orchestration platform that could keep pace with product innovation without sacrificing governance. Using GitHub Actions to manage its complex multi-account AWS setup was restricting visibility and sabotaging efforts to streamline infrastructure management under strictly enforced policies. Adopting Spacelift has enabled the company to reduce deployment times from hours to minutes, achieve consistent policy enforcement, and automate state file creation for each application in each environment/region via Spacelift's Terraform provider.

About Linus Health

Linus Health uses cutting-edge neuroscience, clinical expertise, and artificial intelligence to transform brain health by advancing how cognitive and brain disorders are detected and treated. However, managing multiple AWS accounts with GitHub Actions was creating significant management, security, and networking challenges for its platform team. We spoke to Staff Site Reliability Engineer Joshua Bentler about how Spacelift helped the team to solve them.

The challenge for Linus Health

Before adopting Spacelift, the platform team at Linus Health was experiencing growing friction as its environment expanded. Key challenges included:

- Limited visibility into infrastructure state and drift across multiple AWS accounts
- Large, shared state files that made plans and applies slow and difficult to coordinate
- Complex dependencies between infrastructure stacks
- Inconsistent policy enforcement across environments

As Staff SRE Joshua Bentler explains, the existing toolchain was becoming inadequate as environment complexity grew:



Before Spacelift, our visibility was fragmented. Atlantis logs were difficult to search, and GitHub Actions provided only limited context for historical Terraform runs. Coordinating infrastructure changes across multiple applications was largely manual, and we didn't have a unified view of what was deployed where.

Joshua Bentler
Staff SRE

Large state files compounded the operational overhead. Engineers working on unrelated initiatives still had to coordinate applies, which slowed deployment velocity and introduced unnecessary risk. At the same time, Linus Health needed consistent policy guardrails across environments, especially as the company expanded into more product verticals across accounts and regions.

Why Linus Health chose Spacelift

Linus Health evaluated several options, including Terraform Cloud/Enterprise and env zero. Their must-have criteria were clear:

- Native support for multi-account AWS deployments
- A robust and configurable Terraform module registry
- Policy as code using Open Policy Agent (OPA) and Rego
- GitHub integration for pull request-driven workflows

Nice-to-haves included drift detection and observability integrations such as Datadog.

Spacelift stood out for three reasons:

1. A flexible module registry
2. A comprehensive Terraform provider
3. The ability to run private workers inside their network

Private workers were especially important for maintaining security boundaries while orchestrating infrastructure centrally. Joshua summarizes the decision simply: **"Spacelift provided the governance and flexibility we needed without compromising how we wanted to work."**

Linus Health's Spacelift experience

Linus Health runs Spacelift entirely through the Spacelift Terraform provider. "Our Spacelift configuration is 100% managed via the Spacelift Terraform provider and nothing is defined via ClickOps." This approach ensures their infrastructure orchestration layer follows the same principles as the rest of their infrastructure: It's version-controlled, reviewable, and reproducible.

For example, their iac-common AWS stacks are generated by a bootstrap stack that iterates over directories in an aws/ folder and automatically configures:

- Stack naming conventions
- Backend state configuration
- Environment variables such as AWS account IDs and regions
- Context attachments
- Policy attachments based on environment type

"Adding a new environment is as simple as creating a new directory — the stack is automatically provisioned when that change is merged," says Joshua. This approach eliminates manual configuration and reduces the IaC ceremony required to expand into new product verticals.

Policy as code as a first-class control layer

Policies are central to Linus Health's implementation. They use multiple policy types, including:

- **Approval policies**
Production deployments require asynchronous approval from both engineering management and cloud engineering
- **Git push policies**
Control when stacks trigger based on pull request events
- **Trigger policies**
Manage promotion flow (dev → QA → staging → prod)
- **Plan policies**
Enforce security standards and block dangerous changes
- **Notification policies**
Send Slack alerts for production applies or detected vulnerabilities

Their dependency promotion flow is particularly sophisticated. A custom trigger policy handles Git push events, policy-triggered runs, and targeted deployments differently to avoid unintended cascading changes. This policy-driven model allows them to enable developer self-service while preserving governance and consistency throughout all deployed regions.

Contexts as a scalable configuration model

Contexts are central to how Linus Health manages shared configuration.

They use contexts for:

- Terraform provider configuration with secure secret injection
- Backend state configuration
- Security scanner configuration (including mounted Trivy ignore files)
- Module testing components

They also rely heavily on auto-attach labels.

For example:

- autoattach:space:root automatically attaches their security scanner context
- autoattach:folder:prod attaches production approval policies and deployment notifications

This ensures the right guardrails are applied automatically — without manual intervention.

Managing dependencies without monoliths

Rather than relying on large shared state files, Linus Health has moved toward small, isolated Terraform state files for each application, environment, and region. These are automatically created via the Spacelift Terraform provider. They intentionally keep stacks as self-contained as possible, reducing coupling and making deployments easier to reason about.

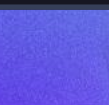
Spacelift's impact on Linus Health

The business impact of adopting the Spacelift platform has been immediate and measurable.

- Deployment time has dropped from hours of coordination to "deploy-at-will" for standard changes.
- Teams no longer need synchronous operations and management involvement for routine deployments.
- Small, isolated state files prevent unrelated changes from blocking each other.

Governance is no longer a manual process. It's enforced as policy as code automatically across environments. **"We've achieved consistent policy enforcement across many AWS accounts and product environments. This has removed the need for operations teams to be synchronously involved in every deployment."**

Spacelift's run history, resource tracking, and integration with notification systems has significantly improved the company's operational visibility and streamlined adherence to compliance obligations. Nearly all of Linus Health's roughly 41,000 deployed resources are tagged with the stack ID that manages them when compatible, making ownership and traceability straightforward.



The Spacelift platform's reliability has also been top-notch, which is critical for such an important piece of our SDLC.

Joshua Bentler
Staff SRE

This could be your story

Empower your platform team with Spacelift.

[Liftoff with Spacelift!](#)