CASE STUDY

FirstCape moves beyond rigid Terraform workflows with Spacelift

500
Company size

60
Engineering team size

Microsoft Azure, Azure Kubernetes Service (AKS), GitHub, GitHub Actions
Stack

Summary

The FirstCape group of wealth and asset management companies needed a consistent, flexible way to manage infrastructure across multiple platforms, teams, and brands. By adopting Spacelift, the platform team standardized infrastructure workflows, reduced manual effort, and dramatically improved run visibility without forcing every team to follow a single rigid model. Today, Spacelift enables developer self-service, clearer governance, and confidence that infrastructure changes are visible, reviewable, and consistent across the business.

Fragmented workflows have been replaced by a single, reliable operating model.

Dramatically improved run visibility has increased reliability significantly.

Eliminating manual applies and ad hoc scripts has lowered risk.

FirstCape is a group of leading wealth management and asset management companies in New Zealand. With around 500 employees and an engineering organization of roughly 60 people, FirstCape supports mission-critical financial systems while continuing to modernize its platform.

We spoke with Lead Cloud Platform Engineer Daniel Edwards about how the team uses Spacelift to manage infrastructure and how it helped them move away from fragmented workflows toward a single, reliable operating model.

The challenge for FirstCape: fragmented workflows and limited visibility

FirstCape's infrastructure challenges were shaped by history.

Because it was formed by bringing multiple businesses together, different parts of the organization relied on a mix of tooling and practices, including manual Terraform applies, shell-script-driven CI/CD pipelines, CircleCI, GitHub Actions, and Terraform Enterprise. Some environments used Terragrunt heavily, with a variety of Terraform versions, whereas newer platforms were built using Terraform Cloud.

This created several problems:

- Inconsistent deployment models depending on where and when infrastructure was created
- Fragile automation, often dependent on custom scripts
- Manual applies from developer laptops, with limited auditability
- Poor run visibility, requiring engineers to switch tools to understand what changed and why

As the tooling landscape grew, even experienced platform engineers found it challenging to quickly see which system managed each part of the infrastructure. This added context switching to day-to-day work and meant new engineers needed more guided onboarding.

Terraform Cloud worked well for newer environments, but migrating older Terragrunt-based configurations would have required significant rework, and the small platform team simply did not have the time for this. Daniel explains that to bring these older configurations into the Terraform cloud environment "we would have also had to completely change the paradigm of how our code was configured because Terraform Cloud doesn't support Terragrunt."

They wanted a clean break from this complexity.

Why FirstCape chose Spacelift

When evaluating options, the team looked closely at:

- HCP Terraform Cloud
- GitHub Actions with custom pipelines
- Azure DevOps with vendor-provided pipelines

Their must-have criteria were clear:

- A transparent, predictable pricing model
- Full support for Terraform, with flexibility for other tools
- Easy integration with existing CI/CD workflows

Flexibility was equally important. As Daniel explains, FirstCape could not assume that every newly onboarded platform would already be using Terraform, and even if they were using Terraform, they might not be using it in the same way.

Spacelift stood out because it allowed the team to:

- Bring different tooling approaches under one orchestration layer
- Standardize workflows without forcing a full rewrite
- Integrate open-source tools and custom checks where needed

They needed to avoid the limitations they had experienced with their previous Terraform provider. As Daniel explains, "we discovered that it worked really well as long as you were on their golden pathway. If you wanted to do something a little different, it either wasn't possible or was difficult. We weren't able to use the open-source versions and plug them in, whereas in Spacelift we can very much incorporate them either as part of our GitHub deployment pipelines or directly in Spacelift."

That flexibility proved critical later, when FirstCape migrated almost its entire estate from Terraform to OpenTofu.

> The process was seamless — it literally just worked. I honestly don't think it would have been as simple on any other platform.
>
> **Daniel Edwards**
> Lead Cloud Platform Engineer

FirstCape's Spacelift experience

Managing Spacelift with infrastructure as code

FirstCape manages its entire Spacelift organization using infrastructure as code. A dedicated "Spacelift management" repository defines stacks, contexts, and permissions, enabling controlled self-service through pull requests with reviews from code owners — without granting broad admin access.

Management repo example structure:

```
spacelift-management-repo
├── spacelift
│   ├── modules
│   │   ├── stack-basic
│   │   │   ├── main.tf
│   │   │   └── variables.tf
│   │   └── stack-advanced
│   │       ├── main.tf
│   │       ├── variables.tf
│   │       └── outputs.tf
│   ├── policies
│   │   ├── approvals.rego
│   │   └── github-pr-notifications.rego
│   ├── main.tf
│   ├── org-config.tf
│   ├── outputs.tf
│   ├── policies.tf
│   ├── spaces.yaml
│   ├── stacks-engineering.yaml
│   ├── stacks-platform.yaml
│   ├── worker-pools.tf
│   ├── variables.tf
│   └── README.md
├── worker-pool
│   ├── main.tf
│   ├── variables.tf
│   ├── outputs.tf
│   └── README.md
├── .github
│   └── workflows
│       └── ci-checks.yml
├── .gitignore
├── CODEOWNERS
└── README.md
```

For application teams, infrastructure typically lives alongside application code. Each repository contains an `infra` directory, split by environment (for example, `dev` and `prod`), with each environment mapped to its own Spacelift stack.

App repo example structure:

```
app-code-repo
├── infra
│   ├── dev
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   ├── variables.tf
│   │   └── dev.tfvars
│   ├── prod
│   │   ├── main.tf
│   │   ├── outputs.tf
│   │   ├── variables.tf
│   │   └── prod.tfvars
│   └── src
│       └── index.ts
├── .github
│   └── workflows
│       ├── build.yml
│       └── ci-checks.yml
├── .gitignore
├── CODEOWNERS
├── package.json
└── README.md
```

This approach:

- Keeps teams in control of their own infrastructure
- Reduces cross-team dependencies
- Minimizes context switching between repositories and tools

Improving run visibility where engineers already work

One of the biggest wins has been run visibility.

FirstCape uses Spacelift notification policies to send plan and apply details directly back to GitHub pull requests as checks and comments. Engineers can review changes, understand impact, and confidently merge (often with auto-apply enabled) without leaving GitHub.

As Daniel says, "when the information is visible and transparent in the pull request, reliability improves hugely. People know what's changing and who changed it."

This tight feedback loop shortens development cycles and ensures Spacelift is only opened when deeper investigation is needed.

Contexts, self-service, and controlled flexibility

Contexts are used extensively, particularly for Terraform variables and shared configuration. Auto-attach labels play a key role in enabling self-service.

For example, when the platform team provisions a new Azure subscription, they automatically create a corresponding Spacelift context with the relevant details and labels. Application teams simply attach the label to their stack and deploy. They don't need any extra assistance from the platform team.

"We value being able to give teams that level of autonomy without losing control," says Daniel.

This pattern delivers governed developer self-service, aligned with FirstCape's existing Terraform-first approach.

Spacelift's impact on FirstCape

With Spacelift in place, FirstCape has achieved meaningful improvements across reliability, consistency, and team experience:

- Clear, consistent infrastructure workflows across platforms and teams
- Dramatically improved run visibility, directly in GitHub
- Reduced manual applies and ad hoc scripts, lowering risk
- Simpler onboarding, with fewer edge cases and exceptions
- A smooth migration to OpenTofu, completed with minimal disruption

Perhaps most importantly, the platform team now has confidence that Spacelift can adapt to whatever comes next — whether that's onboarding new businesses, integrating new tools, or evolving infrastructure practices over time.

As Daniel puts it:

> Spacelift initially felt like sitting in the driver's seat of a Formula 1 car when I'd only ever driven a Corolla before. Before long, we were up to speed and shipping with confidence.
>
> **Daniel Edwards**
> Lead Cloud Platform Engineer