CASE STUDY

How Spacelift enables Magnolia DXP to manage hundreds of customer clusters with a single DevOps engineer

Week → 1 day

Customer onboarding time pre- & post-Spacelift

5 → 1

Engineers required per customer onboarding pre- & post-Spacelift

100s

Customer Kubernetes clusters managed in Spacelift

Summary

Magnolia, a leading composable digital experience platform, expanded from a content management system into a managed platform-as-a-service offering and found that its manual GitLab Terraform pipelines couldn't keep up. Customer onboardings took up to a week and required four to five engineers. Since adopting Spacelift, one DevOps engineer handles the full onboarding in a day, and the team maintains hundreds of customer Kubernetes clusters with consistent standards and drift detection across every one.

Spacelift Blueprints gave Magnolia DXP a cloud-agnostic provisioning framework that works identically across AWS, Azure, Google Cloud, and on-premises deployments.

Drift detection and automated runs let the platform team push changes across hundreds of clusters with confidence.

When critical incidents hit, Spacelift's batched rollout capability cut incident response from a week to hours.

Magnolia is a leading composable digital experience platform (DXP). Based in Switzerland, it helps enterprises build fully integrated customer experiences and speed up digital delivery by unifying their tech stacks. Several years ago, Magnolia expanded beyond its roots as a content management system (CMS) and launched a platform-as-a-service (PaaS) offering, hosting Magnolia DXP on behalf of customers on whichever cloud each customer required. As their customer base grew, the infrastructure became more complex for the platform team to manage.

We spoke to Sebastian Klingberg, Team Lead DevOps Solution Architect at Magnolia, about why he chose Spacelift, how the team rebuilt their provisioning around it, and what happened when they needed it most.

The challenge for Magnolia DXP

Sebastian sits on Magnolia's platform engineering team, which operates the infrastructure underpinning every hosted customer environment. Each new customer means a full Kubernetes cluster, cloud account, storage buckets, networking components, and more, on whichever cloud provider that customer prefers. That could be AWS, Azure, Google Cloud, or on-premises. The team ran Terraform and Terragrunt through GitLab pipelines that technically worked but required manual input for every deployment and were slow to run across a growing fleet.

"It came to a point where we started realizing, 'Hey, the pipelines are working, but this is a very tedious, time-consuming job,'" recalls Sebastian.

As the customer count grew, maintaining consistency across an expanding fleet with scoped, on-demand pipeline runs became untenable. Sebastian started researching managed Terraform options and came across Spacelift at KubeCon Europe 2024. He evaluated it against HashiCorp Terraform Cloud, which lacked Terragrunt support at the time. Not only did Spacelift support Terragrunt, its multicloud design fit Magnolia's requirement to run on any cloud provider without a cloud-specific playbook.

Magnolia's Spacelift experience

The migration to Spacelift was, in Sebastian's words, "surprisingly straightforward," and it became an opportunity to rethink how provisioning was structured. Using Spacelift Blueprints, the team built a cloud-agnostic framework where a single configuration file drives the full onboarding. Cloud-specific differences are handled underneath the abstraction.

The vision driving the redesign had been there from the start: "We want to come to a point where even a non-technical person can fill out an onboarding form with this much storage, this many nodes, this type of node, and so on. That was always the goal, and with Blueprints and our setup, we are very close to it from a technical perspective," Sebastian says. That form-driven approach extends beyond the Blueprint layer itself. Magnolia also makes heavy use of the Spacelift API to automate everything downstream of the form submission: Stacks, spaces, and contexts are created programmatically for each customer based directly on the data captured in the onboarding form, so no manual setup is required in the Spacelift UI. The team has found the API flexible enough to meet nearly all of their automation needs, enabling the onboarding workflow to scale cleanly as customer count grows, without a corresponding growth in operational overhead.

Magnolia also uses Spacelift policies to enforce a four-eyes principle, requiring two approvals before any stack can be destroyed. That governance layer pays off at audit time. "If the auditor asks, 'Okay, what is your process? How do you, for example, destroy things and ensure everything's done?,' it's much easier to show them now," Sebastian says.

Spacelift's impact on Magnolia DXP

Onboarding was the most immediate change for Magnolia after their Spacelift migration. It now takes just a single engineer to onboard a customer — a task that previously required four or five people. Sebastian estimates that deployment times have fallen "from a week to a day or less. And that's just one person. That's the most important part here."

The gains extend well beyond onboarding speed. Magnolia now manages hundreds of customer Kubernetes clusters in Spacelift. Drift detection keeps every cluster in a known good state, with frequent automated runs replacing the scoped, on-demand executions of the old approach. Changes roll out in batches, starting with development clusters, moving to satellite clusters, then production. Manual changes to cloud environments are effectively eliminated.

"You can clearly see what will change and especially also what will not change as we are keeping a better track of everything, and it's all aligned and standardized," Sebastian says.

"Spacelift definitely saved us on two occasions"

The standardization Magnolia has introduced has been tested in production twice, and both times, Spacelift's speed proved the difference.

The first incident came when a bug in the Kubernetes trust relationship configuration caused clusters across multiple customers to disconnect overnight. With Spacelift, the team identified the responsible deployment, fixed the code, and rolled the fix out across all affected customers before the situation cascaded. "With the old pipeline spec, we would definitely have lost more clusters, just due to the sheer amount of time it would take per customer," Sebastian says. "To this day, I say that SpaceLift was our absolute lifesaver that day."

The second crisis arose from a critical CVE in the Ingress controller and a Linux kernel vulnerability known as Dirty Frag. Node rotation across all customer clusters was required. The team batched the rollout, tested each group, waited for reactions, and expanded until every cluster was patched. "We were able to mitigate and react within hours, whereas before, it would have taken us, I don't know, a week," Sebastian says. "Speed is the essence. Speed is it."

On top of the peace of mind Spacelift provides, Sebastian added that his developers actively enjoy using Spacelift and tell him regularly that it is one of their favorite tools in the stack. "They tell me that Spacelift is awesome!"

For any engineering leaders evaluating their infrastructure tooling, Sebastian offers this insight: "Choosing Spacelift was one of the best tool decisions I've ever made, and I'm very proud I made it. It also brought us better engineering principles, and without those, platform management at the scale we have right now would not be possible."