plural

WHITEPAPER

Accelerate Kubernetes Adoption with Plural Continuous Deployment

By: Michael Guarino, Founder and CTO at Plural

Over the last two years, our team has deployed production-ready Kubernetes clusters at scale for hundreds of customers. While the deployment and management process varies by customer, the one constant has been that Kubernetes is extremely challenging. For most organizations, it takes upwards of two to three months to fully ramp into Kubernetes in production.

This process becomes even more complicated when you consider that the talent marketplace for Kubernetes expertise is incredibly thin and the ecosystem of associated tools is sprawling.

All of this has created a level of intimidation around using Kubernetes that is regrettable. At Plural we believe that the way to cut through the noise is by utilizing a unified platform to manage Kubernetes like cattle rather than pets, with true self-serviceability for organizations regardless of their existing Kubernetes expertise.

In particular, our team has seen three primary points of friction in adopting Kubernetes:

- Provisioning and managing Kubernetes clusters through their lifecycle is difficult, especially at scale
- Maintaining a Kubernetes YAML codebase for application development is error-prone and frequently untestable
- Existing tooling is powerful, but lacks enterprise-grade features and still requires integration with the rest of the Kubernetes stack

Challenges

Kubernetes Provisioning is Hard

Even in a world with EKS, GKE, and AKS providing managed Kubernetes provisioning, maintaining a self-service Kubernetes provisioning system is still quite challenging. Toolchains like Terraform and Pulumi can create a small cluster fleet but don't provide a repeatable API to provision Kubernetes at scale. Additionally, the upgrade flow around Kubernetes is fraught with dragons, in particular:

- Most control planes require a full cluster restart to apply a new Kubernetes version to each worker node, which is a delicate process that can diverge a cluster
- Deprecated API versions can cause significant downtime, even a sitewide outage like what happened at Reddit earlier this year

Kubernetes Codebases are Thorny

Kubernetes specifies a rich, extensive REST API for all its functionality, which is often declared using YAML. While that's relatively user-friendly, larger application codebases interacting with that YAML spec can frequently balloon into thousands of lines of code. There have been numerous attempts to moderate this bloat, using templating with tools like Helm or overlays with Kustomize. We believe all of these have significant drawbacks, which impair an organization's ability to adopt Kubernetes, in particular:

- Lack of the ability to reuse code naturally (there's no package manager for YAML)
- Lack of the ability to test your YAML codebases locally or in CI (preventing common engineering practices to detect regressions quickly)

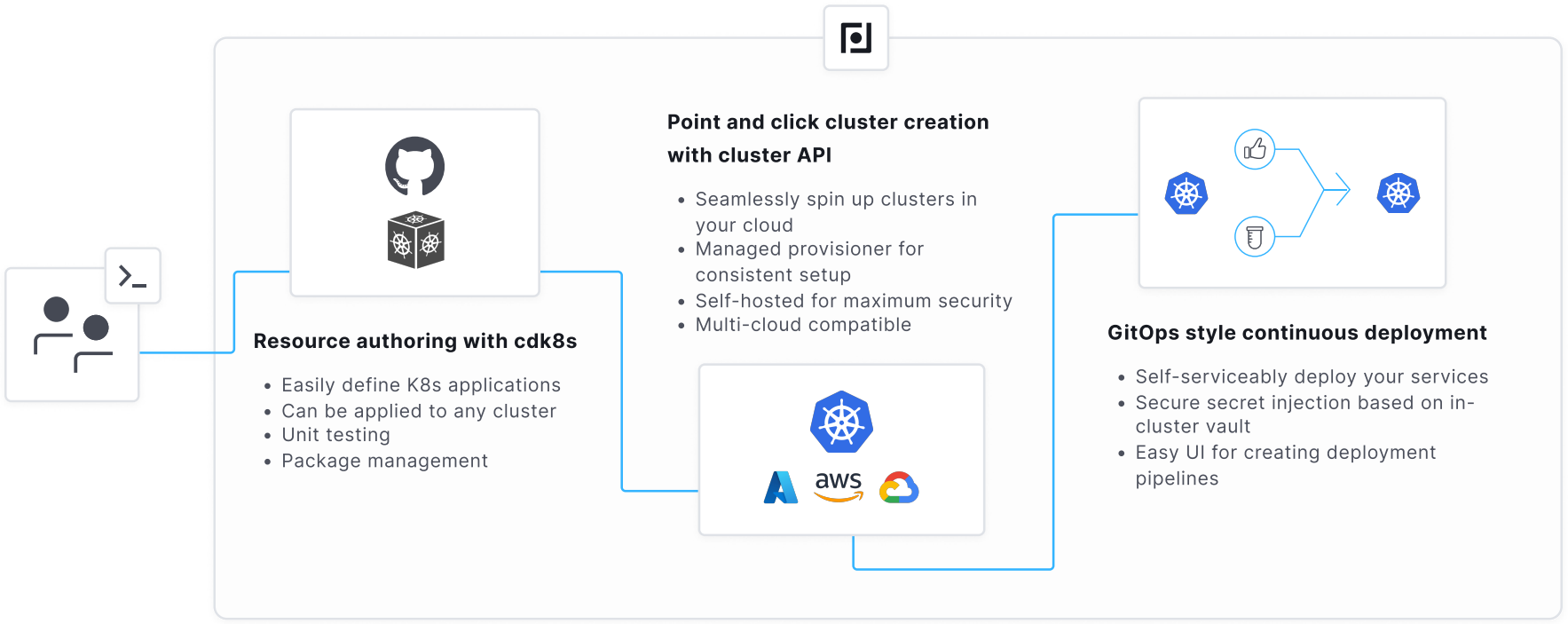
Deployment Pipelines on Kubernetes are Poorly Supported

Kubernetes has a rich ecosystem of CD tooling, with the likes of flux and Argo-CD, but most are built primarily for simple single-cluster deployment use cases out of a unique git repository. To use any of them, you still need lower-level Kubernetes management expertise to provision and administer the cluster to which they deploy. From our experience, to create a mature Kubernetes workflow, you need two major components:

- The ability to rapidly create staged deployments, from dev → staging → production with no effort
- Integration with a Kubernetes provisioning system, so you don't have to manage the complexities of the Kubernetes auth chain to start deploying code

The Solution

These challenges can be solved by tackling each problem within a broader platform, merging the benefits of Cluster API, modern Kubernetes SDKs like CDK8s, and a robust GitOps pipeline engine.



Manage Kubernetes At Scale with Managed Cluster API

The solution for managing cluster provisioning is using Cluster API (CAPI). CAPI provides the dynamic API that enables your Kubernetes usage to scale fluidly for use cases like testing infrastructure or rapidly onboarding a new team. It also provides a robust reconciliation loop to handle complex Kubernetes upgrades in a fault-tolerant way that neither Terraform nor Pulumi's CLI-based, remote-state-driven toolchain is purpose-built to handle.

We're investing heavily into CAPI to provide robust Kubernetes lifecycle management for all Plural users, and believe it is also incredibly useful for internal Kubernetes use cases of any scale.

A Path To Healthier Kubernetes Development with Supercharged CDK8s

We've decided to provide first-order support for CDK8s to solve this problem in one fell swoop. CDK8s allows you to author Kubernetes resources in common programming languages like javascript, golang, or Python. This allows you to strap your Kubernetes code to any existing CI solution you might have and provide a robust testing ecosystem around your K8s code before releasing it into running clusters. You can also seamlessly reuse code using the language's own package management tooling, such as npm for Javascript or PIP for Python.

Additionally, we'll leverage the Plural community to build an ecosystem of starter packages to allow one-command setup for common frameworks like ExpressJS, Django, Rails, spring boot, and more. This should let you get started with your Kubernetes infrastructure without having to deeply familiarize yourself with the Kubernetes API and still have a robust, testable, reusable development workflow.

Robust Kubernetes Release Management with Plural Pipelines

We intend to combine the CAPI-powered Kubernetes management infrastructure defining your Kubernetes fleet with tooling built on top of CDK8s to allow for a seamless, self-serviceable platform for managing complex Kubernetes deployment pipelines. It will provide all-powerful features like manual, automatic, or test-driven promotions, deployment windows, and more. The flexibility of the infrastructure will also allow us to provide more plastic deployment constructs like PR-based environments.

The workflow should be entirely API-driven, so you'll be able to integrate Plural CD with any existing CI solution like GitHub actions, Jenkins, or CircleCI with off-the-shell tooling or the `plural` CLI.

Conclusion

We believe a holistic Kubernetes platform like Plural CD can compress the time to Kubernetes adoption from months to a matter of days. It can enable organizations that have essentially no Kubernetes experience to get going in a manner that will scale with virtually any business use case. This product is a natural extension of our current offering, born out of the battle testing of deploying clusters for hundreds of companies as part of the Plural platform, and our experience deploying code for the best engineering organizations in the world, like Amazon, Facebook, and Palantir. This knowledge should not be reserved for an elite few but should provide value to the industry at large, and we're excited to let Plural CD do just that.

About Plural

At Plural we are building a flexible, scalable solution to application delivery that gives you the autonomy you need to deploy, manage and scale open-source applications. Plural makes it easy for companies to access and deploy the best open-source solutions. Our vision is to establish a more direct relationship between enterprise developers and open-source maintainers, without the cost of cloud-managed services standing between them. For more information reach out to us at plural.sh or follow us on Twitter @plural_sh.