Open Source Kubernetes Clusters as a Service

Kris Nóva

What is the most inventive or innovative thing you've done? It doesn't have to be something that's patented. It could be a process change, product idea, a new metric or customer facing interface – something that was your idea. It cannot be anything your current or previous employer would deem confidential information. Please provide us with context to understand the invention/innovation. What problem were you seeking to solve? Why was it important? What was the result? Why or how did it make a difference and change things?

In October 2016 I opened up an issue [1] in the Kubernetes Kops issue tracker proposing an HTTP(s) wrapper for a soon-to-be implemented API for the Kops codebase. The inspiration originally came as I was managing over 50 Kubernetes 1.04 clusters in AWS EC2 for an engineering organization. At the time, I had shipped a statically linked binary to the org and was manually configuring IAM roles/policies for the end users. As the team grew, so did my workload as the only SRE. It soon became clear that a CLI tool was not going to scale with the demand of the team. We needed a service that the team could interact with and Kubernetes was just the model that we needed to manage Kubernetes itself. I suspected that building an infrastructure management service for the team would offload a lot of my daily chores such that I could monitor it, and program against it. The Kops code base was capable of mutating the AWS infrastructure (compute, networking, and storage) that we would need, as well as installing the Kubernetes control plane on the instances. The only missing component was the service and a client; and of course the right API abstraction and support from the open source community.

I began to contribute to Kops [2], and quickly became a SIG-AWS lead and a Kops maintainer. All the while, attempting to push the command line tool towards a server/client model. At the time CRDs and the operator pattern weren't as fleshed out as they are today. Imagining a new Kubernetes object that represented infrastructure was much more controversial than it seems today. I joined SIG-Cluster-Lifecycle and began to circulate my thoughts on having a declarative service that would be tasked with installing, upgrading, scaling, and managing Kubernetes infrastructure over time. Naturally questions and concerns began to surface.

Where would the controller live? Would it run in the same cluster it was managing? How do you create a meaningful abstraction across clouds? What about having multiple server implementations that could interpret the same configuration in different ways?

There was a lot to answer, and perhaps in my naivety I assumed a lot of the friction was mere implementation detail. I didn't concern myself too terribly much with the details, which ultimately led to my confidence in proposing projects later.

As time progressed it became clear that the other prominent maintainers of Kops had little or no interest in running the Kops internal packages as a service under a server/client model. Furthermore the Kubeadm [3] project was on a clear trajectory to become the official component installation tool for Kubernetes. While Kops showed little interest in adopting Kubeadm, and instead favored a competing tool that was created under Kops called Nodeup [4].

Thus, I started the Kubicorn project. [5][6] This project aimed at solving Kops´ dependencies on Route 53 (Public DNS) and Nodeup by using cluster configuration injection on the Nodes and Kubeadm to start the Kubernetes components. This offered a dramatic security enhancement over Kops, as well as the freedom to re-imagine the shape of the Kubernetes cluster API surface from scratch. The primary lessons were realized in infrastructure reconciliation, and cloud provider specific configuration. These lessons later manifested themselves in my book Cloud Native Infrastructure [7] which drew on many of the lessons learned in the Kubicorn experiment. Cloud Native Infrastructure also introduced a new problem to the Kubernetes infrastructure as a service model, which was coined "*The Bootstrap Problem*". This problem introduced the problem of having a "maiden" cluster in which the cluster management toolchain could run. Comparing the first cluster to the first compilers, we realized that we needed an avenue in which a team could stage a cluster that could manage the service that would be responsible for managing other clusters. There was only one problem: where did the first cluster come from?

After several years of prototyping and working in this space, myself and some like minded engineers from Google started to flesh out what would later be called "Cluster API". [8] A name that to this day, I do not agree with. After consolidating thoughts from Kops, Kubeadm, Kubicorn, and Kube-up we began to cross reference our findings with other components of Kubernetes such as CNI and CSI. [9][10]

I later drafted the original proposal of Cluster API, which to my chagrin was frowned upon by the majority of the Kubernetes community including my long time mentor and close friend Joe Beda, and the lead of SIG Architecture, Brian Grant. [11]

In my eyes Cluster API was the final iteration of managing Kubernetes clusters as a service. Looking back on the original HTTP(s) proposal in 2016 and seeing Cluster API as it stands today is a testament to the iterative nature of Kubernetes cluster management myself and the community had to go through in order to get where we are today. Cluster API addressed all of the concerns of Kops, as well as the bootstrap problem we discovered with Kubicorn and Cloud Native Infrastructure. It embraces Kubernetes control loops and declarative infrastructure, as well as solves for the cloud specific implementation components of the API surface. [12] Furthermore it embraced the Kubeadm toolchain. In my eyes Cluster API is a critical aspect of the Kubernetes ecosystem that unfortunately joined the party 7 years too late.

1. Nóva, Kris.. "Kops API · Issue #633 · Kubernetes/Kops."*GitHub*, Kris Nóva, 11 Oct. 2016, github.com/kubernetes/kops/issues/633#issue-182315914.
2. Nóva, Kris. "Kubernetes/Kops."*GitHub*, Kris Nóva, 2016–021, github.com/kubernetes/kops/commits?author=kris-nova.
3. VMware . "Kubeadm Is Now GA and Stable."*Open Source Blog*, 6 Dec. 2018, blogs.vmware.com/opensource/2018/12/06/kubeadm-general-availability-stable.
4. Kops Authors. "Kubernetes/Kops."*GitHub*, 2014–021, github.com/kubernetes/kops/tree/master/nodeup.