The Kubernetes Journey from Research to Production at Presslabs

by CALIN DON
CTO and Co-founder Presslabs
Published on January 18, 2018

At Presslabs we do managed WordPress hosting for which we have created our own stack dressed with a customer dashboard written in Django.

The dashboard started back in 2011 as a plain WordPress plugin to allow customers to pay their monthly subscription. A year later, we've rewritten every internal service using Python, exposing the public part through a Django app. This was a significant change, as we needed to host it as well on its own stack and add a development process on top. Deployments started with a "written by hand" approach followed by a mix of bash scripts and synctool evolving to automated deployments with Ansible and Drone.

A second rewrite started in 2014 and it involved splitting the monolithic app into microservices, each one with its own API. At first, they ran on virtual or bare-metal machines. Then another significant shift happened in 2015 when we standardized the microservices to run in Docker containers on bare metal, mainly for development purposes. It was a successful rewrite, so we open-sourced what we considered meaningful also for others that were looking for this kind of solutions.

The increasing complexity of deployments and interlinking between microservice components, the intricacy of setting up a staging environment, the neverending maintenance problems such as upgrading the underlying OS for the machines running the services—these were just some of the most pressing problems we've faced at the time. Basically, we had a fear of breaking things, which resulted in slower and less often deployments of new features.

The Switch

In 2016 we started watching container orchestration tools; a year later we decided to experiment with it to see if we could solve at least some of the above challenges. So here we are, sharing key points we learned while moving to Kubernetes, hoping to save you some time if you plan or are already working on a similar switch.

At first, we tested both Docker Swarm and Kubernetes. Although the Docker Swarm was friendlier at the beginning, as the install was really easy and fast, it didn't feel production-ready. For instance, it would "lose" services from DNS, also some nodes' CPU were spiking even without overload.

Kubernetes performed better in production and things generally working as advertised, but it required higher install efforts. With the 1.6 version, Kubernetes started showing a certain maturity and a more generous feature set (like RBAC, Dynamic Volume Provisioning, Applications API), which made it fit for production, so we decided to go through the pain of installing and configuring it. In this journey, besides the Kubernetes documentation, we found two guides which were particularly useful: Kubernetes the hard way and Kubernetes the not so hard way with Ansible.

Kubernetes became fun only when we managed to get it up and running. On September 2017 we started working on the migration of our internal services, to make sure we get all the learnings and best practices before replicating it for our hosting stack as well.

If we should summarize our experience to only three key lessons, here's what we'd like to share with you:

1. Deploying and maintaining your own cluster is unnecessarily hard.

If there are no hard requirements to deploy and operate your own machines, go with a managed Kubernetes provider. We choose among AWS EKS, Azure Container Service and

Google Kubernetes Engine. We chose GKE because at that time the other providers were early on their managed service offering and the only one feeling mature enough (cli and 3rd party tools were available).

Moreover, even though the recent beta version of Kubeadm can help you setup a minimum viable cluster, there are certain shortcomings to using it. Currently, it lacks support for HA master and is still considered beta. Nonetheless, we hope you will feel inspired enough to deploy a lab cluster and face the internals complexity. If you're small like us, your development team will thank you for using a managed service, as maintaining a cluster will require at least a dedicated engineer for this.

2. Using Helm for deploying services to Kubernetes is a good idea.

As we were lacking a standard deployment method, which only made room for laborious extra work and eventual errors, our hopes for migrating to Kubernetes were to also gain some operational speed.

We need reproducible and version-controlled deployments so we explored Kubectl, Terraform and Helm. We've cut Kubectl from the list really early since it lacked support templating and code reuse. We realized that we were going to add a lot of custom code around it to make it behave as needed.

Terraform proved to be more appropriate to do infrastructure provisioning and support for Kubernetes was always lagging. For example, there was still no support for Kubernetes deployments at the time of writing.

3. Managing deployment secrets is painful.

Now it's time to complain about Helm because it's missing an integrated secret management. Luckily, the Helm-Secrets plugin saves the day by integrating helm with SOPS, which is a tool for managing and keeping secrets in version control repositories. The only missing piece was integration with Google Cloud Key Management Service to which we contributed.

The benefits of using Kubernetes

Now that our dashboard and its services have been completely migrated we are really happy with the results. To sum it up, there are 3 key points that made using Kubernetes a total win for us:

1. **Easier deployment.** This has directly influenced our production capabilities, as it has allowed us to speed up on our development cycles, allowing us to perform a few deploys per day rather than per week.
2. **Better availability.** Kubernetes was truly reliable so far; we are no longer afraid of breaking the build and zero deployments downtime is the norm.
3. **Simpler maintenance.** We don't need to schedule downtime for server upgrades anymore, as the machines it runs on are managed by our Cloud provider.

Using Kubernetes also impacts the overall atmosphere in our team as we have more time to research, plan and develop other features that are critical to the success of our product.

Moreover, we have achieved a uniformization of the underlying infrastructure for our customer dashboard, together with automation in managing and deploying it.

In a wider perspective, we're eager to keep the pace with current technology developments. We strongly believe that, at least for the moment, Kubernetes is the solution to-go if you are looking for solutions in orchestrating containers and microservices. When all major Cloud providers opt for managed services on Kubernetes, you might want to listen up and use them, in the same way as we ourselves provide a managed service for hosting WordPress sites.

Probably the most important point is that our enterprise clientele will soon be able to benefit from a more scalable WordPress hosting stack that will each run directly on their Kubernetes cluster.

If you're looking for more information regarding our migration to Kubernetes drop us a line.

Blog