Rice University Explores Container Mangement

Since this is our first foray into running containerized applications, we wanted to explore as many options as we could to find the right fit for our needs, available resources, and comfort level. We want something that we can play with and develop on-premise but will allow for us to migrate into the cloud easily should the opportunity become available.

To that end, we came up with an initial list of possible options: Docker Swarm, Rancher, Kubernetes, OpenShift Origin, and Tectonic. Tectonic was a non-starter; their licensing and installation requirements were not a good fit. Docker Swarm, while being the standard currently used by TIER, was also ruled out due to the lack of a UI for management and the inability to scale up in a non-disruptive manner. Vanilla Kubernetes appeared to require more configuration than the others to get up and running. That left us with Rancher and OpenShift Origin.

We went with Rancher first for its ability to support multiple isolated container orchestration frameworks, including Kubernetes and its own native orchestrator, Cattle. (Besides, this is Texas; ranching is in our blood.) OpenShift Origin was still using Kubernetes v1.6, and there were concerns about how long it would take RedHat (our campus OS distribution of choice) to release updated versions. Rancher also supports easily adding hosts or nodes regardless of the orchestrator via a web front-end that natively supports AWS EC2 and Digital Ocean.

The Cattle orchestrator was initially more attractive because it supports the ability to use the Docker Compose file format and everything can be done within the UI. However we were dissuaded by the lack of auto-scaling and the fact that Secrets support is still experimental. On top of that, it seemed clear that the Rancher project was moving on; Rancher 2.0 was just announced, and it is using Kubernetes as its default orchestrator.

Eager to get something going, we tried setting up the Rancher 2.0 Tech Preview. It was immediately clear that the preview was really just a preview. It's lacking in very basic functionality (authentication, anyone?) So we dropped back down to Rancher 1.6 using Kubernetes as the default orchestrator in the hopes that as 2.0 becomes more viable the transition will be smoother.

We tested and enjoyed working with Kubernetes. We tried out running containers of a few basic applications and explored the auto-scaling features. Things were going great until we hit a wall trying to build an image. We spent some time troubleshooting the issue. It turns out the development VM infrastructure we were given to work with had some I/O constraints that didn't play at all nicely with the overlay2 filesystem. Reads and writes were too slow to perform part of the Dockerfile build for the application we were trying to containerize. We made note of the issue and continued our testing with less I/O intensive applications for now.

After a while, it started to become clear that while Kubernetes was enjoyable to work with, Kubernetes under Rancher 1.6 is less so. We found that in our setup we were having to drop down into the default Kubernetes UI from within the Rancher UI to get tasks done more often than not. Most irritating was the load balancing configuration, which necessitated backing out of the Kubernetes UI to create the config with the Rancher UI and then dropping down into the Kubernetes UI again to configure the nodes. We have hopes that with the focus on Kubernetes for Rancher 2.0 these problems will be mitigated.

So now it's time to try out OpenShift Origin. We're currently waiting for more resources to spin up a dev environment to play with it and see how it compares. Stay tuned!