BASICS OF KUBERNETES

Basics of Kubernetes

Challenges

Containers provide a great way to package, ship, and run applications - that is the Docker motto.

The developer experience has been boosted tremendously thanks to containers. Containers, and Docker specifically, have empowered developers with ease of building container images, simplicity of sharing images via registries, and providing a powerful user experience to manage containers.

However, managing containers at scale and designing a distributed application based on microservices' principles may be challenging.

A smart first step is deciding on a continuous integration/continuous delivery (CI/CD) pipeline to build, test and verify container images. Tools such as Spinnaker, Jenkins and Helm can be helpful to use, among other possible tools. This will help with the challenges of a dynamic environment.

Then, you need a cluster of machines acting as your base infrastructure on which to run your containers. You also need a system to launch your containers, and watch over them when things fail and replace as required. Rolling updates and easy rollbacks of containers is an important feature, and eventually tear down the resource when no longer needed.

All of these actions require flexible, scalable, and easy-to-use network and storage. As containers are launched on any worker node, the network must join the resource to other containers, while still keeping the traffic secure from others. We also need a storage structure which provides and keeps or recycles storage in a seamless manner.

When Kubernetes answers these concerns, one of the biggest challenges to adoption is the applications themselves, running inside the container. They need to be written, or re-written, to be truly transient. A good question to ponder: If you were to deploy Chaos Monkey, which could terminate *any* containers at any time, would your customers notice?