Why Helm

By Mitchell Hockenberry

Introduction:

This document will lay out why it is my opinion that EES should be mindful of the value of helm as a tool for the deployment of Kubernetes resources. It is my hope to explain exactly what helm does, the value it can provide, and explain its potential issues in the EES environment. I want to preface this by saying that I fully want to continue to use helm, but not in the way that initial tests were done on it. This opinion is inspired by an article written by Bartlomej Antoniak, which can be found [here](https://medium.com/virtuslab/think-twice-before-using-helm-25fbb18bc822).

What is Helm:

Helm is package manager for Kubernetes, it works by pulling the helm repository specified, templating the content of the values file and any commands listed in the calling helm command onto several yaml files and then ‘essentially’ running kubectl apply on all the resulting templates. It should be noted that yaml files that create deployments in turn reference docker images and pull/run them with the desired specifications. There are some unknowns with how exactly helm interacts with the cluster because it requires a tiller service account to run, but it is easy to demonstrate that helm is reliant on the state of kubeconfig to apply its changes. Points about tiller are made in Bartlomej’s article, highlighting security concerns.

Helm has a stable repository that contains charts which are considered stable, including rancher. Helm’s mission is to allow for quick deploy of those services to a cluster out of the box. The problem is that because of the variation within clusters this does not work on the first try, and time must be spent making the chart match EES specifications, or the environment must be changed to accommodate the needs of the chart. This is where templating comes into play, options can be written by the author to allow several variations on existing configurations to be applied, however if your desired configuration is not prepared you are out of luck. Additionally, as these projects are open source troubleshooting can be difficult and support nonexistent.

Current applications of Helm:

There are functionally two ways to use helm. You can either create, post, and deploy your own helm charts, or use someone else’s. Creating your own has the advantage of being able to write exactly the specifications you want, while using someone else’s has the advantage of being plug and play. EES at this stage is looking at doing both. There do not currently exist helm charts for the Deluxe owned services in rancher, but there are existing charts for services such as kong, mongodb, and postgresql. As such the current plan is to write charts for the services which don’t have them and use public charts for the services which do.

My objection to this process is twofold. The first is with public charts, which I, alongside Bartlomej, find to be rarely plug and play. They require additional kubectl commands to properly configure, which defeats the purpose of a helm chart. At the very least it would make sense for EES to fork these services into new branches specifically configured for the Deluxe environment, with the proper configurations enabled.

My second objection focuses on the creation of Deluxe exclusive Helm charts. At this point the proposed methodology includes transferring the system (automatically or manually) from rancher 1.6 to rancher 2.2 and using the resulting yaml files as the basis of the new helm chart. This is not inherently a bad idea, but it does suffer from overcomplication, when it would be as effective to use kubectl apply -f on these yaml files and never mention helm. As such it is starting to look like helm is less valuable than originally expected.

Why not Helm:

The most compelling argument against any use of helm is Bartlomej’s, which has to do with security. I make no pretense of understanding the technical details, but the overview is that helm’s connection is another potential security risk, and because tiller operates with super-user privileges, it is one which should not be ignored. It is not my role to evaluate the security of the tool, but I would like to make sure the right person does investigate this concern. Additionally tiller is supposedly non-HA, which to some extent devalues running rancher HA.

For the EES specific objections I feel like it will be more efficient for deployments to be created as yaml files and added to Deluxe specific repositories than to attempt to cobble together the work of a variety of unknown internet authors. That is not to say that Helm should not be used as a Deluxe package manager, but rather that it should be as an after thought rather than as a goal. Helm provides plenty of value as a templating tool, and as a package manager, but I highly suggest that all packages we manage we write ourselves, and that we consider kubectl our deployment tool, with helm only existing to automate it.

Conclusion:

There is need for review with the security concerns of helm, and if found satisfactory it should be used as a templating tool/package manager. Additionally, this opinion is subject to change with new information in the Jenkins migration.