Installing Helm in Google Kubernetes Engine (GKE)

Jonathan Campos Follow

Aug 13, 2018 · 8 min read

When I first started to really get into Kubernetes I would go find the docker image to various necessary programs and just create a template file around their code. This was easy enough and seemed like a good option. *Depending on your needs, this is a fine direction to go.*

However as time went on I would keep reading about Helm and see that some of the applications I was creating templates for already had extremely well created templates in the form of Helm Charts. With this realization — and me being a lazy developer that wants to increase my development speed — it was time to learn Helm!

Helm Does This To Your Kubernetes Development

What Is Helm? What Will It Do For Me?

First, in case you haven't read a ton of articles about Helm already, here is all you need to know about Helm. **Helm is a Package Manager for Kubernetes.** If you come from any languages (and like word puzzles) here is something for you to relate to.

> ***NodeJS*** *is to* ***Kubernetes*** *as* ***NPM*** *is to* ***HELM!***
>
> ***Ruby*** *is to* ***Kubernetes*** *as* ***Gems*** *are to* ***HELM!***
>
> ***Swift*** *is to* ***Kubernetes*** *as* ***CocoaPods*** *are to* ***HELM!***
>
> ***Java*** *is to* ***Kubernetes*** *as* ***Maven*** *is to* ***HELM!***

I could keep going, and the comparison isn't perfect but you should get the idea. Basically Helm takes the effort out of making sharable, packaged, template files that can easily be dropped into other Kubernetes Clusters without a lot of effort. This is a big win for you and your time. Some more terminology that you'll hear that you need to know:

> *A **Helm Chart** is a Helm Package. This is the chart or "instructions" of how to put together your releasable package.*
>
> ***Tiller** is a server that runs inside your Kubernetes Cluster anytime you install Helm. Tiller manages installations of your Helm Charts. As Tiller installs containers into your Kubernetes Cluster on your behalf, security around this process should be a high priority for you.*

Once you have Helm installed in your Kubernetes Cluster and everything going, you can add big functionality with a single line of code.

Want Redis (Or Anything Else) In Your Kubernetes Cluster?

```
helm install stable/redis
```

BOOM! A Redis installation with a Master/Slave configuration for scalability and Persistence Volumes and, for giggles, even a Prometheus metrics exporter.

You want more options? Wordpress? Spinnaker? Sonarqube? PhpBB? Mysql? Jenkins? Drupal? Yup, there are tons there for you to enjoy.

I'm Sold! How Do I Install Helm Into My Kubernetes Cluster?!

Great! It is time to start talking code and installation scripts. There are two directions that you can go when installing Helm into your Kubernetes Cluster. The first is a fairly vanilla install that gets you all the basics but may not be as secure as you'd want for your production cluster. The second will be much more secure using TLS to lockdown your Tiller to Helm connection. In the remainder of this article we are going to look at how to get a basic install working. This sort of installation is best if you have a very locked down Kubernetes Cluster already or if you are running your Kubernetes Cluster in MiniKube (aka, not production).

In my next article I am going to focus specifically on installing Helm with TLS as it is a bit more involved. To be transparent, Helm does have very good documentation on the installation process, however things just don't always go as the documentation promises.

As such I'm going to add some lessons learned that slowed me down so you don't have the same issues.

Update: I published the second article!

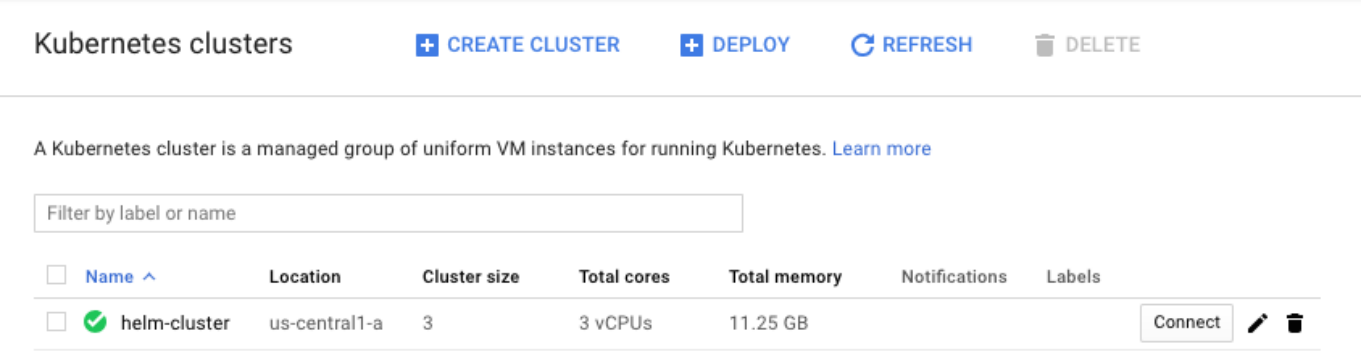
If you haven't gone through or even read the first part of this series you might be lost, have questions where the code is, or what was done previously. Remember this assumes you're using GCP and GKE. I will always provide the code and how to test the code is working as intended.

First, Create Your Kubernetes Cluster

To install Helm into your Kubernetes Cluster you'll first need a Kubernetes Cluster. I've created some scripts to make this easier as creating the Cluster isn't the purpose of this article. If you go to your Google Cloud Shell scripting you can enter in the following commands to create a Google Kubernetes Cluster ready for Helm.

```
$ git clone https://github.com/jonbcampos/kubernetes-series.git
$ cd ~/kubernetes-series/helm/scripts
$ sh startup.sh
```

This will take a moment to complete but when you're done you'll have a Kubernetes Cluster ready and waiting.



Your shiny new Kubernetes Cluster

Second, Install Helm

With your Kubernetes Cluster up and running you can start adding in Helm. Below you will see the actual scripts (with notes) necessary to add a very basic Helm installation into GKE.

```
#!/usr/bin/env bash

echo "install helm"
# installs helm with bash commands for easier command line
integration
curl
https://raw.githubusercontent.com/kubernetes/helm/master/scripts/get
| bash
# add a service account within a namespace to segregate tiller
kubectl --namespace kube-system create sa tiller
# create a cluster role binding for tiller
kubectl create clusterrolebinding tiller \
    --clusterrole cluster-admin \
    --serviceaccount=kube-system:tiller

echo "initialize helm"
# initialized helm within the tiller service account
helm init --service-account tiller
# updates the repos for Helm repo integration
helm repo update

echo "verify helm"
# verify that helm is installed in the cluster
kubectl get deploy,svc tiller-deploy -n kube-system
```

Now that you've seen the code necessary, you can be lazy and just run a script to do the install for you.

```
$ cd ~/kubernetes-series/helm/scripts
$ sh add_helm.sh
```

You'll see everything run through your Shell console very quickly but in the end you should see the following lines, showing that Helm was installed completely.

```
NAME                   DESIRED   CURRENT   UP-TO-DATE   AVAILABLE
AGE
deploy/tiller-deploy   1         1         1            0
1s

NAME                TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
svc/tiller-deploy   ClusterIP   10.11.244.223   <none>
44134/TCP   1s
```

You can even do a double check by running a basic Helm command and see the output.

```
$ helm ls # empty result as we haven't installed anything
```

Third, Install A Chart

Now with Helm installed, let's use it! In the scripts to follow we are going to install Redis into our Kubernetes Cluster with some production values recommended by the Redis Chart maintainers.

```
$ helm install stable/redis \
    --values values/values-production.yaml \
    --name redis-system
```

This will only take a moment but we can see that the Redis Chart was successfully deployed by running the following command.

```
$ helm ls
NAME            REVISION        UPDATED
STATUS          CHART           NAMESPACE
redis-system    1               Thu Aug  9 11:02:23 2018
DEPLOYED        redis-3.7.5     default
```

If we give the Pods a moment to startup fully you can return to your `Kubernetes > Workloads` view and see the Pods all ready for you to interact with them.

Your Redis workloads installed in GKE

That's it! A production quality Redis instance installed and ready for development in just moments. This is huge! And all thanks to Kubernetes and Helm Charts.

Profit!

There are more things that Helm can do through the command line interface. This article isn't intended to be a compendium of Helm knowledge, but more of a starting point that gets you going on your Helm journey. I'm going to give some quick commands though that I've found helpful beyond the `install` command.

```
helm search
```

Allows you to search through the Helm Repo's charts for a chart by name. Just a quick way to get you were you need.

```
helm list
```

As we've seen this will list the deployments you have within your Kubernetes Cluster that Helm is managing.

```
helm delete
```

Will delete a Helm Chart from your Kubernetes Cluster.

```
helm rollback
```

Will rollback an upgrade to your Helm Chart to a previous chart. This is super helpful if there were unintended consequences to an upgrade.

A full list of commands can be found here.

Conclusion

That's it, you have the basics of getting going with Helm with Google Kubernetes Engine. I am already setting up the next installment of this series to go into how to secure your Helm installation with TLS. From the docs, my own struggles, and the amount of questions online around the subject I can assume more help here will be beneficial to everyone.

For now keep moving forward and make sure to share your feedback!

Teardown

Before you leave make sure to cleanup your project so you aren't charged for the VMs that you're using to run your cluster. Return to the Cloud Shell and run the teardown

script to cleanup your project. This will delete your cluster and the containers that we've built.

```
$ cd ~/kubernetes-series/helm/scripts
$ sh teardown.sh
```

Other Posts In This Series

Install Secure Helm In GKE

In my last post I talked a lot about the joys of Helm and why you should spend the time installing it into your...

medium.com

Kubernetes: Running Background Tasks With Batch-Jobs

When building amazing applications, there are times that you might want to handle an action outside of a user's...

medium.com

Kubernetes: Run A Pod Per Node With Daemon Sets

My initial title to this article was just "Daemon Sets" with the assumption that it would be enough to get the point...

medium.com

Kubernetes: Cron Jobs

Sometimes your work isn't transactional. Instead of waiting for a user to click a button and have systems light up we

to click a button and have systems light up we...

medium.com

Kubernetes: DNS Proxy With Services

When building an application it is common that you'll need to interact with external services to complete your business...

medium.com

Kubernetes: Routing Internal Services Through FQDN

I remember when I was first getting into Kubernetes. Everything was new and shiny and about scale. As I continued...

medium.com

Kubernetes: Liveness Checks

Recently I put together a quick article about the Kubernetes Readiness Probe and how important it was for your cluster...

medium.com

Kubernetes: Readiness Probe

In case there was any question about this feature, I am writing about it specifically to state that this is not an...

itnext.io

Kubernetes: Horizontal Pod Scaling

With Pod Autoscaling your Kubernetes Cluster can monitor the load of your existing Pods and determine if we need more...

medium.com

Kubernetes: Cluster Autoscaler

Autoscaling is a huge (and marketed) feature of Kubernetes. When your site/app/api/project makes it big and the flood...

medium.com

Kubernetes: Day One

This is the obligatory step one Kubernetes post. If you're interested in Kubernetes you've probably read 100 of these...

medium.com

Questions? Feedback? I'm very interested to hear what issues you might run across or if this helped you understand a bit better. If there is something I missed feel free to share that too. We are all in this together!

Jonathan Campos is an avid developer and fan of learning new things. I believe that we should always keep learning and growing and failing. I am always a supporter of the development community and always willing to help. So if you have questions or comments on this story please ad them below. Connect with me on LinkedIn or Twitter and mention this story.

Kubernetes Google Cloud Platform Helm Docker Containers