INTRODUCTION TO KUBERNETES HELM CHARTS

In this post we are going to discuss a tool used with Kubernetes called Helm. Part of our multi-part Kubernetes Guide, this article will:

- Explore Helm charts
- Determine when and why to use Helm and Helm Charts
- Provide instructions for getting started

To explore other K8s topics, use the navigation menu on the right-hand side.

(This article is part of our Kubernetes Guide. Use the right-hand menu to navigate.)

What is Helm?

In simple terms, Helm is a package manager for Kubernetes. Helm is the K8s equivalent of yum or apt. Helm deploys charts, which you can think of as a packaged application. It is a collection of all your versioned, pre-configured application resources which can be deployed as one unit. You can then deploy another version of the chart with a different set of configuration.

Helm helps in three key ways:

- Improves productivity
- Reduces the complexity of deployments of microservices
- Enables the adaptation of cloud native applications

Why use Helm?

Writing and maintaining Kubernetes **YAML manifests** for all the required Kubernetes objects can be a time consuming and tedious task. For the simplest of deployments, you would need at least 3 YAML manifests with duplicated and hardcoded values. Helm simplifies this process and creates a single package that can be advertised to your cluster.

Helm is a client/server application and, until recently, has relied on Tiller (the helm server) to be deployed in your cluster. This gets installed when installing/initializing helm on your client machine. Tiller simply receives requests from the client and installs the package into your cluster. Helm can be easily compared to RPM of DEB packages in Linux, providing a convenient way for developers to package and ship an application to their end users to install.

Once you have Helm installed and configured (details below), you are able to install production-ready applications from software vendors, such as MongoDB, MySQL and others, into your Kubernetes cluster with one very simple **helm install** command. Additionally, removing installed applications in your cluster is as easy as installing them.

What are Helm charts?

Helm Charts are simply Kubernetes YAML manifests combined into a single package that can be advertised to your Kubernetes clusters. Once packaged, installing a Helm Chart into your cluster is as easy as running a single **helm install**, which really simplifies the deployment of containerized applications.

Describing Helm

Helm has two parts to it:

- The client (CLI), which lives on your local workstation.
- The server (Tiller), which lives on the Kubernetes cluster to execute what's needed.

The idea is that you use the CLI to push the resources you need and tiller will make sure that state is in fact the case by creating/updating/deleting resources from the chart. To fully grasp helm, there are 3 concepts we need to get familiar with:

- **Chart**: A package of pre-configured Kubernetes resources.
- **Release**: A specific instance of a chart which has been deployed to the cluster using Helm.
- **Repository**: A group of published charts which can be made available to others.

Benefits of Helm

Developers like Helm charts for many reasons:

Boosts productivity

Software engineers are good at writing software, and their time is best spent doing just that. Using Helm allows software to deploy their test environments at the click of a button.

An example of this might be that, in order to test a new feature, an engineer needs a SQL database.

Instead of going through the process of installing the software locally, creating the databases and tables required, the engineer can simply run a single **Helm Install** command to create and prepare the database ready for testing.

Reduces duplication & complexity

Once the chart is built once, it can be used over and over again and by anyone. The fact that you can use the same chart for any environment reduces complexity of creating something for dev, test and prod. You can simply tune you chart and make sure it is ready to apply to any environment. And you get the benefit of using a production ready chart in dev.

Smooths the K8S learning curve

It's no secret that the learning curve for Kubernetes and containers is long for your average developer. Helm simplifies that learning curve: developers do not require a full, detailed understanding of the function of each Kubernetes object in order to start developing and deploying container applications.

Helm easily integrates into CI/CD pipelines and allows software engineers to focus on writing code—not deploying applications.

Simplifies deployments

Helm Charts make it easy to set overridable defaults in the **values.yaml** file, allowing software vendors or administrators of charts to define a base setting. Developers and users of charts can override these settings when installing their chart to suit their needs. If the default installation is required, then no override is required.

Deploying applications to Kubernetes is not a straightforward process, with different objects being tightly coupled. This required specific knowledge of these objects and what their functions are in order to be able to successfully deploy. Helm takes the complexity out of that doing much of the hard work for you.

Describing a Helm chart

Helm has a certain structure when you create a new chart. To create, run "*helm create YOUR-CHART-NAME*". Once this is created, the directory structure should look like:

```
YOUR-CHART-NAME/
  |
  |- .helmignore
  |
  |- Chart.yaml
  |
  |- values.yaml
  |
  |- charts/
  |
  |- templates/
```

- **.helmignore**: This holds all the files to ignore when packaging the chart. Similar to .gitignore, if you are familiar with git.
- **Chart.yaml**: This is where you put all the information about the chart you are packaging. So, for example, your version number, etc. This is where you will put all those details.
- **Values.yaml**: This is where you define all the values you want to inject into your templates. If you are familiar with terraform, think of this as helms variable.tf file.
- **Charts**: This is where you store other charts that your chart depends on. You might be calling another chart that your chart need to function properly.
- **Templates**: This folder is where you put the actual manifest you are deploying with the chart. For example you might be deploying an nginx deployment that needs a service, configmap and secrets. You will have your deployment.yaml, service.yaml, config.yaml and secrets.yaml all in the template dir. They will all get their values from values.yaml from above.

Installing Helm and configuring Helm Charts

Ready to use Helm? Installing and configuring Helm for your K8S cluster is a very quick and straight forward process—there are multiple versions of Helm that can be installed (v1/v2 and most recently v3), all of which can be configured to your organization's needs. Check out the getting started page for instructions on downloading and installing Helm.

Beginning your first Helm chart is as simple as installing some charts from the stable repository, which is available on GitHub. The Helm stable repository is a collection of curated applications ready to be deployed into your cluster.

Helm users can write their own charts or can obtain charts from the stable repository:

- MongoDB
- MySQL
- WordPress

If you want to write your own Helm Charts for your applications, Helm provides a simple Developer's Guide for getting started.

Helm simplifies software deployment

In summary, Helm takes a lot of the complexity out of deploying software and your own applications to Kubernetes. It can be installed in a matter of minutes and you can be deploying charts from the stable repository in no time. Writing your own charts is also a straightforward process, though does require understanding of Kubernetes objects. Helm can empower your developers to work more efficiently giving them the tools they need to test their code locally.

Additional resources

For more on Kubernetes, explore these resources:

- Kubernetes Guide, with 20+ articles and tutorials
- BMC DevOps Blog
- Bring Kubernetes to the Serverless Party
- How eBay is Reinventing Their IT with Kubernetes & Replatforming Program