HOW TO WRITE KUBECTL SUBCOMMANDS

Since the release of Kubernetes v1.12 it has been possible to extend kubectl with subcommands (plugins) to make automating repetitive Kubernetes tasks seem more kubectl native and easier to use for developers.

Before you can start to make use of kubectl subcommands you're going to need to upgrade your version of kubectl cli to at least v1.12. Follow the official Kubernetes documentation for your platform here. (link: https://kubernetes.io/docs/tasks/tools/install-kubectl/).

What are the benefits of kubectl subcommands?

Using subcommands doesn't really add any benefit that you can't introduce by writing your own scripts and distributing them as you would normally. However they do allow your scripts to look like they are built right into kubectl and feel more natural for users if you are distributing these scripts to developer machines.

Kubectl plugins suit 2 major languages each with their own benefits:

Go: The go to choice for cloud native/kubernetes which can be distributed as a single binary.

Bash: Cross platform and can be relied upon due to users invoking kubectl which is a bash command.

As an example this guide is going to focus on building a kubectl subcommand executed with `kubectl cmd` that is going to run an argument based command on all pods in a given namespace

that match a **`| grep`** filter. This example will use `env` as the command to run on pods to simply print out all environment variables however can be invoked with any command you desire.

Save the following script anywhere in your **`$PATH`**.

Make the script executable with **`chmod +x /usr/local/bin/kubectl-cmd`**

```
➜  workspace git:(master) ✗ kubectl plugin list
The following compatible plugins are available:

/usr/local/bin/kubectl-cmd
```

To confirm that kubectl is aware of your new plugin type

`kubectl plugin list`
Now that the plugin is listed as available inside kubectl there is nothing left to do. You can now invoke your new kubectl subcommand with **`kubectl cmd <arg1> <arg2> <arg3>`**.

```
➜  workspace git:(master) ✗ kubectl cmd kube-system kube env

etcd-minikube

PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=minikube
HOME=/root
```

Invoking kubectl plugins

As seen in the example above, arguments that your plugin is invoked with will be passed to your executable in the same way they would if you were to run this in any other way.

You can read more about the kubectl plugin mechanisms on the official Kubernetes GitHub https://github.com/kubernetes/enhancements/blob/c665c8d7203e15cc4b0ad53343d357ca3019c22c/keps/sig-cli/0024-kubectl-plugins.md.

Aside from writing your own kubectl plugins you can use krew (link: https://github.com/kubernetes-sigs/krew) to find and install community plugins written by other developers to extend the built in functionality of kubectl. You can also publish your own plugins to krew for others to use.

The code used in this guide can be found on GitHub so you can quickly get started. I've also included a simple Ansible Playbook to simplify installing any custom plugins you have written which can be found https://github.com/dpmerron-ltd/How-To-Write-Kubectl-Subcommands/tree/master/kubectl-plugin-installer

https://github.com/dpmerron-ltd/How-To-Write-Kubectl-Subcommands