Implementing a kubectl Plugin

Learn how to implement a kubectl plugin.

Overview

The `kubectl` plugins are executable files that can be implemented with any language we like. We can write with shell script, Golang, Java, Python, etc.

In this lesson, we're going to implement a Golang based plugin, where we can make use of multiple go modules shipped by the Kubernetes community. Moreover, it will be a handy reference when we implement a more complex plugin in the future.

Implementation

Below is our plugin implementation, where we define a `main.go` as the entry point of our executable command. This can be used as a scaffold for other `kubectl` plugin implementations. In the `main` functionc(**lines 17–30**), we hook our plugin command function `NewCmdPwk` (**line 26**) from the folder `./plugin`. The core implementation of our plugin is put in the file `./plugin/pwk.go`.

main.go ×

Search in directory...

/
plugin
go.sum
main.go
go.mod

```
 1  package main
 2  
 3  import (
 4      goflag "flag"
 5      "k8s.io/klog"
 6      "math/rand"
 7      "os"
 8      "time"
 9  
10      "github.com/spf13/pflag"
11      "k8s.io/cli-runtime/pkg/genericclioptions"
12      cliflag "k8s.io/component-base/cli/flag"
13  
14      "github.com/educative/pwk/plugin"
15  )
16  
17  func main() {
18      rand.Seed(time.Now().UnixNano())
19  
20      klog.InitFlags(goflag.CommandLine)
```

```
21      defer klog.Flush()
22
23      pflag.CommandLine.SetNormalizeFunc(cliflag.WordSepNormalizeFu
24      pflag.CommandLine.AddGoFlagSet(goflag.CommandLine)
25
26      root := plugin.NewCmdPwk(genericclioptions.IOStreams{In: os.S
27      if err := root.Execute(); err != nil {
28          os.Exit(1)
29      }
30  }
31
```

Our pwk plugin

In `pwk.go`, we define a struct `PwkOptions` (**lines 13–19**) to specify all the options needed for our plugin. Extra fields can be added there. We can add new flags in the function `AddFlags()` (**lines 56–61**) and validate them in the function `Validate()` (**lines 39–43**).

The function `Run` (**lines 45–54**) defines the desired behavior of our plugin, which will output the endpoint of our Kubernetes cluster.

Build and test it out

Let’s build our plugin binary with the following commands, which can be run in the terminal above:

```
go build -o /usr/local/bin/kubectl-pwk ./main.go
```

Build our plugin binary

After the binary is built out, we can invoke our plugin with the following command:

```
kubectl pwk
```

Run our kubectl plugin

The output will be `https://172.17.0.2:6443`.

Ta-da! Our kubectl plugin works!

Back
How kubectl Plugins Work

Next
Quiz on kubectl Plugins