



Kubectl Plugins 101

By
Jonathan Berkahn, IBM
Carolyn Van Slyck, Microsoft



Outline

- Motivation
- Overview of how Kubectl plugins currently work
- Walkthrough & Demo of a simple plugin
- Advice for writing your own plugin
- Looking forward



Motivation

- Service Catalog is a Kube incubator project that implements many custom resources
- SIG Service Catalog wanted a CLI frontend utility for using Service Catalog
 - Novel interaction with custom resources
 - Custom display of those resources
 - Domain-specific verbs
- `svcat`
 - Conversion of a preexisting standalone binary CLI for Service Catalog
 - Usable in both standalone and plugin modes



WE WANT YOU!



Anatomy of a Plugin

Plugin Executable

- Any language
- written however you want
- Takes in flags from Kubectl via environment variables
- Kept in `~/.kube/plugins`


Plugin.yaml

- Description of your plugins commands and their flags
- Stored next to the binary



Demo: A very tiny plugin

github.com/carolynvs/kubectl-flags-plugin




```
name: "flags"
shortDesc: "I echo all the flags that have been passed to my plugin as
            environment variables"
command: "bash -c env"
flags:
  - name: output
    shorthand: o
    desc: output format
```



Demo: A killer plugin

github.com/carolynvs/kubectl-kill-plugin



```
name: "kill"
```


```
shortDesc: "Remove any finalizers on a pod, and delete it"
```

```
command: "./kill"
```

```
flags:
```

```
  - name: grace-period
```


```
    desc: Period of time in seconds given to the resource to terminate gracefully
```



```
func kill(podName string, gracePeriod int64) {
    client, ns := loadConfig()
    removeFinalizers(client, ns, podName)
    deletePod(client, ns, podName, gracePeriod)
}

func removeFinalizers(client *kubernetes.Clientset, ns, podName string) {
    pod := getPod(client, ns, podName)
    pod.SetFinalizers([]string{})
    client.CoreV1().Pods(ns).Update(pod)
}

func deletePod(client *kubernetes.Clientset, ns, podName string, gracePeriod
int64) {
    opts := &metav1.DeleteOptions{GracePeriodSeconds: &gracePeriod}
    client.CoreV1().Pods(ns).Delete(podName, opts)
}
```




```
func init() {
    // Wire up glog flags because glog is silly
    flag.CommandLine.Set("logtostderr", "true")
    flag.CommandLine.Set("v", os.Getenv("KUBECTL_PLUGINS_GLOBAL_FLAG_V"))
}

func main() {
    podName := os.Args[1]

    var gracePeriod int64
    gracePeriodFlag := os.Getenv("KUBECTL_PLUGINS_LOCAL_FLAG_GRACE_PERIOD")
    if g, err := strconv.ParseInt(gracePeriodFlag, 10, 64); err == nil {
        gracePeriod = g
    }

    kill(podName, gracePeriod)
}
```



```
import "k8s.io/kubectl/pkg/pluginutils"

func loadConfig() (*kubernetes.Clientset, string) {
    restConfig, kubeConfig, _ := pluginutils.InitClientAndConfig()

    c := kubernetes.NewForConfigOrDie(restConfig)
    ns, _, _ := kubeConfig.Namespace()

    return c, ns
}
```



Demo: svcat

svc-cat.io/docs/install/#installing-the-service-catalog-cli



svcat: Lessons Learned

- standalone binary vs. plugin
- User Experience
 - Similar verbs should follow kubectl UX
 - Output format
 - Avoid replication of kubectl functionality



Gotchas and Tricks



Environment Variables, Not Flags

```
$ kubectl plugin flags --output json --foo puppies \  
    --namespace demo --context demo-kube --kubeconfig /tmp/rando-cfg
```

```
KUBECTL_PLUGINS_GLOBAL_FLAG_KUBECONFIG=/tmp/rando-cfg
```

```
KUBECTL_PLUGINS_GLOBAL_FLAG_CONTEXT=demo-kube
```

```
KUBECTL_PLUGINS_CURRENT_NAMESPACE=demo
```

```
KUBECTL_PLUGINS_LOCAL_FLAG_OUTPUT=json
```

```
KUBECTL_PLUGINS_LOCAL_FLAG_FOO=puppies
```




❖ k8s.io/kubectl/pkg/pluginutils ❖

```
restCfg, kubeCfg, err := pluginutils.InitClientAndConfig()
```


- REST Config
 - Use to build a client
 - Handles ALL THE FLAGS!
- Kube Config
 - Current Namespace



Getting spf13/cobra and kubectl to play nice

github.com/kubernetes-incubator/service-catalog/cmd/svcat/plugin/plugin.g

o



```
func IsPlugin() bool {
    _, ok := os.LookupEnv("KUBECTL_PLUGINS_CALLER")
    return ok
}

func BindEnvironmentVariables(vip *viper.Viper, cmd *cobra.Command) {
    // Bind glog flags
    vip.BindEnv("v", "KUBECTL_PLUGINS_GLOBAL_FLAG_V")

    // Configure kubectl environment variables
    vip.BindEnv("namespace", "KUBECTL_PLUGINS_CURRENT_NAMESPACE")
    vip.SetEnvPrefix("KUBECTL_PLUGINS_LOCAL_FLAG")

    // Bind cobra flags to the viper-managed environment variables
    vip.BindPFlags(cmd.Flags())
    vip.AutomaticEnv()
    cmd.Flags().VisitAll(func(f *pflag.Flag) {
        if !f.Changed && vip.IsSet(f.Name) {
            cmd.Flags().Set(f.Name, vip.GetString(f.Name))
        }
    })
}
```



Reserved Flags

```
$ kubectl options
```

- kubectl global flags
 - --kubeconfig
 - --context
 - --namespace
- glog flags
 - -v
 - --stderrthreshold
- short flags
 - -n
 - -s



Boolean Flags

```
$ kubectl plugin svcat describe --traverse ✘
```

```
$ kubectl plugin svcat describe --traverse=true ✔
```



Future-Proofing Your Plugin

- Alpha Stability
 - Plugins will continue to be a thing
 - What that means may change
 - May move towards [git plugin model](#)
- In Progress
 - [Boolean flag support PR](#)
 - Support all kubectl global flags in pluginutils
- Recommendation:
 - Binary
 - Parse your own flags, don't rely on kubectl plugin environment variables
 - Don't name your plugin after existing kubectl commands, like logs



Resources

- [kubectl plugin guide](#)
- [kubectl plugin-utils library](#)
- [Generating a plugin.yaml from a spf13/cobra cli app](#)



Plugin Ideas

- CRD custom actions or display
- wait plugin
- bash aliases -> plugin
- troubleshooter plugin
- record 'n play plugin
- plugin plugin ☐



Thank You

Jonathan Berkhahn

- github.com/jberkhahn
- jaberkha@us.ibm.com

Carolyn Van Slyck

- [@carolynvs](https://twitter.com/carolynvs)
- carolyn.vanslyck@microsoft.com

