Useful Kubectl with AWK commands

* First let’s apply ngnix deployment to our cluster and expose it as NodePort, so you can apply commands to these resources.

$ kubectl create deployment nginx --image=nginx   
$ kubectl expose deployment nginx --type=NodePort --port=8080

* Then let’s interact with Kubernetes cluster with Kubectl and awk to see how we can manipulate the output of kubectl. Let’s start with basic command as below:

$ kubectl get pods -o wide | awk ‘{print $1}’

This command will retrieve a list of all pods in the cluster and use awk to print out only the first column, which will be the name of each pod.

* Now imagine what if you want so check which node is hosting a specific pod, let’s check the below command:

$ kubectl describe pod <pod\_name> | awk ‘/Node:/ {print $2}’

This command will retrieve the detailed information about a specific pod and use awk to extract the name of the node where the pod is running.

* Then what if you want to have a list of the services that are running with type of NodePort:

$ kubectl get services -o wide | awk ‘$2 == “NodePort” {print $1}’

This command will retrieve a list of all services in the cluster and use *awk* to filter out only those that have a type of “NodePort”, then print out the name of those services.

* Another magic can be done here while troubleshooting to check which pods are not running:

$ kubectl get pods -o wide | awk ‘$2 == “0/1” {print $1}’

This command will retrieve a list of all pods in the cluster and use awk to filter out only those that have a status of “0/1”, indicating that they are not fully running, then print out the name of those pods.

* At sometimes you may want to delete all the pods in a namespace or some of them, with just one single command line rather than repeating kubectl command. Let’s have a look on the below command.

# kubectl get pods -n <namespace> --no-headers=true | awk ‘/nginx-\*|app-\*/{print $1}’ | xargs kubectl delete -n grafana pod

This command will delete all the pods in a certain namespace following a certain patterns which will be in our case “nginx-\* and app-\*”, **so any pods containing these pattern in their names, will be deleted even if these patterns are in the middle of names of the pods.**

Note here that *xargs* is used to cascade output from awk to *kubectl delete*command.

* Finally what if you want to delete pods matching an exact pattern. Here we should make good use of regex. Let’s have a look on the below command:

# kubectl get pods -n <namespace> --no-headers=true | awk '/^nginx-|^app-/{print $1}' | xargs kubectl delete -n grafana pod

When you run the above command, **all pods starts with ‘nginx-’ and ‘app-’ will be deleted.**You can notice the ‘^’ which indicates a pattern that starts with a specific string.

**A general advise:**When applying a command with multiple pipes, It is always recommended to check the output from each stage before the pipe, so running the last command in the below way, will give good insights on what is the expected output from each pipe stage.

# kubectl get pods -n <namespace> --no-headers=true  
# kubectl get pods -n <namespace> --no-headers=true | awk '/^nginx-|^app-/{print $1}'

So at this point, we have a full list with the pods that can be deleted. Have a look to make sure the output is matching the expected one. Then safely apply the whole command.