
kubectl-neat

Remove clutter from Kubernetes manifests to make them more readable.

Demo

Here is a result of a `kubectl get pod -o yaml` for a simple Pod. The lines marked in red are considered redundant and will be removed from the output by kubectl-neat.

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2019-04-24T19:55:27Z"
  labels:
    name: myapp
  name: myapp
  namespace: default
  resourceVersion: "274103"
  selfLink: /api/v1/namespaces/default/pods/myapp
  uid: e8330f3c-66ca-11e9-b6fa-0800271788ca
spec:
  containers:
  - image: nginx
    imagePullPolicy: Always
    name: myapp
    ports:
    - containerPort: 1234
      protocol: TCP
    resources: {}
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: default-token-nmshj
      readOnly: true
  dnsPolicy: ClusterFirst
  enableServiceLinks: true
  nodeName: minikube
  priority: 0
  restartPolicy: Always
  schedulerName: default-scheduler
  securityContext: {}
  serviceAccount: default
  serviceAccountName: default
  terminationGracePeriodSeconds: 30
```

Why

When you create a Kubernetes resource, let's say a Pod, Kubernetes adds a whole bunch of internal system information to the yaml or json that you originally authored. This includes:

- Metadata such as creation timestamp, or some internal IDs
- Fill in for missing attributes with default values
- Additional system attributes created by admission controllers, such as service account token
- Status information

If you try to `kubectl get` resources you have created, they will no longer look like what you originally authored, and will be unreadably verbose.

`kubectl-neat` cleans up that redundant information for you.

Installation

```
1 kubectl krew install neat
```

or just download the binary if you prefer.

When used as a kubectl plugin the command is `kubectl neat`, and when used as a standalone executable it's `kubectl-neat`.

Usage

There are two modes of operation that specify where to get the input document from: a local file or from Kubernetes.

Local - file or Stdin

This is the default mode if you run just `kubectl neat`. This command accepts an optional flag `-f / --file` which specifies the file to neat. It can be a path to a local file, or `-` to read the file from stdin. If omitted, it will default to `-`. The file must be a yaml or json file and a valid Kubernetes resource.

There's another optional optional flag, `-o / --output` which specifies the format for the output. If omitted it will default to the same format of the input (auto-detected).

Examples:

```
1 kubectl get pod mypod -o yaml | kubectl neat
2
```

```
3 kubectl get pod mypod -oyaml | kubectl neat -o json
4
5 kubectl neat -f - <./my-pod.json
6
7 kubectl neat -f ./my-pod.json
8
9 kubectl neat -f ./my-pod.json --output yaml
```

Kubernetes - kubectl get wrapper

This mode is invoked by calling the `get` subcommand, i.e `kubectl neat get ...`. It is a convenience to run `kubectl get` and then `kubectl neat` the output in a single command. It accepts any argument that `kubectl get` accepts and passes those arguments as is to `kubectl get`. Since it executes `kubectl`, it need to be able to find it in the path.

Examples:

```
1 kubectl neat get -- pod mypod -oyaml
2 kubectl neat get -- svc -n default myservice --output json
```

How it works

Besides general tidying for status, metadata, and empty fields, kubectl-neat primarily looks for two types of things: default values inserted by Kubernetes' object model, and common mutating controllers.

Kubernetes object model defaults

For de-defaulting Kubernetes' object model, we invoke the same code that Kubernetes would have, and see what default values were assigned. If these observed values look like the ones we have in the incoming spec, we conclude they are default. If they weren't, and the user manually set a field to it's default value, it's not a bad thing to remove it anyway.

Common mutating controllers

Here are the recommended admission controllers, and their relation to kubectl-neat:

controller	description	neat
NamespaceLifecycle	rejects operations on resources in namespaces being deleted	ignore
LimitRanger	set default values for resource requests and limits	ignore
ServiceAccount	set default service account and assign token	Remove default-token-* volumes. Remove deprecated <code>spec.serviceAccount</code>
TaintNodesByCondition	automatically taint a node based on node conditions	TODO
Priority	validate priority class and add it's value	ignore
DefaultTolerationSeconds	configure pods to temporarily tolerate notready and unreachable taints	TODO
DefaultStorageClass	validate and set default storage class for new pvc	ignore
StorageObjectInUseProtection	prevent deletion of pvc/pv in use by adding a finalizer	ignore
PersistentVolumeClaimResize	enforce pvc resizing only for enabled storage classes	ignore
MutatingAdmissionWebhook	implement the mutating webhook feature	ignore
ValidatingAdmissionWebhook	implement the validating webhook feature	ignore
RuntimeClass	add pod overhead according to runtime class	TODO
ResourceQuota	implement the resource qouta feature	ignore
Kubernetes Scheduler	assign pods to nodes	Remove <code>spec.nodeName</code>
