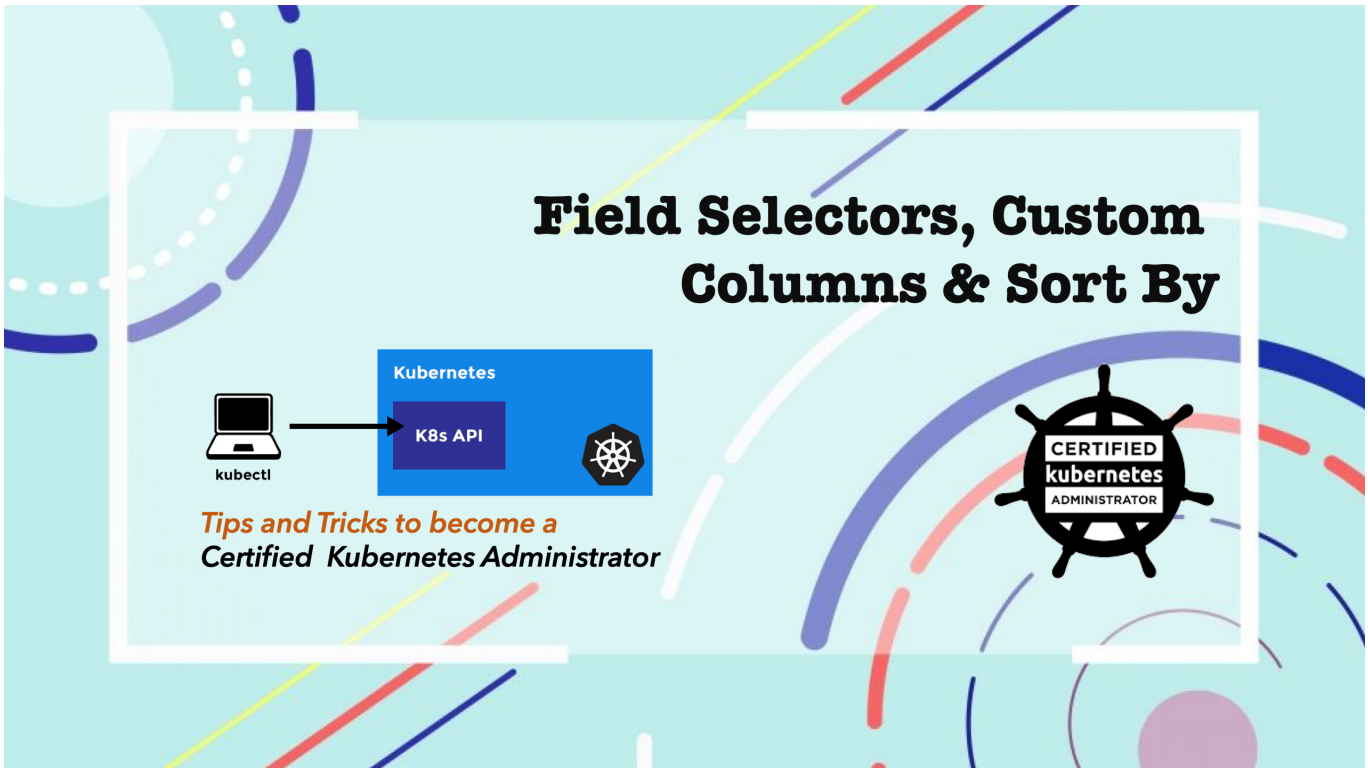
Certified Kubernetes Administrator (CKA) — Tips and Tricks — Part 4

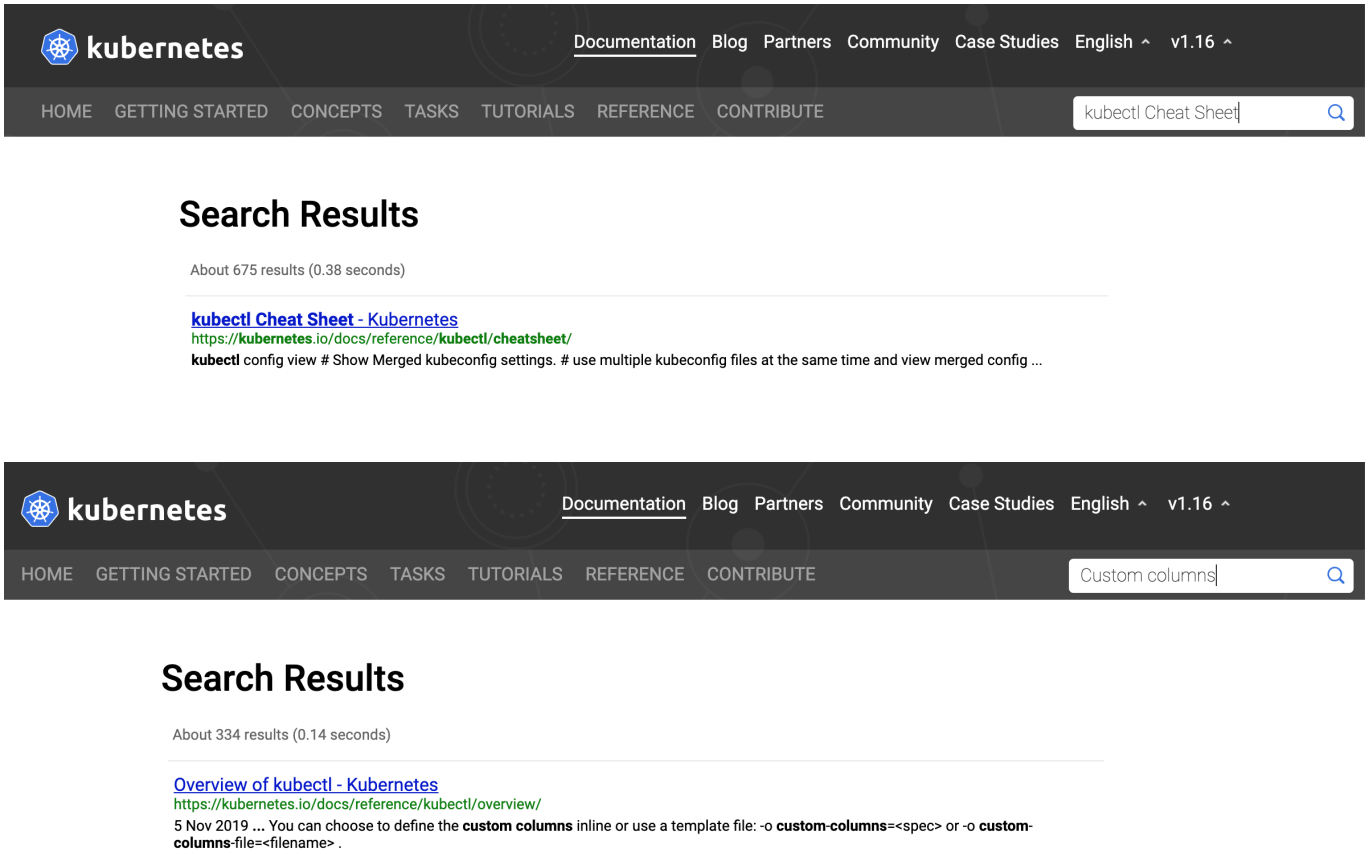
Arun Ramakani Follow
Dec 9 · 5 min read



Mastering “Field Selectors”, “Custom Column” and “Sort By” will help us in a few questions at the CKA exam. In this blog, we will look into a framework that will help us to construct any kubectl “Field Selectors”, “Custom Column” and “Sort By” commands without memorizing much. If you are lucky you will get two questions from the below examples.

Tip 1: Reach Documentation And Escape Memorizing

You will be allowed to refer the Kubernetes documentation page during the exam. From the Kubernetes documentation page (*doc page*) search for “kubectl Cheat Sheet” & “Custom Column” respectively, then from the results click the first link.



Scan through the resulting pages to get hold of the Field Selectors, Custom columns and Sort By commands.

> kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type==”ExternalIP”)].address}’
>
> *kubectl get services — sort-by=.metadata.name*
>
> *kubectl get pods <pod-name> -o custom-columns=NAME:.metadata.name,RSRC:.metadata.resourceVersion*

Tip 2: Looking at the output structure

To start using the JSON path in all these commands, we should know the output structure from kubectl commands. You can either use “-o yaml” or “-o json” on any kubectl command to understand the structure. I prefer “-o yaml”, because the output from “-o json” is less readable than the output of “-o yaml”. You can choose the option which looks better readable for you.

> *kubectl get node -o yaml*

> *kubectl get node -o json*

```
master $ kubectl get nodes -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Node
  metadata:
    annotations:
      kubeadm.alpha.kubernetes.io/cri-socket: /var/run/dockershim.sock
      node.alpha.kubernetes.io/ttl: "0"
      volumes.kubernetes.io/controller-managed-attach-detach: "true"
    creationTimestamp: 2019-12-09T05:39:43Z
    labels:
      beta.kubernetes.io/arch: amd64
      beta.kubernetes.io/os: linux
      kubernetes.io/hostname: master
      node-role.kubernetes.io/master: ""
    name: master
    namespace: ""
    resourceVersion: "1521"
    selfLink: /api/v1/nodes/master
    uid: 4d262563-1a46-11ea-aff6-0242ac11000e
  spec:
    taints:
    - effect: NoSchedule
      key: node-role.kubernetes.io/master
  status:
    addresses:
    - address: 172.17.0.14
      type: InternalIP
    - address: master
      type: Hostname
    allocatable:
      cpu: "4"
```

Tip 3: Find Arrays and Map

The next step is to know, how to find arrays and maps from the YAML structure. Concerning YAML, it's the same as the YAML's that we use to create any Kubernetes resource. The one that starts with "-" is an array. If you look at the below image

1. "addresses" is an Array
2. "allocatable" is a Map

```
status:
  addresses:
  - address: 172.17.0.66
    type: InternalIP
  - address: master
    type: Hostname
```

```
  allocatable:
    cpu: "4"
    ephemeral-storage: "89032026784"
    hugepages-1Gi: "0"
```

Tip 4: Field Selectors

Now we know the base command, how to look and interpret the output structure, the next step is to learn how to construct the given requirement into a kubectl command. We can look into two different examples moving from simple to complex scenarios. so, that we will be able to answer any exam question.

Challenge 1: List all pods name, lastProbeTime from status where the type is ready.

Execute “kubectl get pod -o yaml” and look for the needed fields in the YAML output.

> “pods name:: items : Array → metadata : Map → name”

```
master $ kubectl get pod -o yaml
apiVersion: v1
items:
- apiVersion: v1
  kind: Pod
  metadata:
    creationTimestamp: 2019-12-09T12:34:18Z
    generateName: nginx1-846d65cc74-
    labels:
      pod-template-hash: "4028217730"
      run: nginx1
    name: nginx1-846d65cc74-4krkk
```

> *Command below will give you all the pod names.*
>
> *kubectl get pod -o jsonpath='{.items[*].metadata.name}'*

```
master $ kubectl get pod -o jsonpath='{.items[*].metadata.name}'
nginx1-846d65cc74-4krkk nginx2-789fb7ff4-glwclmaster $
```

Learning 1: Use [*] for getting all array items.

> *lastTransitionTime:: items : Array → status : map → conditions : Array → lastProbeTime*

```
  status:
    conditions:
    - lastProbeTime: null
      lastTransitionTime: 2019-12-09T12:34:18Z
      status: "True"
      type: Initialized
    - lastProbeTime: null
      lastTransitionTime: 2019-12-09T12:34:25Z
      status: "True"
      type: Ready
```

> *Below command will give all the pod names and lastTransitionTime*
>
> *kubectl get pod -o jsonpath='{.items[*].metadata.name}{.items[*].status.conditions[?(@.type==”Ready”)].lastTransitionTime}'*

```
master $ kubectl get pod -o jsonpath='{.items[*].metadata.name}{.items[*].status.conditions[?(@.type=="Ready")].lastTransitionTime}'
nginx1-846d65cc74-4krkk nginx2-789fb7ff4-glwcl2019-12-09T12:34:25Z 2019-12-09T12:34:27Zmaster $
```

Learning 2: Use [?(@.type==”Ready”)] to apply “where condition” with the array.

1. “?” represents an if condition.
2. “@” represents the current element in the array.

Challenge 2: Get all schedulable nodes. Each node should be displayed in a new row.

> *kubectl get nodes -o jsonpath="{range .items[*]}{.metadata.name} {.spec.taints[*].effect}{\"\n\"}{end}"*

```
master $ kubectl get nodes -o jsonpath="{range .items[*]}{.metadata.name} {.spec.taints[*].effect}{\"\n\"}{end}"
master NoSchedule
node01
```

Learning 3: To display every node in an individual row, we have to use a loop to process node by node. Use {range .items[*]} ..... {end} to loop through the list of items.

Learning 4: Within the loop, directly refer the items from the looping node. For example, use {.metadata.name} under {range .items[*]} for referring name. Should not use {.items[*].metadata.name}.

Learning 5: At the end of every iteration use *{\"\n\"}* for new line.

> *kubectl get nodes -o jsonpath="{range .items[*]}{.metadata.name} {.spec.taints[*].effect}{\"\n\"}{end}" | grep -v NoSchedule*

```
master $ kubectl get nodes -o jsonpath="{range .items[*]}{.metadata.name} {.spec.taints[*].effect}{\"\n\"}{end}" | grep -v NoSchedule
node01
```

Learning 6: Use "grep -v NoSchedule" to exclude node with taint 'NoSchedule'. Note: "grep -v" is inverse grep.

Tip 4: Custom Columns

We can achieve the same requirement of challenge 2 with Custom Columns.

> *Get all schedulable nodes. Each node should be displayed in a new row.*
>
> *kubectl get node -o custom-columns=NAME:.metadata.name,TAINT:.spec.taints[*].effect | grep -v NoSchedule*

```
master $ kubectl get node -o custom-columns=NAME:.metadata.name,TAINT:.spec.taints[*].effect | grep -v NoSchedule
NAME      TAINT
node01    <none>
```

> Learning 7: For “custom columns” and “sort by” we will use the same yaml structure, excluding the outer “items array” (i.e exclude — .items[*])
>
> Learning 8: Custom columns can have titles. NAME, TAINT in the above example are titles.


```
master $ kubectl get node -o custom-columns=NAME:.metadata.name,TAINT:.spec.taints[*].effect | grep -v NoSchedule
NAME      TAINT
node01    <none>
```


Tip 5: Sort By

Sort by will help us to order the output based on an attribute.

Challenge 3: Get all persistence volume from kube-system namespace ordered with capacity.

> *Look at the json structure to locate the capacity.*
>
> “*capacity*:: items : Array → spec : Map → capacity : storage”

```
master $ kubectl get pv -n kube-system  -o yaml
apiVersion: v1
items:
```

```
- apiVersion: v1
  kind: PersistentVolume
  metadata:
    creationTimestamp: 2019-12-09T12:31:46Z
    finalizers:
    - kubernetes.io/pv-protection
    name: pv-log-1
    namespace: ""
    resourceVersion: "1571"
    selfLink: /api/v1/persistentvolumes/pv-log-1
    uid: dd38a869-1a7f-11ea-8bef-0242ac11003f
  spec:
    accessModes:
    - ReadWriteMany
    capacity:
      storage: 100Mi
    hostPath:
```

> *The command for both sorted and unsorted output,*
>
> *kubectl get pv -n kube-system — sort-by=.spec.capacity.storage*
>
> *kubectl get pv -n kube-system*

```
master $ kubectl get pv -n kube-system --sort-by=.spec.capacity.storage
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM     STORAGECLASS   REASON    AGE
pv-log-4   40Mi       RWX            Retain           Available                                        21m
pv-log-1   100Mi      RWX            Retain           Available                                        21m
pv-log-2   200Mi      RWX            Retain           Available                                        21m
pv-log-3   300Mi      RWX            Retain           Available                                        21m
master $
master $
master $ kubectl get pv -n kube-system
NAME       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM     STORAGECLASS   REASON    AGE
pv-log-1   100Mi      RWX            Retain           Available                                        21m
pv-log-2   200Mi      RWX            Retain           Available                                        21m
pv-log-3   300Mi      RWX            Retain           Available                                        21m
pv-log-4   40Mi       RWX            Retain           Available                                        21m
master $
```

With this you should be able to handle any “Field Selectors”, “Custom Column” and “Sort By” related question. Also, visit other tips and tricks at

Certified Kubernetes Administrator (CKA) — Tips and Tricks — Part 1

Certified Kubernetes Administrator is a challenging exam by CNCF. Unlike many other certifications, it’s a practical…

medium.com