JSONPath Support

JSONPath Support

- **JSONPath** template is composed of JSONPath expressions enclosed by curly braces {}.
- Kubectl uses JSONPath expressions to **filter on specific fields** in the JSON object and format the output.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mydeploy
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mydeploy
    spec:
      containers:
      - image: nginx
        name: nginx
      - name: apache
        image: httpd
```

Create the deployment and check **yaml** and **json** outputs.

kubectl apply -f deployment.yaml

kubectl get deploy mydeploy -o yaml

kubectl get deploy deploy -o json


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mydeploy
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mydeploy
    spec:
      containers:
      - image: nginx
        name: nginx
      - name: apache
        image: httpd
```

Pull the **replicas** value.

kubectl get deploy mydeploy -o=jsonpath='{@}'

kubectl get deploy mydeploy -o=jsonpath='{.spec}'

kubectl get deploy mydeploy -o=jsonpath='{.spec.replicas}'

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mydeploy
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mydeploy
    spec:
      containers:
      - image: nginx
        name: nginx
      - name: apache
        image: httpd
```

Pull the **maxSurge** value.

```
kubectl get deploy mydeploy \
-o=jsonpath='{.spec.strategy.rollingUpdate.maxSurge}'
```


[{"image":"nginx","imagePullPolicy":"Always","name":"nginx","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"},{"image":"httpd","imagePullPolicy":"Always","name":"apache","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"}]

List the containers

```
kubectl get deploy mydeploy \
-o=jsonpath='{.spec.template.spec.containers}'
```

[**{"image":"nginx","imagePullPolicy":"Always","name":"nginx","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"},**{"image":"httpd","imagePullPolicy":"Always","name":"apache","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"}]

Pull the first container

```
kubectl get deploy mydeploy \
-o=jsonpath='{.spec.template.spec.containers[0]}'
```


[{"image":"nginx","imagePullPolicy":"Always","name":"nginx","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"},**{"image":"httpd","imagePullPolicy":"Always","name":"apache","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"}**]

Pull the second container

```
kubectl get deploy mydeploy \
-o=jsonpath='{.spec.template.spec.containers[1]}'
```

[{"image":"**nginx**","imagePullPolicy":"Always","name":"nginx","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"},{"image":"httpd","imagePullPolicy":"Always","name":"apache","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"}]

Pull the image of the first container

```
kubectl get deploy mydeploy \
-o=jsonpath='{.spec.template.spec.containers[0].container}'
```


[{"image":"nginx","imagePullPolicy":"Always","name":"nginx","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"},{"image":"**httpd**","imagePullPolicy":"Always","name":"apache","resources":{},"terminationMessagePath":"/dev/termination-log","terminationMessagePolicy":"File"}]

Pull the image of the container named **apache**

```
kubectl get deploy mydeploy \
-o=jsonpath='{.spec.template.spec.containers[?(.name=="apache")].image}'
```

{"availableReplicas":2,"conditions":[{"lastTransitionTime":"2024-03-08T09:05:31Z","lastUpdateTime"
:"2024-03-08T09:05:31Z","message":"ReplicaSet \"mydeploy-56485bf89c\" is
progressing.","reason":"ReplicaSetUpdated","status":"True","type":"Progressing"},{"lastTransitionTim
e":"2024-03-08T09:05:32Z","lastUpdateTime":"2024-03-08T09:05:32Z","message":"Deployment
does not have minimum
availability.","reason":"MinimumReplicasUnavailable","status":"False","type":"Available"}],"observed
Generation":3,"readyReplicas":2,"replicas":4,"unavailableReplicas":2,"updatedReplicas":2}

Pull the **status** of the pod

```
kubectl get deploy mydeploy -o=jsonpath='{.status}'
```


{"**availableReplicas":2,**"conditions":[{"lastTransitionTime":"2024-03-08T09:05:31Z","lastUpdateTim
e":"2024-03-08T09:05:31Z","message":"ReplicaSet \"mydeploy-56485bf89c\" is
progressing.","reason":"ReplicaSetUpdated","status":"True","type":"Progressing"},{"lastTransitionTim
e":"2024-03-08T09:05:32Z","lastUpdateTime":"2024-03-08T09:05:32Z","message":"Deployment
does not have minimum
availability.","reason":"MinimumReplicasUnavailable","status":"False","type":"Available"}],"observed
Generation":3,"readyReplicas":2,"replicas":4,"unavailableReplicas":2,"updatedReplicas":2}

Pull the **availableReplicas**

```
kubectl get deploy mydeploy -o=jsonpath='{.status.availableReplicas}'
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mydeploy
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mydeploy
    spec:
      containers:
      - image: nginx
        name: nginx
      - name: apache
        image: httpd
```

Custom Columns

```
$ kubectl get deploy mydeploy -o \
custom-columns=NAME:.metadata.name,REPLICAS:.spec.replicas

NAME      REPLICAS

mydeploy   3
```


apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mydeploy
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mydeploy
    spec:
      containers:
      - image: nginx
        name: nginx
      - name: apache
        image: httpd

Custom Columns

$ kubectl get deploy mydeploy -o \
custom-columns=NAME:.metadata.name,REPLICAS:.spec.replicas

NAME   REPLICAS

mydeploy   3

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mydeploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mydeploy
  strategy:
    rollingUpdate:
      maxSurge: 25%
      maxUnavailable: 25%
    type: RollingUpdate
  template:
    metadata:
      labels:
        app: mydeploy
    spec:
      containers:
      - image: nginx
        name: nginx
      - name: apache
        image: httpd
```

Custom Columns

```
$ kubectl get deploy mydeploy -o \
custom-columns=NAME:.metadata.name,REPLICAS:.spec.replicas

NAME       REPLICAS

mydeploy   3
```



```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: alpha-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
—
apiVersion: v1
kind: PersistentVolume
metadata:
  name: beta-pv-volume
  labels:
    type: local
spec:
  storageClassName: manual
  capacity:
    storage: 5Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data"
```

--sort-by

```
$ k get pv
NAME              CAPACITY   ACCESS MODES   ...
alpha-pv-volume   10Gi       RWO            ...
beta-pv-volume    5Gi        RWO            ...

$ k get pv --sort-by=.spec.capacity.storage
NAME              CAPACITY   ACCESS MODES   ...
beta-pv-volume    5Gi        RWO            ...
alpha-pv-volume   10Gi       RWO            ...
```

THANKS!

Any questions?