**Accessing Log Data - Pods and Containers**

**Multicontainer.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

  name: loggingdemo

  labels:

    app: loggingdemo

spec:

  replicas: 3

  selector:

    matchLabels:

      app: loggingdemo

  template:

    metadata:

      labels:

        app: loggingdemo

    spec:

      containers:

      - name: container1

        image: busybox

        args: [/bin/sh, -c, 'while true; do echo "$(date)": $(hostname): container1; sleep 1; done']

      - name: container2

        image: busybox

        args: [/bin/sh, -c, 'while true; do echo "$(date)": $(hostname): container2; sleep 1; done']

kubectl apply -f multicontainer.yaml

#Pods a specific container in a Pod and a collection of Pods

In Powershell

$PODNAME=$(kubectl get pods -l app=loggingdemo -o jsonpath='{ .items[0].metadata.name }')

echo $PODNAME

#Let's get the logs from the multicontainer pod...this will throw an error and ask us to define which container

kubectl logs $PODNAME

#But we need to specify which container inside the pods

kubectl logs $PODNAME **-c container1**

kubectl logs $PODNAME **-c container2**

#We can access all container logs which will dump each containers in sequence

kubectl logs $PODNAME **--all-containers**

kubectl logs $PODNAME **-A**

#If we need to follow a log, we can do that...helpful in debugging real time issues

#This works for both single and multi-container pods

kubectl logs $PODNAME --all-containers **--follow**

ctrl+c

#For all pods matching the selector, get all the container logs and write it to stdout

kubectl get pods --selector app=loggingdemo

kubectl get pods -l app=loggingdemo

kubectl logs --selector app=loggingdemo --all-containers

#Also helpful is tailing the bottom of a log...

#Here we're getting the last 5 log entries across all pods matching the selector

kubectl logs --selector app=loggingdemo --all-containers **--tail 5**

**Accessing Object Data with JSONPath**

Graphical user interface

Description automatically generated

#Accessing information with jsonpath

#Create a workload and scale it

kubectl create deployment hello-world --image=gcr.io/google-samples/hello-app:1.0

kubectl scale  deployment hello-world --replicas=3

kubectl get pods -l app=hello-world

#We're working with the json output of our objects, in this case pods

#Let's start by accessing that list of Pods, inside items.

#Look at the items, find the metadata and name sections in the json output

kubectl get pods -l app=hello-world **-ojson** > pods.json

#It's a list of objects, so let's display the pod names

kubectl get pods -l app=hello-world -o jsonpath='{.items[\*].metadata.name}'

#Display all pods names, this will put the new line at the end of the set rather then on each object output to screen.

#Additional tips on formatting code in the examples below including adding a new line after each object

kubectl get pods -l app=hello-world -o jsonpath='{.items[\*].metadata.name}{"\n"}'

#It's a list of objects, so let's display the first (zero'th) pod from the output

kubectl get pods -l app=hello-world -o jsonpath='{ .items[0].metadata.name }{"\n"}'

#Get all container images in use by all pods in all namespaces

kubectl get pods --all-namespaces -o jsonpath='{ .items[\*].spec.containers[\*].image }{"\n"}'

#Filtering a specific value in a list

#Let's say there's an list inside items and you need to access an element in that list...

#  ?() - defines a filter

#  @ - the current object

kubectl get nodes c1-cp1 -o json | more

kubectl get nodes -o jsonpath="{.items[\*].status.addresses[?(@.type=='InternalIP')].address}"

#Sorting

#Use the --sort-by parameter and define which field you want to sort on. It can be any field in the object.

kubectl get pods -A -o jsonpath='{ .items[\*].metadata.name }{"\n"}' --sort-by=.metadata.name

#Now that we're sorting that output, maybe we want a listing of all pods sorted by a field that's part of the

#object but not part of the default kubectl output. like creationTimestamp and we want to see what that value is

#We can use a custom colume to output object field data, in this case the creation timestamp

kubectl get pods -A -o jsonpath='{ .items[\*].metadata.name }{"\n"}' \

    --sort-by=.metadata.creationTimestamp \

    --output=custom-columns='NAME:metadata.name,CREATIONTIMESTAMP:metadata.creationTimestamp'

#Clean up our resources

kubectl delete deployment hello-world

**####Additional examples including formatting and sorting examples####**

#Let's use the range operator to print a new line for each object in the list

kubectl get pods -l app=hello-world -o jsonpath='{range .items[\*]}{.metadata.name}{"\n"}{end}'

#Combining more than one piece of data, we can use range again to help with this

kubectl get pods -l app=hello-world -o jsonpath='{range .items[\*]}{.metadata.name}{.spec.containers[\*].image}{"\n"}{end}'

#All container images across all pods in all namespaces

#Range iterates over a list performing the formatting operations on each element in the list

#We can also add in a sort on the container image name

kubectl get pods -A -o jsonpath='{range .items[\*]}{.metadata.name}{"\t"}{.spec.containers[\*].image}{"\n"}{end}' \

    --sort-by=.spec.containers[\*].image

#We can use range again to clean up the output if we want

kubectl get nodes -o jsonpath='{range .items[\*]}{.status.addresses[?(@.type=="InternalIP")].address}{"\n"}{end}'

kubectl get nodes -o jsonpath='{range .items[\*]}{.status.addresses[?(@.type=="Hostname")].address}{"\n"}{end}'

#We used --sortby when looking at Events earlier, let's use it for another something else now...

#Let's take our container image output from above and sort it

kubectl get pods -A -o jsonpath='{ .items[\*].spec.containers[\*].image }' --sort-by=.spec.containers[\*].image

kubectl get pods -A -o jsonpath='{range .items[\*]}{.metadata.name }{"\t"}{.spec.containers[\*].image }{"\n"}{end}' --sort-by=.spec.containers[\*].image

#Adding in a spaces or tabs in the output to make it a bit more readable

kubectl get pods -l app=hello-world -o jsonpath='{range .items[\*]}{.metadata.name}{" "}{.spec.containers[\*].image}{"\n"}{end}'

kubectl get pods -l app=hello-world -o jsonpath='{range .items[\*]}{.metadata.name}{"\t"}{.spec.containers[\*].image}{"\n"}{end}'

**Kubernetes Events**

Event is the result of changes in resource state

Events are having only **one hour** of retention.

Go to log for when something goes wrong

kubectl get events

kubectl describe $TYPE $NAME

#Show events for all objects in the cluster in the default namespace

#Look for the deployment creation and scaling operations from above...

#If you don't have any events since they are only around for an hour create a deployment to generate some

kubectl get events --sort-by='.metadata.creationTimestamp'

#Create a flawed deployment

kubectl create deployment nginx --image ngins

#We can filter the list of events using field selector

kubectl get events --field-selector **type=Warning**

kubectl get events --field-selector **type=Warning,reason=Failed**

#We can also monitor the events as they happen with watch

kubectl get events --watch &

kubectl scale deployment loggingdemo --replicas=5

#break out of the watch

fg

ctrl+c

#These events are also available in the object as part of kubectl describe, in the events section

kubectl describe deployment nginx

kubectl describe replicaset nginx-7df5f8b5cb #Update to your replicaset name

kubectl describe pods nginx