**Logging**

Kubernetes logging by example

Logging is one option to understand what is going on inside your applications and the cluster at large. Basic logging in Kubernetes makes the output a container produces available, which is a good use case for debugging. More advanced setups consider logs across nodes and store them in a central place, either within the cluster or via a dedicated (cloud-based) service.

Let’s create a pod called logme that runs a container writing to stdout and stderr:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/logging/pod.yaml

To view the five most recent log lines of the gen container in the logme pod, execute:

$ kubectl logs --tail=5 logme -c gen

Thu Apr 27 11:34:40 UTC 2017

Thu Apr 27 11:34:41 UTC 2017

Thu Apr 27 11:34:41 UTC 2017

Thu Apr 27 11:34:42 UTC 2017

Thu Apr 27 11:34:42 UTC 2017

To stream the log of the gen container in the logme pod (like tail -f), do:

$ kubectl logs -f --since=10s logme -c gen

Thu Apr 27 11:43:11 UTC 2017

Thu Apr 27 11:43:11 UTC 2017

Thu Apr 27 11:43:12 UTC 2017

Thu Apr 27 11:43:12 UTC 2017

Thu Apr 27 11:43:13 UTC 2017

...

Note that if you wouldn’t have specified --since=10s in the above command, you would have gotten all log lines from the start of the container.

You can also view logs of pods that have already completed their lifecycle. For this we create a [pod](https://github.com/openshift-evangelists/kbe/blob/master/specs/logging/oneshotpod.yaml) called oneshot that counts down from 9 to 1 and then exits. Using the -poption you can print the logs for previous instances of the container in a pod:

$ kubectl apply -f https://raw.githubusercontent.com/openshift-evangelists/kbe/master/specs/logging/oneshotpod.yaml

$ kubectl logs -p oneshot -c gen

9

8

7

6

5

4

3

2

1

You can remove the created pods with:

$ kubectl delete pod/logme pod/oneshot