18.12.2018 Sarajevo

Controlling Kubernetes Scheduler

Bakir Jusufbegović

k8s scheduler - How it works by default?

- In plain/simple terms and from the high level, kubernetes abstracts cluster of machines into one big machine and takes care of scheduling its resources by default.
- Usually, it comes down to this:
  - New pod that needs to be scheduled goes into the queue
  - Kube scheduler takes pods off the queue, calculates (based on applied predicate filters and priority functions) where it could be run and contacts kubelet on the particular node to run the pod
  - Kubelet runs the pod on the designated node
  - In most cases, we don't worry on which node this pod will end up

k8s scheduler - why do we want to mess up with scheduler?

- It is possible to affect schedule process - on the other hand it can be dangerous if we don’t know what we are doing
- Scenarios: running multiple workloads on the same cluster, different types of hardware (ie: disk types) on some nodes, collocating resources on the same availability zone, kube local storage…etc.

k8s scheduler - demo overview

- k8s cluster with 5 nodes
- For the demo purposes, we will deploy prometheus (statefulset) and fluentd (daemonset) on this cluster
- At start, nothing is labeled/tainted and k8s scheduler acts by default

k8s scheduler - taints/tolerations

- Taints - “Bad smell” on the nodes which should discourage scheduling of pods (without tolerations) on them

```
kubectl taint nodes <node-name> <taint-key>=<taint-value>:<effect>
kubectl taint nodes ip-10-0-20-116
taint_node_group=prometheus:NoSchedule
```

- Taint can be: soft (PreferNoSchedule), hard (NoSchedule) and hardest (NoExecute)

- Toleration - declared on the pod level and defines how pod can stand the “smell of taint”

```
tolerations:
- key: "taint_node_group"
  operator: Equal
  value: "prometheus"
```

k8s scheduler - affinities

- Taints and tolerations define where pod can end up but don't restrict where pod will end up
- Affinities - mechanism for attracting pod to specific node (NodeAffinity) or pod (PodAffinity) - but without taint/toleration doesn't guarantee that other pods will not end up on that specific node too
- There is also PodAntiAffinity - ability to discourage pods from each other
- To implement:
  - 1. set label on node

```
kubectl label nodes <node-name> <label-key>=<label-value>
kubectl label nodes ip-10-0-20-225 affinity_node_group=prometheus
```

  - 2. add affinity type into PodSpec:

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: "affinity_node_group"
          operator: In
          values:
          - "prometheus"
```

k8s scheduler - caveats

- What happens with Daemonset if we affect k8s scheduler with taints/tolerations/affinities?
  - fluentd example
  - weave-net example (NoSchedule taint problem?)
- Scaling - Managing CPU/Memory limits in context of taints/tolerations/affinities
- Leave set of nodes untainted so that part of the workload which doesn't have toleration can end up there

Questions & Thank You!