Upgrading and Updating Infinispan 10.0 for Kubernetes

Table of Contents

Chapter 1. Rolling Upgrades and Updates with Kubernetes and OpenShift

Pods running in Kubernetes and OpenShift are immutable. The only way you can alter the configuration is to roll out a new deployment.

Upgrades and updates sound similar but are distinct processes for rolling out new deployments.

1.1. Performing Rolling Updates on Kubernetes

Rolling updates replace existing pods with new ones.

- Rolling Updates
- When to Use a Rolling Deployment

Example DeploymentConfiguration for Rolling Updates

```
- apiVersion: v1
  kind: DeploymentConfig
  metadata:
    name: infinispan-cluster
  spec:
    replicas: 3
    strategy:
      type: Rolling
      rollingParams:
        updatePeriodSeconds: 10
        intervalSeconds: 20
        timeoutSeconds: 600
        maxUnavailable: 1
        maxSurge: 1
    template:
      spec:
        containers:
        - args:
          - -Djboss.default.jgroups.stack=kubernetes
          image: jboss/infinispan-server:latest
          name: infinispan-server
          ports:
          - containerPort: 8181
            protocol: TCP
          - containerPort: 9990
            protocol: TCP
          - containerPort: 11211
            protocol: TCP
          - containerPort: 11222
            protocol: TCP
          - containerPort: 57600
```

```
      protocol: TCP
    - containerPort: 7600
      protocol: TCP
    - containerPort: 8080
      protocol: TCP
    env:
    - name: KUBERNETES_NAMESPACE
      valueFrom: {fieldRef: {apiVersion: v1, fieldPath: metadata.namespace}}
    terminationMessagePath: /dev/termination-log
    terminationGracePeriodSeconds: 90
    livenessProbe:
      exec:
        command:
        - /usr/local/bin/is_running.sh
      initialDelaySeconds: 10
      timeoutSeconds: 80
      periodSeconds: 60
      successThreshold: 1
      failureThreshold: 5
    readinessProbe:
       exec:
          command:
          - /usr/local/bin/is_healthy.sh
       initialDelaySeconds: 10
       timeoutSeconds: 40
       periodSeconds: 30
       successThreshold: 2
       failureThreshold: 5
```

Kubernetes uses very similar concept to Deployment Configurations called `Deployment`.

It is also highly recommended to adjust the JGroups stack to discover new nodes (or leaves) more quickly. One should at least adjust the value of `FD_ALL` timeout and adjust it to the longest GC Pause.

Other hints for tuning configuration parameters are:

- OpenShift should replace running nodes one by one. This can be achieved by adjusting `rollingParams` (`maxUnavailable: 1` and `maxSurge: 1`).
- Depending on the cluster size, one needs to adjust `updatePeriodSeconds` and `intervalSeconds`. The bigger cluster size is, the bigger those values should be used.
- When using Initial State Transfer, the `initialDelaySeconds` value for both probes should be set to higher value.
- During Initial State Transfer nodes might not respond to probes. The best results are achieved with higher values of `failureThreshold` and `successThreshold` values.

1.2. Performing Rolling Upgrades on Kubernetes

Rolling upgrades migrate data from one Infinispan cluster to another.

For both Kubernetes and OpenShift, the rolling upgrade procedure is nearly identical to the procedure for Infinispan server rolling upgrades.

Differences in rolling upgrade procedures

- Depending on configuration, it is a good practice to use OpenShift Routes or Kubernetes Ingress API to expose services to the clients. During the upgrade the Route (or Ingress) used by the clients can be altered to point to the new cluster.
- Invoking CLI commands can be done by using Kubernetes (`kubectl exec`) or OpenShift clients (`oc exec`). Here is an example: `oc exec <POD_NAME> -- '/opt/jboss/infinispan-server/bin/ispn-cli.sh' '-c' '--controller=$(hostname -i):9990' '/subsystem=datagrid-infinispan/cache-container=clustered/distributed-cache=default:disconnect-source(migrator-name=hotrod)'`

Key differences when upgrading using the library mode:

- Client application needs to expose JMX. It usually depends on application and environment type but the easiest way to do it is to add the following switches into the Java boostrap script `-Dcom.sun.management.jmxremote -Dcom.sun.management.jmxremote.port=<PORT>`.
- Connecting to the JMX can be done by forwarding ports. With OpenShift this might be achieved by using `oc port-forward` command whereas in Kubernetes by `kubectl port-forward`.

The last step in the Rolling Upgrade (removing a Remote Cache Store) needs to be performed differently. We need to use Kubernetes/OpenShift Rolling update command and replace Pods configuration with the one which does not contain Remote Cache Store.

A detailed instruction might be found in ISPN-6673 ticket.