Deploy Opendaylight using Helm charts

- Anil Belur
- Andrew Hsia
- Luis Gomez
- Rahul Sharma

@OpenDaylightSDN @askb23
Agenda

• Overview
• Use cases and motivation
• Opendaylight Docker Container
• Opendaylight Helm Charts
• Opendaylight on K8s clusters
• Opendaylight Clustering
• CI Integration and deployment (JJB Templates)
• Future enhancements and use cases
• Improvements
• Discussion
Overview

- Deploy OpendayLight (ODL) as a docker container.
- ODL Container releases are available on Sonatype Nexus 3 (since Silicon SR2 14.2 release).
- Helm charts are created to deploy the ODL container on Kubernetes (K8s) clusters.
- Using COE (Cloud Orchestration Engine), Openstack Magnum API.
- CI/CD jobs templates.
Use cases and motivation

- ODL presence on the microservices platform.
- Learnability for new users (containers, microk8s, docker compose, multi-node K8s).
- Deploy ODL in K8S environment which makes the ODL service seamless and available.
- Ease the deployment of the ODL cluster.
• Base image is `openjdk:11-jdk-slim` (429 MiB).
• Docker build off existing ODL distribution release artifacts (400 MiB uncompressed) 
  `opendaylight-${ODL_VERSION}.tar.gz`
• Combined image was ~1 GiB
• jlink is used to remove redundant java modules.
• Final image size ~512 MiB.
• Override default ODL feature (odl-restconf) by passing an ENV variable “FEATURES”
Start an ODL instance (Silicon SR2) with docker:

```bash
# docker login nexus3.opendaylight.org:10001
# docker pull nexus3.opendaylight.org:10001/opendaylight/opendaylight:14.2.0
# docker run -d -p 8181:8181 --env FEATURES=odl-restconf,odl-netconf-topology opendaylight:14.2.0
```
ODL Docker container

Ports:

TCP 1790 BGP
TCP 2550 CLUSTER
TCP 2830 NETCONF NB
TCP 4189 PCEP
TCP 6633 OPENFLOW
TCP 6640 OVSDB
TCP 6653 OPENFLOW
TCP 6666 NETCONF CALLHOME SERVER
TCP 8101 KARAF SHELL
TCP 8181 RESTCONF
TCP 8185 WEBSOCKETS
TCP 12345 BMP
UDP 4342 LISP
Operationalize ODL in Production

- Containerize ODL provides a consistent runtime environment, but...
- No self-healing if hosting OS partition fills up
- Can’t recover if other processes outside the ODL consumes all physical memory.
- Deterministic hardware failure.
- Non-deterministic hardware failure.
- Running a hybrid environment, ODL on standalone VM and rest of business API in Kubernetes.
ODL Helm Charts

› Goal is to support stateful ODL deployment on Kubernetes.
› Define a standard set of configurable parameter for ODL.
› Customize using values.yaml or using --set flag.
   › Tunable Java memory options and GC options.
   › Support persistence volume storage class or emptyDir volume.
› Different version of ODL docker images.
› Image pull secret.
› ODL features.
› Node selector and affinity.
› When using remote persistence volume (e.g. ceph storage class), ODL can recover on failed worker node by kubernetes on different node over same volume.
› Stateful Set, with predictable pod name during scaling.
ODL Helm Charts

Create single ODL instance

- helm % helm install sdn-controller opendaylight
- NAME: sdn-controller
- LAST DEPLOYED: Sun Jan 9 23:54:28 2022
- NAMESPACE: default
- STATUS: deployed
- REVISION: 1
- TEST SUITE: None

NOTES:
1. Get the application URL by running these commands:
   export POD_NAME=$(kubectl get pods --namespace default -l "app.kubernetes.io/name=opendaylight,app.kubernetes.io/instance=sdn-controller" -o jsonpath="{.items[0].metadata.name}")
   export CONTAINER_PORT=$(kubectl get pod --namespace default $POD_NAME -o jsonpath="{.spec.containers[0].ports[0].containerPort}")
   echo "Visit http://127.0.0.1:$CONTAINER_PORT to use your application"
   kubectl --namespace default port-forward $POD_NAME 8080:$CONTAINER_PORT

Uninstall ODL instance

- helm % helm list
- NAME | NAMESPACE | REVISION | UPDATED | STATUS
- sdn-controller | default | 1 | 2022-01-09 23:54:28.821148 -0500 EST | deployed
- helm %
- helm % helm uninstall sdn-controller
- release "sdn-controller" uninstalled
- helm %
ODL Clustering

Motivation:
- Deploying ODL in a Kubernetes cluster for Higher Availability.
- Ease of deploying a scaled-up ODL cluster by updating replica-count in values.yaml

Opendaylight uses Akka based Clustering between different ODL instances. However setting up a clustering environment in ODL takes manual effort to configure different instances inside a cluster. For details, see here.

Using Helm and Kubernetes, the process is now automated by deploying various ODL instances as pods running inside Kubernetes environment in an Akka based cluster.

This is accomplished by configuring the cluster dynamically on deployment.

- List of nodes is determined using the service FQDN and relying on Kubernetes statefulset to generate consecutive ordinal numbers for each of the replica's created. This list is then passed on to the configure_cluster script to configure each instance during deployment.
- akka.conf is also dynamically updated with the unique ‘member-IDs’ and ‘seed-nodes’ based on each of the pods (ODL nodes) unique FQDN
- Each karaf instance is then restarted to take into account this updated configuration.
To install ODL-cluster

```
$ helm install sdnc opendaylight --set replicaCount=3
   --set config.isClusterDeployment=true --set autoscaling.enabled=false
```

```
kubectl get pods
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>READY</th>
<th>STATUS</th>
<th>RESTARTS</th>
<th>AGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>sdnc-opendaylight-1</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>25m</td>
</tr>
<tr>
<td>sdnc-opendaylight-2</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>25m</td>
</tr>
<tr>
<td>sdnc-opendaylight-0</td>
<td>1/1</td>
<td>Running</td>
<td>0</td>
<td>25m</td>
</tr>
</tbody>
</table>

To uninstall ODL-cluster

```
$ helm uninstall sdnc
```
CI Integration - K8S deploy template and macro

› CI - Jenkins, Nexus3 (Docker and Helm Repo)
› Openstack Magnum API and JJB
› JJB templates are built over the K8s cluster templates
› Released artifacts (Containers and Helm Charts) are available on ODL Nexus 3.
› Job templates are designed with the flexibility to create a K8s cluster setup and pass a custom script from the job definition. The custom script can contain the required test that can be run on the pod running ODL.
CI Jobs - ODL Docker

ODL Docker Jobs:
Templates: global-jjb/jjb/lf-docker-jobs
- "{project-name}-gerrit-docker-jobs"
- "{project-name}-gerrit-release-jobs"
Jobs: releng/builder/jjb/packaging
- packaging-odl-docker
ODL Helm Charts K8S templates and Jobs:

- **Template**: jjb/packaging/openstack-k8s.yaml
  - `{project-name}`-k8s-odl-deploy-test (uses K8S cluster templates k8s-1.8.1)
    - lf-k8s-cluster-create-with-template
    - lf-k8s-cluster-deploy
    - lf-k8s-cluster-cleanup
- **Jobs**: jjb/packaging/
  - "packaging-k8s"
    - Inputs: helm version, master count/type, node count/type, container registry, deploy script.
- **Test job**:
CI - Scope of improvement

• Allow configuration script to be passed to the cluster setup to work with different use cases.
• Documentation
• Tox for helm charts
• Migrate the job templates to releng/global-jjb
Future use cases

- Migrate some CSIT (2node, 3node tests) to work with ODL docker/K8s cluster setup.
- Dynamic scalability of ODL.
Resources

ODL containers and Helm Charts source:
https://git.opendaylight.org/gerrit/integration/packaging

Jenkins CI Jobs (jjb/packaging)
https://git.opendaylight.org/gerrit/releng/builder

Openstack Magnum API
https://docs.openstack.org/magnum/latest/user

ODL Docker/Helm Charts weekly meeting (1:30pm-2:30pm PST)
https://lists.opendaylight.org/g/app-dev/message/977