OpenSDS Bali POC Deployment

January 2019

Author: OpenSDS

Document Revision History

Version	Date	Comments
0.1	7/12/2018	Initial revision.
0.2	12/22/2018	Updated to Bali version.
0.3	1/23/2019	Add ports info
0.4	1/24/2019	Updated ports info

Related Documents

Author	Documents

Table of Contents

1 System requirements

1.1 Hardware

The hardware requirements are described in this section.

For array-based replication, two physical servers and two Dorado arrays are needed.

For host-based replication, two physical servers are needed.

For other tests described in this POC, one physical server or one VM can be used for basic testing.

1.2 Software

The software requirements are described in this section.

1.2.1 OS

Ubuntu 16.04.3 has been used during the testing and therefore should be used in this POC:

```
root@proxy:~# cat /etc/issue
Ubuntu 16.04.3 LTS \n \l
```

For user of OS, please use root user to install OpenSDS.

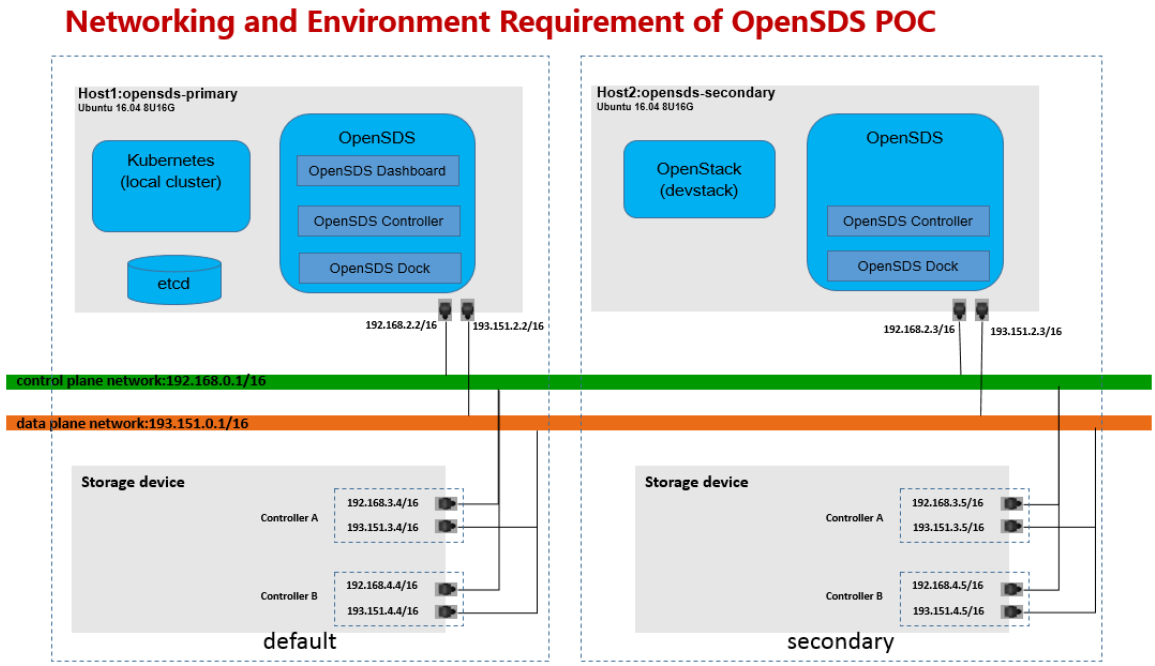
For host-based replication, required DRBD software is described in the relevant section later. Other required software is described in the installation section.

2 Installation

To test Kubernetes, OpenStack, Host-based or Array-based Replication, this section describes how to install OpenSDS in two host nodes. See the following diagram for a summary of deployment.

In the following installation process, we will refer to some of the information in the diagram (such as host name, host IP address, storage IP address, etc.) to express the installation process in more detail. In order to manage storage device, in OpenSDS you only need to configure one of the controllers of storage device if it has two controllers (such as Controller A and B). When you install OpenSDS, please refer to the actual networking. The distributed storage (such as Ceph,

FusionStorage) has different IP network configuration, please refer to your actual environment.



2.1 Prerequisite for two hosts

2.1.1 Packages

Install following packages:

```
apt-get install -y git curl wget libltdl7 libseccomp2 librados2 ceph-common
```

2.1.2 Golang

You can install golang by executing commands blow:

```
wget https://storage.googleapis.com/golang/go1.11.2.linux-amd64.tar.gz
tar -C /usr/local -xzf go1.11.2.linux-amd64.tar.gz
echo 'export PATH=$PATH:/usr/local/go/bin' >> /etc/profile
echo 'export GOPATH=$HOME/gopath' >> /etc/profile

mkdir -p ~/gopath
source /etc/profile
```

Check golang version information:

```
root@proxy:~# go version
go version go1.11.2 linux/amd64
```

2.1.3 **Docker**

Install docker:

```
# Download and install docker ce 18.03
wget
https://download.docker.com/linux/ubuntu/dists/xenial/pool/stable/amd64/docker-ce_18.03.1~ce-0~ubuntu_amd64.deb
dpkg -i docker-ce_18.03.1~ce-0~ubuntu_amd64.deb
```

Version information:

```
root@opensds-primary:~# docker version
Client:
 Version:      18.03.1-ce
 API version:  1.37
 Go version:   go1.9.5
 Git commit:   9ee9f40
 Built:        Thu Apr 26 07:17:20 2018
 OS/Arch:      linux/amd64
 Experimental: false
 Orchestrator: swarm

Server:
 Engine:
  Version:      18.03.1-ce
  API version:  1.37 (minimum version 1.12)
  Go version:   go1.9.5
  Git commit:   9ee9f40
  Built:        Thu Apr 26 07:15:30 2018
  OS/Arch:      linux/amd64
  Experimental: false
```

2.1.4 **Docker-compose**

Install docker-compose:

```
curl -L "https://github.com/docker/compose/releases/download/1.23.1/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

chmod +x /usr/local/bin/docker-compose
```

Version information:

```
docker-compose version
docker-compose version 1.23.1, build b02f1306
docker-py version: 3.5.0
CPython version: 3.6.7
OpenSSL version: OpenSSL 1.1.0f  25 May 2017
```

2.1.5 **Ansible**

To install ansible, run the commands below:

```
add-apt-repository ppa:ansible/ansible-2.4

apt-get update
```

```
apt-get install -y ansible

# Check ansible version, 2.4.x is required.
ansible --version
```

2.1.6 DRBD

Run command blow to install drbd service:

```
add-apt-repository ppa:linbit/linbit-drbd9-stack
apt-get update
apt-get install drbd-utils python-drbdmanage drbd-dkms
```

Check verion information:

```
drbdmanage --version
drbdmanage 0.99.18; GIT-hash: 2bca8c7874462285e1b499c662e52a66f3844403
```

2.2 Deployment on host 1

2.2.1 OpenSDS Deployment

In this section, the steps to deploy an OpenSDS local cluster are described.

2.2.1.1 Download opensds-installer code

```
cd $HOME
git clone -b stable/bali https://github.com/opensds/opensds-installer.git
cd opensds-installer/ansible
```

2.2.1.2 Configure OpenSDS cluster variables

2.2.1.2.1 System environment

Change host_ip to the actual IP address (e.g. 192.168.2.2) in `group_vars/common.yml`:

```
# This field indicates local machine host ip

host_ip: 127.0.0.1
```

Modify the host and port of etcd in `group_vars/osdsdb.yml`. To eliminate the conflict with kubernetes, you should change the etc_host to actual IP address (e.g. 192.168.2.2) and change etcd_port and etcd_peer_port to new ports, such as 2479 and 2480.

```
etcd_host: 192.168.2.2
etcd_port: 2479
etcd_peer_port: 2480
```

2.2.1.2.2 LVM

If lvm is chosen as the storage backend, there is no need to modify `group_vars/osdsdock.yml` because it is the default choice:

```
enabled_backend: lvm # Change it according to the chosen backend. Supported
backends include 'lvm', 'ceph', and 'cinder'
```

Change `tgtBindIp` variable in `group_vars/lvm/lvm.yaml` to your real host IP address (e.g. 192.168.2.2):

```
tgtBindIp: 127.0.0.1 # change tgtBindIp to your real host ip, run 'ifconfig' to
check
```

2.2.1.2.3 Ceph

If `ceph` is chosen as storage backend, modify `group_vars/osdsdock.yml`:

```
enabled_backend: ceph # Change it according to the chosen backend. Supported
backends include 'lvm', 'ceph', and 'cinder'.
```

Configure group_vars/ceph/all.yml with an example below:

```
ceph_origin: repository
ceph_repository: community
ceph_stable_release: luminous # Choose luminous as default version
public_network: "192.168.3.0/24" # Run 'ip -4 address' to check the ip address
cluster_network: "{{ public_network }}"
monitor_interface: eth1 # Change to the network interface on the target machine
devices: # For ceph devices, append ONE or MULTIPLE devices like the example below:
  - '/dev/sda' # Ensure this device exists and available if ceph is chosen
  #- '/dev/sdb'  # Ensure this device exists and available if ceph is chosen
osd_scenario: collocated
```

2.2.1.2.4 Cinder

If cinder is chosen as storage backend, modify `group_vars/osdsdock.yml`:

```
enabled_backend: cinder # Change it according to the chosen backend. Supported
backends include 'lvm', 'ceph', and 'cinder'
# Use block-box install cinder_standalone if true, see details in:
use_cinder_standalone: true
```

Configure the auth and pool options to access cinder in `group_vars/cinder/cinder.yaml`. Do not need to make additional configure changes if using cinder standalone.

2.2.1.3 Check if the hosts can be reached

```
ansible all -m ping -i local.hosts
```

2.2.1.4 Run opensds-ansible playbook to start deploy

```
ansible-playbook site.yml -i local.hosts
```

2.2.2 Configure FusionStorage Backend

FusionStorage installation has not been integrated into the opensds-installer yet, if you want to test the OpenSDS using FusionStorage as the backend, you should configure it manually.

1. Add FusionStorage backend configuration to `/etc/opensds/opensds.conf`

vim `/etc/opends/opensds.conf`

```
[osdsdock]
# ...
enabled_backends = huawei_fusionstorage
# ...

[huawei_fusionstorage]
name = fusionstorage backend
description = This is a fusionstorage backend service
driver_name = huawei_fusionstorage
config_path = /etc/opensds/driver/fusionstorage.yaml
```

2. Add FusionStorage backend configuration to `/etc/opensds/driver/dorado.yaml`, please confirm whether the fmIp and fsaIp are correct.

```
authOptions:
  fmIp: 192.168.3.4
  fsaIp:
    - 192.168.3.4

pool:
  0:
    storageType: block
    availabilityZone: default
    extras:
      dataStorage:
        provisioningPolicy: Thin
        isSpaceEfficient: false
```

```
      ioConnectivity:
        accessProtocol: DSWARE
        maxIOPS: 7000000
        maxBWS: 600
      advanced:
        diskType: SSD
        latency: 3ms
```

3. After configuring, restart the osdsdock.

```
# Check osdsdock
ps -ef | grep osdsdock
# If it exists, kill all and restart
killall osdsdock
/opt/opensds-hotpot-linux-amd64/bin/osdsdock --daemon
# Check osdsdock if it exists.
ps -ef | grep osdsdock
```

2.2.3 Configure Dorado Storage Backend

Dorado installation has not been integrated into the opensds-installer yet, if you want to test the OpenSDS using dorado as the backend, you should configure it manually.

4. Add dorado backend configuration to `/etc/opensds/opensds.conf`

vim `/etc/opends/opensds.conf`

```
[osdsdock]
# ...
enabled_backends = huawei_dorado
host_based_replication_driver = drbd
# ...

[huawei_dorado]
name = huawei_dorado
description = Huawei OceanStor Dorado
driver_name = huawei_dorado
config_path = /etc/opensds/driver/dorado.yaml
# OpenSDS will support array-based replication when support_replication = true,
# OpenSDS will support host-based replication when support_replication = false
support_replication = true
```

5. Add dorado backend configuration to `/etc/opensds/driver/dorado.yaml`

```
authOptions:
  endpoints: "https://196.168.3.4:8088/deviceManager/rest"
  username: "admin"
  password: "Huawei12#$"
  insecure: true
replication:
  authOptions:
    endpoints: "https://196.168.3.5:8088/deviceManager/rest"
    username: "admin"
    password: "Huawei12#$"
    insecure: true

pool:
# Dorado pool that you want to provide to opensds.
  StoragePool001:
    storageType: block
    availabilityZone: default
    extras:
      dataStorage:
        provisioningPolicy: Thin
        isSpaceEfficient: false
      ioConnectivity:
        accessProtocol: iscsi
        maxIOPS: 7000000
        maxBWS: 600
# The ETH ip that you configure in for iSCSI.
targetIp: 193.151.3.4
```

If you want change the storage networking protcol to FC, please replace the **iscsi** with **fibre_channel.**

6. After configuring, restart the osdsdock.

```
# Check osdsdock
ps -ef | grep osdsdock
# If it exists, kill all and restart
killall osdsdock
/opt/opensds-hotpot-linux-amd64/bin/osdsdock --daemon
# Check osdsdock if it exists.
ps -ef | grep osdsdock
```

2.2.4 Configure Host-based replication

Currently, OpenSDS can only support one type of replication at a time, therefore if you want to test host-based replication please confirm the parameter 'support_replication' is false, as described in Section 2.2.2. We recommend that you configure and test host-based replication after testing array-based replication.

1. Add drbd configuration for opensds drbd driver.

```
vi /etc/opensds/drbd.yaml
```

```
#Minumum and Maximum TCP/IP ports used for DRBD replication
PortMin: 7000
PortMax: 8000
#Exactly two hosts between resources are replicated.
#Never ever change the Node-ID associated with a Host(name)
Hosts:
  - Hostname: opensds-secondary # hostname of host 2
    IP: 192.168.2.3
    Node-ID: 1
  - Hostname: opensds-primary # hostname of host 1
    IP: 192.168.2.2
    Node-ID: 0
```

2. Add `/etc/opensds/attacher.conf`

```
[osdsdock]
api_endpoint = 192.168.2.2:50051
log_file = /var/log/opensds/osdsdock.log
bind_ip = 192.168.2.2
dock_type = attacher

[database]
endpoint = 192.168.2.2:2479, 192.168.2.2:2480
driver = etcd
```

3. Startup the osdsdock(attacher)

```
# Start attacher osdsdock
/opt/opensds-hotpot-linux-amd64/bin/osdsdock --config-file
/etc/opensds/attacher.conf --daemon
# Check attacher osdsdock
ps -ef | grep osdsdock
```

2.2.5 Kubernetes Local Cluster Deployment

2.2.5.1 Install Etcd

You can install etcd by executing commands blow:

```
cd $HOME
wget https://github.com/coreos/etcd/releases/download/v3.3.0/etcd-v3.3.0-linux-amd64.tar.gz
tar -xzf etcd-v3.3.0-linux-amd64.tar.gz
cd etcd-v3.3.0-linux-amd64
sudo cp -f etcd etcdctl /usr/local/bin/
```

2.2.5.2 kubernetes local cluster

You can start the latest k8s local cluster by executing commands blow:

```
cd $HOME
git clone https://github.com/kubernetes/kubernetes.git
cd $HOME/kubernetes
git checkout v1.13.0
make
echo alias kubectl='$HOME/kubernetes/cluster/kubectl.sh' >> /etc/profile

ALLOW_PRIVILEGED=true
FEATURE_GATES=CSIPersistentVolume=true,MountPropagation=true,VolumeSnapshotDataSource=true,KubeletPluginsWatcher=true RUNTIME_CONFIG="storage.k8s.io/v1alpha1=true"
LOG_LEVEL=5 hack/local-up-cluster.sh
```

2.2.6 CSI Plugin Deployment

2.2.6.1 Download nbp source code

```
git clone -b stable/bali https://github.com/opensds/nbp.git
cd nbp
```

2.2.6.2 Configure CSI Plugin configmap

Configure OpenSDS endpoint and Keystone IP address.

```
vi csi/server/deploy/kubernetes/csi-configmap-opensdsplugin.yaml
```

The IP (127.0.0.1) should be replaced with the opensds and identity actual endpoint IP (e.g. 192.168.2.2).

```
kind: ConfigMap
apiVersion: v1
    metadata:
name: csi-configmap-opensdsplugin
    data:
    opensdsendpoint: http://127.0.0.1:50040
    osauthurl: http://127.0.0.1/identity
```

2.2.6.3 Install CSI Plugin

Use kubectl command to create OpenSDS CSI pods:

```
kubectl create -f csi/server/deploy/kubernetes
```

After this, three pods can be found by `kubectl get pods` like below:

- `csi-provisioner-opensdsplugin`
- `csi-attacher-opensdsplugin`
- `csi-nodeplugin-opensdsplugin`
- `csi-snapshotter-opensdsplugin`

2.3 Deployment on host 2

2.3.1 OpenSDS Deployment

In this section, the steps to deploy an OpenSDS local cluster are described.

2.3.1.1 Download opensds-installer code

```
cd $HOME
git clone -b stable/bali https://github.com/opensds/opensds-installer.git
cd opensds-installer/ansible
```

2.3.1.2 Configure OpenSDS cluster variables

2.3.1.2.1 System environment

Change host_ip to the actual IP address (e.g. 192.168.2.3) in `group_vars/common.yml`:

```
# This field indicates local machine host ip

host_ip: 127.0.0.1
```

Then, set opensds_auth_strategy as noauth in `group_vars/auth.yml`

```
# OpenSDS authentication strategy, support 'noauth' and 'keystone'.
opensds_auth_strategy: noauth
```

Modify the host and port of etcd in `group_vars/osdsdb.yml`. To eliminate the conflict with kubernetes, you should change the etc_host to actual IP address of **Host 1** (e.g. 192.168.2.2) and change etcd_port and etcd_peer_port to new ports, such as 2479 and 2480.

```
etcd_host: 192.168.2.2
etcd_port: 2479
```

```
etcd_peer_port: 2480
```

2.3.1.2.2 LVM

If lvm is chosen as the storage backend, there is no need to modify `group_vars/osdsdock.yml` because it is the default choice:

```
enabled_backend: lvm # Change it according to the chosen backend. Supported
backends include 'lvm', 'ceph', and 'cinder'
```

Change `tgtBindIp` variable in `group_vars/lvm/lvm.yaml` to your real host IP address (e.g. 192.168.2.3):

```
tgtBindIp: 127.0.0.1 # change tgtBindIp to your real host ip, run 'ifconfig' to
check
```

2.3.1.2.3 Ceph

If `ceph` is chosen as storage backend, modify `group_vars/osdsdock.yml`:

```
enabled_backend: ceph # Change it according to the chosen backend. Supported backends
include 'lvm', 'ceph', and 'cinder'.

Configure group_vars/ceph/all.yml with an example below:
ceph_origin: repository
ceph_repository: community
ceph_stable_release: luminous # Choose luminous as default version
public_network: "192.168.3.0/24" # Run 'ip -4 address' to check the ip address
cluster_network: "{{ public_network }}"
monitor_interface: eth1 # Change to the network interface on the target machine
devices: # For ceph devices, append ONE or MULTIPLE devices like the example below:
  - '/dev/sda' # Ensure this device exists and available if ceph is chosen
  #- '/dev/sdb'  # Ensure this device exists and available if ceph is chosen
osd_scenario: collocated
```

2.3.1.2.4 Cinder

If cinder is chosen as storage backend, modify `group_vars/osdsdock.yml`:

```
enabled_backend: cinder # Change it according to the chosen backend. Supported
backends include 'lvm', 'ceph', and 'cinder'
# Use block-box install cinder_standalone if true, see details in:
use_cinder_standalone: true
```

Configure the auth and pool options to access cinder in `group_vars/cinder/cinder.yaml`. Do not need to make additional configure changes if using cinder standalone.

2.3.1.3 Check if the hosts can be reached

```
ansible all -m ping -i local.hosts
```

2.3.1.4 Run opensds-ansible playbook to start deploy

```
ansible-playbook site.yml -i local.hosts
```

2.3.2 Configure FusionStorage Backend

FusionStorage installation has not been integrated into the opensds-installer yet, if you want to test the OpenSDS using FusionStorage as the backend, you should configure it manually.

1. Add FusionStorage backend configuration to `/etc/opensds/opensds.conf`

vim `/etc/opends/opensds.conf`

```
[osdsdock]
# ...
enabled_backends = huawei_fusionstorage
# ...

[huawei_fusionstorage]
name = fusionstorage backend
description = This is a fusionstorage backend service
driver_name = huawei_fusionstorage
config_path = /etc/opensds/driver/fusionstorage.yaml
```

2. Add FusionStorage backend configuration to `/etc/opensds/driver/dorado.yaml`, please confirm whether the fmIp and fsaIp are correct.

```
authOptions:
  fmIp: 192.168.3.5
  fsaIp:
    - 192.168.3.5

pool:
  0:
    storageType: block
    availabilityZone: default
    extras:
      dataStorage:
        provisioningPolicy: Thin
        isSpaceEfficient: false
```

```
      ioConnectivity:
        accessProtocol: DSWARE
        maxIOPS: 7000000
        maxBWS: 600
      advanced:
        diskType: SSD
        latency: 3ms
```

2.3.3 Configure Dorado Storage Backend

Dorado installation has not been integrated into the opensds-installer yet, if you want to test the OpenSDS using dorado as the backend, you should configure it manually.

1. Add dorado backend configuration to `/etc/opensds/opensds.conf`

vim `/etc/opends/opensds.conf`

```
[osdsdock]
# ...
enabled_backends = huawei_dorado
host_based_replication_driver = drbd
# ...

[huawei_dorado]
name = huawei_dorado
description = Huawei OceanStor Dorado
driver_name = huawei_dorado
config_path = /etc/opensds/driver/dorado.yaml
# OpenSDS will support array-based replication when support_replication = true,
# OpenSDS will support host-based replication when support_replication = false
support_replication = true
```

2. Add dorado backend configuration to `/etc/opensds/driver/dorado.yaml`

```
authOptions:
  endpoints: "https://196.168.3.5:8088/deviceManager/rest"
  username: "admin"
  password: "Huawei12#$"
  insecure: true
replication:
  authOptions:
    endpoints: "https://196.168.3.4:8088/deviceManager/rest"
    username: "admin"
    password: "Huawei12#$"
    insecure: true
```

```
pool:
# Dorado pool that you want to provide to opensds.
  StoragePool001:
    storageType: block
    availabilityZone: default
    extras:
      dataStorage:
        provisioningPolicy: Thin
        isSpaceEfficient: false
      ioConnectivity:
        accessProtocol: iscsi
        maxIOPS: 7000000
        maxBWS: 600
# The ETH ip that you configure in for iSCSI.
targetIp: 193.151.3.5
```

If you want change the storage networking protcol to FC, please replace the **iscsi** with **fibre_channel**.

3. After configuring, restart the osdsdock.

```
# Check osdsdock
ps -ef | grep osdsdock
# If it exists, kill all and restart
killall osdsdock
/opt/opensds-hotpot-linux-amd64/bin/osdsdock --daemon
# Check osdsdock if it exists.
ps -ef | grep osdsdock
```

2.3.4 Configure Host-based replication

Currently, OpenSDS can only support one type of replication at a time, therefore if you want to test host-based replication please confirm the parameter 'support_replication' is false, as described in Section 2.3.2. We recommend that you configure and test host-based replication after testing array-based replication.

1. Add drbd configuration for opensds drbd driver.

```
vi /etc/opensds/drbd.yaml
```

```
#Minumum and Maximum TCP/IP ports used for DRBD replication
PortMin: 7000
PortMax: 8000
#Exactly two hosts between resources are replicated.
```

```
#Never ever change the Node-ID associated with a Host(name)
Hosts:
  - Hostname: opensds-secondary # hostname of host 2
    IP: 192.168.2.3
    Node-ID: 1
  - Hostname: opensds-primary # hostname of host 1
    IP: 192.168.2.2
    Node-ID: 0
```

2. Add `/etc/opensds/attacher.conf`

```
[osdsdock]
api_endpoint = 192.168.2.3:50051
log_file = /var/log/opensds/osdsdock.log
bind_ip = 192.168.2.3
dock_type = attacher

[database]
endpoint = 192.168.2.2:2479, 192.168.2.2:2480
driver = etcd
```

3. Startup the osdsdock(attacher)

```
# Start attacher osdsdock
/opt/opensds-hotpot-linux-amd64/bin/osdsdock --config-file
/etc/opensds/attacher.conf --daemon
# Check attacher osdsdock
ps -ef | grep osdsdock
```

2.3.5 Devstack(Openstack) Deployment

2.3.5.1 Install OpenStack using devstack

The Cinder Compatible API only supports cinder's current Api(v3). You can use devstack to install cinder when testing, but in order to use cinder's current Api(v3), branch for devstack must be stable/queens.

You can reference this document to install devstack(OpenStack).
https://docs.openstack.org/devstack/latest/

2.3.6 Configure Cinder Compatible API in OpenSDS

Cinder Compatible API adapter is not built in as part of the ansible deployment tool. Please confirm that OS user is root, not stack. Follow the following instruction to install it.

2.3.6.1 Installation

1. Initialize OpenStack environment variables.

```
source /opt/stack/devstack/openrc admin admin
export OS_VOLUME_API_VERSION=3
```

2. Change the "cinderv3" endpoint so that OpenStack can access the cinder compatible api. Then export OPENSDS_ENDPOINT and CINDER_ENDPOINT.

```
# Find the <endpoint-id> of cinderv3
openstack endpoint list
# Update cinderv3 endpoint, the ip is the actual ip address of host2
openstack endpoint set <endpoint-id> --url 'http://
192.168.2.3:8777/v3/$(project_id)s'

# Export OPENSDS_ENDPOINT and CINDER_ENDPOINT
export OPENSDS_ENDPOINT=http://192.168.2.3:50040
export CINDER_ENDPOINT=http://192.168.2.3:8777/v3
```

3. Compatible api code, build and run it.

```
cd $GOPATH/src/github.com/opensds/opensds

# Building
go build -o ./build/out/bin/cindercompatibleapi
github.com/opensds/opensds/contrib/cindercompatibleapi

# Run cinder compatible api service
setsid ./build/out/bin/cindercompatibleapi
```

4. Check cinder compatible api.

```
cinder type-list
```

2.4 Check OpenSDS

2.4.1 Check OpenSDS CLI Tool

Configure OpenSDS CLI tool on **host 1**:

```
cp /opt/opensds-hotpot-linux-amd64/bin/osdsctl /usr/local/bin
export OPENSDS_ENDPOINT=http://{your_real_host_ip}:50040
export OPENSDS_AUTH_STRATEGY=keystone
source /opt/stack/devstack/openrc admin admin
osdsctl pool list # Check if the pool resource is available
```

Create a default profile:

```
osdsctl profile create '{"name": "default", "description": "default policy"}'
```

Create a volume:

```
osdsctl volume create 1 --name=test-001
```

List all volumes:

```
osdsctl volume list
```

Delete the volume:

```
osdsctl volume delete <your_volume_id>
```

2.4.2 Check OpenSDS Dashboard

OpenSDS UI dashboard is available at *http://{your_host1_ip}:8088* on **host1**, please login the dashboard using the default admin credentials: *admin/opensds@123*. Create tenant, user, and profiles as admin. Multi-Cloud service is also supported by dashboard.

Logout of the dashboard as admin and login the dashboard again as a non-admin user to manage storage resource:

- Volume Service
- Create volume
- Create snapshot
- Expand volume size
- Create volume from snapshot
- Create volume group

Multi Cloud Service

- Register object storage backend
- Create bucket
- Upload object
- Download object
- Migrate objects based on bucket across cloud

3 Uninstallation

Log in to the two hosts, respectively, do the following steps:

Run opensds-ansible playbook to clean the environment.

```
cd opensds-installer/ansible
ansible-playbook clean.yml -i local.hosts
```

Run ceph-ansible playbook to clean ceph cluster if ceph is deployed (optional).

```
cd /opt/ceph-ansible
sudo ansible-playbook infrastructure-playbooks/purge-cluster.yml -i ceph.hosts
```

In addition, clean up the logical partition on the physical block device used by ceph, using the fdisk tool. And remove ceph-ansible source code (optional).

```
sudo rm -rf /opt/ceph-ansible
```

4 Appendix

4.1.1 Port matrix used in OpenSDS

Port	Service	Description	config file
50040	osdslet	API server default port number	path: opensds-installer/ansible/group_vars/hotpot.yml item: controller_endpoint: "{{ host_ip }}:50040"

50050	osdsdock	Osdsdock grpc default port nubmer	path: opensds-installer/ansible/group_vars/hotpot.yml item: dock_endpoint: localhost:50050
2379	etcd	ETCD port	path: opensds-installer/ansible/group_vars/osdsdb.yml item: etcd_port: 2379
2380	etcd	ETCD port	path: opensds-installer/ansible/group_vars/osdsdb.yml item: etcd_peer_port: 2380
8088	dashbord	Dashbord service port using nginx	Not support
80	keystone	Keystone service port using apache	Not support
11211	memcache	Memcache service which is used by keystone.	Not support
3306	mysql	Memcache service which is used by keystone.	Not support
3260	iscsi-target	If you use lvm as the	Not support

		storage backend, iscsi-target will be used to providing volume to host.	
8089	multi-cloud-api	Mult-Cloud API server default port number	Not support
27017	mongodb	Mongodb port nubmer which is used by multi-cloud	Not support
9092	kafka	Kafka port nubmer which is used by multi-cloud	Not support
2181	zookeeper	zookeeper port nubmer which is used by kafka	Not support

8776	cinder	Cinder API service port number	Not support
5672	rabbitmq	Rabbitmq service port number which is used by cinder	Not support
8080	ceph radosgw	Ceph radosgw civetweb port	path: opensds-installer/ansible/group_vars/ceph/all.yml item: radosgw_civetweb_port: 8080
6789	ceph mon	Ceph monitor services	Not support
6800:7300	ceoh osd	Ceph OSD services bind to ports within the 6800:7300 range	Not support