dlvm Documentation

Release 0.1

yupeng

Aug 01, 2018

Contents

Contents:

CHAPTER 1

Introduce to DLVM

DLVM (Distribute Logical Volume) is a distribute storage system, intends to be a storage backend for a cloud platform, such as openstack cinder.

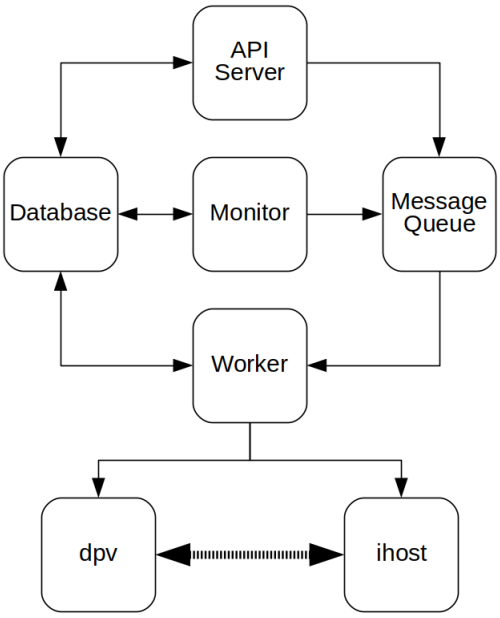
DLVM features:

- REST api interface
- supports create/delete volume, attach/detach volume, create/delete snapshot, online clone, increase volume size
- HA (provided by device mapper mirror target)
- thin provision (provided by device mapper thin provision target)
- more than 100K IOPS random write per volume

DLVM has similar concepts as traditional LVM. The traditional LVM has pv (physical volume), vg (volume group) and lv (logical volume), DLVM has the corresponding conecpts which are dpv, dvg and dlv, the characture 'd' is the abbreviate of 'distribute'. In DLVM, a dpv is a linux server which can export volume(s) through iscsi. A dvg is a group of dpvs, a dlv is a logical aggregation of dpvs in a same dvg. A dlv can be attached to an ihost(initiator host) through iscsi protocol. Below is the architecture of the dlvm service:

There are several components of the dlvm services:

- API Server: provides restful api, such as create volume, attach volume and so on. API Server will only record the request to Database, and send a message to Message Queue, a Worker will get this request and actually work on it
- Worker: accepts requests from Message Queue, sends rpc to dpv or ihost, updates the status in Database.
- dpv: storage server, runs dlvm_dpv_agent on it, accepts rpc call from API Server or Worker, creates volume or exports voluem to ihost
- ihost: the host which the dlv can be attached to, runs dlvm_ihost_agent on it, accepts rpc call from Worker.
- Monitor: runs celery beat on it, sends periodic tasks to Worker
- Database: currently only test mariadb, should be any DB supportted by SQLAlchemy
- Message Queue: rabbitmq or any message queues support by celery

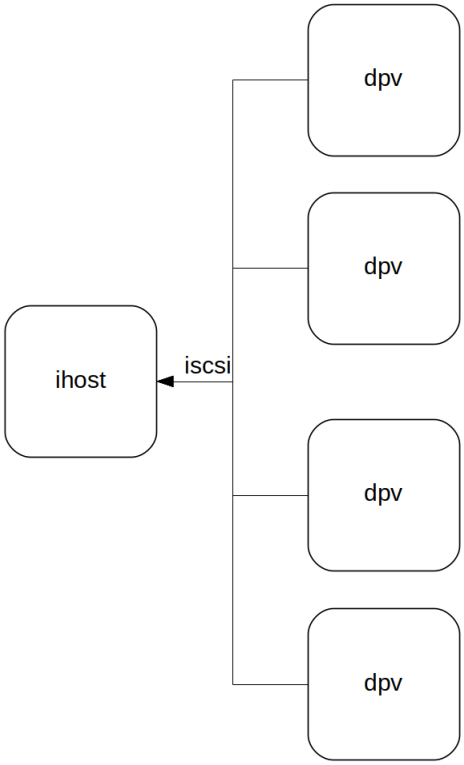


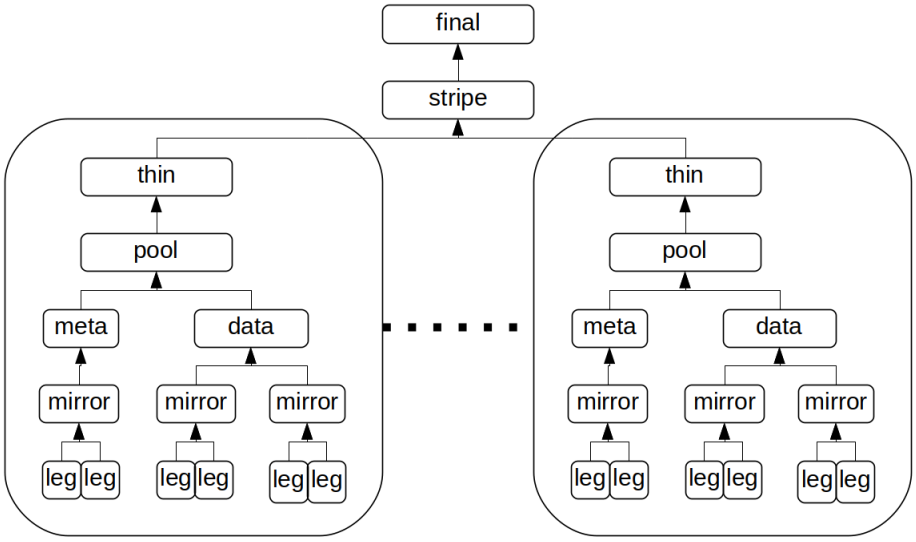
Several dpvs could be exported to an ihost via iscsi:

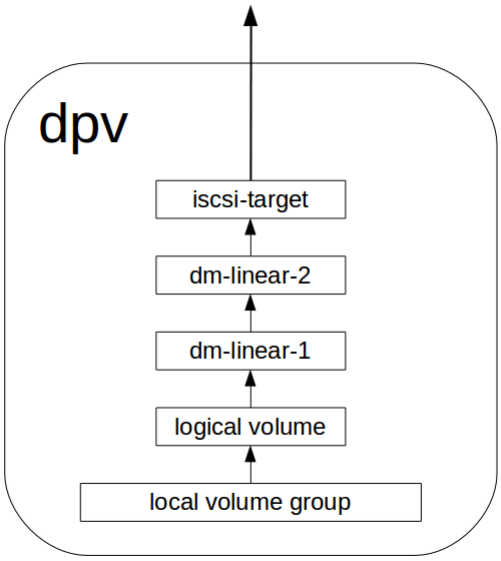
Every dpv is a 'leg' of the dlv, all legs would aggregate to a dlv. Below picture is the dlv structure when it is attached to an ihost:

The dlv is divided to several groups, a group has several legs, every leg is an iscsi device allocated from a dpv.

below picture is the internal strucutre of a leg in dpv:


dpv
dpv
ihost
iscsi
dpv
dpv


final
stripe
thin
pool
meta
data
mirror
mirror
mirror
leg
leg
leg
leg
leg
leg
thin
pool
meta
data
mirror
mirror
mirror
leg
leg
leg
leg
leg
leg


dpv
iscsi-target
dm-linear-2
dm-linear-1
logical volume
local volume group

CHAPTER 2

Install DLVM

Currently, dlvm can only work on ubuntu 16.04 system. It can be deployed by an ansible playbook. Download the playbook from here: https://github.com/dlvm/dlvm_playbook .

2.1 prepare: install ansible

Login to a ubuntu 16.04 system, install ansible:

```
$ sudo apt-get install -y software-properties-common
$ sudo apt-add-repository -y ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install -y ansible
```

2.2 all in one deployment

Download dlvm_playbook and exact it:

```
$ curl -O -L https://github.com/dlvm/dlvm_playbook/archive/master.zip
$ unzip master.zip
```

Go to the dlvm_playbook directory, prepare the ssh login credentials. and configure ssh login credentail depend on the ubuntu system.

Rename the group_vars/all.sample to group_vars/all, and edit it, set test_mode to true, set dlvm_local_lvm and dlvm_all_in_one to true

Run dlvm_playbook:

```
$ ansible-playbook -i "localhost," --user <user_name> -k --become --become-user root␣
↪control_plan.yml
$ ansible-playbook -i "localhost," --user <user_name> -k --become --become-user root -
↪e "dlvm_local_lvm=true" dpv.yml
$ ansible-playbook -i "localhost," --user <user_name> -k --become --become-user root␣
↪ihost.yml
```

If your ssh use keypair to login, please run:

```
$ ansible-playbook -i "localhost," --user <user_name> --private-key <key_path> --
↪become --become-user root control_plan.yml
$ ansible-playbook -i "localhost," --user <user_name> --private-key <key_path> --
↪become --become-user root -e "dlvm_local_lvm=true" dpv.yml
$ ansible-playbook -i "localhost," --user <user_name> --private-key <key_path> --
↪become --become-user root ihost.yml
```

The control_plan.yml will deploy api_server, worker, monitor to the server, and for easy to use, it also deploys a single node rabbitmq and a single node mariadb on localhost. The dpv.yml will deploy dlvm_dpv_agent to the server, set dlvm_local_lvm to true will let it create a local volume group from a loop device, it could be used in test mode, in production, you should configure your own logical volume. The ihost.yml will deploy dlvm_ihost_agent to the server.

By default, the api server is listen on localhost:9521, you could run below command to verify it works:

```
$ curl localhost:9521/
{"data": "dlvm", "req_id": "3662c3b1-88e0-48a3-b4b5-03ec9f5248fe", "message": "succeed
↪"}
```

Create a dpv:

```
$ curl -H "Content-Type: application/json" -X POST -d '{"dpv_name":"localhost"}'␣
↪http://localhost:9521/dpvs
```

Create a dvg:

```
$ curl -H "Content-Type: application/json" -X POST -d '{"dvg_name":"dvg0"}' http://
↪localhost:9521/dvgs
```

Add the dpv to the dvg:

```
$ curl -H "Content-Type: application/json" -X PUT -d '{"dpv_name":"localhost"}' http:/
↪/localhost:9521/dvgs/dvg0/extend
```

Create a dlv:

```
$ curl -H "Content-Type: application/json" -X POST -d '{"dlv_name":"dlv0","dlv_size
↪":1073741824,"stripe_number":1,"init_size":536870912,"dvg_name":"dvg0"}' http://
↪localhost:9521/dlvs
```

Wait for a while, check the status of dlv0, make sure it is 'available':

```
$ curl http://localhost:9521/dlvs
```

Attache the dlv0 to an ihost (localhost in our example):

```
$ curl -H "Content-Type: application/json" -X PUT -d '{"ihost_name":"localhost"}'␣
↪http://localhost:9521/dlvs/dlv0/attach
```

Wait for a while, check the status of dlv0, make sure it is 'attached', and then you could find /dev/mapper/dlvmihost-final-dlv0 on localhost:

```
$ ls /dev/mapper/dlvmihost-final-dlv0
/dev/mapper/dlvmihost-final-dlv0
```

You could use it as a normal block device.

Detache the dlv from ihost:

```
$ curl -H "Content-Type: application/json" -X PUT http://localhost:9521/dlvs/dlv0/
↪detach
```

Delete the dlv:

```
$ curl -X DELETE http://localhost:9521/dlvs/dlv0
```

Remove dpv from dvg:

```
$ curl -H "Content-Type: application/json" -X PUT -d '{"dpv_name":"localhost"}' http:/
↪/localhost:9521/dvgs/dvg0/reduce
```

Delete dpg and dpv:

```
$ curl -X DELETE http://localhost:9521/dvgs/dvg0
$ curl -X DELETE http://localhost:9521/dpvs/localhost
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search