Lab STRG1 - Virtual Block Storage

Introduction and Prerequisites

The objective of this lab session is to obtain an understanding of the operation of typical block storage systems focusing on a ZFS backend behind an iSCSI interface. As you learned in the lecture, such a configuration is representative of a real-world block storage system which can offer volume services to a cloud platform such as Openstack: a VM running on a compute host could mount a remote volume via iSCSI which could provide storage resources which are hosted on a ZFS pool.

In this lab you will:

- Acquire basic hands-on experience on with ZFS and iSCSI
- Apply and verify storage concepts such as storage efficiency and resiliency
- Implement a rudimentary block-level storage service use case, where a storage server (target machine) offers resilient block-level storage (via ZFS) to a remote client (initiator) via ISCSI.

The following resources and tools are required for this laboratory session:

- An account on the OpenStack cluster at https://ned.cloudlab.zhaw.ch
- Your project on Ned contains an image (CCP1-EN-STRG-I) which was created for this lab.
  - The user account for this image is **centos**
- Modern web browser and ssh client
- Mind, This image does not support ED25519 SSH-keys. You should use/create an RSA-key). Further, if you use a modern ssh client you may need to enable legacy ssh-rsa support. See the example below
  - ssh -o HostKeyAlgorithms=+ssh-rsa -o PubkeyAcceptedAlgorithms=+ssh-rsa centos@160.85.31.250

Time & Assessment

The entire session will take 90 minutes. A formal assessment (points relevant for the final mark) is not foreseen.

Task 1

Subtask 1.1 - Set up your working environment

In this lab, you will create a simplified cloud storage service using ZFS to offer block devices to clients over iSCSI.

Login to your cloud accounts on Ned and start an instance from the image `CCP1-EN-STRG-I`. This will be the "target" machine which runs the iSCSI target (server), so name it `target`. Ensure that your keys are installed on the VMs. Associate a floating IP to the VM and ssh into it with the centos user.

Create another VM using the same image: this will be your iSCSI initiator (client), so name it `initiator`. Attach a floating IP to this VM as well.

Assign a security group to both VMs that opens TCP port 3260. You don't need to open the port to the internet, Cloud-internal connectivity is sufficient.

Log into both VMs and create a directory in the users' home directory. Place all your work in this directory.

Notes:

- Do not upgrade the operating system
- When using the `zfs` and `zpool` commands below, you will need to use `sudo`
- To keep things simple and avoid reloading the iSCSI configuration, you should not reboot the VMs

Subtask 1.2 - Create reliable iSCSI resource on the target

All oft the operations in this task are to be performed on the "target" VM with the exception of one action in d-6 where it will be necessary to log in to the "initiator" VM to get the IQN of the iSCSI initiator.

a) Create a compressed (lz4) ZFS pool with 60% storage efficiency and a total of 300 MB of usable storage

- Create a new `devices` directory in your working directory
- Use the `fallocate` command to create file images to be used as storage devices for the pool - these images will be treated like disks.
- Determine the appropriate number of images together with their size to meet the requirement above. Create the image files in the `devices` directory
  - `fallocate -l <size>MB <filename>`
- Create a ZFS pool with name mypool matching the size and efficiency requirements with the zpool command by assembling the images create zpool create mypool …
- Set the `mypool` pool to use `lz4` compression. Verify the setting using
  - `sudo zfs get compression mypool`
- Use the zpool status and zfs list commands to verify your operations

b) Add redundancy to the pool by introducing one hot spare drive

- Create an additional image using fallocate
- Add the new image as a hot spare drive to the pool
- Check that the spare has been added to the mypool pool and that its status is available

c) Create a 30MB block device (ZFS volume) on the “mypool” pool with the name “myvolume”

- Use the zfs create command to create the volume dataset
- Use the zfs list command to verify
- Use the zfs get volsize mypool/myvolume command to verify the volume size

d) Expose the “myvolume” block device over iscsi

- Clear any existing configuration
  - `sudo targetcli clearconfig confirm=true`
  - `sudo targetcli /iscsi set discovery_auth enable=0`
- Create a backstore block device from the myvolume device
  - `sudo targetcli /backstores/block create name=blockmyvolume dev=/dev/zvol/mypool/myvolume`
  - `sudo targetcli /iscsi create`
- Read the name of the created iscsi target and assign it to an environment variable
  - `sudo targetcli ls /iscsi/`
  - `export TARGET_IQN=<IQN>`
- Create a LUN for the backstore block device
  - `sudo targetcli /iscsi/${TARGET_IQN}/tpg1/luns create /backstores/block/blockmyvolume`
- Disable authentication to simplify client operations
  - `sudo targetcli /iscsi/$TARGET_IQN/tpg1 set attribute authentication=0`
- On the iSCSI initiator, read the IQN name and, on the iSCSI target create an ACL rule to allow that initiator to access the exposed block device
  - [On the initiator] `sudo cat /etc/iscsi/initiatorname.iscsi`
  - [On the target] `sudo targetcli /iscsi/${TARGET_IQN}/tpg1/acls create <initiator_iqn>`
- Save the configuration on the target and restart the service
  - `sudo targetcli / saveconfig`
  - `sudo service target restart`
-

Subtask 1.3 - Mount iSCSI volume on initiator

Perform all the operations from **a)** to **f)** of this block on the “initiator“ machine. Perform operations **g)** on the “target” machine.

a) Clean any existing iSCSI records (there should be none, but this will be useful to cleanly restart the block, if needed)

```
sudo iscsiadm -m node --logout
sudo iscsiadm -m node -o delete
```

b) Discover any iSCSI target on the internal IP address of the target

```
sudo iscsiadm -m discovery -t st -p <target-internal-ip>
```

c) Access the iSCSI storage

```
sudo iscsiadm -m node --login
```

d) Verify that a new block device is available on the system

```
dmesg
sudo fdisk -l
```

e) Format the block devices and mount them on a destination folder you created

```
sudo mkfs.ext3 /dev/<device-id>
sudo mount /dev/<device-id> <dest-folder>
```

f) Write data to the filesystem and verify that it persists. Monitor block-level interactions.

Output some data with `sudo bash -c "echo 'hello world' > test.txt"` then unmount the mounted device, disconnect from iscsi and reconnect (`sudo iscsiadm -m node --logout` and `sudo iscsiadm -m node -l`), remount the block device (note that its name may change)

On the intitatior, write a small script that writes data into a file on your remote block-level device, then sleeps for 5s, and continues to do so N times. On the target, install tcpdump and trace the interactions. Redirect your trace into a file and download this file to your local machine. Install Wireshark and open the file. Verify the ISCSI block-level interactions.

g) Simulate device failures by zeroing out the disk images we made with fallocate

NOTE: do not kill more disks than your zpool redundancy can sustain

[On the target] `dd if=/dev/zero of=block1 bs=4M count=1`
[On the target] `sudo zpool scrub mypool`

Observe what happens to the zpool with the `status` command

Task 2

Automate the process of creating a volume and exposing it over iSCSI with a bash script

Note that most of the commands issued at Task 1 point **d)** are only needed once (e.g., a single iSCSI target exposes multiple LUNs), so only three more commands are missing in the script (see point e-2).

- Use the provided automate.sh file as a starting point. Mind resource naming, you may have to adapt names of pools/volumes/etc here and there.
- The script assumes that the `TARGET_IQN` environment variable is defined with the IQN of the iSCSI target and has the following interface
  - `./automate.sh <volume-name> <size>`
  - Example: `./automate.sh light 10MB`
- Add the 3 missing commands (see `TODOs`) and test that it correctly creates ZFS volumes and correctly adds backstores to the iscsi target. Verify the operations with
  - `sudo targetcli ls`
  - `sudo zfs list`
- `Remember to make the script executable with chmod u+x automate.sh`

Cleanup - Stop the Bills!

IMPORTANT: At the end of the lab session:

- **Delete** all OpenStack VMs, volumes, security group rules that were created by your team.
- **Release** all floating IPs back to the central pool for others to use.
    - Go to Network -> Floating IPs to release IPs back to the pool

Additional Documentation

No additional documentation is required for this lab.