Lab STRG-II - Object Storage

Introduction and Prerequisites

In this lab you will experiment with object storage systems, specifically Swift and its API.

In order to do so, you will experiment with a Swift instance (server) by issuing simple curl (client) commands on the command line. `curl` is a command line tool that allows you to issue low level HTTP requests and view the responses of those.

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

- Any modern web browser
- tcpdump installed on your client
- Wireshark on your local PC
- You will need an installation of `curl`
    - Mac: comes with it installed.
    - Linux: typically comes with it installed
- We will use a container that includes Openstack Swift and Keystone. Therefore, docker is a prerequisite on your system.

Time & Assessment

The entire session will take 90 minutes.

Task 0 – Running Swift in a container

We will use a container that includes Openstack Swift and Keystone (see more details here https://github.com/icclab/openstack-swift-keystone-docker).

First, start a plain Ubuntu-based OSTK instance and install docker by running the following commands:

```
sudo apt update
sudo apt install docker.io
```

Hint: To run docker commands without the need to always type sudo in front, you may want to add the user to the docker group with the following command *sudo usermod -aG docker* `$USER` Log out and log in again in the VMs for the last modification to take effect.

Verify that docker is running correctly on the VM - you can do this by simply listing active containers.

Now, on your instance, download and start the container using the following command:

```
docker run -d --rm  -p 5000:5000 -p 35357:35357 -p 8080:8080 --name
ostk_keystone-swift tmbohnert/openstack-keystone-swift:pike
```

Based on the command above, open all the required ports on your instance.

Swift, like any other OSTK project, requires Keystone. Integration of both services while allowing external access to the container can be achieved by using the docker port mapping feature and registration of Swift with Keystone against port 8080. Launch the following command to complete this setup:

```
docker exec -it ostk_keystone-swift /swift/bin/register-swift-endpoint.sh
http://127.0.0.1:8080
```

Note: The command above assumes that you will run client and server on the same machine. In case you opt for using two different instances (slightly more realistic scenario) you need to replace your local loopback IP with your Floating IP.

Task 1 – Experimenting with `curl` and Swift

In this task you will interact with the Swift API through `curl` (executed through the command line). Here you need to list containers (Swift containers), create a container, upload a plain text file as an object to the container and then delete the object and container.

1. Install tcpdump on your client and trace all client-server interactions. Analyze any of the interactions in the following with Wireshark. Especially identify the CRUD operations related to the Object API.
2. In order to access Swift you will need a **temporary authentication token**.
3. We will use the credentials of a `test` user for accessing Keystone via REST (`username=tester, password=testing, tenant_name=test`).
   Open a shell and run the following command to get a token to use later:

```
curl -v -H 'X-Storage-User: test:tester' -H 'X-Storage-Pass: testing'
http://127.0.0.1:8080/auth/v1.0
```

Note: If running the commands on your local machine (client), and the server on Ned, you need to use the Floating-IP to access it remotely instead of localhost (127.0.0.1).

4. Observe the received answer and find the fields `X-Storage-Url` and `X-Auth-Token` which are needed for the following `curl` requests. Please refer to the Swift API documentation (https://docs.openstack.org/api-ref/object-store/index.html) or the lecture slides on how to carry out these requests (note we are using a temporary authentication token https://docs.openstack.org/swift/latest/overview_auth.html , so the URL is a bit different w.r.t. the slides). You may store the values for `X-Storage-Url` and `X-Auth-Token` in env variables using: `export SWIFT_URL=<YOUR_URL> export SWIFT_TOKEN=<YOUR_TOKEN>` and use them in your curl queries (e.g. `-H "X-Auth-Token: $SWIFT_TOKEN"`)

5. Now interact with the server using CRUD operations.

   - As an example, we run `curl` request to list the available containers.
     ```
     curl -v -X GET -H "X-Auth-Token: $SWIFT_TOKEN" $SWIFT_URL
     ```
   - Next, create a container. Pick any name of your choosing.
   - Now that you've created a container you will want to put an object in it. Upload a plain text (content-type is text/plain) object to the container.
   - Now list the contents of the container that you have created
   - Now validate that what was uploaded can be retrieved (download it).
   - How would you implement an object modification?
   - Now delete the container and object that you've created. What happens if you delete the container first?

Additional Documentation

General

- Swift API: https://developer.openstack.org/api-ref/object-store/index.html