How to install TERENA Trusted Cloud Drive with Openstack Swift as storage backend

Guilherme Maluf Balzana

23 de maio de 2013

1 Introduction

TERENA Trusted Cloud Drive is a pilot experiment developing a personal data storage service that builds on a flexible Cloud Broker Platform. The unique features of the platform are:

- Federated access to the service
- Metadata and storage data are kept separate
- Storage data is encrypted and stored in the cloud
- Metadata stored in a trusted place.
- Various cloud storage back-ends can be brokered
- Flexible WebDAV front-end and web app.
- Different user platforms (Windows, MacOS, iOS, Android) are supported
- Code and documentation is fully open-source

TERENA Trusted CloudDrive has been built with the basic idea of separating the storage data (i.e. encrypted content) from the metadata (i.e. encryption keys, filenames, size, date, etc).By keeping the metadata store “on premises” data confidentiality is guaranteed under the assumption that the premises are inside a “trusted domain” – e.g. TERENA.

1.1 OpenStack Swift as Storage Backend

As part of GT-CNC working group (see Disclaimer at page 5) project a module was developed to connect TERENA Trusted CloudDrive project with OpenStack Swift as a storage backend. This module implements necessary commands to establish communication between TERENA TCD and Openstack Services as Keystone for authentication and Swift for storage. The module is implemented using the OpenStack Java SDK and Scala.

Here is shown how to set up a TERENA TCD using the OpenStack Swift as a storage backend and a workaround to allow OpenStack Java SDK to connect to a self-signed certificate URL.

These instructions above were done in a Ubuntu Precise 12.04 environment, but they can be easily replicated in other distros.

2 Requirements

2.1 Package List

- openjdk-6-jre or oracle-jre
- git
- mysql
- cadaver

Install all packages

```
# apt-get install -f openjdk-6-jre git mysql-server cadaver
```

Set up a root and password for mysql.

2.2 Clone TCD git repository

Now you need to get the TTCD code and go inside the directory.

```
# cd /opt
# git clone https://github.com/TERENA/CloudDrive.git
# cd CloudDrive
```

3 Setting up the TERENA Trusted Cloud Drive with the OpenStack Swift

All instructions are done exclusively in TERENA Trusted CloudDrive, there is no need to modify any aspect of OpenStack Swift.

3.1 Run Voldermort, the metadata store

```
$ cd binaries/metadata/voldemort-0.90.1-patched/bin/
$ ./voldemort-server.sh ../config/clouddrive_node_cluster/ &
$ cd -
```

3.2 Creating database and configuring rightfabric

rightfabric is the CloudDrive webdav module.

3.2.1 Configure database

```
$ mysql -u root -psenha -e 'create database rightfabric;'
```

3.2.2 Configure rightfabric

Copy configuration file to */etc/rightfabric/*

```
# mkdir /etc/rightfabric/
# cp config/clouddrive/config.txt /etc/rightfabric/
```

Change the **config.txt** file

```
# sed -i 's/10.0.0.5/localhost/' /etc/rightfabric/config.txt
```

3.2.3 Swift as storage backend

You need to fill Openstack Swift parameters in **/etc/rightfabric/config.txt**.

```
# vim /etc/rightfabric/config.txt
storage = swift

swift-username = admin
swift-password = adminpass
swift-tenant = admin
swift-auth-url = https://keystone_address:5000/v2.0
```

Change, save and exit *:wq*

3.2.4 User creation

There is a little script called addme.scala. Change the user and password, this will be your test user. (default: maarten:geheim)

```
$ cd CloudDrive/src/clouddrive/
$ vim addme.scala
```

Change, save and exit *:wq*

3.2.5 Running WebDAV module

Let's update and run the WebDAV module through Simple Built Tool (sbt)

```
$ cd CloudDrive/src/clouddrive/
$ ./sbt update
$ ./sbt console
scala> :load ./addme.scala
scala> :quit
$ ./sbt run
```

3.2.6 Testing with cadaver

```
$ cadaver localhost:9090
user: maarten
pass: geheim
```

3.2.7 Note for the impatient

If you type

```
$ ./sbt assembly
```

You will get a big JAR file in **target/scala-2.8.1/**. This is the standalone version that you can use for deployments. Just type **java -jar** ***whatever name you gave the jar*** and the WebDAV daemon will start.

3.3 Running CloudDrive website

After setting up the WEBDav module, you can start the website module.

```
# cd /opt/CloudDrive/src/web_clouddrive/
```

In the file **src/main/resources/props/default.props** replace *db.password* with your MySQL root password

```
$ vim src/main/resources/props/default.props
```

Update again and run the website

```
$ ./sbt update
$ ./sbt ~jetty-run
```

3.4 Finishing

Now open your browser in http://localhost:8080 and make your log in.

An advice is to run everything inside tmux or screen, windows for further detachment.

4 References

1. TERENA TCD code repository
   https://github.com/TERENA/CloudDrive/wiki/Two-Quick-starts

2. TERENA Installation Guide
   https://confluence.terena.org/display/CloudStorage/Installation+Guide

3. OpenStack Java SDK code repository
   https://github.com/woorea/openstack-java-sdk

4. Scala Simple Build Tool
   http://www.scala-sbt.org/

Appendix: Self-signed certificate with OpenStack Java SDK

When using Openstack endpoints with SSL, particularly with a self-signed certificate, you can get an error. This happens because JVM doesn't accept self-signed certificates. There are two solutions to solve this problem.

The first approach was to include the self-signed certificate in Java keystore as a trusted certificate. The second approach was to disable certificate verification in Java SSL code, which is not a good idea, since it makes software more vulnerable.

Indeed the first approach worked well and here is shown how it's done.

Firstly you need the self-signed certificates, you can get them through the Export option when managing certificates with a browser or directly with the certificate provider. Here it's call *keystone.pem* and *swift.pem* .

A specific problem that can happen comes from the fact that the certificate is using CN as an IP address instead of a hostname. When this happens the certificate should include the attribute **Subject Alternative Name** . You've to include the attribute *-ext san=ip:10.0.0.1* when using *keytool* to generate a certificate.

Now all you need is to find the keystore location, it will normally be in **$JAVA INSTALL DIR/jre/lib/security/cacerts**.

- Gentoo: **/opt/oracle-jdk-bin-1.7.0.21/jre/lib/security/cacerts**
- Ubuntu OpenJDK: **/usr/lib/jvm/java-6-openjdk-amd64/jre/lib/security/cacerts**
- Java Manual Install: **/usr/local/lib/jdk1.7.0_15/jre/lib/security/cacerts**

Then you should include the certificate inside the keystore using the keytool command:

```
# keytool -import -v -trustcacerts -storepass changeit \
          -keystore /opt/oracle-jdk-bin-1.7.0.21/jre/lib/security/cacerts \
          -alias keystone -file keystone.pem

# keytool -import -v -trustcacerts -storepass changeit \
           -keystore /opt/oracle-jdk-bin-1.7.0.21/jre/lib/security/cacerts \
           -alias swift -file swift.pem
```

Where, *-storepass changeit* is the password used by Java to manage the cacert, *changeit* value is the default password.

You can check if the certificate was included using:

```
$ keytool -list -v -alias keystone \
          -keystore /opt/oracle-jdk-bin-1.7.0.21/jre/lib/security/cacerts

$ keytool -list -v -alias swift \
          -keystore /opt/oracle-jdk-bin-1.7.0.21/jre/lib/security/cacerts
```

After theses steps the application stopped to generate the following errors:

```
javax.ws.rs.client.ClientException:
javax.net.ssl.SSLHandshakeException:
java.security.cert.CertificateException: No subject alternative names present

javax.ws.rs.ProcessingException:
javax.net.ssl.SSLHandshakeException:
sun.security.validator.ValidatorException: PKIX path building failed:
```

Disclaimer

This document was created as part of the activities of the GT-CNC (Grupo de Trabalho Computacao em Nuvem para Ciencia - http://gt-cnc.gercom.ufpa.br/) of RNP (Rede Nacional de Ensino e Pesquisa - www.rnp.br) Brazil.

Developer: Guilherme Maluf Balzana (guimalufb at gmail.com)