Cassandra Update and Restore for Axway API-Manager v7.7.rolling-updates

Prepared by: RKI, Axway DE

State: DRAFT

Initial version: 09.04.2020 / last update: 15.04.2020

Purpose

Following is a test for doing proper Cassandra keyspace updates and restores suitable for an installation of Axway API-Manager v7.6.4/7.7.rolling-updates.

Aim here is to have a procedure to ***clone*** an API-Manager installation along with its configuration database. Cloning of an environment can be particularly useful for doing save upgrades of an API-Gateway topology to conserve the current system to have a quick "way back" in case something might go wrong during upgrades. So, restoring the current state doesn't include a restore but instead requires just restart of the old system. This can minimize any downtime because of administrative tasks.

The way-back might be needed if the upgrade procedure included changes on the API-Manager database structure for some reason. Anther reason for cloning could be to create a similar installation at a geographically different location, e.g. two installations serving different parts of the world, or to create test systems that use real data for testing in case of special errors.

Considerations

Since API-Management v7.7. rolling updates release stream there is a change in the Axway online documentation provided at Axway Open Documentation. Mainly the documentation structure was improved for better readability and task driven instructions.

Now it includes a more handy description on how Apache Cassandra databases need to be operated for Axway API-Manager as this is different from Apache Cassandra used as Big-Table like massive datastore. In conjunction with Axway API-Manager Cassandra is used as an **application configuration datastore** that supports scale out of quite some application instances of API-Gateway as runtime for API-Manager. For an API-Manager cluster its Cassandra database - more precise the used keyspace - is also used as communication layer between all API-Manager instances to handle and replicate application state information like API usage details to apply quota limits. The API-Manager datastore holds:

1. static configuration data
2. application configuration data (seldom changed)
3. usage counters (high update frequency)

The cloning procedure here is considering 1) and 2) but is ignoring 3) for the upgrade process!

For the application Axway API-Manager there is hard dependency between applied server configuration and rules - the FED package containing server parameters and policies - and its data stored within the used Cassandra keyspace. Reason for this are the "encrypted fields" that API-Gateway is offering via its key-

property-store (KPS) abstraction layer. Those encrypted fields are used as secure storage of password hashes and similar data elements. The **encryption is not a Cassandra feature** but instead an application level function provided by the KPS layer of API-Gateway. The encryption key used for encryption within the Cassandra keyspace of an API-Manager instance is the API Gateway configuration encryption passphrase.

If an *API Gateway encryption passphrase* was used to secure the API-Gateway configurations, this passphrase, the FED package and a database backup are needed for restoring an API-Gateway installation (e.g. disaster recovery from backups).

```
 API-MANAGER -> API-Gateway runtime -> KPS layer API-Gateway -> KPS for Apache
Cassandra -> Cassandra server
|------ FED includes policies and KPS table mappings -------|
|-- API-M data --|
|----------------------------------------- sensitive data encryption key -------
-------------------------|
```

Details on *Group Passwords* and their relation to encrypted KPS fields can be found here within documentation API-Gateway v7.6.2 KPS.

What is KPS?

The KPS layer of API-Gateway is a (legacy?) feature that is used to abstract from several data storages with different capabilities. For some of the currently supported storages it is necessary to provide detailed information on the data structure. Otherwise, on application level it would be unknown how to access the needed data on a data store.

API-Gateway is using the KPS structure information deployed to it via FED packages to manage the underling storage system (e.g. creating, updating database tables). KPS storage is separated into *KPS collections* and for Cassandra within *tables* in addition. Each collection is refering to its underlying storage system. The API-Manager component of API-Gateway has a hard dependency on Cassandra as KPS staorage.For Apache Cassandra a keyspace must be configured either manually or by running Axway provided installer scripts.

For more datails on KPS please consult KPS FAQ.

The Axway scripts are part of an API-Gateway + API-Manager installation. The files are located at:

```
<install-dir>/apigateway/posix/bin
```

Cloning an API-Manager

Open Docs for API-Manager includes a chapter on how to administrate the Apache Cassandra database which is delivered along with an Axway API-Manager installation. Documentation reference: Administer Apache Cassandra. Especially the chapter Apache Cassandra backup and restore is important to ensure correct backups for disaster recovery.

For cloning an API-Manager installation we don't want to rely on Cassandra backup and restore mechanics as this is binding us to the Cassandra topology used for the source environment. In case of Cassandra multi-

node clusterst it also seems to be more complex than the API-Gateway provided KPS backup and restore tooling that is part of each API-Gateway installation.

Axway API-Manager doesn't use its Cassandra keyspace not in a way like big-data applications are using a Cassandra database. Especially the amount of data managed by Cassandra for API-Manager is very small. Usually its size is up to 1GB, perhaps slightly more. Cassandra is used here for its replication and consistency mechanisms to support scaling out an API-Manager cluster accross more servers for higher load. For details see Capacity planning and performance.

Because the amount of data is not large we can do an backup/restore in a classic approach by simply exporting the whole dataset at once at one of the API-Gateway nodes of the same API-Gateway Group that hosts API-Manager. Axway is providing a tool for this: **kpsadmin** - Manage KPS using the kpsadmin tool

API-Gateway Configuration Backup

As stated above, it is important to know the *API Gateway encryption passphrase* if it was used!

First step should be saving the currently running FED package from the API-Gateway group that hosts API-Manager. Following Admin-Node-Manager (ANM) API call can be used to list the current API-Gateway topology.

The administrative API of Admin-Node-Manager is documented as API Gateway API v1.0

```
# retrieving topology description
# the sample only has "instance-1" gateway running within "group-2"
#
curl -k -u "admin:changeme"
https://localhost:8090/api/deployment/domain/deployments | python -m json.tool

{
    "result": {
        "group-2": {
            "instance-1": {
                "deploymentTimestamp": 1586366948256,
                "environmentProperties": {
                    "Description": "Original factory configuration (Environment)",
                    "Manifest-Version": "1.0",
                    "Name": "Default factory configuration for API Gateway
(Environment)",
                    "Version": "v1 (Environment)",
                    "VersionComment": "Original factory configuration
(Environment)"
                },
                "policyProperties": {
                    "ChangedBy": "admin",
                    "Description": "The QuickStart Server configuration policies",
                    "Manifest-Version": "1.0",
                    "Name": "QuickStart Server Configuration",
                    "Version": "v1",
                    "VersionComment": "Imported API Manager configuration"
                },
```

```
                "rootProperties": {
                    "Id": "842d18d5-7498-4286-b136-ae512923253b",
                    "Timestamp": "1586366871502"
                },
                "status": "Loaded",
                "user": "admin"
            }
        }
}
```

Next, one can retrieve the currently running configuration (FED archive) from one of the API-Gateway instances of the API-Manager gateway group, either by API call or via Policy Studio in interactive mode.

```
# sample call to query currently running FED package from "instance-1"
#
curl -k -u "admin:changeme"
https://localhost:8090/api/deployment/archive/environment/service/instance-1 |
python -m json.tool > instance-1-config.txt

less instance-1-config.txt
{
    "result": {
        "data": "<base64-fed-archive>",
        "rootProperties": {
            "Description": "Original factory configuration (Environment)",
            "Id": "cfde737f-0362-4d55-9de3-71f6dc6f8fee",
            "Manifest-Version": "1.0",
            "Name": "Default factory configuration for API Gateway (Environment)",
            "Timestamp": "1586884449546",
            "Version": "v1 (Environment)",
            "VersionComment": "Original factory configuration (Environment)"
        }
    }
}

# requesting and saving currently running FED package
# 1) get deployment archive ID
curl -k -u "admin:changeme" https://localhost:8090/api/router/service/instance-
1/api/configuration/archiveId

{
    "result": "842d18d5-7498-4286-b136-ae512923253b"
}

# 2) receive deployment archive
curl -k -u "admin:changeme" https://localhost:8090/api/deployment/archive/group-
2/842d18d5-7498-4286-b136-ae512923253b | python -c 'import sys, json; print
json.load(sys.stdin)["result"]["data"]' | base64 -d > instance-1-config.fed
```

Exporting KPS Collections and Tables from Running FED

In order to clone we need the KPS definitions of the configuration group that is currenly running. These definitions are needed for restoring data into a new Cassandra keyspace in a later step.

1. open Policy Studio
2. *New Project*, name (e.g. Original-FED)
3. chose Project starting point "*From .fed file*"
4. navigate to *Environment Configuration -> Key Property Stores*
5. export KPS configuration: right click and chose *Export all Key Property Stores*; save with name, e.g. KPS-definitions.xml
6. close current policy package project (File -> Close Project)

Hint: *The export now containes only KPS definitions without any policies or server setings. Those details describe the table structure within Cassandra and for the KPS access layer. The data definitions are needed for the export/import to access the Cassandra data store (keyspace). During deployment of an API-Gateway FED package any changes on KPS settings are replicated onto the configured data store (chosen keyspace). Hence, the API-Gateway will create or update the data definitions on the Cassandra database cluster accordingly!*

Creating new target Cassandra keyspace / KPS

For cloning we now need a new target keyspace at the target Cassandra database cluster. As this will be a new keyspace on an existing Cassandra cluster it should be created similar to the original one.

Creation can be done via Cassandra management tool (cqlsh) or via an API-Gateway FED deployment. For a FED deployment the first API-Gateway that received configuration for a Cassandra keyspace will use its confiured user to create the needed keyspace if it is not already available. If the target keyspace already exists the API-Gateways of the deployment group will not change the existing keyspace configuration!

In order to have full control on this operation Cassandra tooling seems to be the better option. Please make sure the configuration matches the setup of your Cassandra cluster configuration. For more details please consult Configure a highly available Cassandra cluster.

Hint: For the special runtime platform *Amazone AWS* third party developers have compiled more detailed descriptions on how to adopt Apache Cassandra on AWS resources for optimum performance and recilience.

- Cassandra on AWS EC2: Best Practices for Running Apache Cassandra on Amazon EC2
- Cassandra K8s managed on AWS: Deploy a highly-available Cassandra cluster in AWS using Kubernetes

Axway Services DE is currently evaluation both options for recomandation and applicaltion samples customers can adopt. In its current state the referenced descriptions are not yet "production ready" because of security and static initial configurations. Nevertheless, the approach is right for modern system operations, auto scaling and auto healing.

Sample for replication option *SimpleStrategy*:

Assumption: Cassandra cluster of 3 nodes. The underlying infrastructure is responsible for resilience. (e.g. AWS region with 3 availablilty zones - one node per AZ)

```
CREATE KEYSPACE apimanagerv77 WITH replication = {'class': 'SimpleStrategy',
'replication_factor': '3'}  AND durable_writes = true;
```

Hint: *According to external informations on the web for Cassandra keyspace configuration - for topologies typically used on AWS for API-Manager the replication option 'Simple Strategy' could be sufficient within same AWS region with 3 availablity zones.*

Sample for replication option *NetworkTopologyStrategy*:

Assumption: Cassandra cluster of 3 nodes. One node per data center; at least two DC's are always available for resilience. The Cassandra nodes are spread accros DC's and configured appropriately.

```
CREATE KEYSPACE apimgr_v77 WITH replication = {'class': 'NetworkTopologyStrategy',
'DC1': '1', 'DC2': '1', 'DC3': '1'}  AND durable_writes = true
```

Configuration details shold be read as: each data row must be stored 3 times; one data copy must be place on a node belonging to DC1, one copy within DC2 and another one in DC3. The data center names must be configured at the Cassandra node configuration as "DC1" to "DC3" within Cassanadra config file */cassandra/conf/cassandra-rackdc.properties*.

Attention: *In case your API-Gateway setup uses policies with the filter "Throttling" and the rate limiting Algorithm is configure to "Smooth Rate Limiting" an appropriate Cassandra keyspace must be properly configured! (Policy Studio: Server Settings - Cassandra - Throttling; this keyspace config is only for Throttling; it creates an addional table "throttling_table" in that keyspace; a dedicated keyspace allows to define different Cassandra distribution settings, tuned for throttling data)*

*See Configure rate limiting within the docs for more details. **API-Manager** itself is **NOT** relying on this feature. It uses its own quota tracking for rate limitting.*

Export all data from currently used Cassandra keyspace

Documentation for *kpsadmin* tool is available at Manage KPS using kpsadmin.

Create full backup in interactive mode

```
<installdir>/apigateway/posix/bin/kpsadmin --username admin --password <secret>
...
    Group Administration:
        25) Clear All
        26) Backup All
        27) Restore All
        28) Re-encrypt All
        29) Group Details
...
Select a Group:
1) CassTestGrp
2) QuickStart Group
Enter selection [1]: 2

Select an API Gateway:
1) QuickStart Server
Enter selection [1]: 1
```

```
Select option: 26

This operation makes a backup of the following tables to backup file on the KPS
API Gateway.
The backup files are placed in: apigateway/groups/<group-id>/<instance-
id>/conf/kps/backup
A unique backup UUID is prepended to each file.
You will need this UUID and the backup files to do a restore.
        API Portal_ApiAppPolicyBindings
        ...
        QuickStart_HeroesCharactersRegistry
Are you sure you wish to continue y/n [n]: y

Backup uuid is: f2c2d44b-d745-45e1-a516-315e0e7781c7
Started backup
Starting backup of table: API Portal_ApiAppPolicyBindings to:
f2c2d44b_d745_45e1_a516_315e0e7781c7_api_portal_apiapppolicybindings_json ...
Backup done in 0:00:01 seconds - 200
...
```

Backup files are stored on the server that is hosting the API-Gateway instance that is used for backup triggered by kpsadmin. File location is */apigateway/groups/group-/instance-/conf/kps/backup/*. All files with the same UUID belong to the backup set. Sizing can be checked using:

```
$ du -h <install-dir>/apigateway/groups/group-2/instance-1/conf/kps/backup/
19M /opt/Axway/APIM/apigateway/groups/group-2/instance-1/conf/kps/backup/

$ du -h <install-dir>/apigateway/groups/group-2/instance-
1/conf/kps/backup/f2c2d44b_d745_45e1_a516_315e0e7781c7_*
```

Prepare the new Cassandra keyspace for KPS import

(establish KPS definitions within new API-Gateway for the import procedure)

...tbd...

Creating new target Cassandra Keyspace / KPS

(copy export data files + we use kpsadmin) ...tbd...

Switch from original Environment to Cloned environment

...tbd...

Upgrade using cloned API-Manager environment

(follow the upgrade procedure layed out in docs) ...tbd... (switch back to source environemnt in case of any error)

Conclusion

For cloning and correct backup for a disaster recovery it is important to understand the KPS layer of API-Gateway and the application level feature of API-Manager with its tight coupling to Apache Cassandra. API-Manager Backups must consist of:

1. API-Gateway configuration and policy deployment packaged (FED)
2. API-Gateway group passphrase
3. Cassandra keyspace backup

Following the procedure outlined above we are able to completely clone an environment. This allows for a faster, more simple switch between major versions. In case of breaking changes are applied at the database structure on Cassandra keyspace it still allowes to quickly switch back to the original setup in case anything should go wrong during such version upgrade. This switch just requires shutdown of the faulty installation and restart of the original environment.

Other tools for Cassandra Backup / Restore handling

Medusa is an Apache Cassandra backup system developed by TLP (now DataStax). Available for free from: https://github.com/thelastpickle/cassandra-medusa Nice feature is the integration with cloud storage from AWS or Google as backup location.

...tbd...

RKI: Worth further testing!

Apache Cassandra at IaaS cloud providers?

AWS: Best Practices for Running Apache Cassandra on Amazon EC2 https://aws.amazon.com/blogs/big-data/best-practices-for-running-apache-cassandra-on-amazon-ec2/

This approach allows an very easy approach for upgrading Apache Cassandra software without service interruption. But requires setup of Cassandra for allowing it (second network interface). Applying rolling updates becomes more easy this way.

RKI: I'm not yet happy with the Apache Cassandra within a container articles I have seen so far.

- A good stating point here us a TLP blog post: Docker Meet Cassandra. Cassandra Meet Docker. -> https://thelastpickle.com/blog/2018/01/23/docker-meet-cassandra.html
- Presentation on SlideShare: Running Apache Cassandra on Docker -> https://www.slideshare.net/JimHatcher/running-apache-cassandra-on-docker