OVIDIU LOGHIN

Key Value project

Prepared for: Key Value assignment, 1&1

Prepared by:Ovidiu Loghin

20 January 2019

OVIDIU LOGHIN

EXECUTIVE SUMMARY

Objective

Implement a key-value store service:
Service exposes a RESTful API that supports the following actions:
get: find key in the storage and return JSON-encoded key-value pair, ex: {"key": "my-key", "value": "my-val"}
put: write given (JSON-encoded) key-value pair to storage
delete: find key in storage and delete its key-value pair
size: return number of key-value pairs in the storage
* keys are strings of 1-64 characters restricted to character set: a-zA-Z0-9_-
* values are strings with maximum length of 1KB (1024 bytes), all characters are allowed (binary data)

Goals

Horizontal scalability: multiple instances of the service running on different machines, operating on the same storage. Implement the storage on disk / filesystem, any solution is acceptable as long as it supports big data (millions or billions of pairs)
Key-value store must be generally consistent, but small inconsistencies are allowed for edge cases: key written on one node might not be instantly available on another node

Solution

In order to make pollution scalable and keep consistence I have implemented services that are centrally controlled. Every service that may have operation on key-value storage could be instantiated distinctive, but will announce his instantiation to a controller. Controller will insure consistency operation throughout every instance on every machine.

The solution contains two distinct parts:

- Federation Controller : needs to be started first and configured in application.properties of each service
- KeyValue Service: could be on distinctive machine or on one single machine allocating different ports

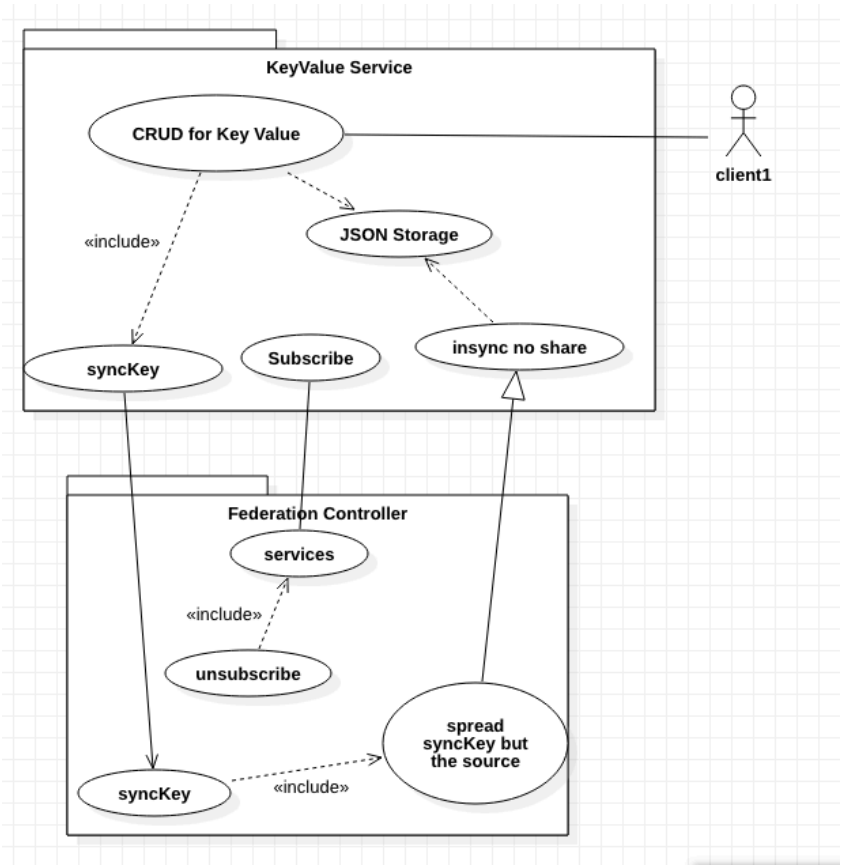
Project Outline

Both controller and federated services are written in Java, using Spring booth framework. The value are store in every local service file. Every value is spread throughout the pollution of services federated to the controller. Every operation on a separate instance of the service is spread to any other instance. In case one service is down the Federation controller will unsubscribe the service automatically. In case service is restated it will look to subscribe to the federation controller.
All operations are web api and Ould be accessed on http.

SOLUTION in Details

The solution contains two distinct parts:

- Federation Controller : needs to be started first and configured in application.properties of each service
- KeyValue Service: could be on distinctive machine or on one single machine allocating different ports



Every time Key Value Service is instantiated will look for the controller (configured in application.properties), if subscribed will send the current values. In the same time will receive values from the controller. Values that are stored in any active Key Value Services known by the controller

Any CRUD operation on Key Value Service, besides of being performed in the JSON storage, will be send trough the /synckey to the controller to be spread out to any other service but the source.
In case a service is not responding to the controller, it will be subscribed automatically
Services could be interrogate to /service api

A Key Value Service tries every time to sync with the controller. First pings the controller, then calls the service. Flowing paths are useful to be included in the test scenarios.

KEY VALUE SERVICE API

GET

http://{{ip}}:{{port}}

/keys	gets all keys in the storage
/keys/{key}	gets the key:value pair of the {{key}}
/keys/find?key={key}	gets the key:value pair of the {{key}}
/init?size=nnn	initialise the current storage with nnn value. Does not send values to controller
/keys/size	gets the number of key-value pairs in the database
/environment	the current service ip and port
/help	basic api path of the service, no explanation

POST

/keys updates a {"key": "key5559", "value": “value5559"} pair if exits, or stores it as new

/insync Called by the controller to update or create keys from other subscribed services

Example of json body

```
{
        “source”:{ip:"127.0.0.1", “port”: “8080”},
        “keys” :{“key1”:"value1","key2":"value2"}
}
```

PUT

/keys creates {"key": "key5559", "value": “value5559"} if it does not exist

/insync Called by the controller to create keys from other subscribed services

Example of json body

```
{
        “source”:{ip:"127.0.0.1", “port”: “8080”},
        “keys” :{“key1”:"value1","key2":"value2"}
}
```

DELETE

/keys/{{key}} deletes the key

/indelsync called by the controller to delete keys deleted in other service storage

FEDERATION CONTROLLER

http://{{ip}}:{{port}}

GET

/services get all subscribed services

/services/count number of subscribed services

PUT

/synckeys called by the service that received a /key PUT action in order share the value with all other i l instances

Example of jsonbody {“source”:{“ip”:"127.0.0.1", “port”: “8080”},

“keys:”{"key1":"value1","key2":"value2"}}

DELETE

/syncdelkeys :: delete the keys from json_body

//Ex of jsonbody {source:{ip:"127.0.0.1", port: 8080},keys:{"key1":"value1","key2":"value2"}}

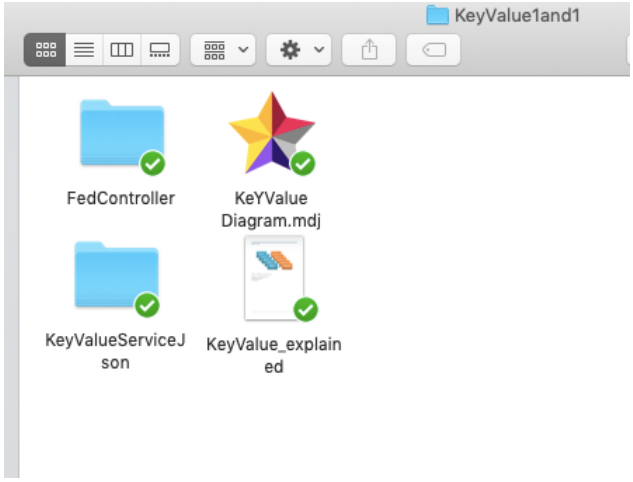
POST

/synckeys :: updates the values in all other instances

//Ex of jsonbody {source:{ip:"127.0.0.1", port: 8080},keys:{"key1":"value1","key2":"value2"}}

BUILD AND RUN THE SOLUTION

Within the archive you will sed the following content



1. First build and run the Federation controller

In aterminal

cd FedController/

mvn clean package

java -jar ./target/FedController-0.0.1-SNAPSHOT.jar

2. Modify the application properties of KeyValueServiceJson

In a new terminal :

cd ../KeyValueServiceJson/

nano src/main/resources/

application.properties

If you run the federation controller on localhost with KeyValue Service leave the ip on local host

Default KeyValue Service port is 8099 in this example

Consistency.policy specifies what Key Value Service accepts when instantiating
EXISTING means that any new instance will erase its value and get values from any existing service

CTLR X saves the changes

Build the solution with : *mvn clean package*

In case of build success, run this instance with: *java -jar ./target/KeyValueServiceJson-0.0.1-SNAPSHOT.jar*

3. For multiple instance build and run

Go to step 2, change server.port build and run instances

BUILDING WITH DOCKER

Within the archive you will sed the following content

1. First build and run the Federation controller

In a terminal, starting the archive root : *cd FedController/*

My docker images looks like this: *docker images*

```
FedController — -bash — 128×24
192-168-0-108:FedController ovidiu.loghin$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
java                8                   d23bdf5b1b1b        2 years ago         643MB
192-168-0-108:FedController ovidiu.loghin$
```

Run mvn clean install docker:build

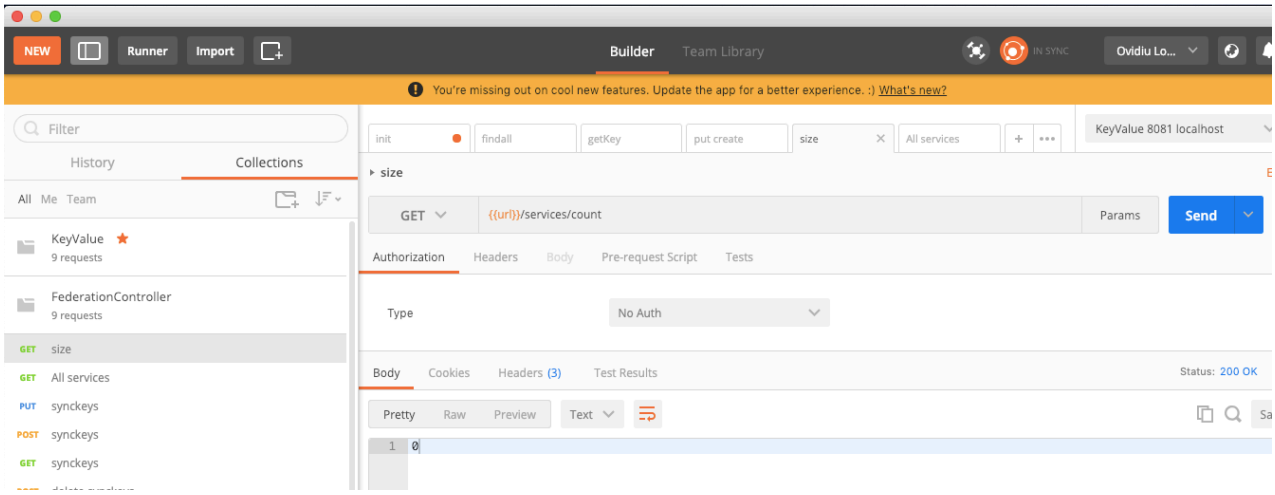
Docker Images needs to look like this

```
FedController — -bash — 85×24
Step 2/3 : ADD /FedController-0.0.1-SNAPSHOT.jar //

 ---> 8cf61dda5e5d
Step 3/3 : ENTRYPOINT ["java", "-jar", "/FedController-0.0.1-SNAPSHOT.jar"]

 ---> Running in 1eaffcc44ae5
Removing intermediate container 1eaffcc44ae5
 ---> 6807efd1276d
ProgressMessage{id=null, status=null, stream=null, error=null, progress=null, progres
sDetail=null}
Successfully built 6807efd1276d
Successfully tagged fedctl:latest
[INFO] Built fedctl
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  15.089 s
[INFO] Finished at: 2019-01-21T00:14:14+02:00
[INFO] ------------------------------------------------------------------------
192-168-0-108:FedController ovidiu.loghin$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
fedctl              latest              6807efd1276d        6 seconds ago       660MB
java                8                   d23bdf5b1b1b        2 years ago         643MB
192-168-0-108:FedController ovidiu.loghin$
```

RUN CONTROLLER

Run the containerised with: *docker run -p 8081:8080 fedctl*

For now postman http://localhost:8081/services/count will return 0 as no key value service has subscribed



2. Find the docker ip of Federation to make any key value aware

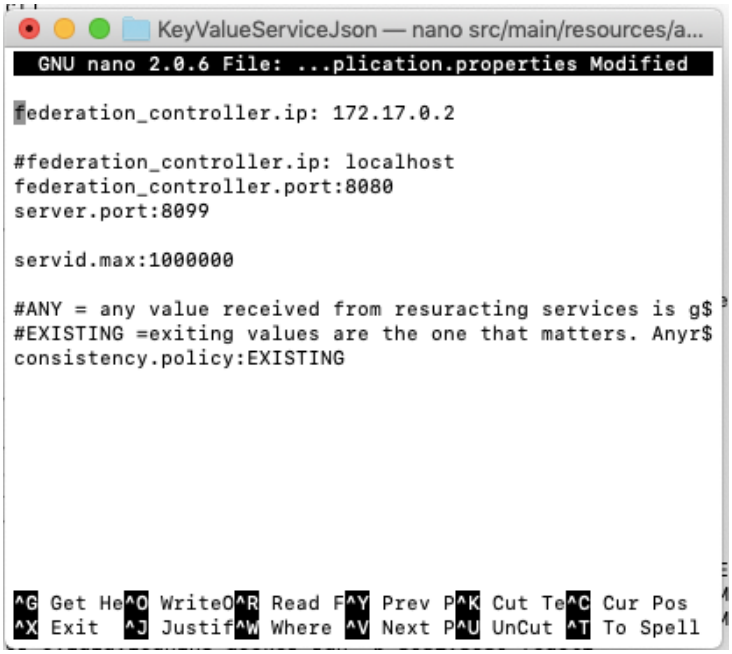
docker container ls will point out the my container id is 4269fe26731c

docker container inspect 4269fe26731 will show that my Federation controller is instantiated on "IPAddress": "172.17.0.2", port 8080

Go to KeyValueServiceJson with *cd KeyValueServiceJson*

Modify properties with *nano src/main/resources/application.properties*

Make sure the Federation controller ip an port are the one in docker



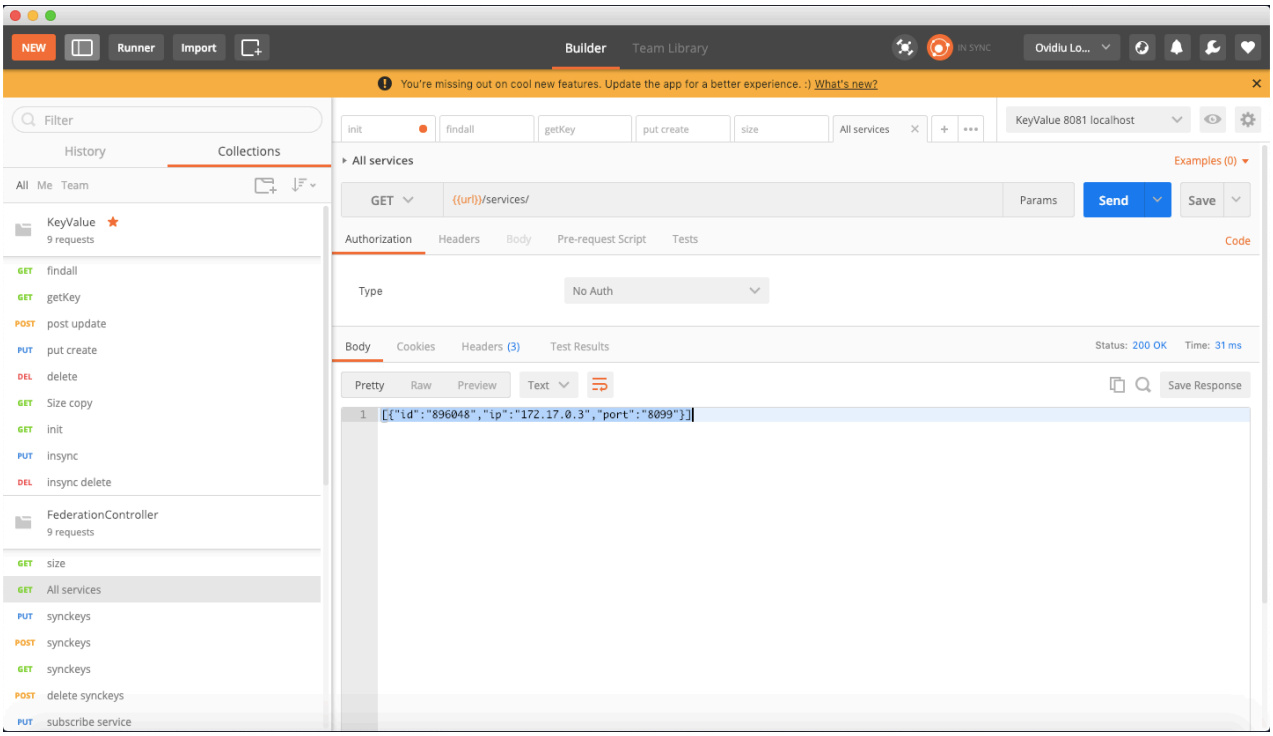
3. Build and run multiple Key Value Service instances in docker

mvn clean install docker:build
Docker images will look like this

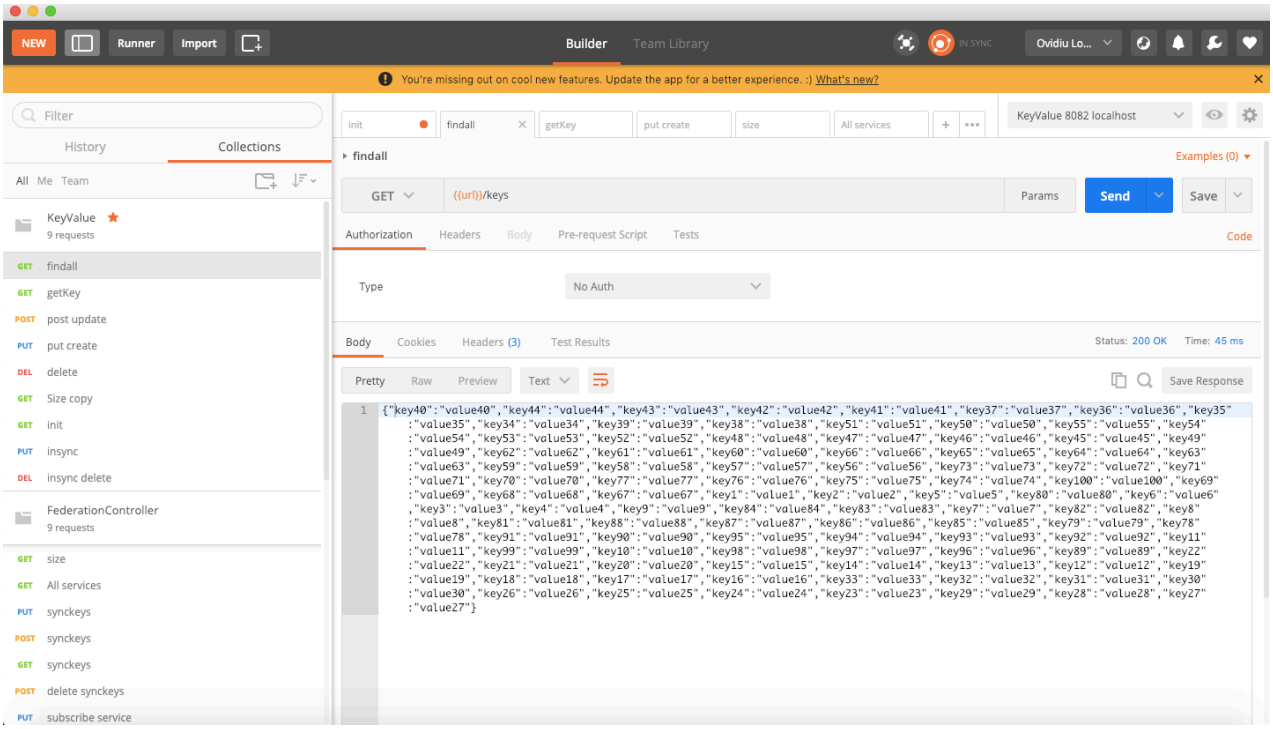
```
KeyValueServiceJson — -bash — 115×24
Step 2/3 : ADD /KeyValueServiceJson-0.0.1-SNAPSHOT.jar //

 ---> a4070016c73b
Step 3/3 : ENTRYPOINT ["java", "-jar", "/KeyValueServiceJson-0.0.1-SNAPSHOT.jar"]

 ---> Running in b3c5f750e7f0
Removing intermediate container b3c5f750e7f0
 ---> ee7f2aaacee1
ProgressMessage{id=null, status=null, stream=null, error=null, progress=null, progressDetail=null}
Successfully built ee7f2aaacee1
Successfully tagged kvjsonsrv:latest
[INFO] Built kvjsonsrv
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  16.011 s
[INFO] Finished at: 2019-01-21T00:48:30+02:00
[INFO] ------------------------------------------------------------------------
192-168-0-108:KeyValueServiceJson ovidiu.loghin$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
kvjsonsrv           latest              ee7f2aaacee1        13 seconds ago      661MB
fedctl              latest              6807efd1276d        34 minutes ago      660MB
java                8                   d23bdf5b1b1b        2 years ago         643MB
192-168-0-108:KeyValueServiceJson ovidiu.loghin$
```

Running images will be done with
Docker run docker run -p 8082:8099 kvjsonsrv

You cannot test the first instance of key value service running on local machine with public port 8082
The controller http://localhost :8081/services will show the subscribed services



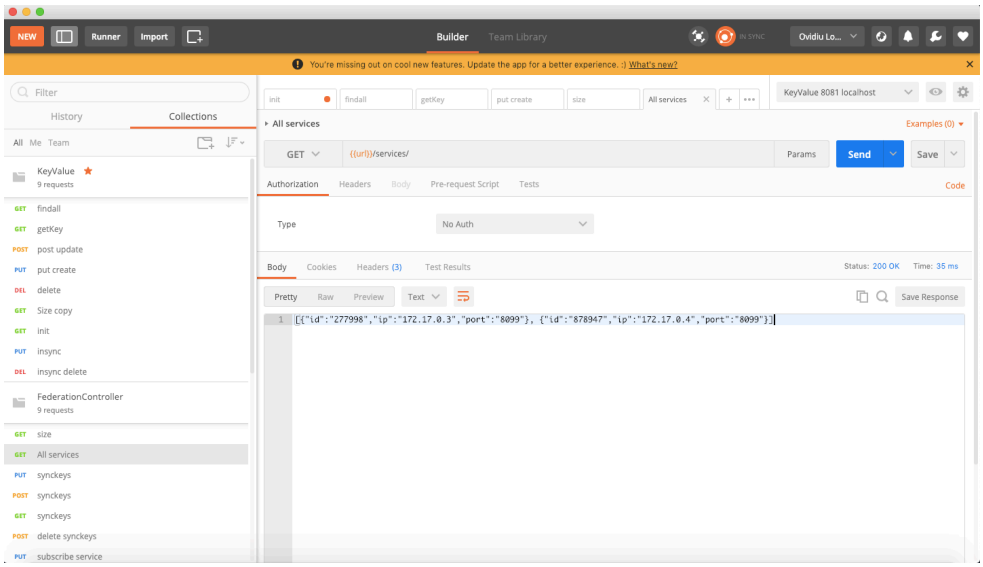
The Values on the new instances are http://localhost:8082/keys



4. Running a new instance and make CRUD operations in both instances

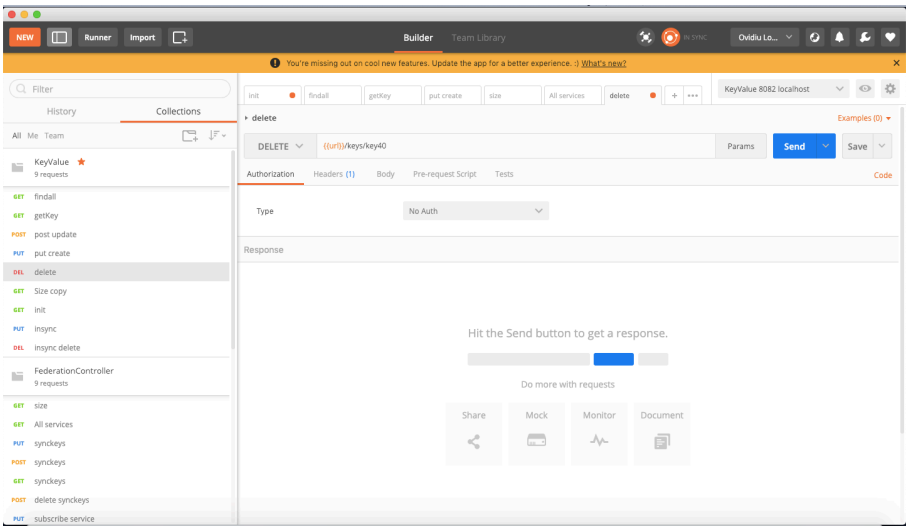
Docker run docker run -p 8083:8099 kvjsonsrv

The second instance instantiated correctly and subscribed to controller

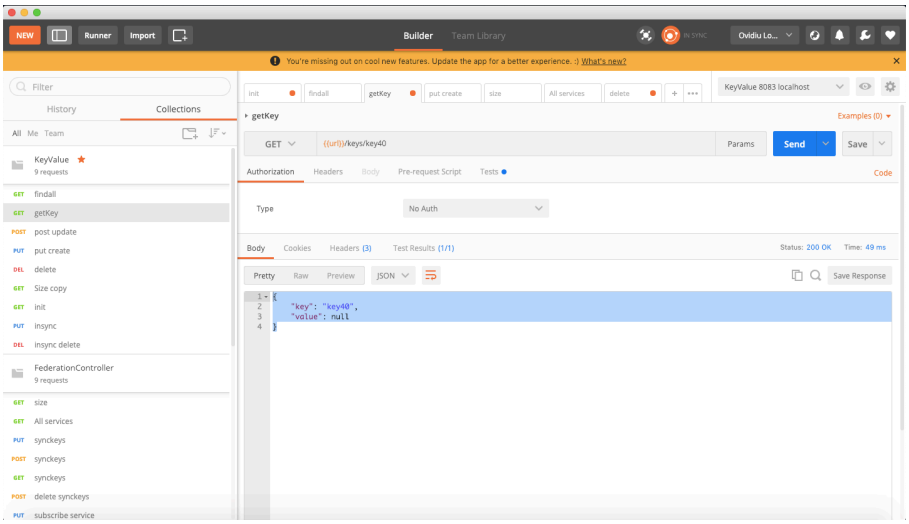


5. Running a delete and put command on one instance will spread keys to all node

Deleting key40 from instance running on 8082 will delete value from also from 8083



{"Key":"key40", "value":"value40"} Permanently deleted!



FINAL CONSIDERATION

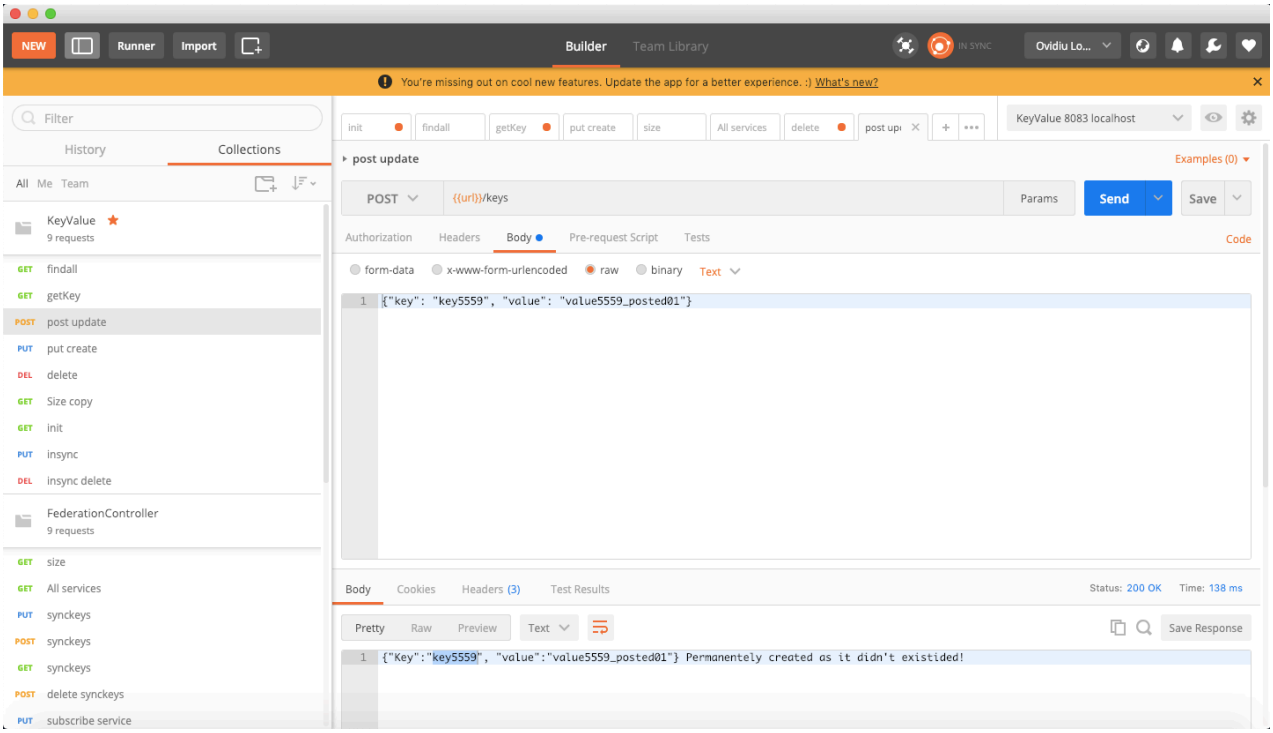
All CRUD operations behaves the same.

Key-Values are stored in any instance the same.

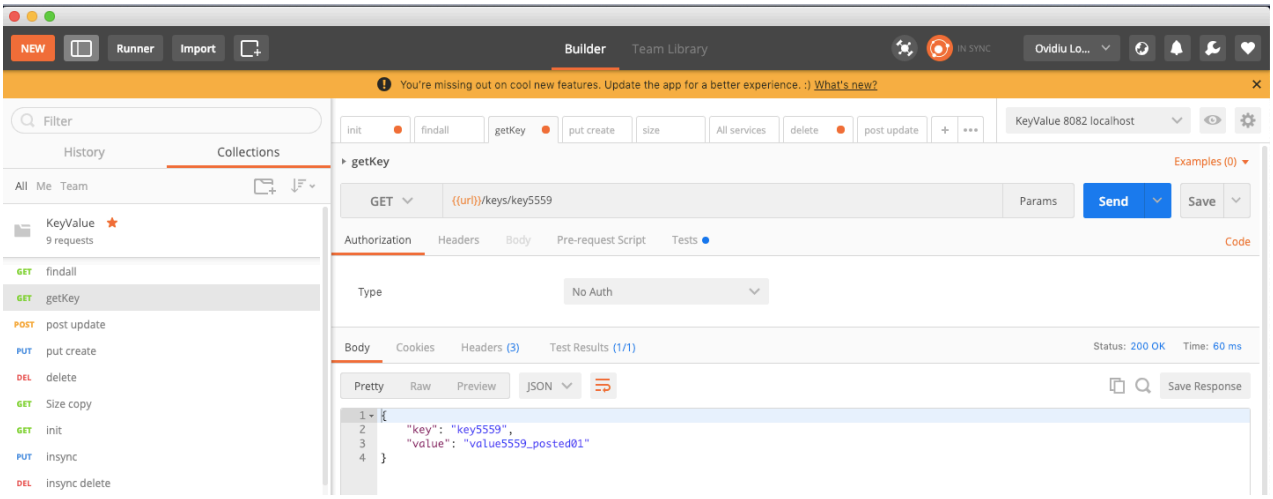
In case the controller fails, the services act independently and not synchronised. In this case it is best to set all all instance on consistency.policy=ANY and restart in order to make nothing is lost.

More test might be done by sharing the postman project and environment.

POST key5555 in 8083



Will result in the same value on 8082



Please be aware that a Federation Controller does not store information, storage resides on any key values service instance.