Zato-DHIS2 Integration on Ubuntu 16.04

# 

# Install Zato and Redis server

$ sudo apt-get install apt-transport-https curl redis-server

$ curl -s https://zato.io/repo/zato-3.1-C9B13DF28CFE287D.pgp.txt | sudo apt-key add -

$ sudo add-apt-repository \

"deb [arch=amd64] https://zato.io/repo/stable/3.1/py27/ubuntu$(lsb\_release -cs) main"

$ sudo apt-get update

$ sudo su - zato

$ cd /opt/zato/current && git checkout -- ./requirements.txt

$ ./update.sh

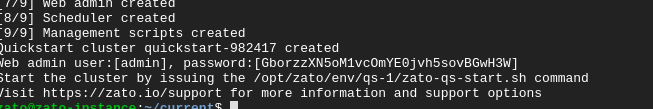
Zato should now be installed.

# Create and start a Zato cluster

$ mkdir -p ~/env/qs-1

$ export path=~/env/qs-1

$ zato quickstart create ~/env/qs-1 sqlite localhost 6379 --kvdb\_password '' --verbose

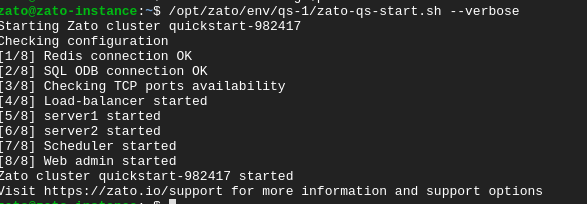


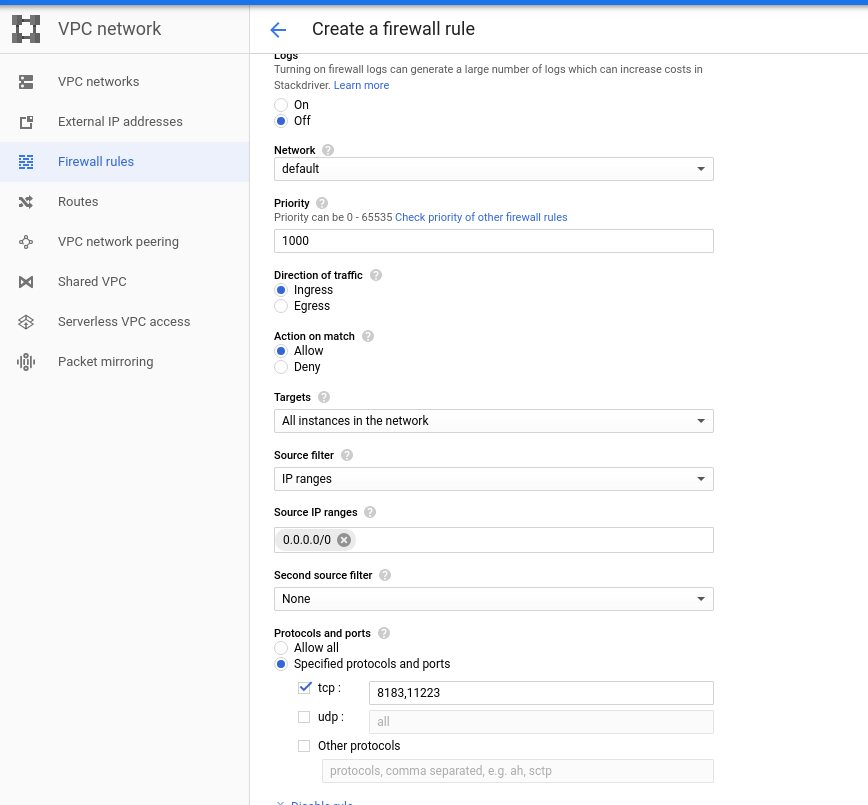
This will generate a random password like as shown in figure above. Keep this secure as this is necessary to login to admin interface.

Now start Zato cluster using:

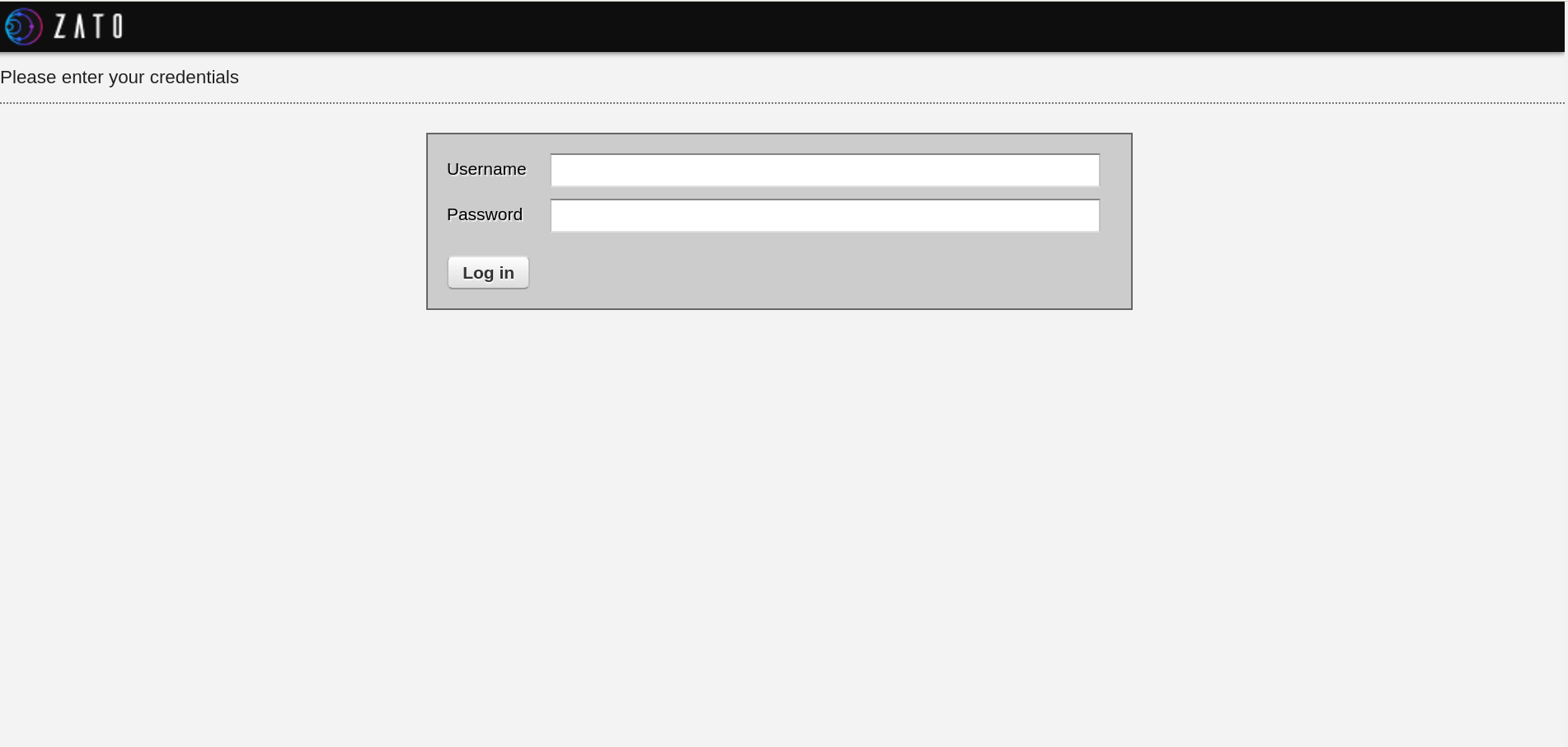
$ /opt/zato/env/qs-1/zato-qs-start.sh --verbose

This should bring up Zato cluster like as shown in figure below:



Zato uses port 8183 for admin interface, and 11223 for HTTP endpoints. Depending on your configuration, you should allow these TCP ports. For instance on Google Cloud Console, you can do it via the VPC firewall rules:

Once done, you should be able to access Zato at http://<server\_ip>:8183/ which will bring up a login interface:

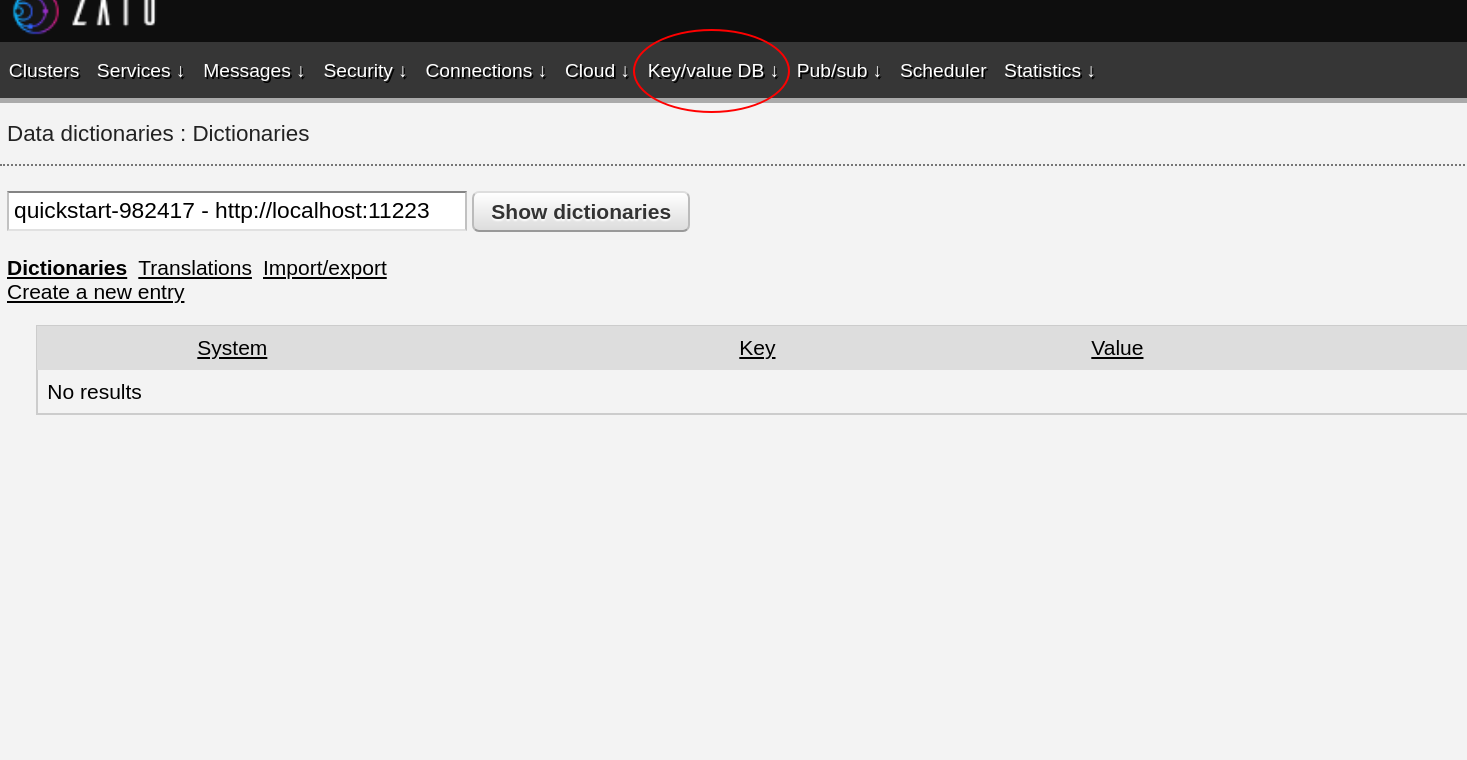


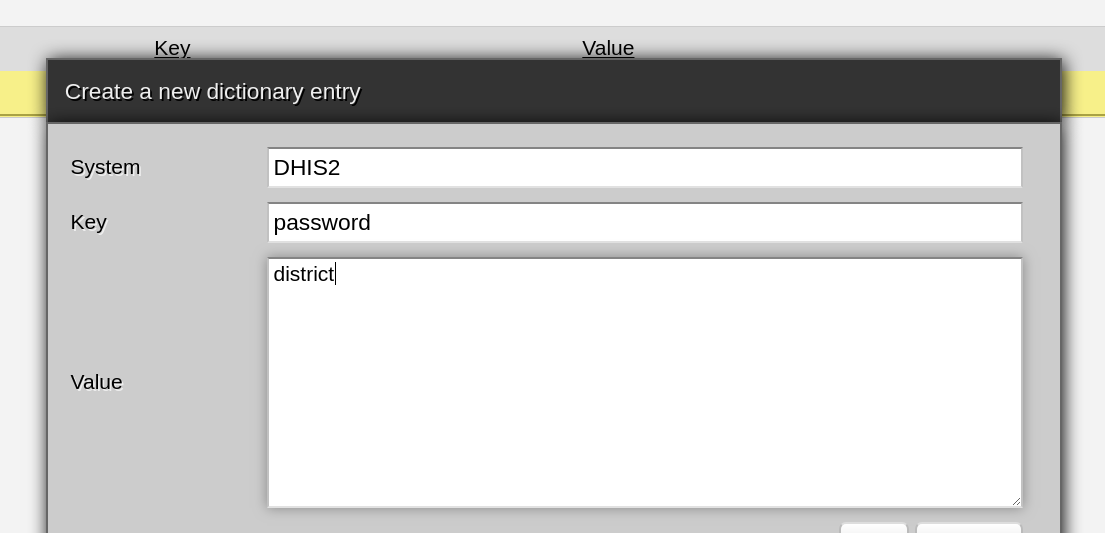
Use earlier-generated admin-password combination to login.

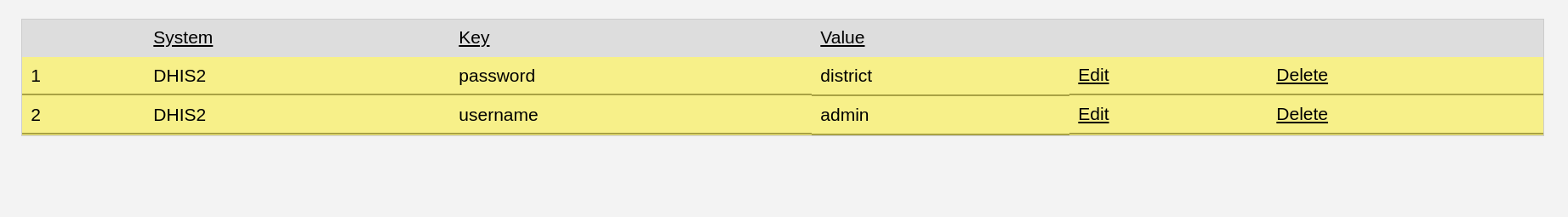
# Key Features Useful for DHIS2 Integration

## Key Value Store

Key Value Store can be used to store DHIS2 related configurations. For instance, we can store username/password combination for a DHIS2 instance like:

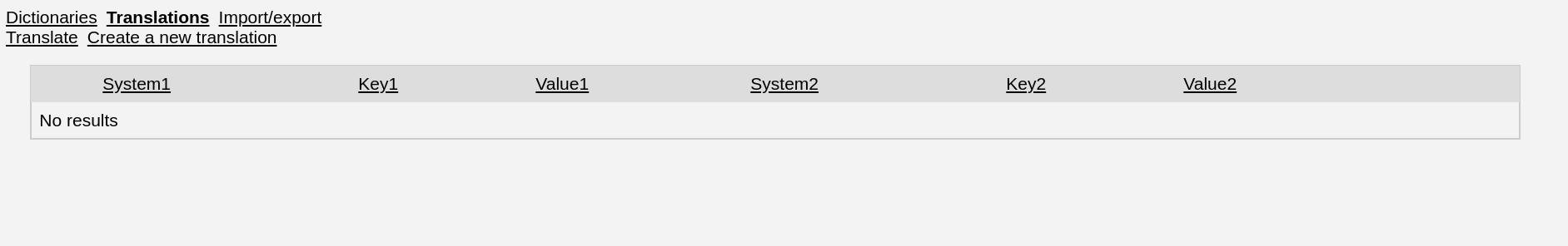


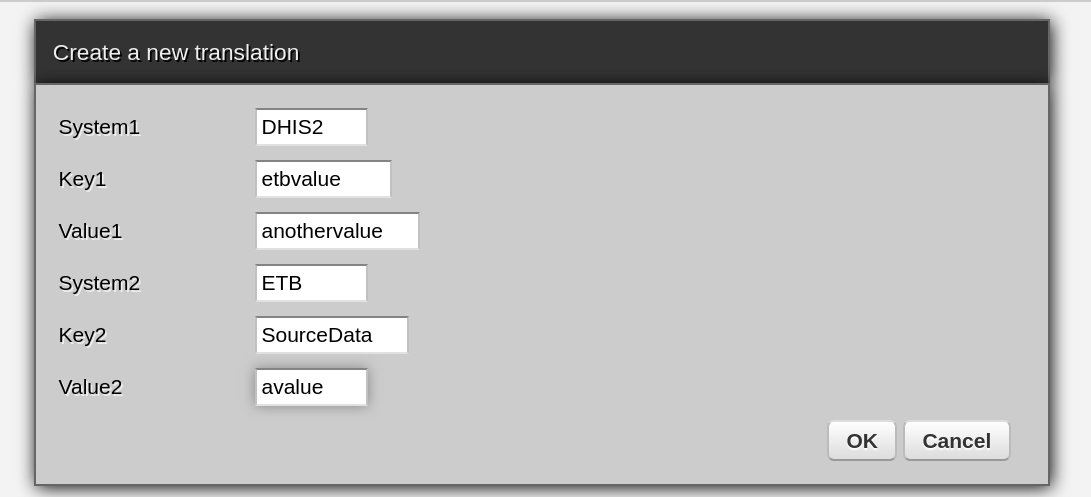




## Translations

Translations in Zato is very useful for integrating multiple systems. For instance, we can store a mapping of variables in Malaria Surveillance System to their data elements in HMIS2 based DHIS2 system.





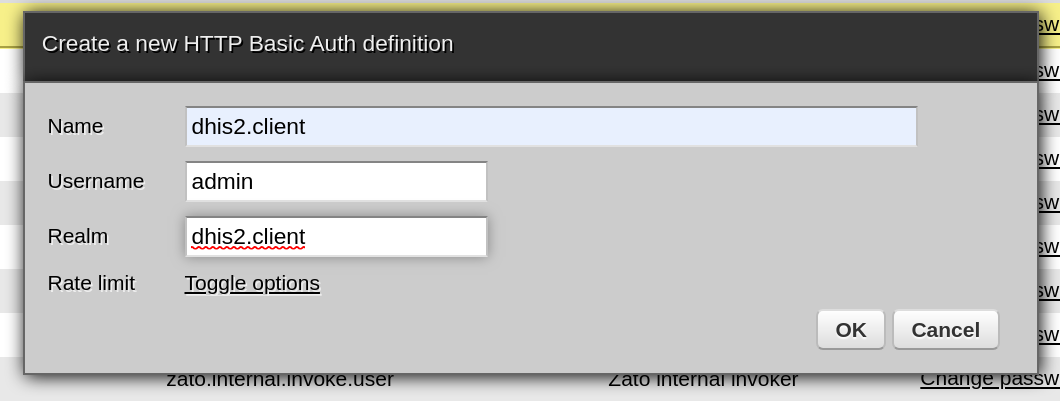
# Zato Use Case: Creating a DHIS2 based service

Organisation Units are the basic unit of data collection in interoperable health systems. Consistent Organisation Units are useful when integrating diverse health information systems. Let’s consider the case of HMIS. The source of truth for organisation units is the HMIS based DHIS2 system. In theory, other systems could query HMIS based DHIS2 instance to get updated list of organisation units. However, a vanilla DHIS2 instance doesn’t permit you to make unauthenticated queries to /api/organisationUnits. It’s also not scalable to provide each new system with a credential to access this API. To get around this, we’ll create a DHIS2 based service that will use stored credentials to query DHIS2 APIs and expose this over a HTTP endpoint that doesn’t require any authentication.

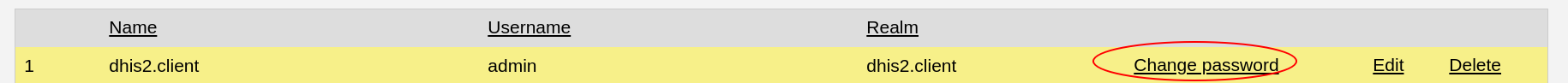
## Step 1: Add Authentication Details

Go to ***Security -> HTTP Basic Auth*** and create a new HTTP Basic Auth definition:

<https://zato.io/docs/web-admin/security/basic-auth.html>



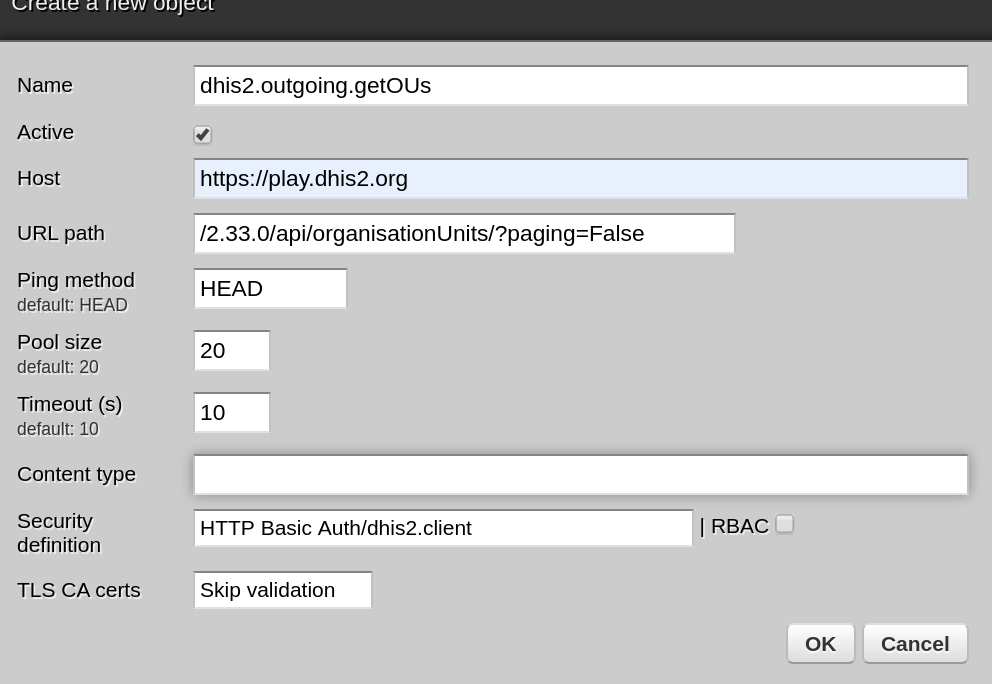
For outgoing connections, realm doesn’t matter. You can set it to anything. Once added, click on “change password” to set the password:



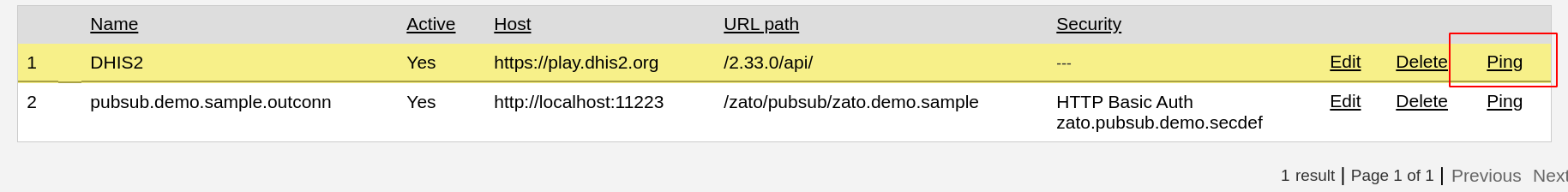
## 

## Step 2 : Add Outgoing REST connection to DHIS2 Organisation Unit API

Go to ***Connections -> Outgoing -> REST and add a new outgoing co***



Try pinging it to confirm credentials and URL paths are correct.



## Step 3: Create a service that queries this API and returns results:

$ nano dhis2\_ou.py

Paste the following:

from zato.server.service import Service

class GetOrganizations(Service):

class SimpleIO:

input\_required = ()

output\_required = ('data')

def handle(self):

syst = self.outgoing.plain\_http.get('dhis2.outgoing.getOUs')

response = syst.conn.get(self.cid)

self.response.payload['data'] = response.json()

Use *Ctrl-X* and then *Y* to save.

## Step 4: Deploy the service

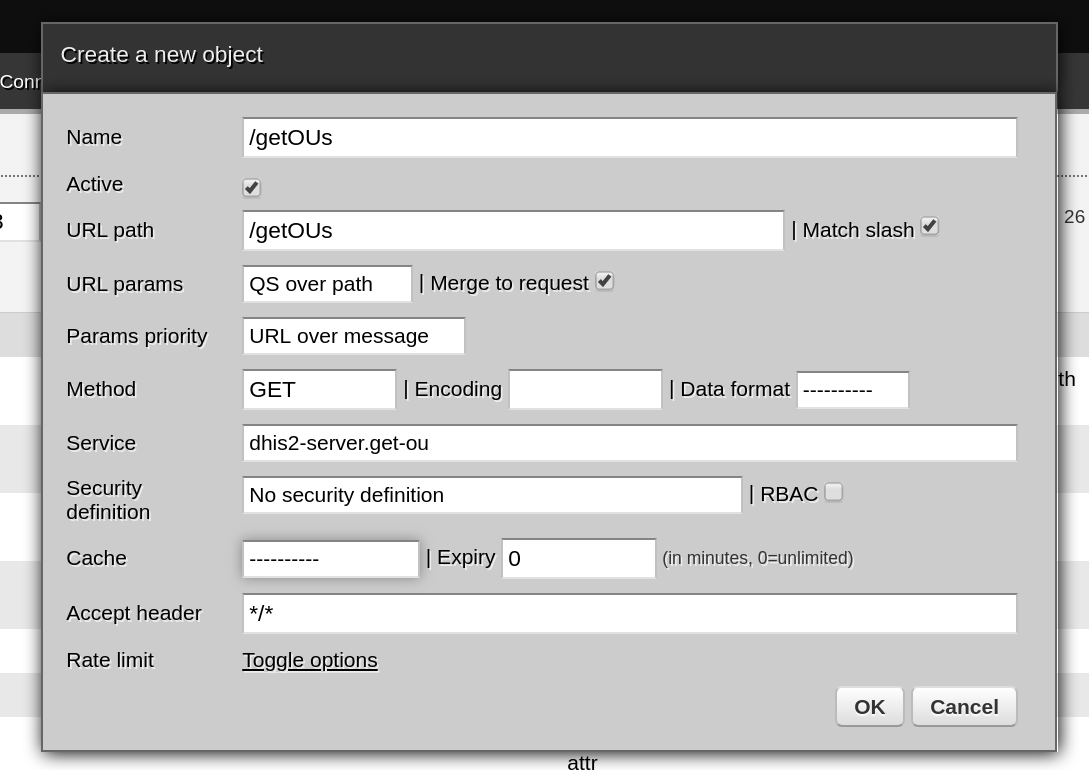
$ cp dhis2\_ou.py $path/server1/pickup/incoming/services

Once you do this, the cluster will pick up this service which you can verify by going to “Services” section of the admin interface:



## Step 5: Create an unauthenticated REST API that calls this service

To invoke the service we just created, add a new REST connection from : ***Connections-> Channels -> REST***



The URL path is what other clients will use when querying the Zato server to call this service. Security definition specifies the kind of authentication we want for this service. In this case, set to “No Security definition”.

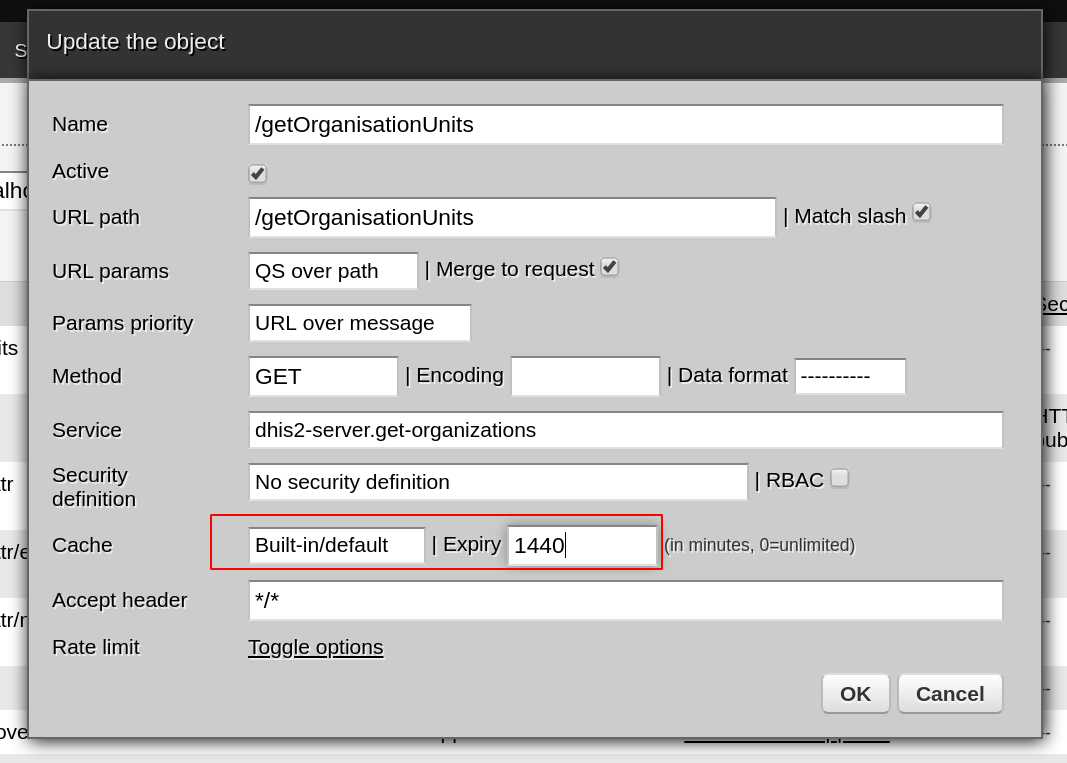
Once added, you can test if the service is working or not using:

$ curl localhost:11223/getOUs

The service is ready now.

## Note: Caching

Querying full list of organisation units everytime can be resource-intensive to the DHIS2 instance. We can setup a cache to cache the results of service invocation in Zato so instead of running the service again and again, Zato just returns the output from cache until the cache expires. To enable caching in the service we created, edit it and set cache, for instance to 1440 minutes (a day):



After this is done, subsequent queries to this service will be very fast.