**Change Environment Variables of a Module**

Using the single server deployment guide, but not deploying with Vagrant, requires to make some changes to the deployment descriptions in that guide. Roughly speaking, it suffices not to use the Vagrantfile proposed in that guide and to use the IP address of the host that you are deploying on instead of the IP 10.0.2.15 of the Vagrant host (for example in "okapiurl" in okapi.conf or as the env var "DB_HOST"). And to use the directory folio-install/runbooks/single-server (that you have checked out from github) instead of /vagrant. But I discovered that, since mod-pubsub and Kafka came into the play (thus since Q1-2020), that's not all.

Modules start up with environment variables, and if you don't change them, the default environment variables are being used. Some environment variables are being set in the single server prescription, namely those for connecting the modules to the database. They are being set by posting global environment variables to Okapi, like e.g. this:

```bash
curl -w '
' -D - -X POST -H "Content-Type: application/json" -d "{\"name\":\"DB_HOST\",\"value\":\"10.0.2.15\"}" http://localhost:9130/_/env
```

For mod-pubsub, this is not enough. In particular, mod-pubsub needs to connect to Kafka and uses the env vars KAFKA_HOST and KAFKA_PORT for this, as can be seen e.g. here [https://hub.docker.com/r/folioorg/mod-pubsub](https://hub.docker.com/r/folioorg/mod-pubsub). It also needs OKAPI_URL to be set to the correct value. The default values are connecting to the Vagrant host 10.0.2.15.

I applied the following steps to change the environment variables of a module. Firstly, it depends upon whether the module has already been deployed or not:

### 1 When the module has already been deployed

In this case, the module first needs to be undeployed, and then re-deployed. There are two ways to do the re-deploy:

#### 1.1 Deploying without Okapi

**Prescription:**

1. Create a new, running container for module pub-sub by hand (or by docker standard tools, that is) (or you might use your own deployment method)

```bash
# 1.) Pull the desired module+version out of dockerhub and build a container from the image:
docker pull folioorg/mod-pubsub:1.2.5
# 1.1) Run the container and pass the desired environment variables to it (those which are deviating from the default values) (this example is running on localhost = 10.9.2.62):
docker run --detach --name mod-pubsub-1.2.5 -e KAFKA_HOST=10.9.2.62 -e OKAPI_URL="http://10.9.2.62:9130" -e DB_HOST=10.9.2.62 -e DB_PORT=5432 -e DB_USERNAME=okapi -e DB_PASSWORD=okapi25 -e DB_DATABASE=okapi folioorg/mod-pubsub:1.2.5
```

2. Build deployment descriptor

First we need to find out, where in the docker network the module has been deployed (by docker); we are using "docker inspect" for this:

```bash
docker inspect -f '{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' mod-pubsub-1.2.5
cat > pubsub-deployment-descriptor.json <<END
{
    "srvcId": "mod-pubsub-1.2.5",
    "instId": "mod-pubsub-1.2.5",
    "url": "http://172.17.0.54:8081"
}
END
```

3. Disable all modules for the tenant, except for Okapi:

Fetch the list of modules for your tenant and write them into a file:

```bash
curl -w '
' -XGET http://localhost:9130/_/proxy/tenants/diku/modules > disable_modules.sh
```

Edit this file to make it a shell script which sends "DELETE" to `_/proxy/tenants/<tenantId>/modules/<moduleId>` for each module in the list. But make sure that you don't disable Okapi (kick it out of the list, first) !!!

```bash
curl -w '
' -XDELETE http://localhost:9130/_/proxy/tenants/diku/modules/folio_calendar-1.0.100018
curl -w '
' -XDELETE http://localhost:9130/_/proxy/tenants/diku/modules/folio_checkin-1.1.100049
curl -w '
' -XDELETE http://localhost:9130/_/proxy/tenants/diku/modules/folio_checkout-1.1.4000158
...```
Run the shell script multiple times. Because of missing dependencies, not all modules are being deleted. Delete the missing dependencies in the right order by hand (send "DELETE" to the modules which depend on other modules that cannot be deleted). At the end you should only have the Okapi module left:

```
curl -w "\n" -X GET http://localhost:9130/_/proxy/tenants/diku/modules
 [ { "id": "okapi-3.1.2" } ]
```

4. Underdeploy the module instance of mod-pubsub which has been registered as a service to Okapi (the one which is running with the wrong env vars):

```
# First get Service-Id and InstId of the module:
curl -w "\n" -X GET http://localhost:9130/_/discovery/modules | jq '.[]. | .srvcId + "/" + .instId' | grep pubsub
curl -w "\n" -D - -X DELETE http://localhost:9130/_/discovery/modules/mod-pubsub-1.2.5/8cd2ed01-08b3-44f1-8935-ec57a65f386b
HTTP/1.1 204 No Content
Content-Type: application/json
X-Okapi-Trace: DELETE okapi-3.1.2 /_/discovery/modules/mod-pubsub-1.2.5/8cd2ed01-08b3-44f1-8935-ec57a65f386b: 204 10759321us
```

5. Send the new deployment descriptor of mod-pubsub to $OKAPI_URL/_/discovery/modules:

```
curl -i -w "\n" -X POST -H 'Content-type: application/json' -d @pubsub-deployment-descriptor.json http://localhost:9130/_/discovery/modules
HTTP/1.1 201 Created
Content-Type: application/json
Location: /_/discovery/modules/mod-pubsub-1.2.5/8cd2ed01-08b3-44f1-8935-ec57a65f386b: mod-pubsub-1.2.5
X-Okapi-Trace: POST okapi-3.1.2 /_/discovery/modules: 201 9906us
content-length: 105

  { "instId": "mod-pubsub-1.2.5", "srvcId": "mod-pubsub-1.2.5", "url": "http://172.17.0.54:8081" }
```

6. Re-enable all modules for your tenant:

```
# Enable all backend modules:
# Also enable frontend modules and edge modules:
```

Do a health check to see if everything is O.K.:

```
curl -w "\n" -X GET http://localhost:9130/_/discovery/health
# => All "OK"
```

1.2 Deploying with Okapi

Do deploy with Okapi, the launch descriptor (a part of the module descriptor) needs to be re-posted to Okapi. Okapi only lets you do that if the module has not been enabled for any tenant.

(Thanks to Florian Rueckelshausen):

You can only post the modified module descriptor to OKAPI, once the old one has been deleted.

Here is a prescription:
# 1. Copy module descriptor into a file

curl -w \n-D - http://localhost:9130/_/proxy/modules/mod-pubsub-1.2.5 > pubsubdescriptor.json

# 2. Edit module descriptor

Change KAFKA_HOST and OKAPI_URL:

```json
{
    "name": "KAFKA_HOST",
    "value": "10.9.2.62"
},
{
    "name": "OKAPI_URL",
    "value": "http://10.9.2.62:9130"
}
```

Save modified module descriptor.

# 3. Disable all modules, except for Okapi => Equals to step 3. of 1.1

curl -w \n-XDELETE curl -w \n-XDELETE http://localhost:9130/_/proxy/tenants/diku/modules/...

# 4. Undeploy Pubsub

curl -w \n-D -X DELETE http://localhost:9130/_/discovery/modules/mod-pubsub-1.2.5

# 5. Delete module descriptor of mod-pubsub

curl -X DELETE -w "\n"http://localhost:9130/_/proxy/modules/mod-pubsub-1.2.5

# 6. Send modified module descriptor to Okapi

curl -w \n-D -X POST -H "Content-Type: application/json" -d @pubsubdescriptor.json http://localhost:9130/_/proxy/modules

# 7. Redploy mod-pubsub

cat > okapi-deploy-pubsub.json <<END
{
    "srvcId": "mod-pubsub-1.2.5",
    "nodeId": "localhost"
}
END

curl -w \n-D -s -X POST -H "Content-type: application/json" "
-d @okapi-deploy-pubsub.json http://localhost:9130/_/discovery/modules

# 8. Re-enable all modules == Step 6 of 1.1

2 When the module has not yet been deployed

This use case typically occurs when you spin up a new system or re-deploy the complete system, e.g. after a major release change.

2.1 Deployment without Okapi

Tweak the environment variables of the module in the deployment tool of your choice (e.g. in Rancher-UI, as command line parameters to "docker", ...)

2.2 Deployment with Okapi

Two different methods come to my mind:

2.2.1 Set env vars in the module descriptor (to be more precise: in the launch descriptor)

If you follow https://github.com/folio-org/folio-install/tree/master/runbooks/single-server, then after the step "Install and configure Okapi" : "Pull module descriptors from the central registry", fetch the single module descriptor for mod-pubsub seperately:

```
wget http://folio-registry.aws.indexdata.com/_/proxy/modules/mod-pubsub-1.2.5 -O pubsub-module-descriptor.json
```

Change variables therein manually; adjust the launch descriptor inside the module descriptor:
vim pubsub-module-descriptor.json
"launchDescriptor": {
    "dockerImage": "folioorg/mod-pubsub:1.2.5",
    "dockerPull": true,
    "env": [
        { "name": "JAVA_OPTIONS", "value": "-XX:MaxRAMPercentage=66.0 -XX:+HeapDumpOnOutOfMemoryError" },
        { "name": "DB_HOST", "value": "10.9.2.62" },
        { "name": "DB_PORT", "value": "5432" },
        { "name": "DB_USERNAME", "value": "okapi" },
        { "name": "DB_PASSWORD", "value": "okapi25" },
        { "name": "DB_DATABASE", "value": "okapi" },
        { "name": "DB_QUERYTIMEOUT", "value": "60000" },
        { "name": "DB_CHARSET", "value": "UTF-8" },
        { "name": "DB_MAXPOOLSIZE", "value": "5" },
        { "name": "KAFKA_HOST", "value": "10.9.2.62" },
        { "name": "KAFKA_PORT", "value": "9092" },
        { "name": "OKAPI_URL", "value": "http://10.9.2.62:9130" }
    ]
},

Overwrite the module descriptor in /_/proxy/modules überschreiben (this is possible, because the module has not yet been deployed and enabled):
curl -i -w "
-X POST -H 'Content-type: application/json' -d @pubsub-module-descriptor.json http://localhost:9130/_/proxy/modules

Now proceed with the standard installation prescription (Create tenant, ..., deploy and enable all modules with Okapi (deploy=true)).

2.2.2 Set env vars as global environment variables (⚠️ will that work ? Haven't tested that, yet!)

You could try to do the system installation following the standard installation prescription (for a single server with Vagrant), so as if you would use the default module environment variables. Not changing the module descriptor. But, instead of the step "Post data source information to the Okapi environment for use by deployed modules" you post in addition the new module environment variables as global environment variables (in addition to the database connection parameters). You expect them to be effective for all folio modules, then:
curl -w "
-X POST -H "Content-Type: application/json" -d {"name": "KAFKA_HOST", "value": "10.9.2.62"} http://localhost:9130/_/env
curl -w "

Then proceed as usual, deploying all backend modules for the tenant with Okapi by doing a

curl -w "
-X POST -H "Content-Type: application/json" -d @/usr/folio/platform-complete/okapi-install.json

As I said, the proof that this will work is still missing.