**Creating Tenants**

first we create a description json files for our two tenants , in paths /tmp/okapi-testtenant1.json , /tmp/okapi-testtenant2. Json , with names test tenant1 and test tenant2 and id testtenant1 ,testtenant2.

For the first one :

cat > /tmp/okapi-testtenant1.json <<END

{ "id": "testtenant1","name": "Test Tenant1", "description": "the first tenant"}

END

Now for the second one:

cat > /tmp/okapi-testtenant2.json <<END

{ "id": "testtenant2","name": “Test Tenant2", "description": "the second tenant"}

END

Then we post the two tenants to /-/proxy:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d @/tmp/okapi-testtenant1.json \

http://localhost:9130/\_/proxy/tenants

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d @/tmp/okapi-testtenant2.json \

http://localhost:9130/\_/proxy/tenants

Okapi shows that for the first one:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants : 201 16368us

content-length: 89

{

"id" : "testtenant1",

"name" : "Test Tenant1",

"description" : "the first tenant"}

And for the second one :-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants : 201 23345us

content-length: 91

{

"id" : "testenant2"

"name" : "Test Tenant2",

"description" : "the second tenant"}

}

now the two tenants are created and saved to tenants table in okapi database in postgresql.

**The Modules**

1.Mod-permissions:-

This module stores and contains a hierarchy of permissions and its sub permissions , it makes permissions act like roles , it’s a pre\_request for mod-authtoken.

First we Fetch and compile module using those commands , after writing it the module will be built and its files would be created.

git clone --recursive https://github.com/folio-org/mod-permissions

cd mod-permissions

git checkout master

mvn clean install

Now as module files are created , module descriptor file is generated and we need to post the module to okapi(to /-/ proxy).

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-permissions/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-permissions/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-permissions.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-permissions.json \

http://localhost:9130/\_/discovery/modules

Now the permissions module is running in okapi.

2.Mod-users:-

It manages users in folio systems.

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-users

cd mod-users

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy).

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-users/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

njn

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-users.json \

http://localhost:9130/\_/discovery/modules

cat mod-users/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-users.json

Now the users module is running in okapi.

3.Mod-login:-

It verifies the identity of user and gets a JWT for him , which the user uses to access system , by default it validates users by user name and password.

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-login

cd mod-login

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy)

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-login/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-login/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-login.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-login.json \

http://localhost:9130/\_/discovery/modules

Now the login module is running in okapi.

4.Mod-authtoken:-

It filters the proxy traffic , sees the token and determines whether it’s valid or not ,and sees permissions available for a user makes an action depending on those permissions and requests user listed in his given path , in addition to working as a filter it has endpoints that can be accessed as a part of okapi system :-

* **/token:** returns an access token (JWT),it requires the permission **auth.signtoken** , accepts post of a json object,it’s returned inside a json object.
* **/refreshtoken:**returns a refresh token (JWE),requires the permission **auth.signrefreshtoken** ,accepts post of a json object.
* **/refresh :**takes a valid refresh token , returns new access token inside a json object ,accepts a post of json object , with filed **refreshtoken** required .

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-authtoken

cd mod-authtoken

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy)

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-authtoken/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-authtoken/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-authtoken.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-authtoken.json \

http://localhost:9130/\_/discovery/modules

Now the authtoken module is running in okapi.

5.Mod-inventory-storage:-

Works as a complement for mod-inventory as it Provides storage for based on postgresql.

So we need to deploy it before deploying mod-inventory .

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-inventory-storage

cd mod-inventory-storage

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy)

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-inventory-storage /target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-inventory-storage /target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-inventory-storage.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy- inventory-storage.json \

http://localhost:9130/\_/discovery/modules

Now the inventory storage module is running in okapi.

We also need to Fitch and compile mod-source-record-storage

Before mod-inventory as mod-inventory pre-requires it.

* **Mod-source-record-storage:-**

Provides PostgreSQL based storage to complement the data import module.

Fetching and building it:-

git clone --recursive https://github.com/folio-org/mod-source-record-storage

cd mod-source-record-storage

git checkout master

mvn clean install

Posting it to okapi:-

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-source-record-storage e /target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

deploying the module :-

cat mod-source-record-storage/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-source-record-storage.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-source-record-storage.json \

http://localhost:9130/\_/discovery/modules

6.Mod-inventory:-

Provides basic physical item inventory management, also needs mod-inventory-storage, mod-source-record-storage as pre-requirement.

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-inventory

cd mod-inventory

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy)

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-inventory/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-inventory/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-inventory.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-inventory.json \

http://localhost:9130/\_/discovery/modules

Now the inventory module is running in okapi

7.Mod-orders-storage:-

Works as Persistent storage of order data , it’s a pre-request for mod-orders.

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-orders-storage

cd mod-orders-storage

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy)

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-orders-storage/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-orders-storage/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-orders-storage.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-orders-storage.json \

http://localhost:9130/\_/discovery/modules

Now the orders storage module is running in okapi

We also need to Fitch and compile mod-configuration , mod-organizations-storage, before mod-orders as mod-orders pre-requires them.

**mod-configuration :-**

built using the raml-module-builder, using the PostgreSQL async client to implement some basic configuration APIs. there are many features that the mod-configuration module inherits from the raml-module-builder (The goal of it is to abstract away as much boilerplate functionality as possible and allow a developer to focus on implementing business functions. In other words: **simplify the process of developing a micro service module).**

Fetching and building it:-

git clone --recursive https://github.com/folio-org/mod-configuration

cd mod-configuration

git checkout master

mvn clean install

Posting it to okapi:-

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-configuration/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

deploying the module :-

cat mod-configuration/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-configuration.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-configuration.json \

http://localhost:9130/\_/discovery/modules

**mod-organizations-storage:-**

it works as Persistent storage of organizations data.

Fetching and building it:-

git clone --recursive https://github.com/folio-org/mod-organizations-storage

cd mod-organizations-storage

git checkout master

mvn clean install

Posting it to okapi:-

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-organizations-storage/target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

deploying the module :-

cat mod-organizations-storage/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-organizations-storage.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-organizations-storage.json \

http://localhost:9130/\_/discovery/modules

8-Mod-orders:-

Works as an orders business logic module, it pre-requests mod-orders-storage, mod-configuration and mod-organizations-storage.

First we Fetch and compile module using those commands.

git clone --recursive https://github.com/folio-org/mod-orders

cd mod-orders

git checkout master

mvn clean install

Now we post the module to okapi(to /-/ proxy)

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @mod-orders /target/ModuleDescriptor.json \

http://localhost:9130/\_/proxy/modules

Then we need to deploy the module (posting it to discovery).

cat mod-orders/target/DeploymentDescriptor.json \

| sed 's/..\///' | sed 's/embed\_postgres=true//' > /tmp/deploy-orders.json

curl -w '\n' -D - -X POST \

-H "Content-type: application/json" \

-d @/tmp/deploy-orders.json \

http://localhost:9130/\_/discovery/modules

Now the orders module is running in okapi.

**Enabling Modules For Tenants**

Okapi is a multi-tenant system, so each request must be done on behalf of some tenant, so after deploying modules, creating tenants, we need to enable those modules for our two tenants.

* Mod-login

We write the following lines to enable the login modules to the first tenant:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-login"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

Okapi responds with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-login-6.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 2866151us

content-length: 39

{"id" : "mod-login-6.1.0-SNAPSHOT"}

And now we do the same for the second, first we enable the login module to the second tenant:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-login"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-login-6.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 2866151us

content-length: 39

{"id" : "mod-login-6.1.0-SNAPSHOT"}

* Mod-users

Enabling the users module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-users"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-users"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-users-15.7.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 2263868us

content-length: 40

{

"id" : "mod-users-15.7.0-SNAPSHOT"

}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-users-15.7.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 2263868us

content-length: 40

{

"id" : "mod-users-15.7.0-SNAPSHOT"

}

* Mod-permissions

Enabling the permissions module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-permissions"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-permissions"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-permissions-5.9.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 2933974us

content-length: 45

{

"id" : "mod-permissions-5.9.0-SNAPSHOT"

}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-permissions-5.9.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 971559us

content-length: 45

{

"id" : "mod-permissions-5.9.0-SNAPSHOT"

}

* Mod-authtoken

Enabling the authtoken module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-authtoken"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-authtoken"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-authtoken-2.3.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 23214us

content-length: 43

{

"id" : "mod-authtoken-2.3.0-SNAPSHOT"

}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-authtoken-2.3.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 16224us

content-length: 43

{

"id" : "mod-authtoken-2.3.0-SNAPSHOT"

}

* Mod-inventory

Before we enable this module for tenants it pre requests enabling the **mod-inventory-storage** and **mod-source-record-storage** in order to enable mod-inventory to our tenants so we do it first then we enable it to tenants

Enabling the inventory module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-inventory"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-inventory"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-inventory-12.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 396163us

content-length: 44

{

"id" : "mod-inventory-12.1.0-SNAPSHOT"

}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-inventory-12.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 366677us

content-length: 44

{

"id" : "mod-inventory-12.1.0-SNAPSHOT"

}

* Mod-inventory-storage

Now as we mentioned before,this modules should be enabled for tenants before enabling mod-invntory

Enabling the inventory storage module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-inventory-storage"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-inventory-storage"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-inventory-storage-17.0.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 9823301us

content-length: 52

{

"id" : "mod-inventory-storage-17.0.0-SNAPSHOT"

}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-inventory-storage-17.0.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 3149077us

content-length: 52

{

"id" : "mod-inventory-storage-17.0.0-SNAPSHOT"

}

* Mode-orders

In order to enable this module for our tenants mod-orders-storage, mod-configuration and mod-organizations-storage should be enabled for tenants first.

Enabling the orders module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-orders"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-orders"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-orders-7.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 353181us

content-length: 40

{"id" : "mod-orders-7.1.0-SNAPSHOT"}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-orders-7.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 405192us

content-length: 40

{"id" : "mod-orders-7.1.0-SNAPSHOT"}

* Mod-orders-storage

Enabling the orders storage module to the tenants:-

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-orders-storage"}' \

http://localhost:9130/\_/proxy/tenants/testtenant1/modules

curl -w '\n' -X POST -D - \

-H "Content-type: application/json" \

-d'{"id":"mod-orders-storage"}' \

http://localhost:9130/\_/proxy/tenants/testtenant2/modules

Okapi responds for each tenant with:-

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant1/modules/mod-orders-storage-7.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant1/modules : 201 4464496us

content-length: 48

{"id" : "mod-orders-storage-7.1.0-SNAPSHOT"}

HTTP/1.1 201 Created

Content-Type: application/json

Location: /\_/proxy/tenants/testtenant2/modules/mod-orders-storage-7.1.0-SNAPSHOT

X-Okapi-Trace: POST okapi-2.32.0-SNAPSHOT /\_/proxy/tenants/testtenant2/modules : 201 1125300us

content-length: 48

{ "id" : "mod-orders-storage-7.1.0-SNAPSHOT"}

**Concluosion**

So overall in this document we covered

* creating two tenants .
* building and running login , users authtoken , permissions ,inventory, inventory storage , orders and orders storage modules .
* enabling each module for our two tenants .