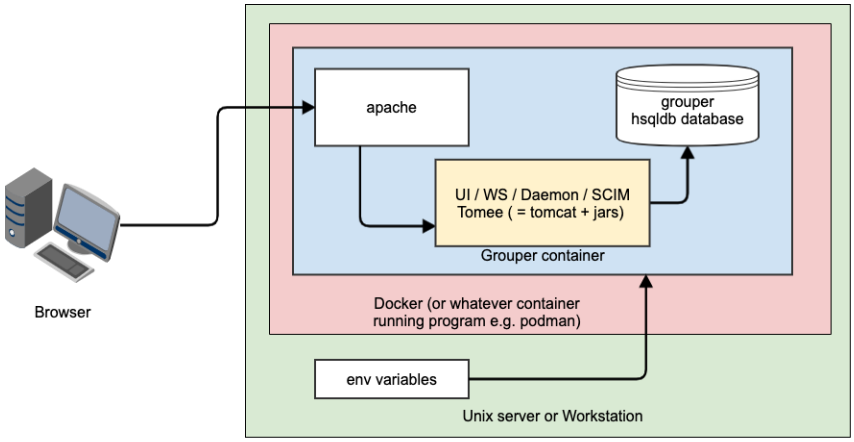
Install the Grouper v2.5 container maturity level -1 quick start

Description

This quickstart is the easiest way to start Grouper. You need a computer with Docker (or another container technology that runs Docker format containers). Run the container and it will start an embedded HSQLDB database (that does not persist across restarts unless you mount the database directory outside of your container... as outlined below).

All Grouper processes will run in this container at once (UI/WS/daemon/SCIM). As you evolve your Grouper practice you will run your processes in separate containers.



Get a server

Here is an example with AWS, basically for this example you need a Unix-based server (or Mac). Install Docker as well

Install the container

1. See which version of Grouper to run (at least v2.5.27)
2. Issue a run command to run the quick start.

    a. Note, for the morph string encrypt and quick start pass, just make up a 16 char alphanumeric string or generate from a password manager.
    b. Note, this is ***not good security***. It is for quick starts only. As you evolve to maturity level 0, you can set a different password encrypted in the database which will not be in a script file or in an env variable, and you can further evolve to Shibboleth or another authentication system.
    c. Note: the first port is the port for apache SSL, change that to whatever you need on your host (i.e. one that is not in use and one that you can get to through the network)

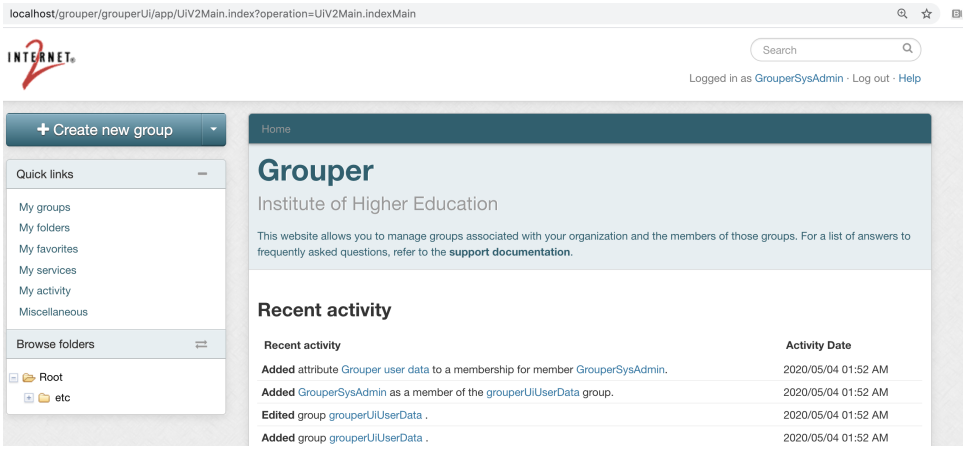
```
$ docker run --detach --name grouper-qs \
  --publish 443:443 -e GROUPER_MORPHSTRING_ENCRYPT_KEY=******** \
  -e GROUPERSYSTEM_QUICKSTART_PASS=******** i2incommon/grouper:2.5.XX quickstart
```

e.g.

```
docker run --detach --name grouper-qs \
  --publish 443:443 -e GROUPER_MORPHSTRING_ENCRYPT_KEY=abcdefg12345dontUseThis \
  -e GROUPERSYSTEM_QUICKSTART_PASS=thisPassIsCopyrightedDontUse i2incommon/grouper:2.5.27
quickstart
```

3. Log in to UI (note, the first log in can take a minute as HSQLDB database is started and initted.)
    a. Note: change "localhost" to your server IP address or domain name, and add the port if not 443. e.g. https://server.whatever.edu:1234/grouper/

```
Go to: https://localhost/grouper/
Log in with username : GrouperSystem
Password is the password you specified in the GrouperSystem QuickStart pass
```



4. Try a web service call

```
Get the client out of the container (or download from maven)

$ docker cp grouper-qs:/opt/grouper/grouperWebapp/WEB-INF/lib/grouperClient-2.5.XX.jar .

Now you should have a grouper client jar in your directory
Make a config file in the same directory

$ vi grouper.client.properties

#  note add the port after localhost if not using 443
grouperClient.webService.url = https://localhost/grouper-ws/servicesRest
grouperClient.webService.login = GrouperSystem
grouperClient.webService.password = ****** is the password you specified in the GrouperSystem QuickStart
pass

# turn off SSL until a real SSL certificate is installed
# NOTE, THIS IS NOT GOOD SECURITY AND IS FOR THE QUICK START ONLY!
grouperClient.https.customSocketFactory = edu.internet2.middleware.grouperClient.ssl.EasySslSocketFactory



$ java -jar grouperClient-2.5.XX.jar --operation=getSubjectsWs --subjectIds=GrouperSystem
Index: 0: success: T, code: SUCCESS, subject: GrouperSystem
$
```

```
grouperContainer $ java -jar grouperClient-2.5.0-SNAPSHOT.jar --operation=getSubjectsWs --
subjectIds=GrouperSystem --debug=true
Reading resource: grouper.client.properties, from: /Users/mchyzer/grouper/docker/grouperContainer
/grouper.client.properties
WebService: connecting as user: 'GrouperSystem'
WebService: connecting to URL: 'https://localhost/grouper-ws/servicesRest/2.5.0-SNAPSHOT/subjects'

################ REQUEST START (indented) ###############

POST /grouper-ws/servicesRest/2.5.0-SNAPSHOT/subjects HTTP/1.1
```

```
Connection: close
Authorization: Basic xxxxxxxxxxxxxxx
User-Agent: Jakarta Commons-HttpClient/3.1
Host: localhost:-1
Content-Length: 161
Content-Type: text/xml; charset=UTF-8

<WsRestGetSubjectsRequest>
  <wsSubjectLookups>
    <WsSubjectLookup>
      <subjectId>GrouperSystem</subjectId>
    </WsSubjectLookup>
  </wsSubjectLookups>
</WsRestGetSubjectsRequest>

################ REQUEST END ###############



################ RESPONSE START (indented) ###############

HTTP/1.1 200 OK
Date: Mon, 04 May 2020 02:38:16 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips
Strict-Transport-Security: max-age=15768000
Set-Cookie: JSESSIONID=xxxxxxxxxxxx; HttpOnly
X-Grouper-resultCode: SUCCESS
X-Grouper-success: T
X-Grouper-resultCode2: NONE
Content-Type: application/xml;charset=UTF-8
Connection: close
Transfer-Encoding: chunked

<WsGetSubjectsResults>
  <wsSubjects>
    <WsSubject>
      <resultCode>SUCCESS</resultCode>
      <success>T</success>
      <id>GrouperSystem</id>
      <name>GrouperSysAdmin</name>
      <sourceId>g:isa</sourceId>
    </WsSubject>
  </wsSubjects>
  <resultMetadata>
    <resultCode>SUCCESS</resultCode>
    <resultMessage>Queried 1 subjects</resultMessage>
    <success>T</success>
  </resultMetadata>
  <responseMetadata>
    <resultWarnings></resultWarnings>
    <millis>19</millis>
    <serverVersion>2.5.0-SNAPSHOT</serverVersion>
  </responseMetadata>
</WsGetSubjectsResults>

################ RESPONSE END ###############


Output template: Index: ${index}: success: ${success}, code: ${wsSubject.resultCode}, subject:
${wsSubject.id}, available variables: wsGetSubjectsResults, grouperClientUtils, index, wsSubject,
wsGroup, success
Index: 0: success: T, code: SUCCESS, subject: GrouperSystem
Elapsed time: 612ms
grouperContainer $
```

Advanced

1. Verify the digest of the image. This is a best practice when using Grouper images

a. Pull the image

```
bin $ docker pull i2incommon/grouper:2.5.XX
```

b. Make sure the digest is correct (from release notes page)

```
[root@ip-172-30-3-152 ~]# docker image inspect i2incommon/grouper:2.5.XX | grep i2incommon
/grouper@sha256
            "i2incommon/grouper@sha256:b675bb410bf873xxxxxxxxxxxxxx5e58a3a42a8048381a33b79fd19"
```

2. Make the start command in a script so you have it to run it later consistently

```
grouperContainer $ vi grouperQsDockerRun.sh

#!/bin/bash  (or whatever shell)
# this is the run command from above that you diddocker run --detach --name grouper-qs \
  --publish 443:443 -e GROUPER_MORPHSTRING_ENCRYPT_KEY=******** \
  -e GROUPERSYSTEM_QUICKSTART_PASS=******** i2incommon/grouper:2.5.XX quickstart


grouperContainer $ chmod +x grouperQsDockerRun.sh

# remove the old container if you want to run it again, or stop, or whatevergrouperContainer

$ docker rm -f grouper-qsgrouperContainer
$ ./grouperQsDockerRun.sh

(Optional) Check logs:

grouperContainer $ docker logs grouper-qs

(Optional) Shell in:

grouperContainer $ docker exec -it grouper-qs /bin/bash
```

3. (Optional) Mount your database files outside of Docker to persist your changes across container restarts. Note, this is still not a robust database, it is only for non production use.

```
$ mkdir hsqldb

Change your start command to include a mount of this directory

grouperContainer $ vi grouperQsDockerRun.sh

Add this mount in your command

--mount type=bind,src=/path/to/hsqldb,dst=/opt/hsqldb

You might need to open up permissions on that directory:

$ chmod 777 hsqldb

Delete the current container

$ docker rm -f grouper-qs

Start it again

$ ./grouperQsDockerRun.sh

You will see database files in that dir on your host

grouperContainer $ ls -latr hsqldb/
total 6192
drwxr-xr-x  22 mchyzer  staff      704 May  3 22:58 ..
drwxr-xr-x   2 mchyzer  staff       64 May  3 22:58 grouperHSQL.tmp
-rw-r--r--   1 mchyzer  staff     1536 May  3 22:58 grouperHSQL.script
-rw-r--r--   1 mchyzer  staff       85 May  3 22:58 grouperHSQL.properties
drwxrwxrwx   7 mchyzer  staff      224 May  3 22:58 .
-rw-r--r--   1 mchyzer  staff       16 May  3 23:00 grouperHSQL.lck
-rw-r--r--   1 mchyzer  staff  2854600 May  3 23:00 grouperHSQL.log
```

4. Evolve to maturity level 0