

Team:

PAVAN KUMAR KOIKUR SUDHAKAR

PRIYANKA SHARMA

ANTHONY SERRATO

Table of Contents

1) Installation.....	3
1.1 Install Kong.....	3
1.1.1 To start kong:.....	3
1.2 Setup Minio:.....	4
1.2.1 Start Minio.....	4
2) Setup Services.....	4
2.1 Start kong:.....	4
2.2 Start Minio:.....	5
2.3 Create tracks bucket in Minio.....	5
2.4 Connect kong with minio.....	5
2.5 Start 3 instances of microservices.....	6
2.6 Setup kong for load balancing user micro-service (3 instances).....	6
2.7 Setup kong for load balancing descriptions micro-service (3 instances).....	8
2.8 Setup kong for load balancing tracks micro-service (3 instances).....	9
2.9 Setup kong for load balancing Playlist micro-service (3 instances).....	10
3) How Install Memcached.....	11
4) How To Start Services.....	12
4.1 Initialize databases:.....	12
4.2 Start 3 instance of all microservices:.....	12
4.3 Start kong:.....	12
4.4 Start Minio:.....	12
4.5 Start XSPF generator service:.....	12
4.5 Start Memcached server:.....	13
5) Instructions to use the USER Micro-service:.....	14
5.1 Get User Deatils:.....	14
5.2 Authenticate user.....	14
5.3 Create User.....	14
5.4 Update User Password.....	15
5.5 Delete User.....	15
6) Instructions to use the TRACKS Micro-service:.....	16
6.1 Get Track:.....	16
6.2 Create Track.....	16
6.3 Edit Tracks.....	16
6.4 Delete Track.....	17
7) Instructions to use the DESCRIPTIONS Micro-service:.....	18
7.1 GET a User's Description of a track :.....	18

7.2 Create a User's Description of a track.....	18
8) Instructions to use the Playlist Micro-service:.....	19
8.1 GET a Playlist:.....	19
8.2 GET method all Playlists:.....	19
8.3 GET all Playlists created by a particular User:.....	19
8.4 Create a new Playlist:.....	19
8.5 Delete a Playlist :.....	20
9) XSPF Generator Microservice:.....	20
10) Run Tavern files.....	20

1) Installation

skip this section if have already installed kong and minio.

1.1 Install Kong

sudo apt-get update

sudo apt-get install openssl libpcre3 procps perl

download https://bintray.com/kong/kong-deb/download_file?file_path=kong-1.4.0.trusty.amd64.deb
and then

sudo dpkg -i kong-1.4.0.*.deb

sudo apt install --yes postgresql

sudo -u postgres psql -c "CREATE USER kong WITH ENCRYPTED PASSWORD 'kong'"

sudo -u postgres psql -c 'CREATE DATABASE kong OWNER kong'

sudo cp /etc/kong/kong.conf.default /etc/kong/kong.conf

sudoedit /etc/kong/kong.conf

-> search for pg_password

-> uncomment the line

-> **pg_password = kong**

1.1.1 To start kong:

sudo kong migrations bootstrap

ulimit -n 4096 && sudo kong start

Test Kong : curl -i http://localhost:8001/

1.2 Setup Minio:

Install Minio:

```
wget https://dl.min.io/server/minio/release/linux-amd64/minio
```

```
chmod +x minio
```

1.2.1 Start Minio

```
./minio server /home/student/data (if u use /data you will get error)
```

→ Copy minio tokens and keep for future

Endpoint: http://10.0.2.15:9000 http://127.0.0.1:9000

AccessKey: CHKH8D13MGHGWFOZR1TD

SecretKey: qD2VXhvJ3Yc3wXmmZdfsjuXwiK1xFso+ffYsWSeb

2) Setup Services

If you had already configured all the services you can skip this section

2.1 Start kong:

```
sudo kong migrations bootstrap
```

```
ulimit -n 4096 && sudo kong start
```

To check whether kong is working

```
-> curl -i http://localhost:8001/
```

2.2 Start Minio:

`./minio server /home/student/data`

→ Copy minio tokens and keep for future

Endpoint: `http://10.0.2.15:9000` `http://127.0.0.1:9000`

AccessKey: `CHKH8D13MGHWFOZR1TD`

SecretKey: `qD2VXhvJ3Yc3wXmmZdfsjvXwiK1xFso+ffYsWSeb`

Note: The accesskeys will differ from system to system

2.3 Create tracks bucket in Minio

Open minio browser -> `http://127.0.0.1:9000`

→ Login with credentials (AccessKey and SecretKey)

-> create new bucket named "tracks"

-> Edit the policy for this bucket to add the prefix * as Read Only.

2.4 Connect kong with minio

To setup minio to use the kong execute the following commands:

1. Create a service named media

`curl -i -X POST http://localhost:8001/services/ \`

`-d 'name=media' \`

`-d 'url=http://10.0.2.15:9000/tracks/'`

→ Output should be 201 created with details about the service in the json. Copy the value for **id**

2. Add the routes for the service media. Use the value of **id** for **service.id**

```
curl -i -X POST http://localhost:8001/routes/ \  
  -d 'hosts[]=localhost' \  
  -d 'paths[]=media' \  
  -d 'service.id=1f4aa3c8-d87f-4b3d-9bb4-b7267ff99207'
```

3. Now try opening http://localhost:8000/media/<file_name>

if service is up you will be able to access the file.

2.5 Start 3 instances of microservices

```
foreman start -c user=3,descriptions=3,tracks=3,playlist=3
```

2.6 Setup kong for load balancing user micro-service (3 instances)

Follow below instructions :

1. Setup the upstream

```
curl -X POST http://localhost:8001/upstreams \  
  --data "name=Users_Upstream"
```

2. Add three targets(user micro-service) generated by foreman to the upstream. Make sure port numbers are correct.

```
curl -X POST http://localhost:8001/upstreams/Users_Upstream/targets \  
--data "target=127.0.0.1:5000" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Users_Upstream/targets \  
--data "target=127.0.0.1:5001" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Users_Upstream/targets \  
--data "target=127.0.0.1:5002" --data "weight=100"
```

3. Create the service "<name>" pointing to upstream "<host>"
(path refers to the path which has to be appended to the target)

```
curl -X POST http://localhost:8001/services/ \  
--data "name=Users_Service" \  
--data "host=Users_Upstream" \  
--data "path=/api/v1/resources/user"
```

4. Add a Route as an entry-point into the Service
(paths here refers to the matching in the url)

```
curl -X POST http://localhost:8001/services/Users_Service/routes/ \  
--data "hosts[]=localhost" --data "paths=/api/v1/resources/user"
```

2.7 Setup kong for load balancing descriptions micro-service (3 instances)

Follow below instructions :

1. Setup the upstream

```
curl -X POST http://localhost:8001/upstreams \  
--data "name=Descirptions_Upstream"
```

2.Add three targets(descriptions micro-service) generated by foreman to the upstream. Make sure port numbers are correct.

```
curl -X POST http://localhost:8001/upstreams/Descirptions_Upstream/targets \  
--data "target=127.0.0.1:5100" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Descirptions_Upstream/targets \  
--data "target=127.0.0.1:5101" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Descirptions_Upstream/targets \  
--data "target=127.0.0.1:5102" --data "weight=100"
```

3. Create the service "<name>" pointing to upstream "<host>"
(path refers to the path which has to be appended to the target)

```
curl -X POST http://localhost:8001/services/ \  
--data "name=Descirptions_Service" \  
--data "host=Descirptions_Upstream" --data "path=/api/v1/resources/descriptions"
```


4. Add a Route as an entry-point into the Service
(paths here refers to the matching in the url)

```
curl -X POST http://localhost:8001/services/Descirptions_Service/routes/ \  
--data "hosts[]=localhost" --data "paths=/api/v1/resources/descriptions"
```

2.8 Setup kong for load balancing tracks micro-service (3 instances)

Follow below instructions :

1. Setup the upstream

```
curl -X POST http://localhost:8001/upstreams \  
--data "name=Tracks_Upstream"
```

2. Add three targets(tracks micro-service) generated by foreman to the upstream. Make sure port numbers are correct.

```
curl -X POST http://localhost:8001/upstreams/Tracks_Upstream/targets \  
--data "target=127.0.0.1:5200" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Tracks_Upstream/targets \  
--data "target=127.0.0.1:5201" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Tracks_Upstream/targets \  
--data "target=127.0.0.1:5202" --data "weight=100"
```

3. Create the service "<name>" pointing to upstream "<host>"
(path refers to the path which has to be appended to the target)

```
curl -X POST http://localhost:8001/services/ \  
--data "name=Tracks_Service" \  
--data "host=Tracks_Upstream" --data "path=/api/v1/resources/tracks"
```

4. Add a Route as an entry-point into the Service
(paths here refers to the matching in the url)

```
curl -X POST http://localhost:8001/services/Tracks_Service/routes/ \  
--data "hosts[]=localhost" --data "paths=/api/v1/resources/tracks"
```

2.9 Setup kong for load balancing Playlist micro-service (3 instances)

Follow below instructions :

1. Setup the upstream

```
curl -X POST http://localhost:8001/upstreams \  
--data "name=Playlist_Upstream"
```

2. Add three targets (playlist micro-service) generated by foreman to the upstream. Make sure port numbers are correct.

```
curl -X POST http://localhost:8001/upstreams/Playlist_Upstream/targets \
--data "target=127.0.0.1:5300" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Playlist_Upstream/targets \
--data "target=127.0.0.1:5301" --data "weight=100"
```

```
curl -X POST http://localhost:8001/upstreams/Playlist_Upstream/targets \
--data "target=127.0.0.1:5302" --data "weight=100"
```

3. Create the service "<name>" pointing to upstream "<host>"
(path refers to the path which has to be appended to the target)

```
curl -X POST http://localhost:8001/services/ \
--data "name=Playlist_Service" \
--data "host=Playlist_Upstream" --data "path=/api/v1/resources/playlist"
```

4. Add a Route as an entry-point into the Service
(paths here refers to the matching in the url)

```
curl -X POST http://localhost:8001/services/Playlist_Service/routes/ \
--data "hosts[]=localhost" --data "paths=/api/v1/resources/playlist"
```

3) How Install Memcached

```
pip3 install --user pymemcache
```

```
sudo apt-get install memcached
```

```
sudo service memcached start
```

4) How To Start Services

4.1 Initialize databases:

FLASK_APP=user flask init

FLASK_APP=tracks flask init

4.2 Start 3 instance of all microservices:

foreman start -c user=3,descriptions=3,tracks=3,playlist=3

4.3 Start kong:

sudo kong migrations bootstrap

ulimit -n 4096 && sudo kong start

4.4 Start Minio:

./minio server /home/student/data

4.5 Start XSPF generator service:

FLASK_APP=xspf_service flask run -p 5400

4.5 Start Memcached server:

To start

sudo service memcached start

To stop

sudo service memcached stop

5) Instructions to use the USER Micro-service:

To initialize the database use command:

FLASK_APP=user flask init

To start all the Micro-services use command: foreman start)

Note: if you want to use kong use url <http://localhost:8000/api/v1/resources/user>

5.1 Get User Deatils:

- To a retrieve User's profile (GET method) use the following curl command:

```
curl -v 'http://127.0.0.1:5000/api/v1/resources/user?username=user_pavan'
```

5.2 Authenticate user

You can authenticate user using 2 ways:

1. Using the POST request by passing only uesrname and password in json.

```
curl -X POST -v http://127.0.0.1:5000/api/v1/resources/user -d '{"username": "user_pavan",  
"password": "12ds"}'
```

***NOTE: If other fields are passed in json, then it will be treated as create new user scenario.**

2. Using the GET request by passing the uesrname and password in the url parameters (not advised because of security concerns):

```
curl -v 'http://127.0.0.1:5000/api/v1/resources/user?username=user_pavan&password=12ds'
```

5.3 Create User

- To create a new User profile use the following curl command (POST method):

```
curl -X POST -v http://127.0.0.1:5000/api/v1/resources/user -d '{"username": "joker12",  
"display_name": "Joker", "password": "serious", "homepage_url": "joker.com", "email":  
"joker@joker.com"}'
```

5.4 Update User Password

- **Change User's Password use the following curl command:**

PATCH: curl -X PATCH -v <http://127.0.0.1:5000/api/v1/resources/user?username=joker12> -d '{ "password": "serious123" }'

5.5 Delete User

- **To delete a User's use the following curl command:**

DELETE: curl -X DELETE -v 'http://127.0.0.1:5000/api/v1/resources/user?username=user_pavan'

6) Instructions to use the TRACKS Micro-service:

6.1 Get Track:

- To retrieve a particular Track (GET method) by passing the track_url, use the following command:

```
curl -v "http://127.0.0.1:5200/api/v1/resources/tracks?  
track_uuid=275fc399a955403dacb158cdb6f273b5"
```

- To retrieve all Track's (GET method) use the following curl command:

```
curl -v 'http://127.0.0.1:5200/api/v1/resources/tracks'
```

6.2 Create Track

- To create a new Track (POST method) use the following command:

```
curl -X POST -v http://127.0.0.1:5200/api/v1/resources/tracks -d '{"track_title": "tango", "album_title":  
"Joker", "artist": "brad", "length": "121212", "track_url":  
"http://localhost:8000/media/ThanBefore.mp3", "album_art_url": "tuffy.com"}'
```

***NOTE:** Add all the field. If no value then leave the double quotes empty. Ex. "".

6.3 Edit Tracks

- To edit a Track (PUT method) use the following command:

```
curl -X PUT -v http://127.0.0.1:5200/api/v1/resources/tracks?track_uuid=  
275fc399a955403dacb158cdb6f273b5 -d '{"track_title": "JustMe", "album_title": "Joker", "artist":  
"rf", "length": "12:60:12", "track_url": "http://localhost:8000/media/Newsong.mp3", "album_art_url":  
"LetsSee.com", "track_uuid"="275fc399a955403dacb158cdb6f273b5"}
```


*NOTE: Add all the field. If no value then leave the double quotes empty. Ex. "".

6.4 Delete Track

- **To delete a Track (DELETE method) use the following command:**

```
curl -X DELETE -v 'http://127.0.0.1:5200/api/v1/resources/tracks?  
track_uuid=275fc399a955403dacb158cdb6f273b5'
```

7) Instructions to use the DESCRIPTIONS Micro-service:

7.1 GET a User's Description of a track :

```
curl -v "http://127.0.0.1:5100/api/v1/resources/descriptions?  
username=user_anthony&track_uuid=275fc399a955403dadb158cdb6f273b5"
```

7.2 Create a User's Description of a track

```
curl -X POST -v http://127.0.0.1:5100/api/v1/resources/descriptions -d '{"username":"user_anthony",  
"track_uuid":"275fc399a955403dadb158cdb6f273b5","description":"Looks good"}'
```

***NOTE:** Add all the field. If no value then leave the double quotes empty. Ex. "".

8) Instructions to use the Playlist Micro-service:

8.1 GET a Playlist:

```
curl -v 'http://127.0.0.1:5300/api/v1/resources/playlist?playlist_title=All&username=user_priyanka'
```

8.2 GET method all Playlists:

```
curl -v 'http://127.0.0.1:5300/api/v1/resources/playlist'
```

8.3 GET all Playlists created by a particular User:

```
curl -v 'http://127.0.0.1:5300/api/v1/resources/playlist?username=user_anthony'
```

8.4 Create a new Playlist:

```
curl -X POST -v http://127.0.0.1:5300/api/v1/resources/playlist -d '{ "all_tracks": [  
  {  
    "track_uuid": "275fc399a955403dacb158cdb6f273b5"  
  },  
  {  
    "track_uuid": "ac3d3e62d61140138bc922d90623d5db"  
  }  
], "playlist_title": "NewPlaylistIsThis", "username": "user_priyanka", "description": "This playlist  
contains some of my songs"}
```

*NOTE: Track's microservice url can also be directly be placed in the location of track_uuid.

Ex. Instead of this: "track_uuid": "275fc399-a955-403d-acb1-58cdb6f273b5",

we can also pass:

"track_uuid": "http://127.0.0.1:5200/api/v1/resources/tracks?track_uuid=275fc399-a955-403d-acb1-58cdb6f273b5"

ii) Add all the field. If no value then leave the double quotes empty. Ex. "".

8.5 Delete a Playlist :

```
curl -X DELETE -v 'http://127.0.0.1:5300/api/v1/resources/playlist?username=user_priyanka&playlist_title=NewPlaylistIsThis'
```

9) XSPF Generator Microservice:

```
curl -v 'http://127.0.0.1:5400/api/v1/resources/music.xspf?playlist\_title=All&username=user\_priyanka'
```

10) Run Tavern files

```
py.test test_tracks.tavern.yaml -v  
py.test test_playlist.tavern.yaml -v  
py.test test_user.tavern.yaml -v  
py.test test_descriptions.tavern.yaml -v
```