Precondition

To run the demo successfully, you need to install

1. docker
2. docker-composer

local port used: 9001 and 8081, you need to make sure these 2 ports are not occupied.

The docker images are pushed at docker hub, so you need to make sure your environment could access ducker repository.

Version reference:

docker-compose version 1.24.1 for windows

docker 2.1.0.4

How to run the Demo

Put ‘docker-compose.yml’ to local directory, you can find it in the project.

here is an example:

/home/xubin/docker-compose.yml. go to /home/xubin/ and then run the command

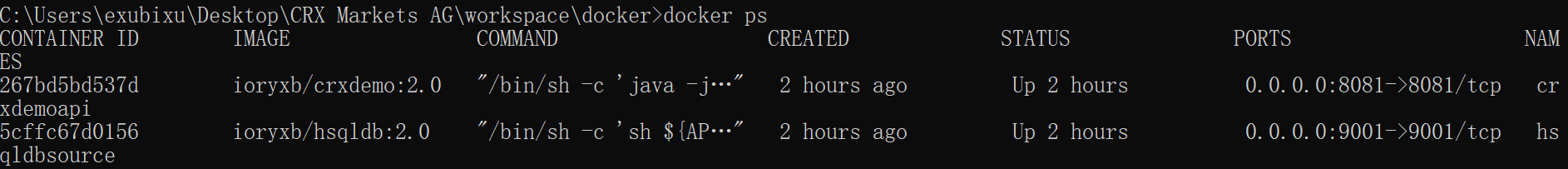
e.g.

cd /home/xubin

docker-compose up -d

images will pull from docker.io.

you will see 2 containers are up and running.



Manager the container

If you must manually manager the container, use docker command. The alia of DB container must be aliashsqldbsource and the port of DB must be 9001, they are hard code in ioryxb/crxdemo:2.0

Image.

e.g.:

docker run -d -p 9001:9001 --name hsqldbsource ioryxb/hsqldb:1.0

docker run -d -p 8081:8081 --link hsqldbsource:aliashsqldbsource ioryxb/crxdemo:2.0

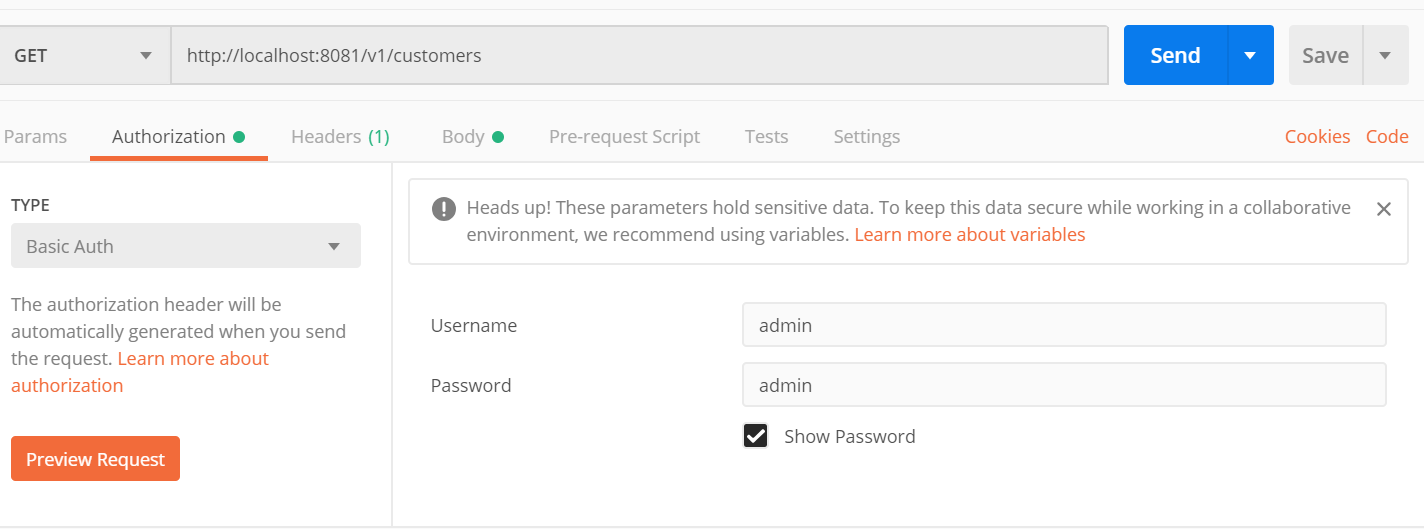
REST APIs

The REST API are secured with basic auth, you can use

username: admin

password: admin

An example from postman:



**List all the Customers**

Method: GET

URI: <http://localhost:8081/v1/customers>

Body: None

**Create a new Customer**

Method: POST

URI: <http://localhost:8081/v1/customers>

Body:

{

"name": "jinjin",

"address": "doncast east",

"tickets": []

}

**Edit existing Customer**

Method: PUT

URI: [http://localhost:8081/v1/customers/<cid](http://localhost:8081/v1/customers/%3ccid)>

e.g. <http://localhost:8081/v1/customers/1> to edit customer with cid 1.

Body:

{

"name": "jinjin2tai",

"address": "doncast east",

"tickets": []

}

**List all the Tickets for a Customer**

Method: GET

URI: [http://localhost:8081/v1/customer/<cid>/tickets](http://localhost:8081/v1/customer/%3ccid%3e/tickets)

e.g. <http://localhost:8081/v1/customer/1/tickets> to list all tickets of customer with cid 1.

Body: None

**Assign a new Ticket to a Customer**

Method: POST

URI: [http://localhost:8081/v1/customer/<cid>/tickets](http://localhost:8081/v1/customer/%3ccid%3e/tickets)

e.g. <http://localhost:8081/v1/customer/1/tickets> to assign a new ticket to customer with cid 1.

Body:

{

"price":139,

"state":"Generated",

"date":"20191110",

"validFrom":"20191020",

"validTo":"20191120"

}

**Confirm the Ticket**

Method: PATCH

URI: [http://localhost:8081/v1/customer/<cid>/tickets/<tid](http://localhost:8081/v1/customer/%3ccid%3e/tickets/%3ctid)>

e.g. <http://localhost:8081/v1/customer/2/tickets/5> to change ticket, which is with tid 5 of customer with cid 2, status to “Confirmed”

PS: this API will change ticket status to Confirmed.

Body: None

**Cancel the Ticket**

Method: DELETE

URI: [http://localhost:8081/v1/customer/<cid>/tickets/<tid](http://localhost:8081/v1/customer/%3ccid%3e/tickets/%3ctid)>

e.g. <http://localhost:8081/v1/customer/2/tickets/5> to unassign ticket, which is with tid 5 from customer with cid 2.

PS: "date" is the departure day of a ticket. API will report error when we try to cancel a ticket which is less than 2 days to the departure day.

Body: None

Project structure

baseimg: you will find the docker file which is used for building base image here, a ubuntu with JDK.

composer: you will find the docker-compose.yml file here.

docker: you will find the docker file which is used for building hsqldb image.

ticketoffice: the project and all source code. Image of crxdemo:2.0 are built by maven plugin.

