JHipster, Ruby Micro Services Setup in Docker

1. Setup JHipster, Ruby and Docker
2. Create Micro Services
3. Create Micro Services gateway
4. Create Ruby application with standalone database
5. Generate ruby swagger code
6. Import Ruby project into Docker container
7. Connect Ruby application to micro services through Swagger
8. Run Docker & Micro Services

Setup JHipster, Ruby and Docker

## Setup JHipster

Setting up JHipster can be done in any one of the following ways.

## Local installation with Yarn

1. Setup JHipster by following instructions mentioned [here](http://www.jhipster.tech/installation/). Follow “Local installation with Yarn” process
2. Install Java 8 from [the Oracle website](http://www.oracle.com/technetwork/java/javase/downloads/index.html).
3. Install Node.js from [the Node.js website](http://nodejs.org/) (prefer an LTS version)
4. Install Yarn from [the Yarn website](https://yarnpkg.com/en/docs/install)
5. Install JHipster: yarn global add generator-jhipster

## Local installation with NPM (alternative to Yarn)

This is the same procedure as using Yarn, with two differences:

1. Instead of installing Yarn in step 3, update NPM: npm install -g npm
2. Use npm install -g instead of yarn global add, for example:
   * To install Yeoman, type: npm install -g yo
   * To install JHipster, type: npm install -g generator-jhipster

**Docker Installation**

1. Install docker from <https://docs.docker.com/engine/installation/> and select community edition (if enterprise version not available) for development. Use Enterprise for production.

**Ruby Installation**

1. Install ruby from <http://railsapps.github.io/installrubyonrails-mac.html>. This website explains the process from scratch along with dependencies

Create Micro Services

Create home directory in user home folder (ex. jhipsterapps) before proceeding

1. <http://www.jhipster.tech/creating-an-app/> explains step by step process to create micro service application.
2. Once application generated use ./mvnw package -Pprod dockerfile:build to build image on Linux/MacOS/Windows PowerShell to compile application
3. Then move to src/main/docker/ directory and type docker-compose -f app.yml up
4. You can use same process to create multiple services. Make sure to create new directory for each micro service

Create Micro Service Gateway

1. <http://www.jhipster.tech/creating-an-app/> explains step by step process to create micro service gateway.
2. Once application generated use ./mvnw package -Pprod dockerfile:build to build image on Linux/MacOS/Windows PowerShell to compile application
3. Then move to src/main/docker/ directory and type docker-compose -f app.yml up
4. To start Gateway UI, open separate terminal and type command **yarn start**

## Generating a custom Docker-Compose configuration for multiple applications

We can use the specific docker-compose sub-generator, which will generate a global Docker Compose configuration for all selected applications. This will allow you to deploy and scale your complete architecture with one command. To use the docker-compose subgenerator:

* You need to have all your monolith(s), gateway(s) and microservices in the same directory.
* Create another directory, for example mkdir docker-compose.
* Go into that directory: cd docker-compose.
* Run the sub-generator: jhipster docker-compose
* To start Gateway UI, open separate terminal and type command **yarn start**

This will generate a global Docker Compose configuration, type docker-compose up to run it, and have all your services running at once.

Create Ruby Project

1. Generate sample rails app using <https://www.railstutorial.org/book/toy_app> tutorial.

Generate Ruby Swagger Code

1. To install Swagger-codegen , run brew install swagger-codegen
2. Go to microservice gateway on <http://localhost:9000/#/> then Administration🡪API-> Select gateway.
3. The path should look like <http://localhost:9000/v2/api-docs>
4. Generate code using swagger-codegen generate -i <http://localhost:9000/v2/api-docs> -l ruby -o <your target directory>
5. Once it’s generated, rename the target directory to ‘swagger\_api’ and copy this into root directory of Ruby application generated in last page
6. Generate code for all other micro services in ’swagger\_api’ folder.
7. Add following code to swagger\_client.rb if not exists. This will allow swagger\_cleint to access all the libraries inside.

*# Common files  
require* **'swagger\_client/api\_client'***require* **'swagger\_client/api\_error'***require* **'swagger\_client/version'***require* **'swagger\_client/configuration'***# Models  
require* **'swagger\_client/models/login\_vm'***require* **'swagger\_client/models/jwt\_token'***require* **'swagger\_client/models/user'***require* **'swagger\_client/models/projects'***require* **'swagger\_client/models/key\_and\_password\_vm'***require* **'swagger\_client/models/managed\_user\_vm'***require* **'swagger\_client/models/profile\_info\_vm'***require* **'swagger\_client/models/route\_vm'***require* **'swagger\_client/models/service\_instance'***require* **'swagger\_client/models/uri'***require* **'swagger\_client/models/user\_dto'***# APIs  
require* **'swagger\_client/api/profileinforesource\_api'***require* **'swagger\_client/api/projectsresource\_api'***require* **'swagger\_client/api/userjwtcontroller\_api'***require* **'swagger\_client/api/accountresource\_api'***require* **'swagger\_client/api/gatewayresource\_api'***require* **'swagger\_client/api/userresource\_api'**

1. If Docker cannot find above files, add workspace path manually to **app/helpers/railsapp\_swagger\_client.rb**file process will be explained later
2. Execute following commands and build gem out of swagger\_client

To build the Ruby code into a gem:

$gem build swagger\_client.gemspec

Then either install the gem locally:

$gem install ./swagger\_client-1.0.0.gem

Connect Ruby & Micro Services

1. Stop MySql service on host system if you get port issue
2. Go to Ruby root directory and create file called ‘Dockerfile’ with no extension
3. Paste the following code

FROM ruby:2.3.4

ENV LANG C.UTF-8

RUN apt-get update && \

apt-get install -y nodejs \

vim \

mysql-client \

--no-install-recommends && \

rm -rf /var/lib/apt/lists/\*

#Cache bundle install

WORKDIR /tmp

ADD ./Gemfile Gemfile

ADD ./Gemfile.lock Gemfile.lock

ADD swagger\_api /swagger

RUN bundle install

ENV APP\_ROOT /workspace

RUN mkdir -p $APP\_ROOT

WORKDIR $APP\_ROOT

COPY . $APP\_ROOT

EXPOSE 3000

CMD ["rails", "server"]

1. Now go to docker-compose directory under jhipsterapps or whatever the name of your microservices home directory (created in microservices gateway step) and open docker-compose.yml file, add following code under **services:** line (after line 2).

rails-mysql:

image: mysql:5.7.20

environment:

- MYSQL\_USER=root

- MYSQL\_ALLOW\_EMPTY\_PASSWORD=yes

- MYSQL\_DATABASE=expenses

command: mysqld --lower\_case\_table\_names=1 --skip-ssl --character\_set\_server=utf8

--explicit\_defaults\_for\_timestamp

rails:

build: ../railsapp

environment:

RAILS\_ENV: development

ports:

- '3000:3000'

volumes:

- ../railsapp:/workspace

stdin\_open: true

tty: true

1. Replace the build path with your rails application path. My application path “../railsapp” i.e railsapp present in parent directory. If it is in parent parent directory use ../../railsapp
2. Change the config/database.yml file as follows (make sure to rename database and usernames)

**default:** &default  
 **adapter:** mysql2  
 **encoding:** utf8  
 **host:** rails-mysql  
 *#host: localhost* **pool:** 5  
  
**development:  
 <<:** \*default  
 **username:** root  
 *#password: jadda007* **database:** railsapp\_development

1. Create new model class called projects\_resource.rb under app/models with following content

*#require* ***'active\_resource'*class *ProjectsResource*** < ***ActiveResource***::***Base*** headers[**"Authorization"**] = ***RailsappSwaggerClient***.instance.authenticate  
 **self**.site = **"http://dockercompose\_gateway\_timesheets-app\_1:8080/timesheets/api/"  
 self**.include\_format\_in\_path = **false  
end**

1. Make sure to replace self.site value based on your API gateway.

**self**.site = **"http://<running docker gateway instance name>:<gateway port>/<microservice name>/api/"**

1. Modify projects.rb model class to below. This will disconnect Rails model from local rails database and gets data from API specified in above step

*#class Project < ApplicationRecord***class *Projects*** < ***ProjectsResource*end**

1. Repeat steps 7,8,9 if you have more model classes and micro services. Always remember you can use same projects\_resource.rb or filename.rb if the model class you are trying to access available in same micro service. Do not create different resource classes for model classes in one micro service.
2. Create another file railsapp\_swagger\_client.rb under “app/helpers/”, This file helps to communicate Ruby to micro services. Make changes to configuration based your application

*require* **'singleton'***require* **'swagger\_client'***require* **'uri'***# Common files  
require* **'/workspace/swagger\_api/lib/swagger\_client/api\_client'***require* **'/workspace/swagger\_api/lib/swagger\_client/api\_error'***require* **'/workspace/swagger\_api/lib/swagger\_client/version'***require* **'/workspace/swagger\_api/lib/swagger\_client/configuration'***# Models  
require* **'/workspace/swagger\_api/lib/swagger\_client/models/login\_vm'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/jwt\_token'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/user'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/projects'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/key\_and\_password\_vm'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/managed\_user\_vm'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/profile\_info\_vm'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/route\_vm'***require* **'/workspace/swagger\_api/lib/swagger\_client/models/service\_instance'***#****require '/workspace/swagger\_api/lib/swagger\_client/models/uri'****require* **'/workspace/swagger\_api/lib/swagger\_client/models/user\_dto'***#****API****'S  
require* **'/workspace/swagger\_api/lib/swagger\_client/api/userjwtcontroller\_api'***require* **'/workspace/swagger\_api/lib/swagger\_client/api/profileinforesource\_api'***require* **'/workspace/swagger\_api/lib/swagger\_client/api/projectsresource\_api'***require* **'/workspace/swagger\_api/lib/swagger\_client/api/accountresource\_api'***require* **'/workspace/swagger\_api/lib/swagger\_client/api/gatewayresource\_api'***require* **'/workspace/swagger\_api/lib/swagger\_client/api/userresource\_api'  
  
  
  
  
class *RailsappSwaggerClient*** *include* ***Singleton****#:8080/api/authenticate* **def** *initialize* **@authorized** = **false  
 @authorization** = **''  
 *SwaggerClient***.configure **do** |*c*|  
 *c*.scheme = **'http'** *c*.host = **'dockercompose\_gateway\_timesheets-app\_1:8080'** *c*.base\_path = **''  
 end  
  
 end   
  
 def** *authenticate* **if** !**@authorized then** authorize  
 **@authorized** = **true  
 end   
  
 return @authorization  
 end  
  
 def** *reauthorize* authorize  
 **end** *#****sample*** *to get clients* **def** *getClients* authenticate  
 *apiclient* = ***SwaggerClient***::***ApiClient***.*new   
 apiclient*.config.base\_path = **'/contracts/'** *apiclient*.default\_headers = *apiclient*.default\_headers.merge({**"Authorization"** *=>* **@authorization**})  
 *api\_instance* = ***SwaggerClient***::***ClientresourceApi***.*new*(*apiclient*)  
  
 **begin** *#****getAllClients*** *result* = *api\_instance*.get\_all\_clients\_using\_get  
 p *result* **rescue *SwaggerClient***::***ApiError*** *=> e* puts **"Exception when calling ClientresourceApi->get\_all\_clients\_using\_get:** #{*e*}**"  
 end  
 end** *#****sample*** *to get clients* **def** *getProjects* authenticate  
 *apiclient* = ***SwaggerClient***::***ApiClient***.*new   
 apiclient*.config.base\_path = **'/projects/'** *apiclient*.default\_headers = *apiclient*.default\_headers.merge({**"Authorization"** *=>* **@authorization**})  
 *api\_instance* = ***SwaggerClient***::***ClientresourceApi***.*new*(*apiclient*)  
  
 **begin** *#****getAllClients*** *result* = *api\_instance*.get\_all\_clients\_using\_get  
 p *result* **rescue *SwaggerClient***::***ApiError*** *=> e* puts **"Exception when calling ClientresourceApi->get\_all\_clients\_using\_get:** #{*e*}**"  
 end  
 end***private* **def** *authorize  
 #****binding****.****pry*** *api\_instance* = ***SwaggerClient***::***UserjwtcontrollerApi***.*new  
 login\_vm* = ***SwaggerClient***::***LoginVM***.*new* **password**: **'admin'** , **rememberMe**: **true**, **username**: **'admin'** *# LoginVM | loginVM* **begin** *#****authorize*** *result*, *header* = *api\_instance*.authorize\_using\_post(*login\_vm*)  
 **@authorization** = *header*[**"Authorization"**]  
 *#****@authorization****=* ***"Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImF1dGgiOiJST0xFX0FETUlOLFJPTEVfVVNFUiIsImV4cCI6MTUxNzUxNTY0Mn0.s29nK9m8\_UGTkbVy-ag\_N\_64tebOBq5ykNeQhtllFxnpexli269v9qUPxSoc1Ofm\_w9PLkJKQu56R9z0Q3TFmA"****;* **@authorized** = **true  
  
 rescue *SwaggerClient***::***ApiError*** *=> e* puts **"Exception when calling UserjwtcontrollerApi->authorize\_using\_post:** #{*e*}**"  
 end  
 end  
  
end**

1. In above code, under initialize method change c.host value to your gateway and change credentials in following line

*login\_vm* = ***SwaggerClient***::***LoginVM***.*new* **password**: **'admin'** , **rememberMe**: **true**, **username**: **'admin'**

1. Methods getClients and getProjects are not required, you can delete/comment them
2. Finally add this to the Rails Gemfile if not added

gem 'swagger\_client', '~> 1.0.0'

Run Docker & Micro Services

1. It is good to rails and micro services in different terminals
2. Navigate to docker-compose directory and execute following command

$docker-compose up rails

1. Open new terminal type following command, this will start rails mysql (although we don’t use it)

$docker-compose up rails-mysql

1. Open new terminal type following command, this will start all the micro services including gateway. It may take 2-5 minutes to start all micro services.

$docker-compose up –d

1. “docker ps” command shows all the running process
2. Open new terminal type following commands to create rails database and migrate in docker

$docker exec -it <Docker Contianer Name> rake db:create

$docker exec -it <Docker Contianer Name> rake db:migrate

1. To access rails application running docker use following command

# connect to rails docker

docker exec -it <Docker Contianer Name> bash

1. Navigate to localhost:3000/projects to see all projects
2. If any issues occurred refer to ‘Known Issues’ document