**Recipe application is an java based rest api to create, get, update and delete recipes:**

<https://github.com/babumyb/recipe-app.git>

docker pull 627909/recipe-app:0.0.1

|  |  |
| --- | --- |
| **Technology/Standard/tool** | **Purpose** |
| Open jdk Java 8.0 |  |
| Spring Boot 2.6.5 |  |
| Swagger 3.0 | For documenting the API’s |
| H2 | In memory database for persisting data |
| Docker | To containerizing the application |
| Okta API | For OpenID protocol authentication and Oauth authorization. |
| Maven | Building the images |
| Git | Source control versioning |
| Tomcat 9.0 image | Web container |
| Docker hub | Image registry for application |

**How to run application ?**

1. Open the command terminal and run the command

docker container run -p 8080:8080 -v C:/temp/logs:/usr/local/tomcat/logs --env CATALINA\_OPTS="-Dlog.level=error -Dspring.datasource.url=jdbc:h2:mem:test -Dspring.datasource.username=sa -Dspring.datasource.password= -DyourOktaDomain=dev-12462395.okta.com -DclientId=<ID> -DclientSecret=<SECRET>" 627909/recipe-app:0.0.1

Note that this is **not preferred** method to pass the credentials or secret. In production this could be done using config map in kubernetes.

What this command does ?

* It pulls the image **627909/recipe-app:0.0.1** from docker hub if not present (make sure you have docker engine running on your system)
* **-v C:/temp/logs:/usr/local/tomcat/logs** creates volume in your host if not present and mounts the logs from container to host. Make sure you have following directory in your system. On production logs can used with elk stack solution like **splunk or sumo logic** to get statistical insights.
* Passes **--env CATALINA\_OPTS** as environment variables so that parameters are available during application startup.
* **-p 8080:8080** binds port 8080 from container to 8080 port of the host. Make sure 8080 port on your system is available for allocation.
* **run** creates the instance of your image which starts the application

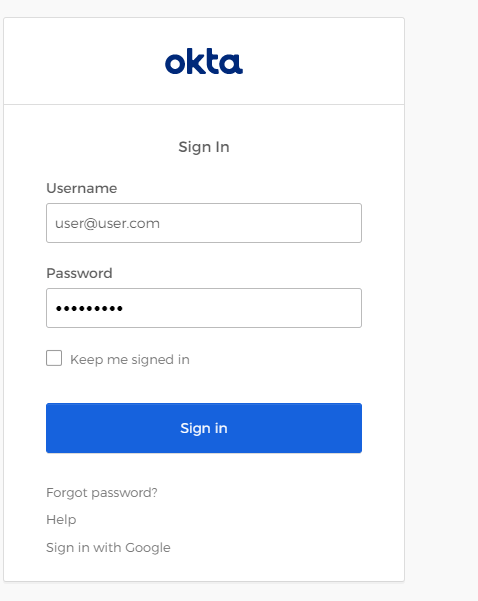
1. After successful container startup, open go to link [**http://localhost:8080/recipe-app/swagger-ui/index.html**](http://localhost:8080/recipe-app/swagger-ui/index.html)**.** Here you will see all the endpoints of the api and related documentation.
2. API’s are secured using Oauth authorization protocol and oidc authentication protocol and Okta Is chosen for both. An access token would be needed to access the API’, so on your browser go to,

<https://dev-12462395.okta.com/oauth2/default/v1/authorize?client_id=0oa4du3deqKuHngrZ5d7&redirect_uri=https%3A%2F%2Foidcdebugger.com%2Fdebug&scope=openid&response_type=token&response_mode=form_post&state=test&nonce=6di5f61nrco>

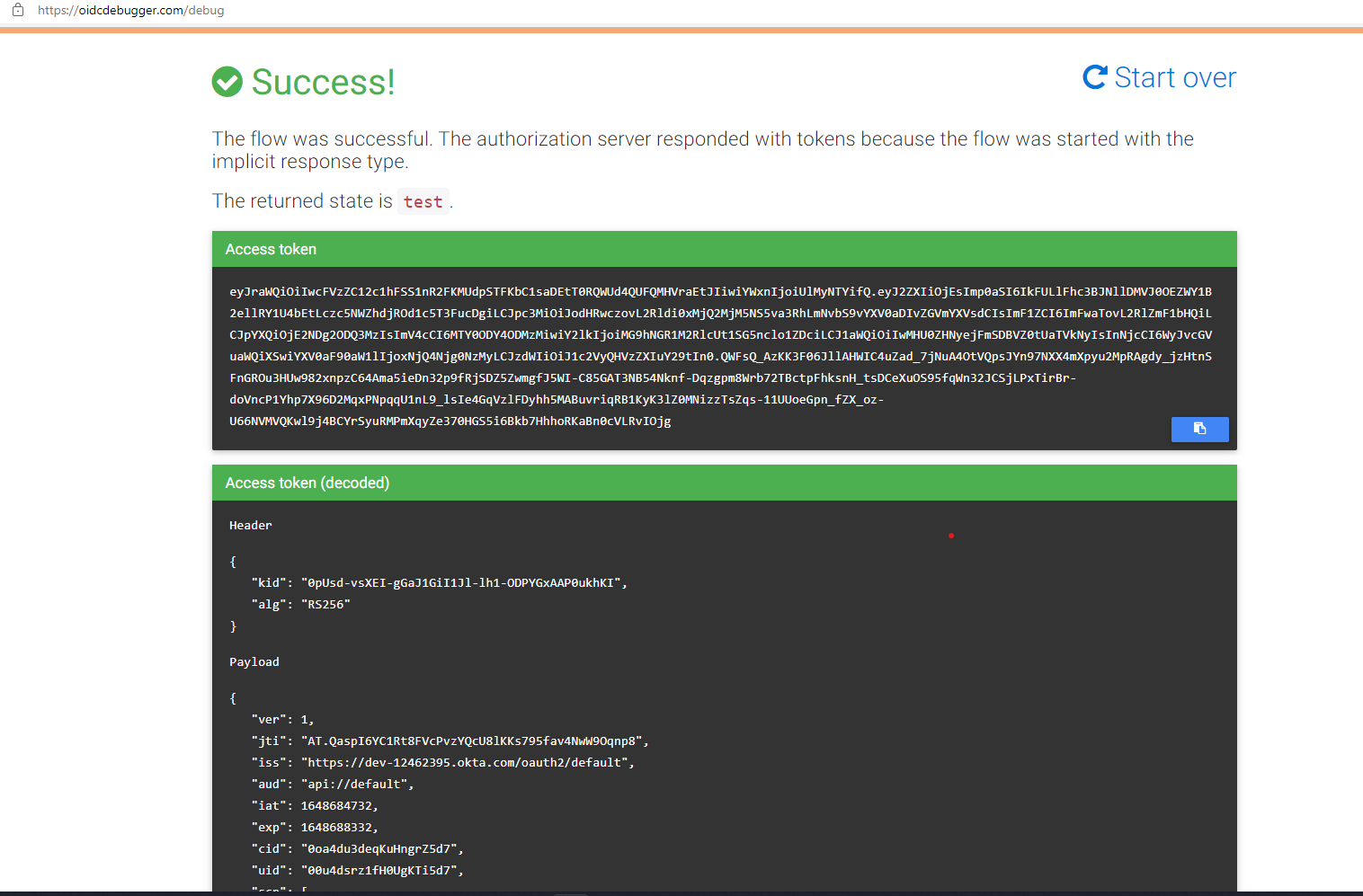
You would be redirected to Okta login page as below,

Use user name : [xxxx](mailto:user@user.com)

Password : xxxxx

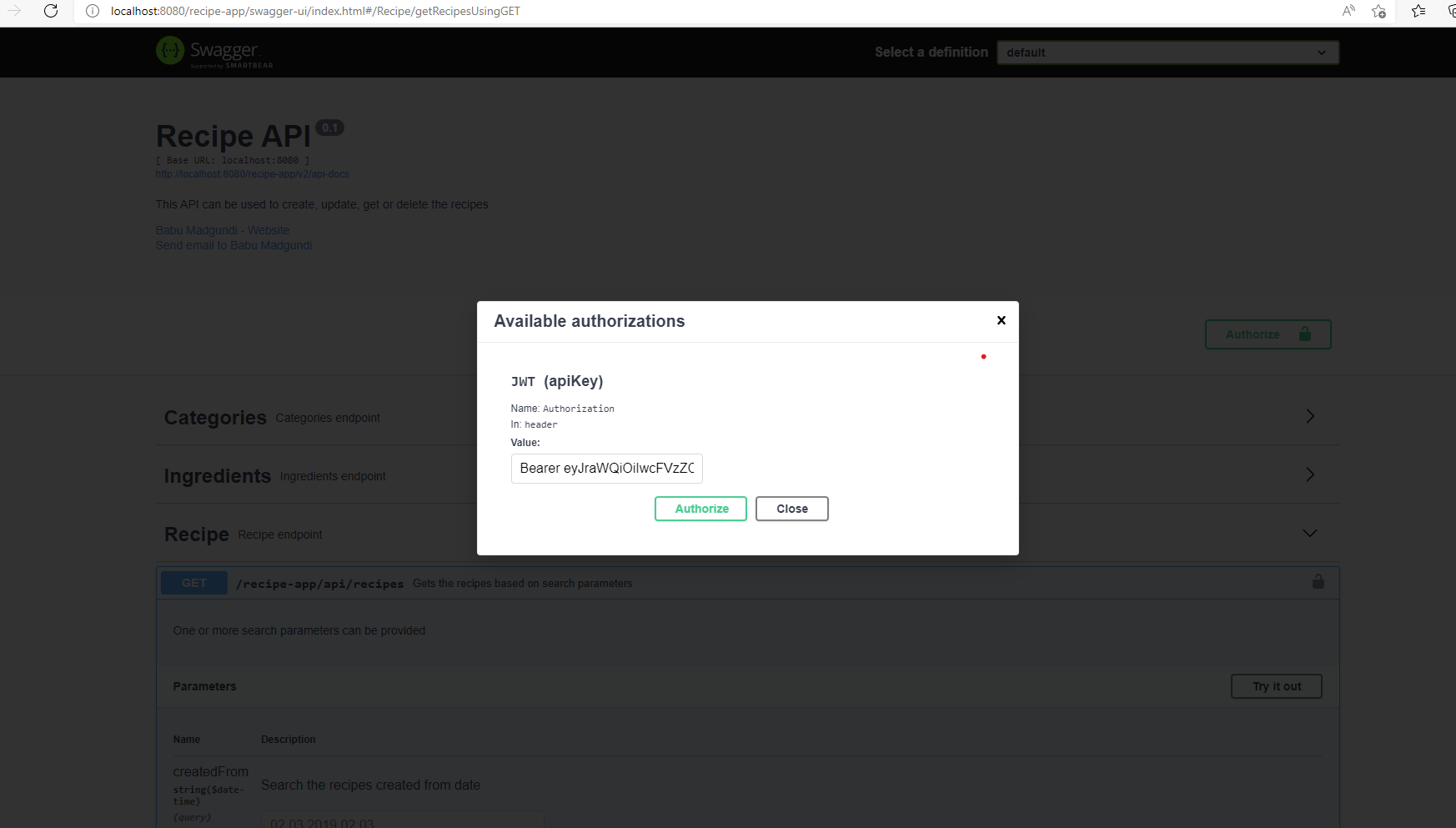
****

After successful login, you would see page,

****

Copy access token from first box.

1. Go back to you application and paste the authorization code with prefix “Bearer” by clicking on Authorize button as shown below,



1. That’s it, the api is ready for use, you would find all the related documentation about paths, request models.

**Design considerations :**

1 . Spring boot is chosen because of its superior dependency management where the related dependencies are neatly managed. Although it makes life easier, we can quickly run into loads of jars in our class path which we may never use and also making maintenance of check for vulnerabilities time consuming. However in this case where application is build like a microservice the benefits of speedier development, support for embedded servers etc. far outweigh the disadvantages.

2. Swagger is chosen for documentation because it creates automated documentation based on the annotations.

3. Third party security provider Okta provides out of box solution to secure apps based on Oauth and oidc protocol. It is preferred way to separate authentication and authorization from application logic which helps in horizontal scaling.

4. Docker is used for containerizing application. Application image is built from tomcat 9.0. The idea behind choosing docker is how simple it is to scale the application when used with Kubernetes.

5. Maven is used because managing the binaries and images can be automated to large extent. io.fabric plugin is used to build the image, it can easily be extended to push and run the containers for CI and CD pipelines.

6. H2 is used for simplicity purpose. Should not be used on production.

7. Do note that a service layer can be added between persistence and presentation layer to implement business logic. I choose not to because application is very simple and service layer does not have any injected beans other that data beans. Once the application starts growing we can implement this layer.

8. Open source library like object mapper could be used instead of converter beans once application starts growing.

**Other notes:**

1. Run mvn clean install command to build the image after checking out the project.
2. If for some reasons you are not able to run the docker images, you could it as Java application. Ensure okta related properties are updated in application.properties. See the okta URL above for property details.