

GitHub Repo Details

https://github.com/Syed-UsmanAli/OAuth2_Authorization_Server

Final code is placed in the default “master” branch and not on “main” branch.

The screenshot shows the GitHub repository page for 'Syed-UsmanAli / OAuth2_Authorization_Server'. The repository has 11 commits, 2 branches, and 0 tags. The master branch contains files like .mvn/wrapper, src, .gitignore, Dockerfile, README.md, and pom.xml. The README.md file contains the text 'OAuth2'. The repository has 1 star and 0 forks. It includes sections for About, Releases, Packages, and Languages, with Java being the primary language at 98.6%.

IDE

Spring Tool Suite 4 IDE was used and the project has been developed as a Spring Boot Project. Maven is the build tool. The project has also been Dockerized and image has been uploaded on Docker Hub.

Jenkins was used for CI .Pipeline was configured to trigger maven build on each commit to GIT. It would then build and boot up the Spring Boot Application, followed by executing the Unit test cases placed in the project.

Screenshot of Project Folder Structure inside SpringBoot App

The screenshot shows the IntelliJ IDEA interface with the following details:

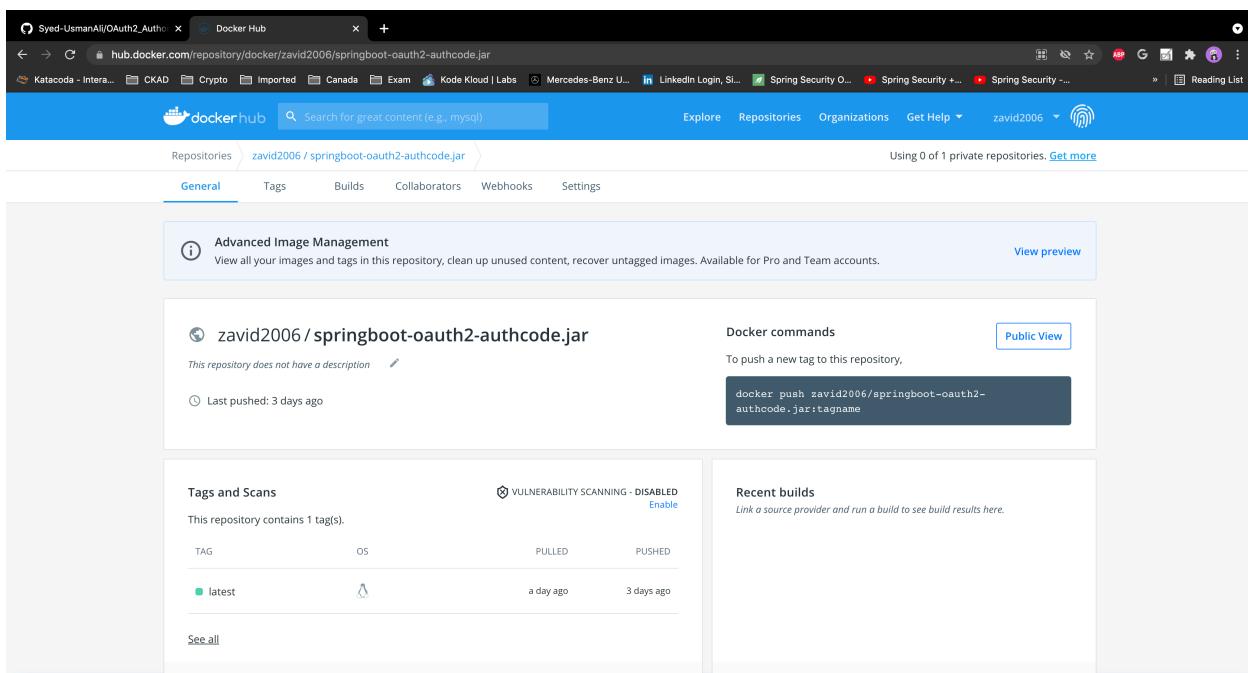
- Project Explorer:** Shows the project structure under "OAuth2 (boot) [OAuth2 master]". It includes:
 - src/main/java:
 - com.example.demo
 - OAuth2Application.java
 - OAuth2ApplicationTests.java
 - Dockerfile
 - AuthorizationServerConfig.java
 - UserManageConfig.java
 - src/test/java
 - Maven Dependencies
 - target/generated-sources/annotations
 - target/generated-test-sources/test-annotations
 - src:
 - main
 - test
 - java
 - example
 - Demo
 - target
 - Dockerfile
 - HELP.md
 - pom.xml
 - README.md
- Code Editor:** Displays the `OAuth2Application.java` file with the following content:

```
1 package com.example.demo;
2
3 import org.springframework.boot.SpringApplication;
4
5
6 /**
7 * This is a demo application to show case a lightweight OAuth 2.0 authorization server.
8 * Following functionality has been implemented
9 *
10 * - Authorization endpoint to get a client's authorization request and validate it.
11 * - Built in Spring Boot Login Form to authenticate user
12 * - Obtain an authorization decision from the resource owner after a successful authentication.
13 * - When the access request is granted, redirect the user-agent to the provided
14 *   client redirect URL and provide an authorization code to the client application.
15 * - Implement a token endpoint. Upon receiving an access token request,
16 *   validate client's credentials and the request itself.
17 * - When the access token request is valid and authorized,
18 *   return an access token to the client application.
19 */
20 public class OAuth2Application {
21
22     /**
23      * The main method.
24      *
25      * @param args the arguments
26      */
27     public static void main(String[] args) {
28         SpringApplication.run(OAuth2Application.class, args);
29     }
30 }
31
32 /**
33  * @param args the arguments
34 */
35 public static void main(String[] args) {
36     SpringApplication.run(OAuth2Application.class, args);
37 }
38
39
40 /**
41  * @param args the arguments
42 */
43 }
```

- Bottom Bar:** Shows tabs for Console, Problems, and Debug Shell, with a message indicating "No consoles to display at this time".

Screenshot of Uploaded Docker Hub Image

<https://hub.docker.com/repository/docker/zavid2006/springboot-oauth2-authcode.jar>



How likely are you to recommend Docker Hub to another developer?

Not at all likely 0 1 2 3 4 5 6 7 8 9 10 Extremely likely

powered by InMoment

Screenshot of Local Jenkins

The screenshot shows the Jenkins dashboard at localhost:8080. On the left, a sidebar lists various Jenkins management options like 'New Item', 'Build History', and 'Check File Fingerprint'. The main area displays a table of jobs. One job, 'Jenkins-Docker', is listed with the following details:

S	W	Name	Last Success	Last Failure	Last Duration
✓	cloud	Jenkins-Docker	17 min - #13	12 hr - #9 zavid2006/springboot-oauth2-authcode.jar	14 sec

Below the table, there are three links for Atom feeds: 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'. At the bottom right of the dashboard, it says 'REST API Jenkins 2.289.1'.

Jenkins Pipeline was configured with

- 1) GitHub Repo to trigger maven build on target “install” on every commit to “master” branch

The screenshot shows the configuration page for the 'Jenkins-Docker' job. The top navigation bar includes tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is selected.

General Tab Configuration:

- Source Code Management:** Git is selected.
- Repositories:** A repository URL is set to https://github.com/Syed-UsmanAli/OAuth2_Flow.git.
- Credentials:** An administrator credential named 'admin/***** (Jenkins_Admin)' is assigned.

Build Tab Configuration:

- Build Triggers:** The 'GitHub project' trigger is selected, with the 'Project url' field set to https://github.com/Syed-UsmanAli/OAuth2_Flow/.
- Post-build Actions:** Several optional actions are listed, such as 'Commit agent's Docker container', 'Run the build inside Docker containers', 'Define a Docker template', and 'Discard old builds'. The 'GitHub project' action is checked.

Syed-UsmanAli/Docker2_Auth... | Docker Hub | Jenkins-Docker Config [Jenkins] | +

localhost:8080/job/Jenkins-Docker/configure

Katacoda - Interact CKAD Crypto Imported Canada Exam Kode Kloud | Labs Mercedes-Benz U... LinkedIn Login, S... Spring Security O... Spring Security +... Spring Security -...

Dashboard > Jenkins-Docker >

General Source Code Management Build Triggers Build Environment Build Post-build Actions

(Auto)

Additional Behaviours

Add ▾

Build Triggers

Trigger builds remotely (e.g., from scripts) Build after other projects are built Build periodically GitHub hook trigger for GITScm polling Poll SCM Schedule

⚠️ Do you really mean "every minute" when you say "***"? Perhaps you meant "H * * * *" to poll once per hour**

Would last have run at Tuesday, July 6, 2021 at 10:08:59 AM Eastern Daylight Time; would next run at Tuesday, July 6, 2021 at 10:08:59 AM Eastern Daylight Time.

Ignore post-commit hooks

Build Environment

Delete workspace before build starts Use secret text(s) or file(s) Abort the build if it's stuck Add timestamps to the Console Output Inspect build log for published Gradle build scans With Ant

Build

Save Apply

Invoke top-level Maven targets

Build Environment

Delete workspace before build starts Use secret text(s) or file(s) Abort the build if it's stuck Add timestamps to the Console Output Inspect build log for published Gradle build scans With Ant

Build

Invoke top-level Maven targets

Maven Version: Maven3

Goals: install

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply

Syed-UsmanAli/Docker2_Auth... | Docker Hub | Jenkins-Docker Config [Jenkins] | +

localhost:8080/job/Jenkins-Docker/configure

Katacoda - Interact CKAD Crypto Imported Canada Exam Kode Kloud | Labs Mercedes-Benz U... LinkedIn Login, S... Spring Security O... Spring Security +... Spring Security -...

Dashboard > Jenkins-Docker >

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Build Environment

Delete workspace before build starts Use secret text(s) or file(s) Abort the build if it's stuck Add timestamps to the Console Output Inspect build log for published Gradle build scans With Ant

Build

Invoke top-level Maven targets

Maven Version: Maven3

Goals: install

Advanced...

Add build step ▾

Post-build Actions

Add post-build action ▾

Save Apply

One such sample “console output” from Jenkins

The screenshot shows two separate Jenkins job consoles. The top one is for a job named "Jenkins-Docker #13" and the bottom one is for "Jenkins-Docker #13". Both consoles display Maven build logs.

```

Started by an SCM change
Running as SYSTEM
Building in workspace /Users/syedusman/.jenkins/workspace/Jenkins-Docker
The recommended git tool is: NONE
using credential Jenkins_Admin
> git rev-parse --resolve-git-dir /Users/syedusman/.jenkins/workspace/Jenkins-Docker/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/Syed-UsmanAli/OAuth2_Flow.git # timeout=10
Fetching upstream changes from https://github.com/Syed-UsmanAli/OAuth2_Flow.git
> git -version # timeout=10
> git --version # 'git version 2.20.1 (Apple Git-117)'
using GIT_ASKPASS to see credentials Jenkins_Admin
> git fetch --tags --force --progress -- https://github.com/Syed-UsmanAli/OAuth2_Flow.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^(commit) # timeout=10
Checking out Revision eac254ba685be2cd99b103a4f16693d0224c2 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f eac254ba685be2cd99b103a4f16693d0224c2 # timeout=10
Commit message: "Modified test cases"
> git rev-list --no-walk ddd4388590f399cfbe137cf19f48f72b984 # timeout=10
[JENKINS-Docker] $ /Users/syedusman/.jenkins/tools/hudson.tasks.MavenInstallation/Maven3/bin/mvn install
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.security:OAuth2:jar:0.0.1-SNAPSHOT
[WARNING]   'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: org.springframework.boot:spring-boot-starter-test:jar > version (?) vs
2.5.1 @ line 51, column 15
[WARNING]   'build.plugins.plugin.(groupId:artifactId)' must be unique but found duplicate declaration of plugin org.springframework.boot:spring-boot-maven-plugin @
line 122, column 13
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----
[INFO] Building OAuth2 0.0.1-SNAPSHOT
[INFO] -----
[INFO] [ jar ] -----
[WARNING] The POM for com.sun.xml.bind:jaxb-osgi:jar:2.2.10 is invalid, transitive dependencies (if any) will not be available, enable debug logging for more
details
[INFO]
[INFO] --- maven-resources-plugin:3.1.0:resources (default-resources) @ OAuth2 ---

```



```

org.springframework.security.web.authentication.ui.DefaultLogoutPageGeneratingFilter@318864,
org.springframework.security.web.savedrequest.RequestCacheAwareFilter@85c8ab9de,
org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestWrapperFilter@57416e9,
org.springframework.security.web.authentication.AnonymousAuthenticationFilter@7d61468c, org.springframework.security.web.session.SessionManagementFilter@bfe47a8,
org.springframework.security.web.access.ExceptionTranslationFilter@8d77767, org.springframework.security.web.access.Intercept.FilterSecurityInterceptor@1a345298]
2021-07-06 09:50:59.625  WARN 4645 --- [           main] s.o.SpringCloudSecurityAutoConfiguration : All Spring Cloud Security modules and starters are deprecated.
They will be moved to individual projects in the next major release.
2021-07-06 09:50:59.646  INFO 4645 --- [           main] o.s.w.a.e.EndpointLinksResolver          : Exposing 2 endpoints beneath base path '/actuator'.
2021-07-06 09:50:59.743  INFO 4645 --- [           main] com.example.demo.Oauth2ApplicationTests : Started Oauth2ApplicationTests in 3.984 seconds (JVM running for
4.922)
2021-07-06 09:51:00.515  INFO 4645 --- [           main] o.s.b.m.w.SpringBootTestContext : Initializing Spring TestDispatcherServlet ''
2021-07-06 09:51:00.516  INFO 4645 --- [           main] o.s.t.web.servlet.TestDispatcherServlet : Initializing Servlet ''
2021-07-06 09:51:00.533  INFO 4645 --- [           main] o.s.t.web.servlet.TestDispatcherServlet : Completed initialization in 17 ms
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 5.614 s - in com.example.demo.Oauth2ApplicationTests
2021-07-06 09:51:01.074  INFO 4645 --- [extShutdownHook] o.s.s.concurrent.ThreadPoolTaskExecutor : Shutting down ExecutorService 'applicationTaskExecutor'
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ OAuth2 ---
[INFO] Building jar: /Users/syedusman/.jenkins/workspace/Jenkins-Docker/target/springboot-oauth2-authcode.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.3.12.RELEASE:repackage (repurchase) @ OAuth2 ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] --- spring-boot-maven-plugin:2.3.12.RELEASE:repackage (default) @ OAuth2 ---
[INFO] Replacing main artifact with repackaged archive
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ OAuth2 ---
[INFO] Installing /Users/syedusman/.jenkins/workspace/Jenkins-Docker/target/springboot-oauth2-authcode.jar to
/Users/syedusman/.m2/repository/com/security/Oauth2/0.0.1-SNAPSHOT/Oauth2-0.0.1-SNAPSHOT.jar
[INFO] Installing /Users/syedusman/.jenkins/workspace/Jenkins-Docker/pom.xml to /Users/syedusman/.m2/repository/com/security/Oauth2/0.0.1-SNAPSHOT/Oauth2-0.0.1-
SNAPSHOT.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 11.627 s
[INFO] Finished at: 2021-07-06T09:51:03+04:00
[INFO] -----
Finished: SUCCESS

```

Steps to download, setup and run locally to test and validate

Pre-requisite:

JDK 8 or higher is installed
Docker is installed like Docker for Desktop
Maven is Installed
Any IDE like Spring Tool Suite or IntelliJ or Eclipse is available
Jenkins is Installed
Postman client is installed

There are **2 ways** to start this spring boot project.

- A) Download the Code from GIT and import the project in to any IDE , launch as Spring boot App and execute the Junit test case in the project .

Note: Junit Test case has been written only for grant Type=password flow

Or

- B) Pull the Docker image from the docker hub and docker run to launch the spring boot app.

Note: Testing through Postman Client is available for both grant Type=password and Grant type= Authorization Code flow.

Pull down the code from the GitHub repo for reference and import it to any IDE as a Spring Boot Project

https://github.com/Syed-UsmanAli/OAuth2_Authorization_Server

Command :

On any directory where the project needs to be setup execute below command

Let's say directory is c:/users/syedusman/OAuth2_Project

In Terminal /Command Prompt execute the below commands , as shown in screenshot.
Verify it is cloned properly with Project source code files, pom.xml and Dockerfile.

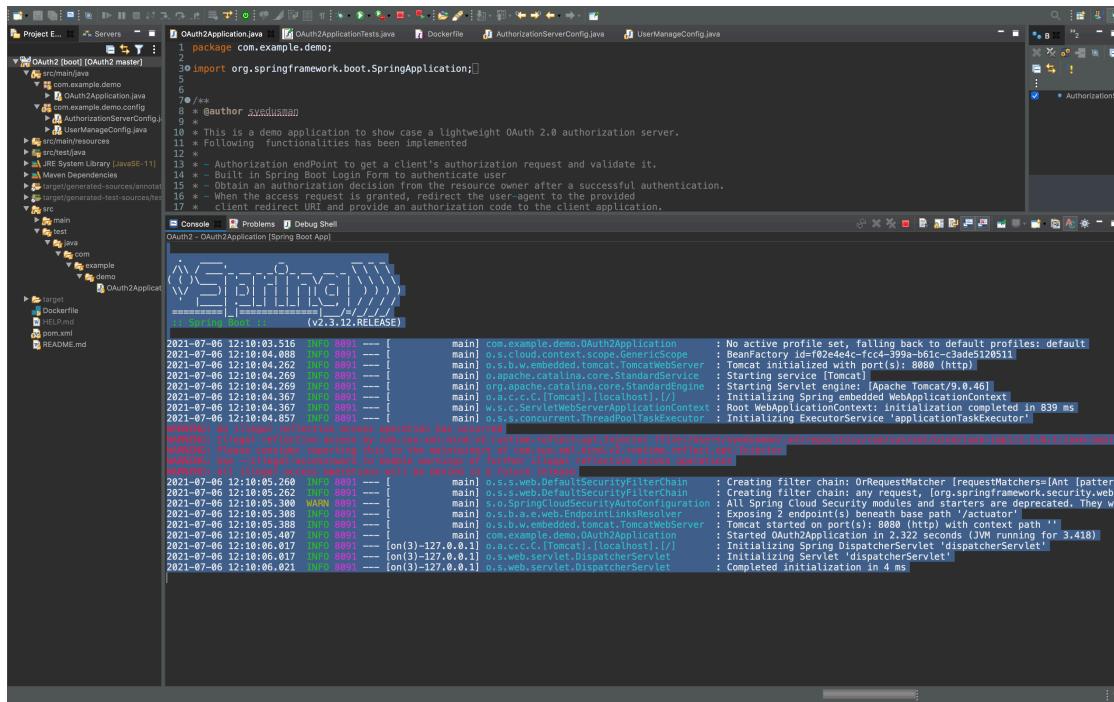
```
- git init
- git clone https://github.com/Syed-UsmanAli/OAuth2_Authorization_Server.git
```

```

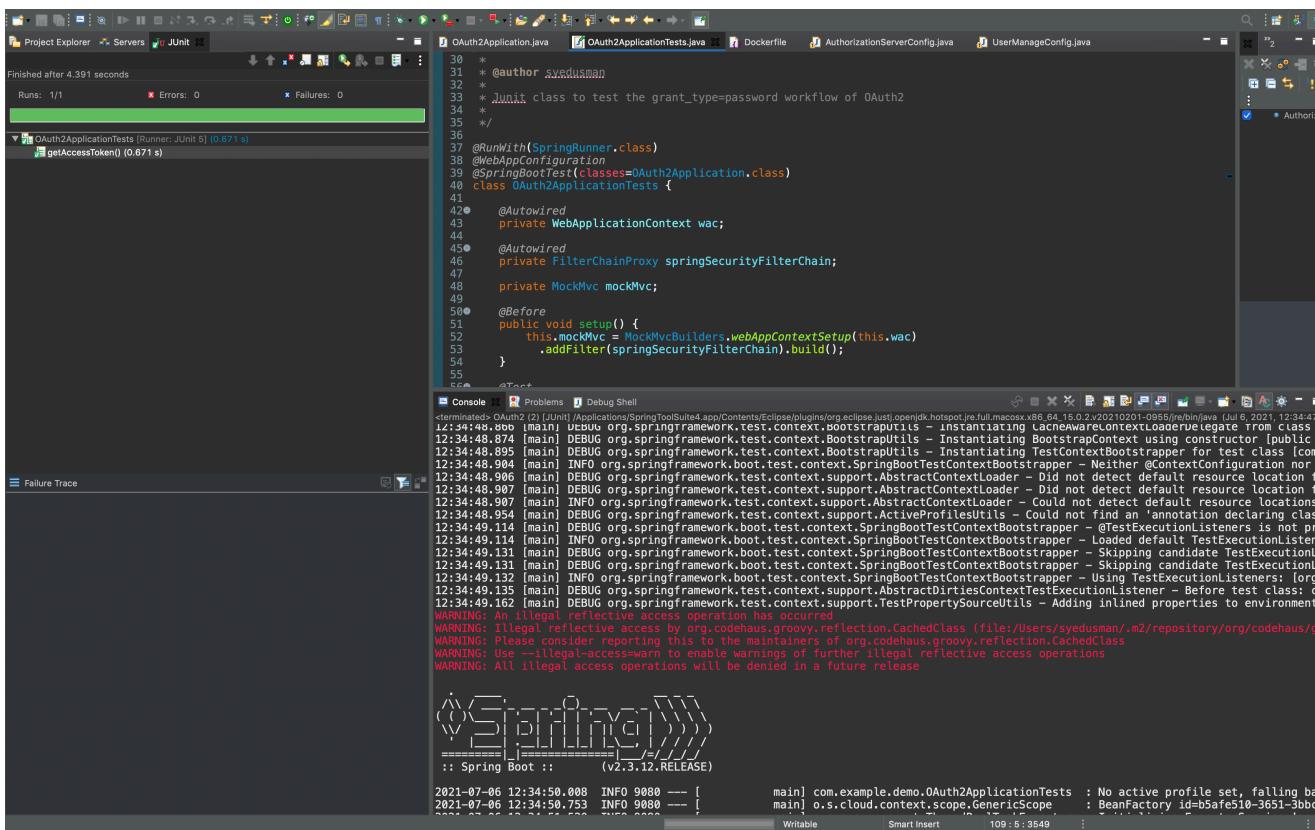
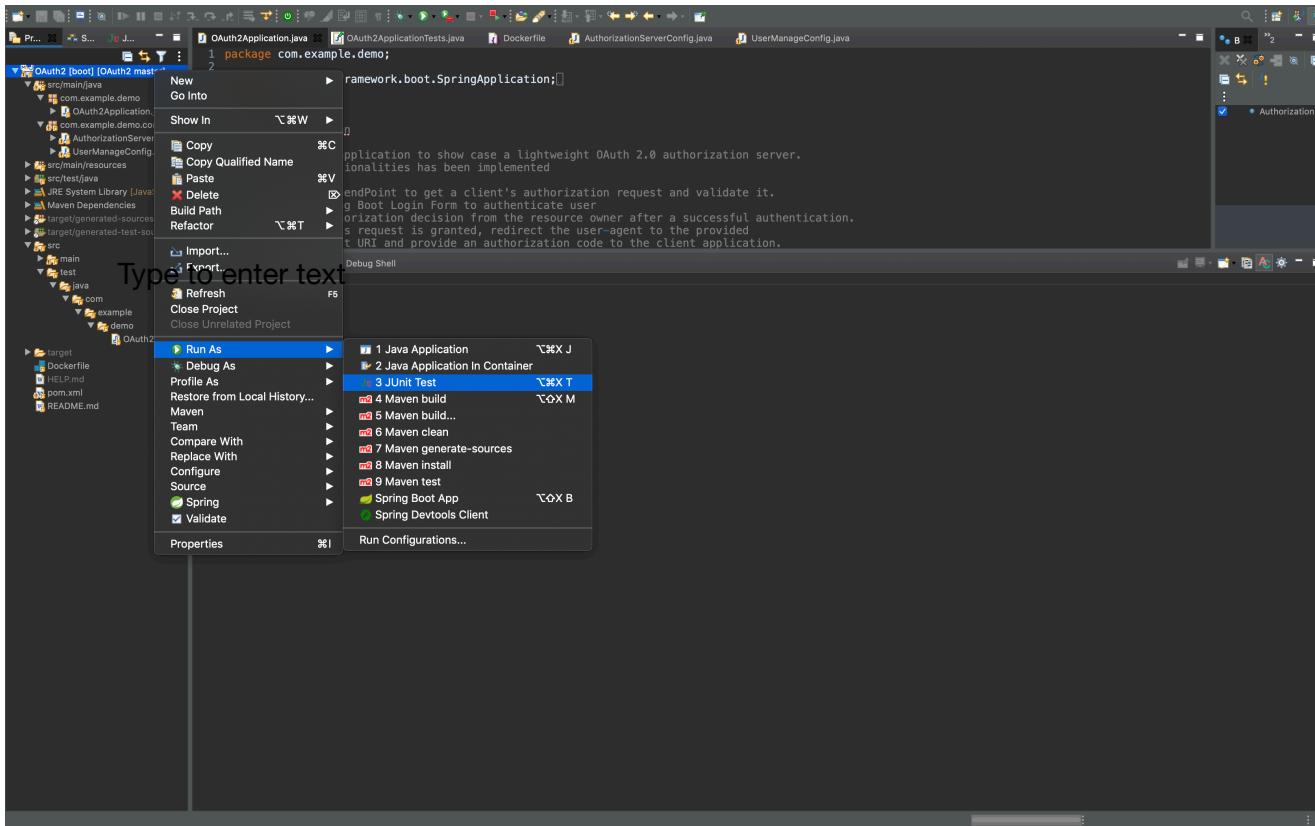
Syeds-MacBook-Pro:OAuth2_Project syedusman$ pwd
/Users/syedusman/OAuth2_Project
Syeds-MacBook-Pro:OAuth2_Project syedusman$ git init
Initialized empty Git repository in /Users/syedusman/OAuth2_Project/.git/
Syeds-MacBook-Pro:OAuth2_Project syedusman$ git clone https://github.com/Syed-UsmanAli/OAuth2_Authorization_Server.git
Cloning into 'OAuth2_Authorization_Server'...
remote: Enumerating objects: 142, done.
remote: Counting objects: 100% (142/142), done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 142 (delta 32), reused 125 (delta 21), pack-reused 0
Receiving objects: 100% (142/142), 61.00 KiB | 2.54 MiB/s, done.
Resolving deltas: 100% (32/32), done.
Syeds-MacBook-Pro:OAuth2_Project syedusman$ ls
OAuth2_Authorization_Server
Syeds-MacBook-Pro:OAuth2_Project syedusman$ cd OAuth2_Authorization_Server
Syeds-MacBook-Pro:OAuth2_Authorization_Server syedusman$ ls
Dockerfile README.md pom.xml src
Syeds-MacBook-Pro:OAuth2_Authorization_Server syedusman$ ls -lrt
total 32
-rw-r--r-- 1 syedusman staff 161 Jul 6 11:56 Dockerfile
-rw-r--r-- 1 syedusman staff 9 Jul 6 11:56 README.md
-rw-r--r-- 1 syedusman staff 4359 Jul 6 11:56 pom.xml
drwxr-xr-x 4 syedusman staff 128 Jul 6 11:56 src
Syeds-MacBook-Pro:OAuth2_Authorization_Server syedusman$ 

```

- 3) Import the downloaded project in to an IDE, say STS.Right click on Project root directory and run as Spring Boot App



In order to execute Junit test case, stop the Spring boot Server as the Junit will also bring up the spring boot app on the same port 8080 and then run the Test class defined in the project under “src/main/test” package

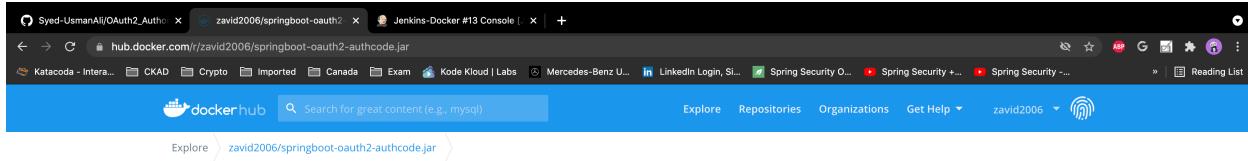
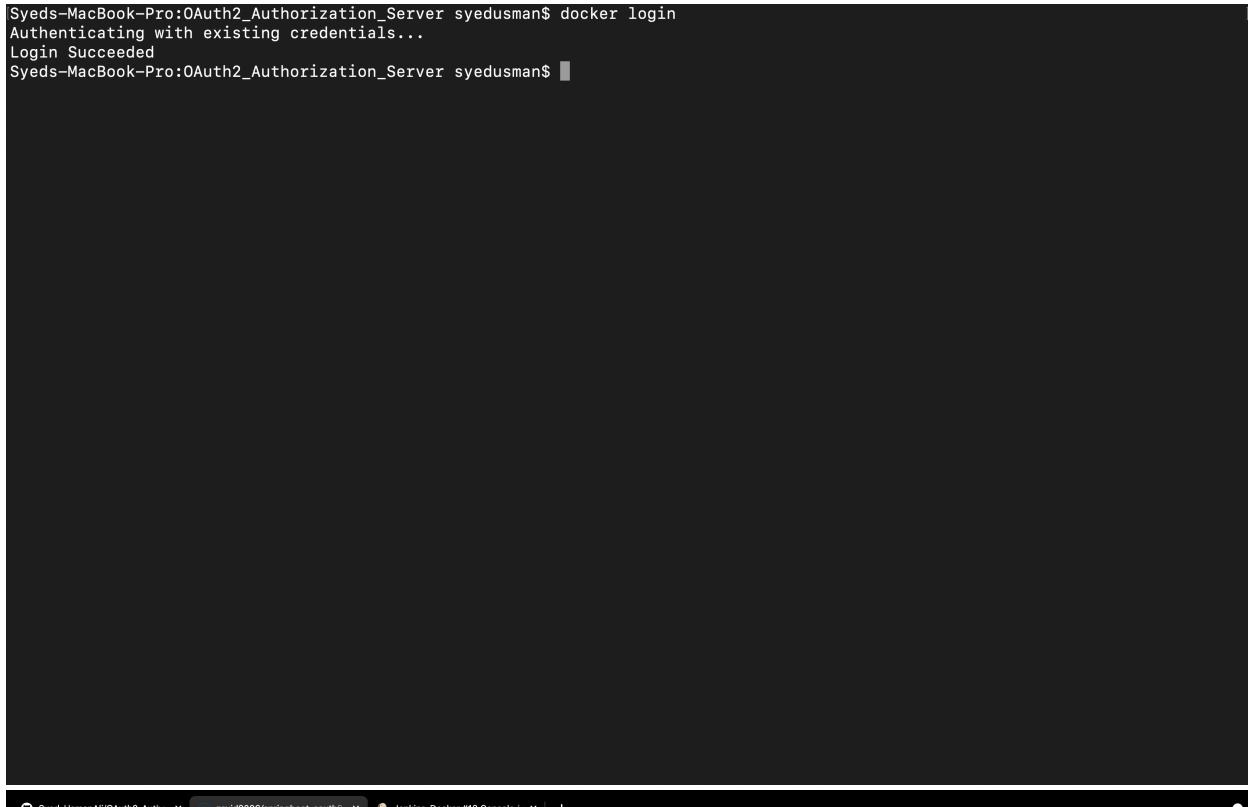


Now, we will see the Alternate way of how to pull the docker image from docker hub and run instead of downloading the code from GitHub.

Prerequisite : Docker to be installed and a docker account is available.

1) Execute Docker Login in Terminal / Command Prompt

```
Syeds-MacBook-Pro:OAuth2_Authorization_Server syedusman$ docker login
Authenticating with existing credentials...
Login Succeeded
Syeds-MacBook-Pro:OAuth2_Authorization_Server syedusman$
```



A detailed screenshot of the Docker Hub repository page for "zavid2006/springboot-oauth2-authcode.jar". The top navigation bar includes "Explore", "Repositories", "Organizations", "Get Help", and a user dropdown for "zavid2006". The main content area has tabs for "Overview" (selected) and "Tags". The "Overview" tab shows a placeholder message "No overview available" and a note "This repository doesn't have an overview". The "Tags" tab is currently empty. To the right, there's a "Docker Pull Command" section with the command "docker pull zavid2006/springboot-oauth2-authcode.jar" and a copy icon. Below that is an "Owner" section showing a profile picture and the name "zavid2006".

Pull the Image from docker hub by executing below command in terminal in any directory as shown below

docker pull zavid2006/springboot-oauth2-authcode.jar

```
Syeds-MacBook-Pro:OAuth2Docker syedusman$ pwd
/users/syedusman/OAuth2Docker
Syeds-MacBook-Pro:OAuth2Docker syedusman$ docker login
Authenticating with existing credentials...
Login Succeeded
Syeds-MacBook-Pro:OAuth2Docker syedusman$ docker pull zavid2006/springboot-oauth2-authcode.jar
Using default tag: latest
latest: Pulling from zavid2006/springboot-oauth2-authcode.jar
Digest: sha256:018637de065b7c6abc78e7554d072443772c661f2626f5f2aea7833a906aa6e3
Status: Image is up to date for zavid2006/springboot-oauth2-authcode.jar:latest
docker.io/zavid2006/springboot-oauth2-authcode.jar:latest
Syeds-MacBook-Pro:OAuth2Docker syedusman$ █
```

Once the docker image is downloaded, execute the below command by running the spring boot application say on the local port 9090

docker run -p 9090:8080 zavid2006/springboot-oauth2-authcode.jar

9090 —> LocalPort

8080 —> Port on which Spring Boot application is up

Once executed, notice below screenshot where the spring boot application launches

```

Syeds-MacBook-Pro:OAuth2Docker syedusman$ pwd
/users/syedusman/OAuth2Docker
Syeds-MacBook-Pro:OAuth2Docker syedusman$ docker login
Authenticating with existing credentials...
Login Succeeded
Syeds-MacBook-Pro:OAuth2Docker syedusman$ docker pull zavid2006/springboot-oauth2-authcode.jar
Using default tag: latest
latest: Pulling from zavid2006/springboot-oauth2-authcode.jar
Digest: sha256:018637de065b7c6abc78e7554d072443772c661f2626f5f2aea7833a906aa6e3
Status: Image is up to date for zavid2006/springboot-oauth2-authcode.jar:latest
docker.io/zavid2006/springboot-oauth2-authcode.jar:latest
Syeds-MacBook-Pro:OAuth2Docker syedusman$ docker run -p 9090:8080 springboot-oauth2-authcode.jar

```
 _\ / _`_ - _ - _(`)_ -- _ - _\ ``\
 (()_`_ | [] | [] | [] | \ [])
 ` | [] | [] | [] | ([])))
 ` []_ . . [] | [] | [] , / / /
=====| _=====| _/_=/ / / /
:: Spring Boot :: (v2.3.12.RELEASE)

2021-07-06 16:48:57.986 INFO 1 --- [main] com.example.demo.OAuth2Application : No active profile set, falling
back to default profiles: default
2021-07-06 16:49:00.097 INFO 1 --- [main] o.s.cloud.context.scope.GenericScope : BeanFactory id=1442c386-8e74-3b
49-8f24-9b1d07547fb8
2021-07-06 16:49:00.732 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s)
: 8080 (http)
2021-07-06 16:49:00.755 INFO 1 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-07-06 16:49:00.756 INFO 1 --- [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apach
e Tomcat/9.0.46]
2021-07-06 16:49:00.905 INFO 1 --- [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded We
bApplicationContext
2021-07-06 16:49:00.906 INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: ini
tialization completed in 2862 ms

```

Application is now ready to accept request as shown below.

```

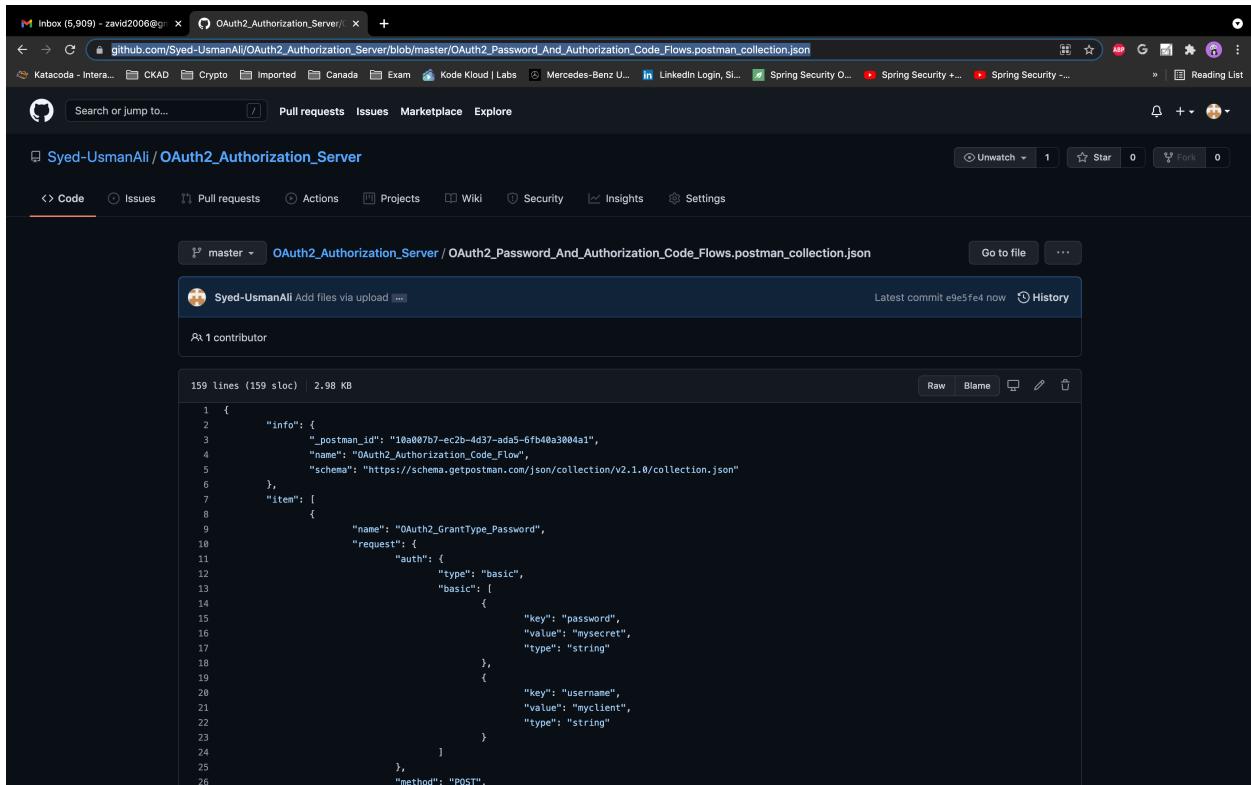
2021-07-06 16:49:00.906 INFO 1 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: ini
tialization completed in 2862 ms
2021-07-06 16:49:02.159 INFO 1 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'a
pplicationTaskExecutor'
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by com.sun.xml.bind.v2.runtime.reflect.opt.Injector (jar:file:/springboot-oauth2-authcode.jar
!/BOOT-INF/lib/jaxb-impl-2.3.0.1.jar!) to method java.lang.ClassLoader.defineClass(java.lang.String,byte[],int,int)
WARNING: Please consider reporting this to the maintainers of com.sun.xml.bind.v2.runtime.reflect.opt.Injector
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
2021-07-06 16:49:03.385 INFO 1 --- [main] o.s.s.web.DefaultSecurityFilterChain : Creating filter chain: OrReques
tMatcher [requestMatchers=[Ant [pattern='/oauth/token'], Ant [pattern='/oauth/token_key']], Ant [pattern='/oauth/check_token']]]
[org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@78d50a3c, org.springframework.security.
web.context.SecurityContextPersistenceFilter@88a8218, org.springframework.security.web.header.HeaderWriterFilter@44d70181, org.
springframework.security.web.authentication.logout.LogoutFilter@26f1249d, org.springframework.security.web.authentication.www.Ba
sicAuthenticationFilter@4943defe, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@4163f1cd, org.springfram
ework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@23aae65, org.springframework.security.web.authentication.A
nonymousAuthenticationFilter@2fb69ff6, org.springframework.security.web.session.SessionManagementFilter@23c650a3, org.springfram
ework.security.web.access.ExceptionTranslationFilter@18ca3c62, org.springframework.security.web.access.intercept.FilterSecurityI
nterceptor@6df20aade]
2021-07-06 16:49:03.394 INFO 1 --- [main] o.s.s.web.DefaultSecurityFilterChain : Creating filter chain: any requ
est, [org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter@5ba745bc, org.springframework.secu
rity.web.context.SecurityContextPersistenceFilter@5d10455d, org.springframework.security.web.header.HeaderWriterFilter@2617f816,
org.springframework.security.web.csrf.CsrfFilter@27f3b6d6, org.springframework.security.web.authentication.logout.LogoutFilter@5
ee63cad, org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter@6759f091, org.springframework.secu
rity.web.authentication.ui.DefaultLoginPageGeneratingFilter@55b62629, org.springframework.security.web.savedrequest.RequestCacheAwareFilter@535b8c24, org.springfram
ework.security.web.servletapi.SecurityContextHolderAwareRequestFilter@4a951911, org.springframework.security.web.authenticati
on.AnonymousAuthenticationFilter@55b5e331, org.springframework.security.web.session.SessionManagementFilter@676f0a60, org.spring
framework.security.web.access.ExceptionTranslationFilter@757f675c, org.springframework.security.web.access.intercept.FilterSecur
ityInterceptor@867ba60]
2021-07-06 16:49:03.518 WARN 1 --- [main] s.o.SpringCloudSecurityAutoConfiguration : All Spring Cloud Security modul
es and starters are deprecated. They will be moved to individual projects in the next major release.
2021-07-06 16:49:03.542 INFO 1 --- [main] o.s.b.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath
base path '/actuator'
2021-07-06 16:49:03.675 INFO 1 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080
(HTTP) with context path ''
2021-07-06 16:49:03.705 INFO 1 --- [main] com.example.demo.OAuth2Application : Started OAuth2Application in 7.
532 seconds (JVM running for 8.769)

```

## TESTING through Postman Client

Note : Postman collection is uploaded to GitHub Repo. Please download and import those requests in to your postman client for local testing

[https://github.com/Syed-UsmanAli/OAuth2\\_Authorization\\_Server/blob/master/OAuth2\\_Password\\_And\\_Authorization\\_Code\\_Flows.postman\\_collection.json](https://github.com/Syed-UsmanAli/OAuth2_Authorization_Server/blob/master/OAuth2_Password_And_Authorization_Code_Flows.postman_collection.json)



The screenshot shows a GitHub repository page for 'Syed-UsmanAli / OAuth2\_Authorization\_Server'. The 'Code' tab is selected, displaying the contents of the 'OAuth2\_Password\_And\_Authorization\_Code\_Flows.postman\_collection.json' file. The file contains 159 lines of JSON code, representing a Postman collection. The code defines an OAuth2 Authorization Code Flow with a basic authentication type, using 'password' as the grant type and 'myclient' as the username and 'mysecret' as the password.

```
1 {
2 "info": {
3 "_postman_id": "18a007b7-ec2b-4d37-ada5-6fb40a3004a1",
4 "name": "OAuth2_Authorization_Code_Flow",
5 "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
6 },
7 "item": [
8 {
9 "name": "OAuth2_GrantType_Password",
10 "request": {
11 "auth": {
12 "type": "basic",
13 "basic": {
14 "key": "password",
15 "value": "mysecret",
16 "type": "string"
17 },
18 "key": "username",
19 "value": "myclient",
20 "type": "string"
21 }
22 }
23],
24],
25 "method": "POST"
26}
```

## TEST 1 - GrantType = Password Flow

Authorization End Point - /oauth/token

Query Parameters -

grant\_type=password&scope=read&username=user1&password=pass1

Authorization Header - Basic Auth , username is Client ID and password is ClientSecret which in this case are 'myclient' & 'mysecret'

On Submitting the Post request, In the response, as shown in above screenshot

The screenshot shows the Postman application interface. A POST request is made to `http://localhost:9090/oauth/token?grant_type=password&scope=read&username=user1&password=pass1`. The Authorization tab is selected, showing 'Basic Auth' with 'myclient' as the Username and 'mysecret' as the Password. The response status is 200 OK with a JSON payload containing an access token, token type (bearer), refresh token, expiration time (43199), and scope (read).

```

1 {
2 "access_token": "8d1fdbfa3-4e4f-4747-b5b0-2c194035c71b",
3 "token_type": "bearer",
4 "refresh_token": "0fffa7e19-2dd7-4348-89e9-98c0369f500e",
5 "expires_in": 43199,
6 "scope": "read"
7 }

```

**“Access Token” and “Refresh Token”** are generated with an expiry time.

This “Access Token” can then be used by the Client Application to access resource from the “Resource Server”.

## TEST 2 - GrantType = Authorization Code

In this flow, first **“Authorization Code”** needs to be obtained by the client before, sending it to get the **“Access Token”** in order to access the Resource from Resource Server.

**Test Steps:**

A) Open any web browser, and type in the below

[http://localhost:9090/oauth/authorize?  
response\\_type=code&client\\_id=myappclient&scope=read](http://localhost:9090/oauth/authorize?response_type=code&client_id=myappclient&scope=read)

**Note: 9090 - Port on which we exposed the spring boot application to run locally**

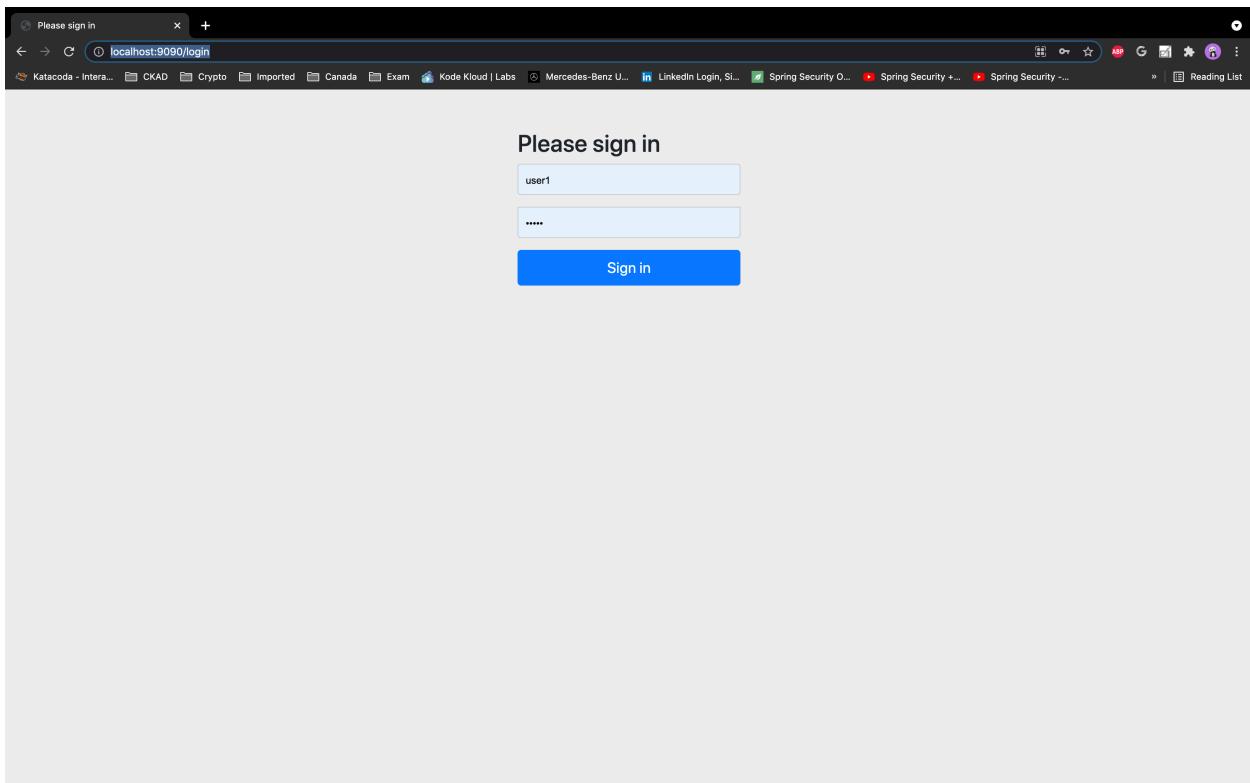
Response type = Code ( Represents the Authorization Code Flow)

Client ID - ID of the client application

Scope - What scopes are made available on the resources

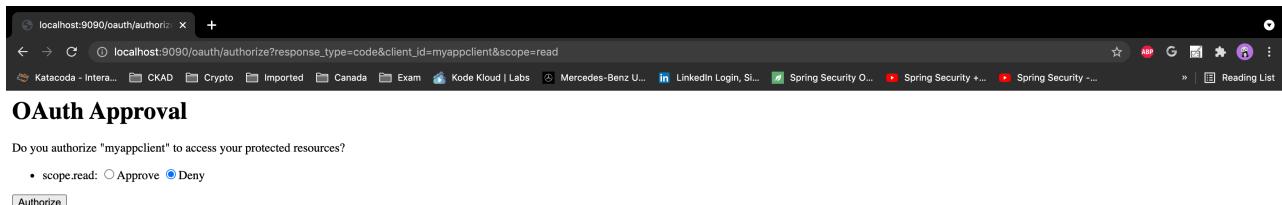
On loading the url, it launches the default Spring Boot Login Form , in order for the user to enter his login credentials as shown below.

<http://localhost:9090/login>



On Click of Sign In, it navigates to the **/oauth/authorize** **OAuth Approval Page** where it requests for **user consent** to authorize whether the **client “myappclient”** is allowed to access the **protected resource** on the **scopes** defined

Once the user selects “**Approve**” & clicks on “**Authorize**” button , the auth code is sent back to the “**Redirect URI**” defined. Since this



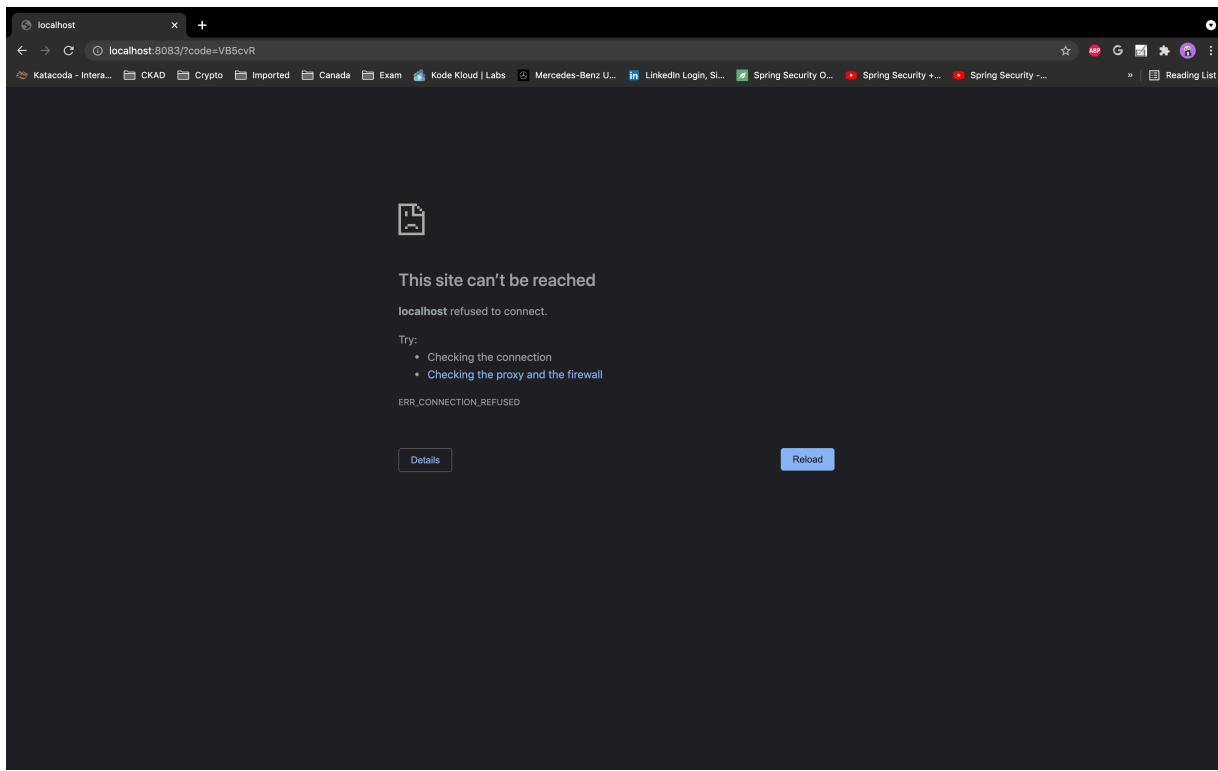
is for demo purpose only, the redirect uri is defined as **localhost** on a different port in order to receive the **auth code** as shown below.

Notice the **auth code generated on the browser url** , example in this case, codes generated as **code=VB5cvR** and ignore the site not reached. In real world applications, it will be redirected to the client applications

Now pass this Auth Code to the **/oauth/token** to get the “Access Token” through Postman Client as shown below

```
http://localhost:9090/oauth/token?
grant_type=authorization_code&scope=read&code=VB5cvR
```

Note :  
**Grant Type** is “Authorization Code”



**Code** is the generated code value from the previous /oauth/authorize flow.

Note : In this example, since the port is exposed on 9090, in the **postman** request, it is mentioned as **localhost:9090**, please change it accordingly as per the port that you would expose on the docker run command that was executed earlier.

The screenshot shows the Postman interface with a successful OAuth2 token exchange. The request URL is `http://localhost:9090/oauth/token?grant_type=authorization_code&scope=read&code=VB5cvR`. The response body is a JSON object containing the access token, token type, refresh token, expiration time, and scope.

```
1 {
2 "access_token": "bddc0290-58f3-456b-a28c-6b27302dff37",
3 "token_type": "bearer",
4 "refresh_token": "e16061b0-0a2d-49f8-a453-3e47c4a2a330",
5 "expires_in": 43199,
6 "scope": "read"
7 }
```

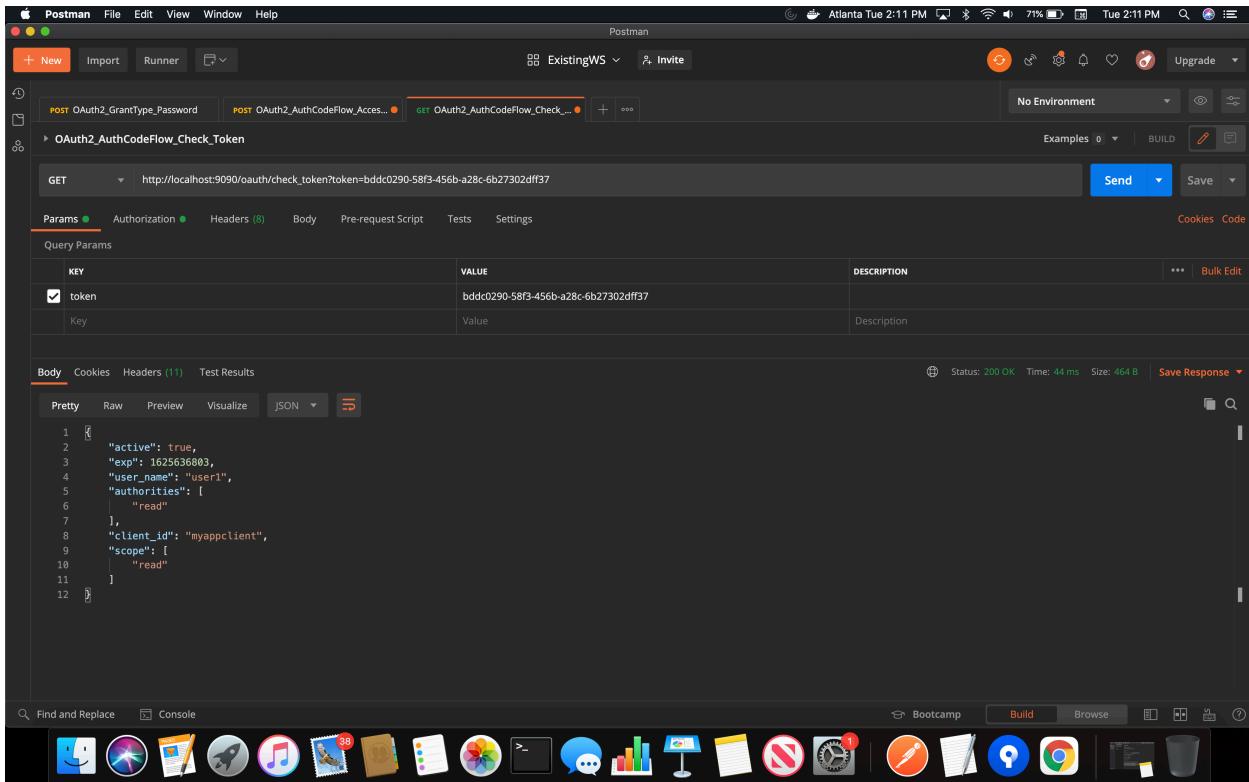
**“Access Token” is successfully generated!!! for the “OAuth2 - Authorization code” flow**

Now, using this “Access Token”, the client application can access the needed resource from the “Resource Server”.

### TEST 3 - Check Access Token

In this Test case, validity of the “Access Token” that was generated in the previous step is checked by accessing the url /oauth/check\_token

As shown in the below screenshot, since the “Access Token” is still valid, it would respond the “user” info along with the “scope” defined.



In Summary, we have implemented the “**Authorization Server**”, for both types of **Grant Type Flow - “password” and “authorization code”** to generate a valid “**Access Token**”. Also “**Refresh Token**” is generated and the validity of the Access token is checked through check token. Using the generated “Access Token”, the needed resource can then be accessed from the “Resource Server”.