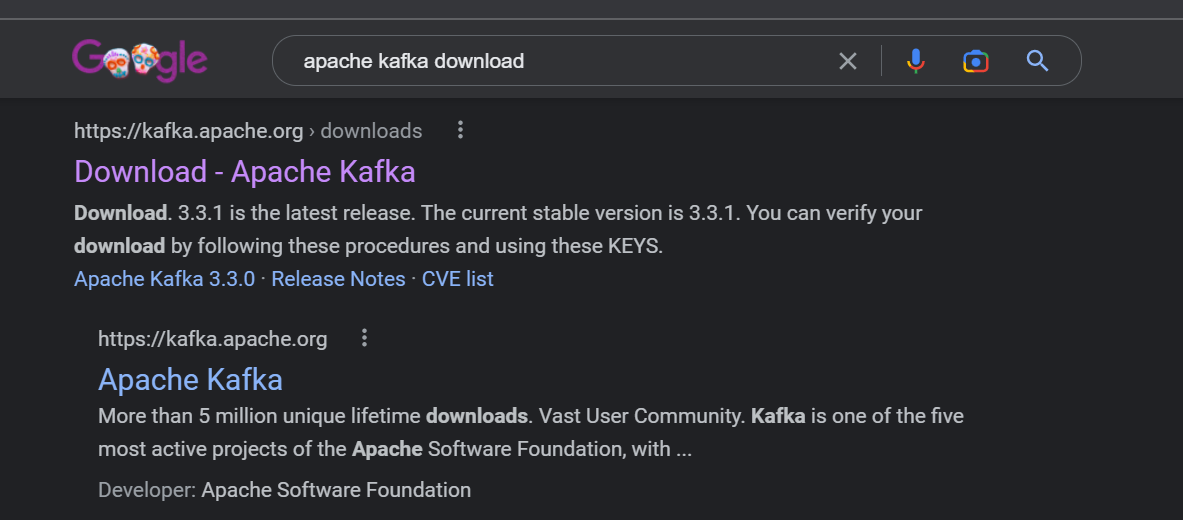
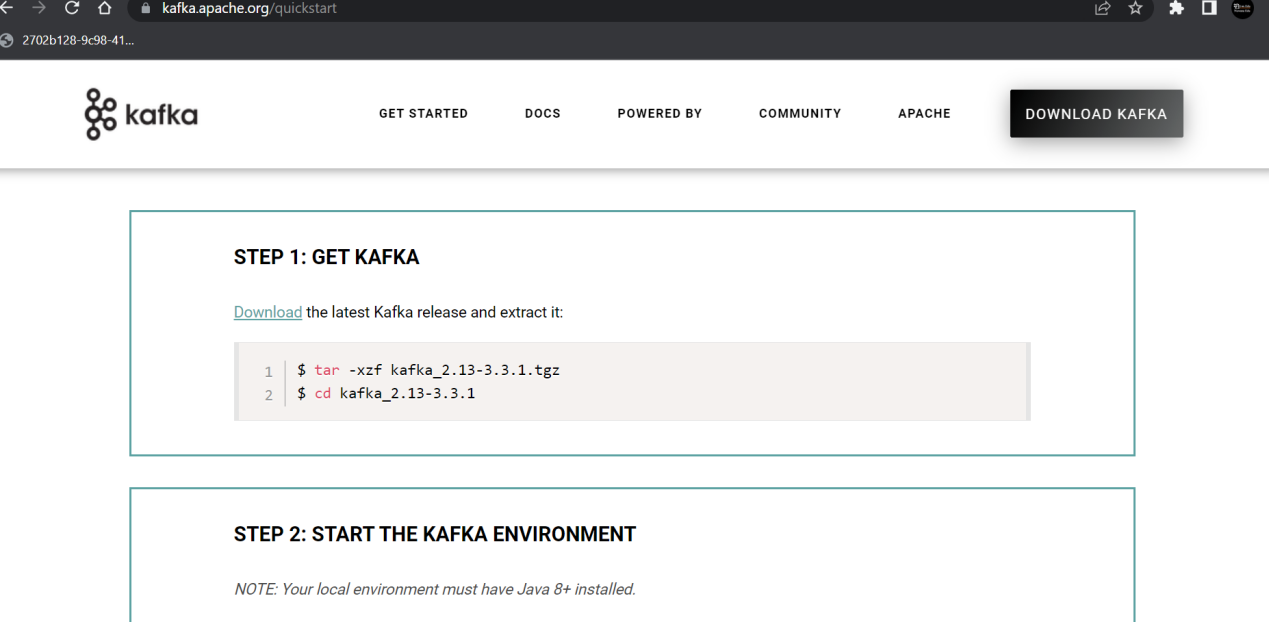


KAFKA

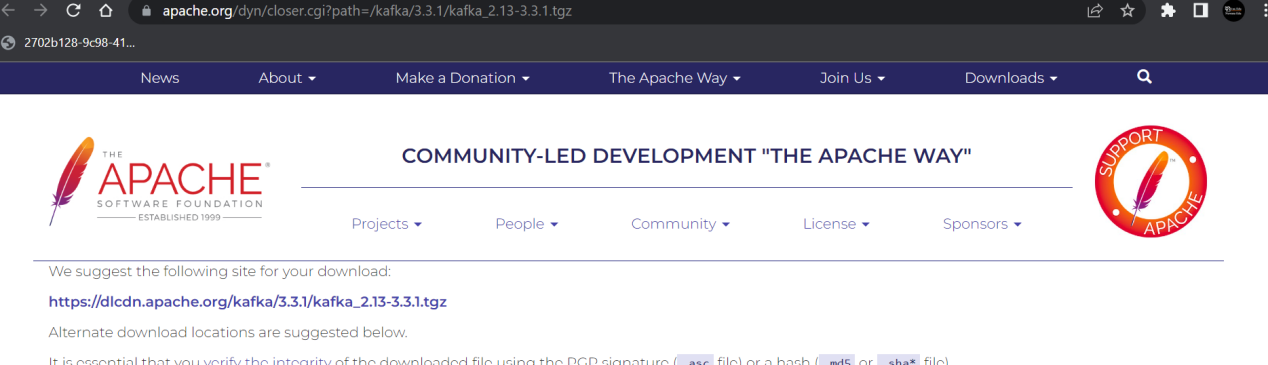


<https://kafka.apache.org/downloads>

GetStart - QuickStart



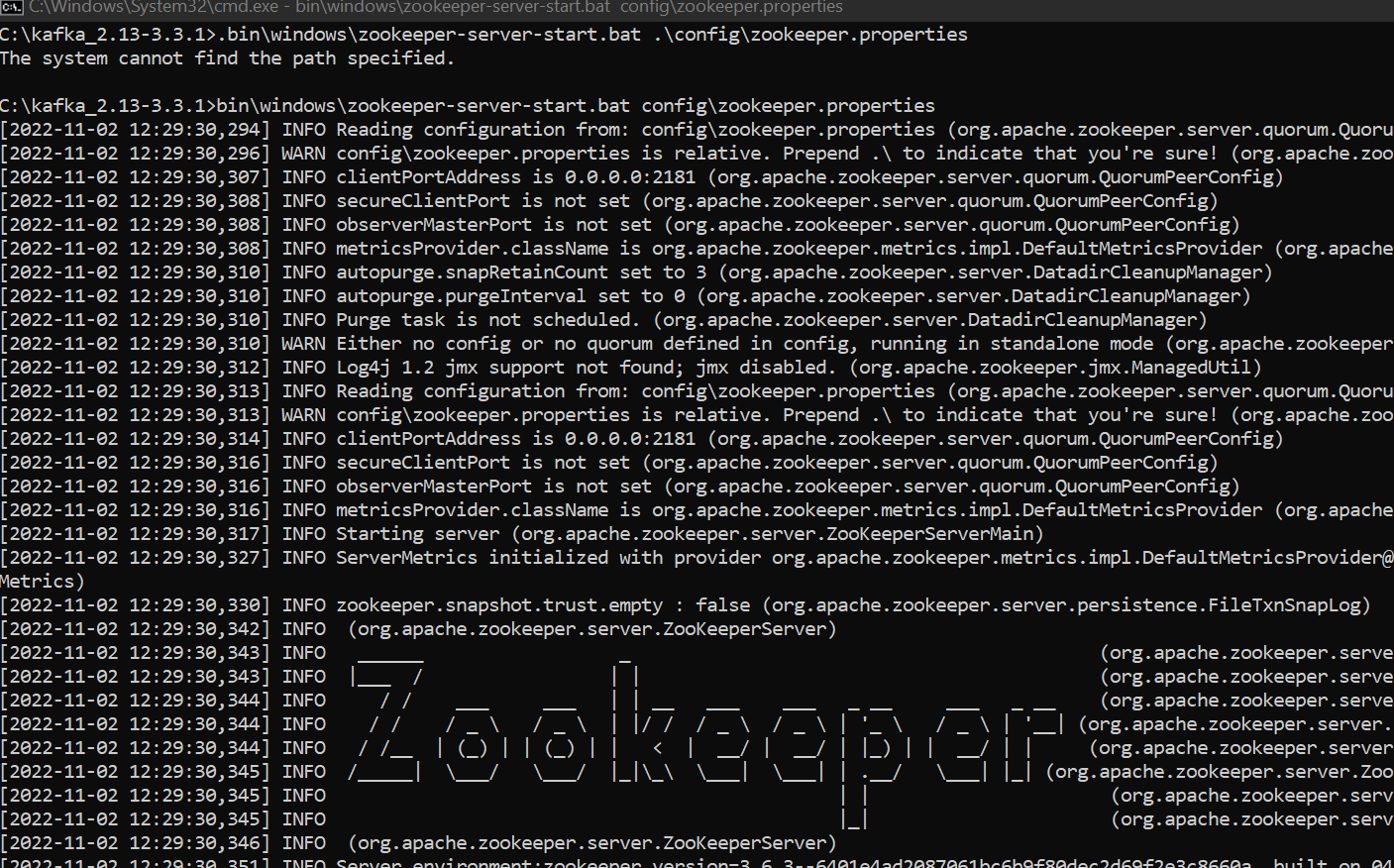
Download



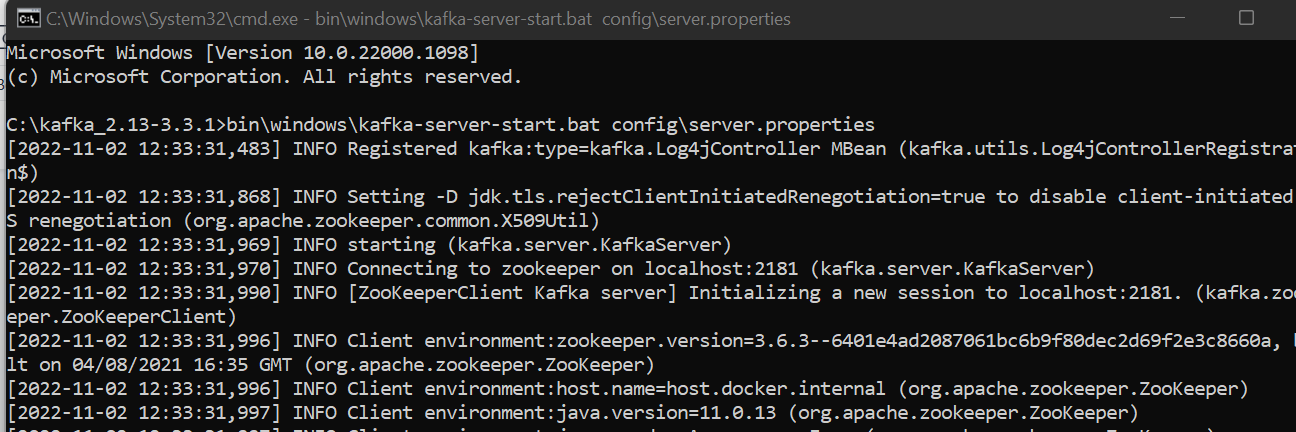
Download and extract.

Java 8 is needed.

bin\windows\zookeeper-server-start.bat config\zookeeper.properties



bin\windows\kafka-server-start.bat config\server.properties



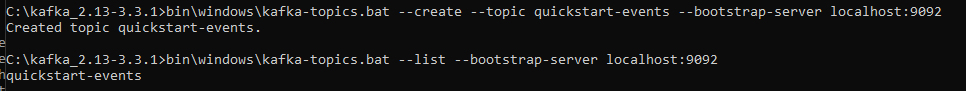
Zookeeper default port : 2181

Broker default port : 9091

Kafka default port : 9092

bin\windows\kafka-topics.bat --create --topic quickstart-events --bootstrap-server localhost:9092

bin\windows\kafka-topics.bat --list --bootstrap-server localhost:9092



STEP 1: DOWNLOAD AND INSTALL KAFKA

https://dlcdn.apache.org/kafka/3.2.0/...

STEP 2: START THE KAFKA ENVIRONMENT

# Start the ZooKeeper service

.\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties

# Start the Kafka broker service

.\bin\windows\kafka-server-start.bat .\config\server.properties

STEP 3: CREATE A TOPIC TO STORE YOUR EVENTS

.\bin\windows\kafka-topics.bat --create --topic myTopicOne --bootstrap-server localhost:9092

STEP 4: WRITE SOME EVENTS INTO THE TOPIC (ctl+c)

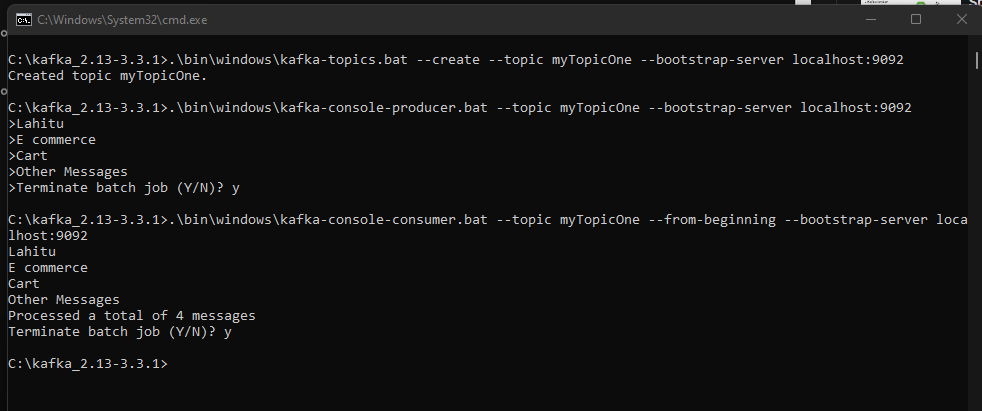
.\bin\windows\kafka-console-producer.bat --topic myTopicOne --bootstrap-server localhost:9092

STEP 5: READ THE EVENTS

.\bin\windows\kafka-console-consumer.bat --topic myTopicOne --from-beginning --bootstrap-server localhost:9092

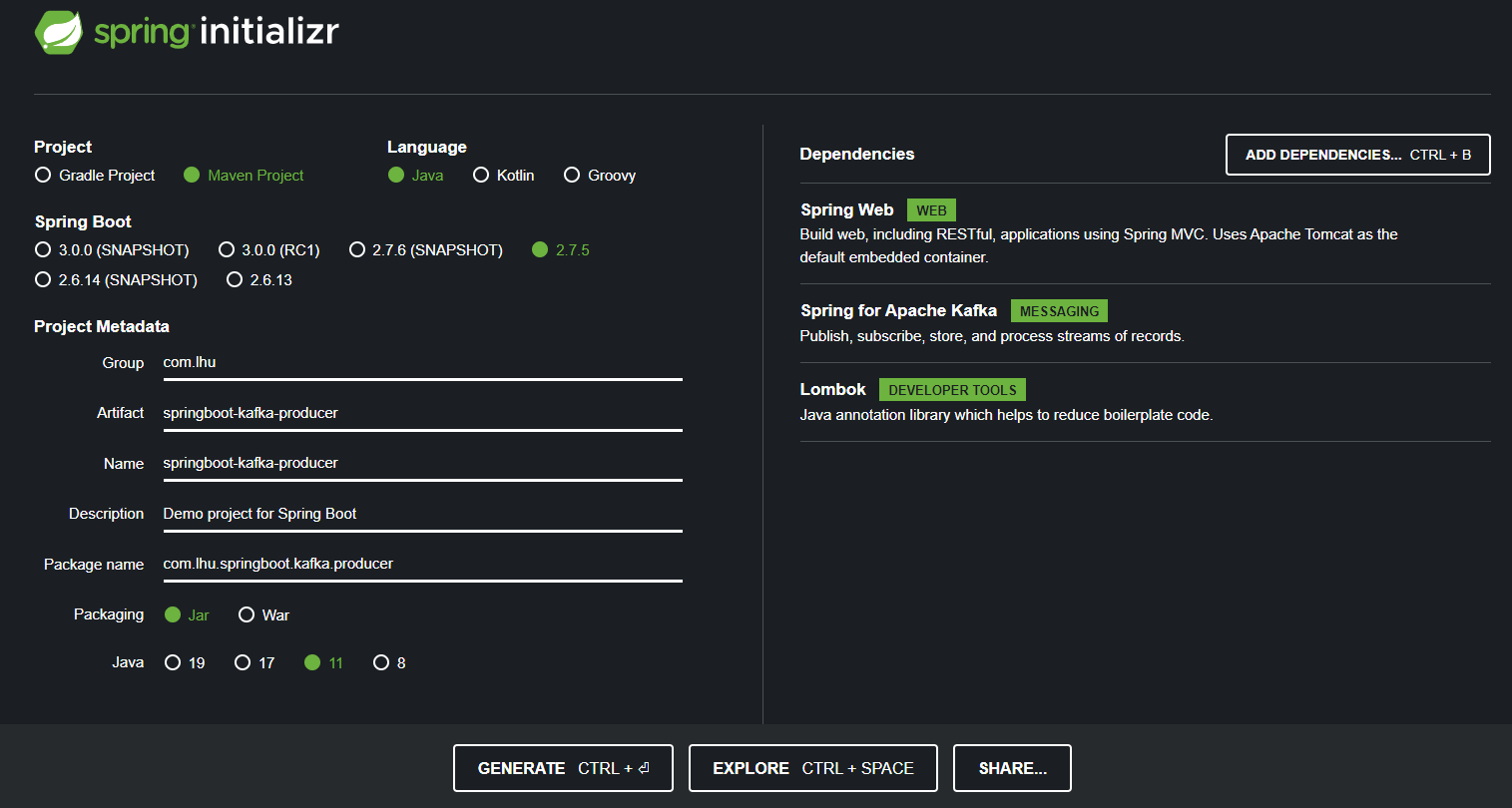
hello world

topic demo



Code Example -

01.Producer



….………………………………………………………………………………………………………………………………………………………

Installing

<https://github.com/Java-Techie-jt/springboot-rabbitmq-example>

1.Download and install ERlang http://erlang.org/download/otp\_win64\_22.3.exe

2.Downlaod and install RabbitMQ https://github.com/rabbitmq/rabbitmq-server/releases/download/v3.8.8/rabbitmq-server-3.8.8.exe

3.Go to RabbitMQ Server install Directory C:\Program Files\RabbitMQ Server\rabbitmq\_server-3.8.3\sbin

4.Run command rabbitmq-plugins enable rabbitmq\_management

5.Run command rabbitmq-service.bat start

6.Open browser and enter http://localhost:15672/ to redirect to RabbitMQ Dashboard

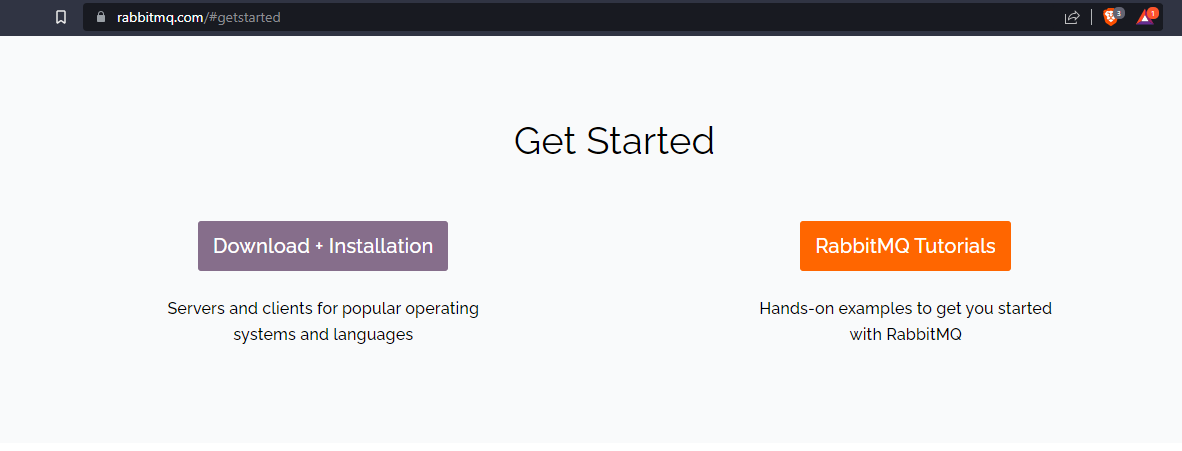
7.Also we can Open it with IP Address http://127.0.0.1:15672

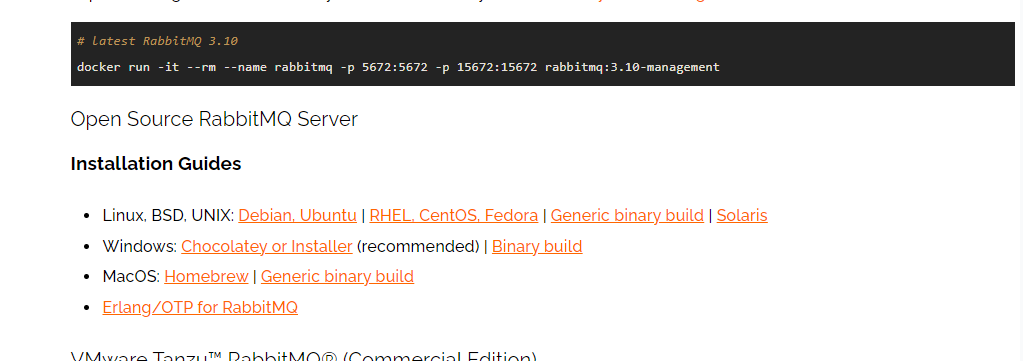
8.Login page default username and password is guest

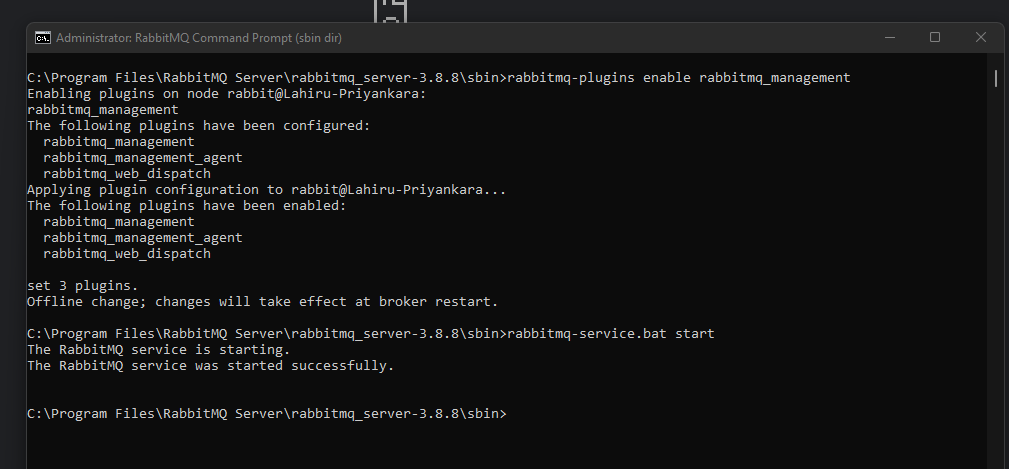
9.After successfully login you should see RabbitMQ Home page

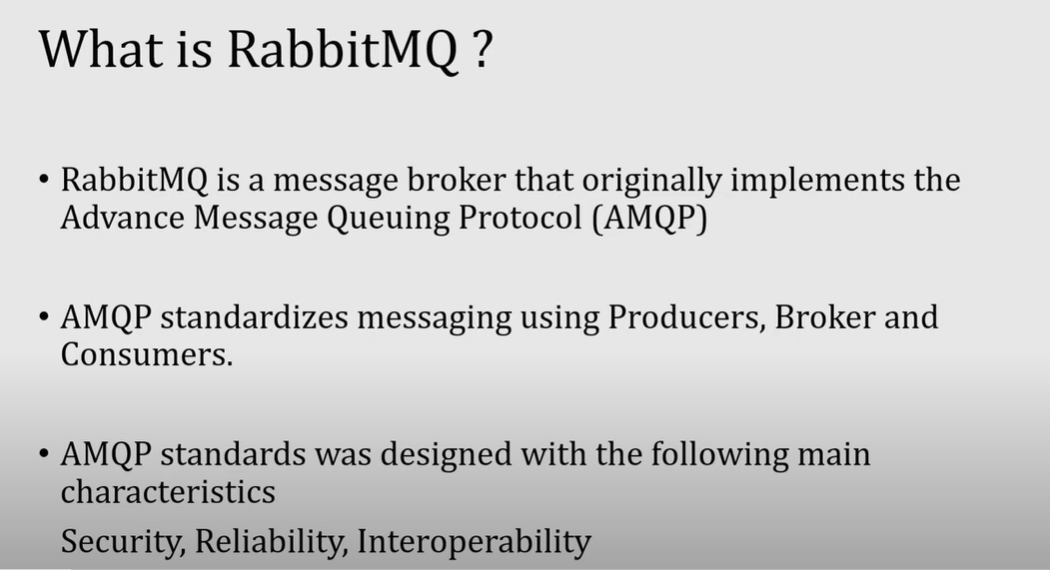
Go to - <https://www.rabbitmq.com/>

Go to Get started - <https://www.rabbitmq.com/#getstarted>





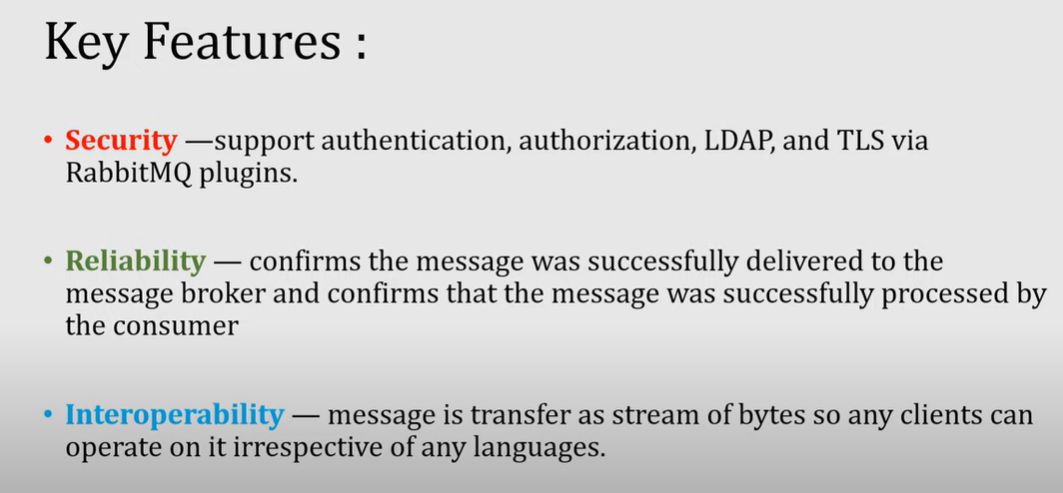




Producer - produce the message

Broker-Hold the message

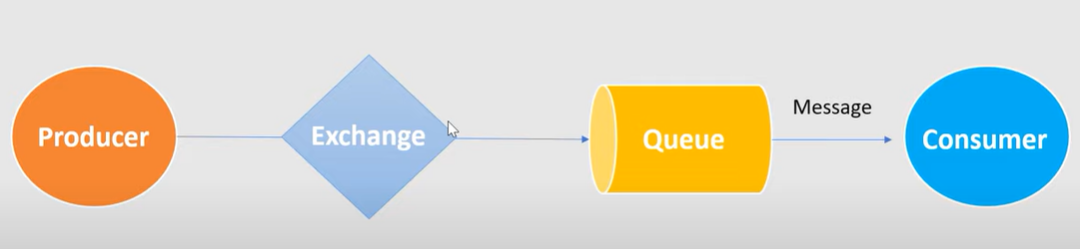
Consumer-Consume the message



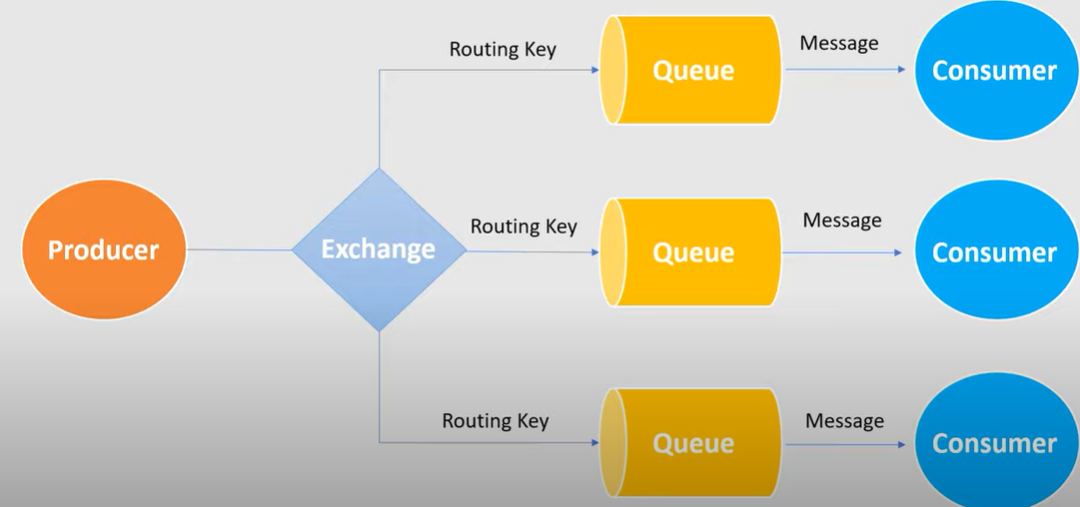
Traditional queue



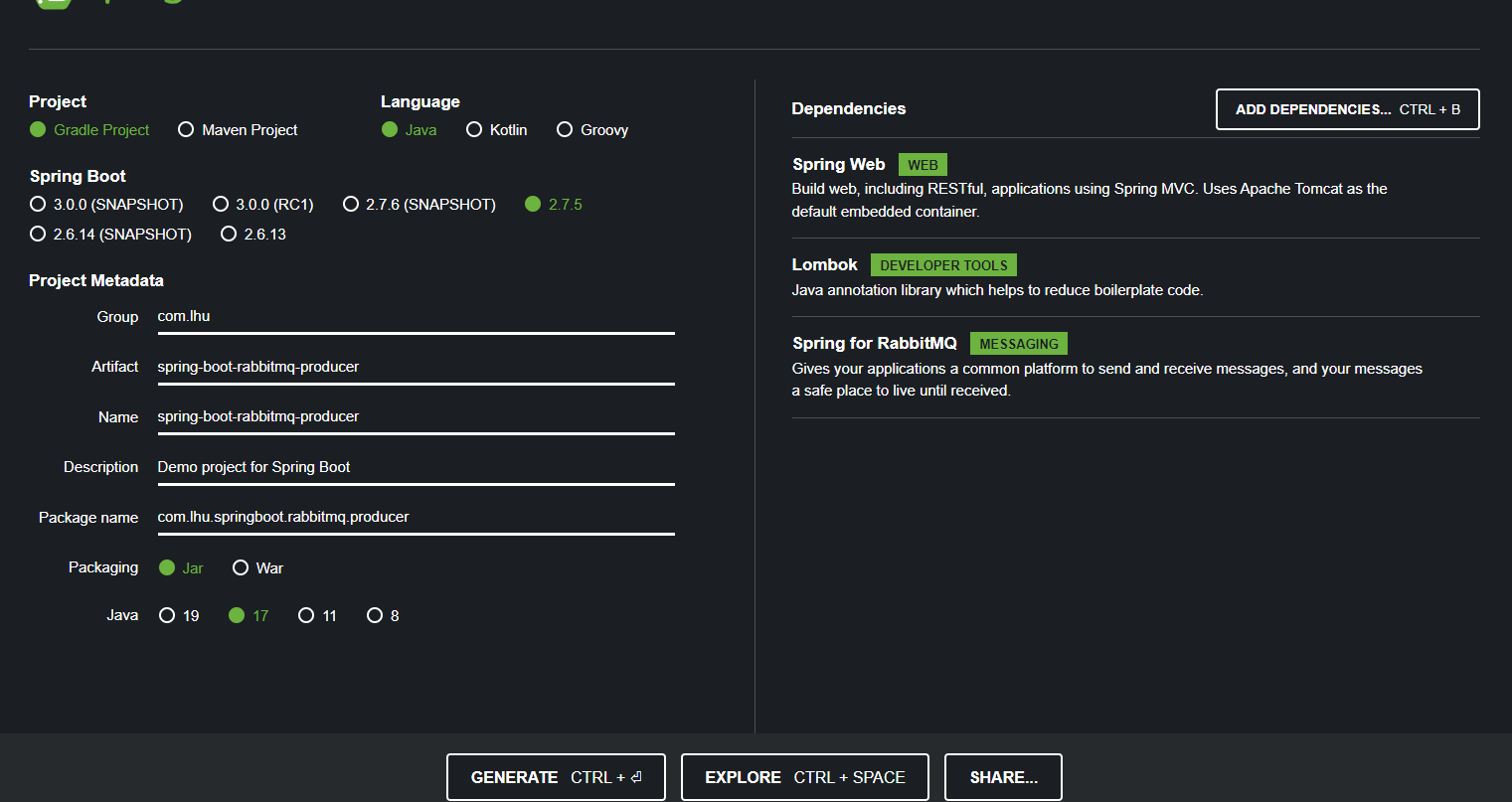
In rabbit MQ



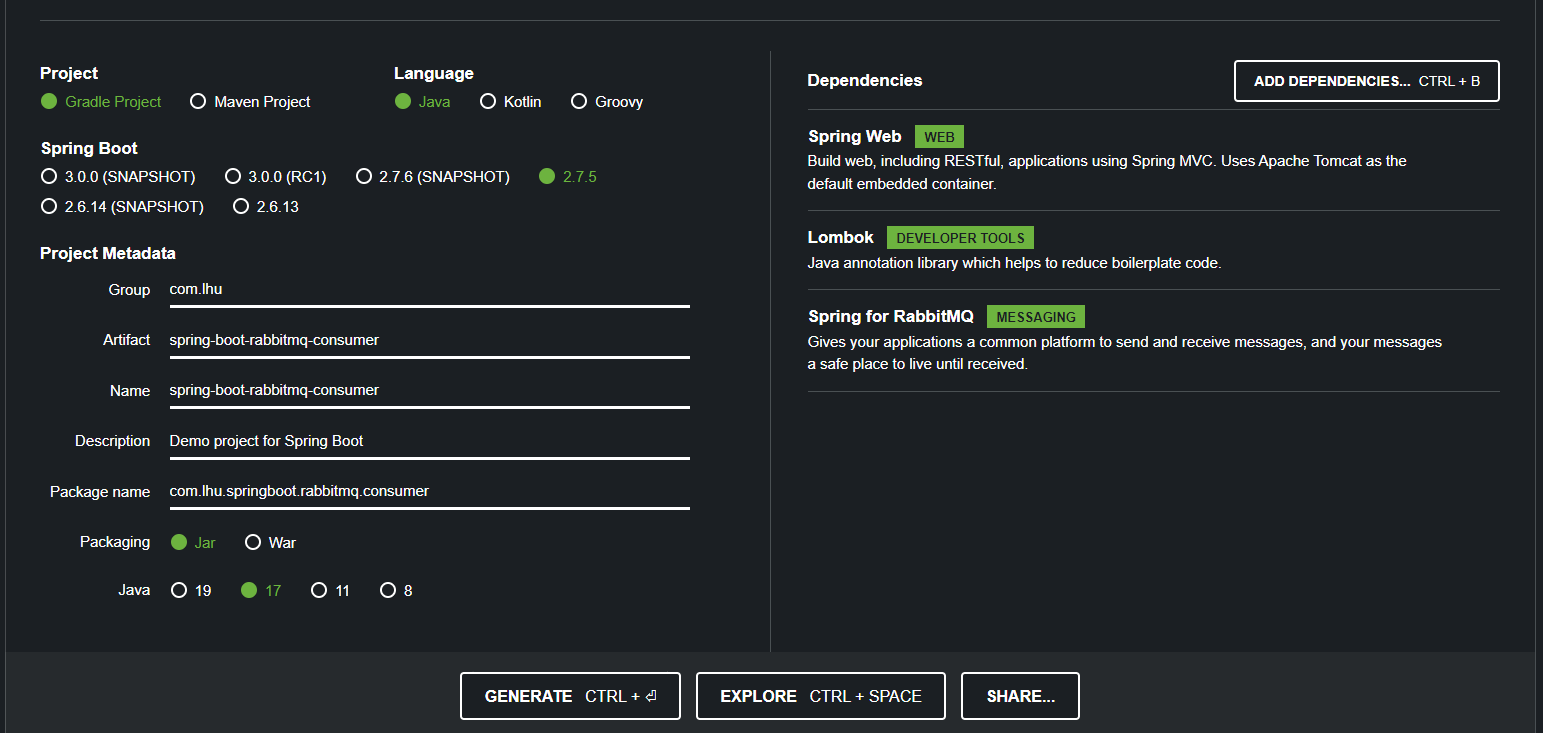
RabbtiMq in advance



Producer



Consumer



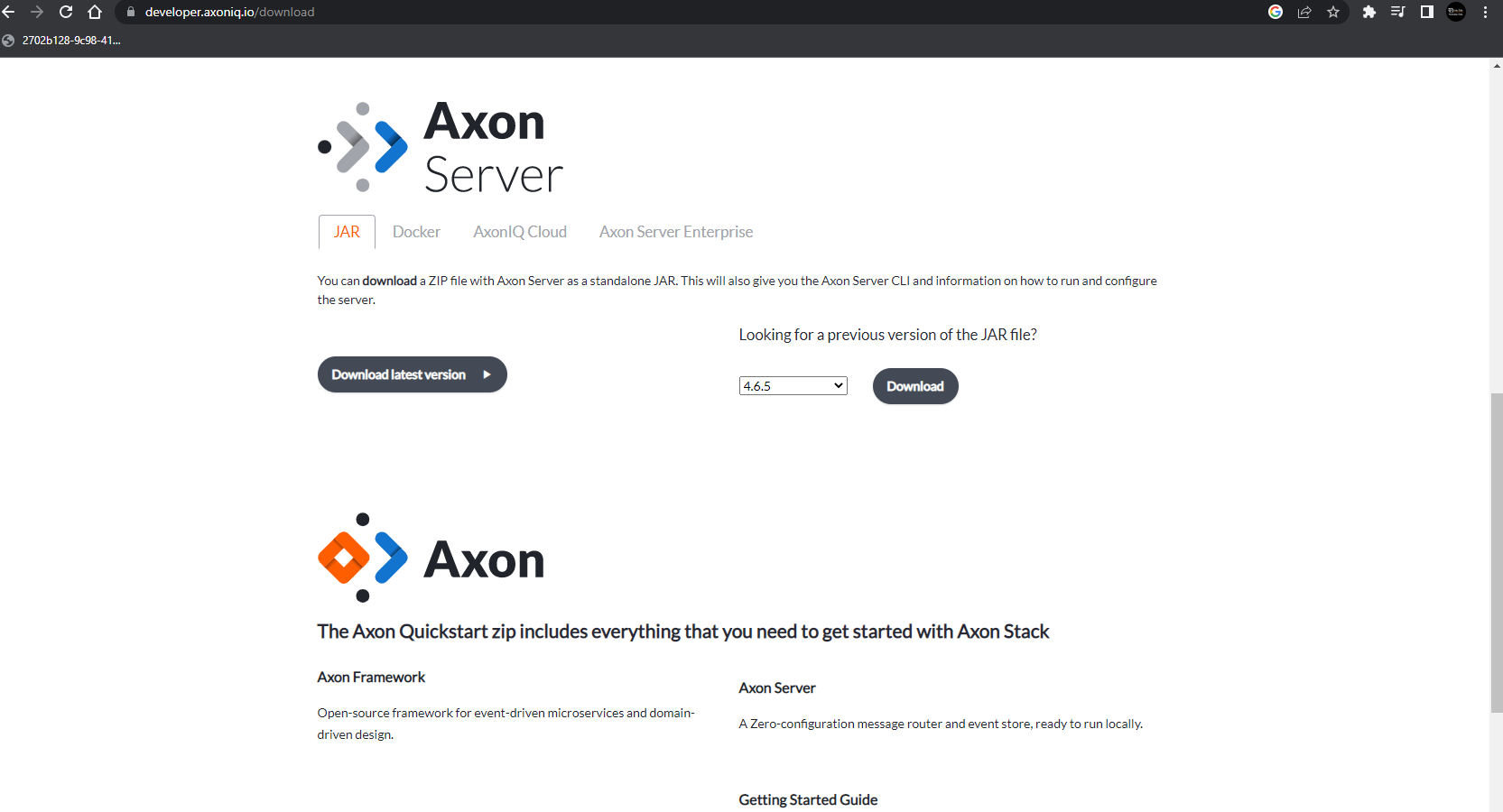
..

SAGA - OCHASTRATION

Instal axon server..

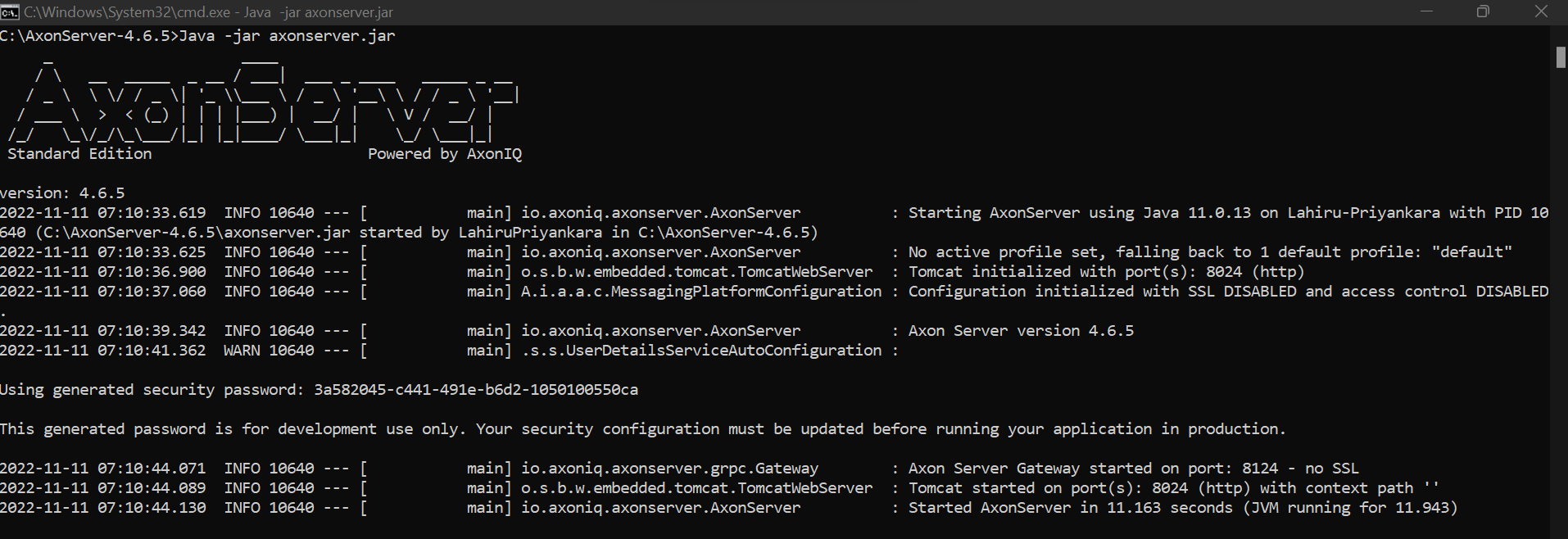
Got to

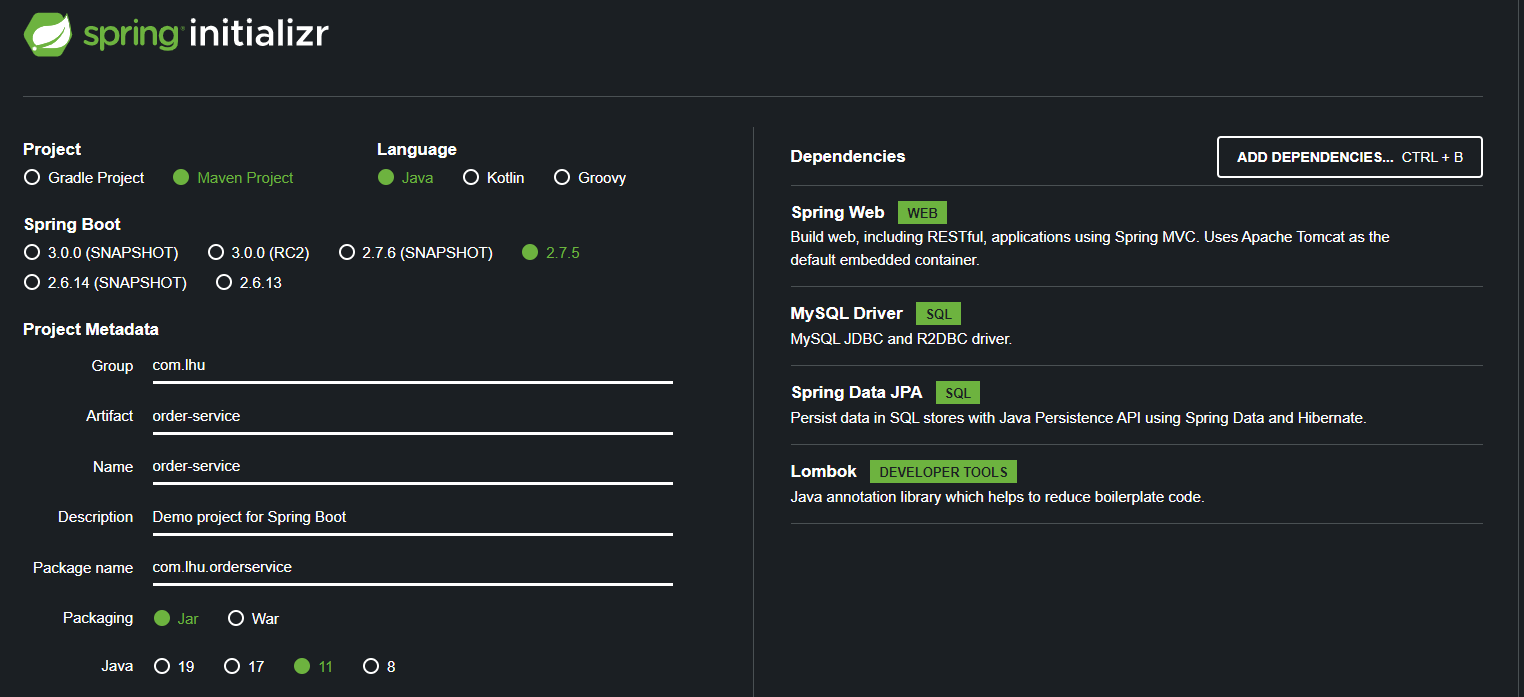
<https://developer.axoniq.io/download>



UnZip and go in to folder and run bellow command and Server will start on port:8124

Java -jar axonserver.jar





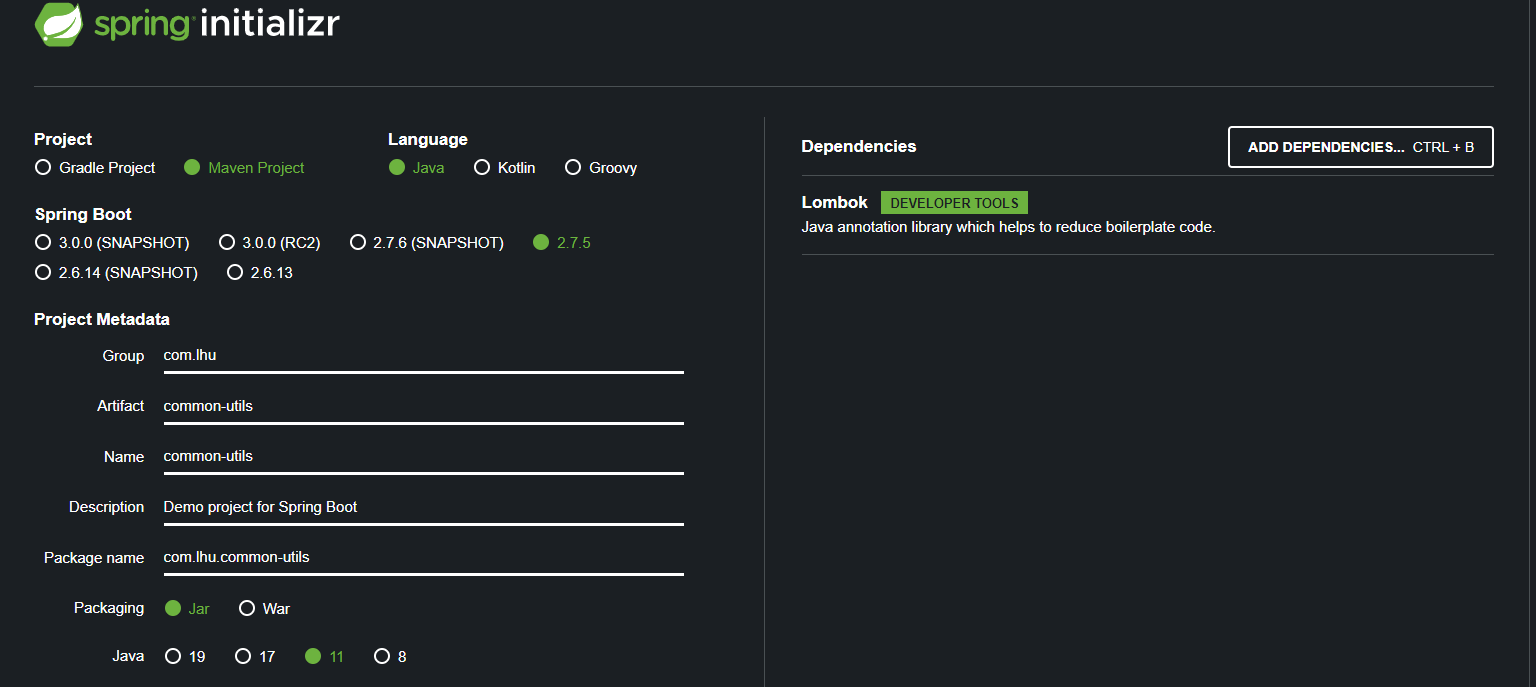
Instead of ablove dependencies…bellow dependencies also needed to work with axon

<!-- https://mvnrepository.com/artifact/org.axonframework/axon-spring-boot-starter -->  
<dependency>  
 <groupId>org.axonframework</groupId>  
 <artifactId>axon-spring-boot-starter</artifactId>  
 <version>4.5.3</version>  
</dependency>  
<!-- https://mvnrepository.com/artifact/com.google.guava/guava -->  
<dependency>  
 <groupId>com.google.guava</groupId>  
 <artifactId>guava</artifactId>  
 <version>31.0.1-jre</version>  
</dependency>

And Comman util can be added like

<dependency>  
 <groupId>com.dailycodebuffer</groupId>  
 <artifactId>CommonService</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
</dependency>

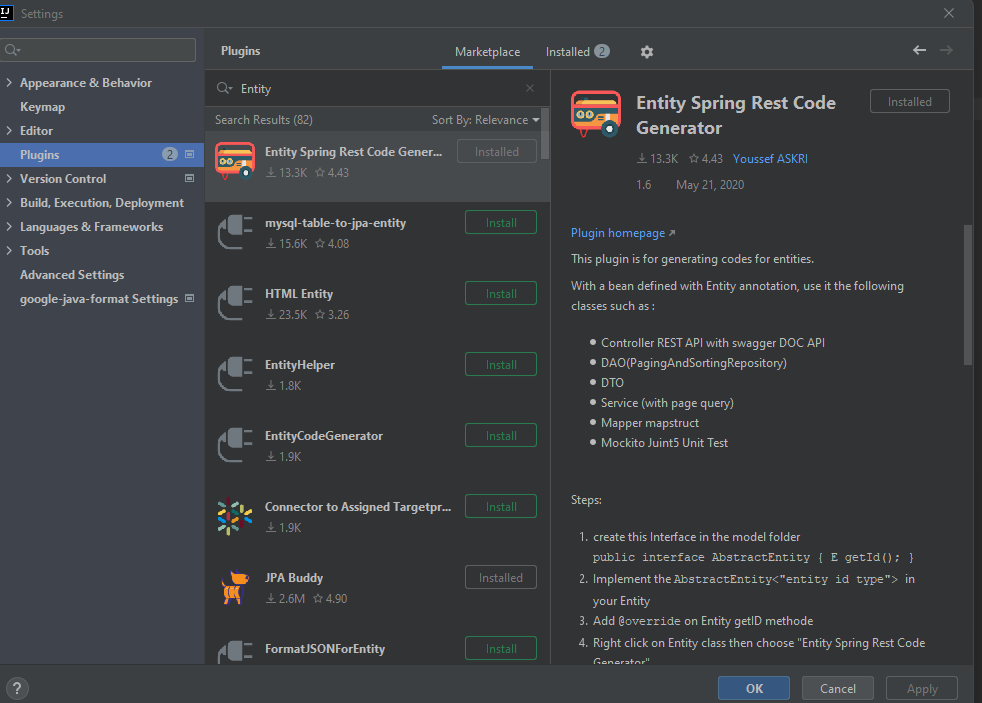
Common-utils

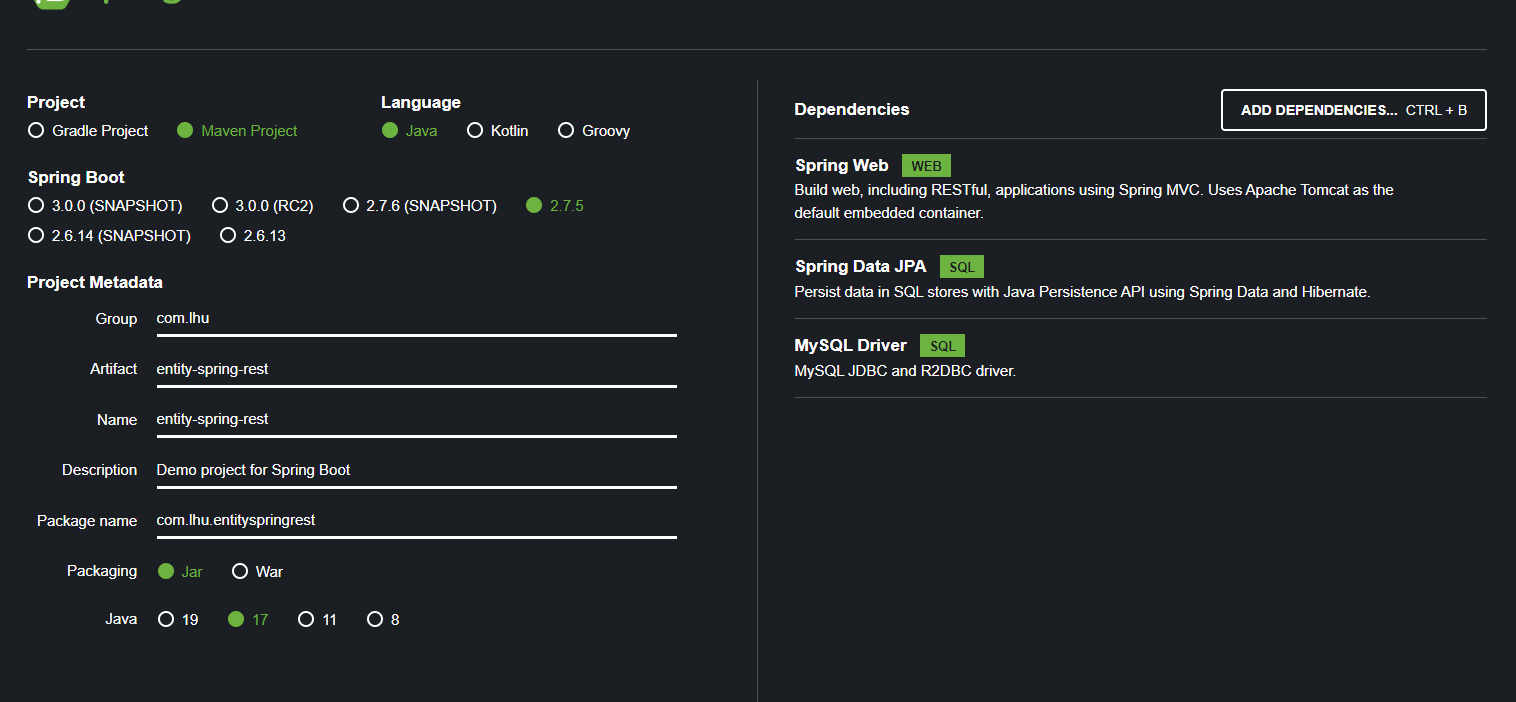


This also needed..

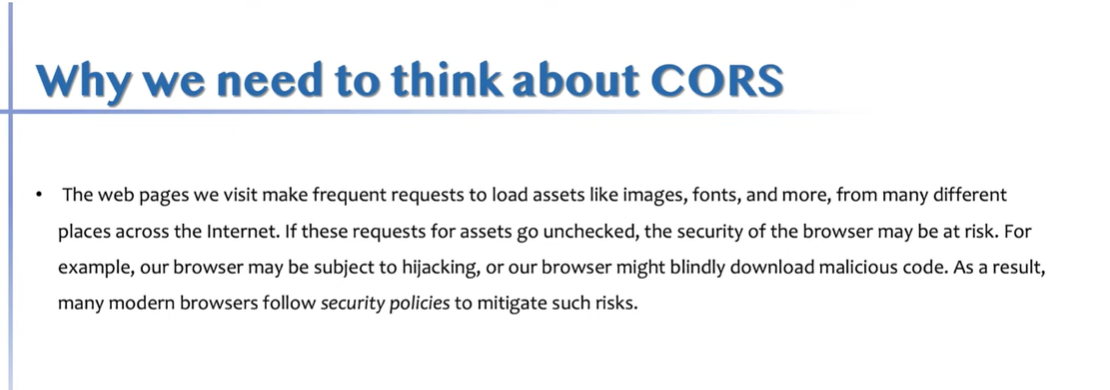
<!-- https://mvnrepository.com/artifact/org.axonframework/axon-spring-boot-starter -->  
<dependency>  
 <groupId>org.axonframework</groupId>  
 <artifactId>axon-spring-boot-starter</artifactId>  
 <version>4.5.3</version>  
</dependency>

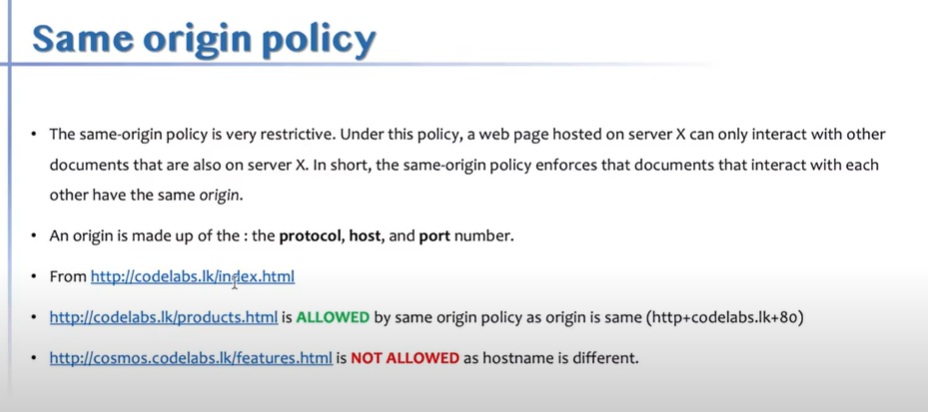
….………….

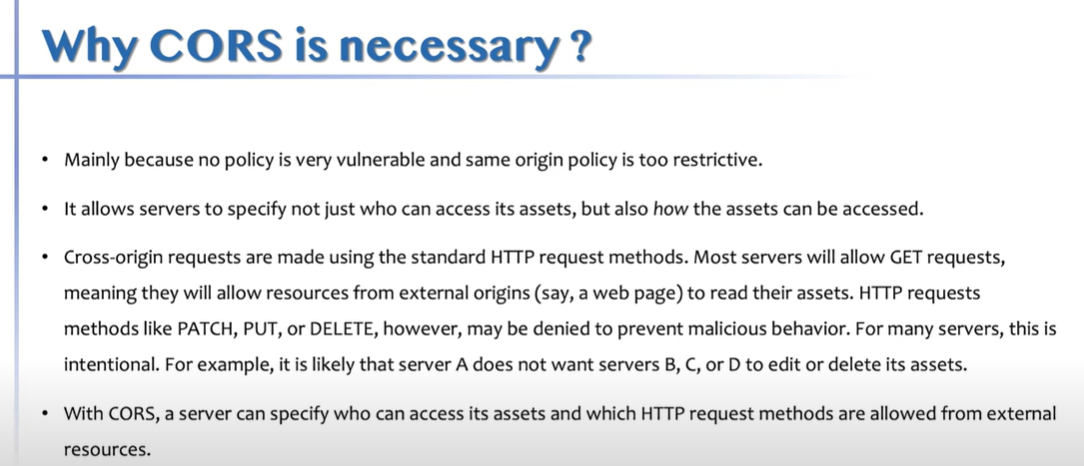


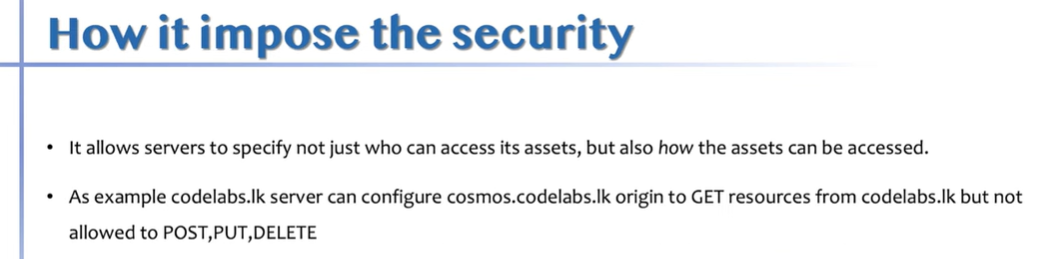


CORS origin problem..Corss origin resource sharing.









Cors validation happen in 3 different ways.

1. coming from real origin which we allowed.
2. Whether you are carrying allowed headers(Authorization header + content type header).
3. Method - GET,POST,PUT…

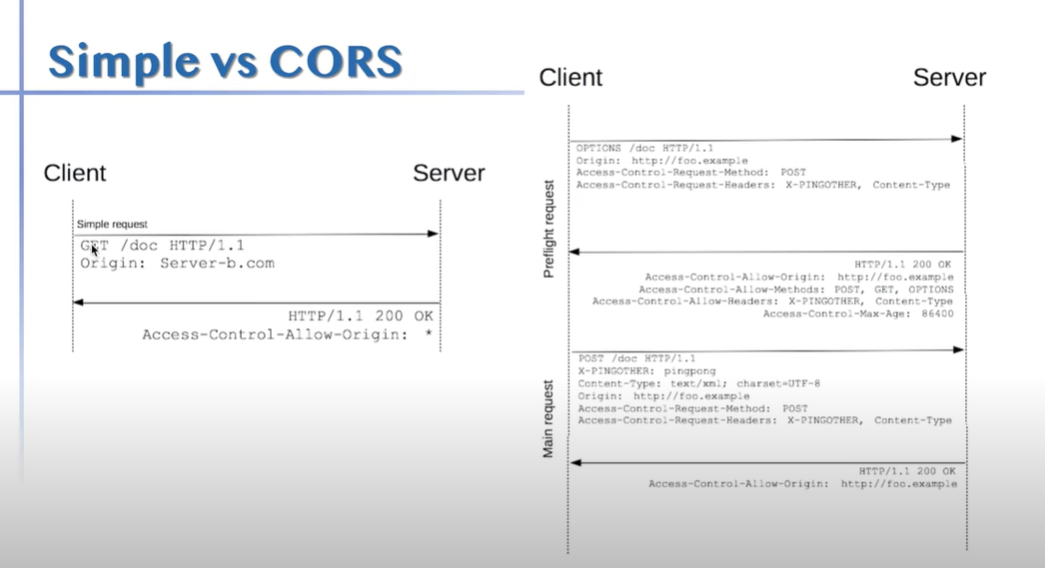
I am coming from this origin,I am carrying this much of messages and can I fly - Pre flight request.

Postman - insominia.

Cors only experience in browser.

Sent pre-flight request.If server is cors enagled, server replies back with I am allowwiing these origin,these headers ,these methods..If server is not cors enagled,then server not respond to pre flight(Optional call) request.then borwser says an error says cors fail.

Cors fail - You are not allowed cors at all OR You are allowed cors at all and send from different domain or different method or allow 2 headers you are trying to send 3 headers.



Browser interceptors to bypass this.

**GraphQL**

**Tutorial -**

**Channel :**

**<https://www.youtube.com/@DanVega>**

**Video:**

**<https://www.youtube.com/watch?v=TVk2fMEezO4&list=RDCMUCc98QQw1D-y38wg6mO3w4MQ&index=3&ab_channel=DanVega>**

New API standard that was invented and open source by Facebook.

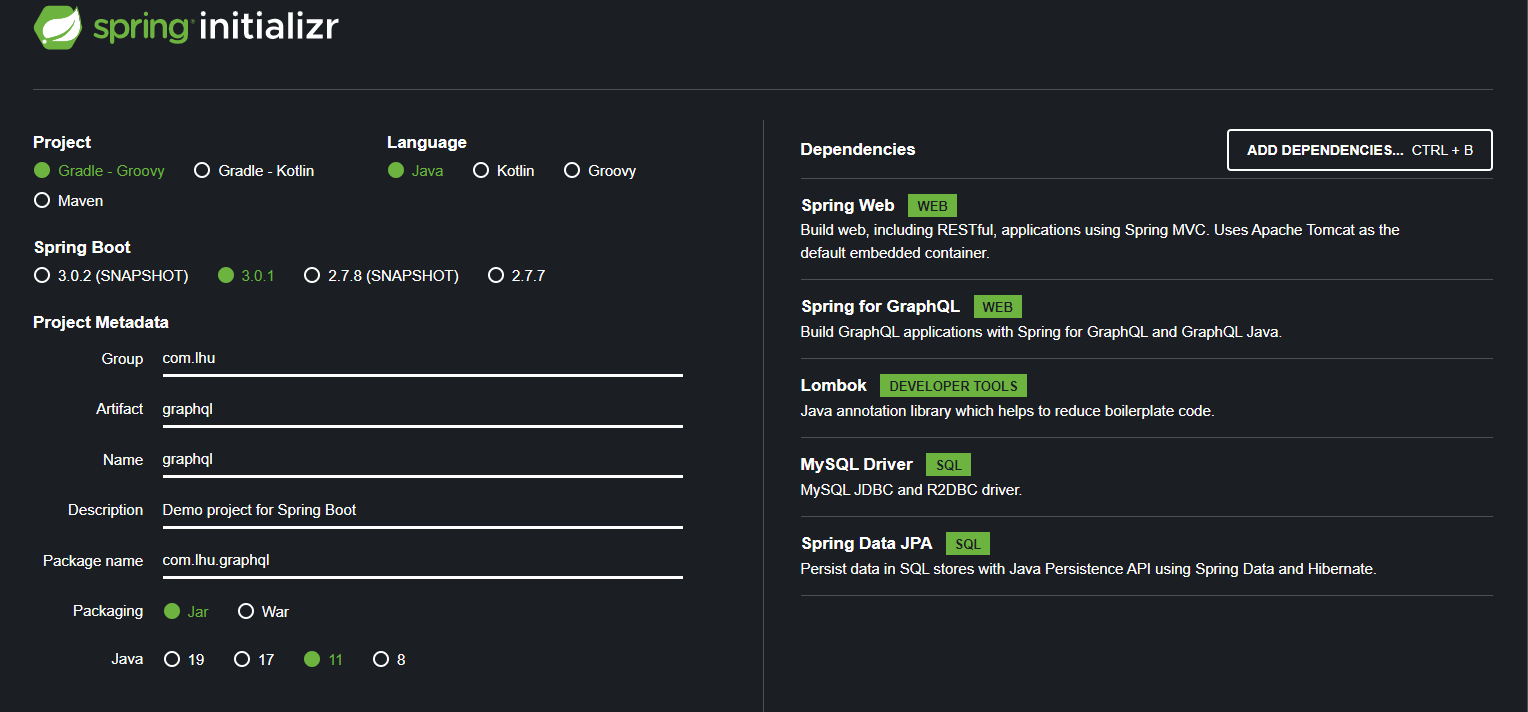
Enable declarative data fetching.

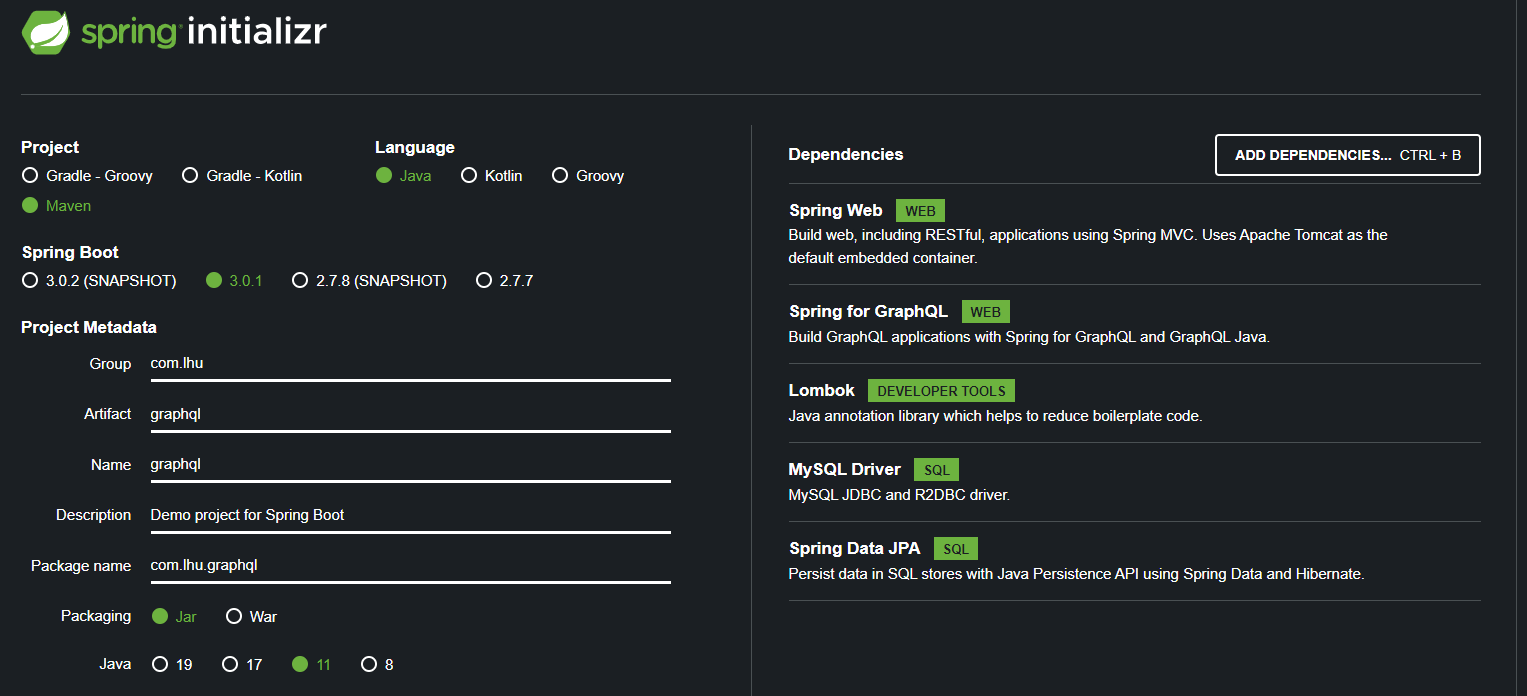
CKient ta puluwan decide kranna mona wage datada API eken fetch karaganan one kiyla..namuth rest walanm API eken dena tika apita gana wenwa.Namuth mekedi client one data tika witarak gannwa.

GraphQL sever exposes single endpoint and responds to queries.

Multiplae EndPoint hadala fixed data structure ekak tiyna data privide karnawa wada eka end point ekak hadala එක multiple consumers ලට ඔහුන්ට අවශ්‍ය data පමණක් access kara ගන්න පුළුවන් විදිහට provide කරන technology එකක් වේ.

https://www.howtographql.com/basics/0-introduction/





Query - to retrieve data.

Mutation - to changed the data (Create,Update,delete)

Subscription - Which allows allows you to create a connection read data and keep that connection for when data changes(15.00).

<http://localhost:2020/graphiql?path=/graphql>

# Welcome to GraphiQL

#

# GraphiQL is an in-browser tool for writing, validating, and

# testing GraphQL queries.

#

# Type queries into this side of the screen, and you will see intelligent

# typeaheads aware of the current GraphQL type schema and live syntax and

# validation errors highlighted within the text.

#

# GraphQL queries typically start with a "{" character. Lines that start

# with a # are ignored.

#

# An example GraphQL query might look like:

#

#     {

#       field(arg: "value") {

#         subField

#       }

#     }

#

# Keyboard shortcuts:

#

#   Prettify query:  Shift-Ctrl-P (or press the prettify button)

#

#  Merge fragments:  Shift-Ctrl-M (or press the merge button)

#

#        Run Query:  Ctrl-Enter (or press the play button)

#

#    Auto Complete:  Ctrl-Space (or just start typing)

#

query{

  allStudent{

    id

    firstName

    lastName

    age

    studentClass{

      id

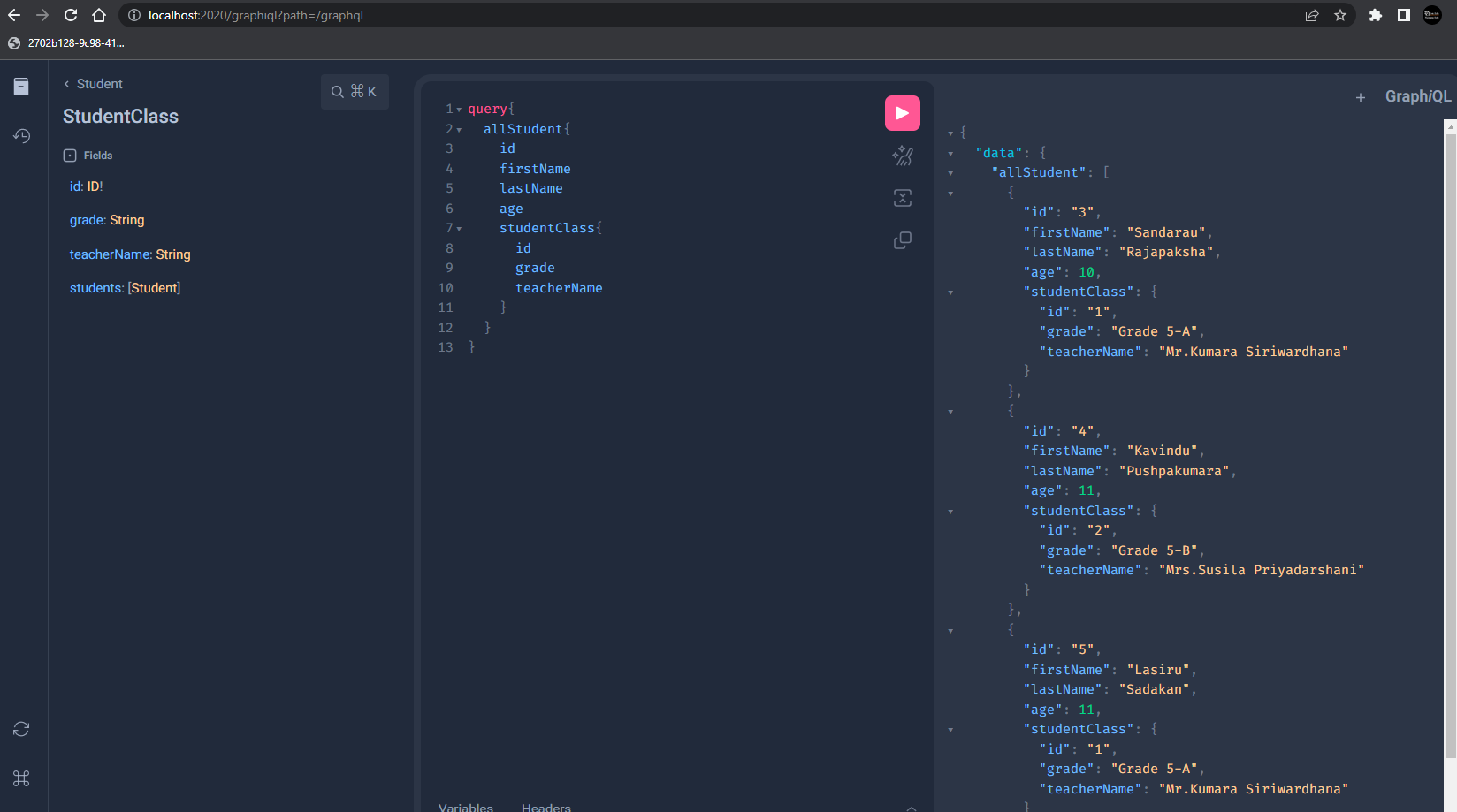
      grade

      teacherName

    }

  }

}



query{

  findStudent(id:4){

    id

    firstName

    lastName

    age

    studentClass{

      id

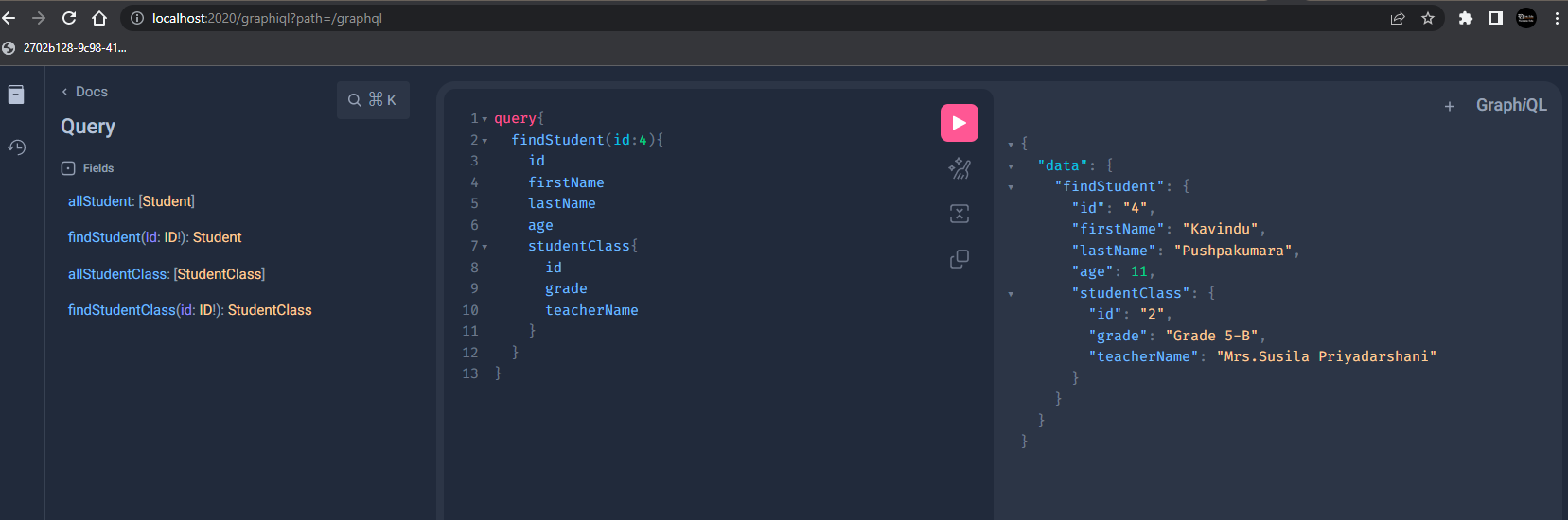
      grade

      teacherName

    }

  }

}



query{

  allStudentClass{

    id

    grade

    teacherName

    students{

      id

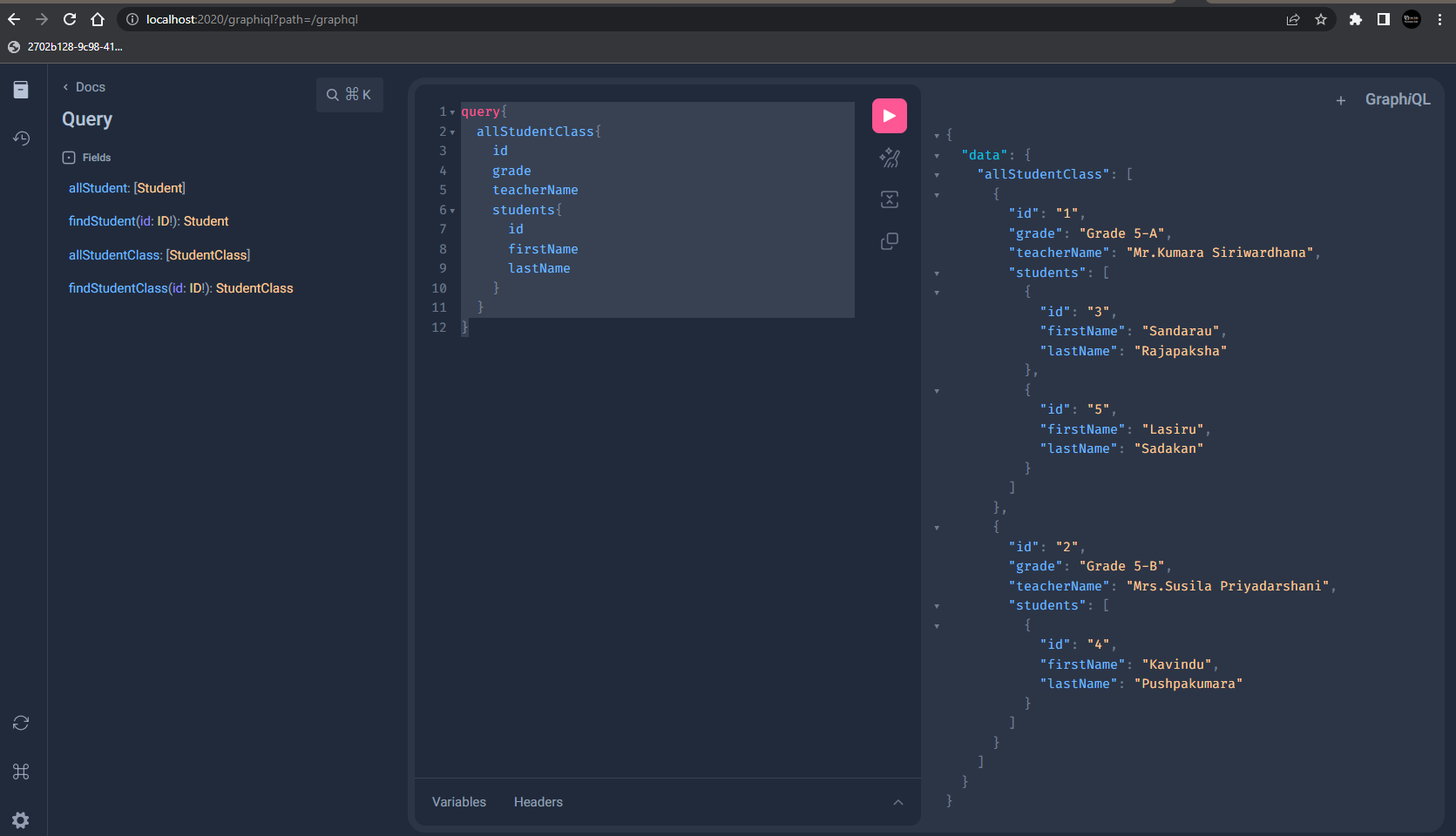
      firstName

      lastName

    }

  }

}



query{

  findStudentClass(id:2){

    id

    grade

    teacherName

    students{

      id

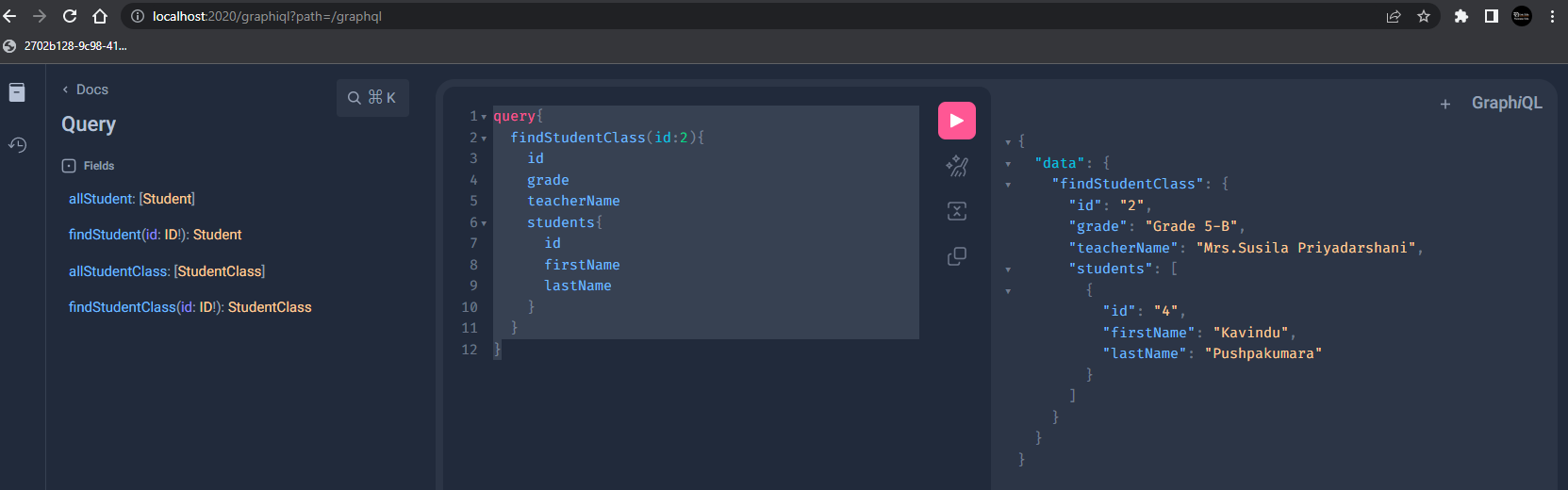
      firstName

      lastName

    }

  }

}



mutation{

  createStudentClass(grade:"Grade 6-A",teacherName:"Mr.fernando"){

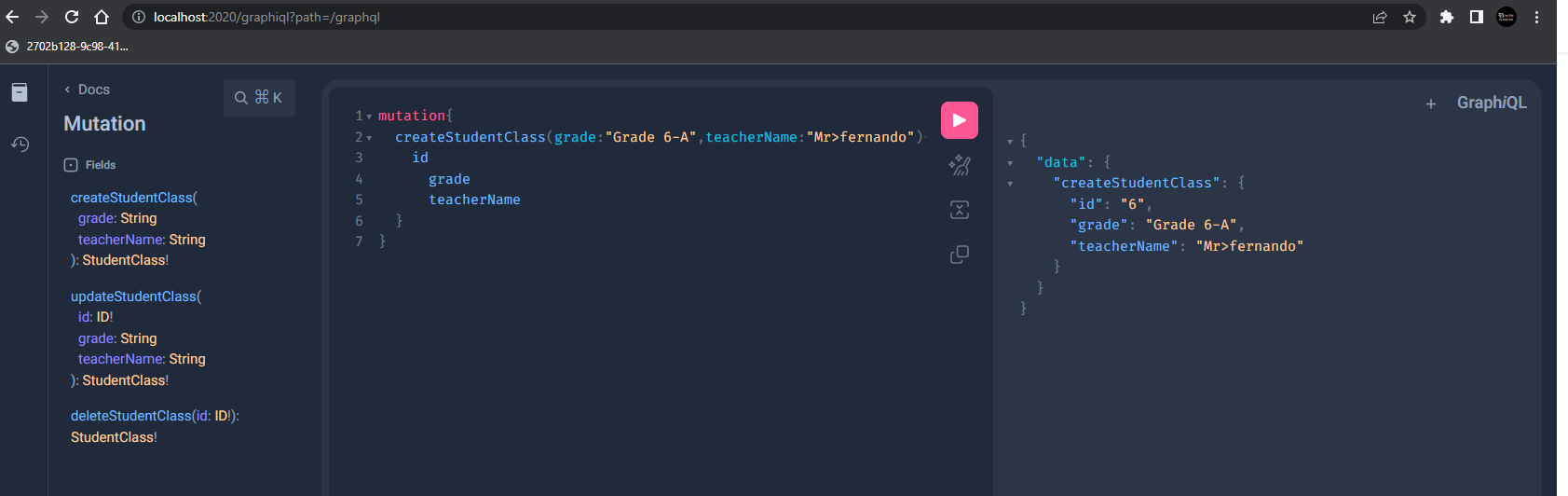
    id

      grade

      teacherName

  }

}



mutation{

  updateStudentClass(id:6,grade:"Grade 6-A",teacherName:"Mr.fernando"){

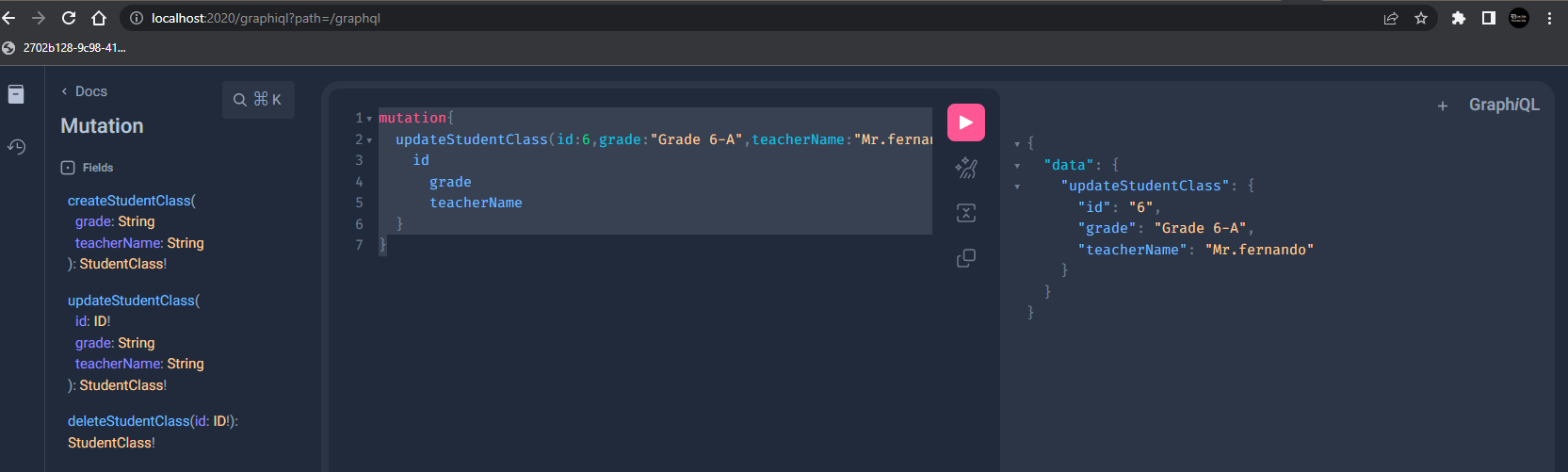
    id

      grade

      teacherName

  }

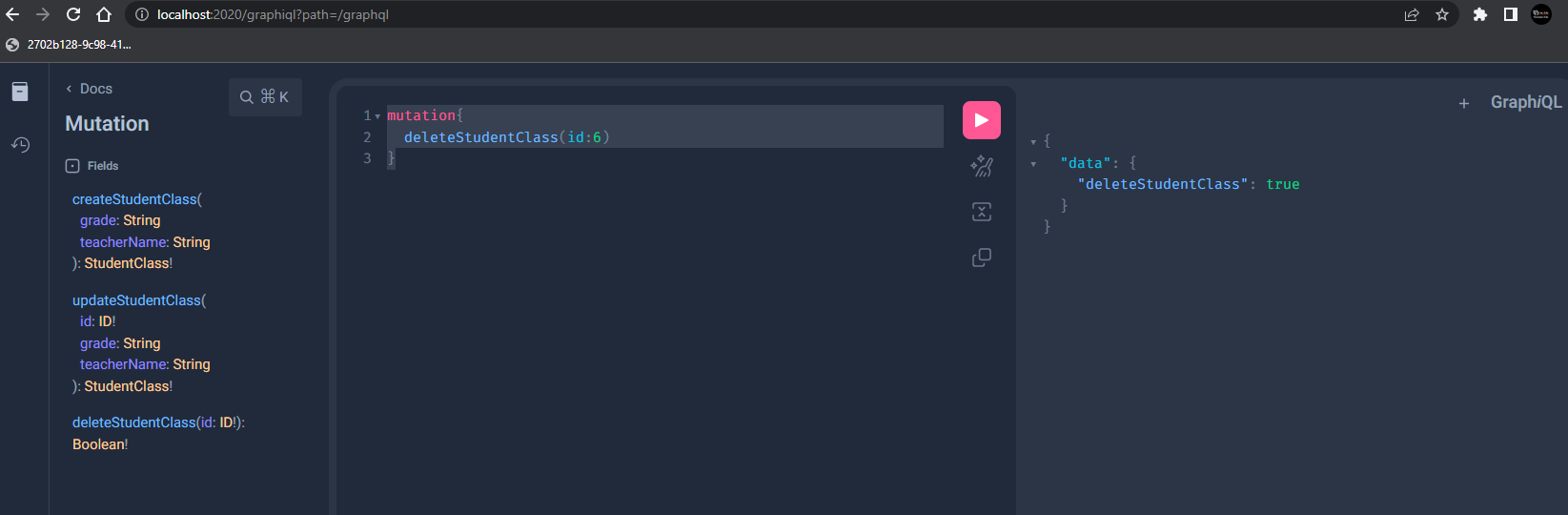
}



mutation{

  deleteStudentClass(id:6)

}



query{

  allStudentClassPage(page:0,size:2){

     id

    grade

    teacherName

    students{

      id

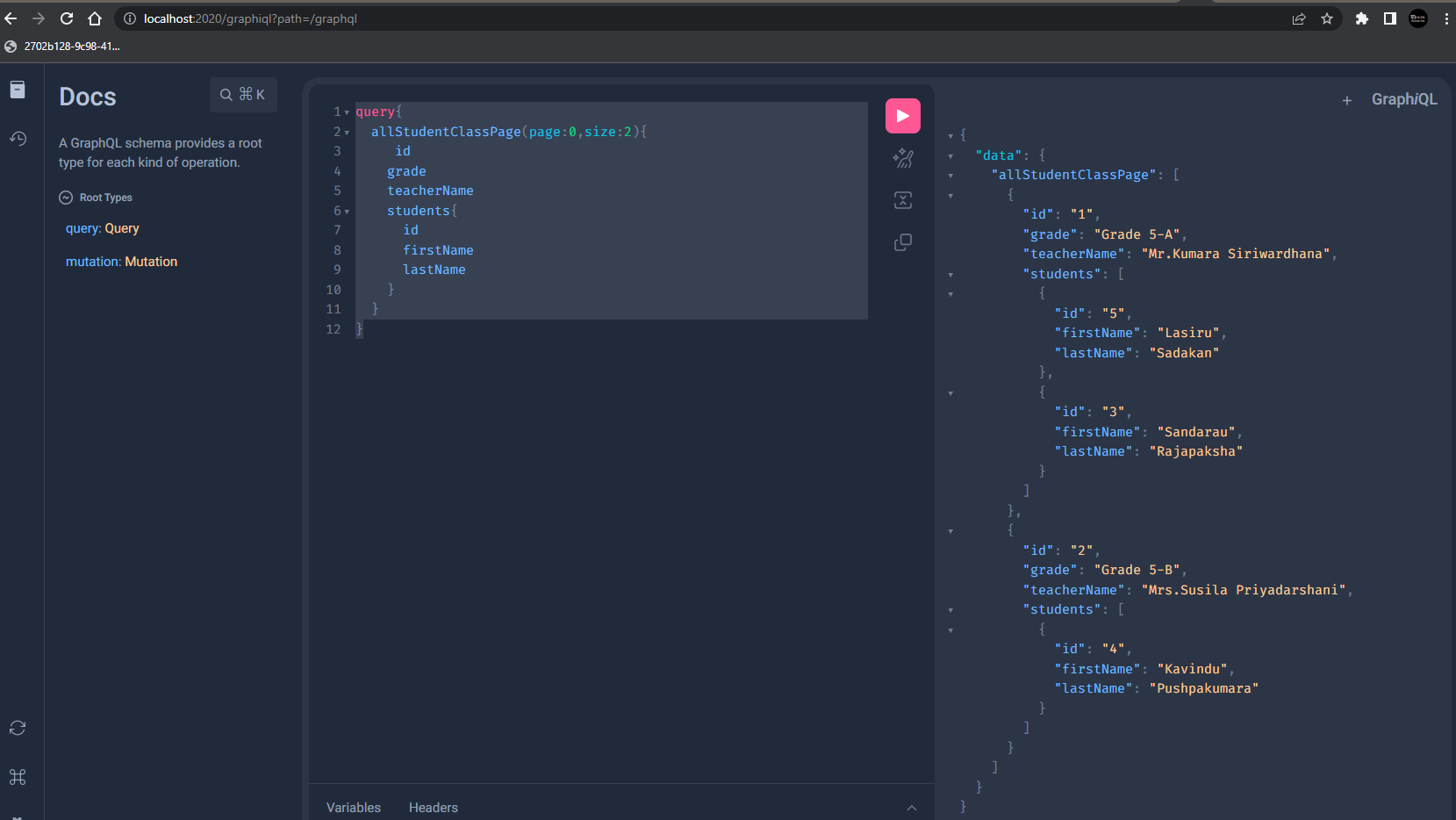
      firstName

      lastName

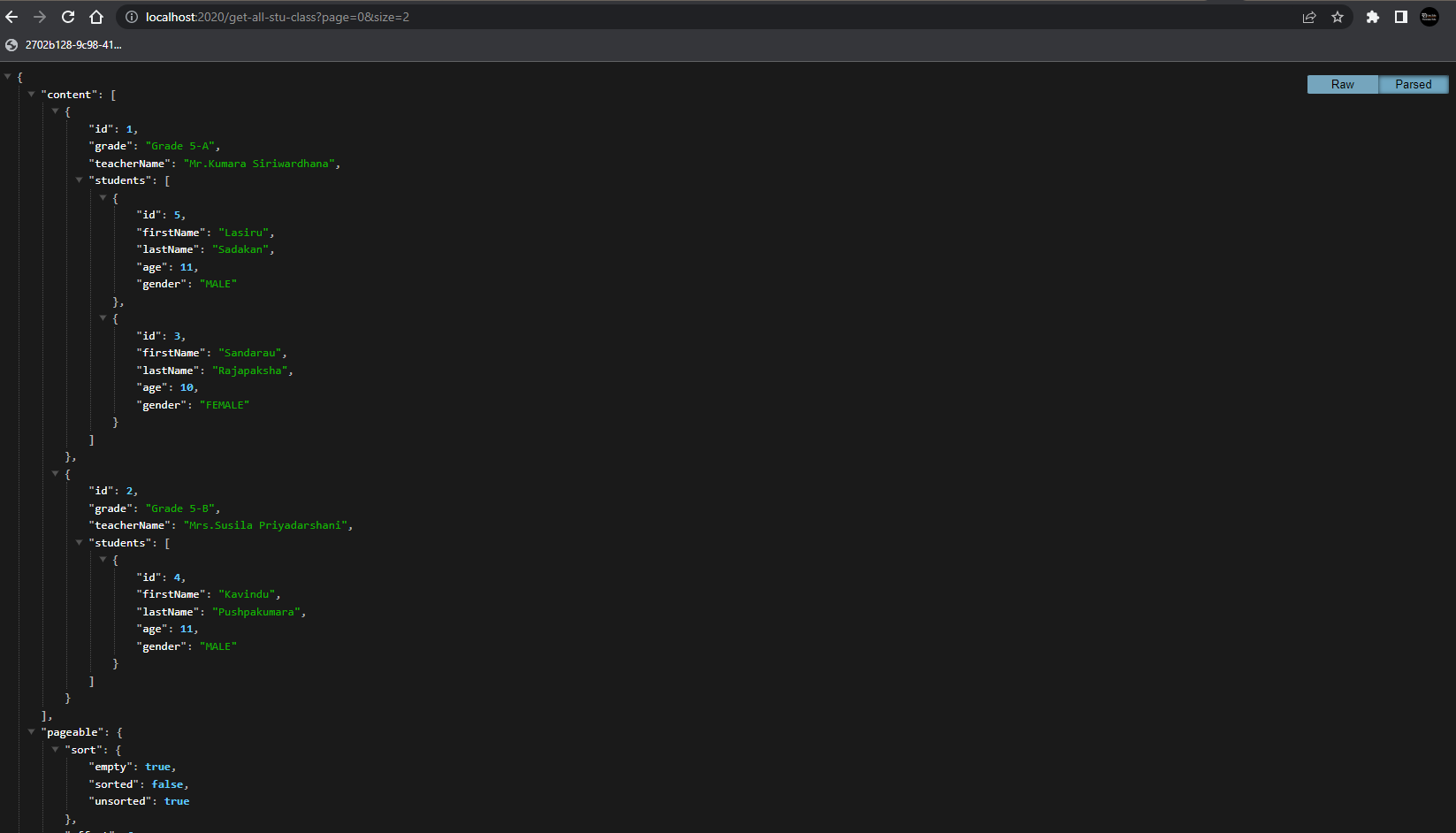
    }

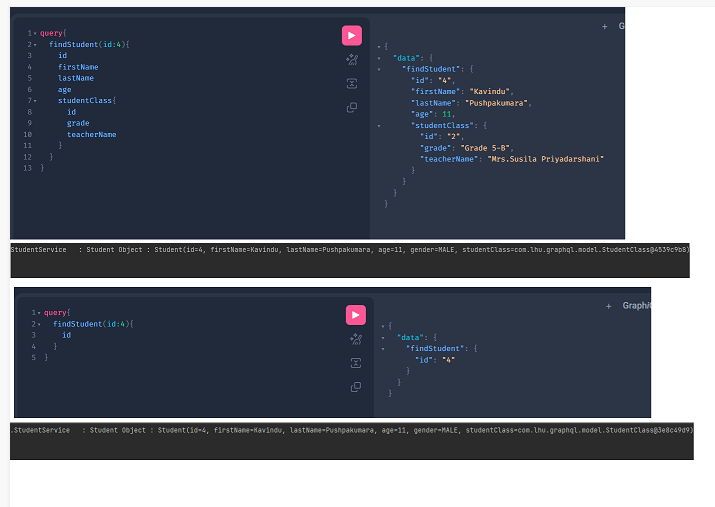
  }

}



<http://localhost:2020/get-all-stu-class?page=0&size=2>





Test using Postman

