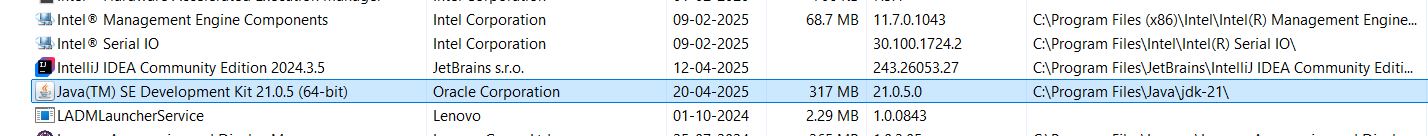
**Day1: Download, Install and configure**

Download -> Install -> configure path in system environment variables

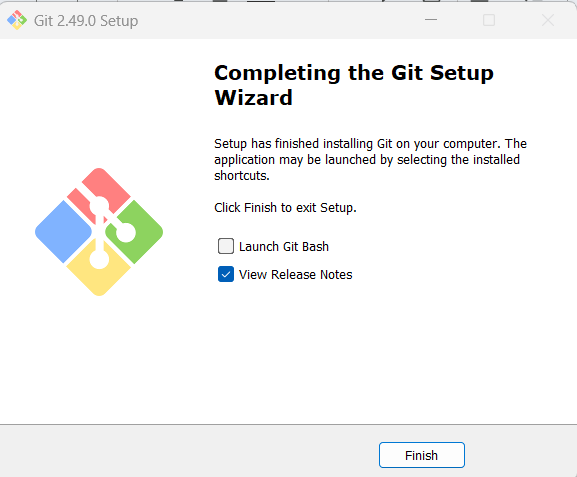
* Java

<https://download.oracle.com/java/21/archive/jdk-21.0.5_windows-x64_bin.exe>



* Git

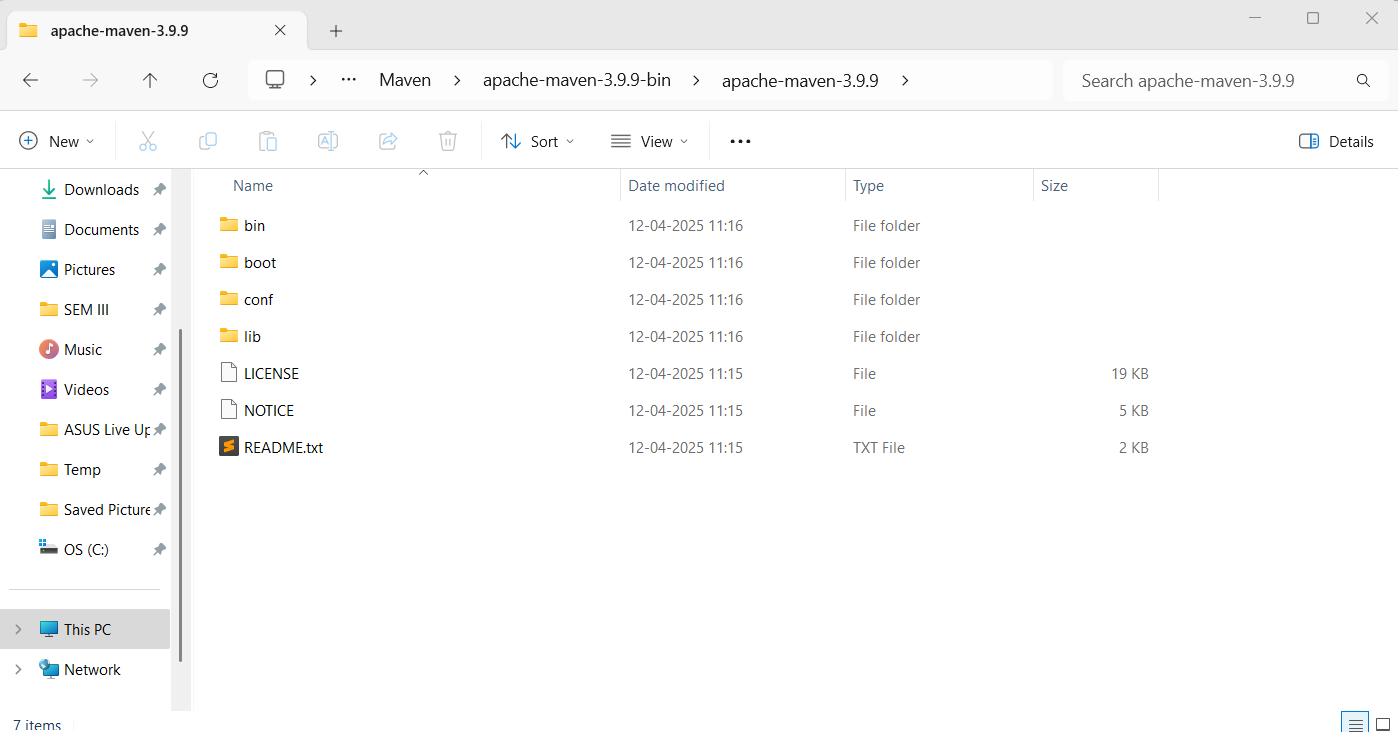
<https://github.com/git-for-windows/git/releases/download/v2.49.0.windows.1/Git-2.49.0-64-bit.exe>



Download --> configure path in system environment variables

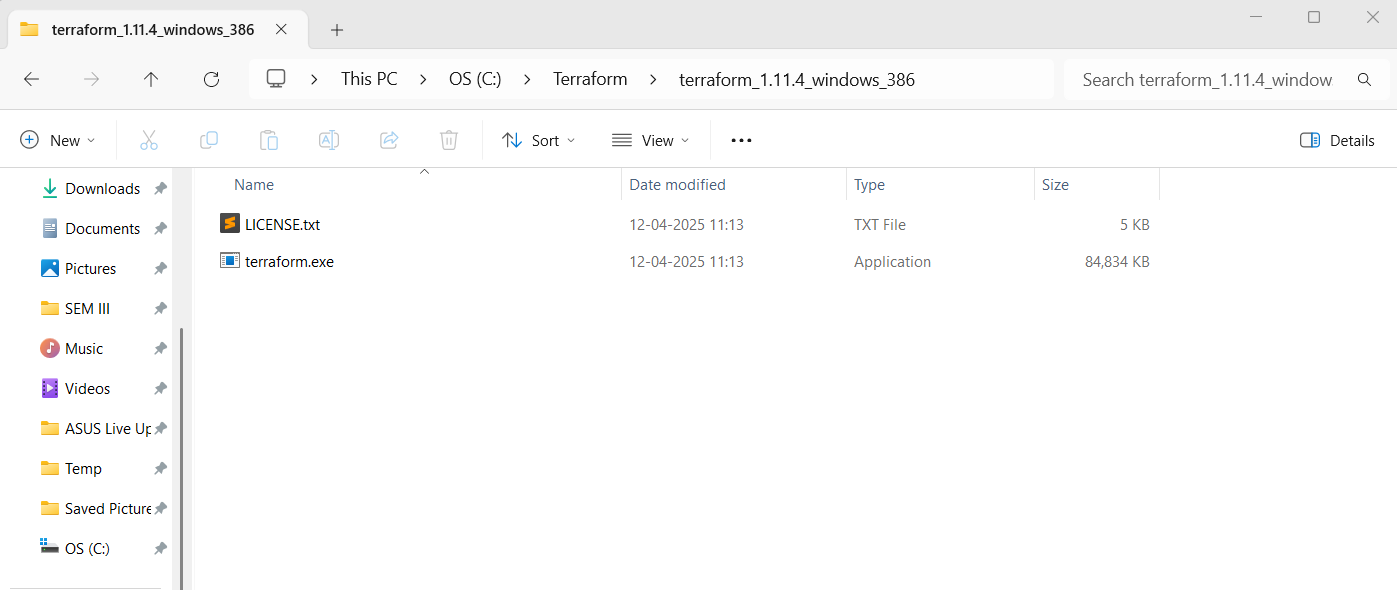
* Maven

<https://dlcdn.apache.org/maven/maven-3/3.9.9/binaries/apache-maven-3.9.9-bin.zip>



* Terraform

<https://releases.hashicorp.com/terraform/1.11.4/terraform_1.11.4_windows_386.zip>





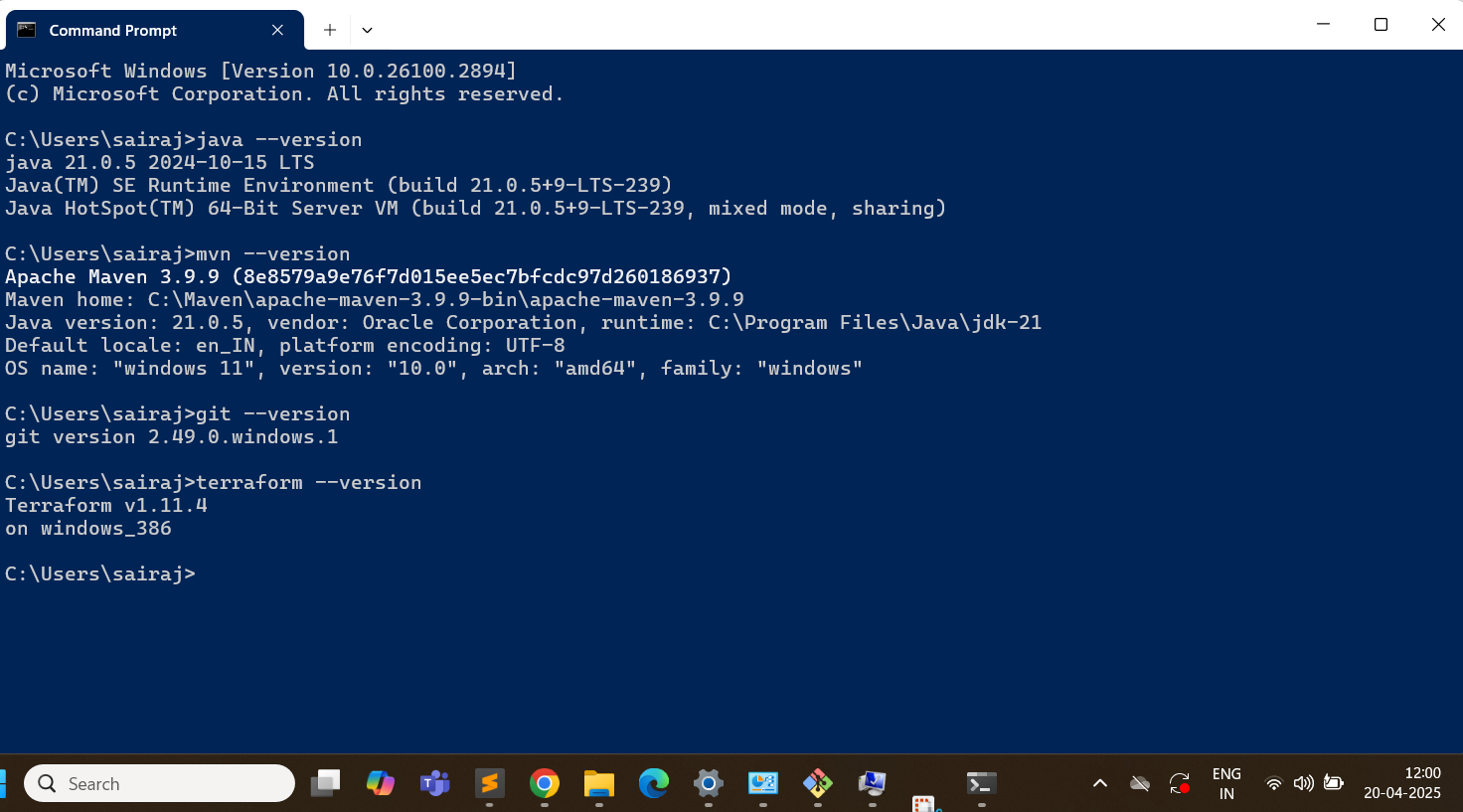
**Validate the installation:**

>java --version

>mvn --version

>git --version

>terraform --version



**Install IntelliJ IDEA & VSCode:**

IntelliJ IDEA Community Edition

<https://www.jetbrains.com/idea/download/?section=windows>

VSCode

<https://code.visualstudio.com/download>

**Create accounts:**

* AWS account
* Github Account
* Docker hub account

**Day3: Creating our first microservices**

Go to <https://start.spring.io/>

Select:

Project: Maven

Language: Java,

Springboot version: 3.310

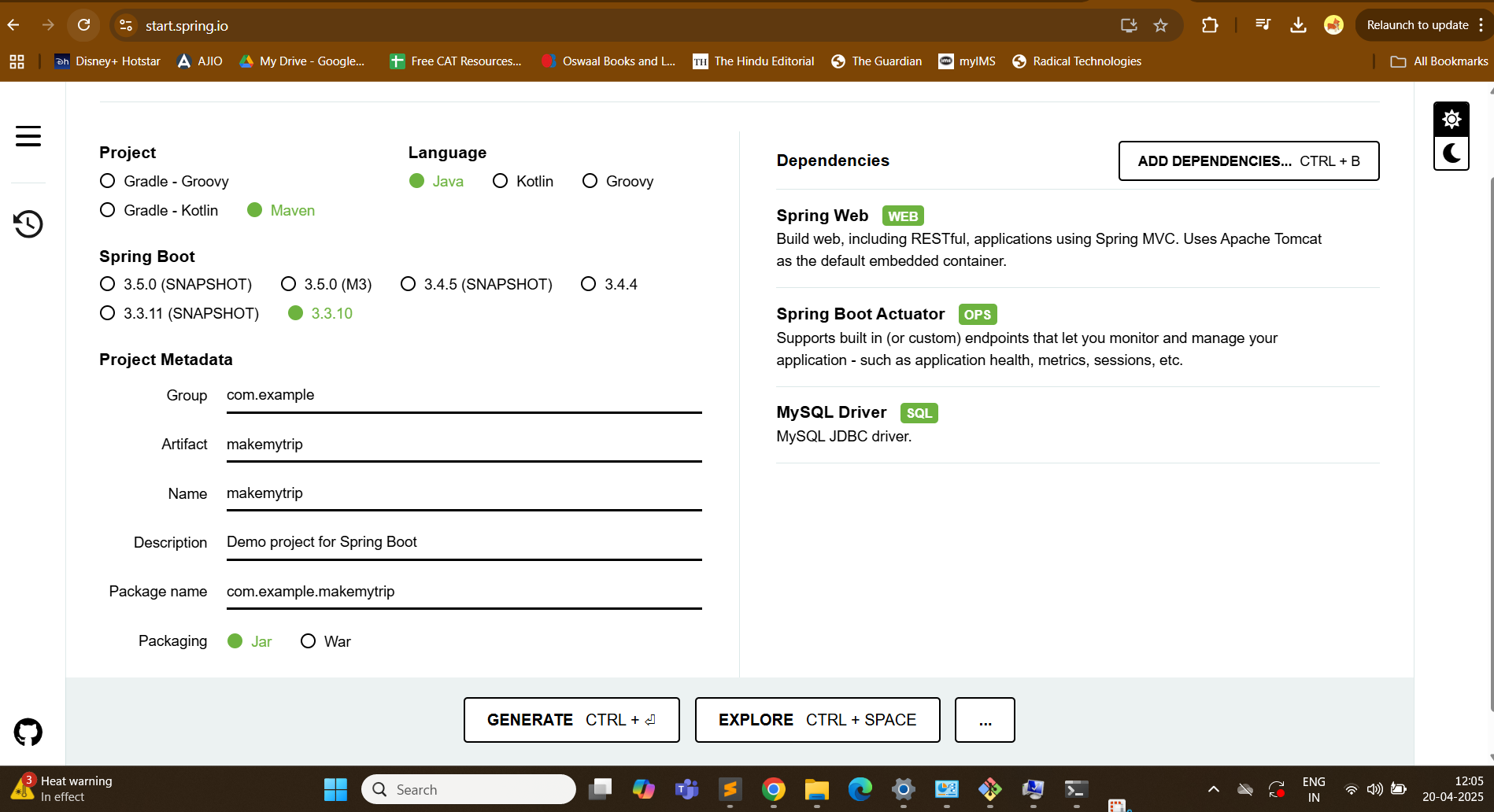
Artifact name: makemytrip

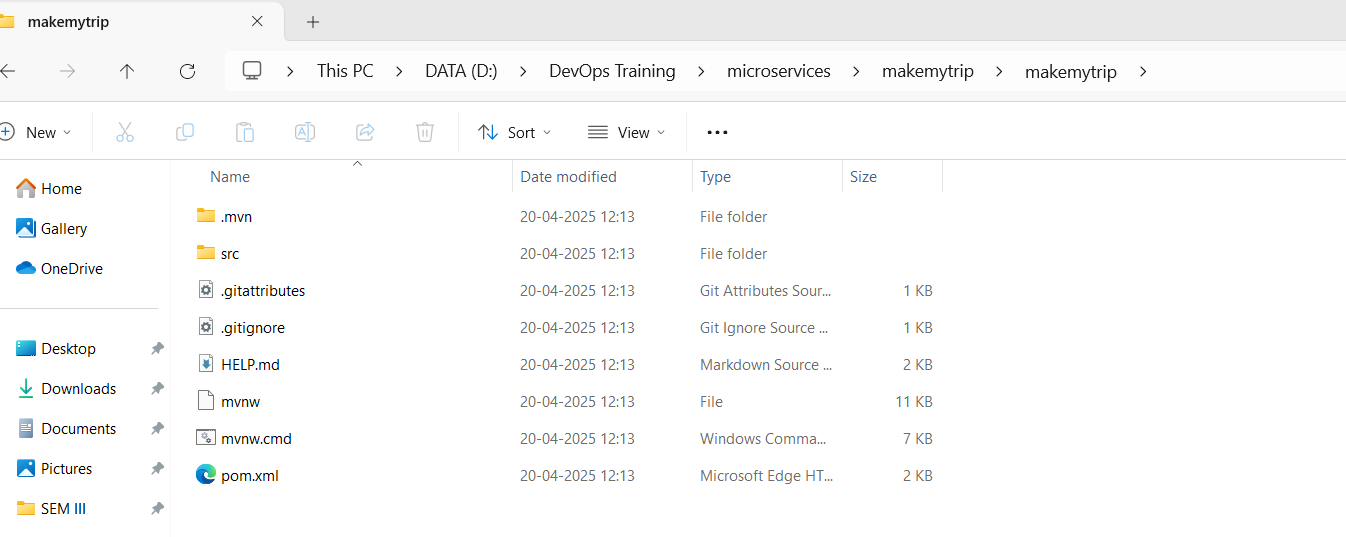
Packaging: Jar

Add the dependencies: Spring Web, String Boot Actuator

Generate the zip file

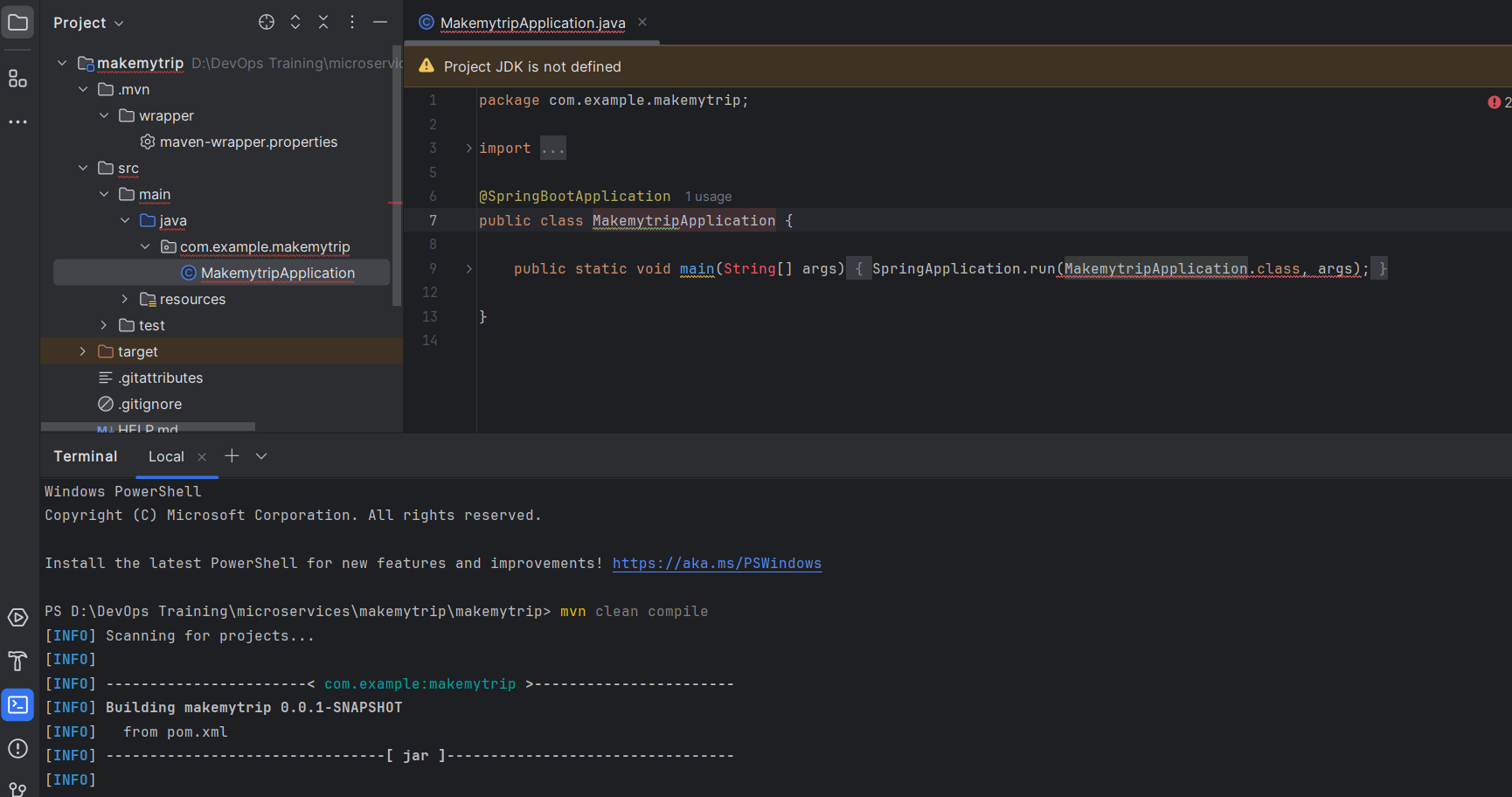
Extract the zip file



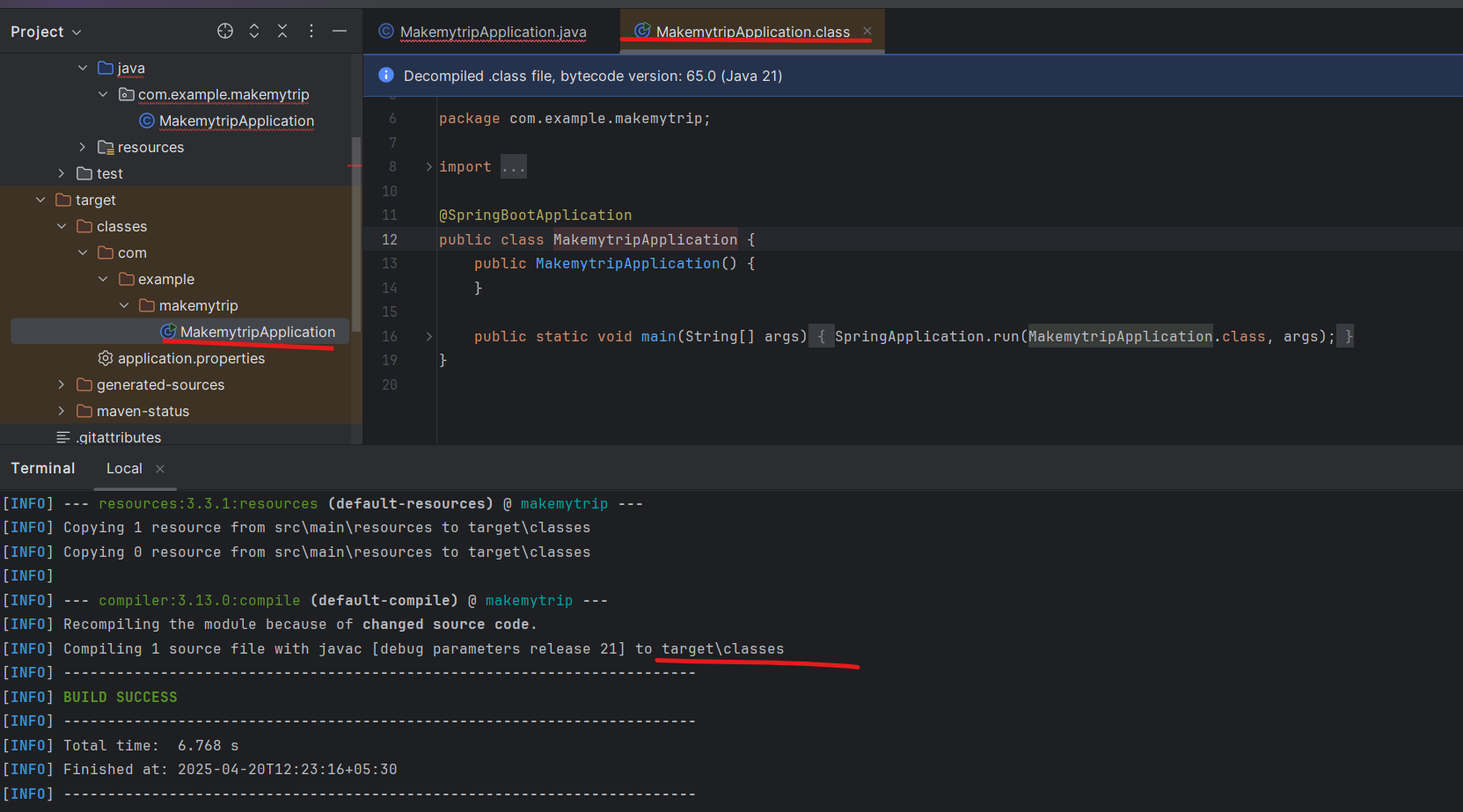


> mvn clean compile

(Creates .class file)



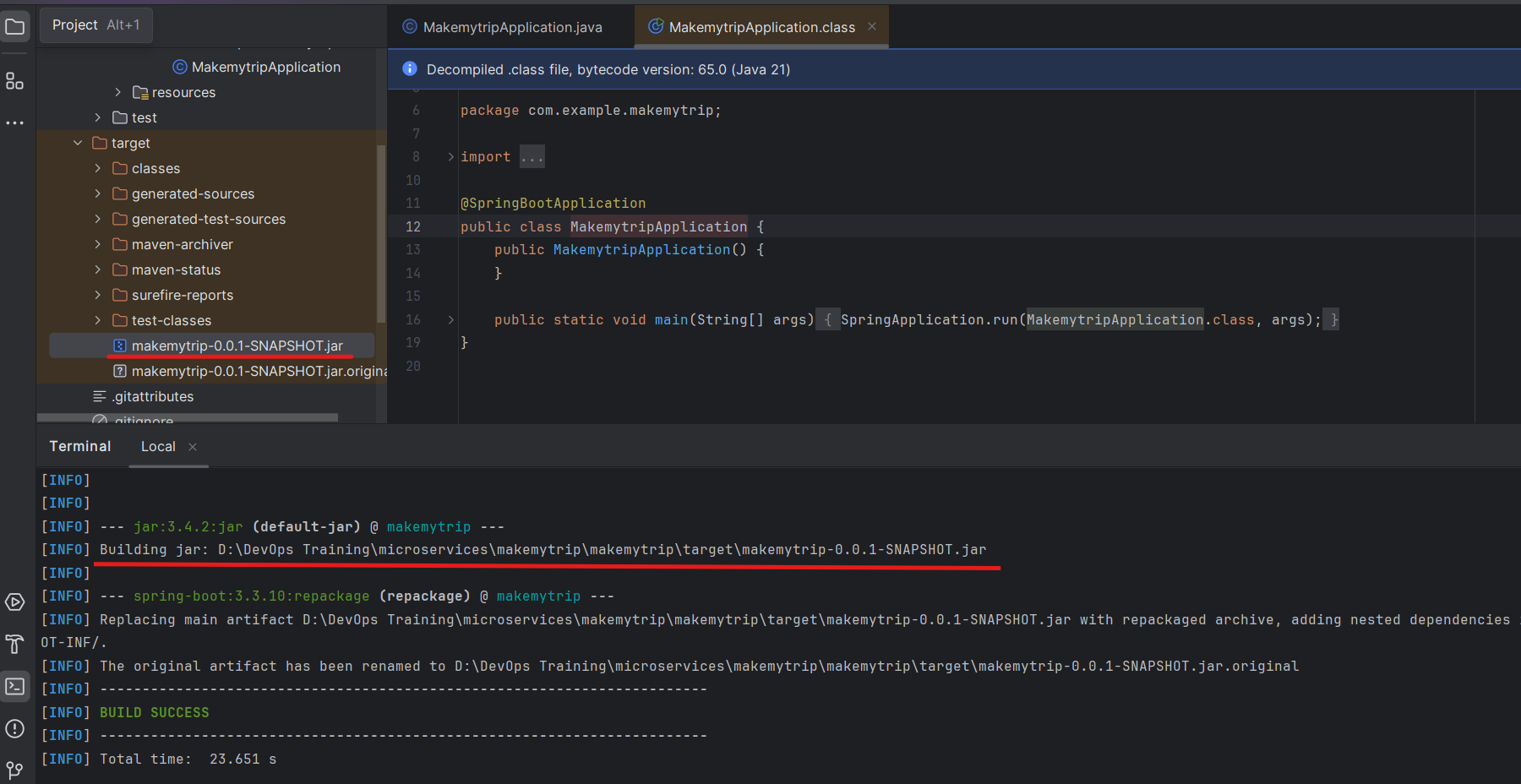
Location of .class file



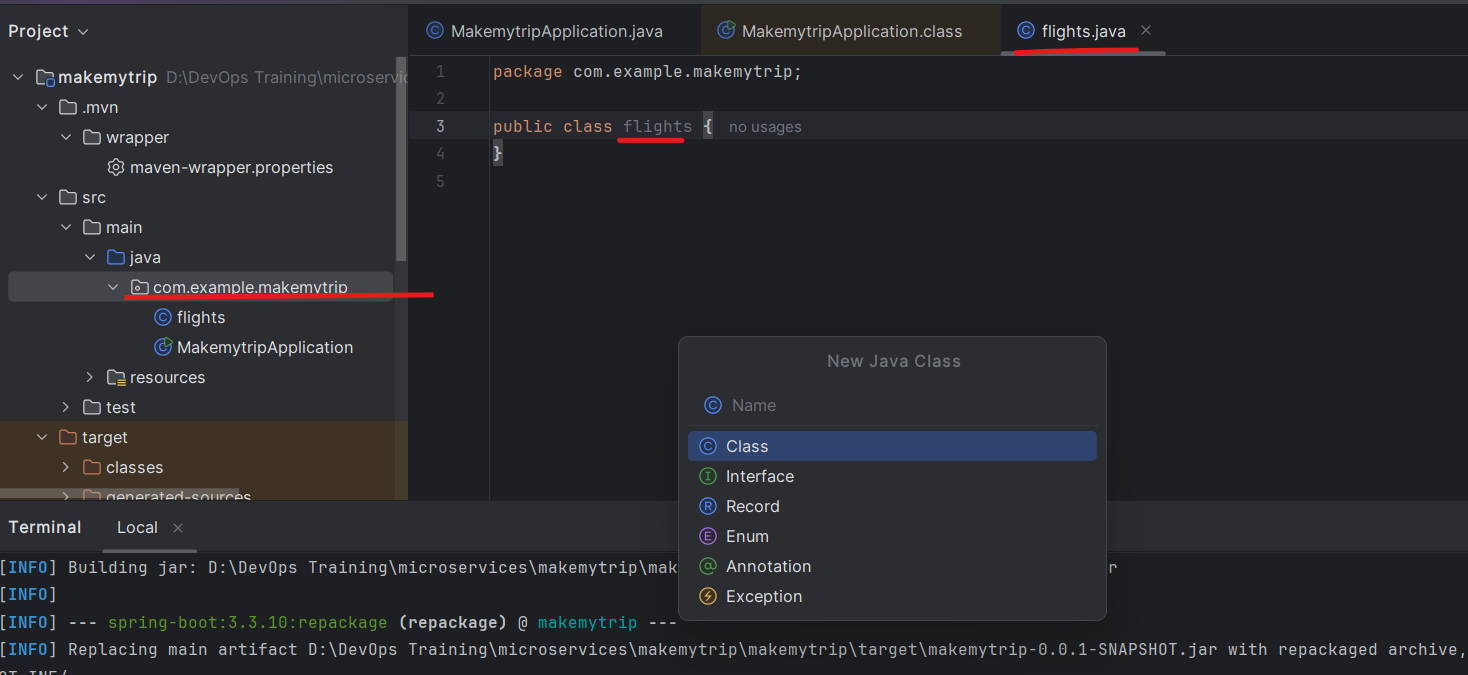
Now package:

> mvn clean package

Default package is .jar



Now add a java fileRight click on java skeleton > new > java class > flights



Add following to to this file flights.java:

package com.example.makemytrip;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class flights {

@GetMapping("/flights")

public String getData() {return "Book your FLIGHTS from Kolkata to Pune!";}

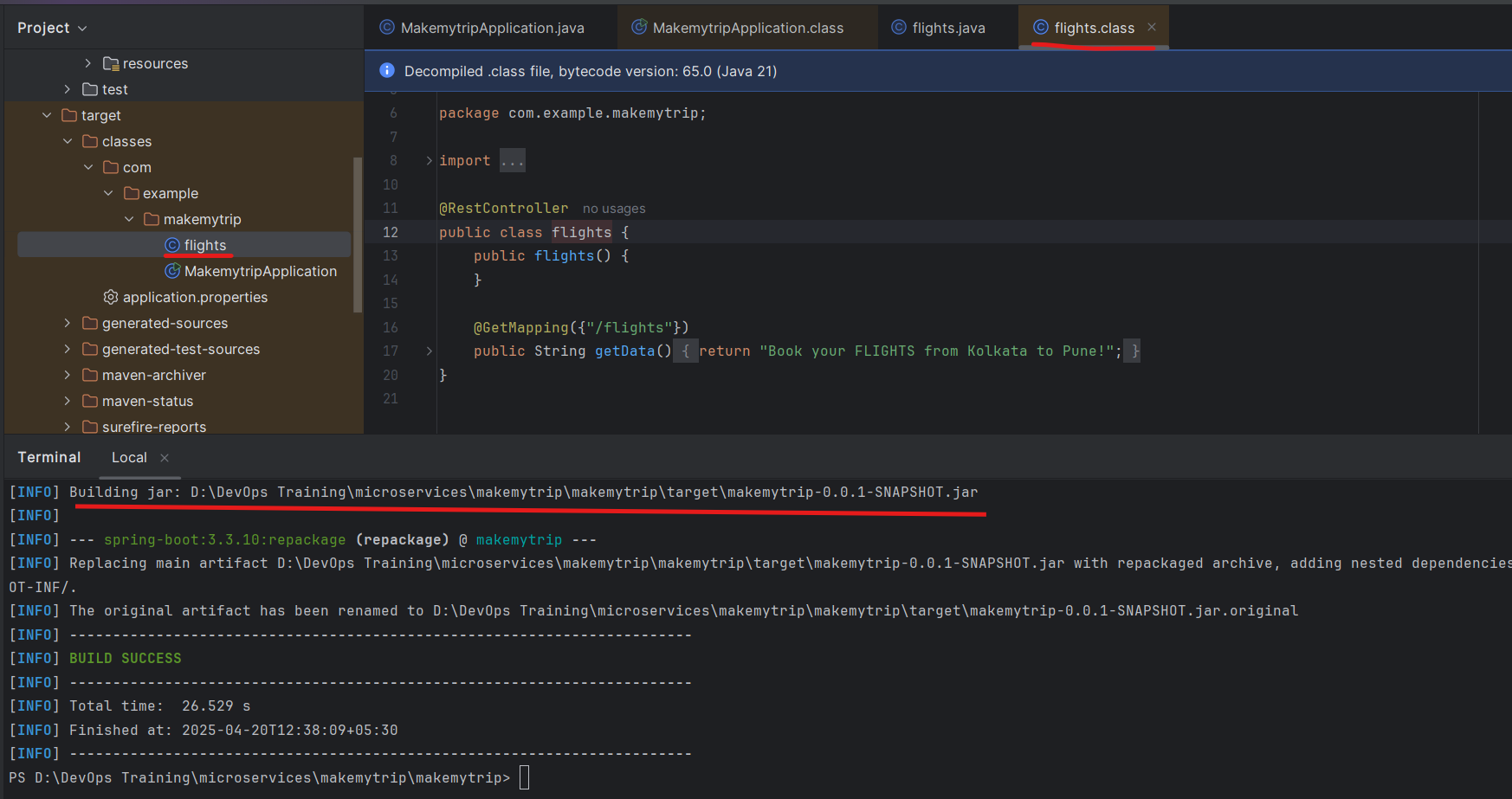
}

After ading new code - you need to compile and package it again.

> mvn clean compile

> mvn clean package

.class and .jar file is generated



> mvn spring-boot:run

With help of this we will run the application on local system.

We can see the application is running on port 8080

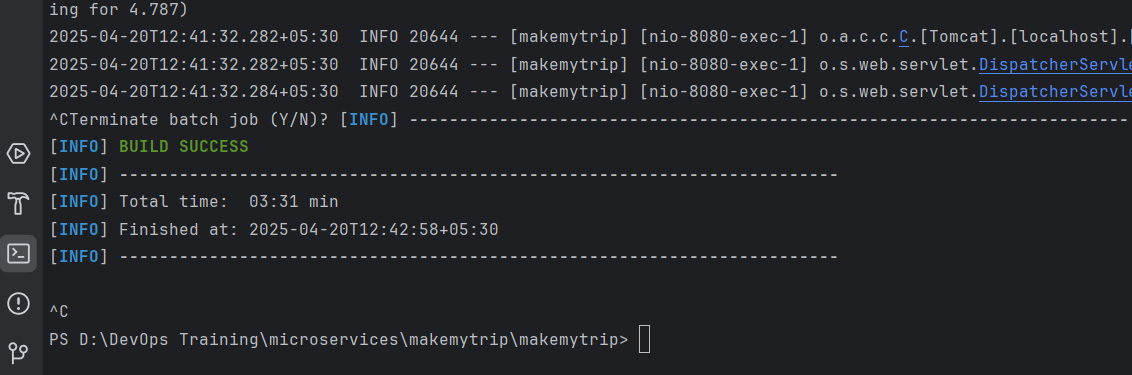


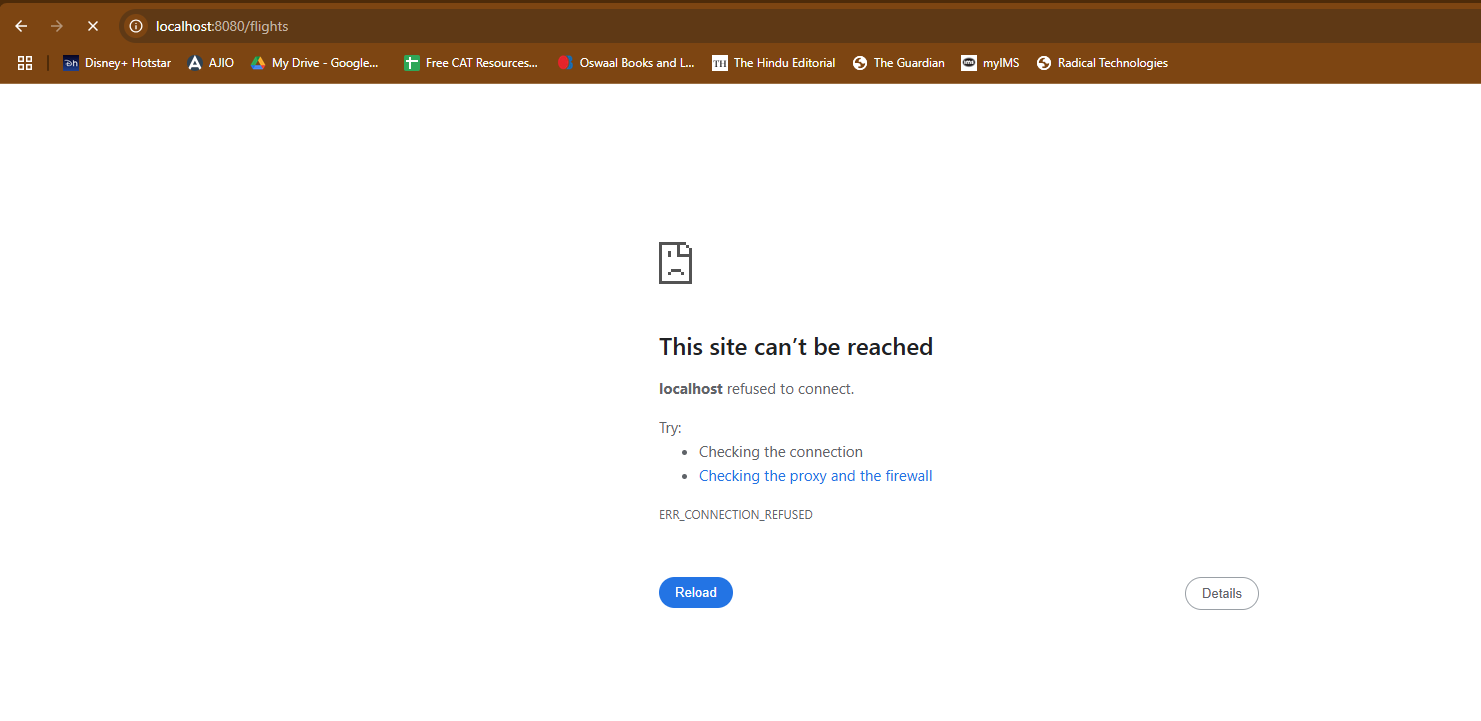
Now run the flights microservice on localhost

<http://localhost:8080/flights>



Press control C on terminal and the application will stop. If you check the link again it wont work anymore





Add new feature now: hotels.java

package com.example.makemytrip;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

@RestController

public class hotels {

public hotels() {

}

@GetMapping({"/hotels"})

public String getData() {

return "Book your HOTELS for Pune at 50% OFF!";

}

}

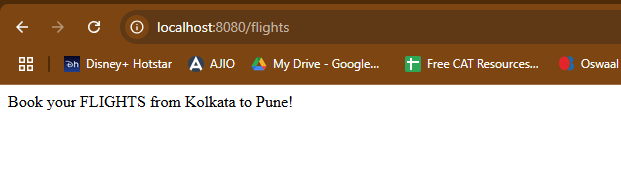
Now clean compile and package to make sure the new code is not spoiling the old working code.

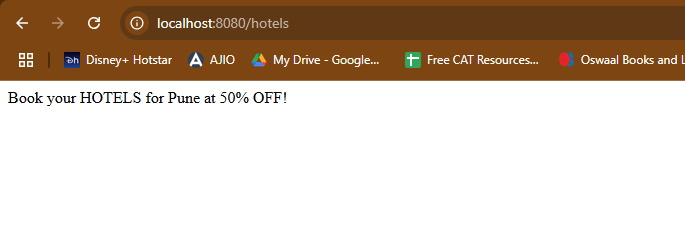
> mvn clean compile

> mvn clean package

And then verify both the applications: hotels and flights

> mvn spring-boot:run

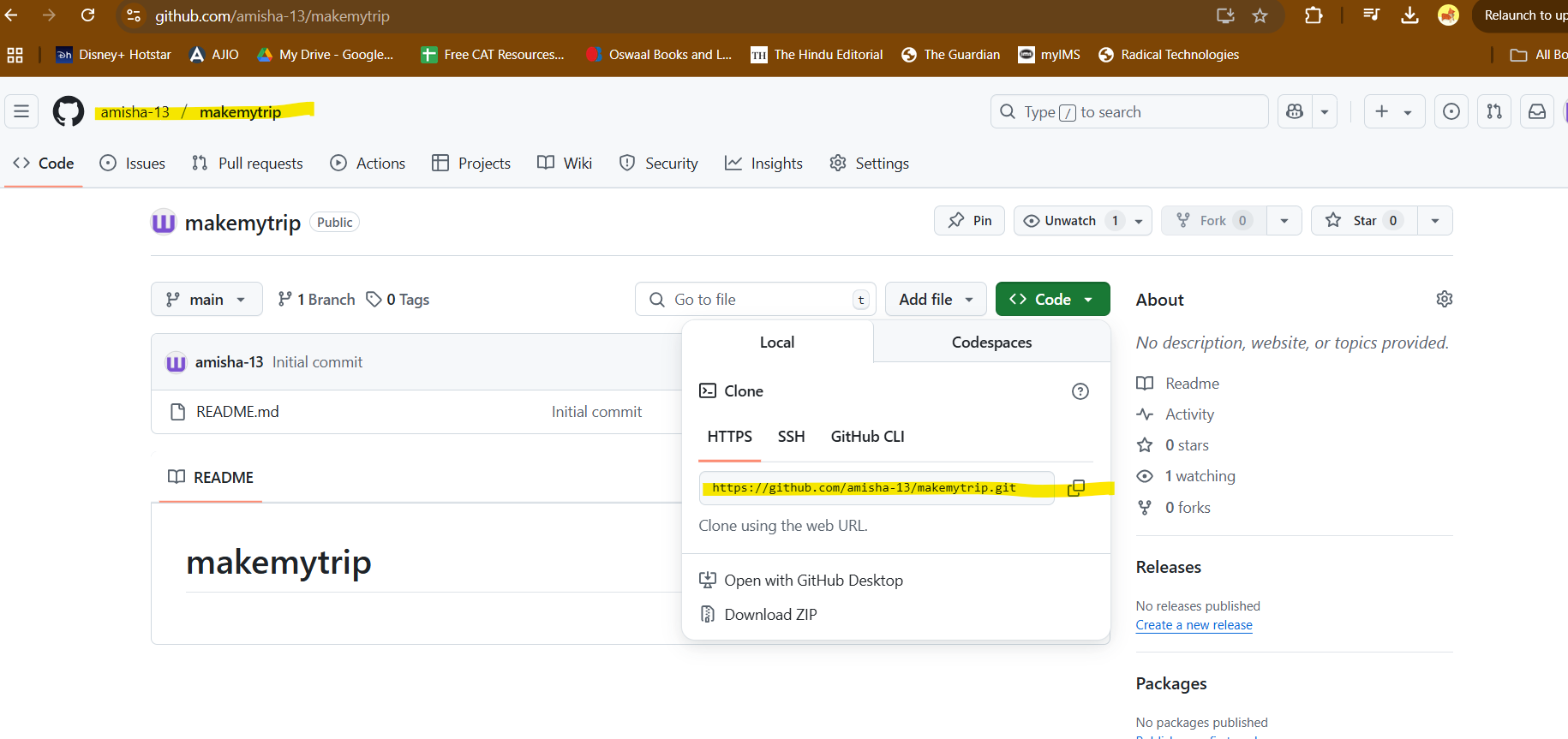




So we can see both features are working on local system.

Once the functionalities are verified and code is working - the code is supposed to be pushed on **github - shared remote code repository**.

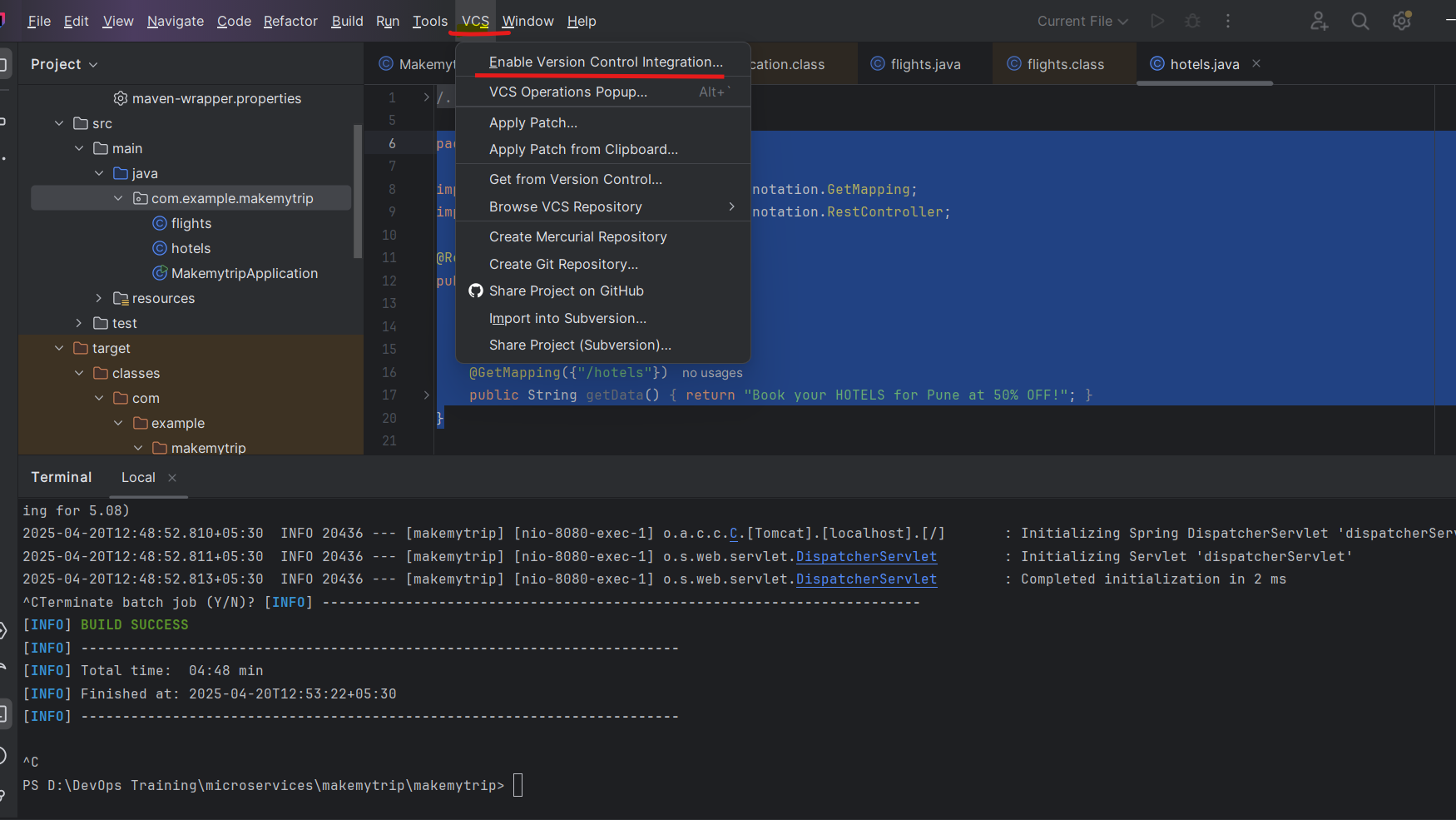
Create new repository on Github: makemytrip



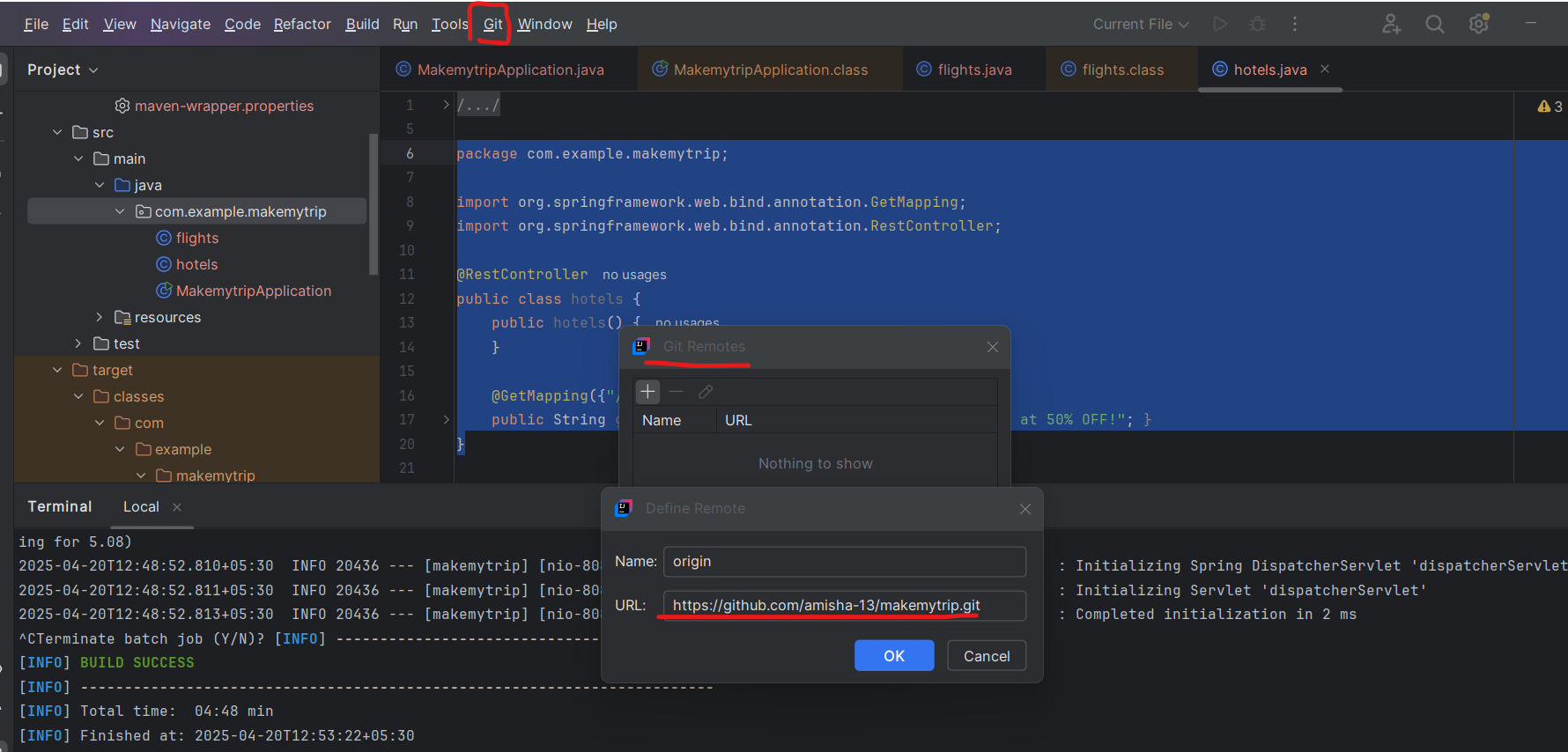
Push the code from local to remote repository

In IntelliJ IDEA enable version control integration:

VCS > Enable version control integration

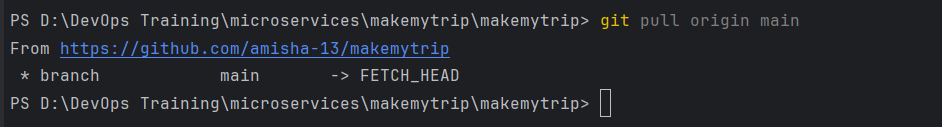


VCS > Manage remote > Enter URL of remote repository just copied



In my case - my default branch name is main and not master

> git pull origin main



1st time you run this:

git pull

git pull origin master

git add --all

git commit -m "CodeRefactor"

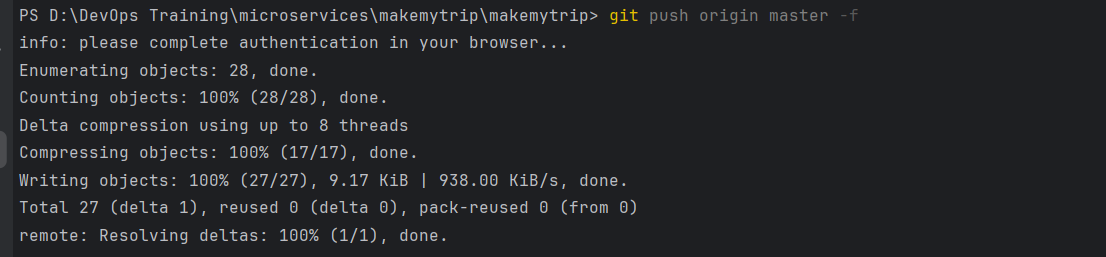
git push origin master -f

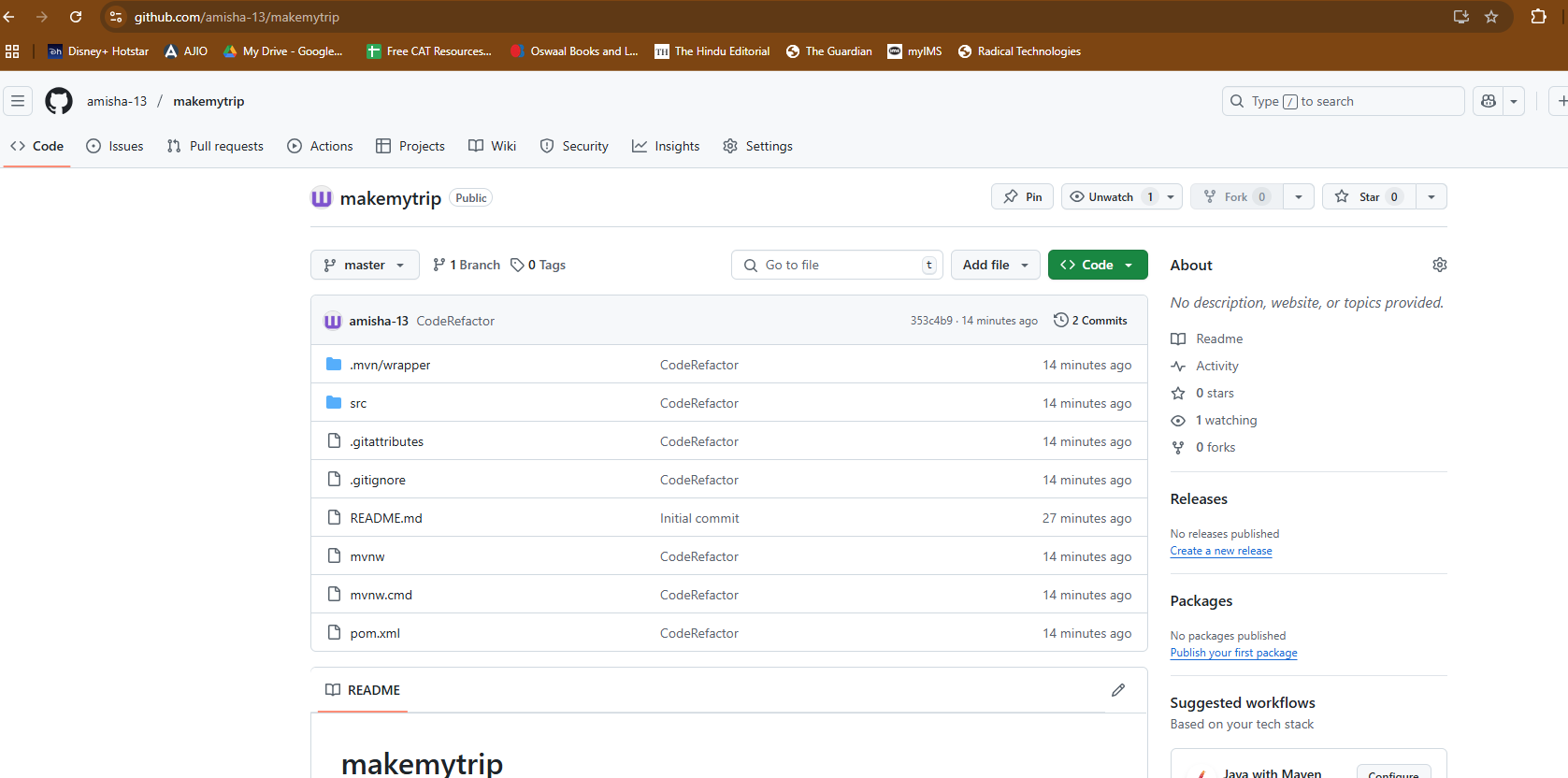
2nd and all times

git add --all

git commit -m "message"

git push origin master





Add new class - bus.java

package com.example.makemytrip;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

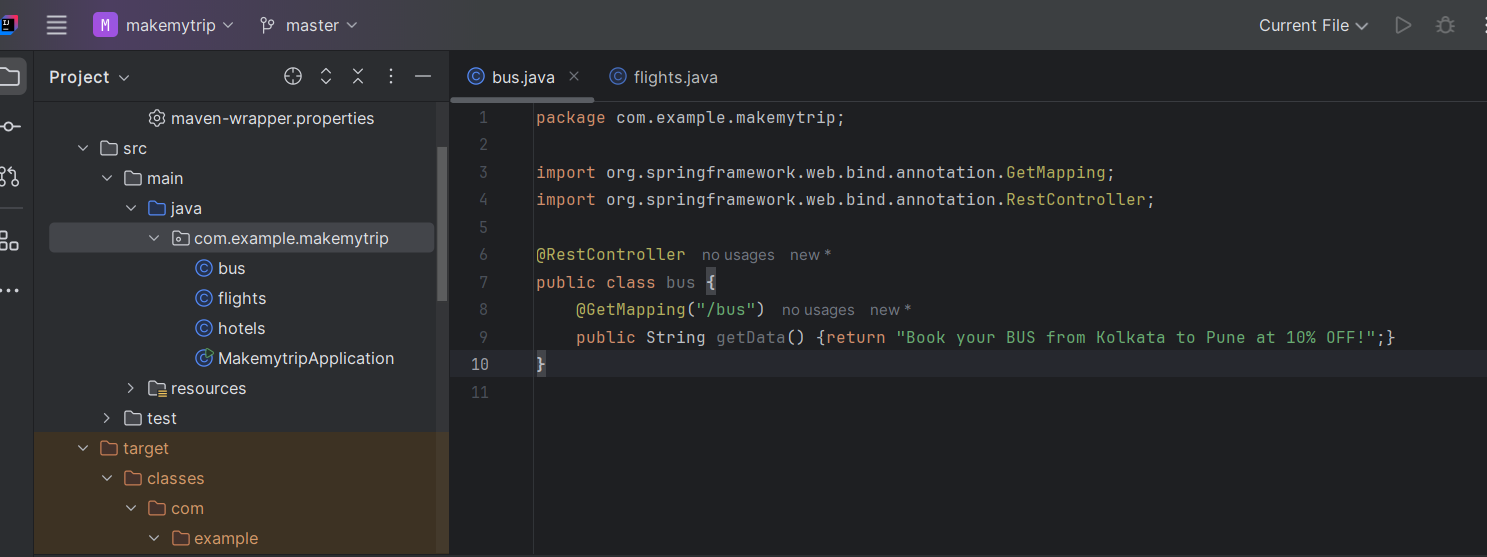
@RestController

public class bus {

@GetMapping("/bus")

public String getData() {return "Book your BUS from Kolkata to Pune at 10% OFF!";}

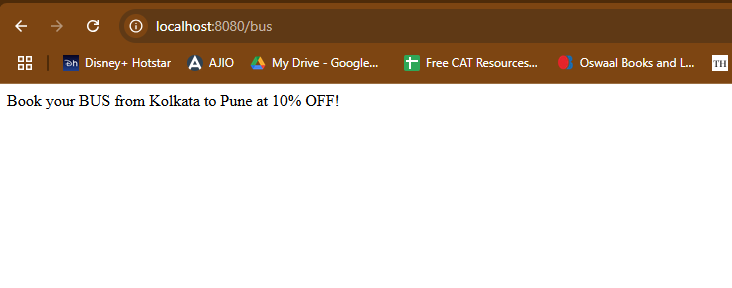
}



> mvn clean compile

> mvn clean package

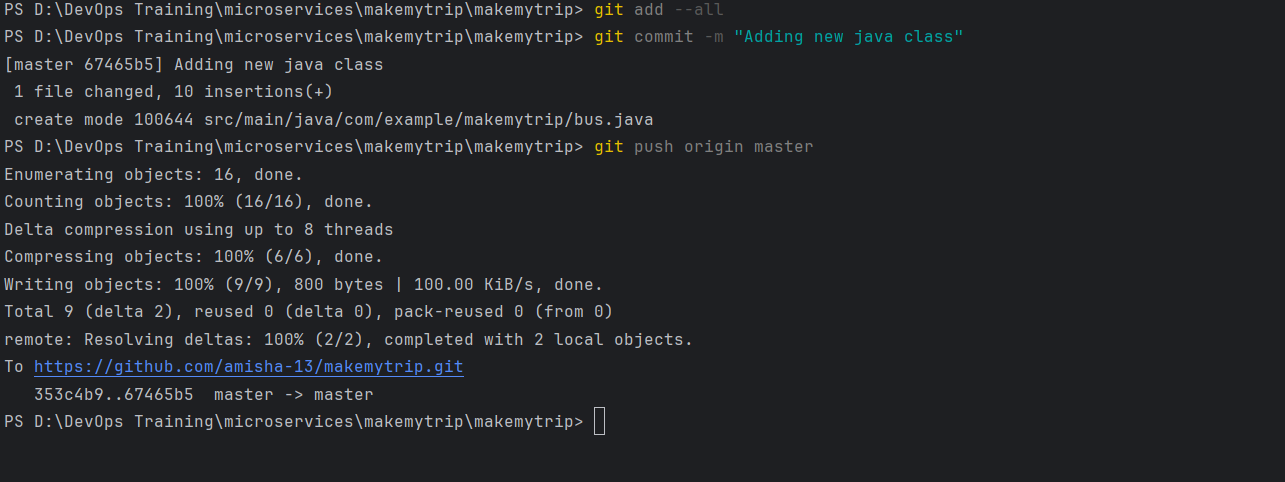
> mvn spring-boot:run



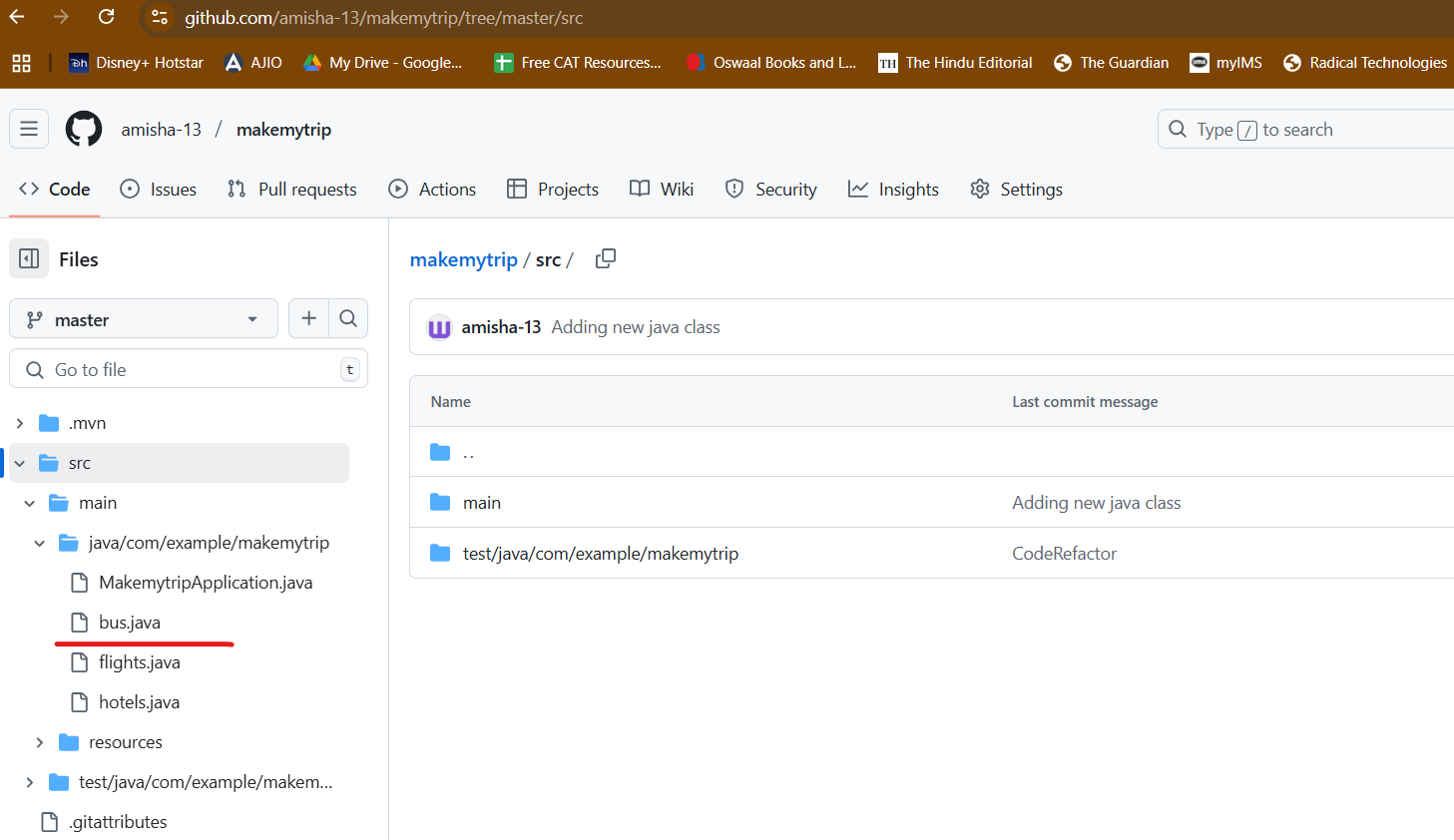
> git add --all

> git commit -m "Adding new java class"

> git push origin master



Code is now uploaded



Day 4

Delete repo

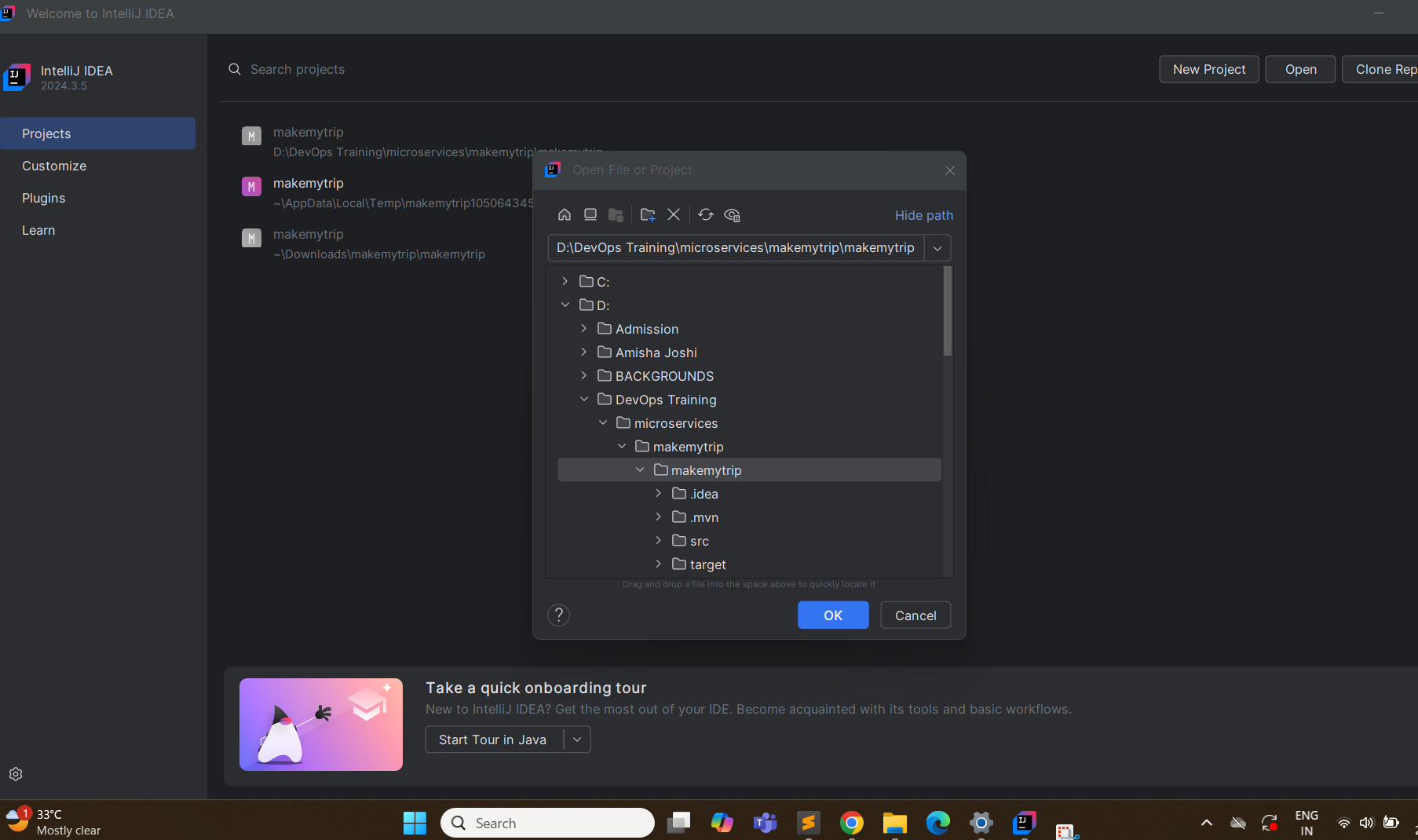
Delete makemytrip folder

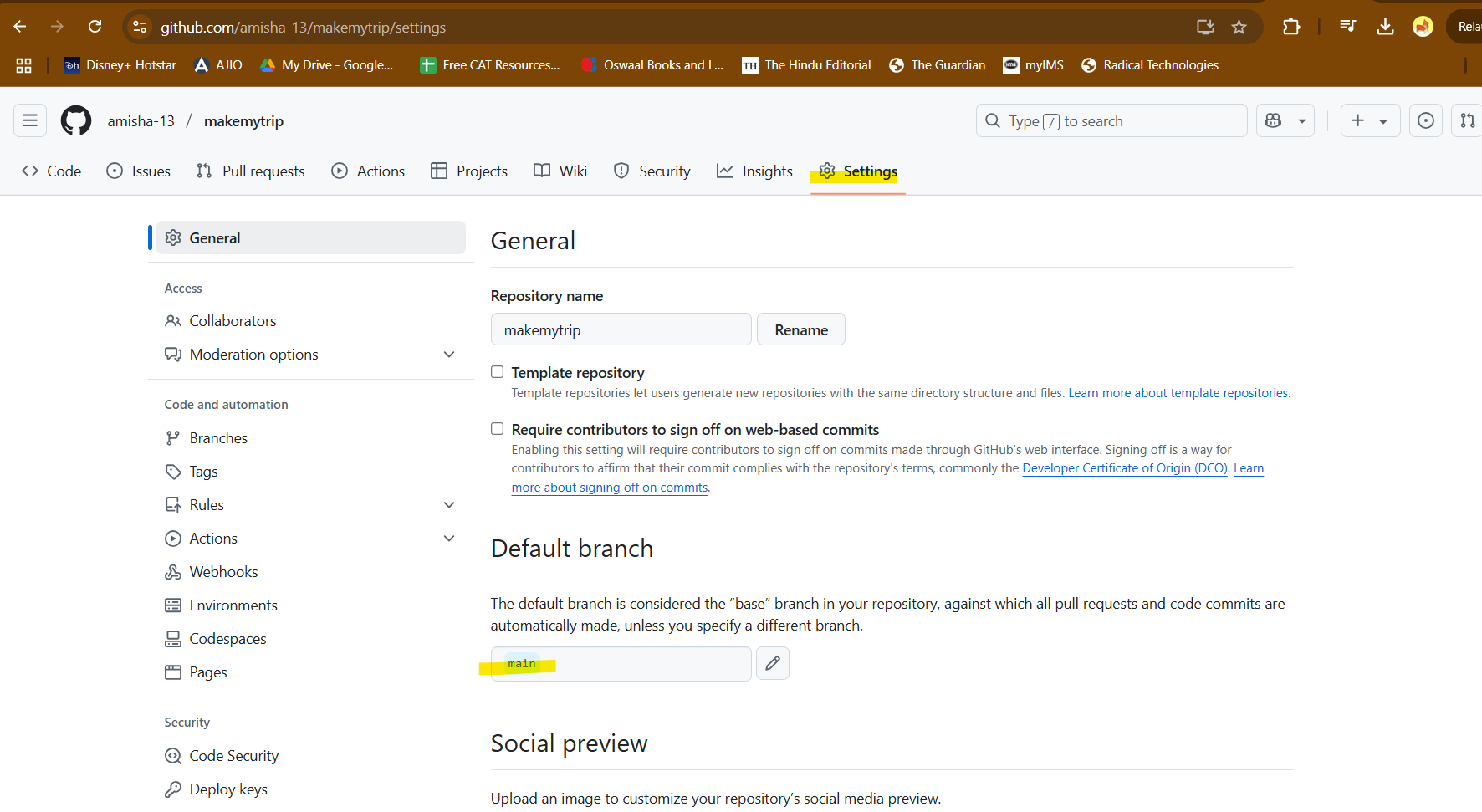
Perfrom cleanup

Download spring framework

Reopen code in IntellIJ Idea

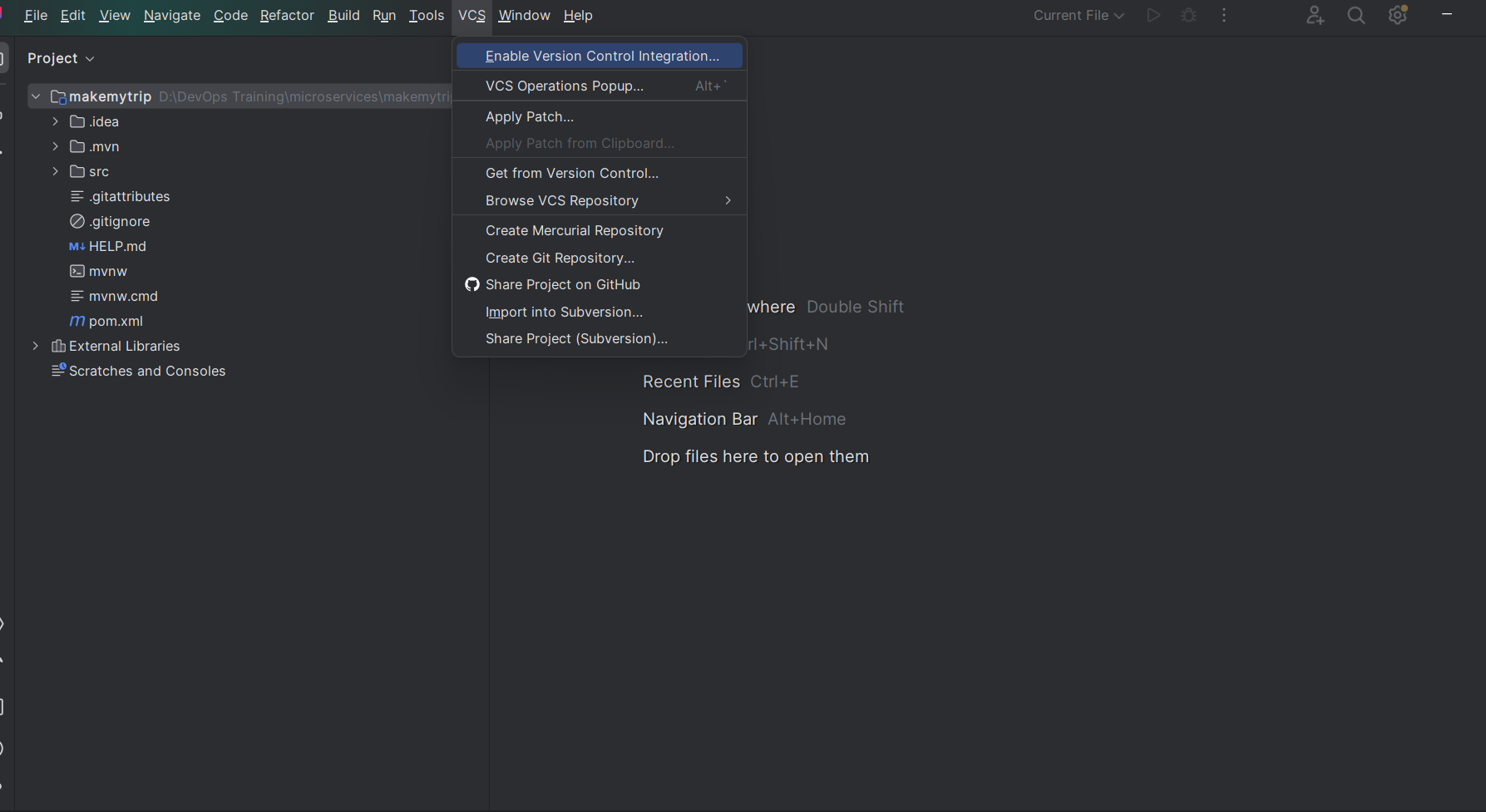
Create new repo: makemytrip



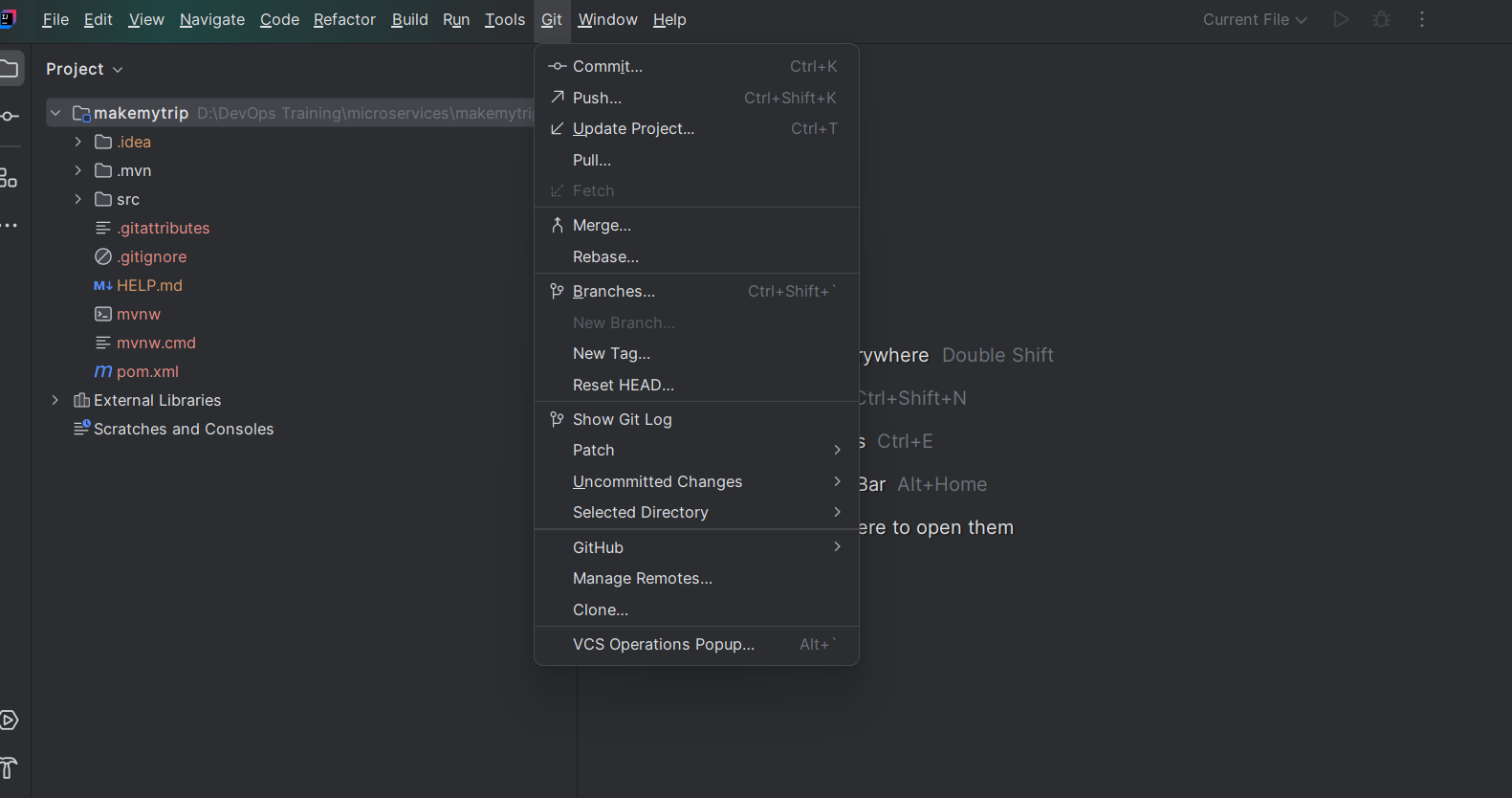


Change branch name to master

Enable Version COntrol on IntelliJ



Select Git as the tool



Now add remote repo: VCS> Manage Remote: add your git URL



First time run:

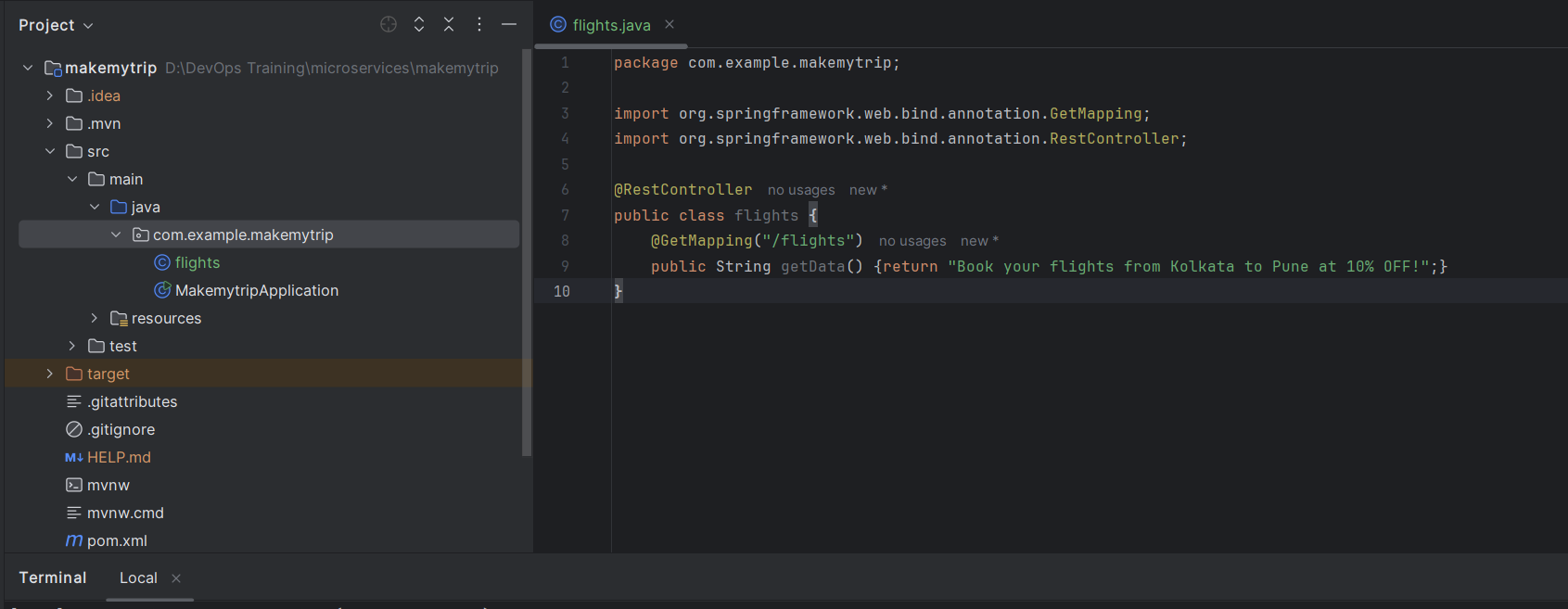
> git pull origin master

> git add --all

> git commit -m "CodeRefactor"

> git push origin master -f

Created flights.java

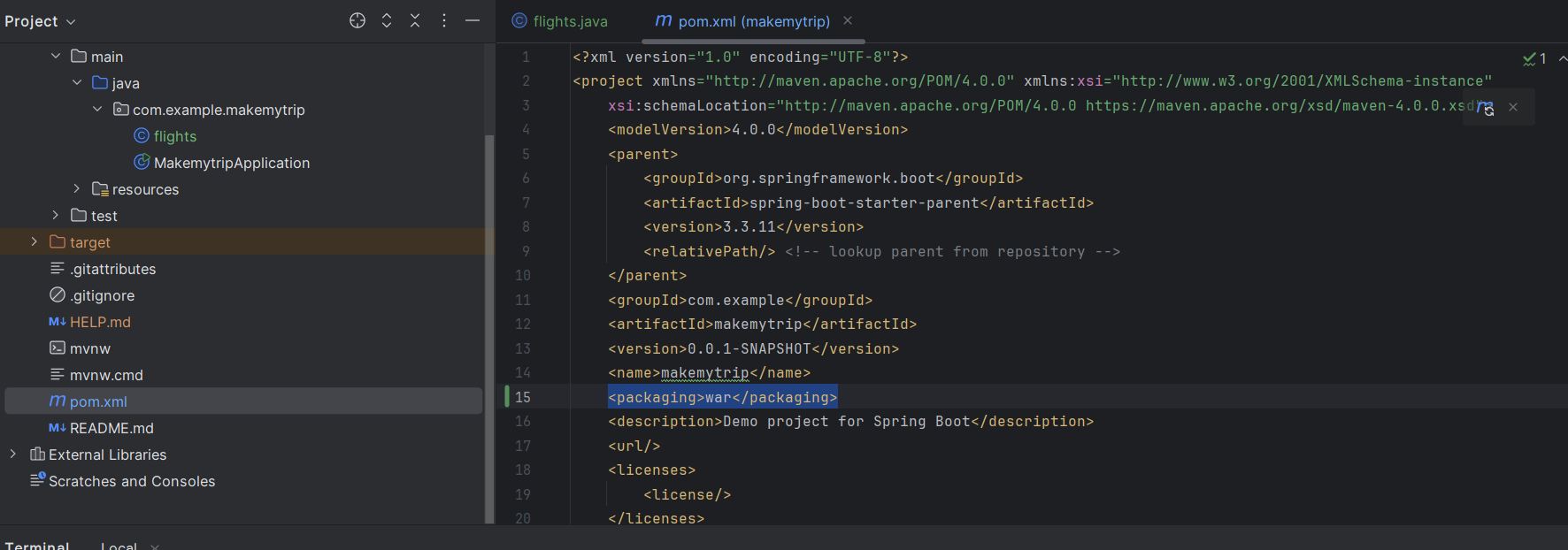


> mvn clean compile

> mvn clean package – jar file created

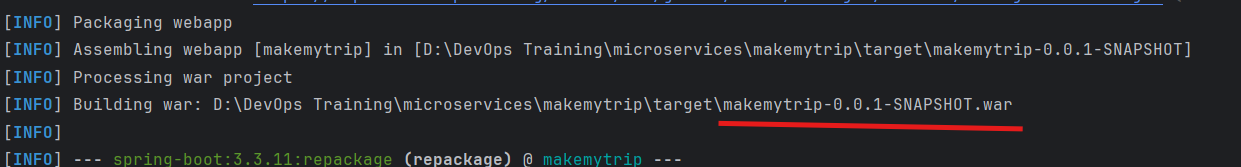
To create .war packaging - add this in pom.xml

<packaging>war</packaging>



Now > mvn clean package

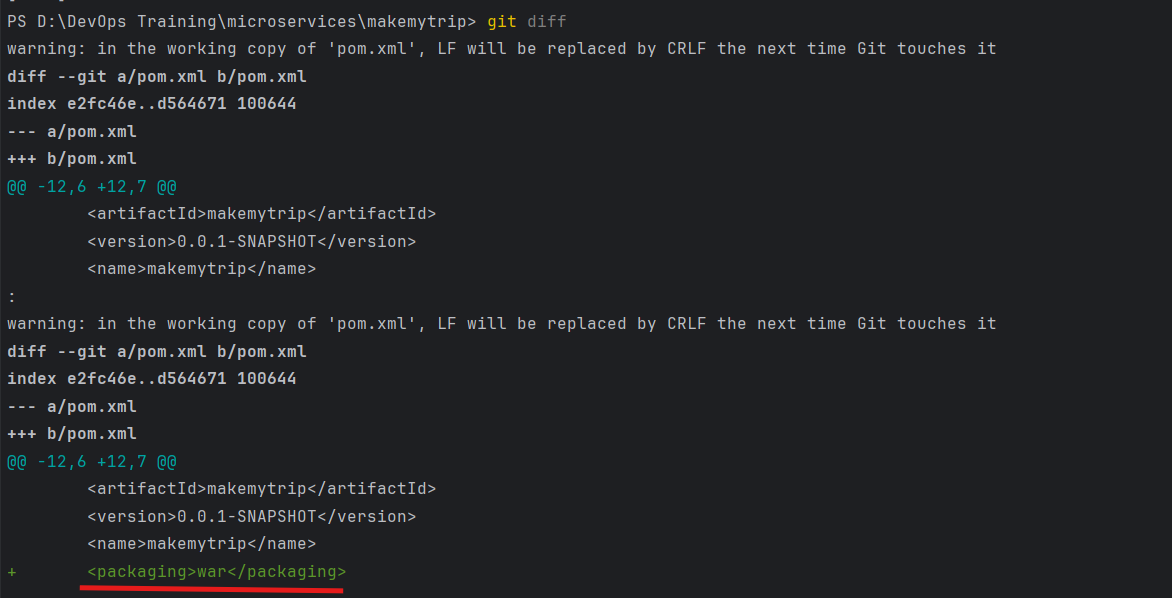
Created war file



This is not added in remote repo:

> git diff

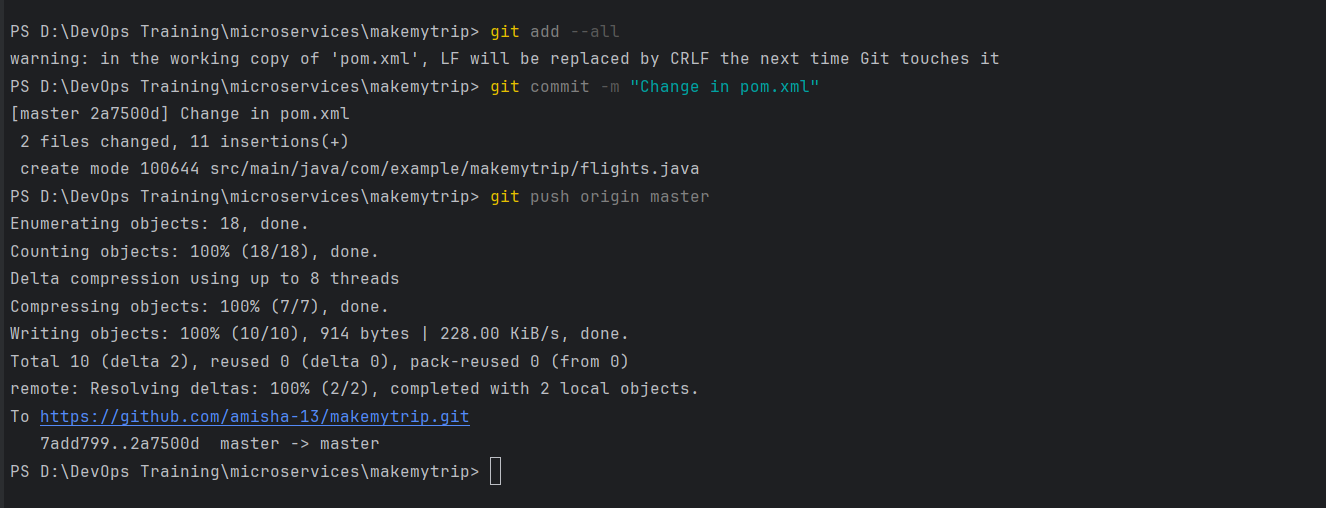
Tells the changes



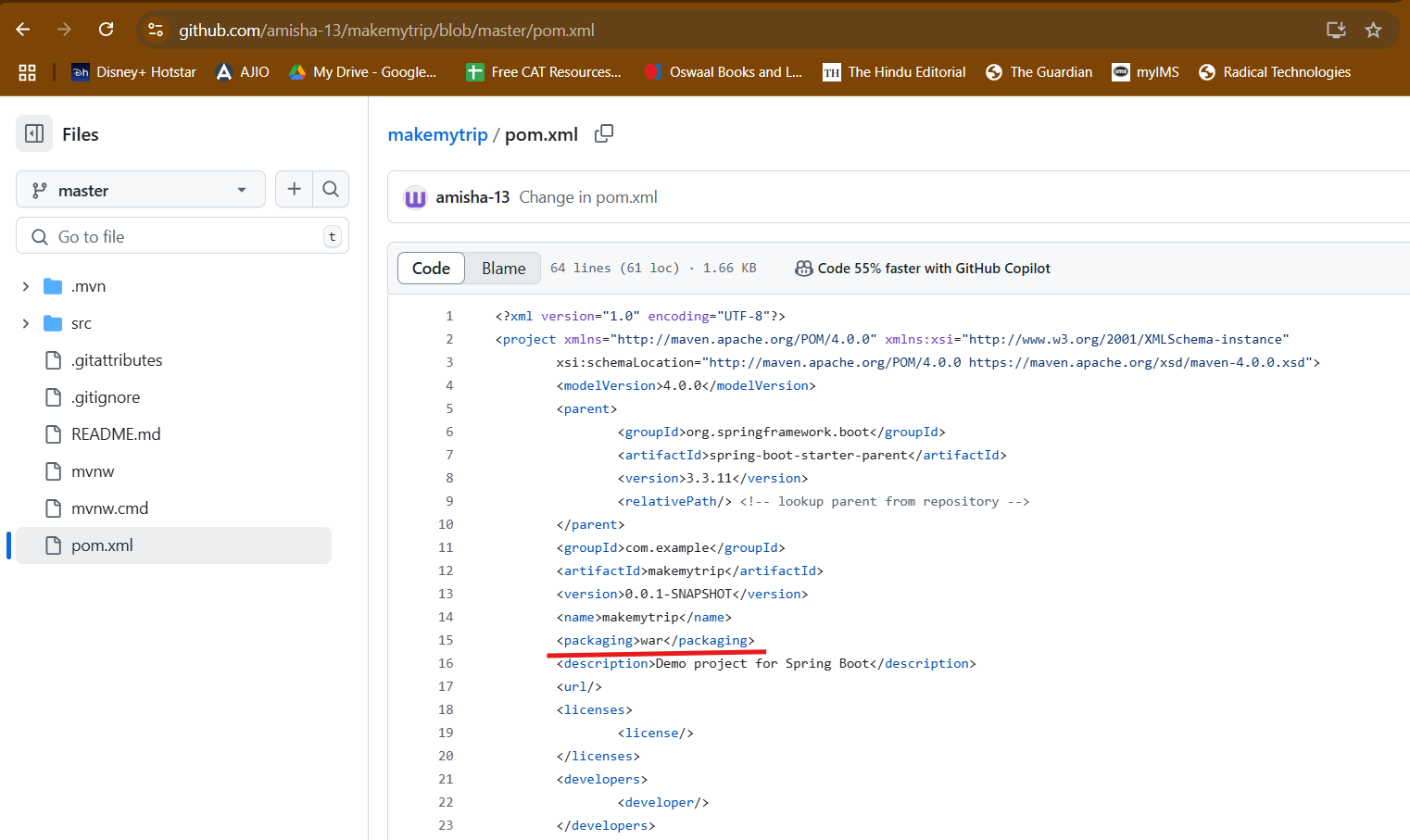
> git add --all

> git commit -m "Change in pom.xml"

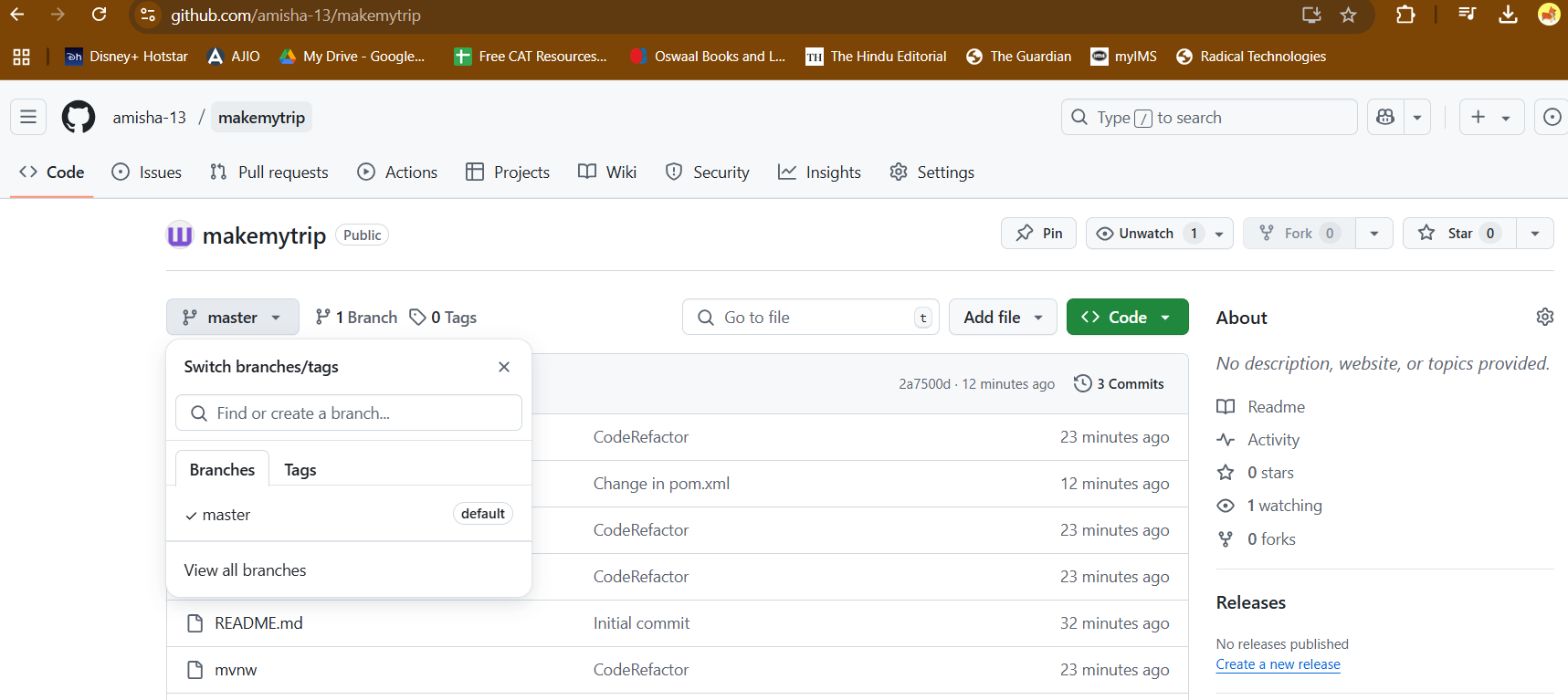
> git push origin master — Move from local to remote

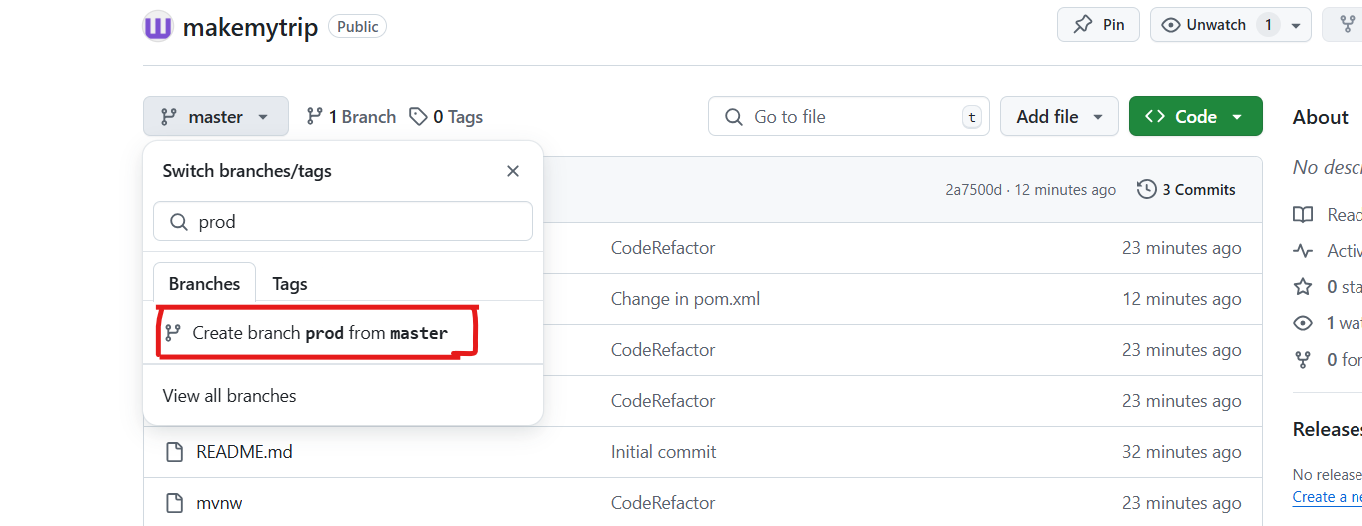


This is reflected in remote repo as well

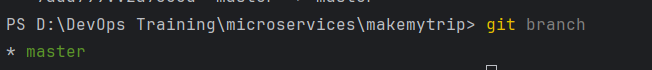


Create brand prod from master, preprod from prod and dev from preprod





CHeck which branches you have

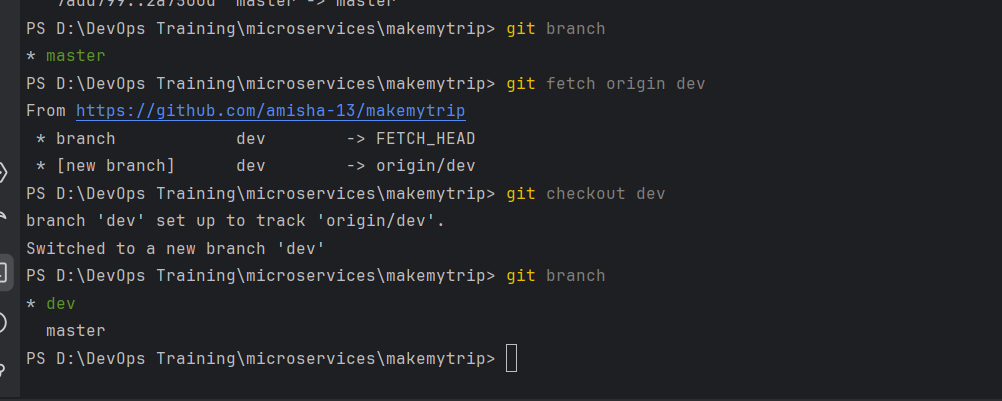


Dev is the lowest branch and We dont have dev branch in local

So:

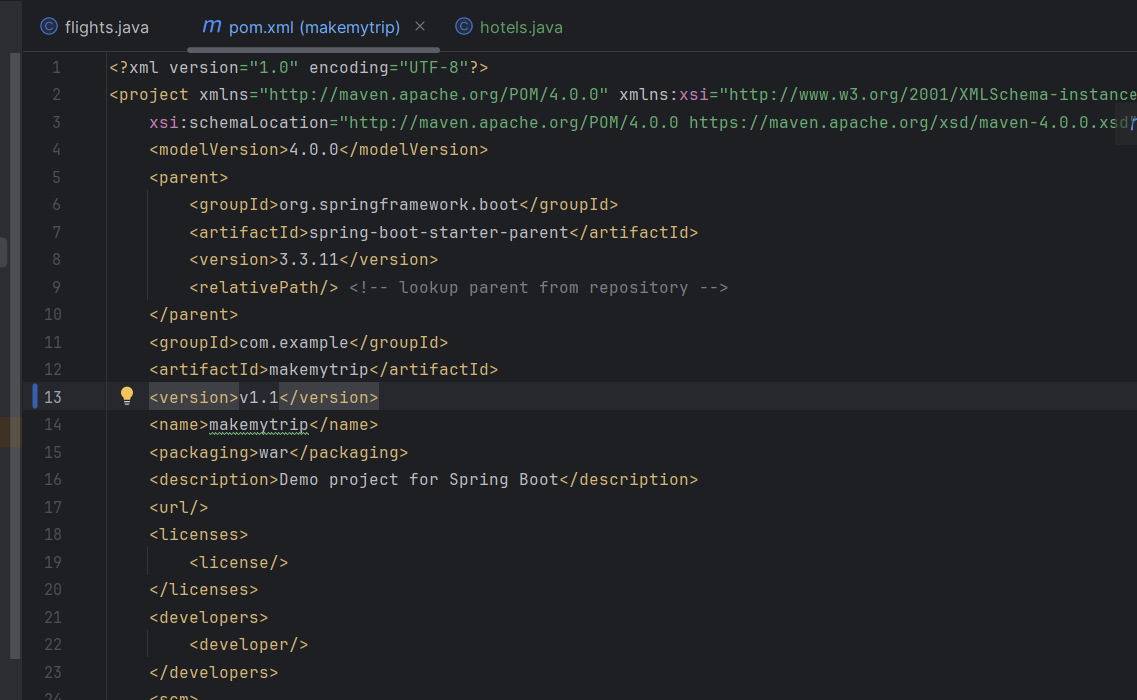
Git fetch origin dev – get from remote to local

Git checkout dev – switch the dev branch

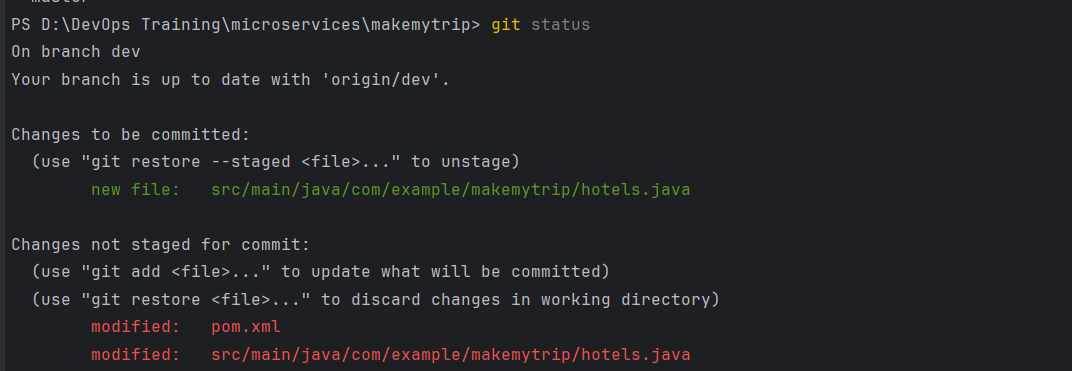


Make new java class : flights.java

Make some changes in pom.xml



Git status – will tell changes modified in working directory



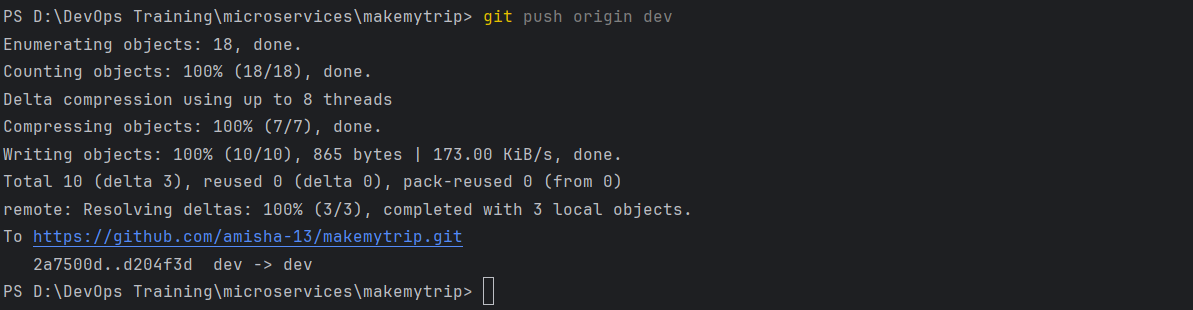
Now since i added new java file, package it

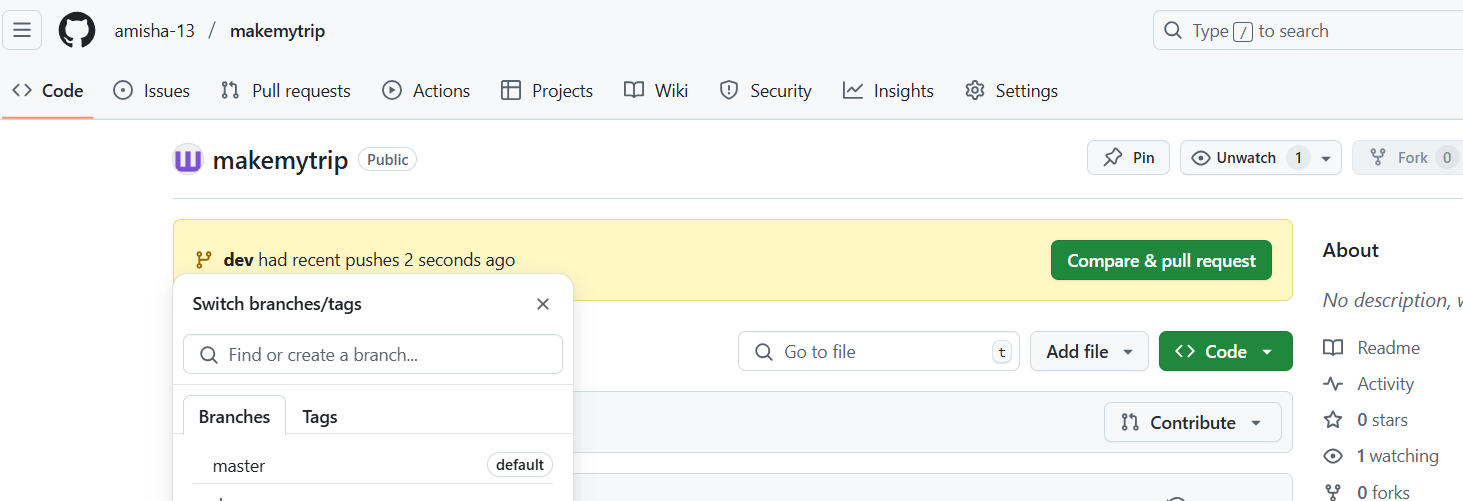
> mvn clean package (Runs compile also by default)

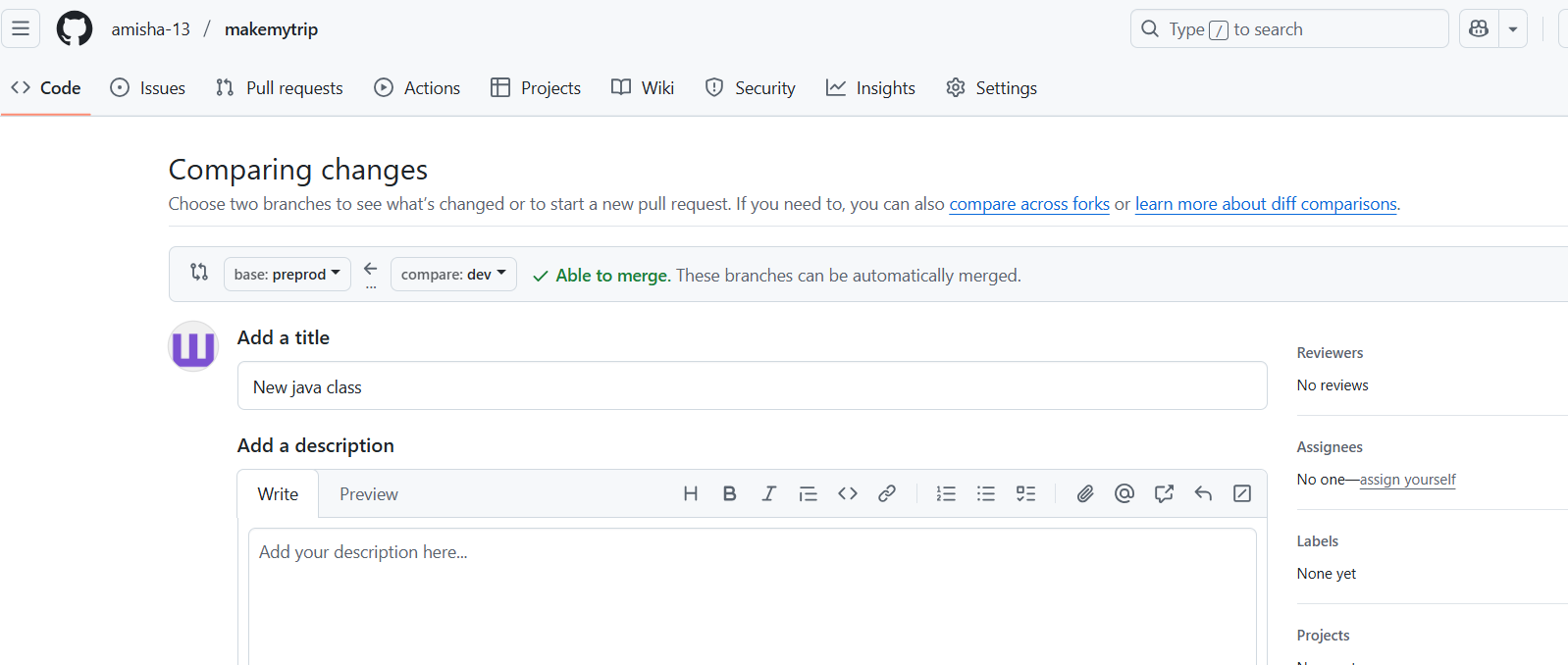
> git add --all

> git commit -m "New java class"

As a dev, I cannot push code in any other branch but dev



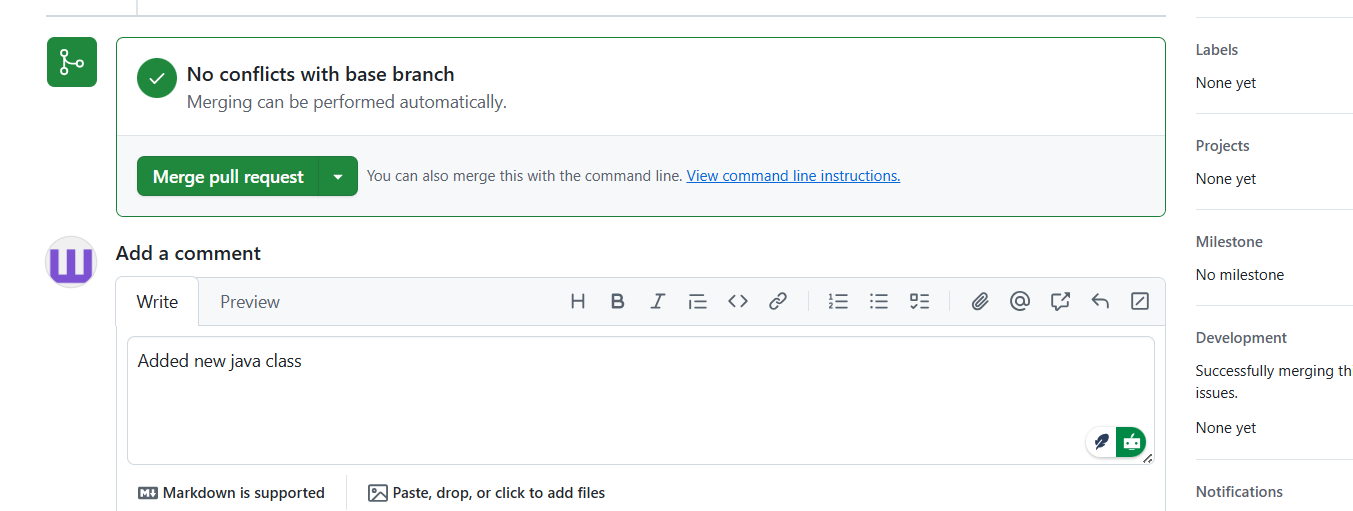


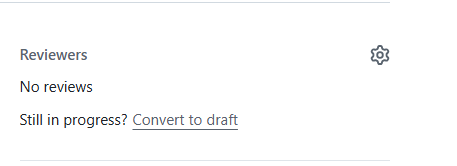


Dev changes will be merged into preprod only

And Preprod in prod

Prod need not be merged with master



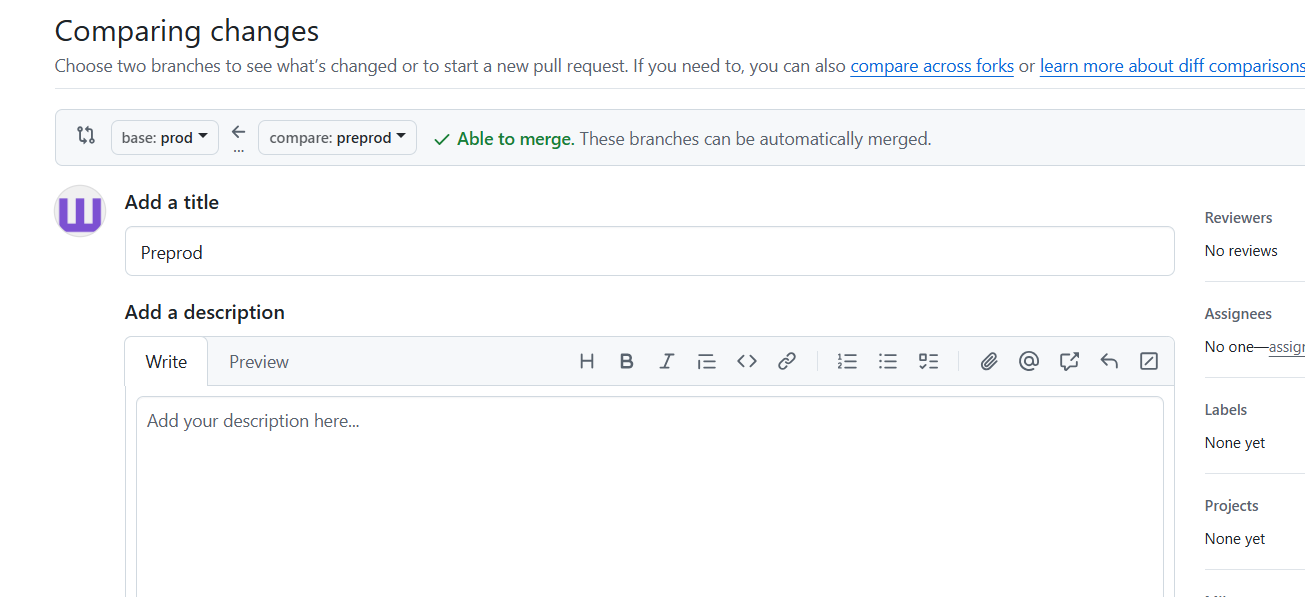


Add in reviewers, but you cannot approve it yourself



Confirm merge

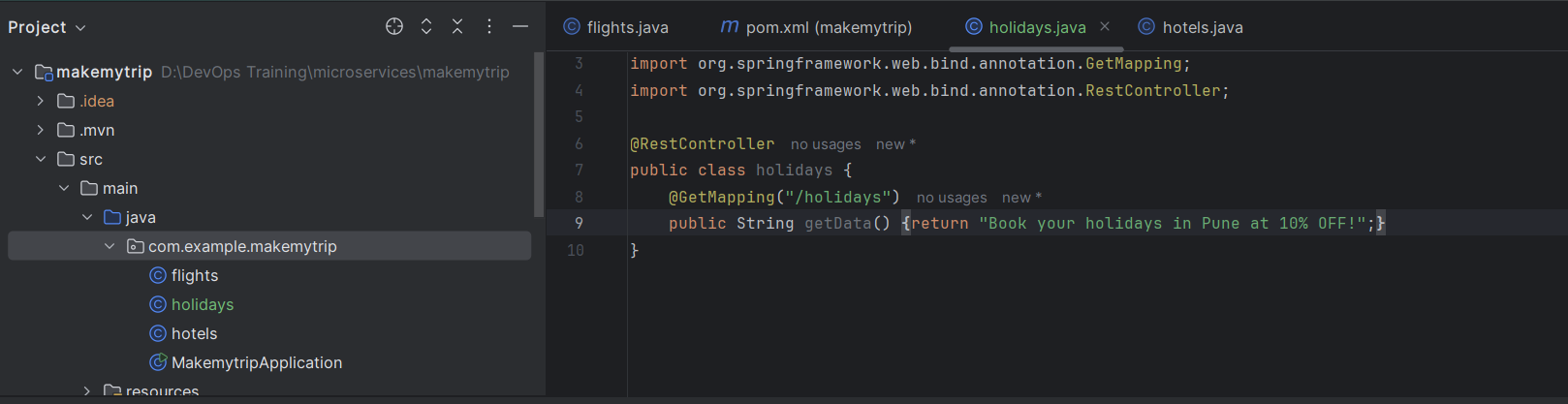
Now push preprod changes to prod

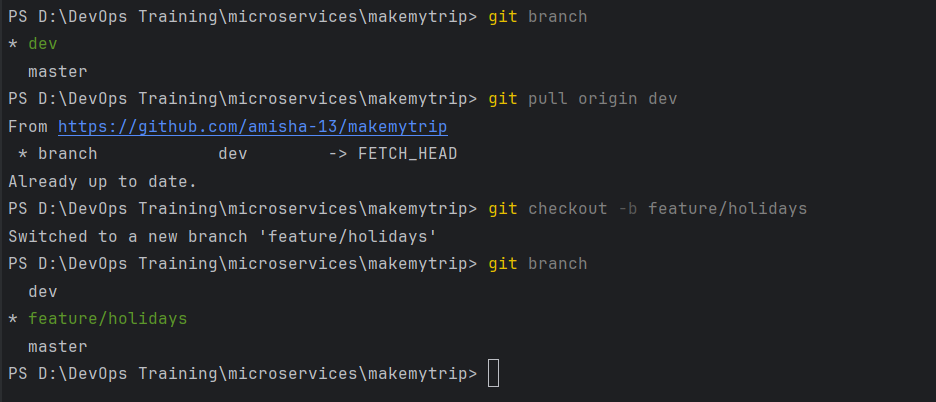


Because code is now deployed in prod already so no need to add it into master

> git pull origin dev – Dev branch files of remoted will be synced with local repository

> git checkout -b feature/holidays – new new branch from dev in local

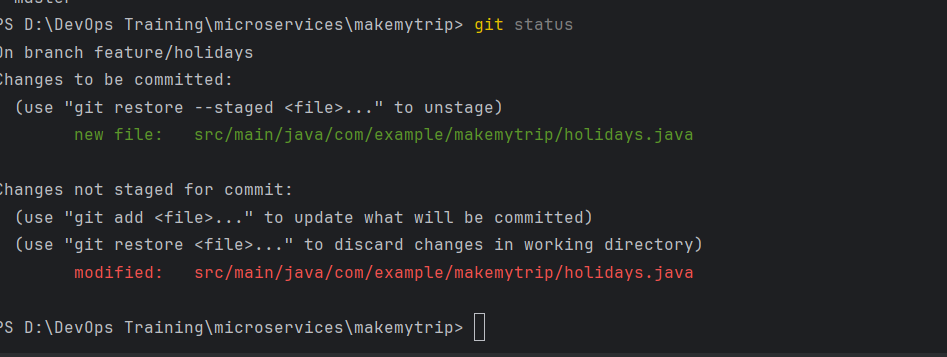




Make a new java class file: holidays.java

Run

>git status – to see the differences



> mvn clean package

> git branch – make sure it is feature branch

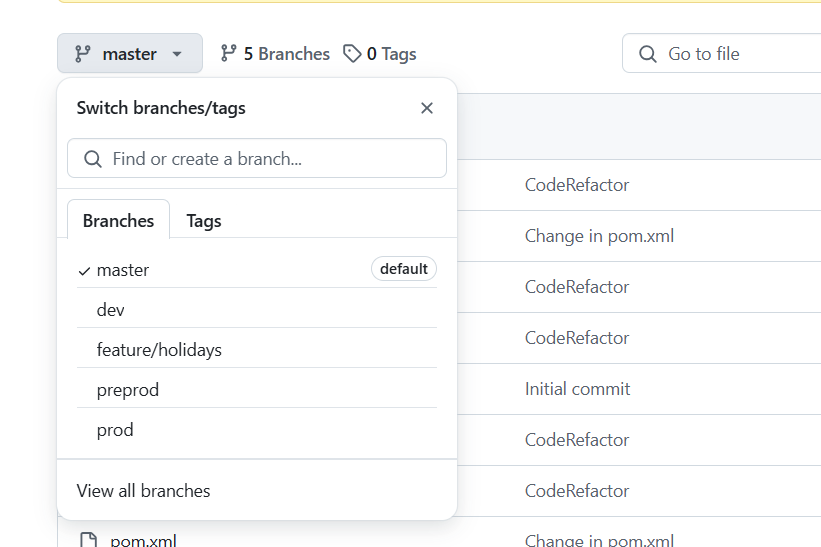
> git add --all

> git commit -m "Adding new java class"

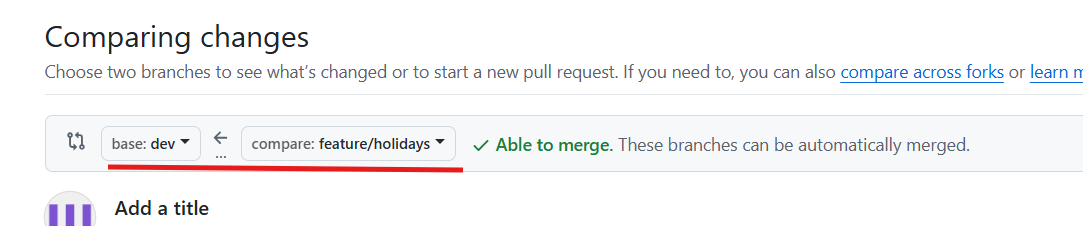
> git push origin feature/holidays

Now pushed into feature/holidays (local branch)

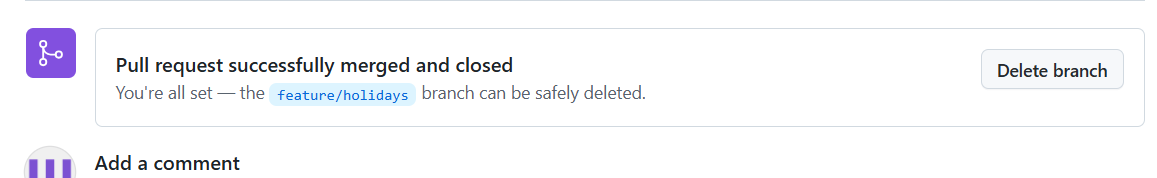
We can see new branch coming up in remote

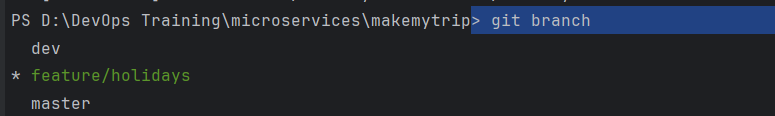


Push changes to dev



After merge, feature branch can be successfully deleted if you want





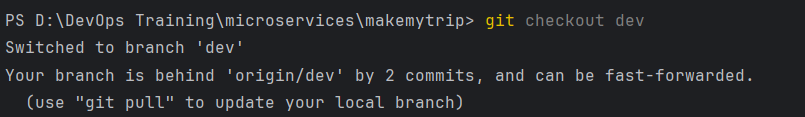
Pull dev from origin and Switch to dev

> git pull origin dev

> git checkout dev

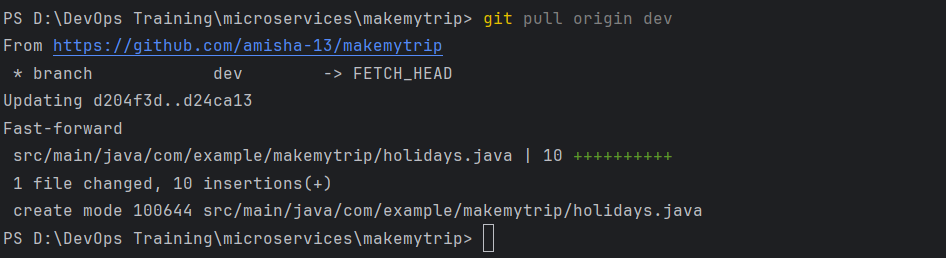
This shows my dev branch is behind the remote

SO remote is UPSTREAM



To bring in sync, pull the changes:

> git pull origin dev



Now it is up to date with remote