Developing & Committing backend & Frontend code using Openshift Dev Spaces

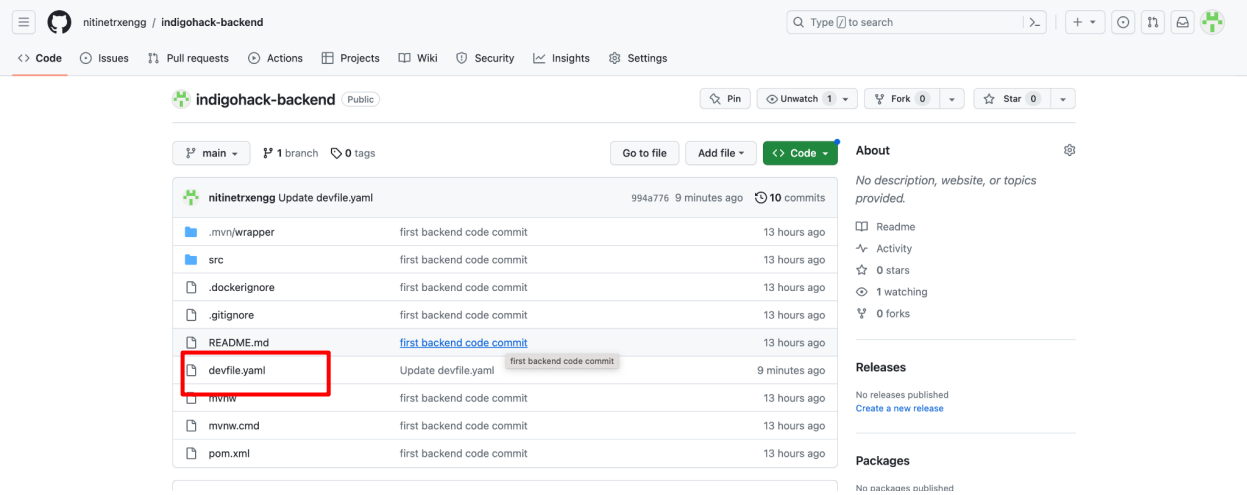
Before we start working on code development for backend & frontend services, copy/commit the corresponding devfile as per the code language from the following link & put at root of your project directory in your GitHub repository. **Name the file as devfile.yaml**.

Your GitHub Repository code structure should look like this

Your Github repository = Your Code + devfile.yaml (devfile.yaml links provided below)

Note : These devfiles are provided with most of the generic entries.. Do not change the generic configurations provided but you can add your environment variables in this devfile as per your code requirements under env section. But do not delete any other environment variables or any other entries.

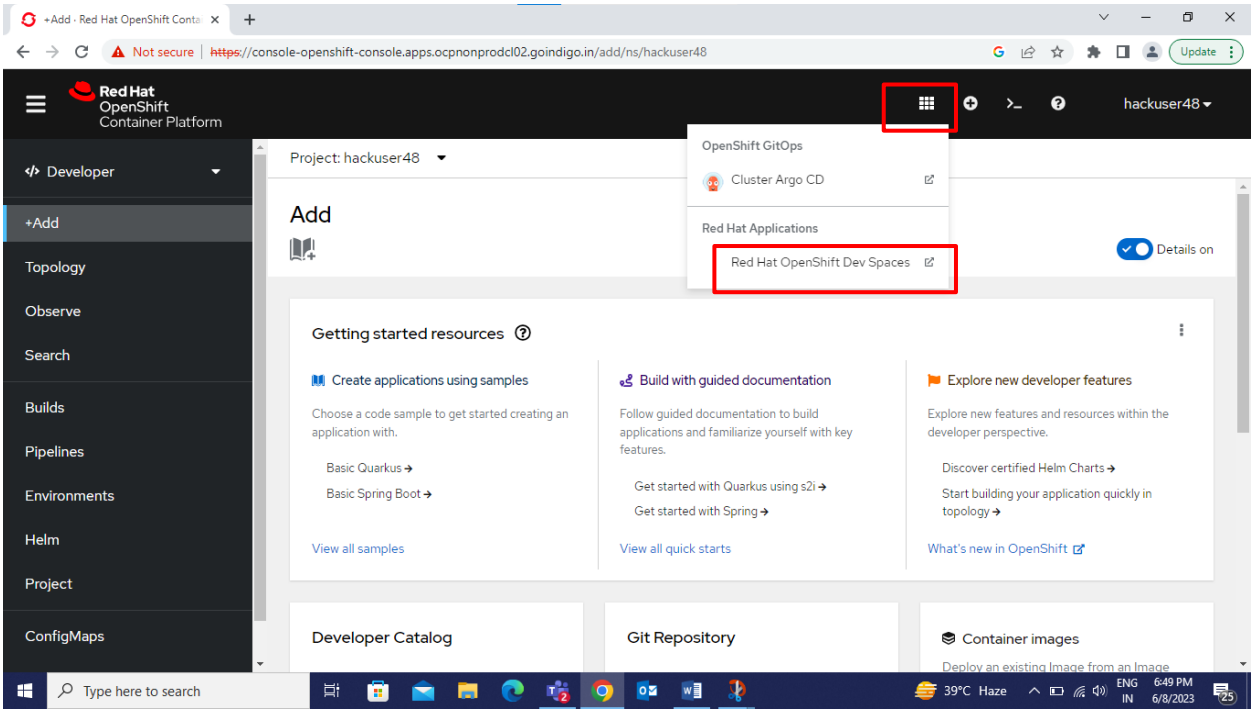
- Spring Boot (Java) Devfile
  - https://github.com/nitinetrxengg/indigohackdevfiles/blob/main/javaspringboot/devfile.yaml
- .Net Devfile
  - https://github.com/nitinetrxengg/indigohackdevfiles/blob/main/dotnet/devfile.yaml
- Nodejs Devfile
  - https://github.com/nitinetrxengg/indigohackdevfiles/blob/main/nodejs/devfile.yaml
- React Devfile
  - https://github.com/nitinetrxengg/indigohackdevfiles/blob/main/react/devfile.yaml
- Python Devfile
  - https://github.com/nitinetrxengg/indigohackdevfiles/blob/main/python/devfile.yaml



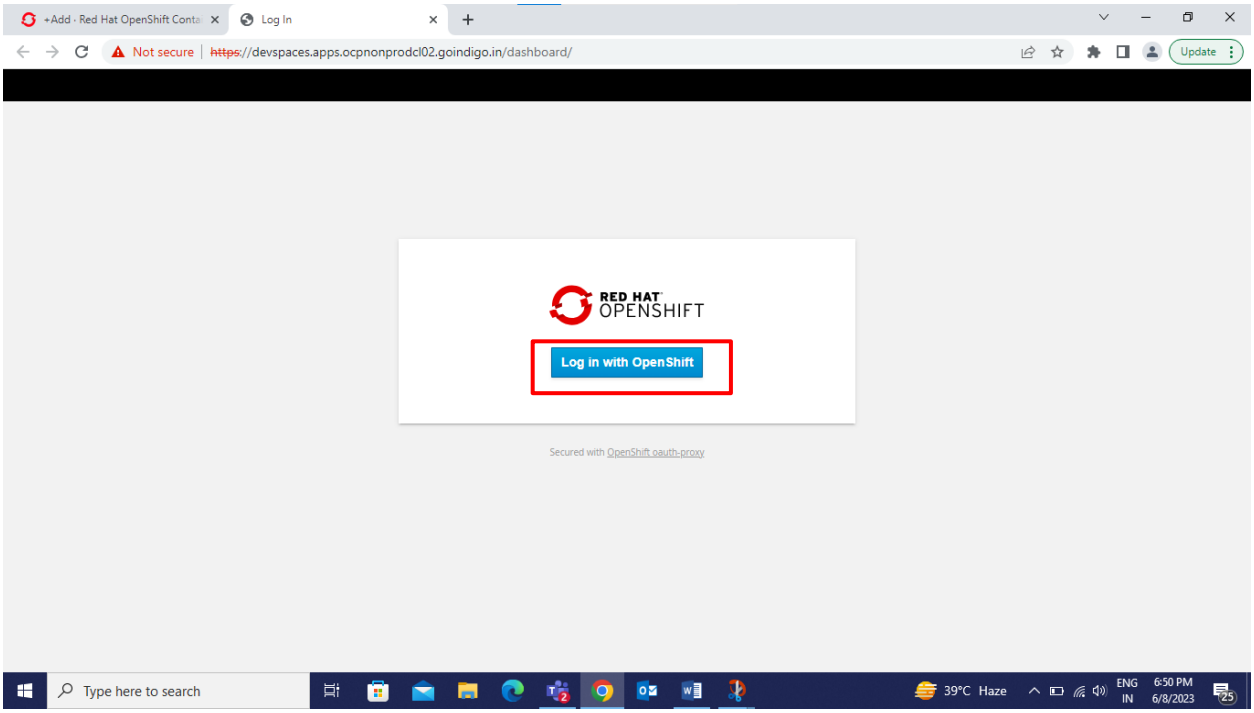
Post committing the devfile.yaml in your GitHub repositories, follow the below steps for doing backend & frontend code development

1. Click on the OCP Web console as shown below to access the Red Hat Openshift Dev Spaces which will provide workspaces based on VScode IDE to further develop your project code.

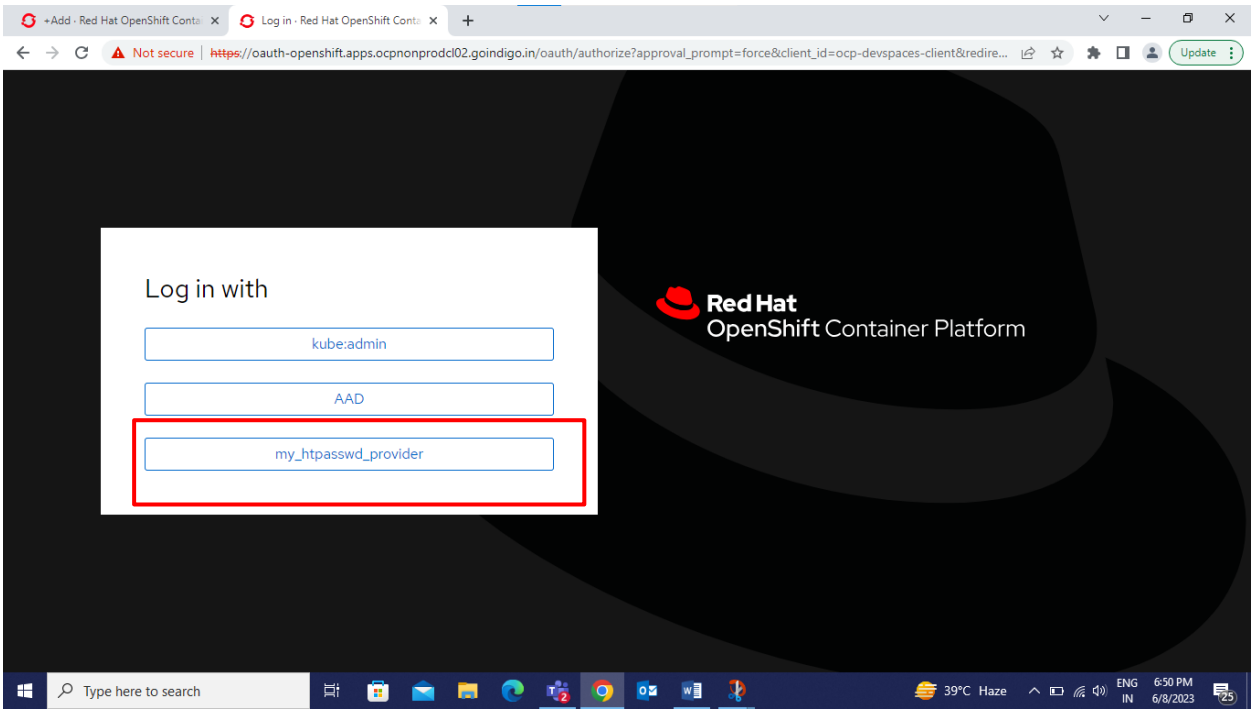
Note: If it displays the certificate security warning then ignore it & proceed to the link



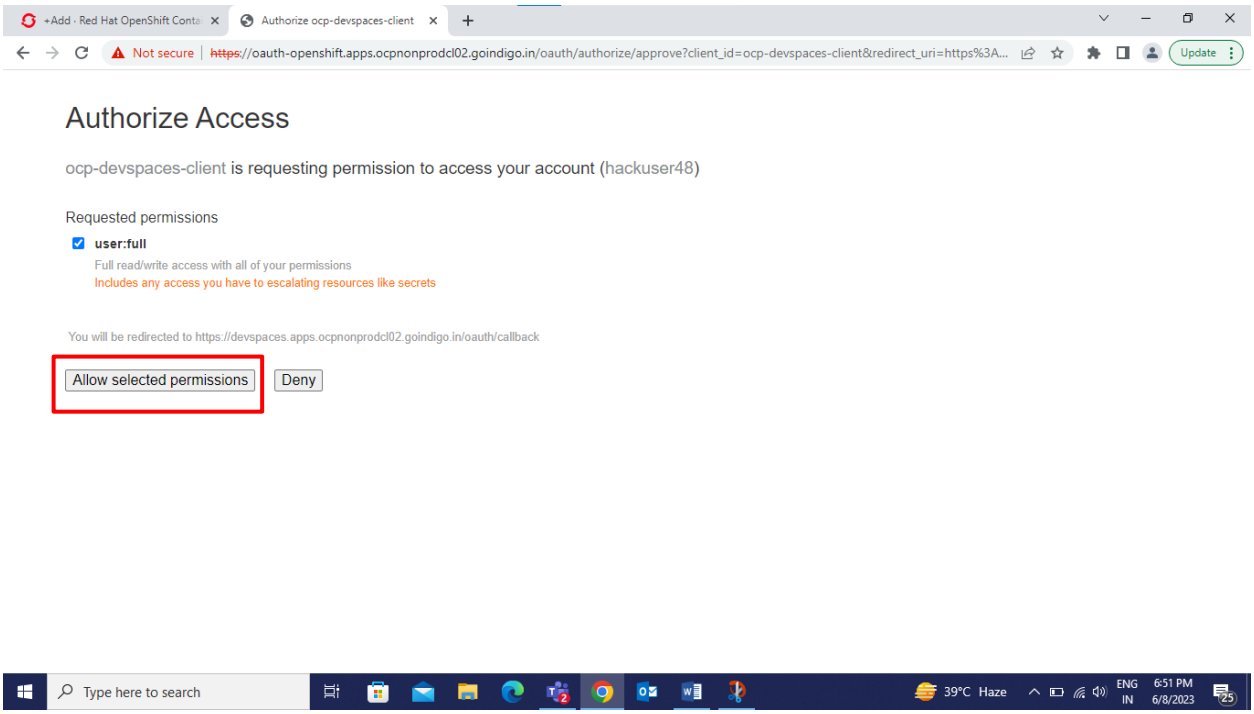
2. It will prompt you for login & Click on Login with Openshift



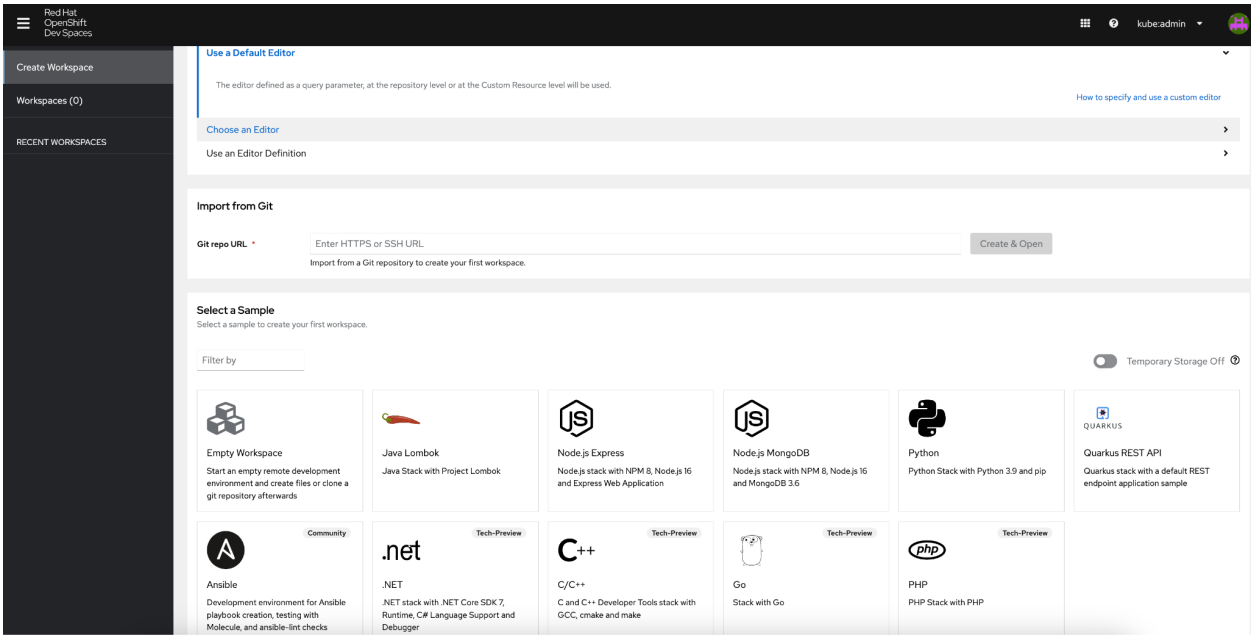
3. Use the same "my_htpasswd_provider" method & enter the same credentials used earlier for login. Do not select any other method.



4. Click on “Allow selected permissions” & this will take you to the Red Hat Openshift Dev Spaces Screen

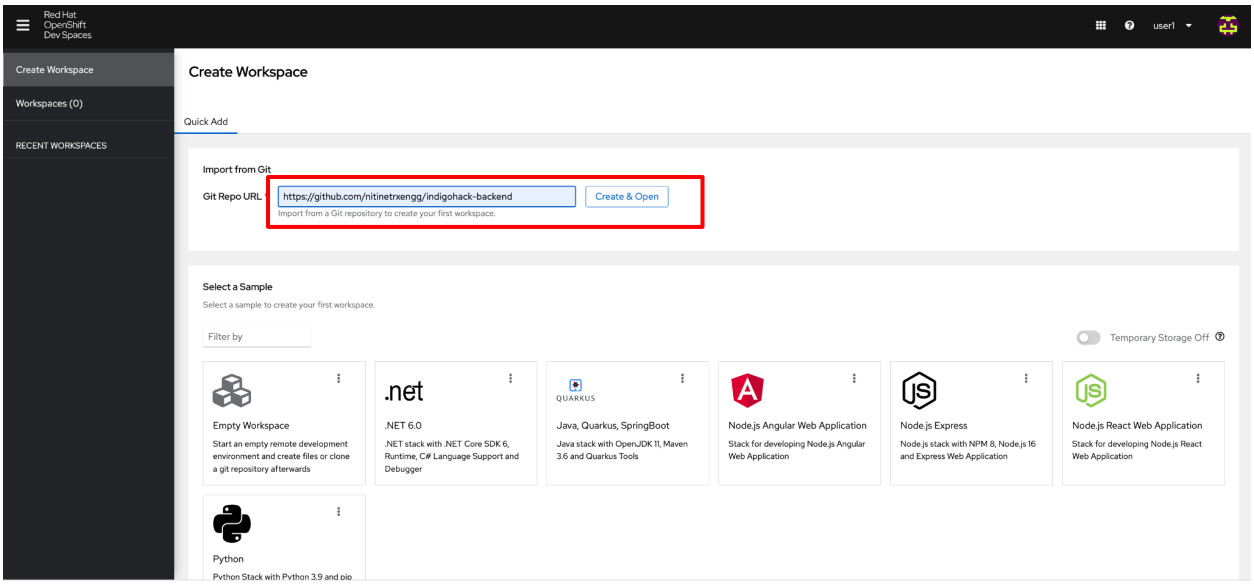


5. It will display the default Red Hat Openshift Dev Spaces console with the default development stacks.

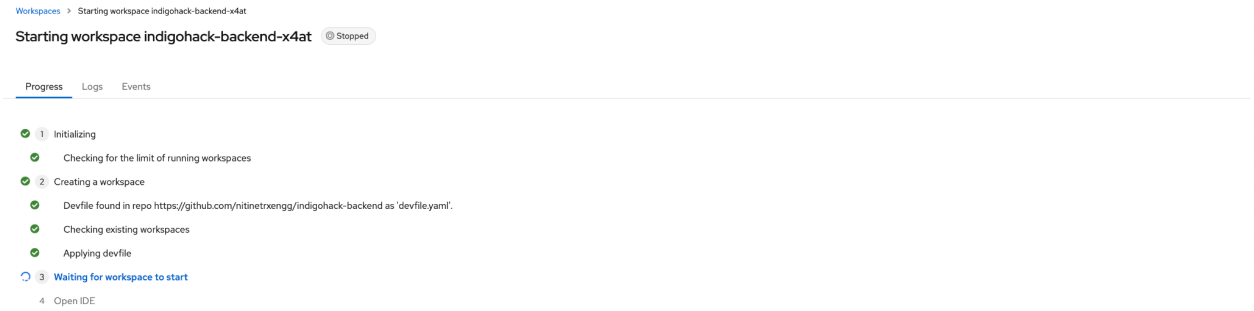


As already you have code in your existing GitHub repositories for backend & frontend development, you can simply clone your GitHub Repository to create a workspace for development in Red Hat Openshift Dev Spaces

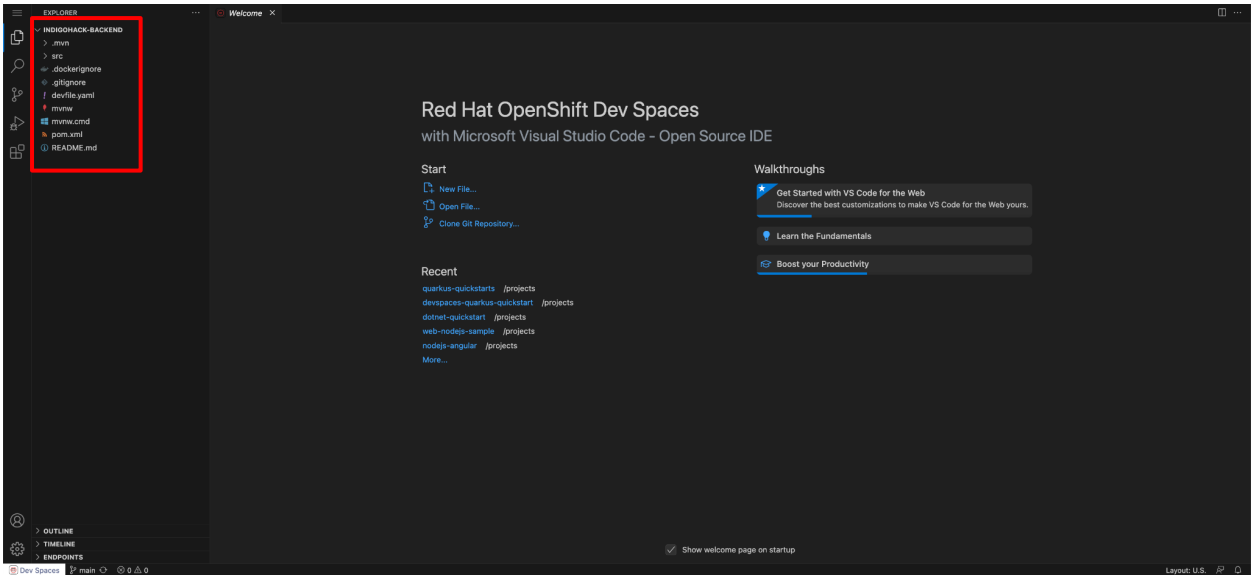
6. Create a workspace by cloning the GitHub repository of backend code & then click on “Create & Open”. In this example, quarkus (Java) has been used as a coding language for backend development.



1. This will spin up a process to create a workspace for you to perform your development & write your code as shown below



This is how the workspace will be available for you to perform your changes in the code & start your development. The GitHub repository is cloned in the workspace as shown below.

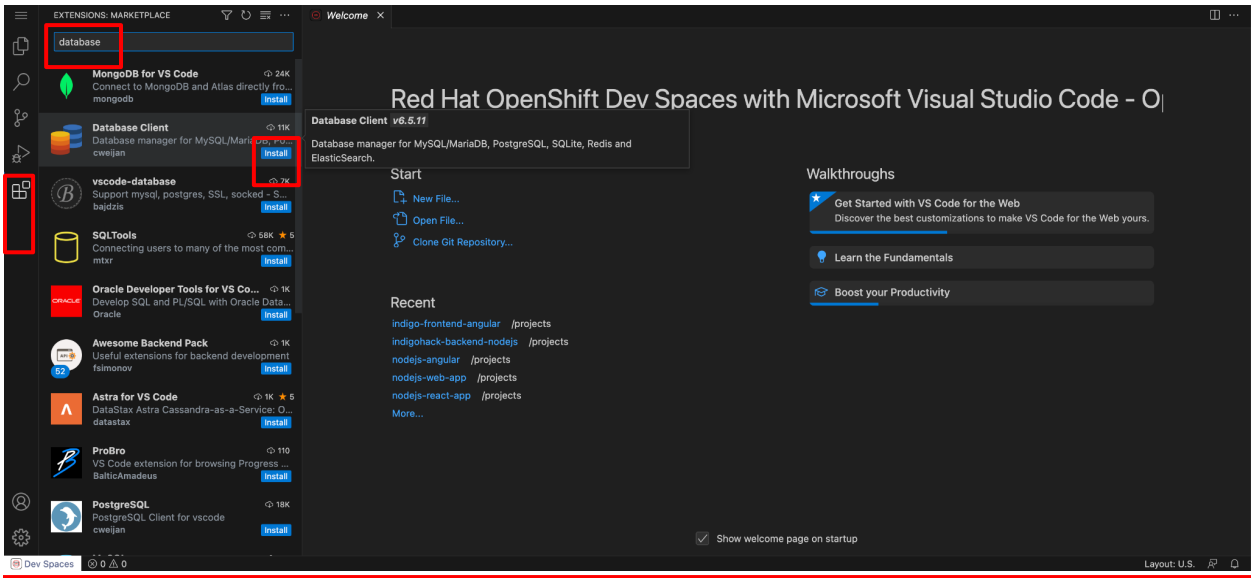


2. Before proceeding with the code development for your backend, follow the below two steps
    a. Configure Database connection properties in your backend code as per the database configurations defined in database deployment step earlier.

b. Install the database client depending as per the database has been deployed earlier in the database deployment step guide. This DB client will be used for performing CRUD operations on your database .

For connecting to Database from IDE (MySQL & PostgreSQL) - Install Database client. Click on Extensions, type database & Click on Install for Database client Plugin for MySQL or PostgreSQL.

Note: In case of MongoDB, please scroll down for the MongoDB section



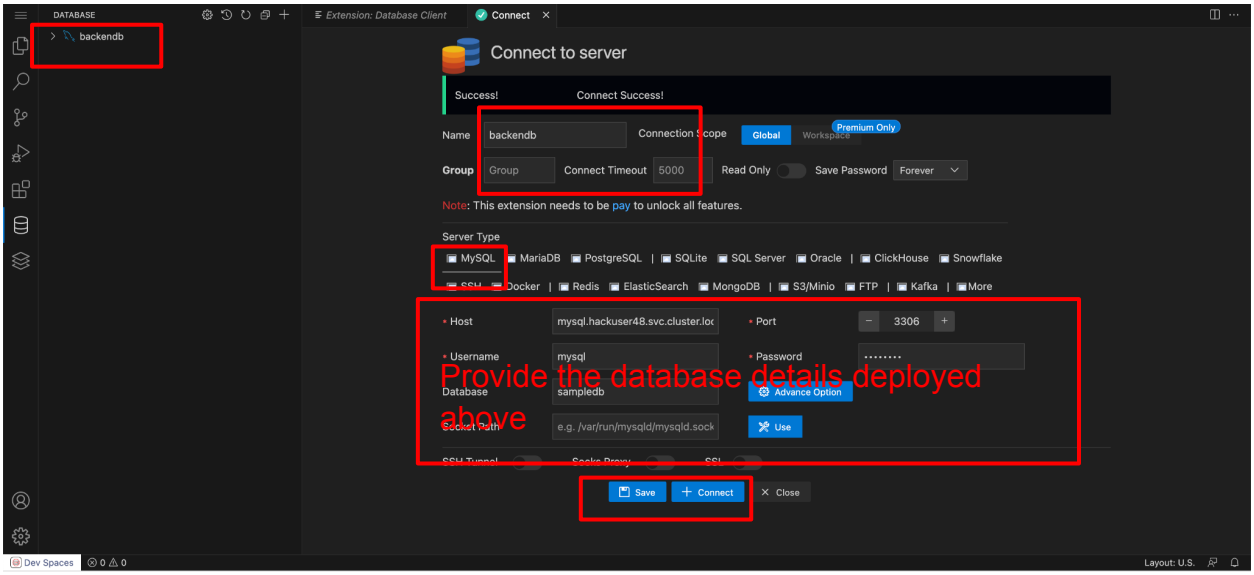
Successful installation of Database client will show the below screen

Now click on database on the IDE & click on Create connection as shown below

Provide the details of the database which you have created in the section “Deploying Database” & select the right database either MySQL or PostgreSQL. In connection details enter the following details & refer screenshot below

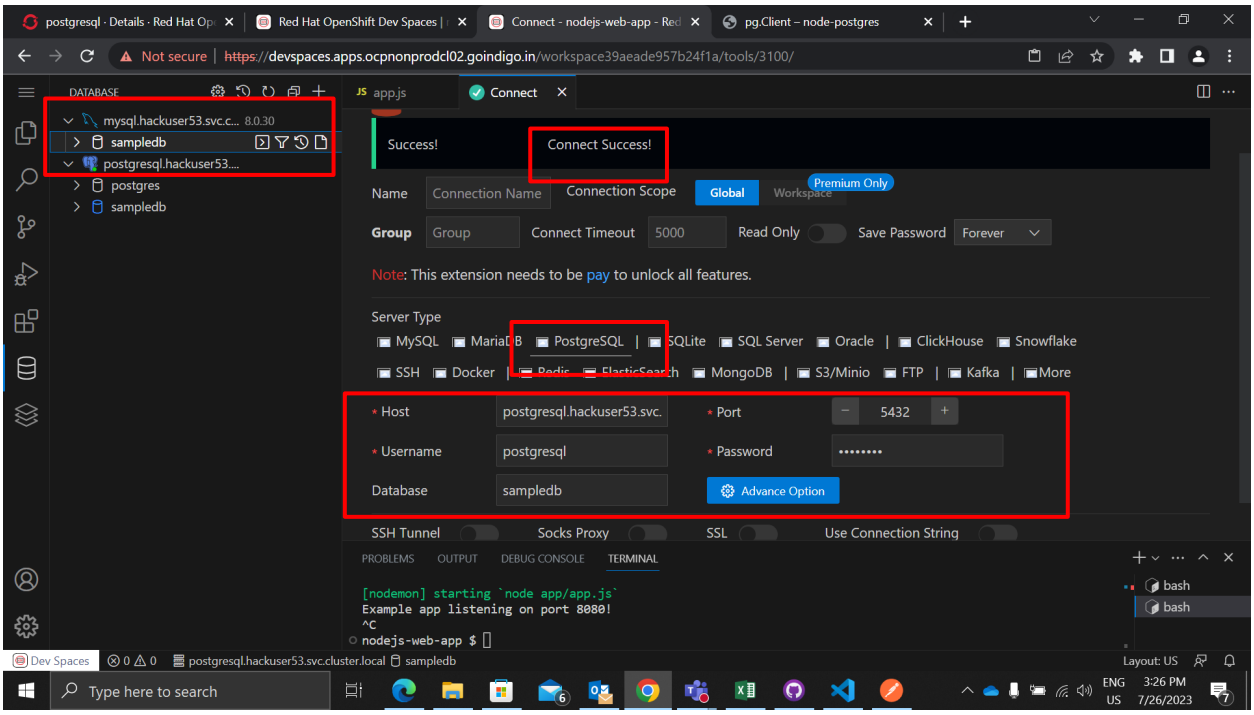
- Name: **You can provide any name here**
- Host - **mysql or postgresql.<YourHackuser>.svc.cluster.local**. Provide the database host name depending which database you have deployed above.
- Username - **mysql or postgresql** (depending on which database you deployed above)
- Database - **sampledb**
- Port - **3306 for MySQL or 5432 for PostgreSQL**
- Password - **password**

Below is MySQL Screenshot.



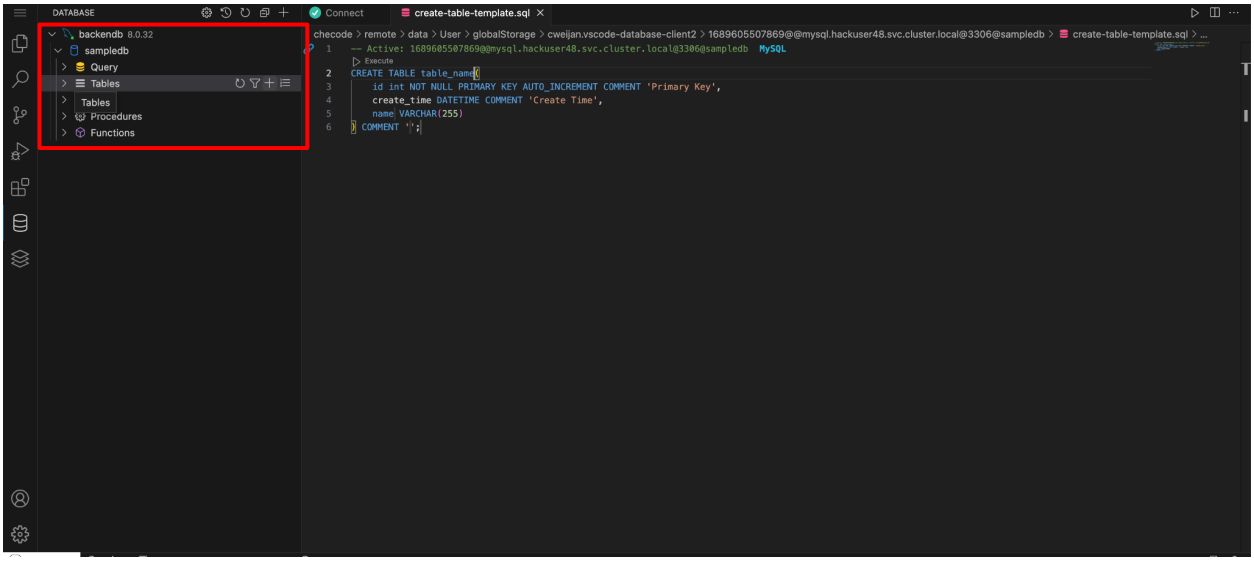
OR

PostgreSQL screenshot



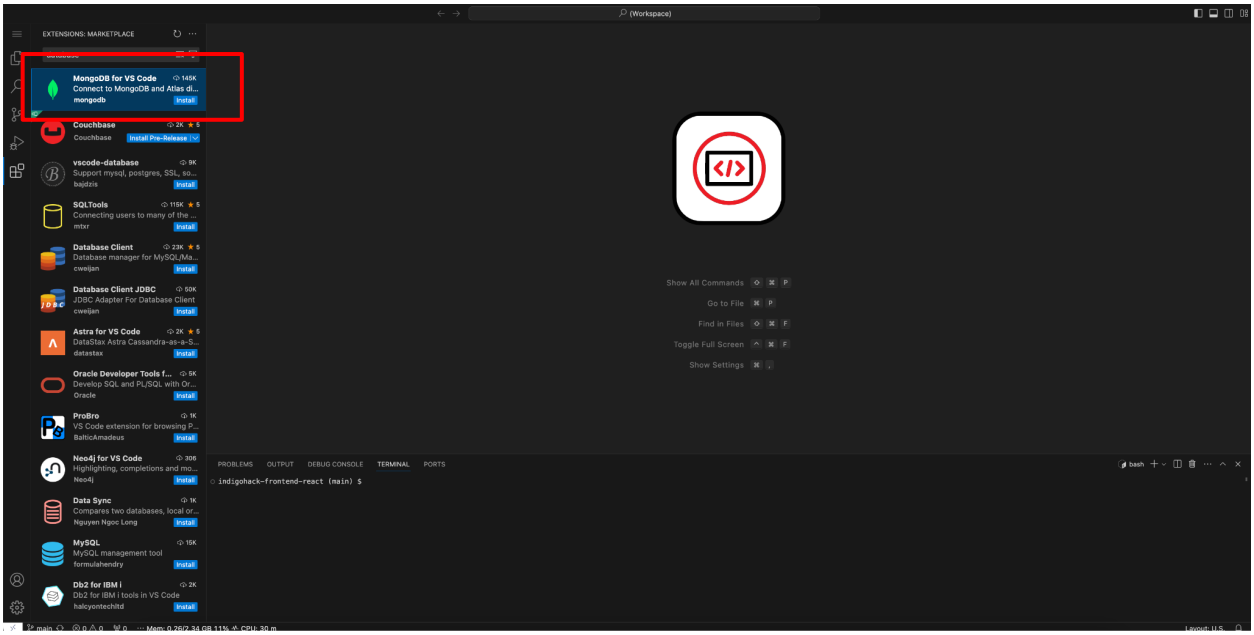
Post providing the details, Click on Save & Connect, it will show the “Connect Success” Message & will display the database connected on the Left hand side of the IDE.

Expand the database & create tables & execute statements on the database as per the requirements.

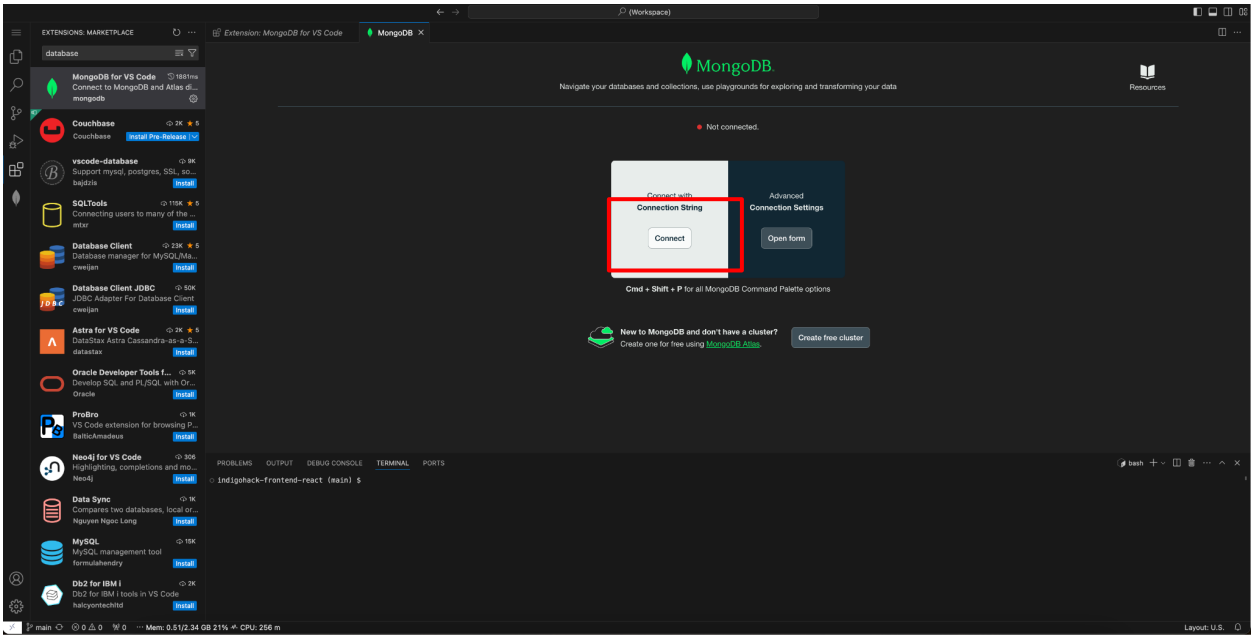


For connecting to Database from IDE (MongoDB)

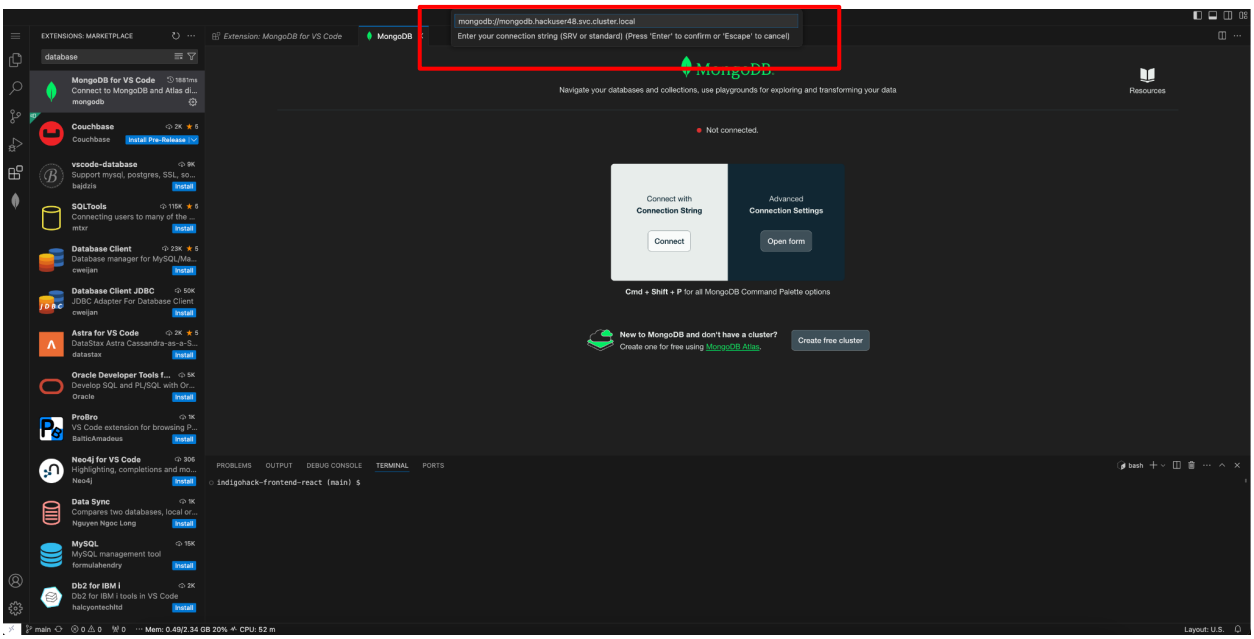
Install the MongoDB client plugin as shown below



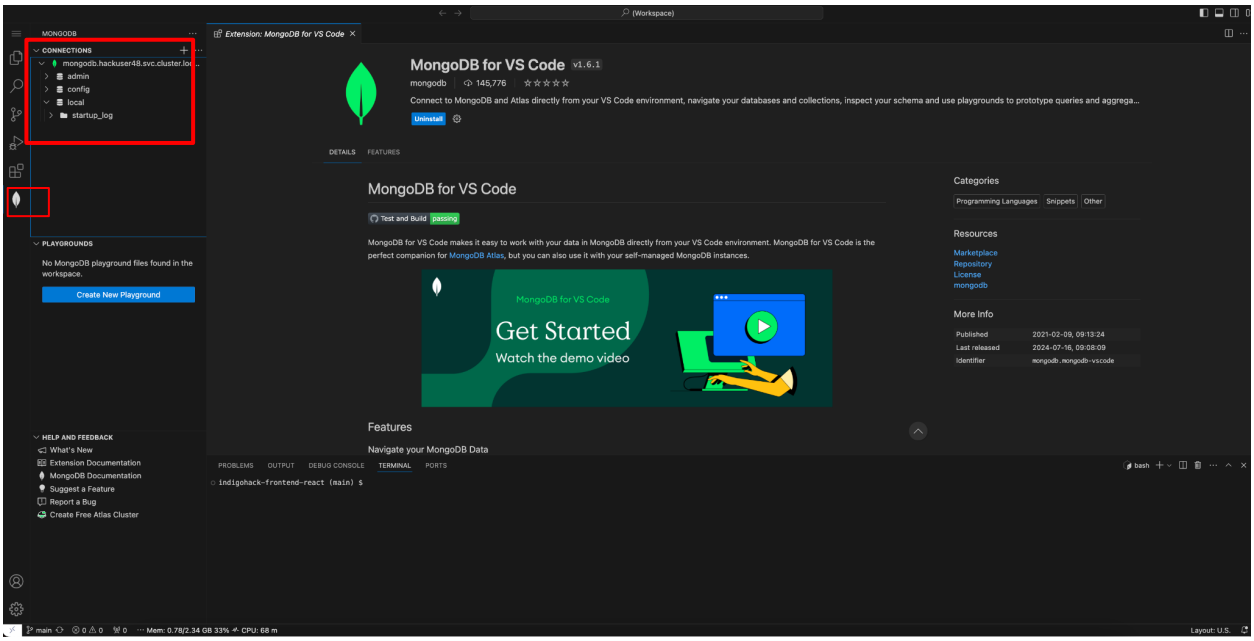
On successful installation, it will show the below screen, click on connect



Enter your mongodb connection string as shown below. Replace hackuser48 with your hackuser<number>

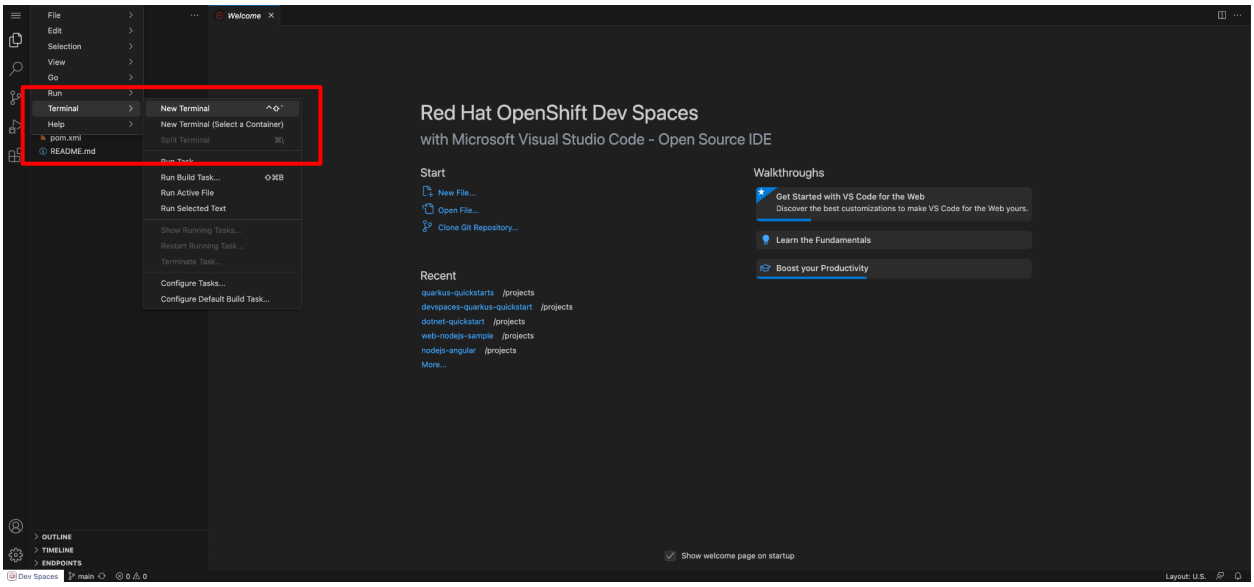


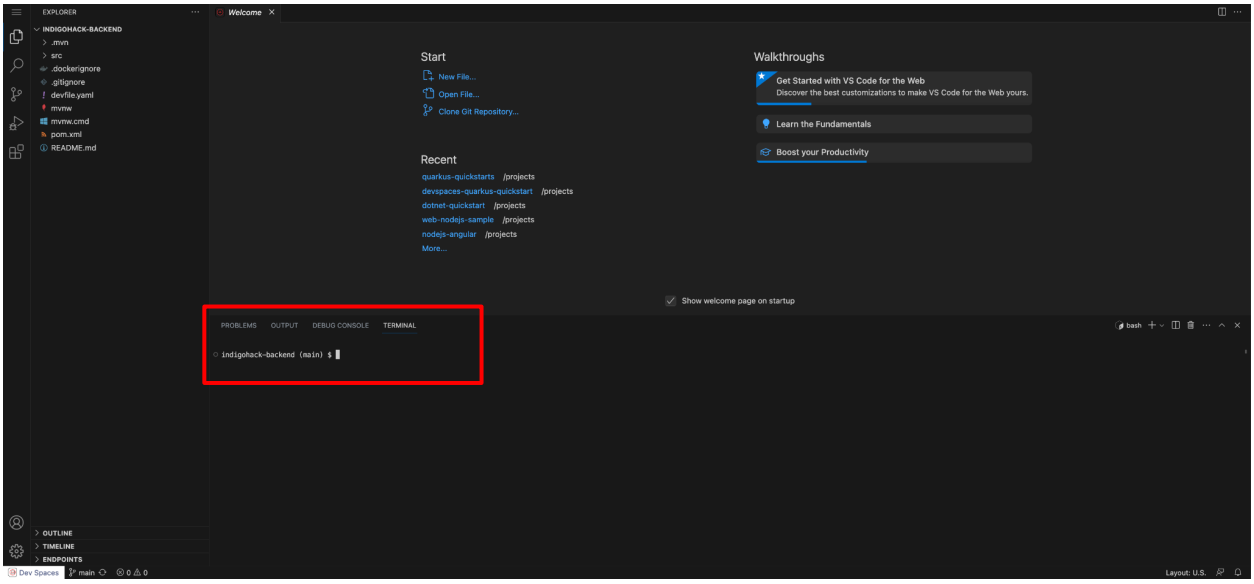
On successful connection, it will connect to MongoDB database & then you can perform CRUD operations on database as required



Now the backend code is available & DB client is also configured, perform the coding & do necessary builds & run the code for testing

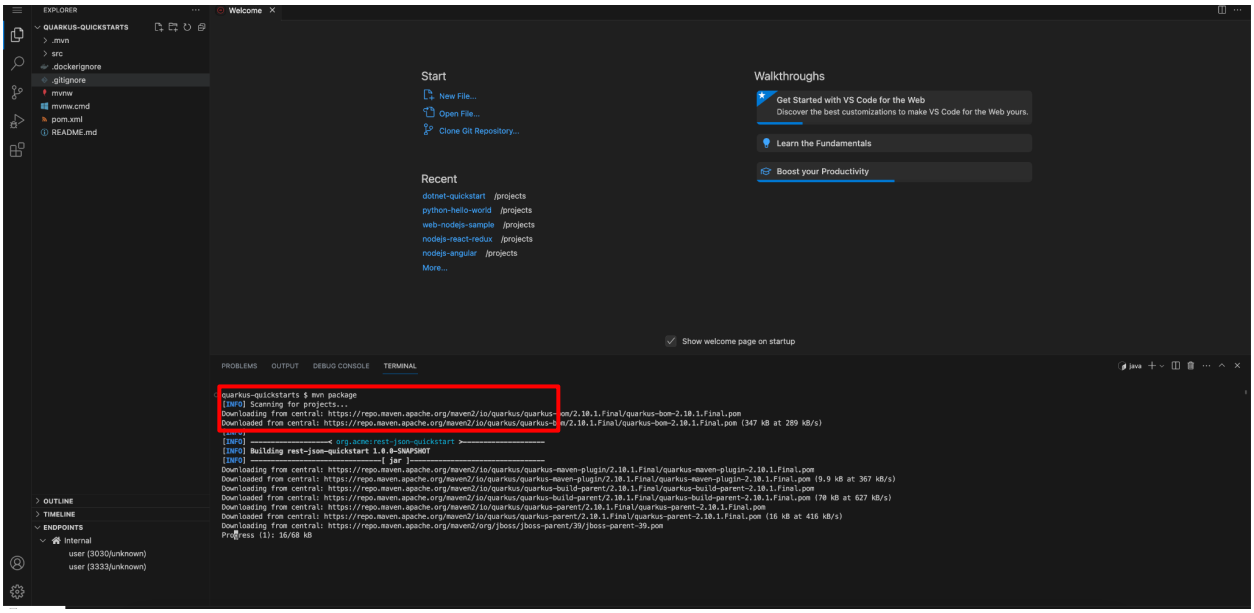
3. If the terminal is not visible in the workspace, then follow the below screenshot to open the terminal. A terminal will appear at the bottom of the screen to execute the build & running the code locally.



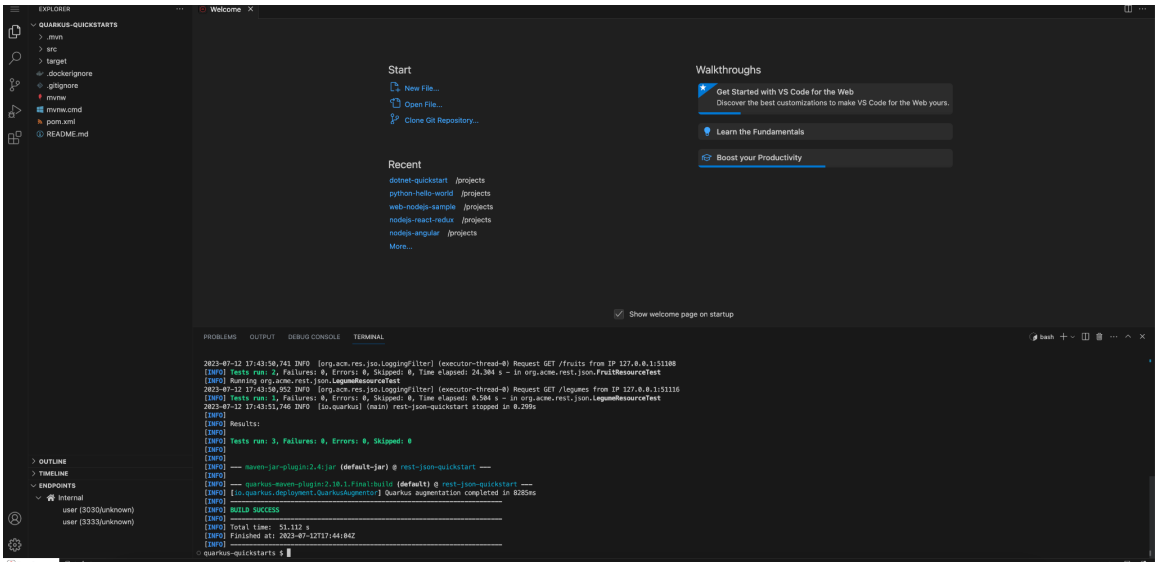


4. Now as you have begun developing the code & you can build & run it locally.

   Build code - To test the code, you can execute standard build commands in the terminal to run your code. I have shown the quarkus (java) build command in the example below. But you can choose the build command as per the code language opted.

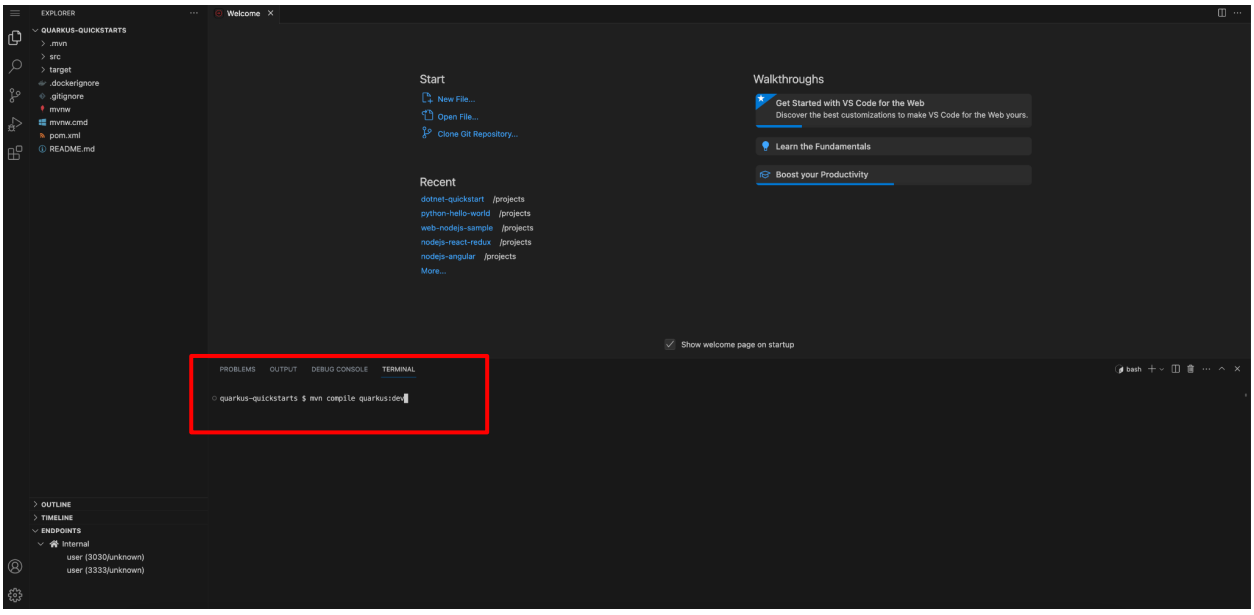


It will display the message for build success in the terminal.

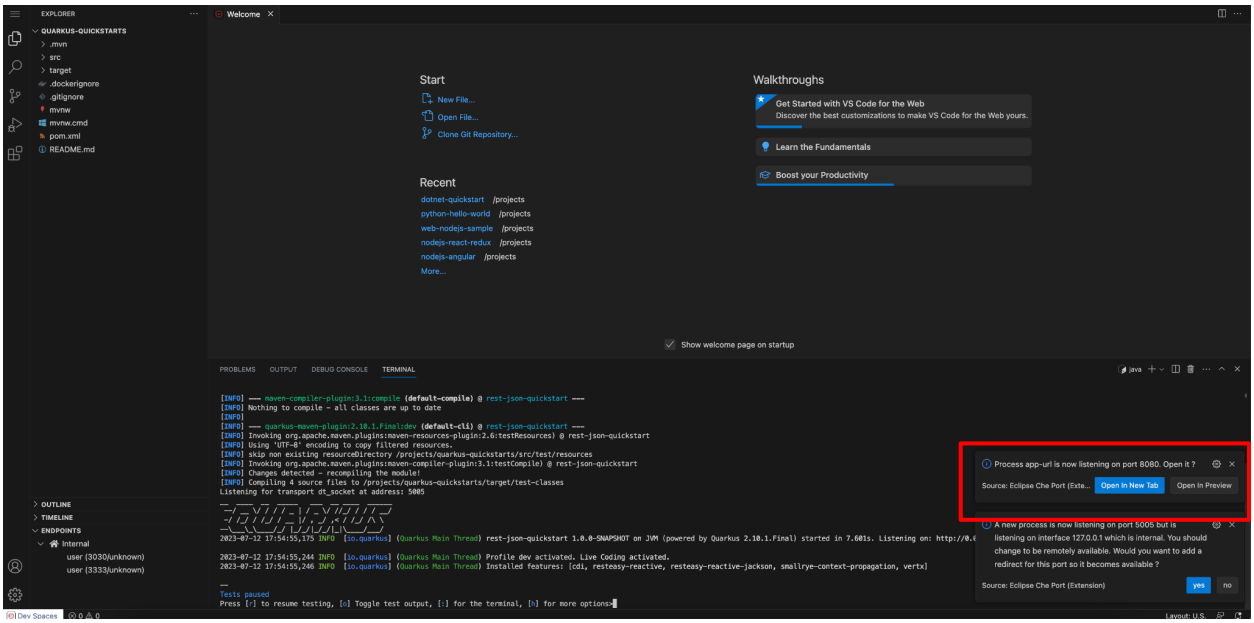


Run Code Locally - Post build, you can run the code locally to test the application. Post run, it will display one prompt which is the application URL to test the application.

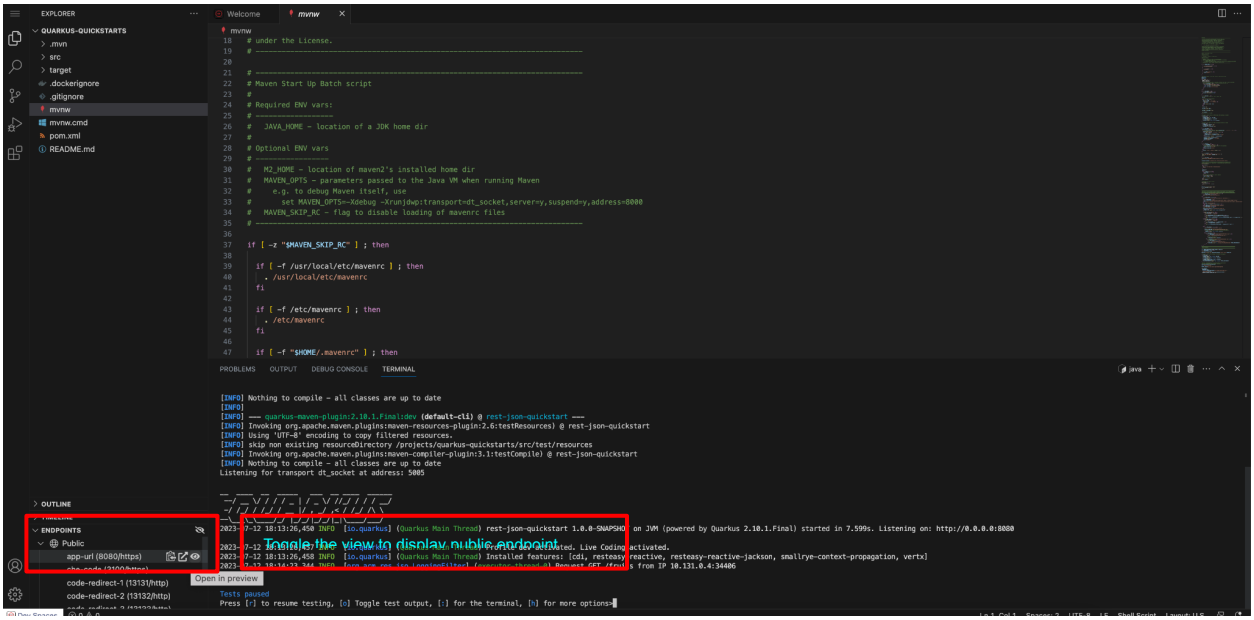
Executing the code by executing the command directly in the terminal. This example is for Quarkus (Java) & this command will differ as per the coding language chosen.



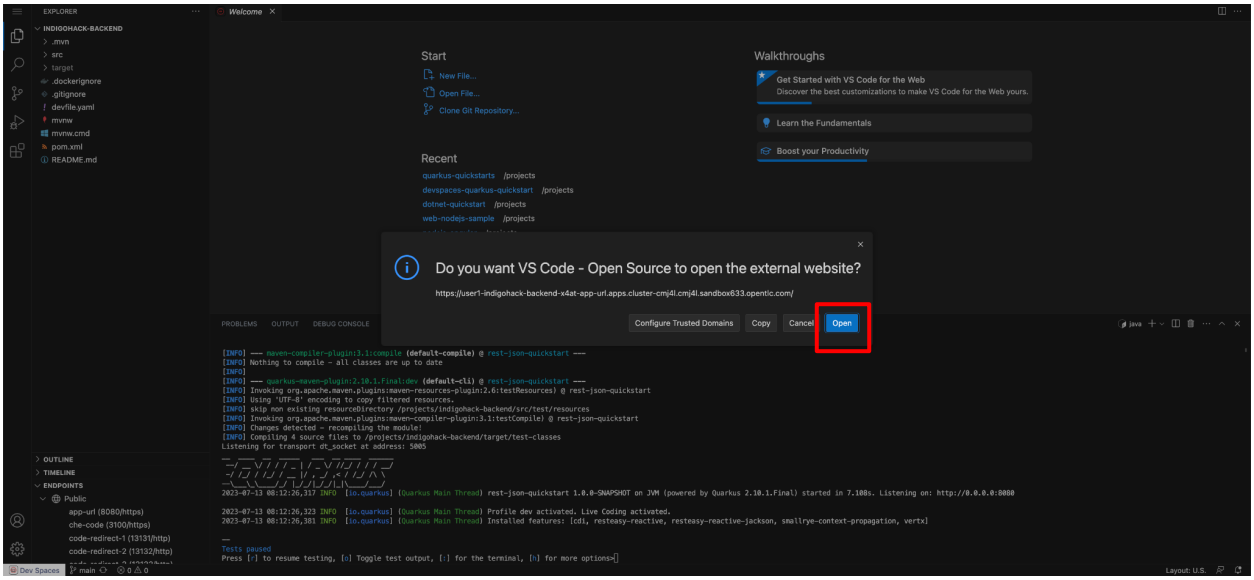
On executing the code, it will display the notification with "app-url" which is the endpoint or URL to test your application as shown in the screenshot below. Click on "Open in new tab" & it will display the web page of the application.



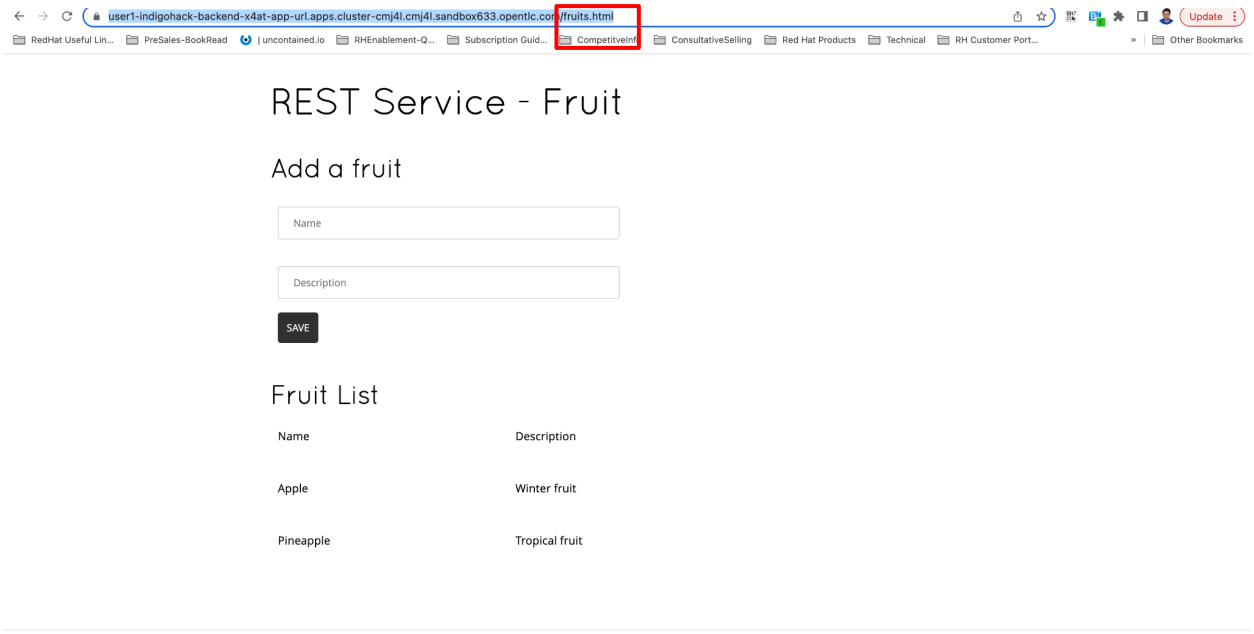
Optional - If you miss this app-url pop , then you can also access this app-url endpoint as shown below. Click on endpoint



5. Click on Open



6. It will open the application in a new tab.. I have appended my example context path (fruits.html) as per this example application to access the API/web page . Below screenshot shows the web page for this example Java application



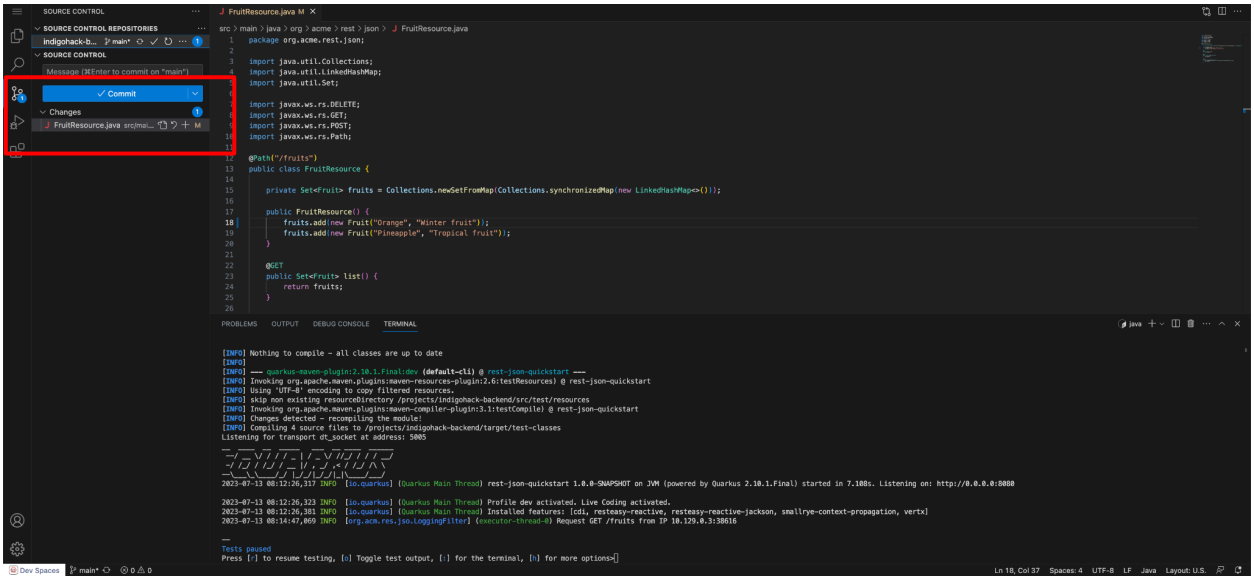
Important Point - The above URL displayed is your application endpoint. Like in above example the application endpoint URL is https://user1-indigohack-backend-x4at-app-url.apps.cluster-cmj4l.cmj4l.sandbox633.opentlc.com/fruits.html

In your case it may be different. But keep a note of it. In your case, the application web page may not come as you may be developing backend application APIs without any front web page.

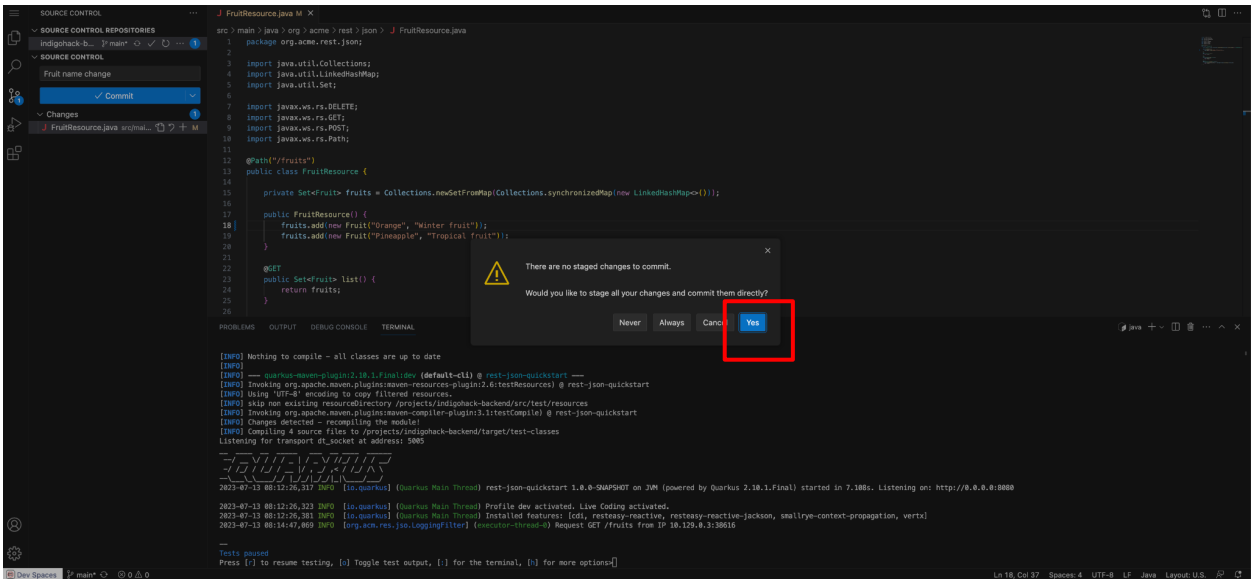
Note: As ideally you are developing a backend application so it may not have a web page. So in that case the above "app-url" will become the application endpoint for your backend APIs so you can test using Postman on your laptop. In postman, you have to use the above "app-url" endpoint with ur APIs context path to validate backend APIs.

7. Now as you have performed the changes & these changes have to be pushed to GitHub repository.. In the below screenshot, I am just showing as an example on how to change the code & commit to GitHub repository.

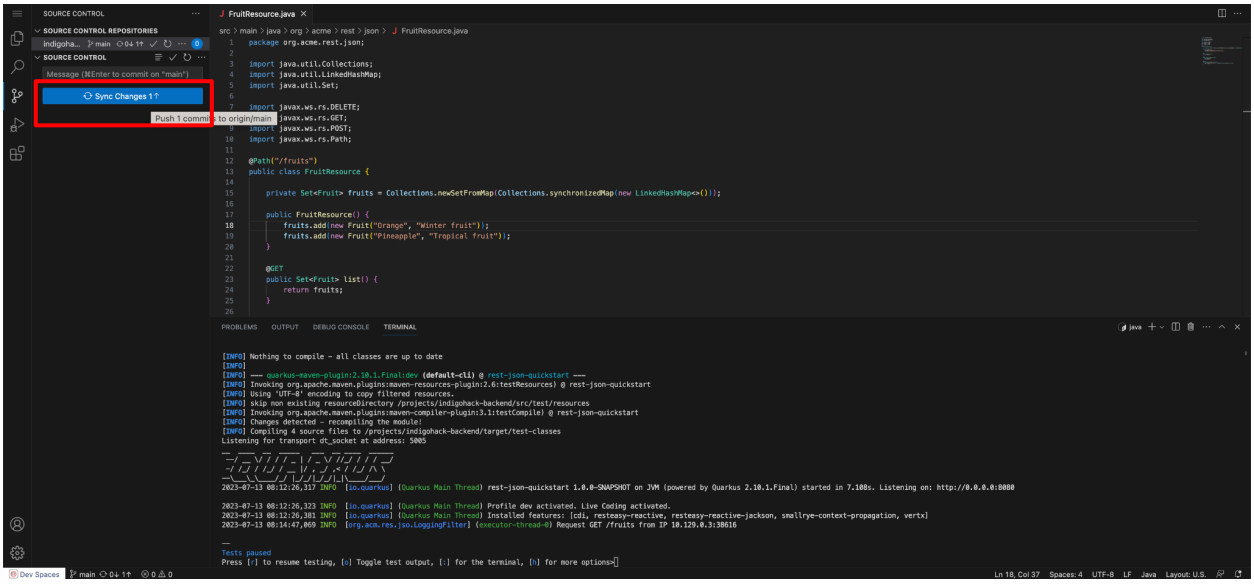
8. As the code will be changed, the inbuilt Git client will show the changes to commit the code with the filename where changes have been made.

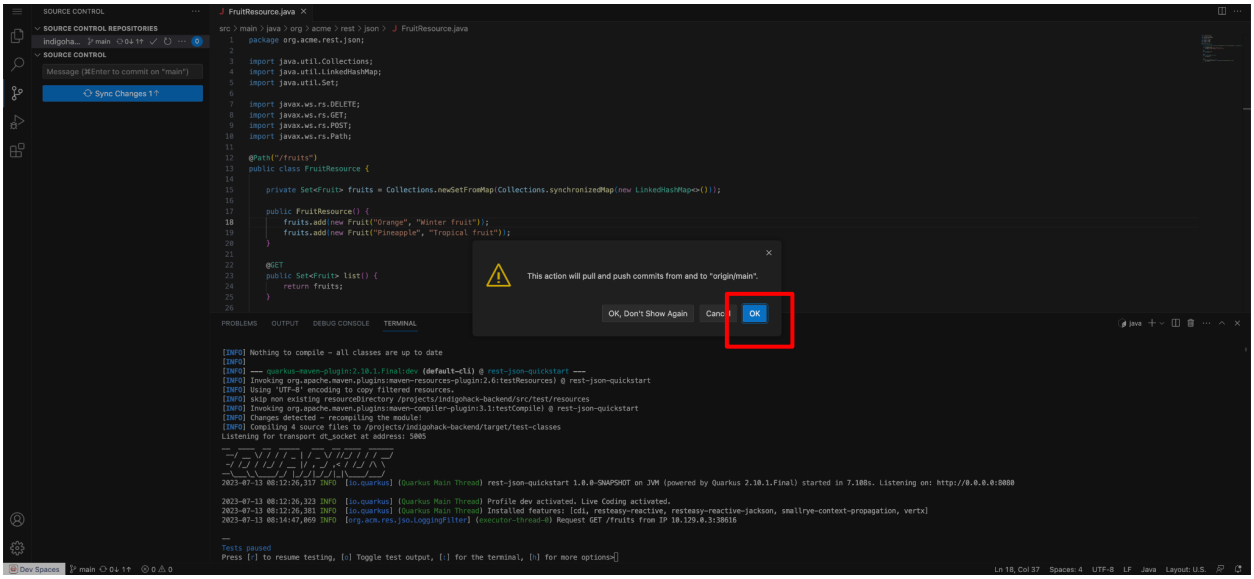


9. Now write the commit message & click on Commit to prepare the code to sync with the remote GitHub repository. Click on Yes to push all the changes .

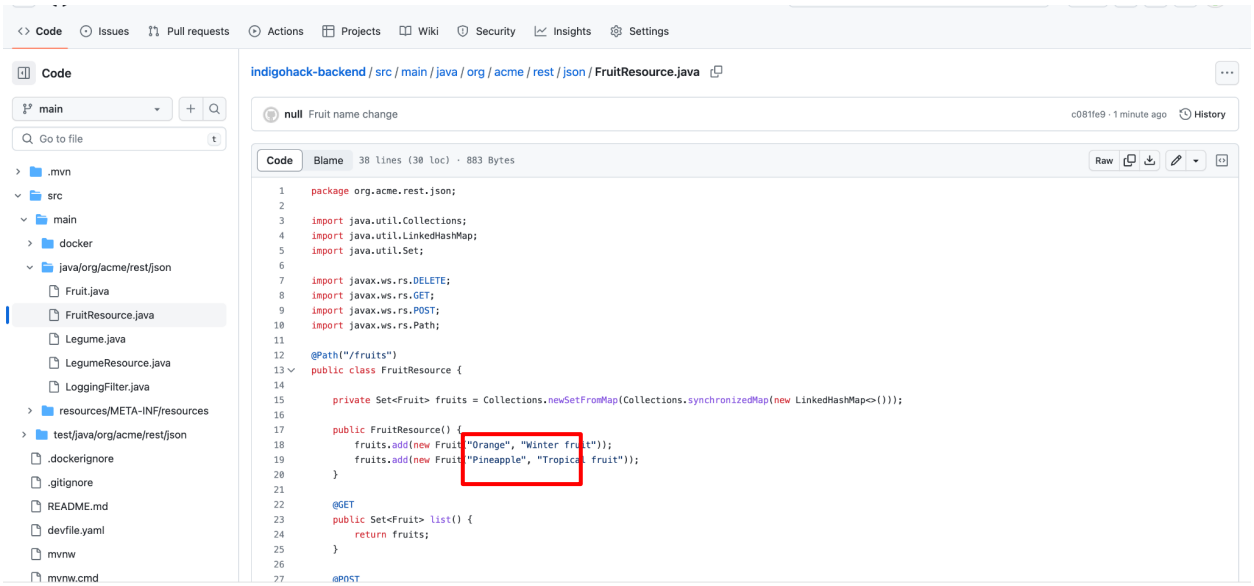


10. Click on Sync to push the changes to your remote GitHub repository. Click on OK in the next screenshot.





11. You can verify in your GitHub repository that changes have been pushed or not to confirm it.



Frontend code development

12. Now, you need to create another workspace in Red Hat Openshift Dev Spaces for your frontend code by cloning GitHub repository for your frontend & follow the same steps to perform development & committing the changed code to GitHub repository for frontend code.

Note: Do not forget to put the right devfile.yaml file at the root of your GitHub repository & then clone it to create the workspace.

Important step

Your frontend code requires the backend code connection details for frontend code to talk to backend code. In the next guide, you will be deploying applications for backend & frontend code.

But before that, you can perform the following connection configurations in your frontend code to talk to backend code deployment

Backend application name - indigohack-backend
Backend Port number - <port number that you have coded in your backend code>. Ideally it should be 8080.

Next step - Move to Guide 4-DeployCode

Post committing the code for backend & frontend in GitHub repositories, you would require to deploy these applications on Openshift Platform to test the complete full stack application availability.
So follow the guide 4 - Deploy Code for deploying your code applications on Openshift Platform.