Cloud Project
By Roy Gebrayel
202111125

Overview

1 - created all the resources : sql database , sql server , keyvault , webapp , functionapp , docker , acr , self hosted vm , terraform , test case , storage account

2 - created the web app in golang and connected it with the function app written in nodejs

3 - connecting the web app with the sql database on the login post api (written as a trigger in the nodejs function app)

4 - connecting the web app with the storage account on the upload-file api … the upload-file api uploads a file from your local pc to the storage account (written as a trigger in the nodejs function app)

5 - Written tasks in the azure boards and mark them as complete when i complete them

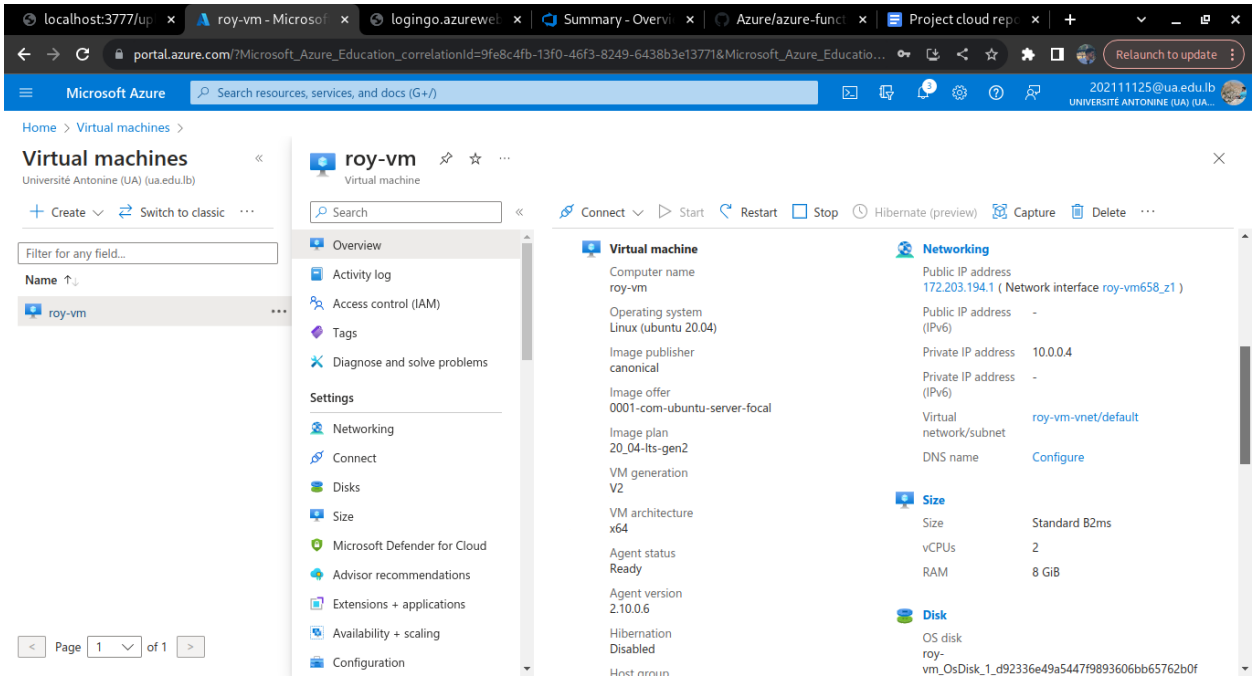
6- Written a pipeline that triggers on master the building and deployment to the acr and release of the logingo project

7 - Tried the test case

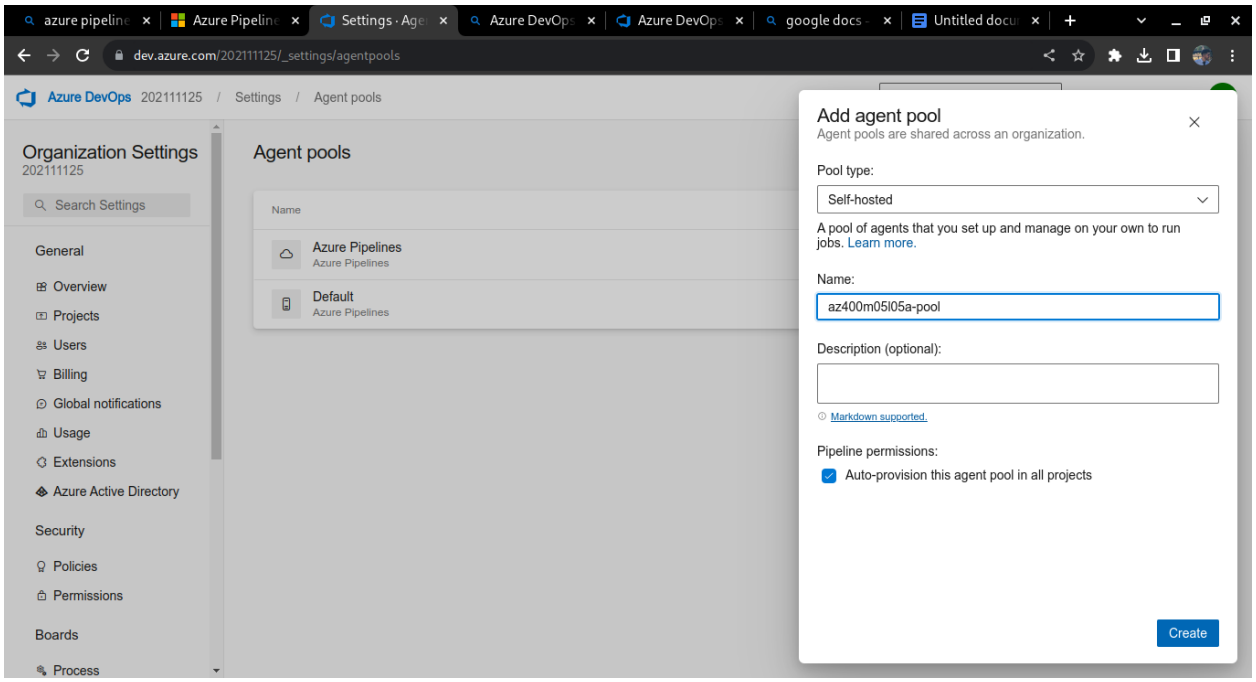
8 - tried to use terraform to understand it and managed it correctly

1 - RESOURCES CREATION

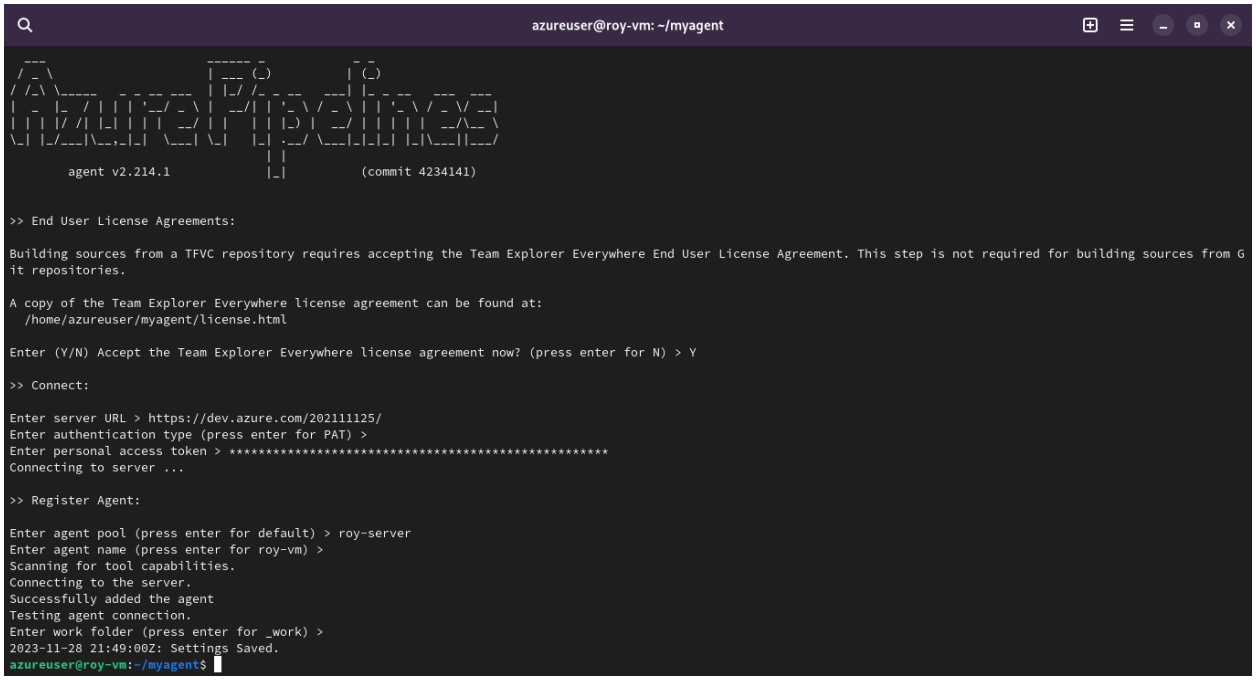
1 - vm creation :



Created the vm using ubuntu 20.04 linux os and configured to enable docker and install it

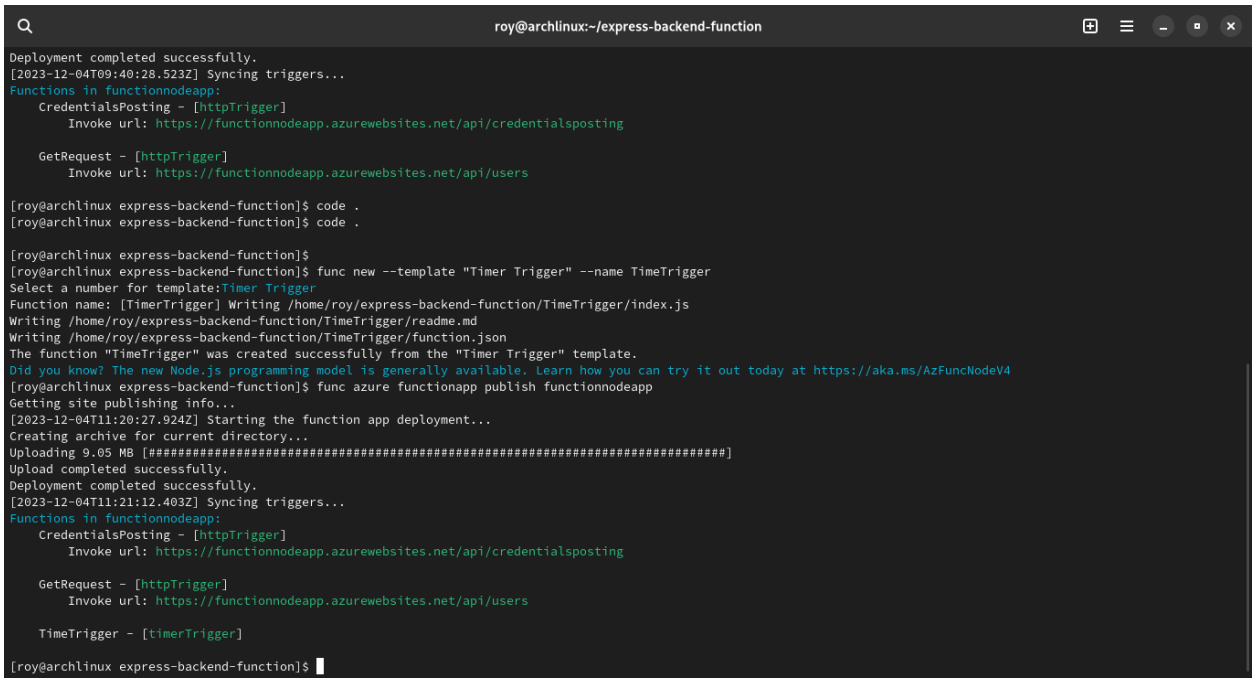


Add agent pool to the logingo

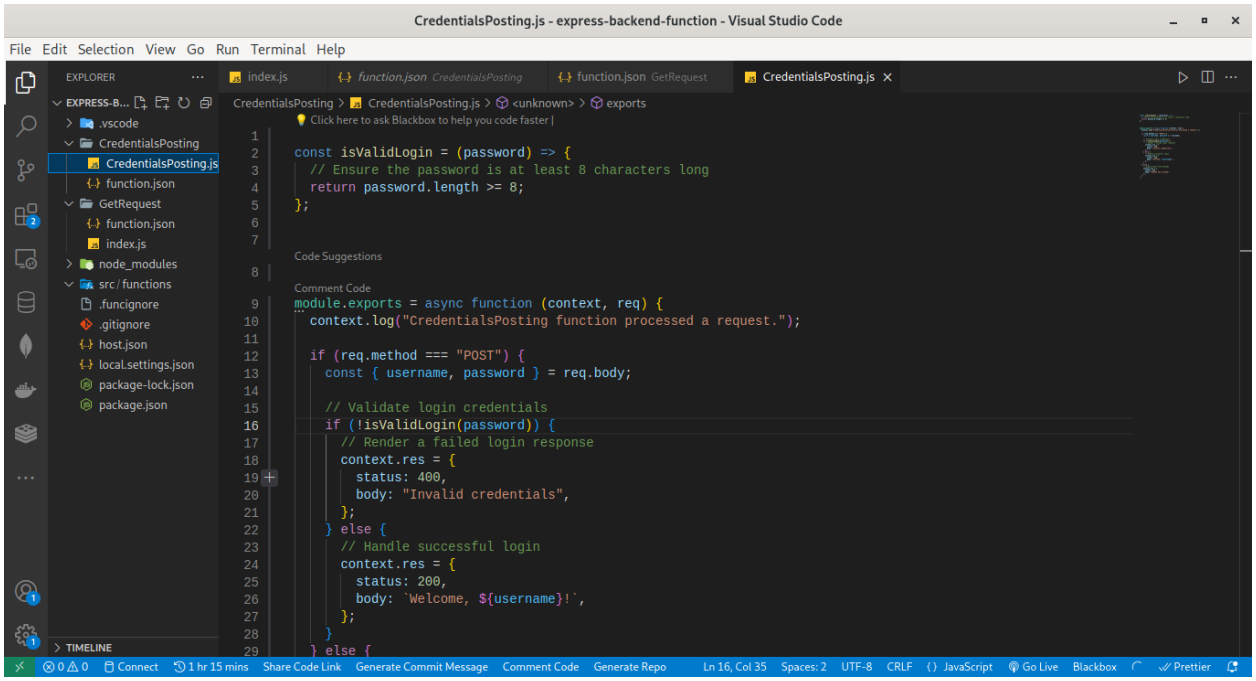


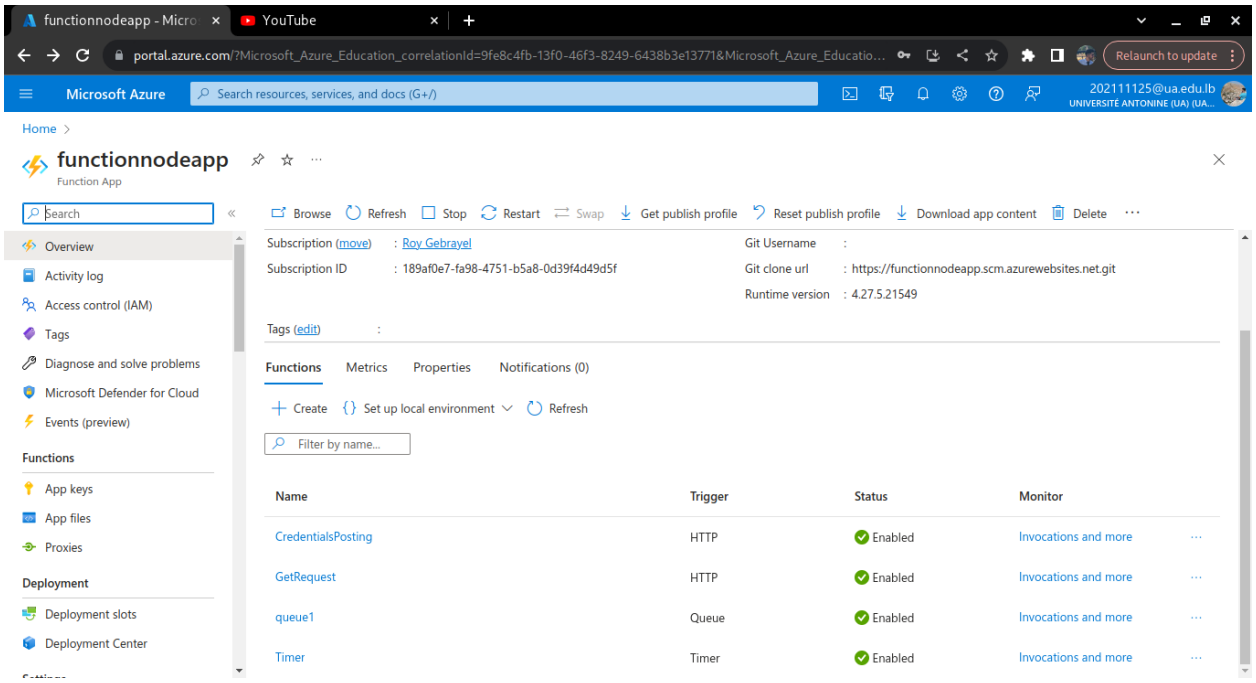
Connecting the self agent server to the logingo web app

2 - function app :

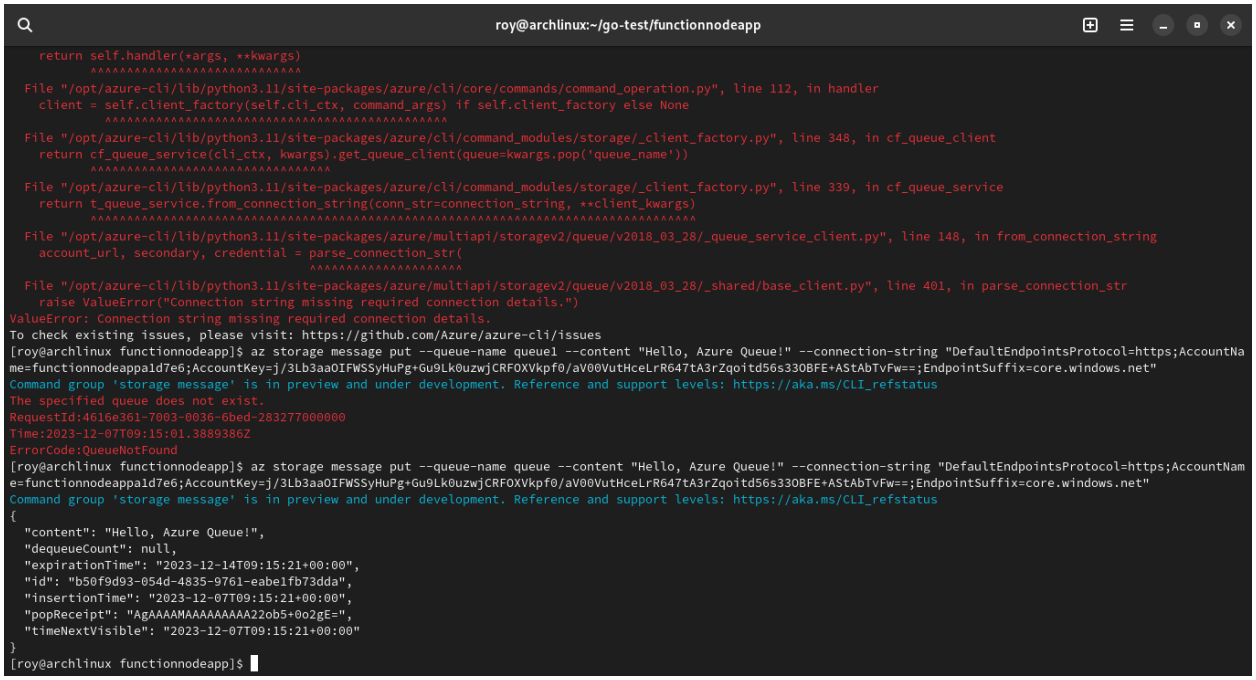


Created four triggers with the func in the terminal

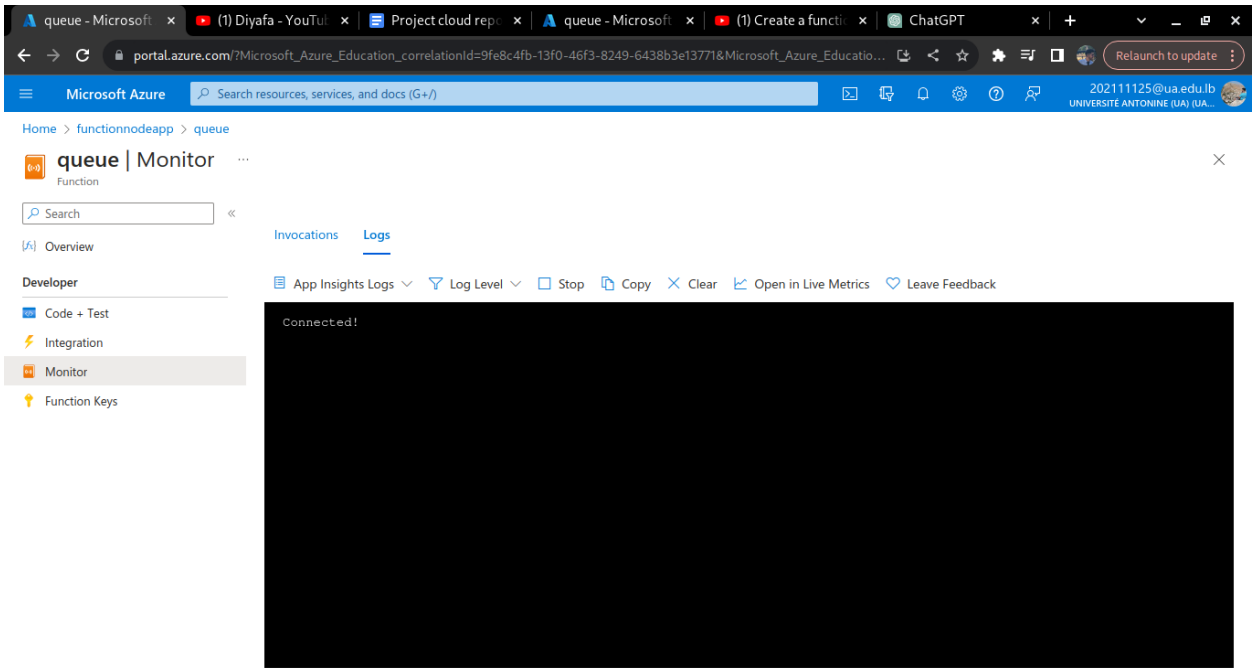




Created the function app in vs code with node js runtime stack and created four triggers CredentialsPosting (post http request) , GetRequest (get http request) , Timer (timer trigger) and queue1 (azure queue storage trigger)

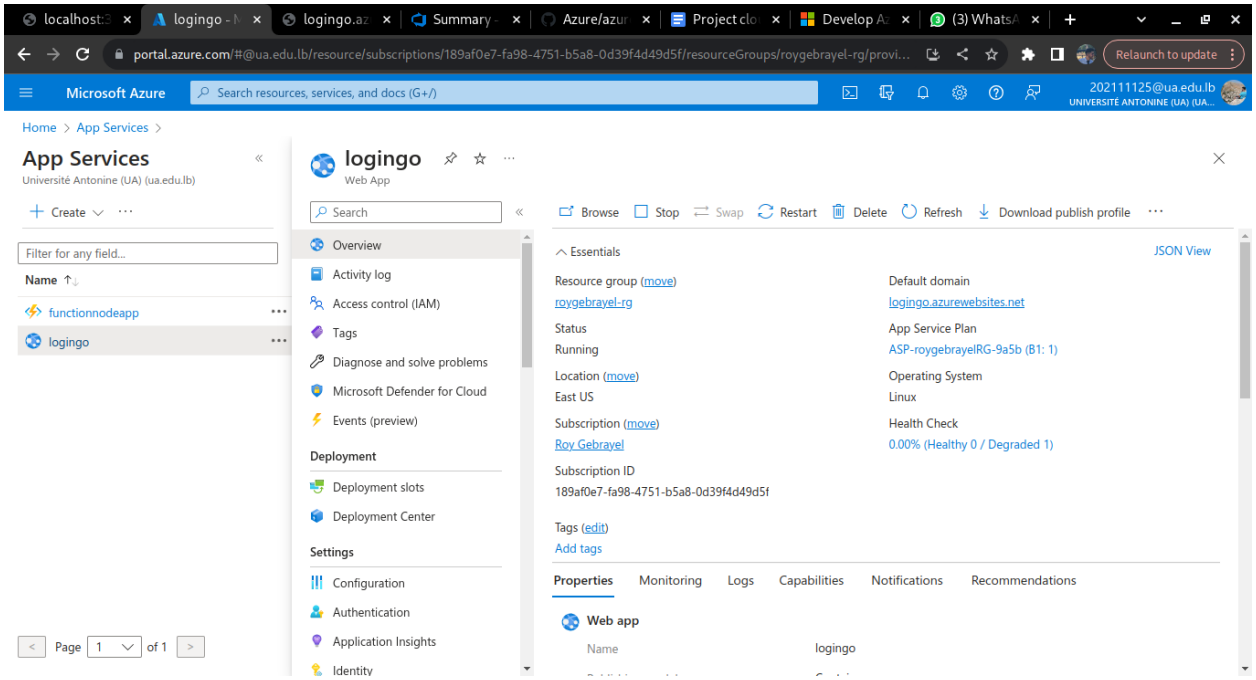


Sending message to the queue



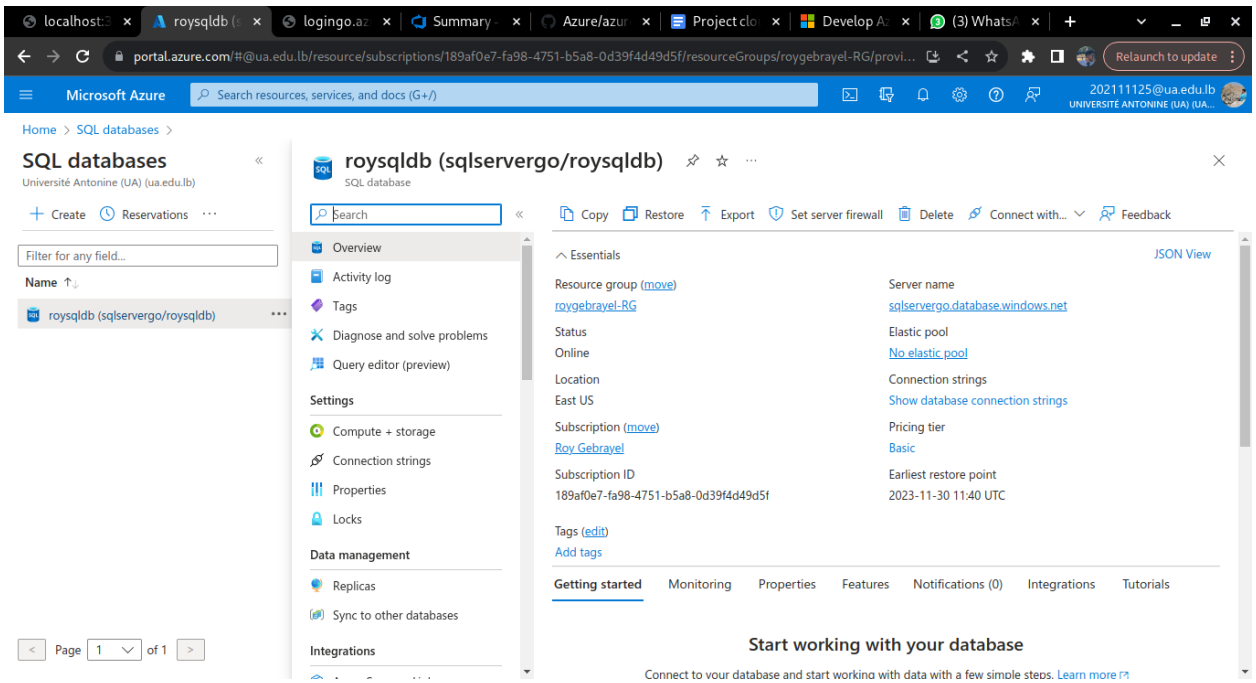
Here we see that the queue is connected

3 - webapp :



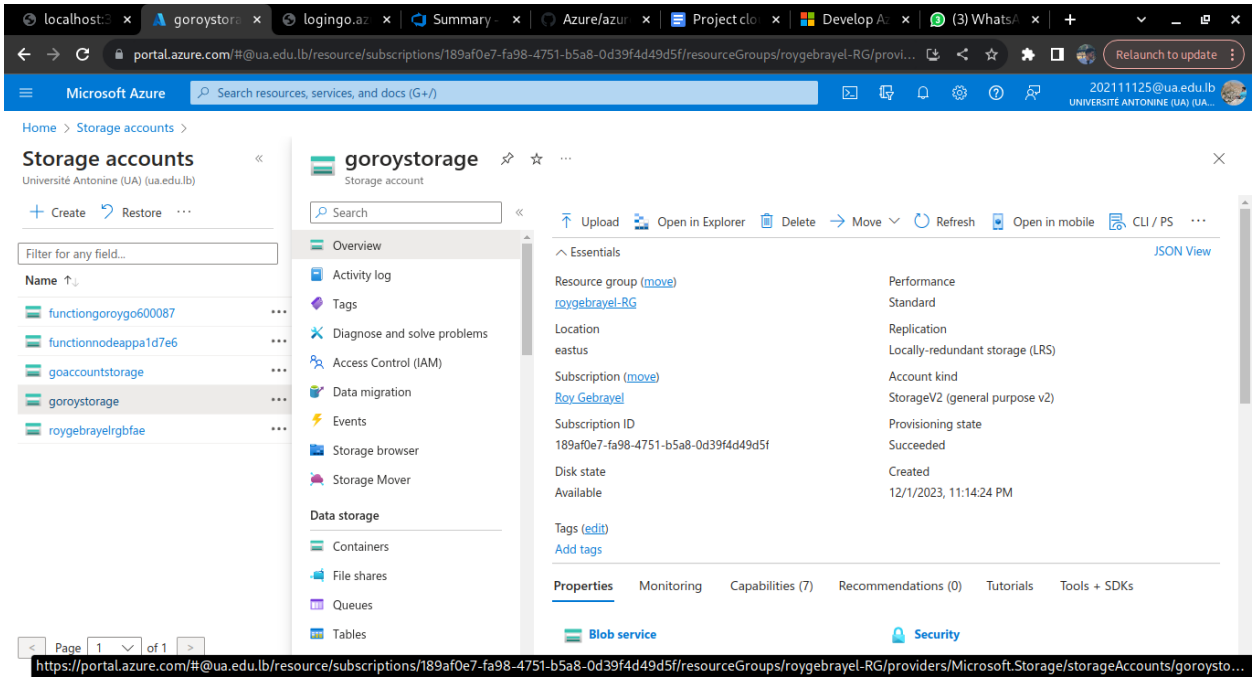
Created the logingo web app and written it in golang also built it with docker and pull it from the acr

4 - SQL database :

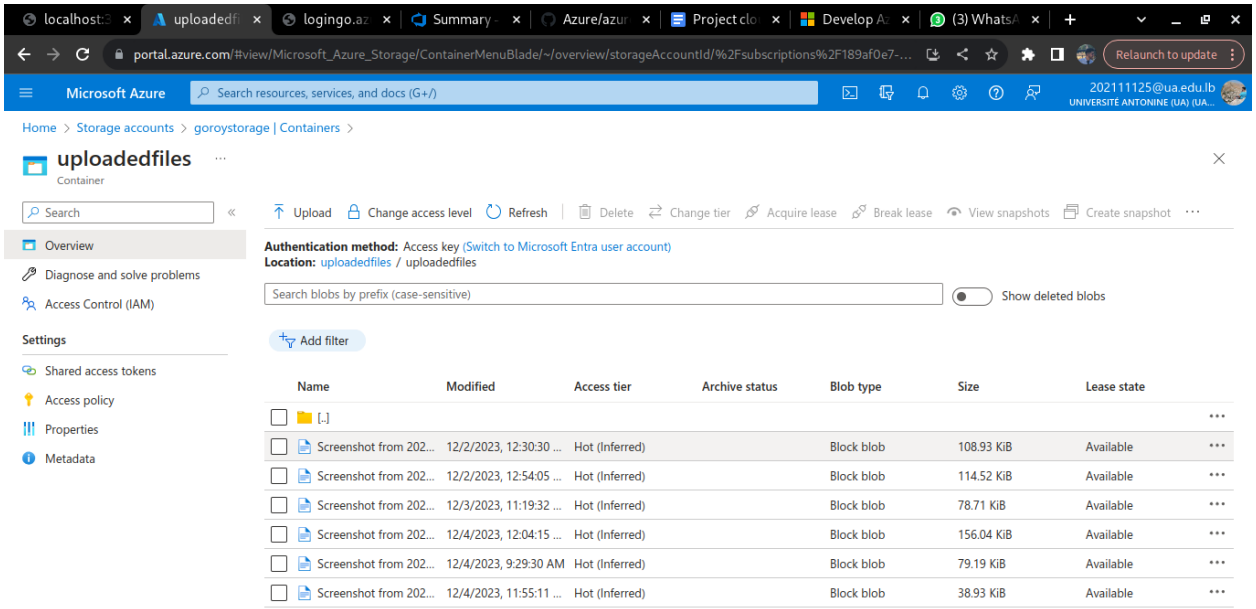


Created a sql db with the sql authentication password

5 - Storage account :

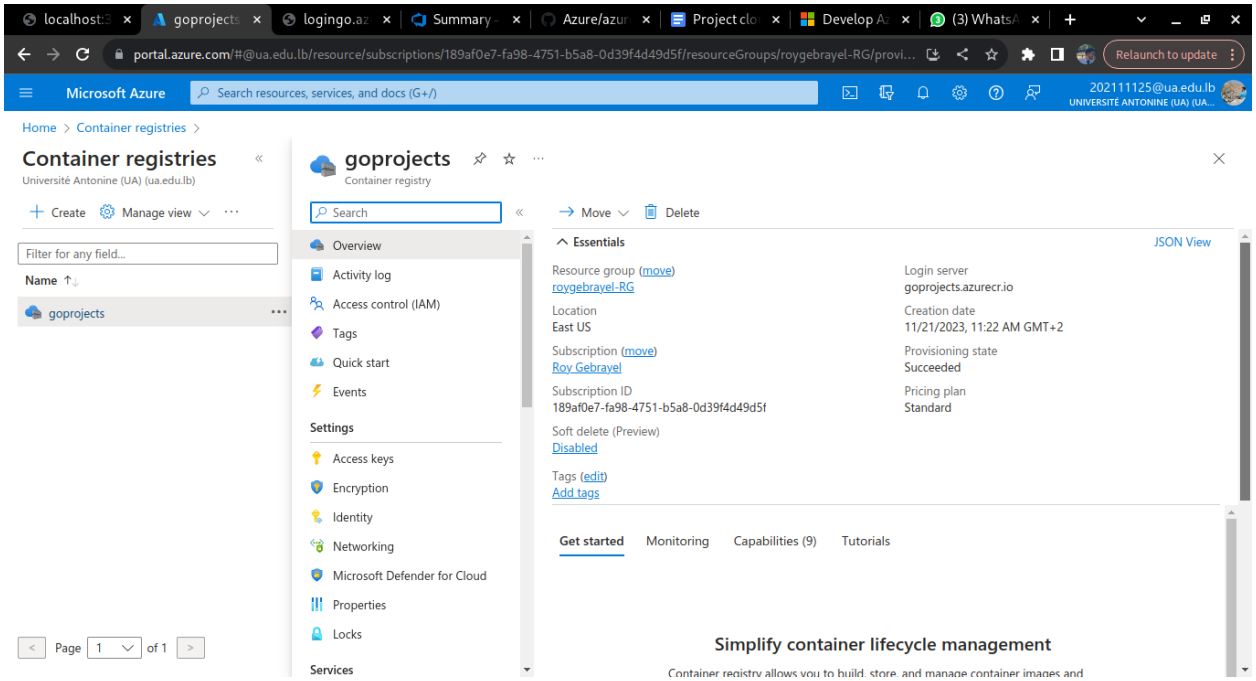


Created the storage account named go roy storage



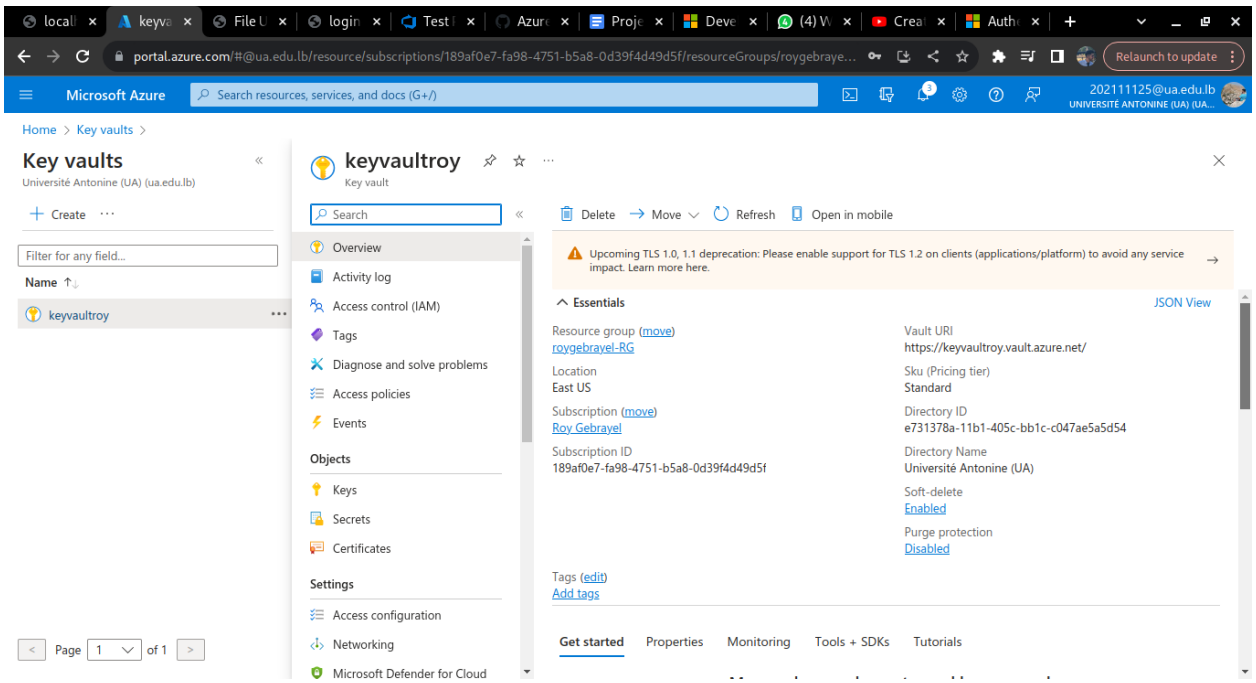
This is the uploaded files from the web to the storage account

6 - acr



Deployed the docker container into the registry

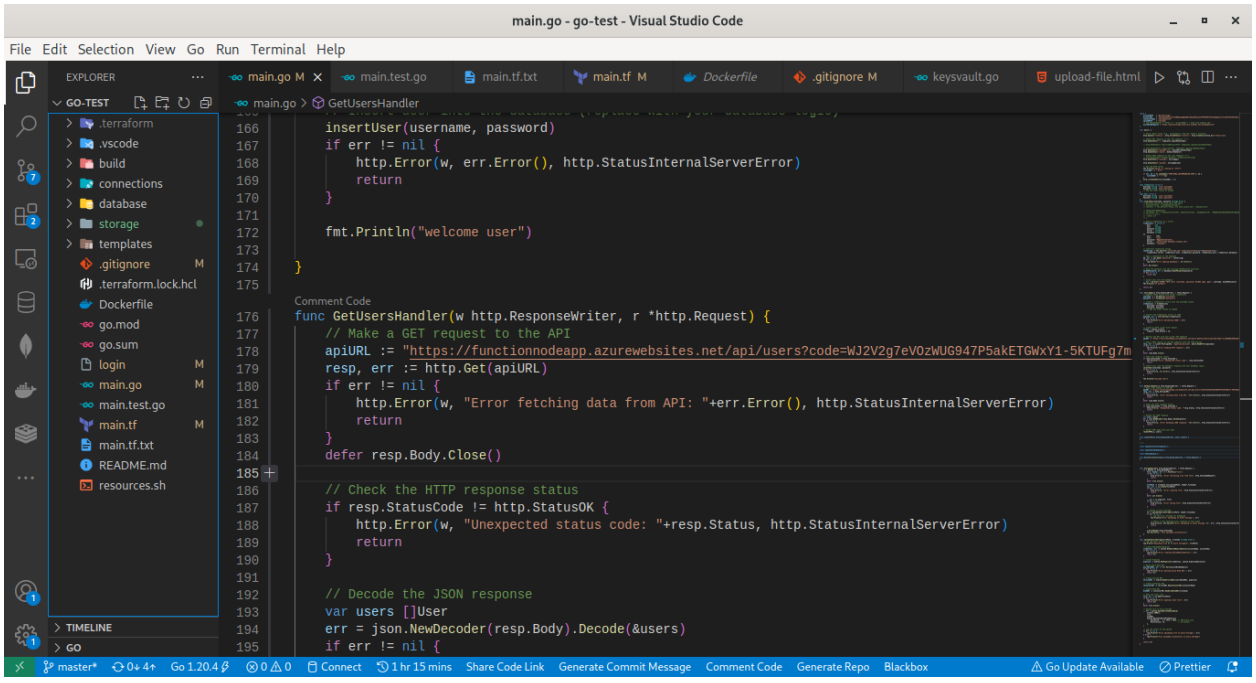
7 - keyVault :



Created a keyvault and inserted secrets

2 - Web App

1 - Content of the web app :



1 - this project is written in go :

Build folder : a folder to put all the yaml pipelines in the project

Connection folder : a folder having all the sql db connection strings to be able to connect it with the main.go file

Templates folder : a folder having the html templates for this project

Dockerfile : a file to write my docker configuration

Main.tf : a terraform file to try it and test how it is done

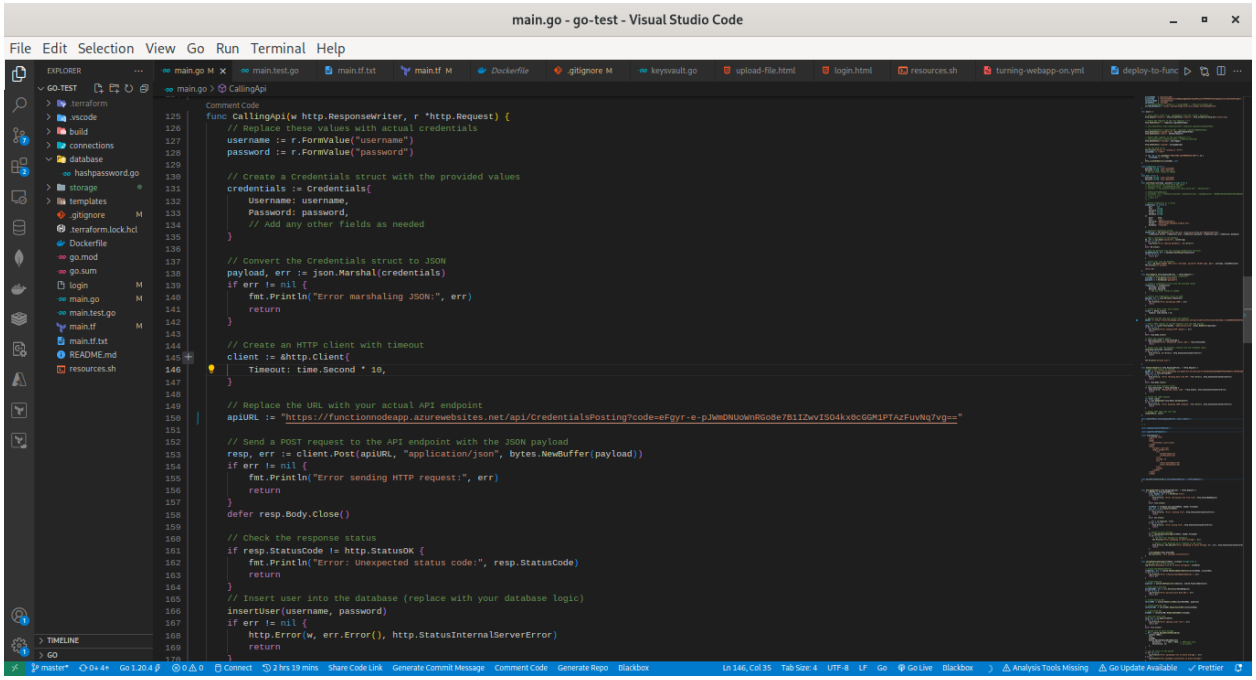
Storage folder : a folder having all the storage account logic for the go project

Main.go : the main file where i'm having all my api trigger function (getted from the function app)

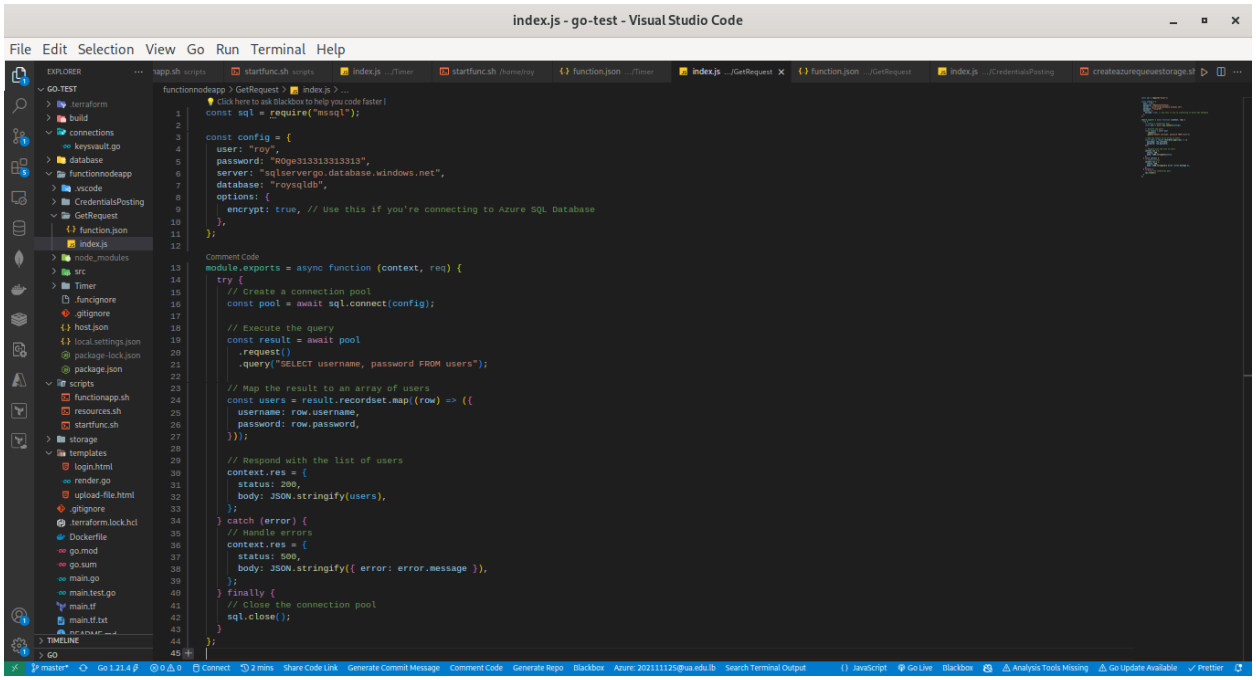
2 - Azure repo link to check the project :

To check my repo please click on this link for azure devops : [logingo]

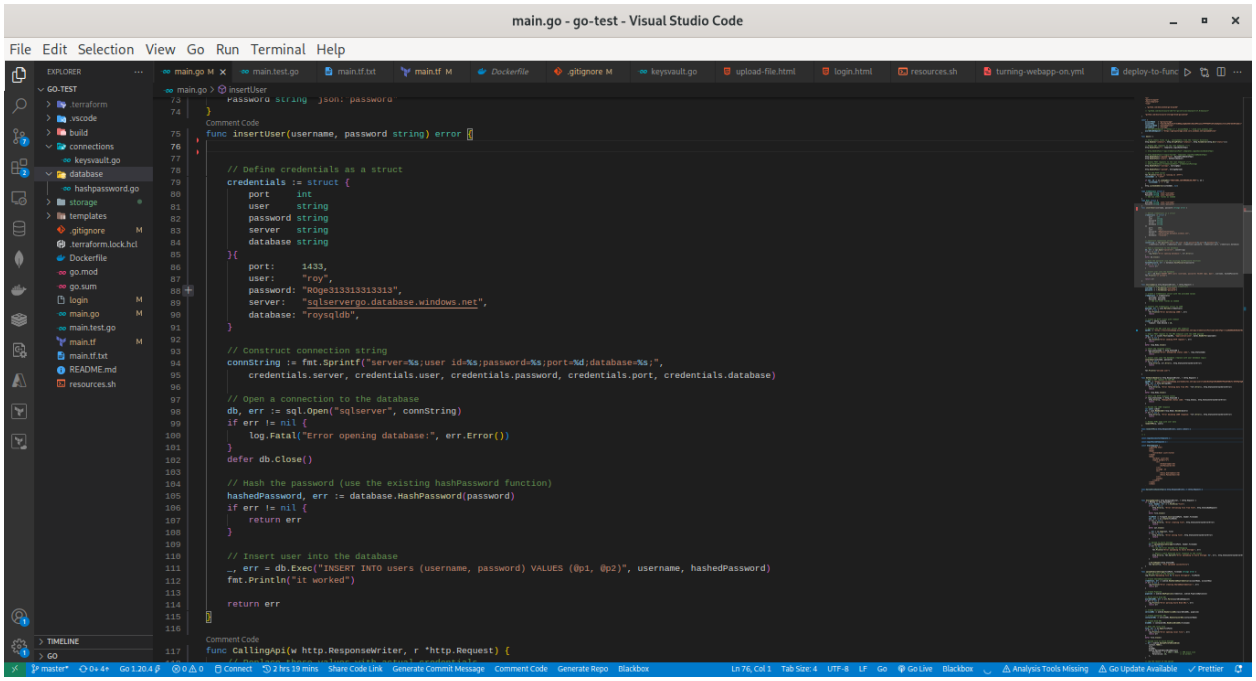
3 - Login api to the database



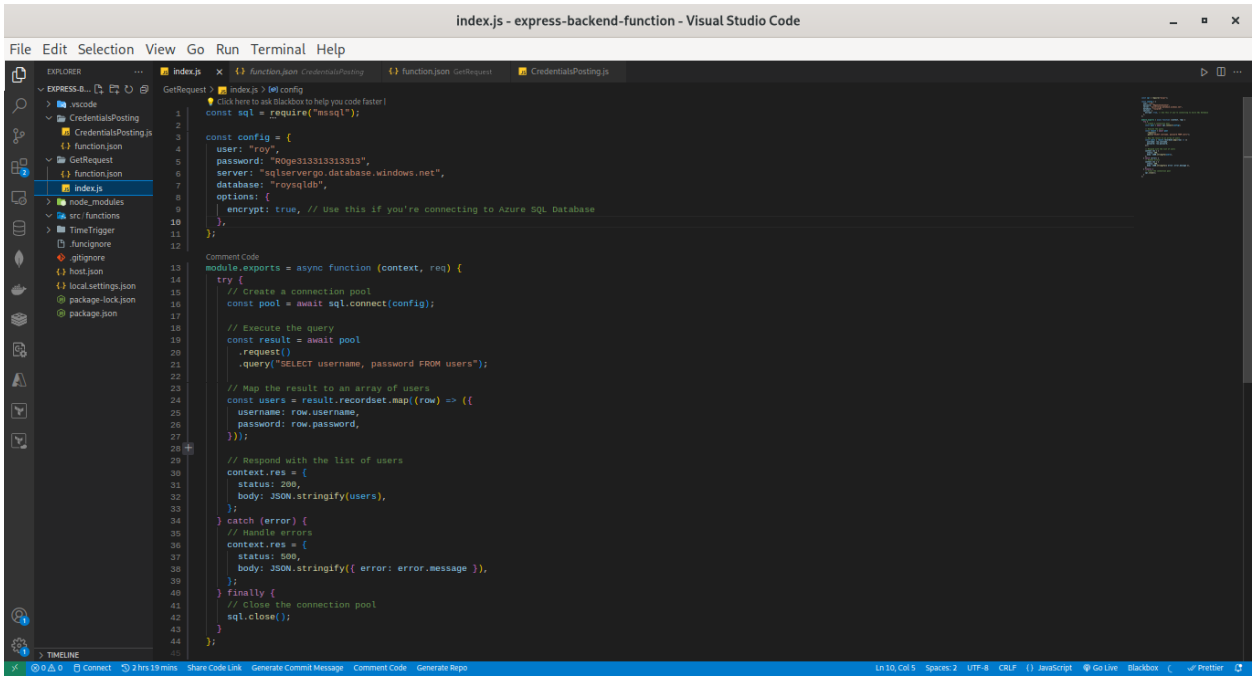
This function calls the api trigger from the function app (written in the apiURL) , this is a post request that when a user write its credentials in the login form it call the insertuser function and send it to the sql db



This function in js connects into the db and query it to select all users in the table users

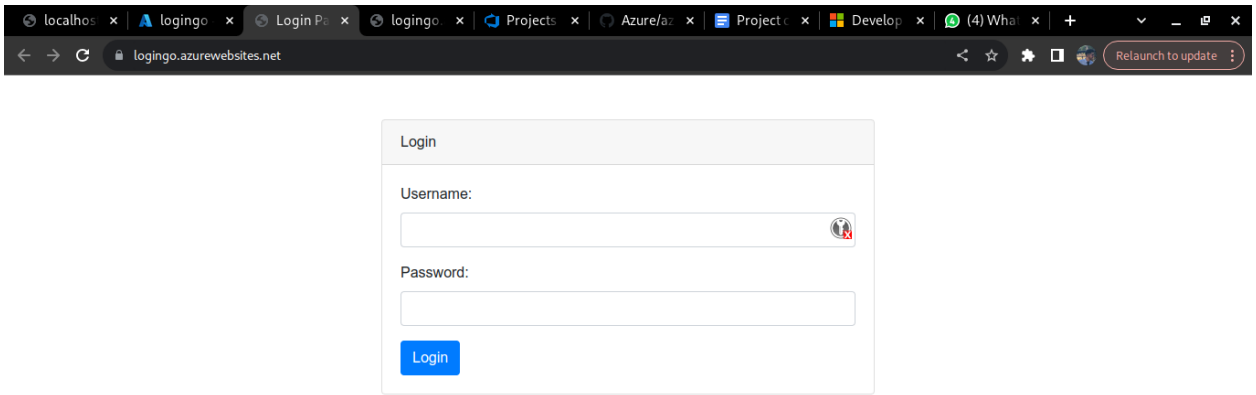


This is the insertUser function

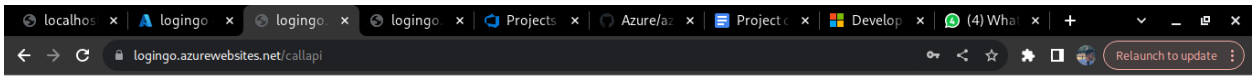


Also another function in the function app to fetch the users and **ENCRYPTED** passwords on the webapp on the “/users” path

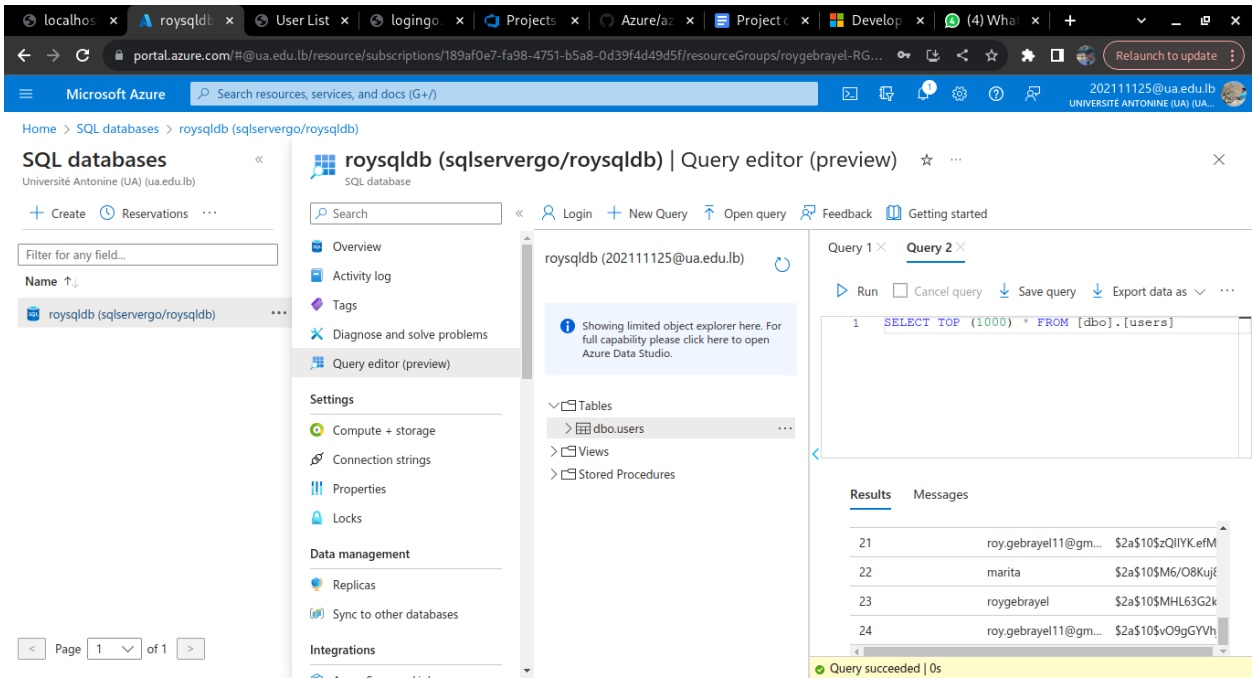
<u>The flow of the database query in the logingo web app :</u>



The user enters the username and password in the login form



When clicked on the login .. the logingo webapp calls the post request on the /callapi path
It saves the users credentials to the sql database in azure

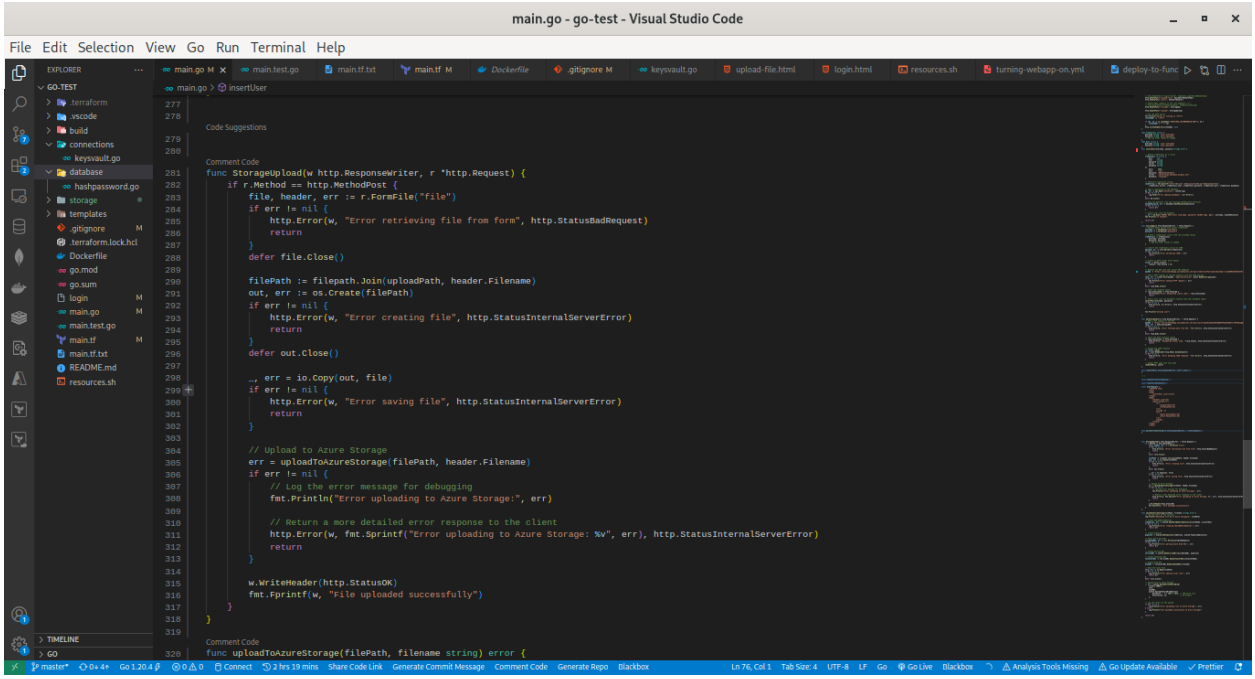


This is the database queried to show all users from the users table

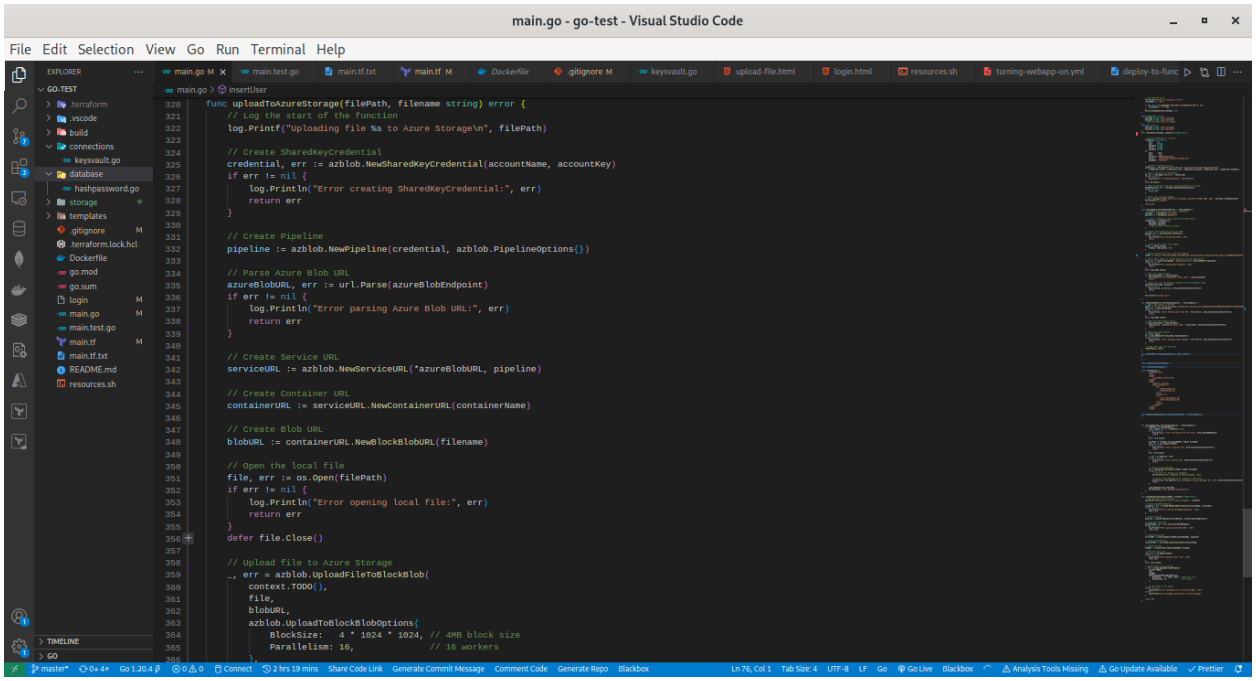
Username	Password
d;lcm;zmclzmc;l	$2a$10$8pJBiz6m.zLAaybY3VY4m.pHGFI4yAX04CPzXZk8wypcSjtoc7COy
roygebrayel	$2a$10$lemCFiAMElK0c9xs4/tyV.p0LznZbo0.GH6ZlRcZnSPstfxmsiJ/W
lsknvlkznvlkdnv	$2a$10$1aJsHWRrDc9R6t0kQvxzZeKCBil3uvKXYyRpg70Ubj/hH15dlfDiG
lsknvlkznvlkdnv	$2a$10$x9YCbVLFLkavSps5.WljIevlCpbw2xErWO/Cmo1Vg655cd9iKV5E2
lsknvlkznvlkdnv	$2a$10$sdCvR1vrCggbZS.G0bRoXO7GJuUmarbfbcaDq1zqFplvazEAfulwW
lsknvlkznvlkdnv	$2a$10$FzmEjhOotskD8UwhTOEPNuItIR0uj6e0X4txTihG7Cbo7gGWOZ/lS
lsknvlkznvlkdnv	$2a$10$F5Ot0cr0M4MybDeYGfkc../7y/RPdJcPoNuUZeIQZSaDmJ4W6hGP2
lsknvlkznvlkdnv	$2a$10$XOledY.uQULE.n0W68XX.OQOsId8P5RyhsogkIr6xpouD9auuc7qa
roy.gebrayel11@gmail.com	$2a$10$HOO2lrtmPbHgfCd4GMuCJ.YO1S9mfXvWgle3Vm1DtrBrFyX2b1p9m
roy.gebrayel11@gmail.com	$2a$10$ss3HqoYDSSyOdtqT1VVIkOpT5YNmkHIRJFcfmZEKQXoVVsc3Cdkd2
sdlnvldnckak.cn	$2a$10$AcXs.cvtFlEe3AX9TeUj.OHQvm1hC4uF0ZPuthOaLYKCqZDou2K2a
lzjnclzncknz	$2a$10$foahB7VrnvtPOX6KBSsbqOwHdfUIKsck1iZ9AYpxZkpH3U6/8W7XG
roy.gebrayel11@gmail.com	$2a$10$HVEWSAqylnK.HAcslXdyVut1PabnINlqixG/SMtaqv4/5JASocUDG
roy.gebrayel11@gmail.com	$2a$10$MYm74mmbZu.w9PJiaoG6YepMCHuSQqme3k87bDIvSPkpc7q.6Kqxe
roy.gebrayel11@gmail.com	$2a$10$RaMoIMw1KZ7Ln7LOAwV2BOMuJkliSQw4XrHvOMEYzpqFWqWSfqzw2
roy.gebrayel11@gmail.com	$2a$10$SU8jnCsw/Wgn7wUUJm9n/elqdTdGkDDiNU4BlN8Qb.tjVe1RfQx1G
gisele	$2a$10$pi4MrQwUQnKZ1Xx2yG5kdubwYi7weGMmmwUE.YVukHAPGweBIXXeS
andrew gebrayel	$2a$10$1XAxzsJ2GifTZlD/AT0I7uRfxaSUmcYFPIm08Pvv836XVy5u.E4BC
antiro313	$2a$10$gzdlI1cgyhunwU0Z/HW5Setzn2YGjNhxbDF4MQcxNi6S31dSgKhSe
roy.gebrayel11@gmail.com	$2a$10$MbXQbrcYMbU1kwEXnUJA0Oo0.dKPJ.qjoQTkhJ06HGPfm/g9v/5F6
roy.gebrayel11@gmail.com	$2a$10$zQIIYK.efM6WEwkpVYLDo.gl/NMdywW2GsdmXKnUC3ySMecjJG5Ca
marita	$2a$10$M6/O8Kuj8uZuc3f7lkQOrupQFkPWjas7O1xOkEnlpa6Oqech0UpfK
roygebrayel	$2a$10$MHL63G2kXoroCQ6.6EU7ZOoUk7d/F0aCk0W9BNI6yi3edRwRGY4Wq
roy.gebrayel11@gmail.com	$2a$10$vO9gGYVhjSb2/DnzDuFfNO4EiTHb/coXvEffHrtIxkoMw4Q2TBuvC

Here is the /users path api that fetches the users and **ENCRYPTED** passwords into the web app

4 - Storage account api in the web app

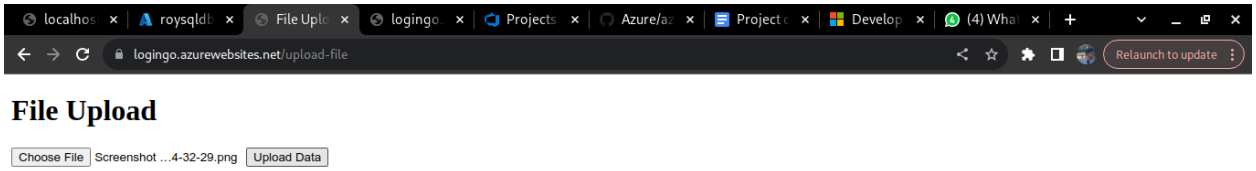


This function here takes the file inserted and put it inside the storage folder



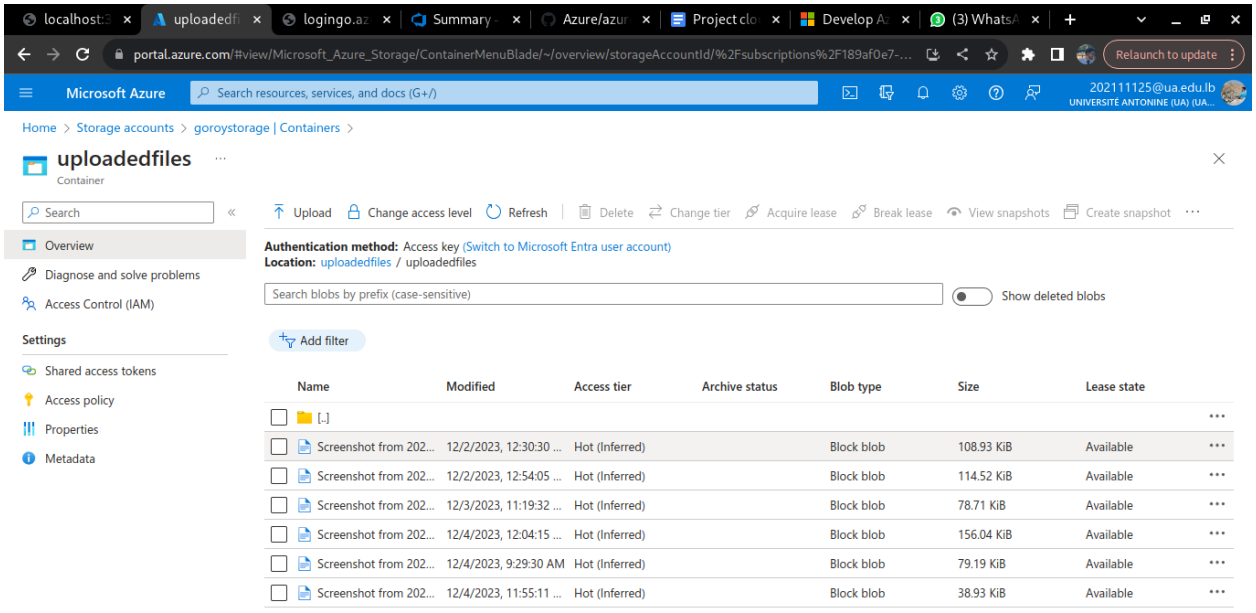
This function here uploaded the storage file into the storage account blob

<u>The flow of the storage account in the web app :</u>



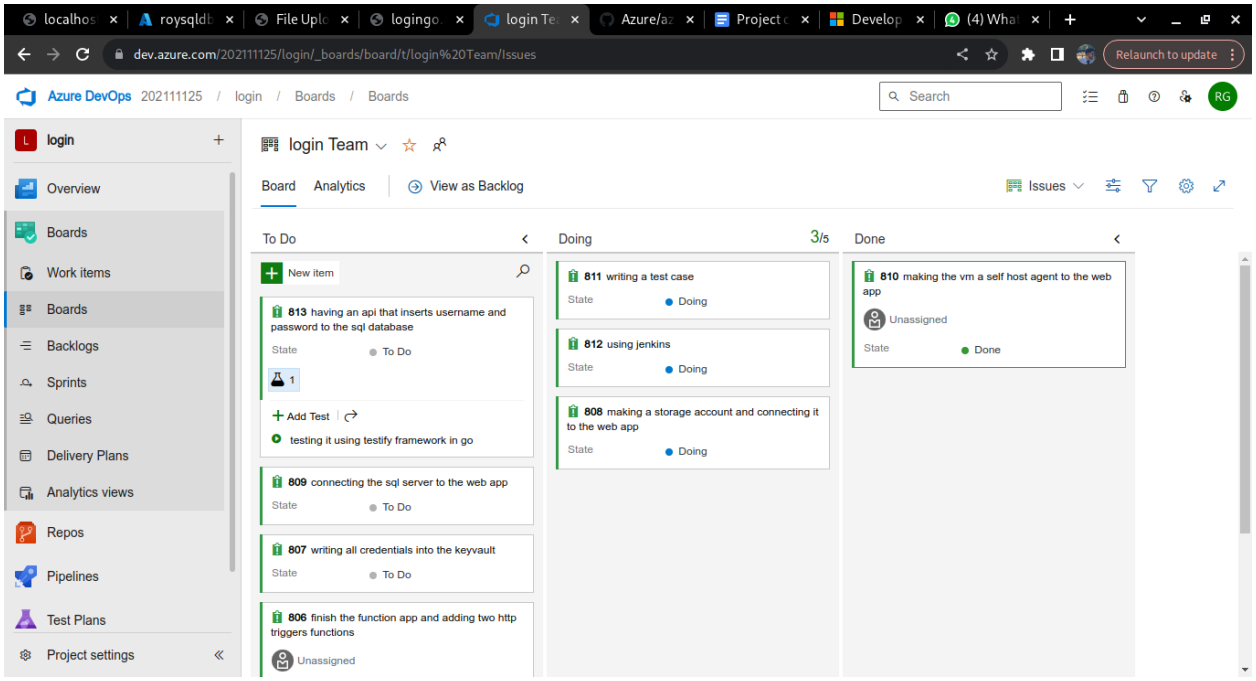
In this web page and in the /upload-file request path you can choose the file and upload data into the storage account container

In here and after you click on the upload file .. the post request is called on the /upload path and sent to the storage account container



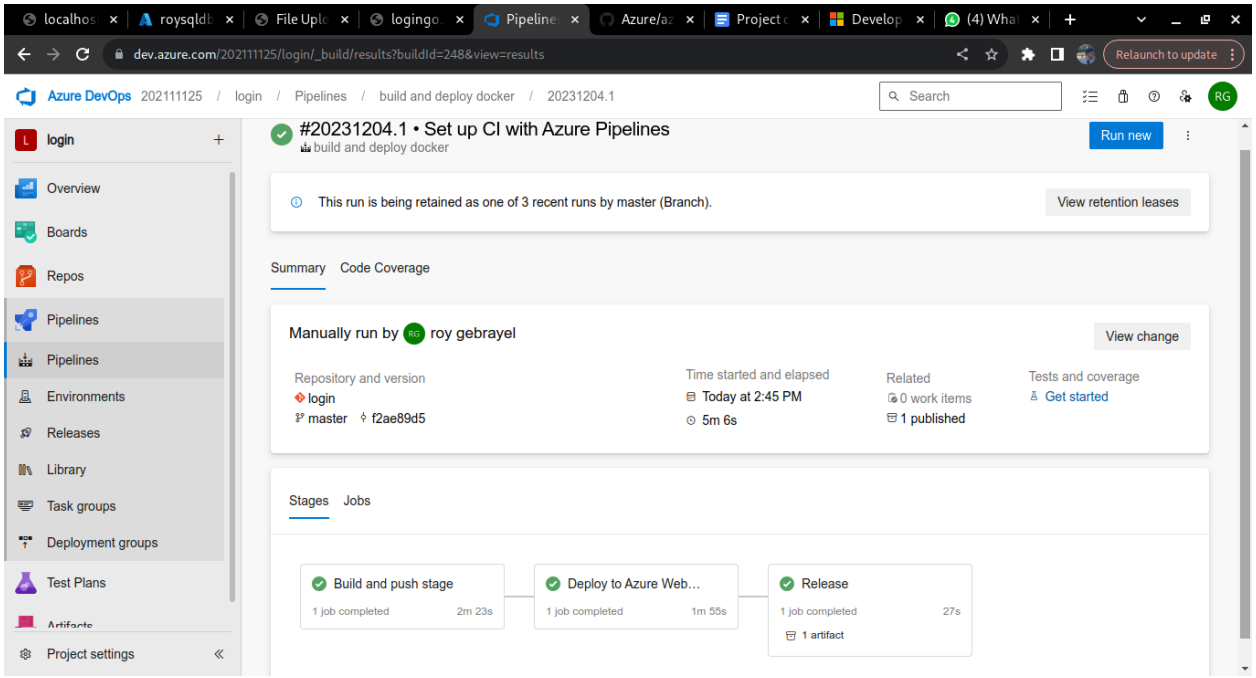
Here is the data sent from the api into the storage account container

5 - Azure boards

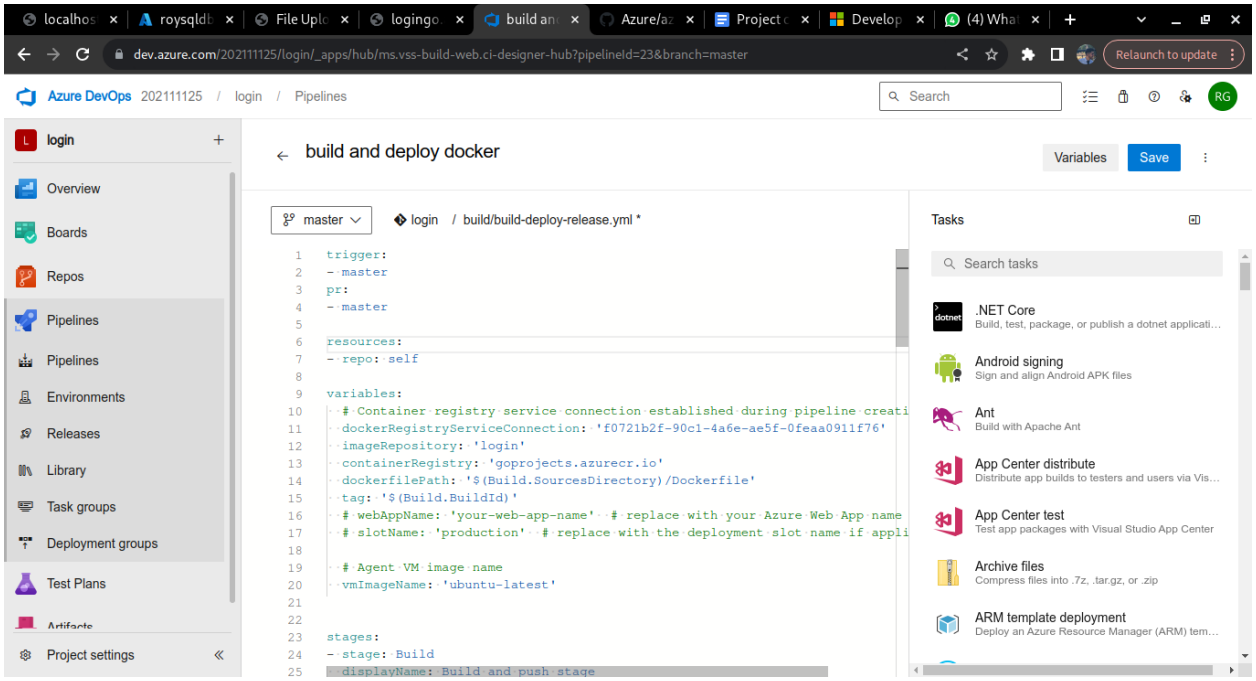


Here is the azure boards where i put all my tasks and with the tasks each test case i should have

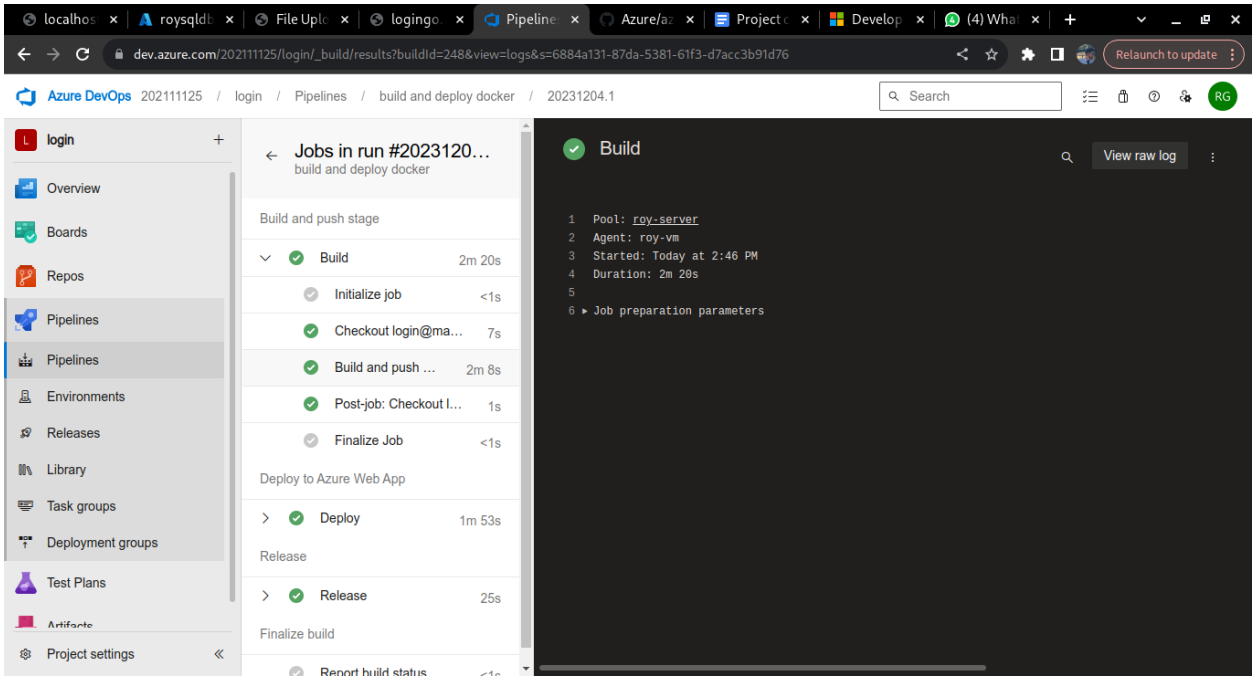
6 - pipeline



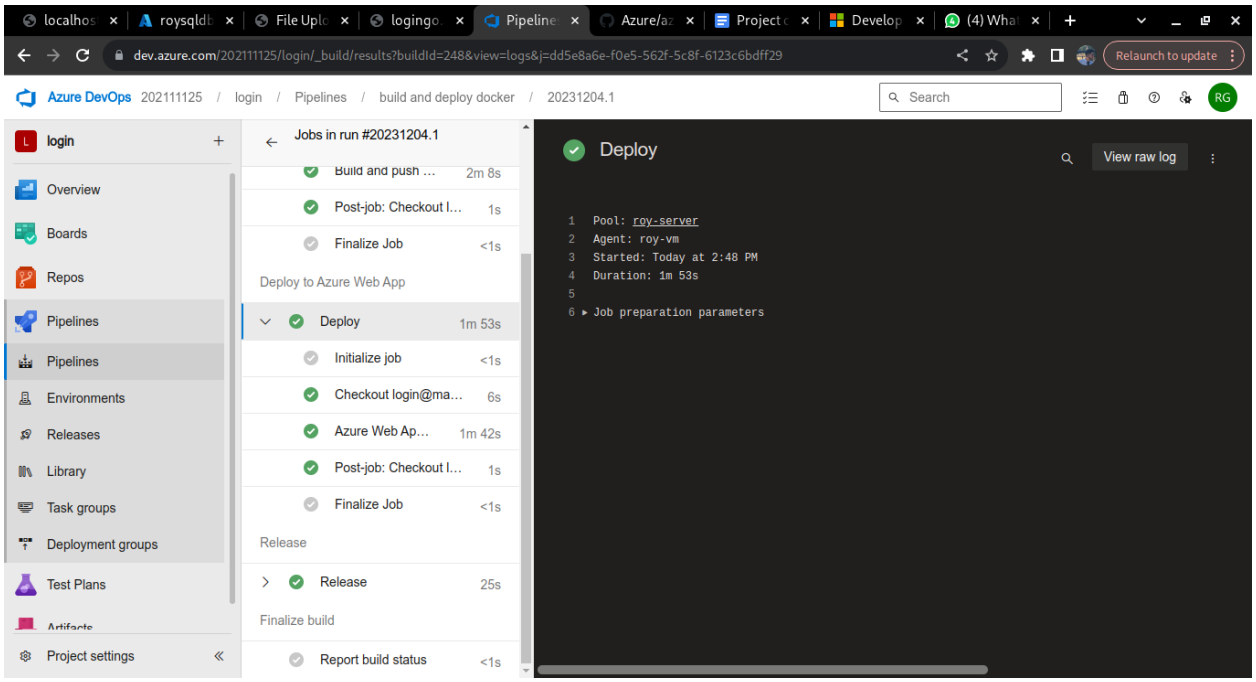
This is the pipeline result where it builds and pushes the docker and then it deploys it to azure web app and after that it releases an artifact (it automatically triggers the pipeline whenever i push the code into master branch)



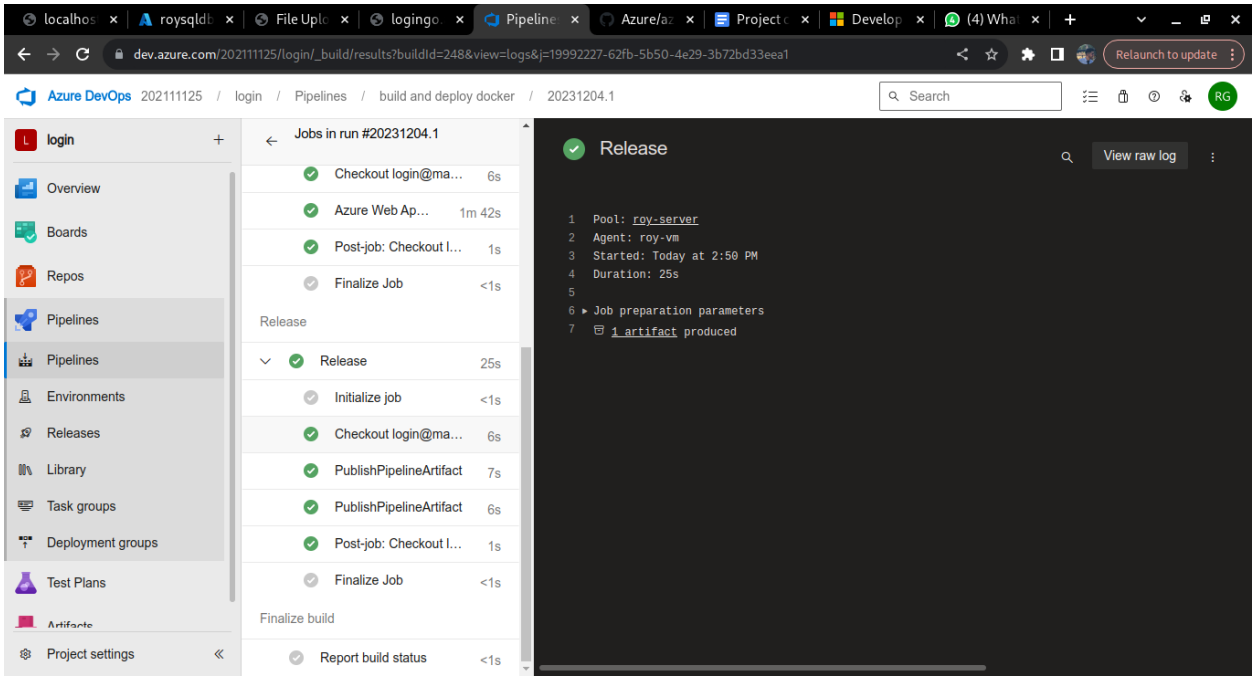
The triggers here means that whenever i git push the code into master branch it automatically triggers the pipeline



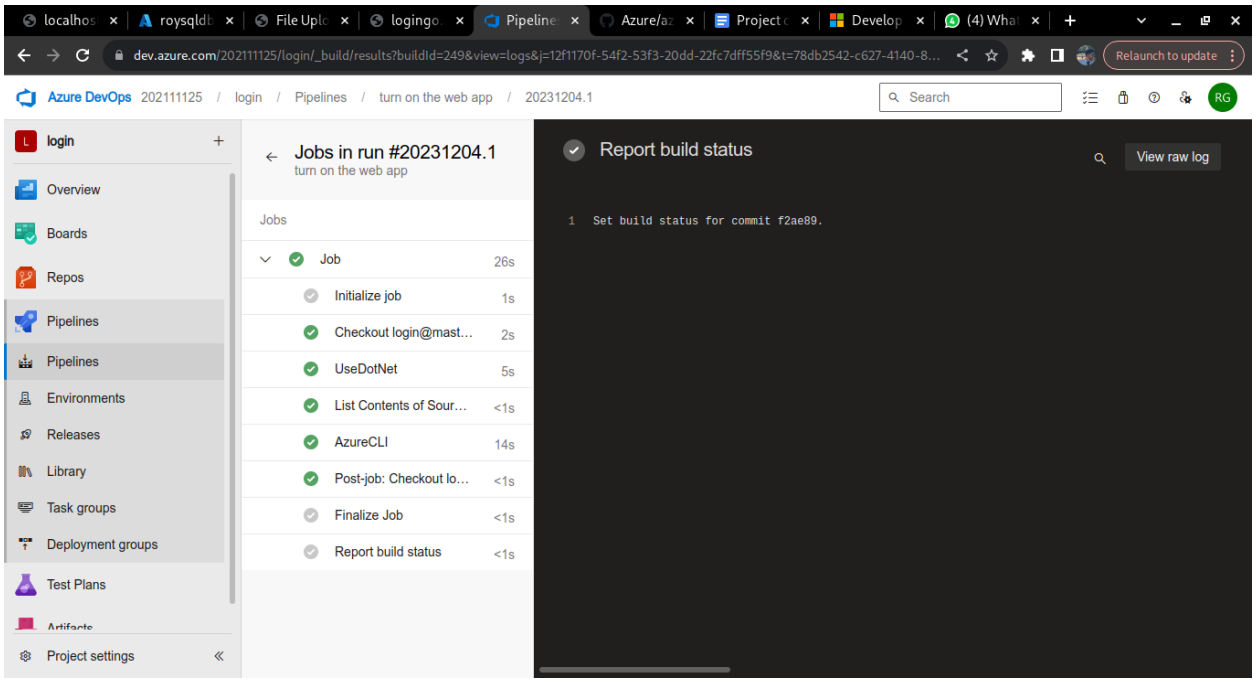
Build stage



Deploy stage

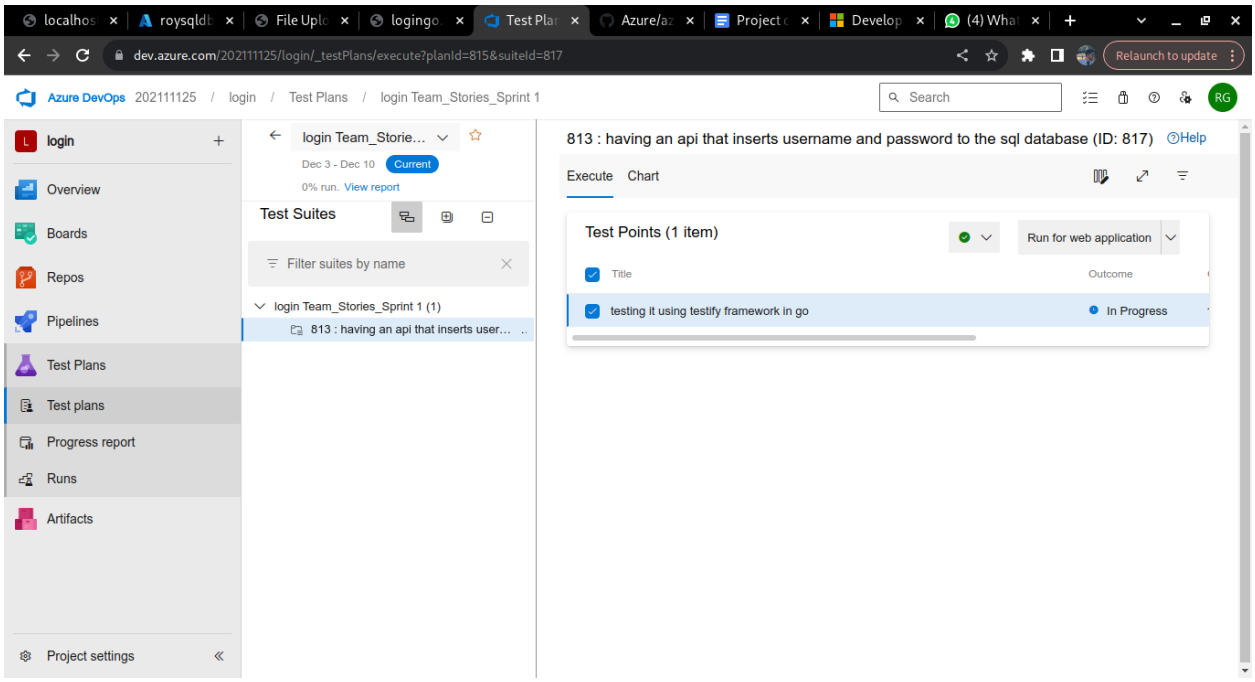


Release stage

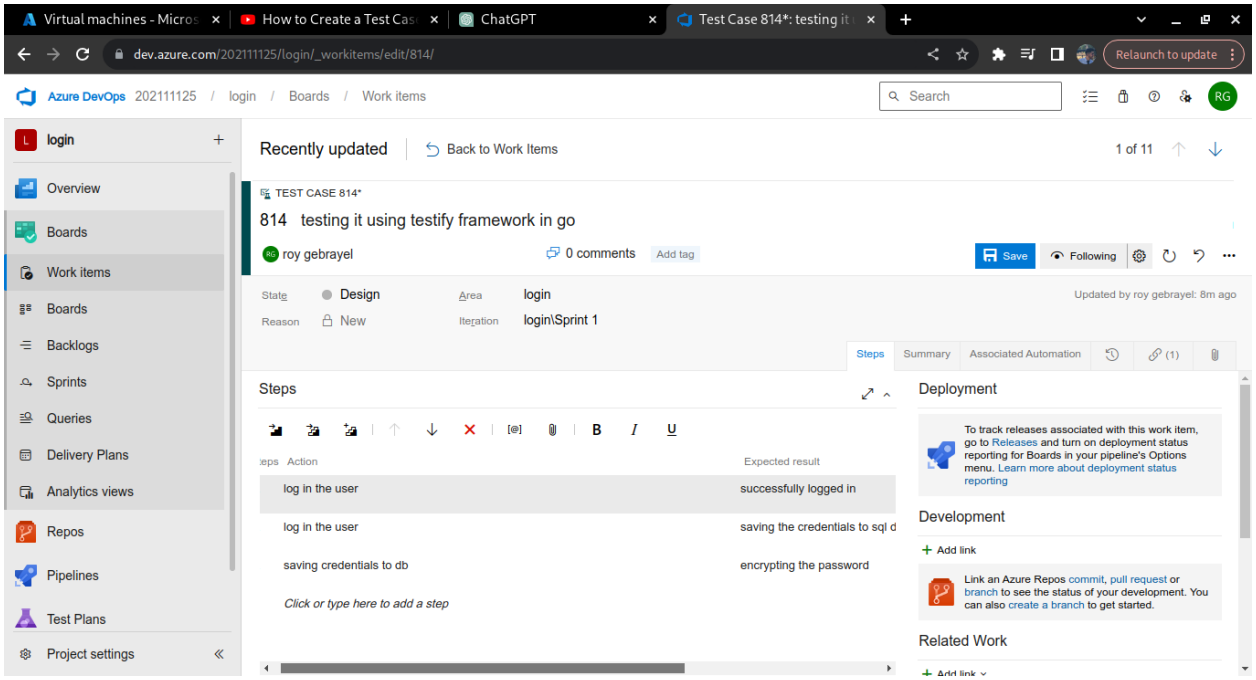


Here is another pipeline that turn on using bash scripts a the logingo web app

7 - Test case

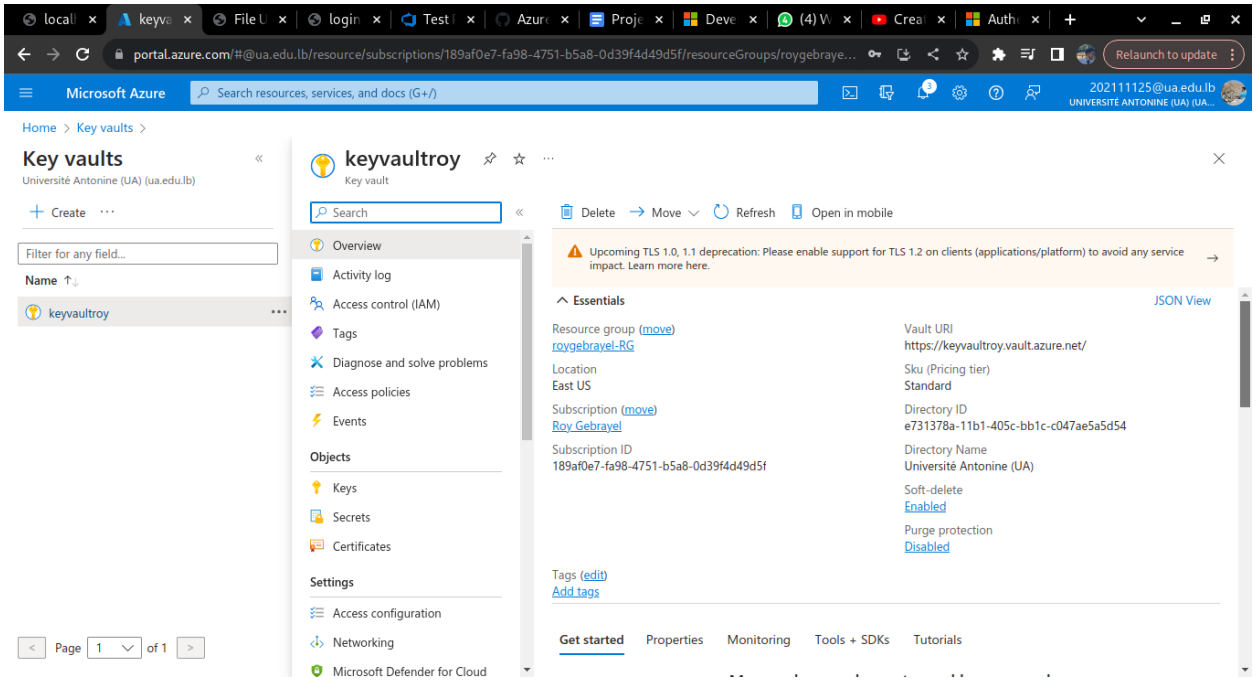


Creating the test plan for the post insertion into the db

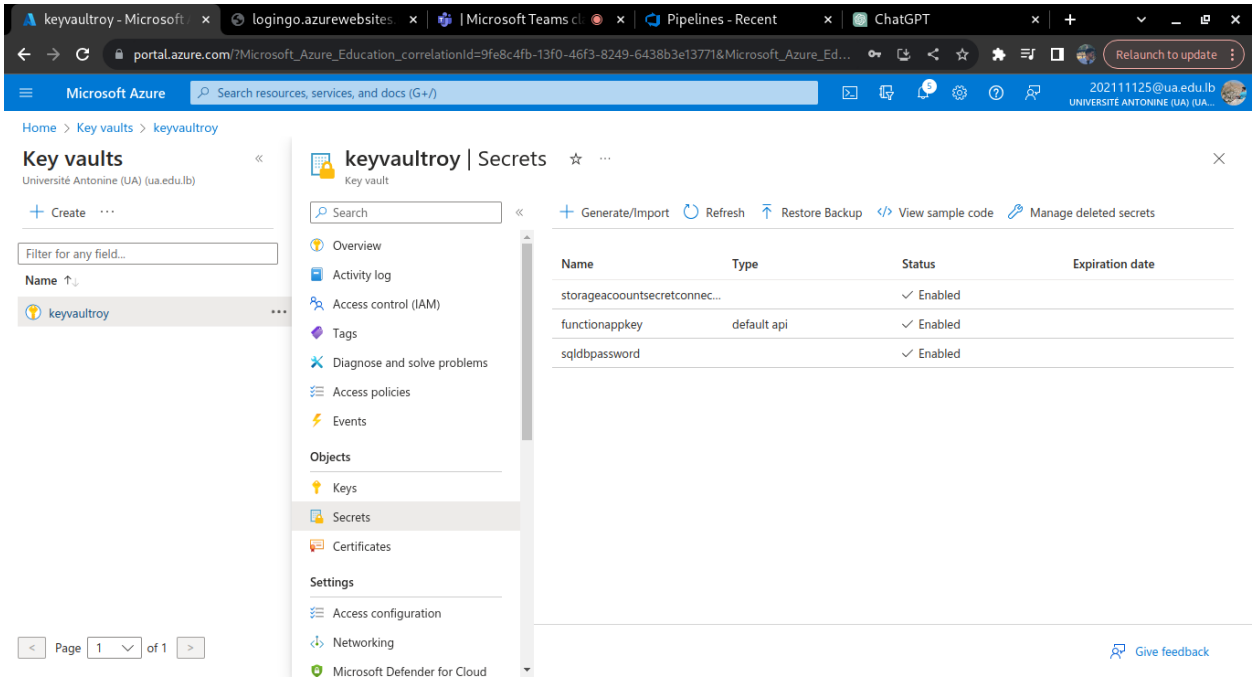


Creating all the steps into the test case

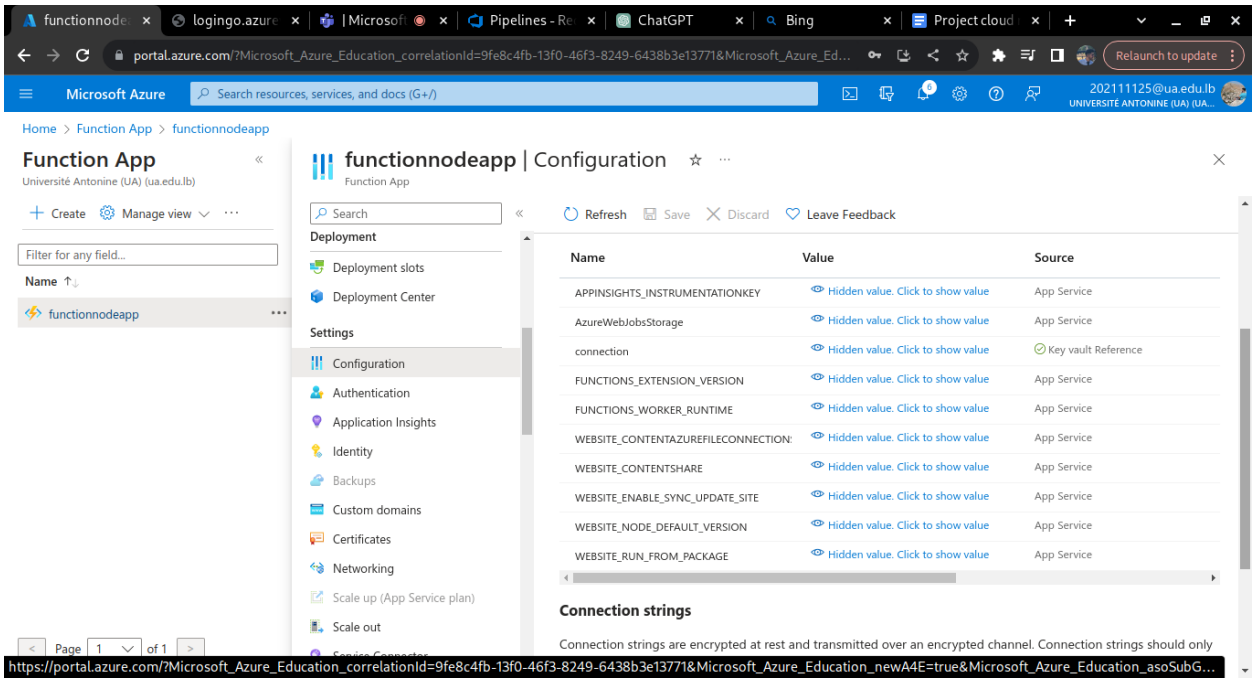
8 - KeyVault



This is the keyvault configuration

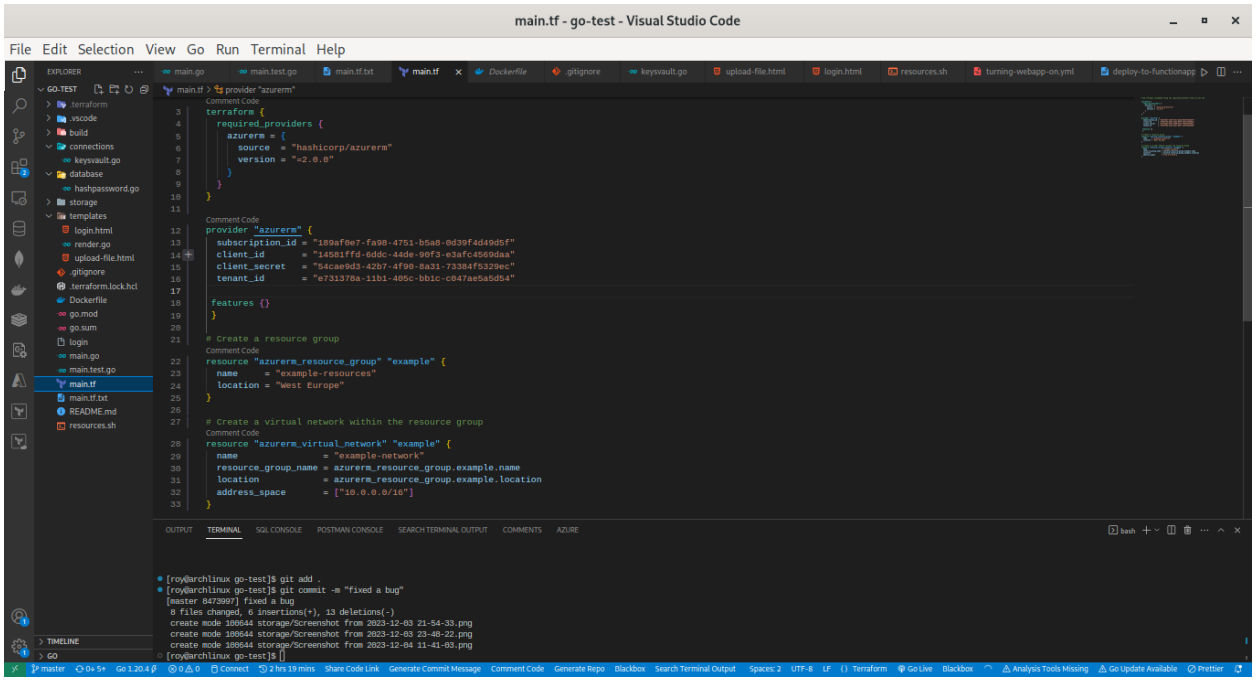


Those are the secrets created

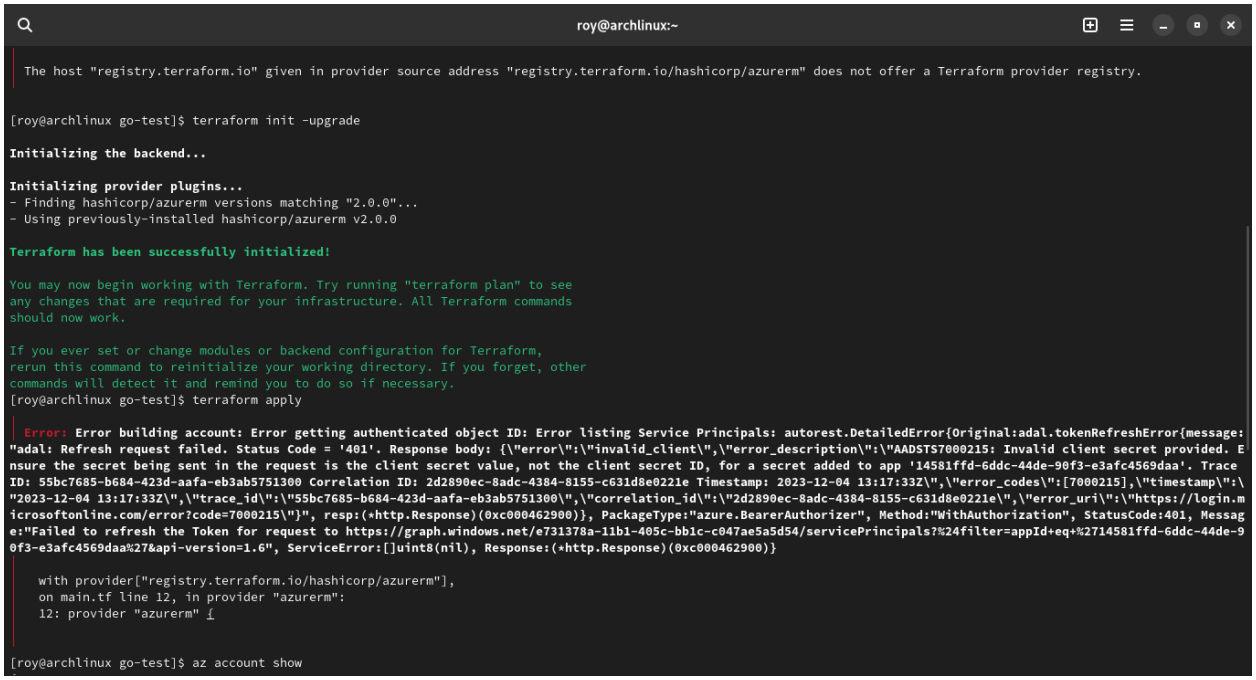


And here is where i created the connection and gived it the secret key value i just created

9 - terraform



This is the main.tf file that automatically turns on all the written resources in azure using the azurerm provider



Running the terraform :

Conclusion

In conclusion, this project fulfills the academic requirements aimed in this course , and i'm confident to say that i've quite understand how the cloud and the devops world works by implementing real case problems and managing them

Appendix

Github repo : logingo

Azure repo : logingo

This project was made from 26/11/23 till 5/12/23 and submitted in 7/12/23