

Final Deliverables Report

Date	14.11.2022
Team ID	PNT2022TMID28817
Project Name	Inventory Management System for Retailers

Team members and their Contribution:

Name	Roll no	Contribution
Bojja Naveen Kumar	410719106073	Frontend – 5 Pages, Integration of Sendgrid, Deployment of using docker and Kubernetes.
Dasari Chethana	410719106022	Frontend – 5 Pages, Documentation
Shaik Shehanaz	410719106091	Frontend – 4 Pages, Documentation
Nagappagari Devasankarsai	410719106069	Backend Fully (For all 14 Pages), Integration of IBM Cloud, Deployment of using docker and Kubernetes.

Introduction:

1. Sprint 1 – Backend
2. Sprint 2 – Frontend
3. Sprint 3 – IBM Cloud Integration + Integration of SendGrid
4. Sprint 4 – Deploying the application using Docker and Kubernetes

Sprint 1 – Backend:

All the routes to each page and APIs are created.

Example, (For Products page)

The screenshot shows a Visual Studio Code editor with a Flask application. The Explorer sidebar on the left lists the project files: `IMSR-1`, `__pycache__`, `templates`, `app.py`, `email_alert.py`, `requirements.txt`, and `DigiCertGlobalRootCA.crt`. The main editor window displays the code for `app.py`, which includes a route for `/products` that queries a database and renders a template. The bottom status bar shows the current file is `app.py`, line 33, column 29, and the Python version is 3.10.7 64-bit. The bottom panel shows the terminal output, which includes a warning about using a development server and a message about restarting with `stat`.

```

22
23 #Products
24 @app.route('/products')
25 def products():
26     sql = "SELECT * FROM products"
27     stmt = ibm_db.prepare(conn, sql)
28     result=ibm_db.execute(stmt)
29
30     products=[]
31     row = ibm_db.fetch_assoc(stmt)
32     while(row):
33         products.append(row)
34         row = ibm_db.fetch_assoc(stmt)
35     products=tuple(products)
36     #print(products)
37
38     if result==0:
39         return render_template('products.html', products = products)
40     else:
41         msg='No products found'
42         return render_template('products.html', msg=msg)
43
44 #Locations
45 @app.route('/locations')
46 def locations():
47

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** JUPYTER

```

* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 512-196-405

```

Ln 33, Col 29 Spaces: 4 UTF-8 CRLF Python 3.10.7 64-bit Go Live

Sprint 2 – Frontend:

The frontend is written using HTML, CSS (using Bootstrap) and JavaScript for all the pages to which the routes created in Sprint 1.

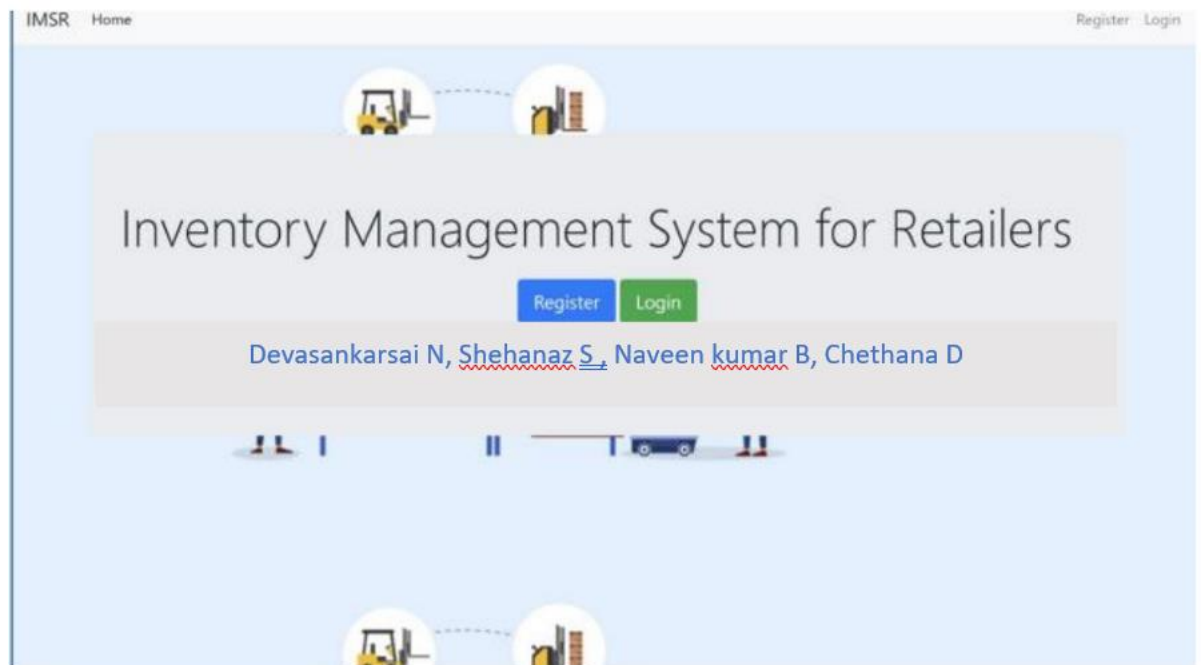
For Example, (The Hierarchy of different pages and the code for login page)

The screenshot displays the Visual Studio Code interface with the following components:

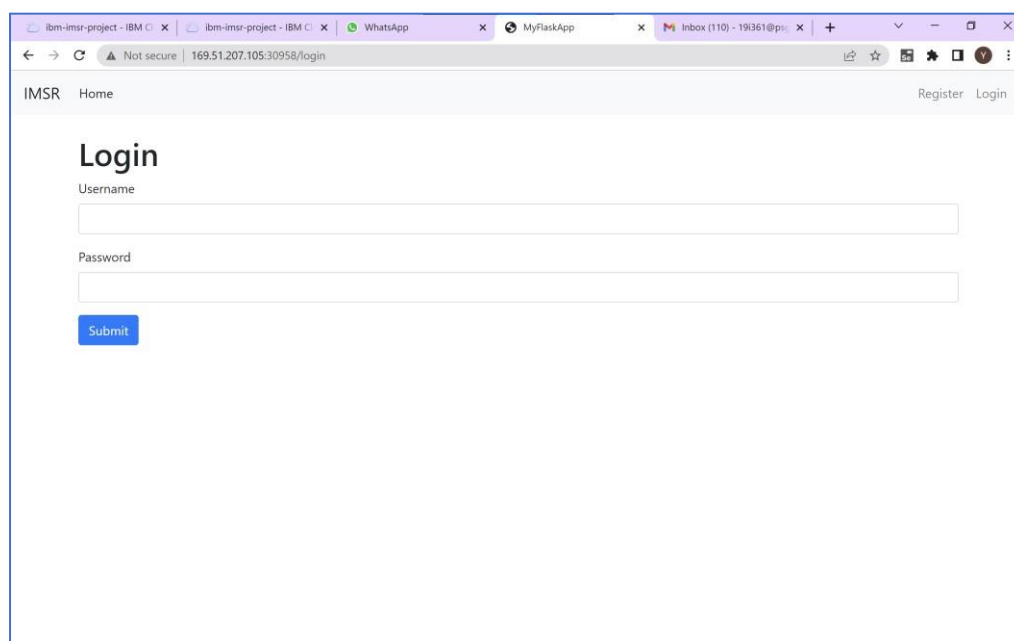
- Explorer View (Left):** Shows the project structure for 'IMSR-1'. The 'login.html' file is selected and highlighted in red. Other files visible include 'app.py', 'email_alert.py', 'requirements.txt', and various HTML templates.
- Editor View (Center):** Displays the content of 'login.html'. The code is an HTML form for user login, featuring fields for 'Username' and 'Password', and a 'Submit' button. The form uses Bootstrap classes for styling.
- Terminal View (Bottom):** Shows the output of running the Flask application. The output includes a warning about development vs. production mode, the server running on http://127.0.0.1:5000, and instructions to press CTRL+C to quit. The terminal also shows the server restarting with stat, the debugger is active, and the debugger PIN: 512-196-405.

Sample FrontEnd Pages,

Home Page,



Login Page,



Register Page,

IBM ibm-imrs-project - IBM C ibm-imrs-project - IBM C WhatsApp MyFlaskApp Inbox (110) - 19i361@ps

← → ↻ ⚠ Not secure 169.51.207.105:30958/register

Register Login

IMSR Home

Register

Name

Email

Username

Password

Confirm Password

Submit

Products Page,

IBM ibm-imrs-project - IBM C ibm-imrs-project - IBM C WhatsApp MyFlaskApp Inbox (110) - 19i361@ps

← → ↻ ⚠ Not secure 169.51.207.105:30958/products

Logout Dashboard

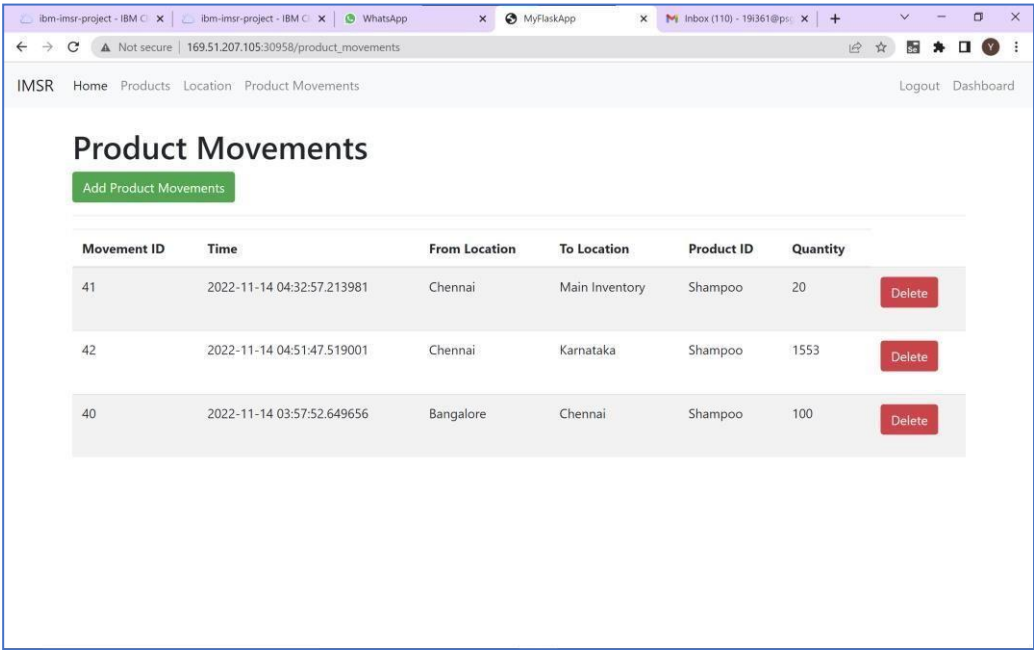
IMSR Home Products Location Product Movements

Products

Add Product

Product ID	Product Cost	Product Quantity		
Bedspreads	600	100	Edit	Delete
Cutlery	1500	495	Edit	Delete
Shampoo	50	520	Edit	Delete

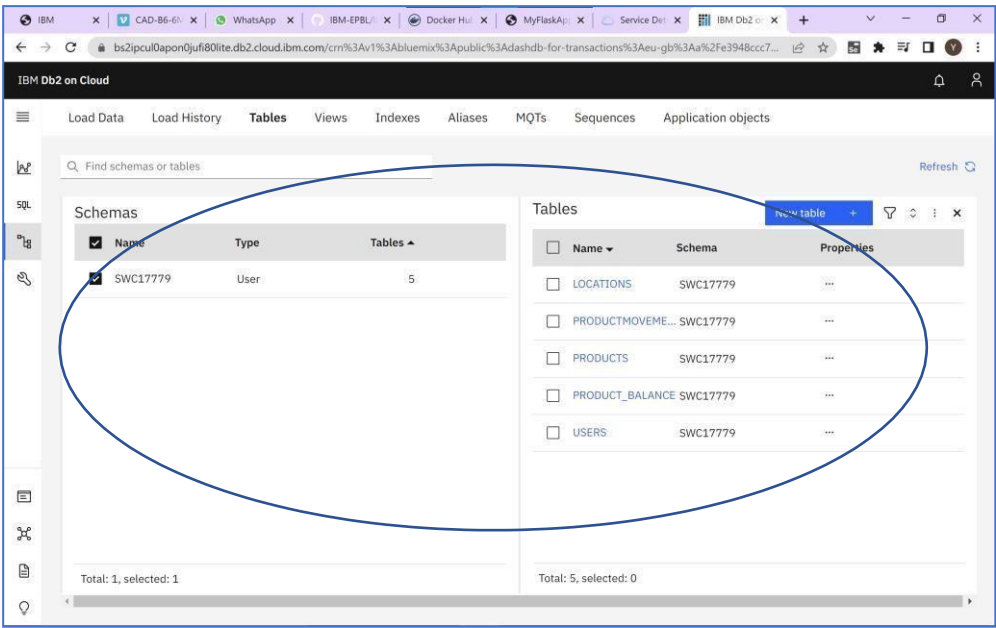
Product Movements Page,



Movement ID	Time	From Location	To Location	Product ID	Quantity	
41	2022-11-14 04:32:57.213981	Chennai	Main Inventory	Shampoo	20	Delete
42	2022-11-14 04:51:47.519001	Chennai	Karnataka	Shampoo	1553	Delete
40	2022-11-14 03:57:52.649656	Bangalore	Chennai	Shampoo	100	Delete

Sprint 3 - IBM Cloud Integration + Integration of SendGrid:
IBM Cloud Integration:

5 tables created for our project,



Name	Type	Tables
SWC17779	User	5

Name	Schema	Properties
LOCATIONS	SWC17779	...
PRODUCTMOVEME...	SWC17779	...
PRODUCTS	SWC17779	...
PRODUCT_BALANCE	SWC17779	...
USERS	SWC17779	...

Schema of the particular table (For Example, Product_Balance)

The screenshot shows the IBM Db2 on Cloud web interface. The 'Tables' tab is selected, and a list of tables is displayed. The 'PRODUCT_BALANCE' table is selected, and its schema is shown in the 'Table definition' panel on the right.

Name	Data type	Nullable	Length	Scale
ID	INTEGER	N		0
PRODUCT_ID	VARCHAR	Y	255	0
LOCATION_ID	VARCHAR	Y	255	0
QTY	INTEGER	Y		0

Data of a particular table (For Example, Product_Balance)

The screenshot shows the IBM Db2 on Cloud web interface with the 'PRODUCT_BALANCE' table selected. The data is displayed in a table with 8 rows and 4 columns: ID, PRODUCT_ID, LOCATION_ID, and QTY.

ID	PRODUCT_ID	LOCATION_ID	QTY
1	Shampoo	Kerala	1350
2	Bedspreads	Kerala	100
3	Shampoo	Chennai	1452
4	Shampoo	Mumbai	100
5	Shampoo	Karnataka	-202
6	Shampoo	Punjab	100
7	Shampoo	Bangalore	-451
8	Cutlery	Bangalore	55

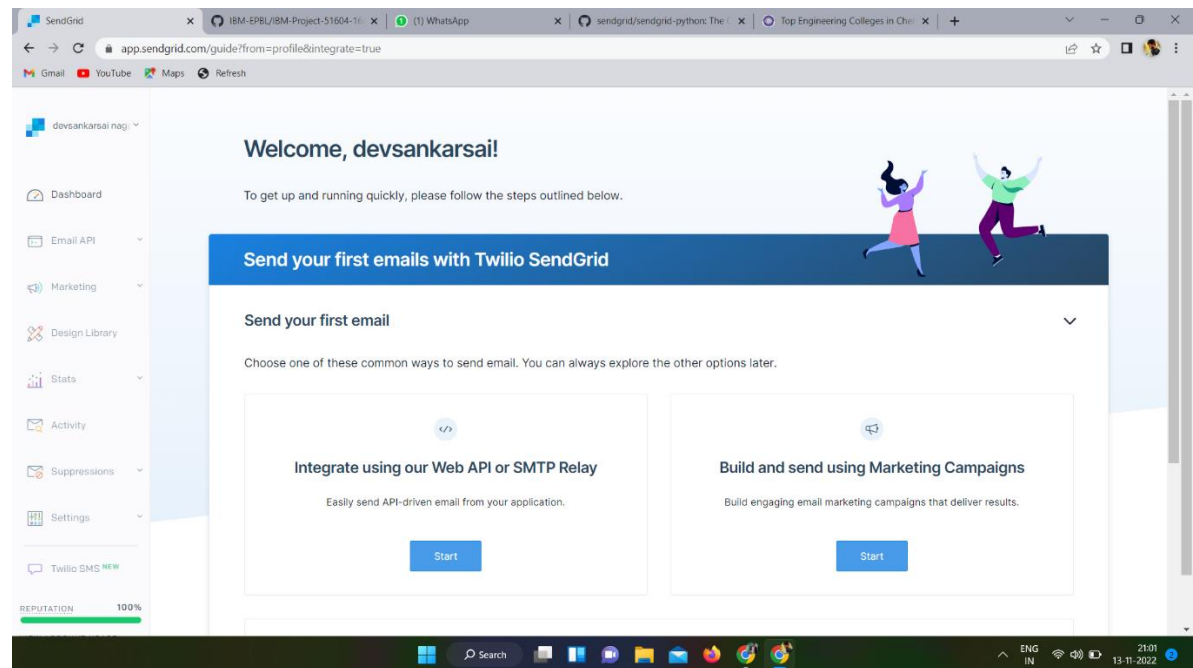
Code for Connection of IBM Database,

```
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=55fbc997-9266-4331-afd3-888b05e734c0.bs2io90l08kqb1od8lcg.databases.appdomain.cloud;PORT=;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=;PWD=;","")
```

Note: DigiCertGlobalRootCA.crt should be downloaded and configured within the project folder.

SendGrid Integration:

Creation of SendGrid account,



Code for email alert:

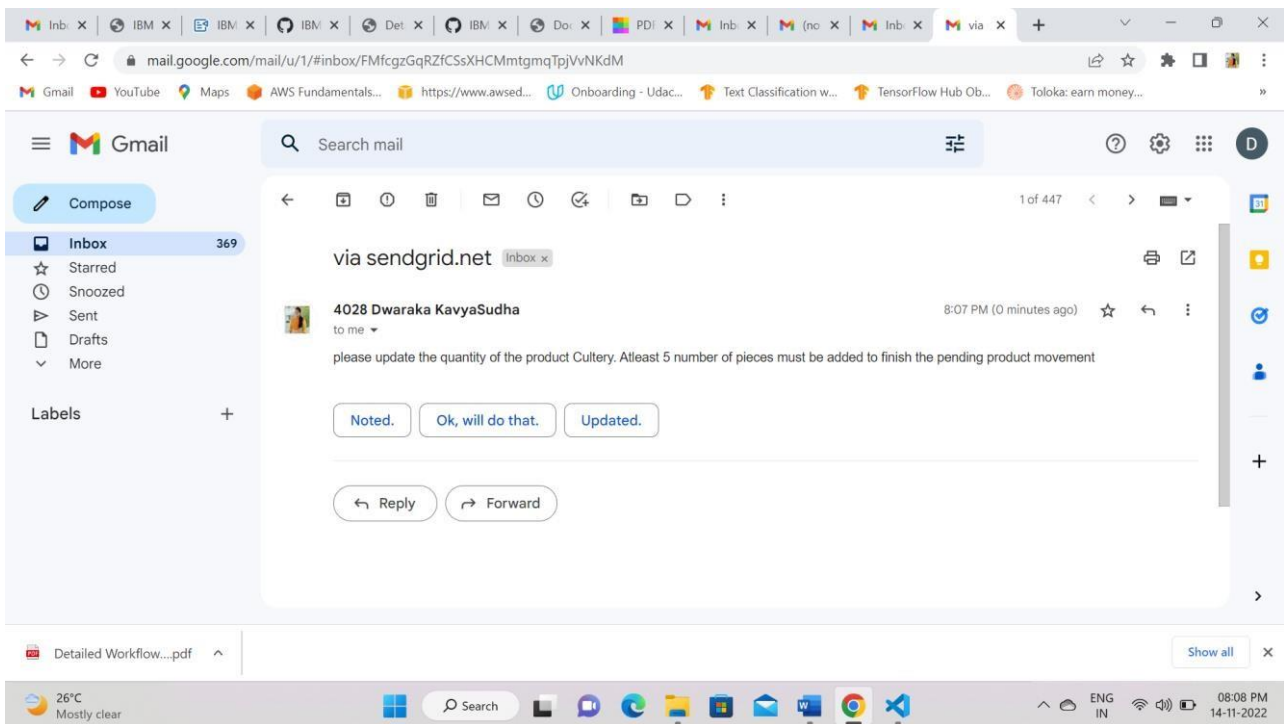
```
File Edit Selection View Go Run Terminal Help
email_alert.py - IMSR-1 - Visual Studio Code

EXPLORER
IMSR-1
  > __pycache__
  > templates
  > app.py
  > DigiCertGlobalRootCA.crt
  > email_alert.py
  > requirements.txt

email_alert.py
1 import smtplib
2 from email.mime.multipart import MIMEMultipart
3 from email.mime.text import MIMEText
4 from email.mime.base import MIMEBase
5
6 def alert(main_msg):
7     mail_from = '191361@psgtech.ac.in'
8     mail_to = '191303@psgtech.ac.in'
9     msg = MIMEMultipart()
10    msg['From'] = mail_from
11    msg['To'] = mail_to
12    msg['Subject'] = 'Alert Mail On Product Shortage! - Regards'
13    mail_body = main_msg
14    msg.attach(MIMEText(mail_body))
15
16    try:
17        server = smtplib.SMTP_SSL('smtp.sendgrid.net', 465)
18        server.ehlo()
19        server.login('apikey', 'API_KEY')
20        server.sendmail(mail_from, mail_to, msg.as_string())
21        server.close()
22        print("mail sent")
23    except:
24        print("issue")
25

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python + - + + +
* Detected change in 'C:\Users\yaswa\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1\email_alert.py', reloading
* Restarting with stat
* Debugger is active!
* Debugger PIN: 512-196-405
* Detected change in 'C:\Users\yaswa\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1\email_alert.py', reloading
* Restarting with stat
```

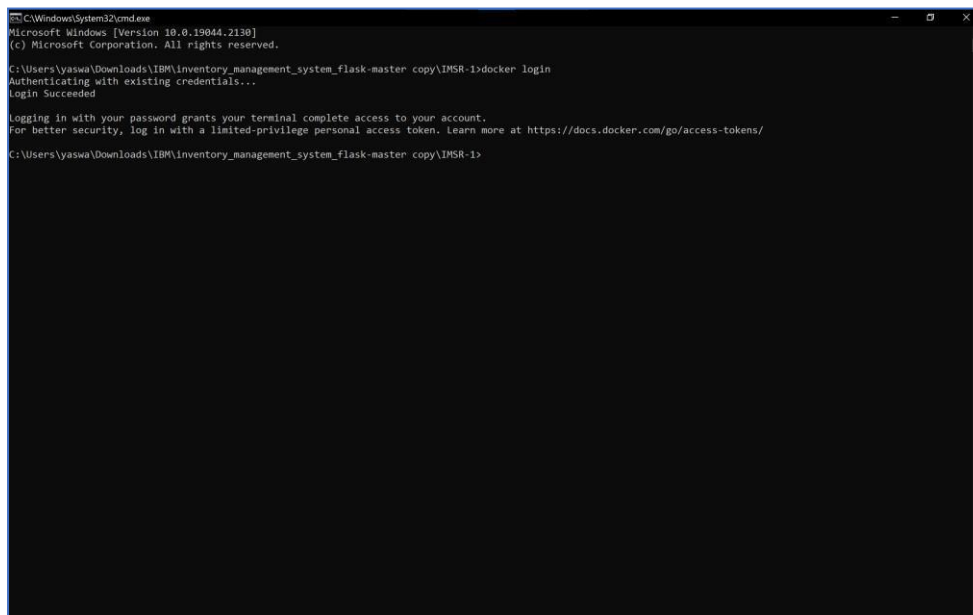
Email Received on Shortage of materials at particular warehouse or Main Inventory:



Sprint 4 (Deploying the application using Docker and Kubernetes):

Note: Make sure to create a Dockerfile in the project folder.

Login into DockerHub in Project Folder using command prompt. This connects local docker desktop to cloud docker hub.



Building an image for our project,

```
File "/usr/local/lib/python3.11/site-packages/flask/app.py", line 1820, in full_dispatch_request
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker build -t yaswanthmanoharan/ibm_imsr .
[+] Building 2.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 32B                                                0.0s
=> [internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                    0.0s
=> [internal] load metadata for docker.io/library/python:latest                 2.4s
=> [auth] library/python:pull token for registry-1.docker.io                   0.0s
=> [internal] load build context                                                 0.0s
=> => transferring context: 24.29kB                                              0.0s
=> CACHED [2/5] WORKDIR /inventory                                              0.0s
=> CACHED [3/5] COPY requirements.txt requirements.txt                          0.0s
=> CACHED [4/5] RUN pip install -r requirements.txt                             0.0s
=> [5/5] COPY . .                                                                0.0s
=> exporting to image                                                            0.1s
=> => exporting layers                                                            0.0s
=> => writing image sha256:0afb0c793a704eaf85acc886443c57a0cbeca9473b841897ef4a9162f3c4bd06 0.0s
=> => naming to docker.io/yaswanthmanoharan/ibm_imsr                            0.0s

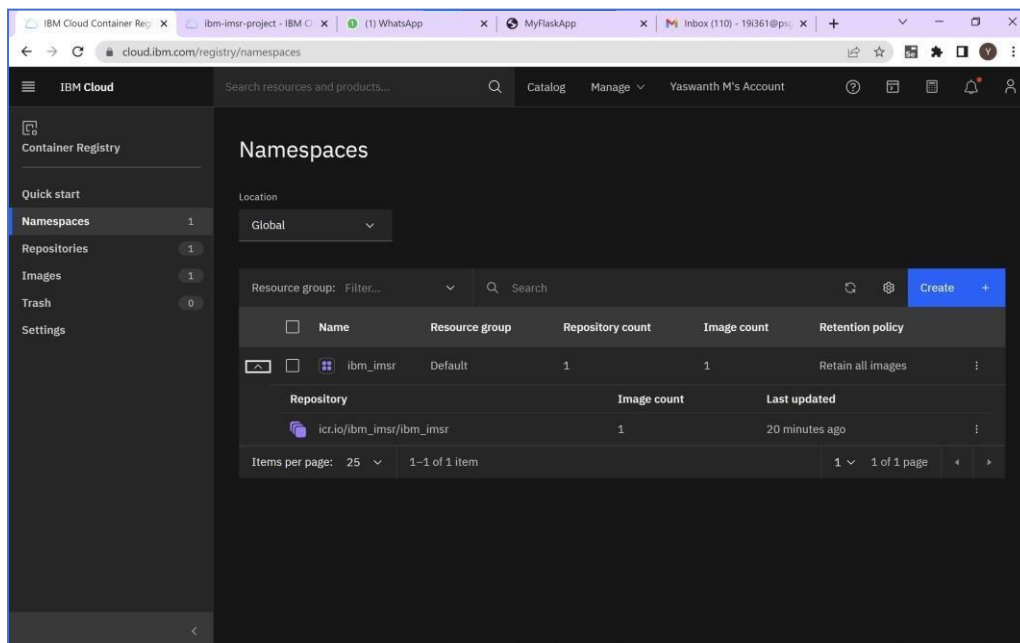
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker run -p 8080:5000 yaswanthmanoharan/ibm_imsr
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI serve
r instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
172.17.0.1 - - [14/Nov/2022 03:57:11] "GET /login HTTP/1.1" 200 -
172.17.0.1 - - [14/Nov/2022 03:57:22] "POST /login HTTP/1.1" 302 -
172.17.0.1 - - [14/Nov/2022 03:57:23] "GET /dashboard HTTP/1.1" 200 -
172.17.0.1 - - [14/Nov/2022 03:57:27] "GET /product_movements HTTP/1.1" 200 -
172.17.0.1 - - [14/Nov/2022 03:57:30] "GET /add_product_movements HTTP/1.1" 200 -
[2022-11-14 03:57:37,822] ERROR in app: Exception on /add_product_movements [POST]
Traceback (most recent call last):
```

Create a

valid Deployment.yaml file,

```
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> kubectl apply -f deployment.yaml
deployment.apps/ibmimsr created
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> █
```

Create a namespace in IBM Container registry,



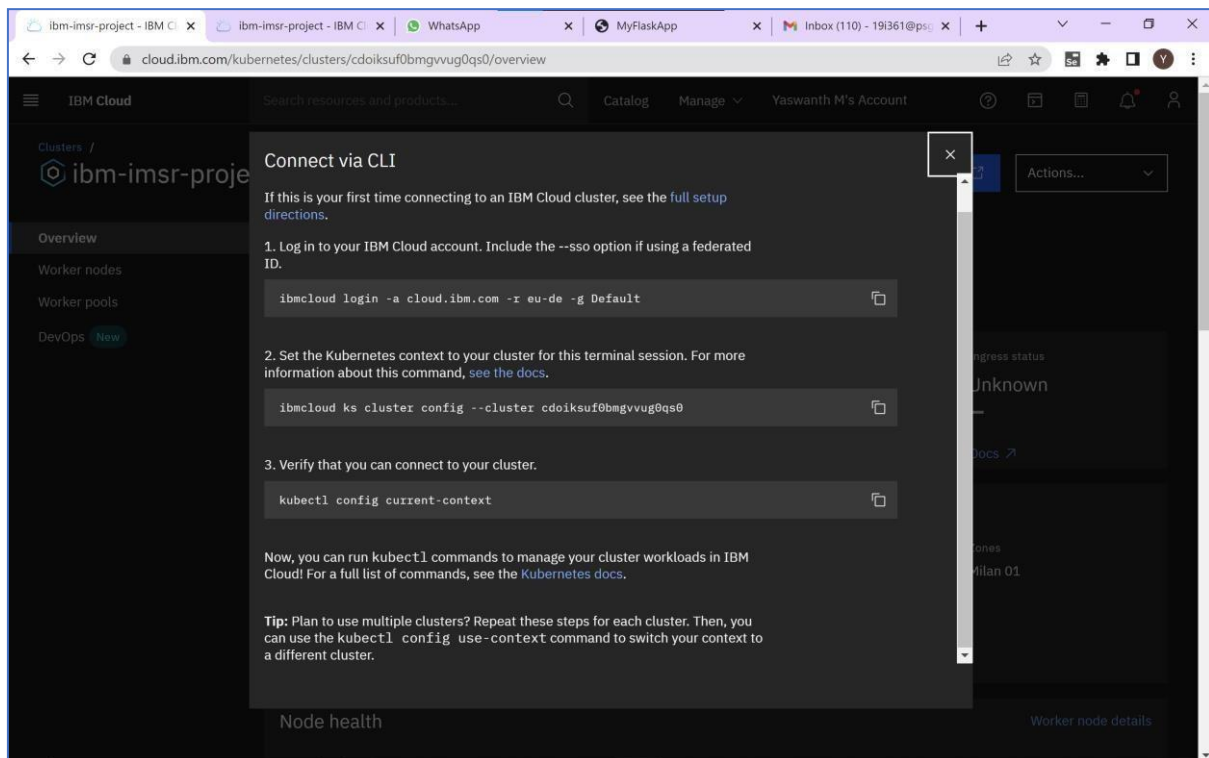
Pushing the project into IBM container Registry,

```
Select Command Prompt

C:\Users\yaswa>docker tag yaswanthmanoharan/ibm_imsr icr.io/ibm_imsr/ibm_imsr
5b3f1ed98915: Pushing [====>] 6.053MB/67.7882fd36bfd35: P
ushing 174.2MB/529MB
d5b2c4afb8d6: Pushing [=====>] 40.6MB/191.6MB
Using default tag: latest
The push refers to repository [icr.io/ibm_imsr/ibm_imsr]
6b183c62e3d7: Pushing [=====>] 6.465MB/18.48MB
d5b2c4afb8d6: Pushing [====>] 17.2MB/191.6MB
d5b2c4afb8d6: Pushing [=====>] 75.71MB/191.6MB
4db7c1329ec9: Pushed
6f6e69c2c592: Pushed
882fd36bfd35: Pushing [=====>] 308.4MB/529MB
d5b2c4afb8d6: Pushing 138.5MB/191.6MB
d5b2c4afb8d6: Pushed
6b183c62e3d7: Pushing [=====>] 5.285MB/18.48MB
882fd36bfd35: Pushing 175.3MB/529MB
882fd36bfd35: Pushing [=====>] 319MB/529M5b3f1ed98915: P
ushed
d1dec9917839: Pushing [>] 2.735MB/152M882fd36bfd35: Pushed
d1dec9917839: Pushed
d1dec9917839: Pushing 70.76MB/152MB
d38adf39e1dd: Pushed
d9d07d703dd5: Pushed
latest: digest: sha256:0575b171d321ade1d5a3def1d1bb5afe8a00d00c1f7e157a5347aca6a6ee1470 size: 3052
882fd36bfd35: Pushing [=====>] 265.7MB/529MB
C:\Users\yaswa>dshing [=====>] 264MB/529MB
d1dec9917839: Pushing [>] 1.62MB/152MB
```

Note: Create a Kubernetes Cluster in IBM Cloud and wait for the work node to get fully deployed.

Then, Login into Kubernetes Cluster using the following commands,



Expose your application using the following command and check for the port number using the next command.

```
OK
The configuration for cdoiksuf0bmgvvug0qs0 was downloaded successfully.

Added context for cdoiksuf0bmgvvug0qs0 to the current kubeconfig file.
You can now execute 'kubectl' commands against your cluster. For example, run 'kubectl get nodes'.
If you are accessing the cluster for the first time, 'kubectl' commands might fail for a few seconds while RBAC synchronizes.

C:\Users\yaswa>kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
ibm-inventory-management-system-for-retailers-6cd7dfcc7b-8q2w2    1/1     Running   0           10h
ibm-project-9bbb47d-5vn2w                                           1/1     Running   0           9h
ibmimsr-586d66c8c8-kkjqp                                           0/1     ContainerCreating   0           26s

C:\Users\yaswa>kubectl expose deployment ibmimsr --type=NodePort --name=ibmimsr
service/ibmimsr exposed

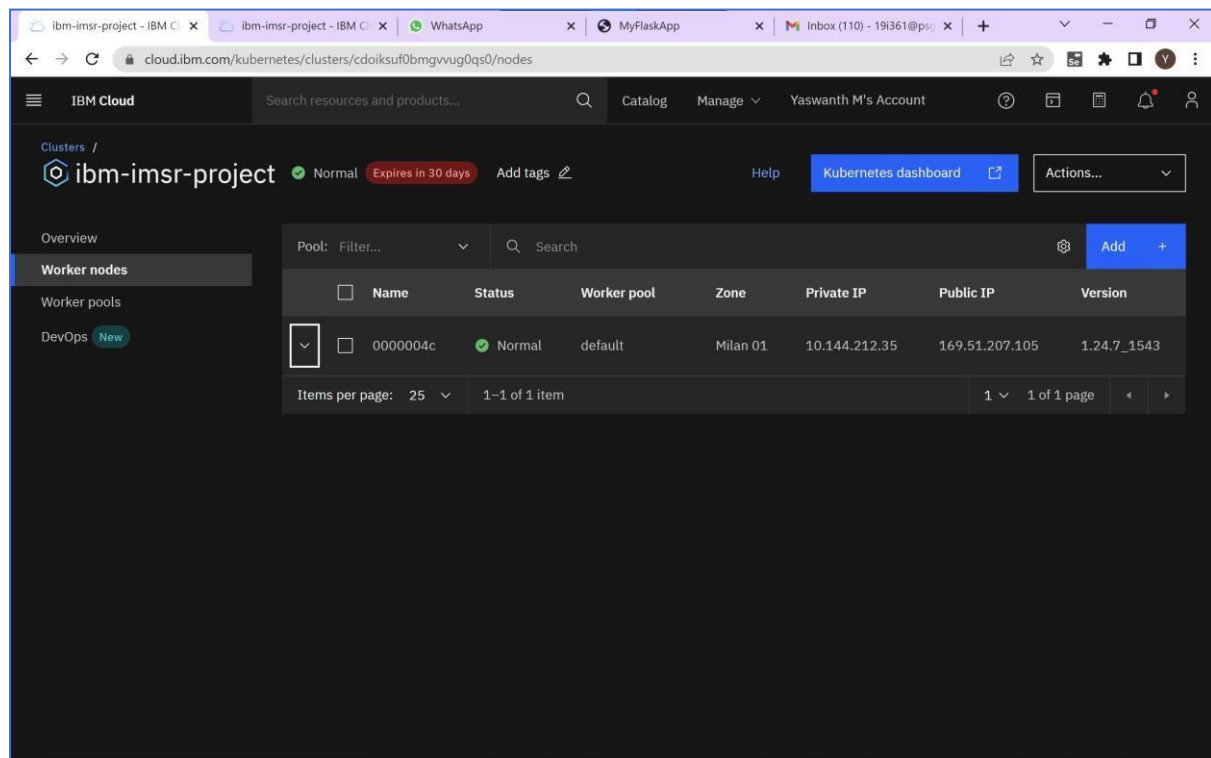
C:\Users\yaswa>kubectl describe service ibmimsr
error: unknown command "describe" for "kubectl"

Did you mean this?
    describe

C:\Users\yaswa>kubectl describe service ibmimsr
Name:                ibmimsr
Namespace:           default
Labels:              app=ibmimsr
Annotations:         <none>
Selector:            app=ibmimsr
Type:               NodePort
IP Family Policy:   SingleStack
IP Families:        IPv4
IP:                 172.21.98.28
IPs:                172.21.98.28
Port:               <unset> 5000/TCP
TargetPort:         5000/TCP
NodePort:           <unset> 30958/TCP
Endpoints:          172.30.116.13:5000
Session Affinity:   None
External Traffic Policy: Cluster
Events:             <none>

C:\Users\yaswa>
```

Then, Check for the public IP address in your IBM Kubernetes Cluster under Worker Node,

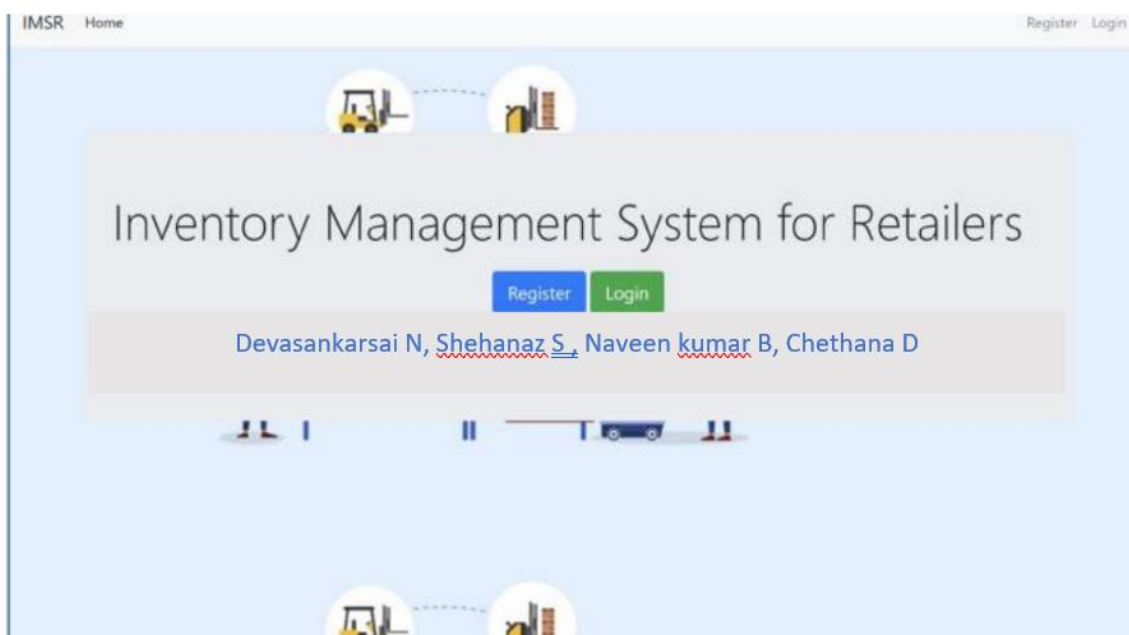


Thus we have the Public IP address and the Nodeport.

Now just type in this format - <Public_IP>:<NodePort>

For our Inventory management system application it is, **169.51.207.105:30958**

Type this in the browser and click enter to access the deployed application,



Result:

Thus In this way We developed a “Inventory management System for Retailers” using Python, Sendgrid and IBM Cloud Services (IBM DB2, IBM Container registry, IBM Kubernetes).

Thank You!