

Final Deliverables Report

Date	18.11.2022
Team ID	PNT2022TMID24681
Project Name	Inventory Management System for Retailers

Team members and their Contribution:

Name	Contribution
P RATISH	Frontend – 5 Pages, Integration of Sendgrid, Deployment of using docker and Kubernetes.
SURENDIRAN S	Frontend – 5 Pages, Documentation
SCHWARTZ A	Frontend – 4 Pages, Documentation
DINESHMONGIYA G	Backend Fully (For all 14 Pages), Integration of IBM Cloud, Deployment of using docker and Kubernetes.

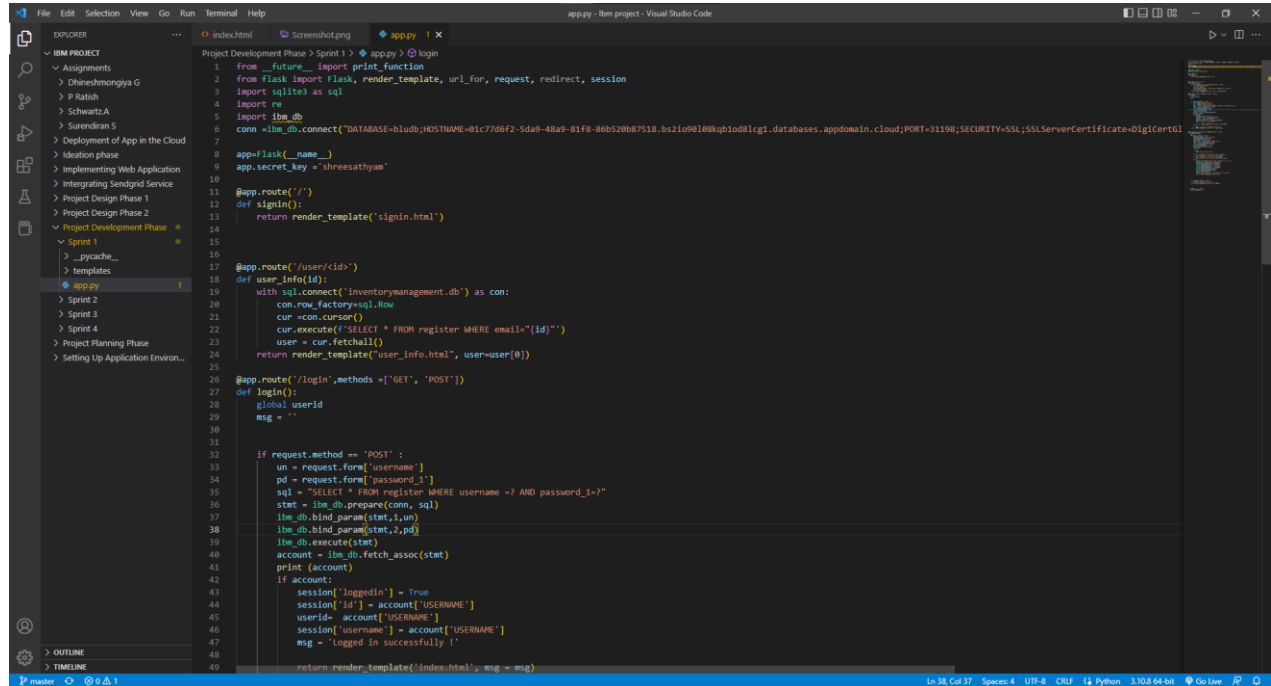
Introduction:

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)

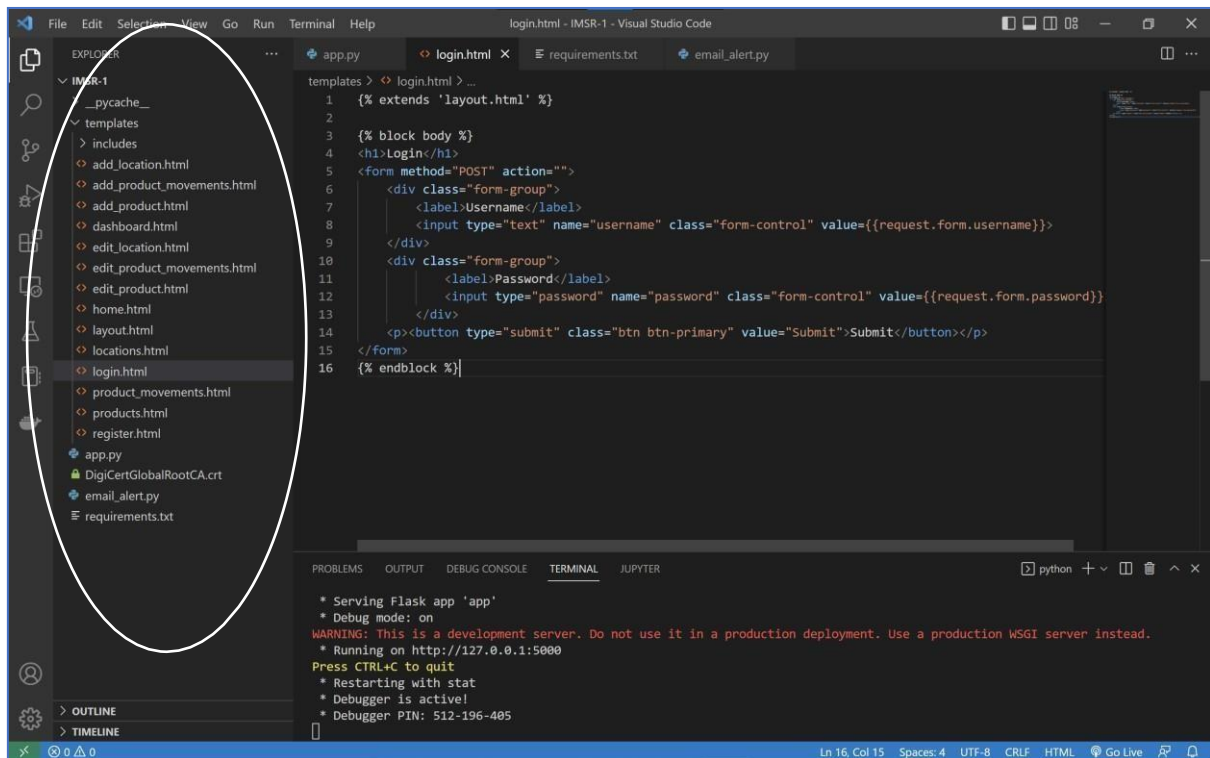


```
1 from flask import Flask, render_template, url_for, request, redirect, session
2 from flask import Flask, render_template, url_for, request, redirect, session
3 import sqlalchemy as sql
4 import re
5 import ibm_db
6 conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=61c7796f2-5da9-48a9-81f8-86b520b87518.bs2io9el08kqbi0d1cgt.databases.appdomain.cloud;PORT=31198;SECURITY=SSL;SSLServerCertificate=DigiCertG
7
8 app = Flask(__name__)
9 app.secret_key = "shreessathyan"
10
11 @app.route('/')
12 def signin():
13     return render_template('signin.html')
14
15
16 @app.route('/user/cld')
17 def user_info(id):
18
19     with sql.connect("Inventorymanagement.db") as con:
20         con.row_factory = sql.Row
21         cur = con.cursor()
22         cur.execute("SELECT * FROM register WHERE email= '{id}'")
23         user = cur.fetchall()
24         return render_template("user_info.html", user=user[0])
25
26 @app.route('/login', methods = ['GET', 'POST'])
27 def login():
28     global userid
29     msg = ''
30
31
32     if request.method == 'POST':
33         un = request.form["username"]
34         pd = request.form["password_1"]
35         sql = "SELECT * FROM register WHERE username =? AND password_1=?"
36         stmt = ibm_db.prepare(conn, sql)
37         ibm_db.bind_param(stmt, 1, un)
38         ibm_db.bind_param(stmt, 2, pd)
39         ibm_db.execute(stmt)
40         account = ibm_db.fetch_assoc(stmt)
41         print(account)
42         if account:
43             session['loggedin'] = True
44             session['id'] = account['USERNAME']
45             user_id = account['USERID']
46             session['username'] = account['USERNAME']
47             msg = 'Logged in successfully !'
48
49     return render_template('index.html', msg = msg)
```

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)



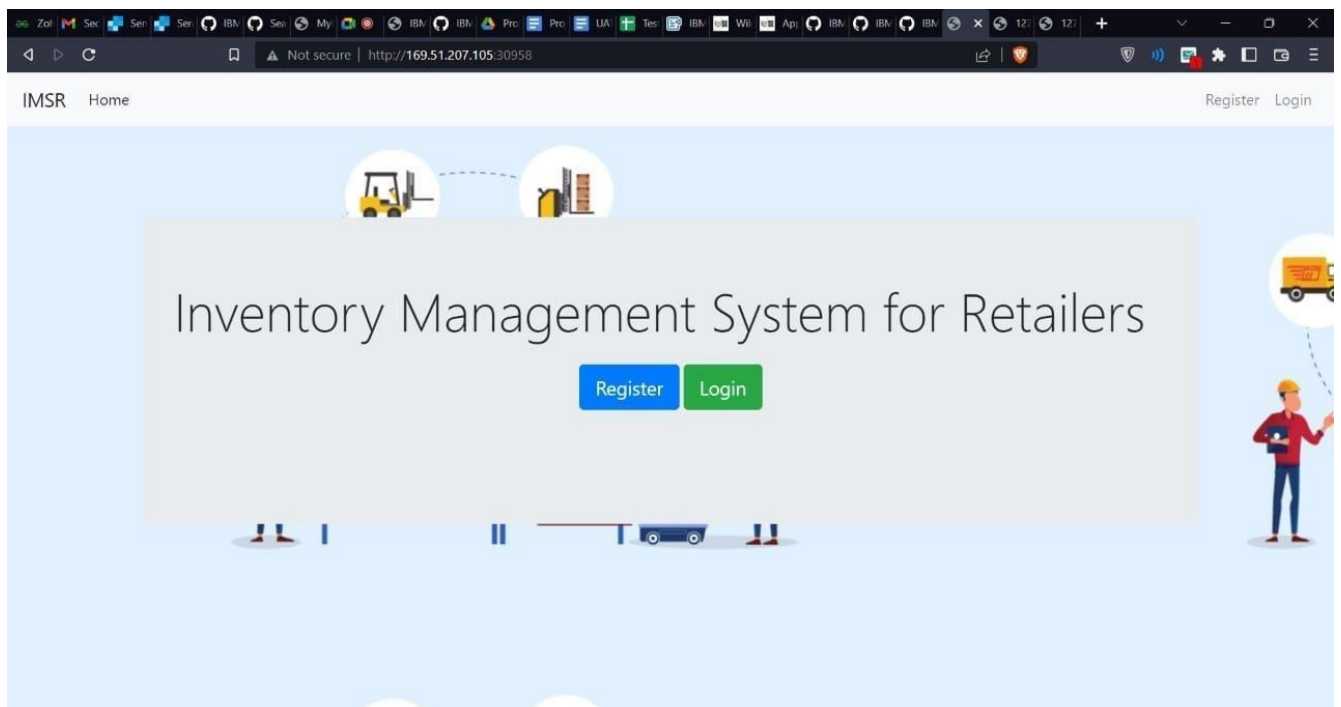
```
login.html - IMSR-1 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
login.html x requirements.txt email_alert.py
templates > login.html > ...
1 {% extends 'layout.html' %}
2
3 {% block body %}
4 <h1>Login</h1>
5 <form method="POST" action="">
6   <div class="form-group">
7     <label>Username</label>
8     <input type="text" name="username" class="form-control" value="{{request.form.username}}>
9   </div>
10  <div class="form-group">
11    <label>Password</label>
12    <input type="password" name="password" class="form-control" value="{{request.form.password}}>
13  </div>
14  <p><button type="submit" class="btn btn-primary" value="Submit">Submit</button></p>
15 </form>
16 {% endblock %}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

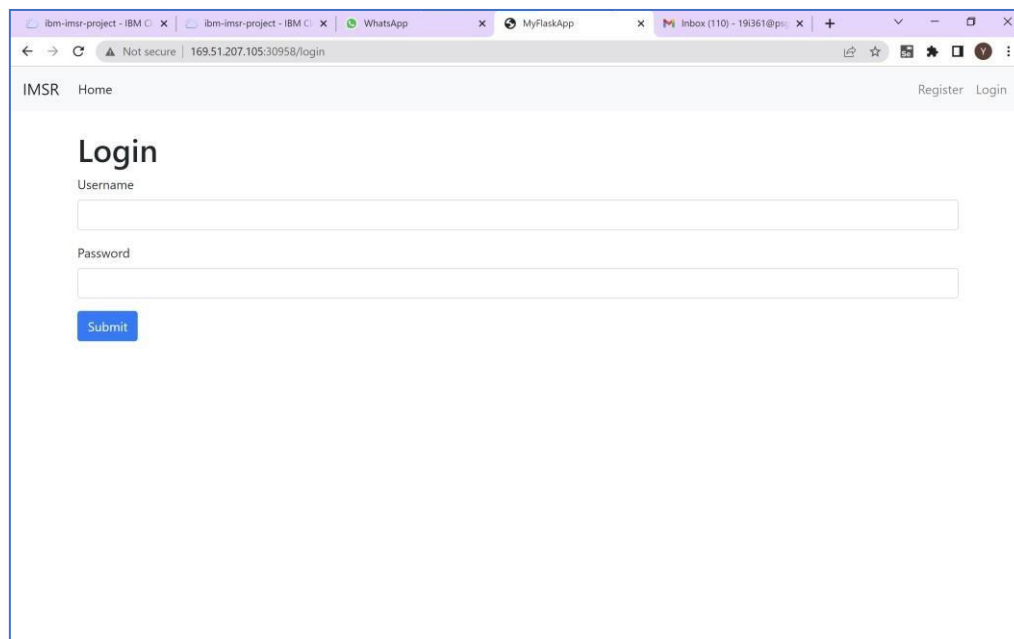
```
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 512-196-405
```

Sample FrontEnd Pages,

Home Page,



Login Page,



A screenshot of a web browser displaying the IMSR Login page. The browser's address bar shows the URL "169.51.207.105:30958/login". The page has a header with "IMSR Home" on the left and "Register Login" on the right. The main content area features a "Login" heading, followed by "Username" and "Password" labels, each with a corresponding text input field. A blue "Submit" button is positioned below the password field.

IMSR Home Register Login

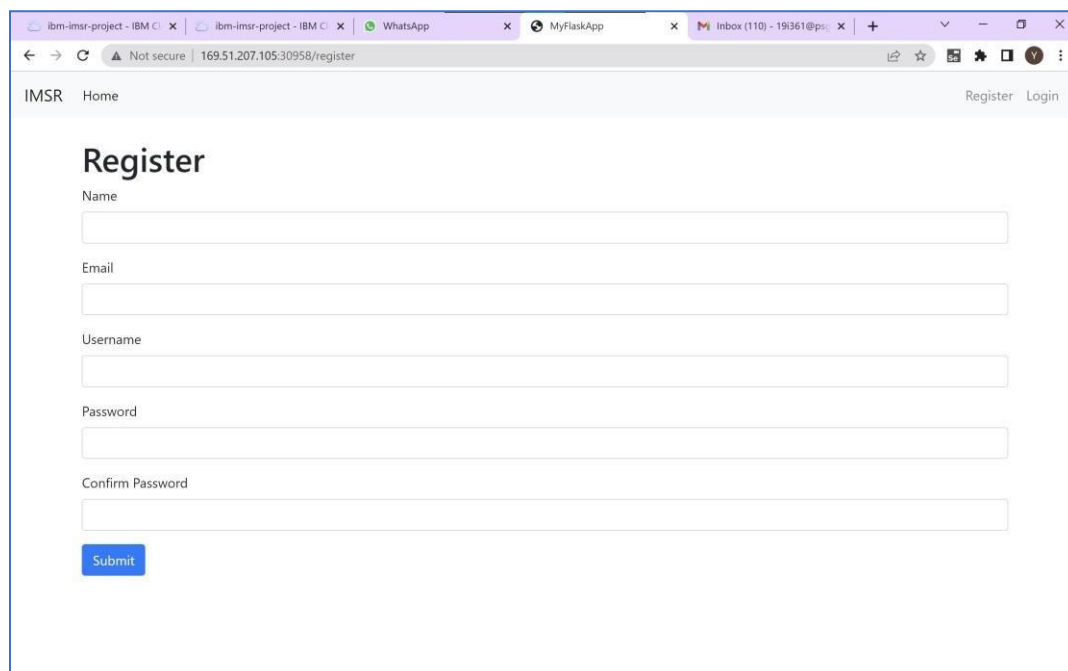
Login

Username

Password

Submit

Register Page,



A screenshot of a web browser displaying the IMSR Register page. The browser's address bar shows the URL "169.51.207.105:30958/register". The page has a header with "IMSR Home" on the left and "Register Login" on the right. The main content area features a "Register" heading, followed by "Name", "Email", "Username", "Password", and "Confirm Password" labels, each with a corresponding text input field. A blue "Submit" button is positioned below the "Confirm Password" field.

IMSR Home Register Login

Register

Name

Email

Username

Password

Confirm Password

Submit

Products Page,

ibm-imsr-project - IBM C x ibm-imsr-project - IBM C x WhatsApp x MyFlaskApp x Inbox (110) - 19361@psd x

Not secure | 169.51.207.105:30958/products

IMSR Home Products Location Product Movements Logout Dashboard

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,

ibm-imsr-project - IBM C x ibm-imsr-project - IBM C x WhatsApp x MyFlaskApp x Inbox (110) - 19361@psd x

Not secure | 169.51.207.105:30958/product_movements

IMSR Home Products Location Product Movements Logout Dashboard

Product Movements

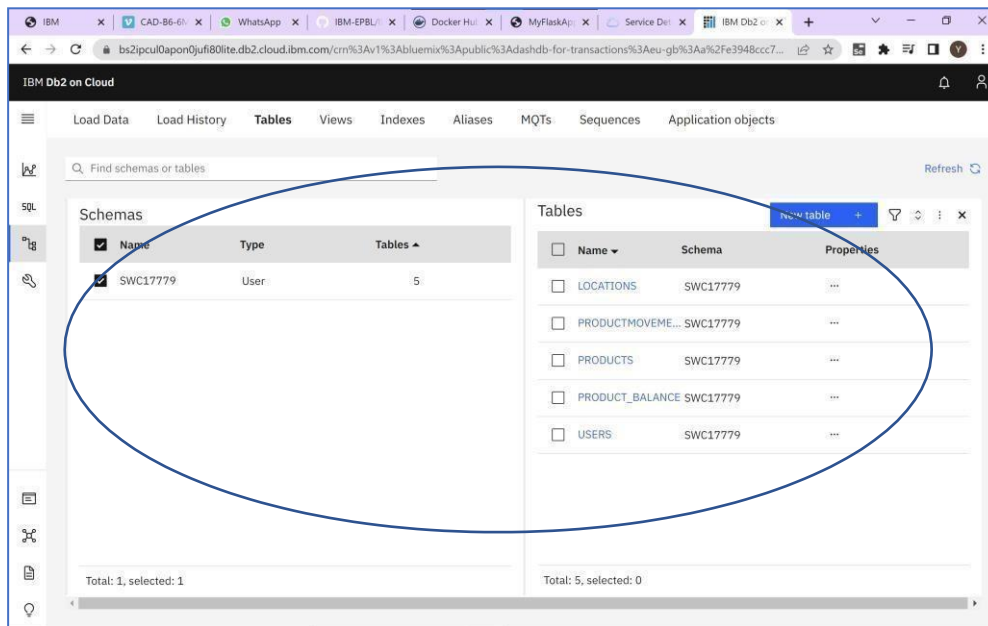
Add Product Movements

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

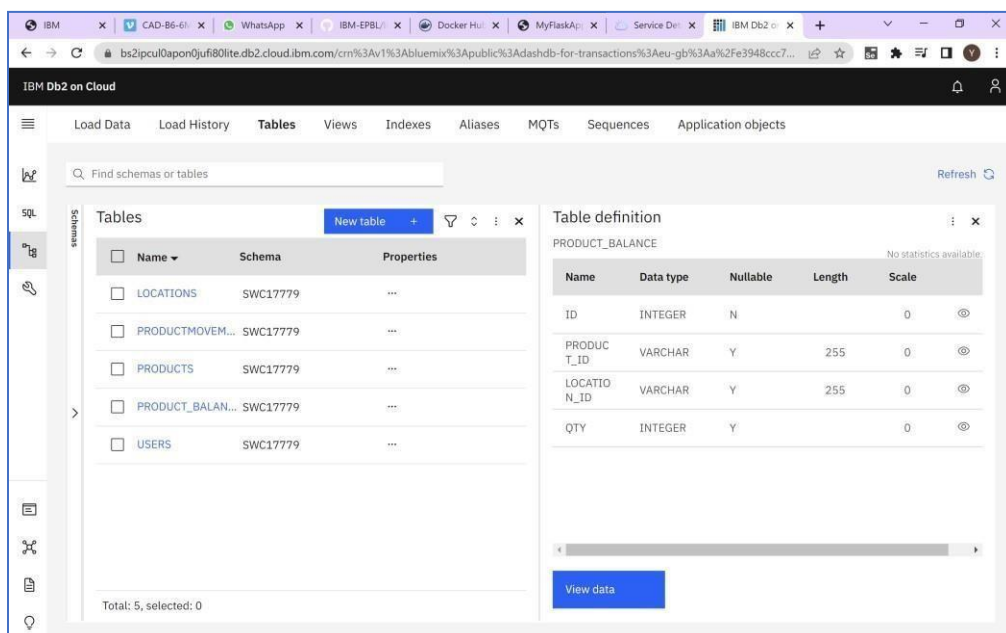
Sprint 3 - IBM Cloud Integration + Integration of SendGrid:

IBM Cloud Integration:

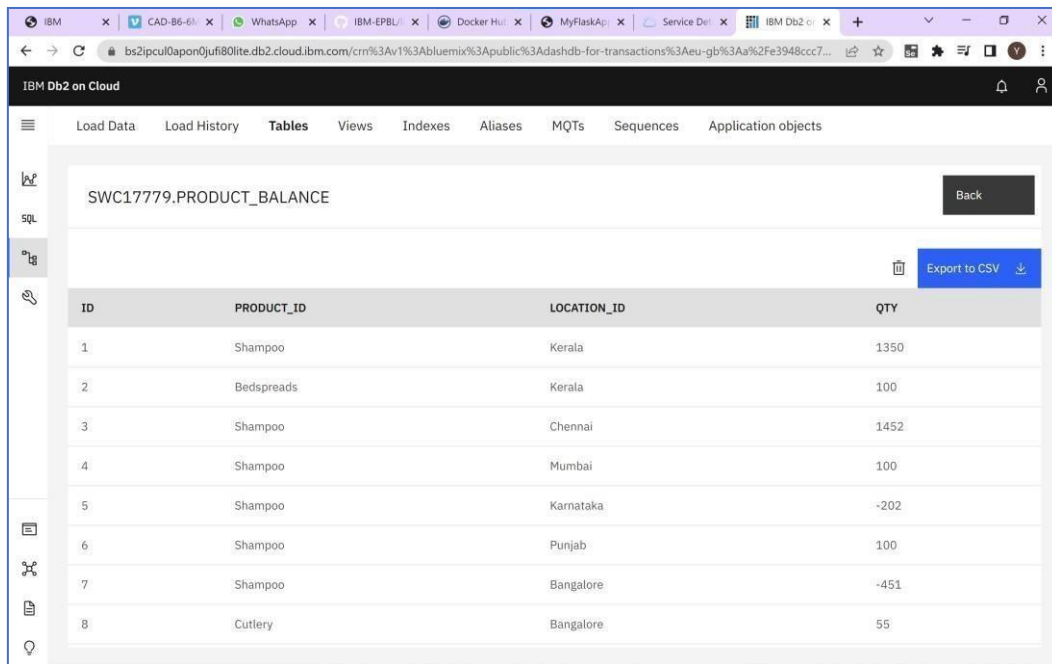
5 tables created for our project,



Schema of the particular table (For Example, Product_Balance)



Data of a particular table (For Example, Product_Balance)



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

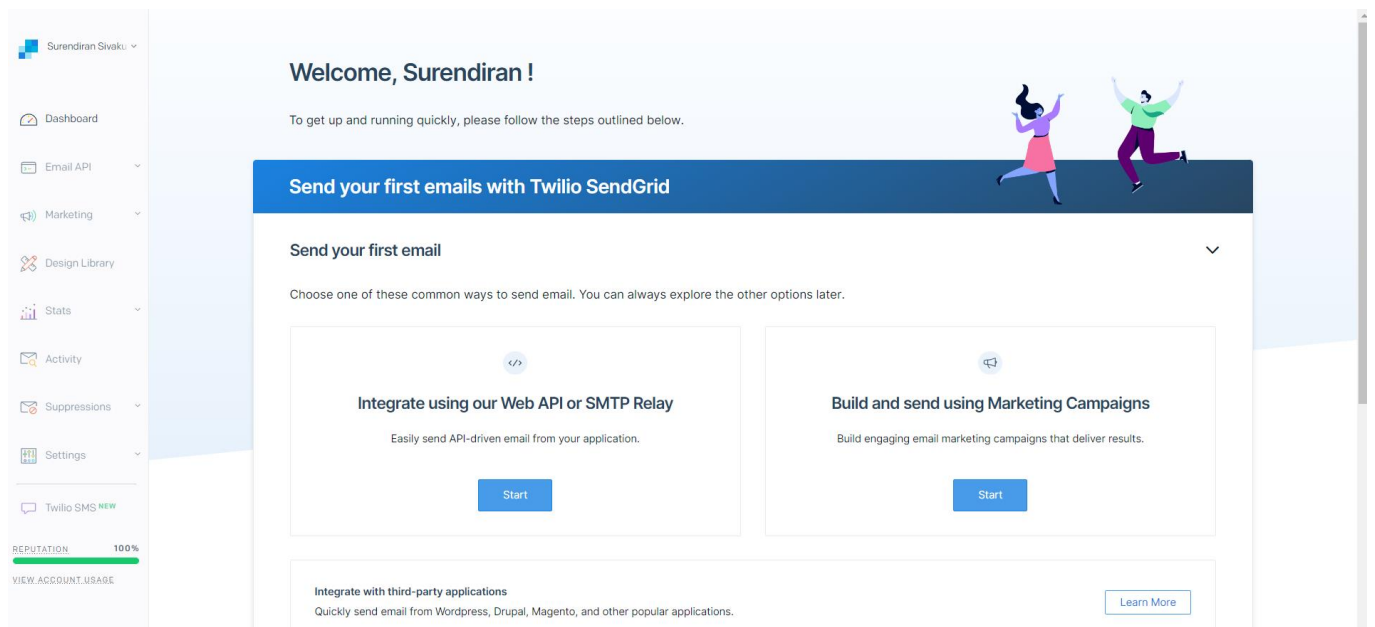
Code for Connection of IBM Database,

```
conn=ibm_db.connect("DATABASE=bludb;HOSTNAME=55fbc997-9266-4331-afd3-888b05e734c0.bs2io90I08kqb1od8lcg.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,



Welcome, Surendiran !

To get up and running quickly, please follow the steps outlined below.

Send your first emails with Twilio SendGrid

Send your first email

Choose one of these common ways to send email. You can always explore the other options later.

Integrate using our Web API or SMTP Relay

Easily send API-driven email from your application.

[Start](#)

Build and send using Marketing Campaigns

Build engaging email marketing campaigns that deliver results.

[Start](#)

Integrate with third-party applications

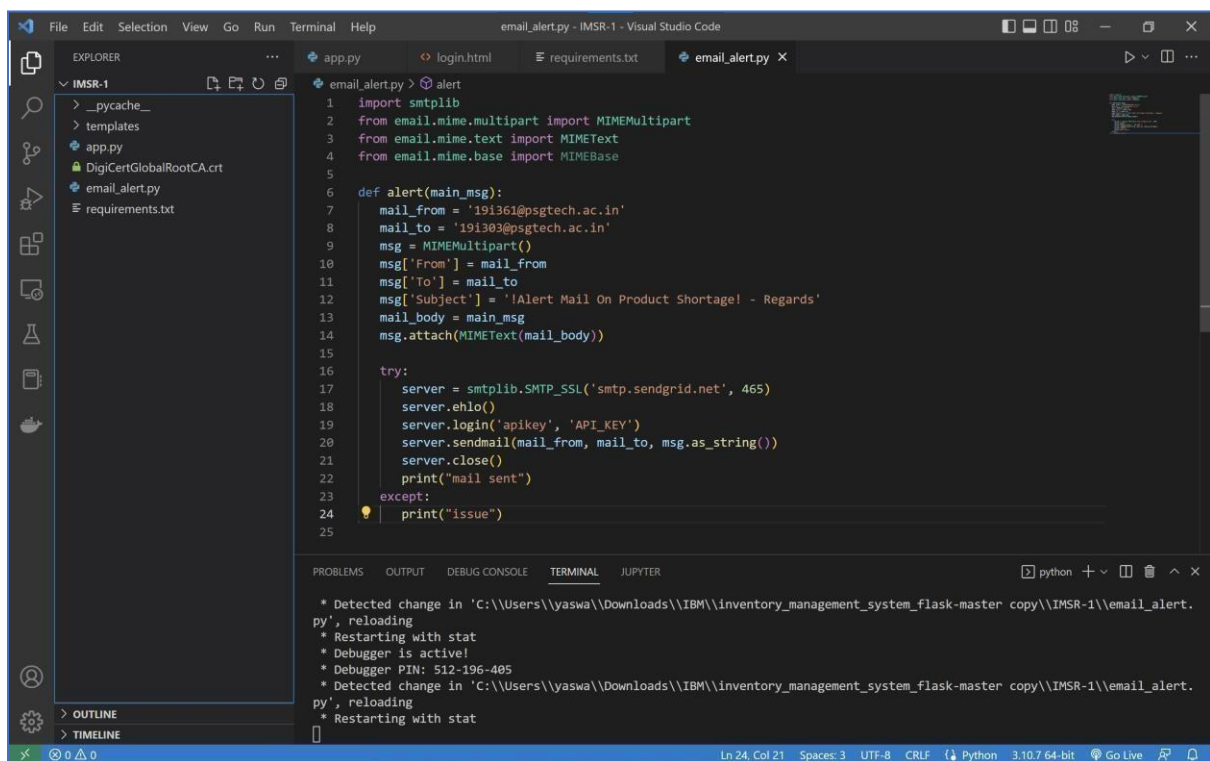
Quickly send email from Wordpress, Drupal, Magento, and other popular applications.

[Learn More](#)

REPUTATION 100%

VIEW ACCOUNT USAGE

Code for email alert:



The screenshot shows the Visual Studio Code interface with the file explorer on the left displaying the project structure for 'IMSR-1'. The main editor window shows the code for 'email_alert.py'. The code imports 'smtplib' and 'email.mime' modules, defines an 'alert' function that constructs an email message with a subject 'Alert Mail On Product Shortage! - Regards', and attempts to send it using 'SMTP_SSL' on 'smtp.sendgrid.net'. The terminal at the bottom shows the output of running the script, indicating a successful email send.

```
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
```

Terminal Output:

```
* 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.

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19044.2130]
(c) Microsoft Corporation. All rights reserved.

C:\Users\yaswa\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1>docker login
Authenticating with existing credentials...
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go/access-tokens/

C:\Users\yaswa\Downloads\IBM\inventory_management_system_flask-master copy\IMSR-1>
```

Building an image for our project,

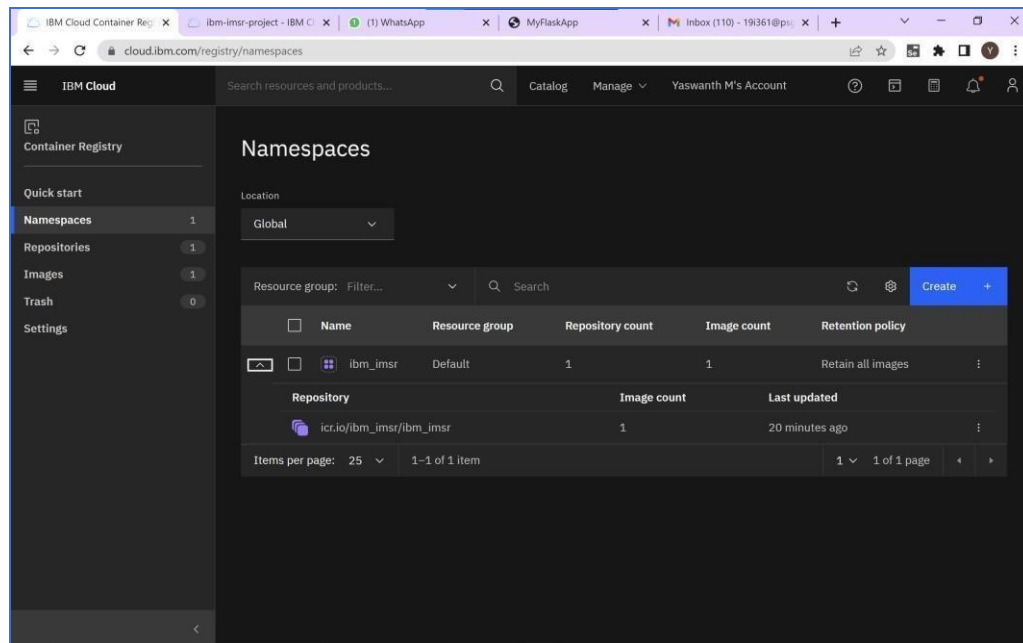
```
File "/usr/local/lib/python3.11/site-packages/flask/app.py", line 1820, in full_dispatch_request
PS C:\Users\yaswa\Downloads\IBM\IMSR-1> docker build -t yaswanthmanoharan/ibm_imsr .
[+] Building 2.7s (11/11) FINISHED
=> [internal] load build definition from Dockerfile                                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 server 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):
```

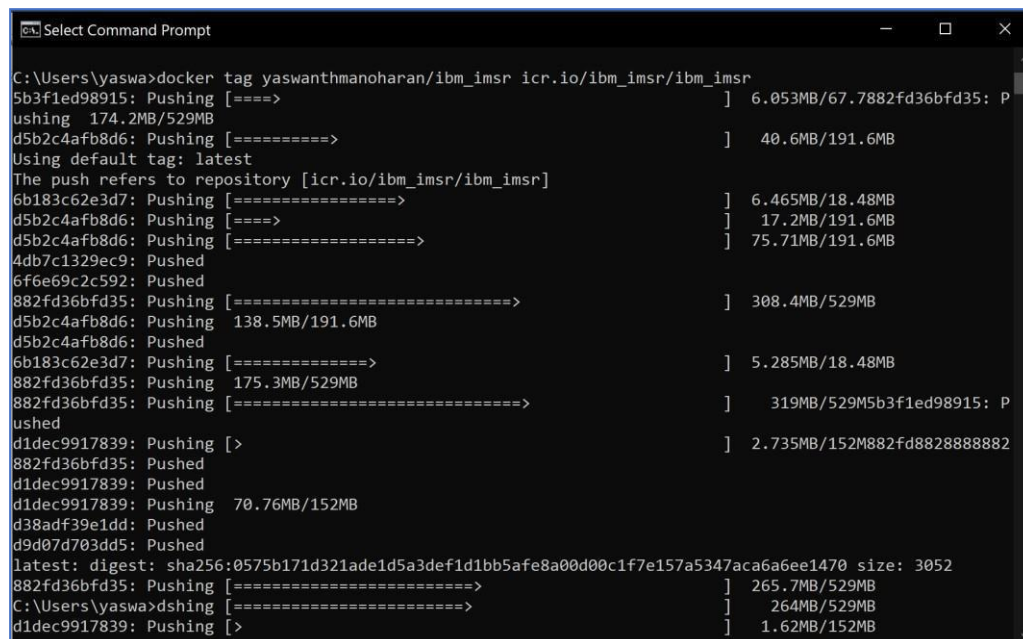
Create a valid Deployment.yaml file,

```
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,

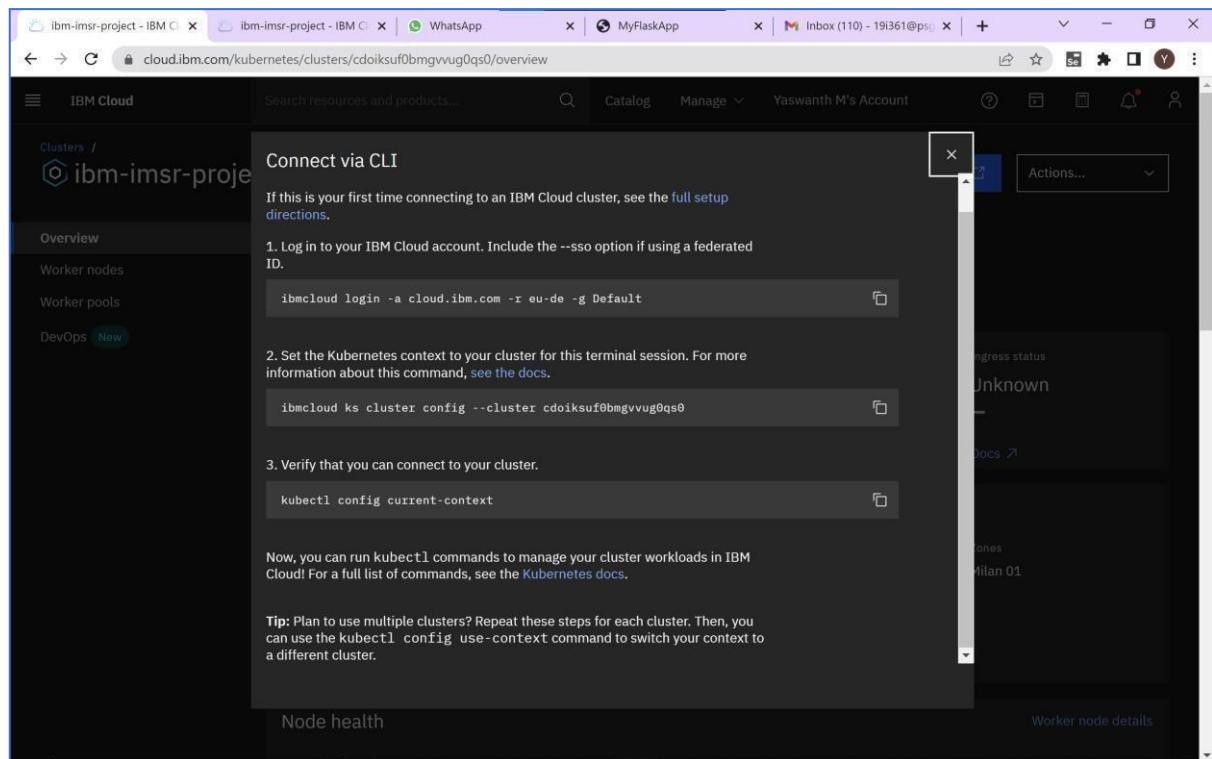


Pushing the project into IBM container Registry,

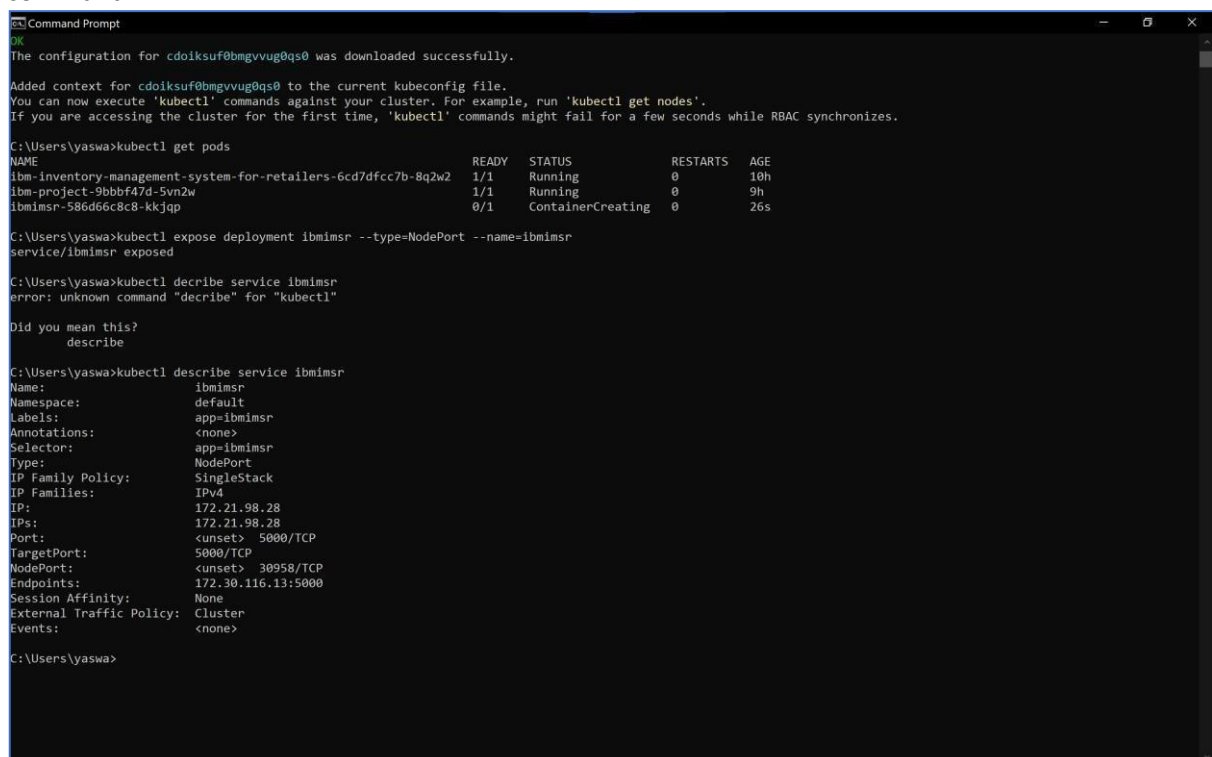


Note: Create a Kubernetes Cluster in IBM Cloud and wait for the work node to get fully deployed.

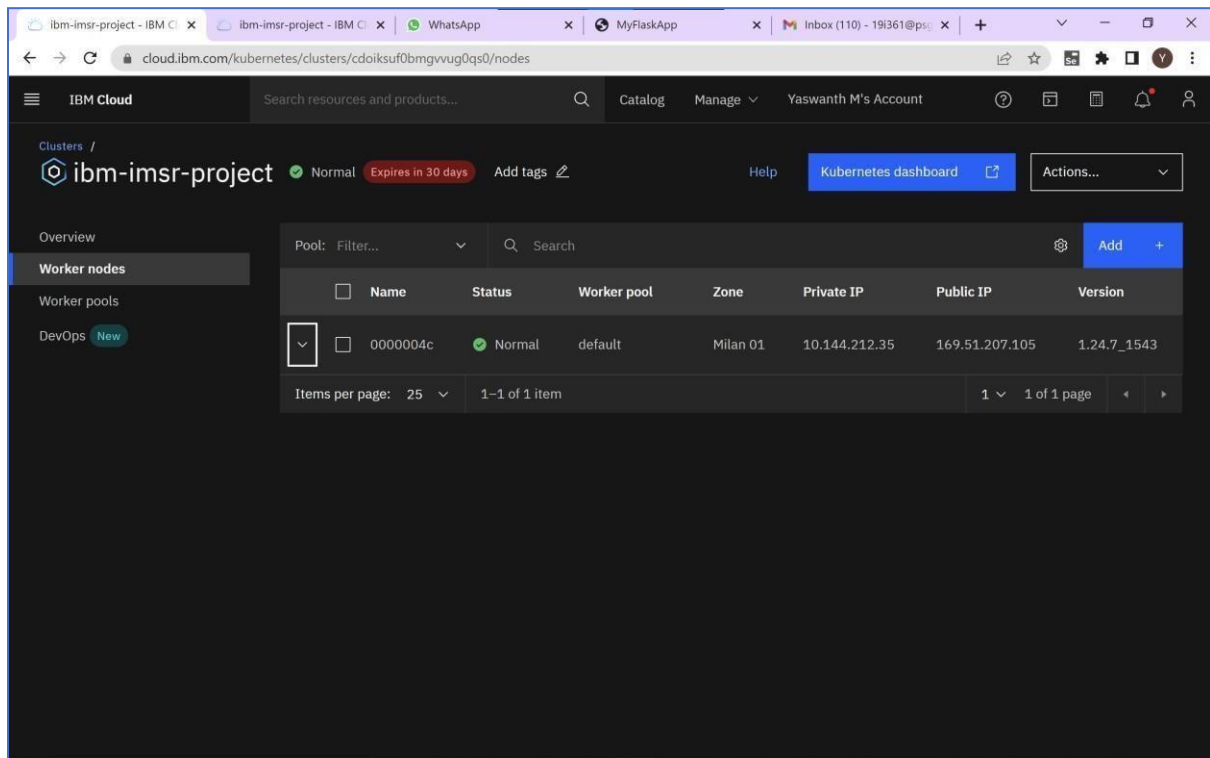
Then, Login into Kubernetes Cluster using the following commands,



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

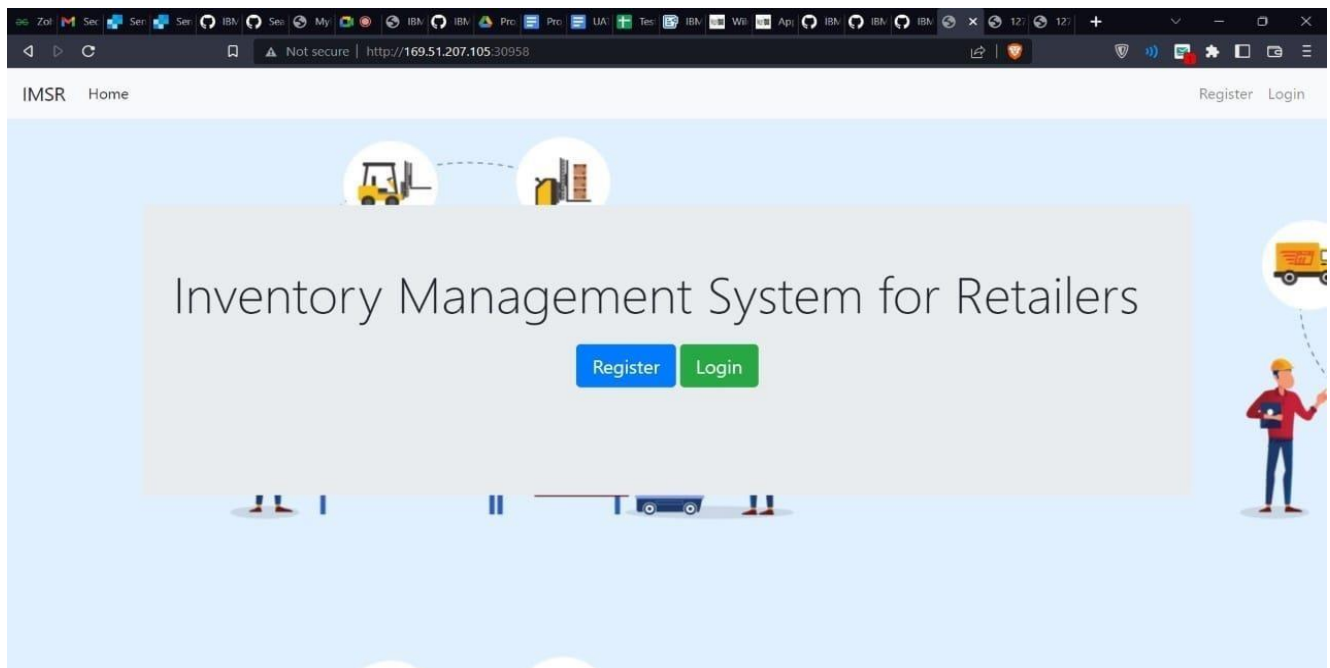


Then, Check for the public IP address in your IBM Kubernetes Cluster under Worker Node,



The screenshot shows the IBM Cloud console interface. The left sidebar has a menu with 'Overview', 'Worker nodes', 'Worker pools', and 'DevOps'. The main content area is titled 'ibm-imsr-project' and shows a table of worker nodes. The table has columns for Name, Status, Worker pool, Zone, Private IP, Public IP, and Version. One node is listed with the name '0000004c' and a public IP of '169.51.207.105'.

Name	Status	Worker pool	Zone	Private IP	Public IP	Version
0000004c	Normal	default	Milan 01	10.144.212.35	169.51.207.105	1.24.7_1543



The screenshot shows a web browser displaying the IMSR (Inventory Management System for Retailers) application. The page has a light blue background with illustrations of a forklift, a warehouse, and a delivery truck. The main heading is 'Inventory Management System for Retailers', and there are 'Register' and 'Login' buttons. The browser's address bar shows the URL 'http://169.51.207.105:30958'.

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!