

PLASMA DONOR APP

1. INTRODUCTION

1.1 Overview

Plasma donor App is an online web based project. During the COVID-19 crisis, requirement for the plasma becomes very high and donor count being low, saving the donor information and helping the needy notifying the current donors would be a helping hand. In regard to the problem faced, an application is to be built which would take the donor's details, store them and inform them upon a request. People can help us by registering the Plasma Donor app if they are willing to donate their plasma when needed. Person who want to donate may login our website with the help of Username and Password. The person who needs the plasma donor, they can search and find the plasma donor using our website.

1.2 Purpose

The purpose is mainly towards people who are willing to donate the plasma to the patients. Through this system, it will be easier to find the donor for the exact blood type and easy to build the connection between the donor and patient. The main intend of building this software is to formal the procedure of plasma donation and motivate the donors to donate the plasma.

2. LITERATURE SURVEY

2.1 Existing Problem

In the present scenario searching for plasma donors can takes place through blood bank centers or by toll free numbers. So far it is a time taken process, because it involves lots of manual work. It is s waste of time to go to every blood bank if the plasma of particular group is not available and most of the time user has to wait in a queue.

Most of the application developed for this system are monolithic application. A monolithic application is built as single unit which consist of a database, a client side user interface and server side application. To make any changes to this type of enterprise application a developer must build and deploy an updated version of server-side application.

2.2 Proposed Solution

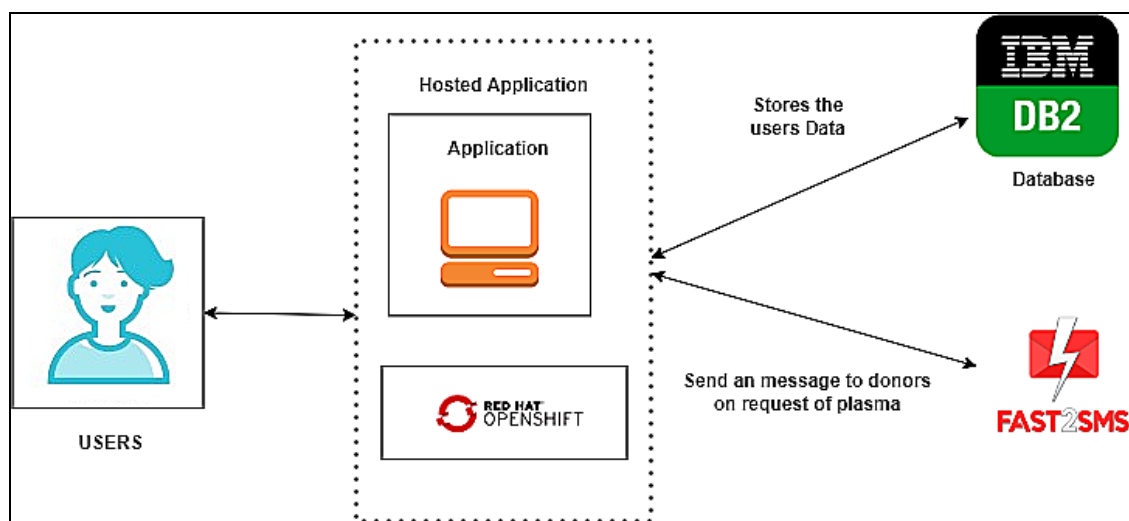
The project is mainly towards people who are willing to donate the blood to the patients. Through this system it will be easier to find exact blood type and easy to build the connection between the donor and needy person. Users need to register an account and login to the application. Once user logs in, he will have a dashboard to view the total number of donors and count of people

with specific blood groups. User will have the option to request the blood. Once user requests, all the people with that blood group will be notified with SMS.

In contrast to monolithic application, A microservice architecture — in concert with cloud deployment technologies, API management, and integration technologies — provides a different approach to software development. The monolith is instead disassembled into a set of independent services that are developed, deployed and maintained separately. In this project we are deploying our app on cloud platform named as Red Hat Open-Shift Dedicated dev space.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



This architecture consists of User, Web Application ,Cloud platform(Red Hat Open-shift Dedicated) , and IBM DB2 which is a data management software.

- User can interact with the application
- The person who is willing to donate the plasma can register using registration form.
- These details get stored in IBM DB2 database.
- If user(needy patient) posts a request then concerned blood group donor will get notification through SMS.
- This web application we can deploy locally as well as on Red hat open-shift Dedicated. To deploy the application on Red hat open-shift, we need to create the docker image file

3.2 Hardware / Software designing

Hardware requirements

1GHz or High Processor
512MB or High RAM
Min 500MB Hard Disk

Good Internet Connection

Software requirements

1. Setting up Application Environment:
Set up your environment with Python IDLE/any other Editor (VisualStudio code) for deploying application locally as well as on Red Hat Open-shift.
2. Install necessary libraries: Flask,ibm_db
3. Create IBM DB2 service in IBM Cloud, You will require to register for an IBM account to get access to the various services of IBM Cloud from catalog
4. Github Account :
Create a new Repository in your Github account and upload the source code
5. Create Red Hat account and login into Red Hat Open shift Dedicated Console. This account is used to deploy your web app into Red Hat Open shift dev space. You will require the docker image of your app to deploy the application.

4. EXPERIMENTAL INVESTIGATIONS

The main purpose of this project is towards people who are willing to donate blood to the needy patients. To achieve this objective, we need to store the person details on a special platform so that we can access them quickly without wasting time. The IBM cloud platform combines platform as a service (PaaS) with the infrastructure as a service (IaaS) to provide an integrated experience. Globally deployed across data centers around the world, the solution you build on IBM Cloud® spins up fast and performs reliably in a tested and supported environment you can trust. IBM Cloud provides solutions that enable higher levels of compliance, security, and management, with proven architecture patterns and methods for rapid delivery for running mission-critical workloads.

4.1 Storing data on IBM DB2 Cloud

IBM DB2 on cloud is a trusted data source for apps , it is a fully managed SQL cloud database that offers dedicated operations team point-in-time recovery high-availability disaster recovery (HADR) technology with multi-zone region support and independent scaling to protect your enterprise application. You can use Db2 on Cloud just as you would use any database software, but without the time and expense of hardware setup or software installation and maintenance. You can connect command-line interfaces, IBM or third-party applications and tools, or apps that you create to your Db2 on Cloud database.

Prerequisites

Before attempting to connect to your Db2 on Cloud database, verify that you have the prerequisites.

1. Collect database details and credentials

To connect to your database, you need database details (such as the host name), and credentials (such as a user ID and password.) You can collect this connection information from the Service credentials table of the Db2 on Cloud service instance or from the Db2 on Cloud web console.

2. Verify that a supported driver is installed

If your application or tool already contains the Db2 v11.5 IBM Data Server Driver Package, then your application or tool is able to connect to your Db2 on Cloud database by using that driver. Otherwise, install the Db2 driver package

3. Configure your environment

Add entries to the driver configuration file, db2dsdriver.cfg, for your database.

4. Secure Sockets Layer (SSL)

Connection details, such as which port to use and the connection string, depend on whether you use SSL connections. For Enterprise and Standard plans, SSL is the default connection method. For legacy plans, using SSL is strongly recommended because of the stronger security it provides.

In this project we have integrated the python flask application with IBM DB2 service.

NAME	EMAIL	PHONE	CITY	INFECT	BLOOD	PASSWORD
Rohini	rdnwork411@gmail.com	45632566666	Nashik	infected	AB Positive	1234567
User1	user123@gmail.com	1234567892	Pune	infected	AB Positive	1234568
User3	user3@gmail.com	23456789	Nashik	infected	A Negative	user3
user4	user4@gmail.com	34567899	Pune	infected	AB Positive	user4

4.2 Deploying the App on Red Hat Open shift dedicated

OpenShift Dedicated is professionally managed by Red Hat and hosted on Amazon Web Services (AWS) or Google Cloud Platform (GCP). Each OpenShift Dedicated cluster comes with a fully managed control plane (Control and Infrastructure nodes), application nodes, installation and management by Red Hat Site Reliability Engineers (SRE), premium Red Hat Support, and cluster services such as logging, metrics, monitoring, notifications portal, and a cluster portal.

Kubernetes

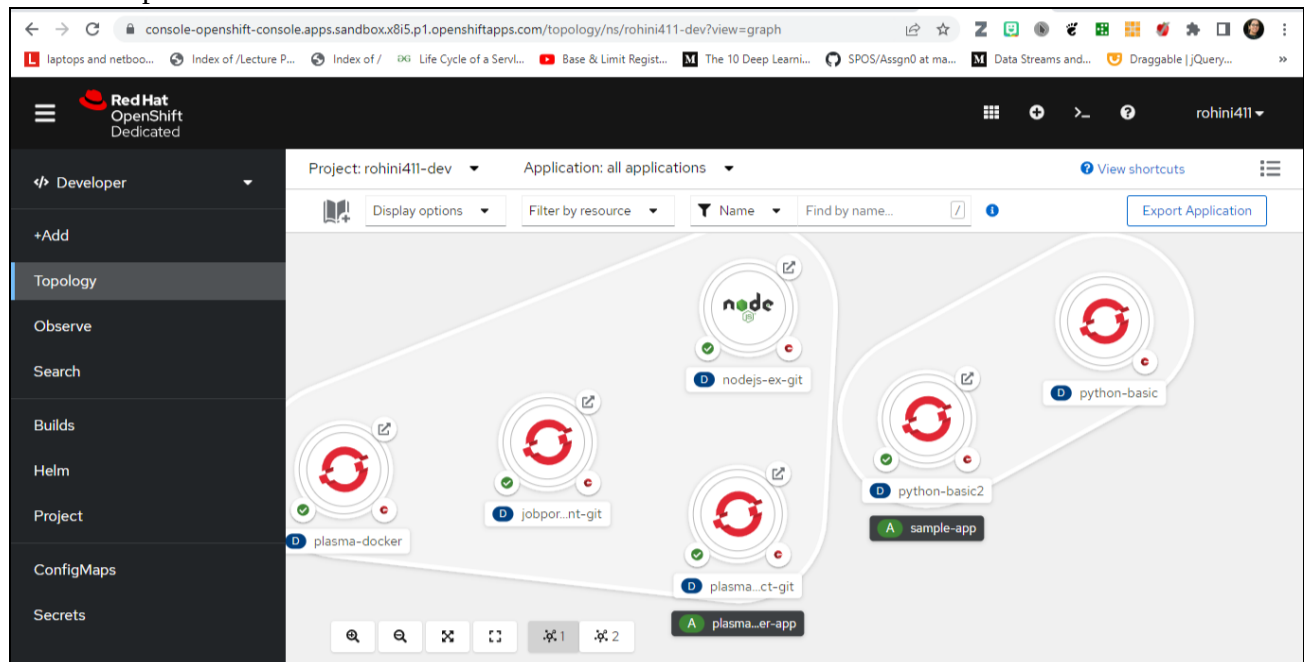
Kubernetes is an open source container orchestration engine for automating deployment, scaling, and management of containerized applications. The general concept of Kubernetes is fairly simple:

- Start with one or more worker nodes to run the container workloads.

- Manage the deployment of those workloads from one or more control nodes.

- Wrap containers in a deployment unit called a pod. Using pods provides extra metadata with

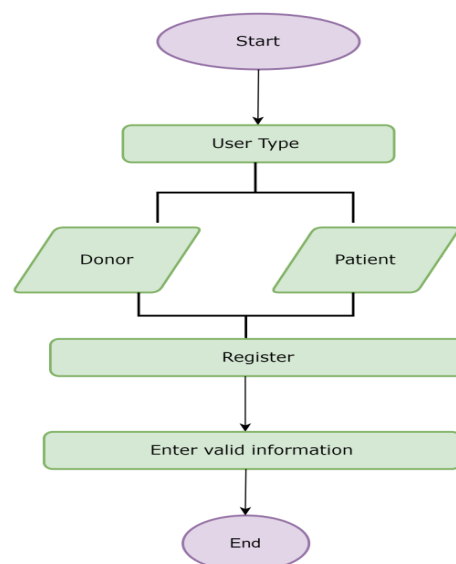
the container and offers the ability to group several containers in a single deployment entity. Create special kinds of assets. For example, services are represented by a set of pods and a policy that defines how they are accessed. This policy allows containers to connect to the services that they need even if they do not have the specific IP addresses for the services. Replication controllers are another special asset that indicates how many pod Replicas are required to run at a time. We can use this capability to automatically scale your application to adapt to its current demand.

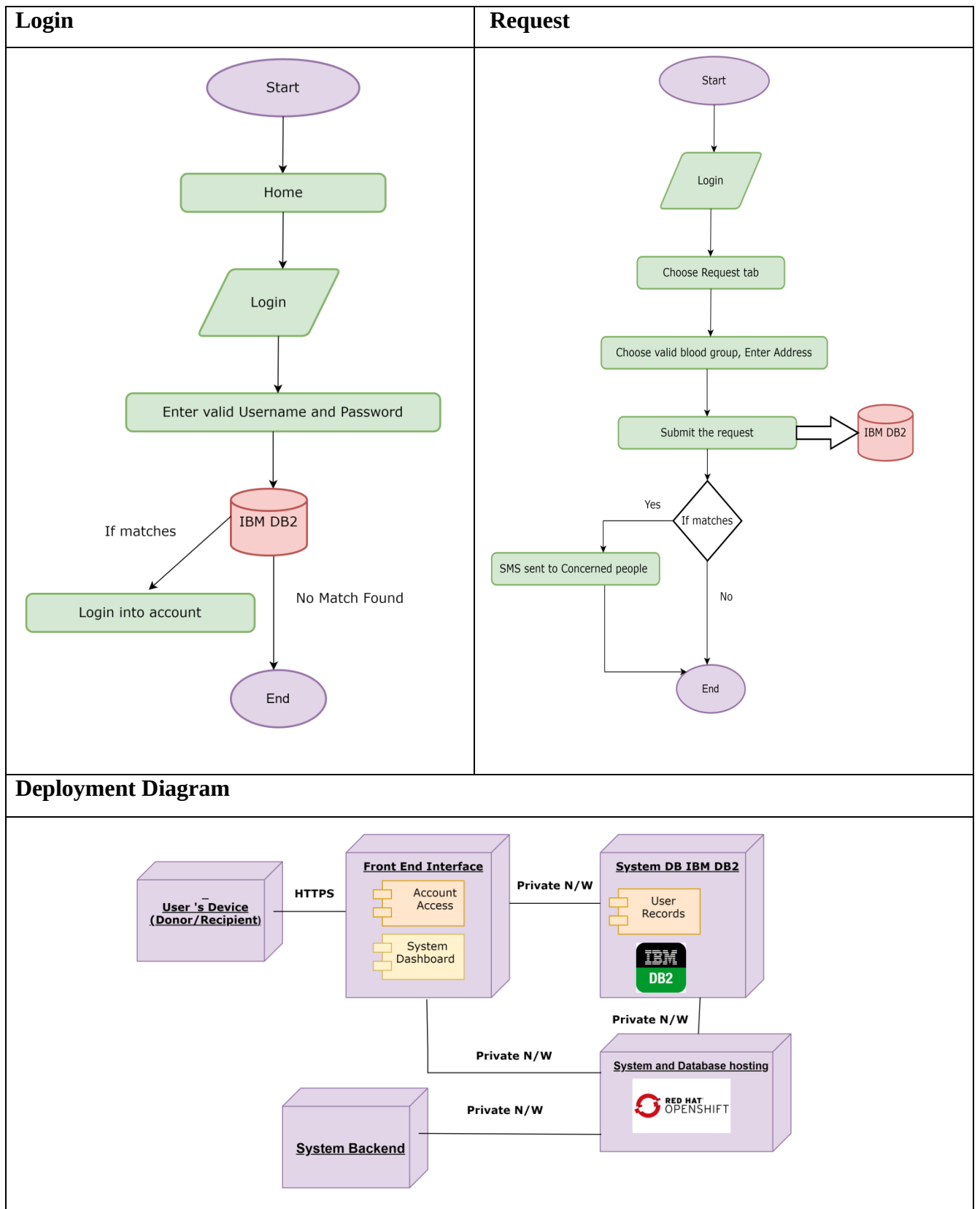


In this project we have deployed our app on Red Hat open-shift dedicated as shown in the Topology tab.

5. FLOWCHART

Registration





6. RESULT

The user interface and screenshots of the project for local deployment are given below

Registration page: On this Home page user will register himself/herself as donor/receiver

The image displays two screenshots of a web application titled "Plasma Donor App". The top screenshot shows the registration form with the following fields and values: Username (user4), Email (user4@gmail.com), Mobile Number (34567899), City (Pune), COVID Infection Status (Infected), Blood Group (AB Positive), and Password (masked with dots). A "Register" button is at the bottom. The bottom screenshot shows the same form with placeholder text: "Enter Your Name", "Enter Email", "Enter 10-digit mobile number", "Enter Your City Name", "Select COVID infection status", "Choose your blood group", and "Enter Password". Below the form, a message states: "Registration Successful, please login using your details".

IBM DB2 Console : After successful registration data will get stored on IBM DB2 cloud. To make successful connection with the IBM DB2 we required the user credentials which consist of Database name, Host name, Port Number, SSL certificate which is unique for each user. This SSL certificate is available in downloadable format based on the operating system.

IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

WQM60817.USER

Back

Export to CSV

NAME	EMAIL	PHONE	CITY	INFECT	BLOOD	PASSWORD
Rohini	rdnwork411@gmail.com	45632566666	Nashik	infected	AB Positive	1234567
User1	user123@gmail.com	1234567892	Pune	infected	AB Positive	1234568
User3	user3@gmail.com	23456789	Nashik	infected	A Negative	user3
user4	user4@gmail.com	34567899	Pune	infected	AB Positive	user4

Login Page for user login

Plasma Donor App

Home Register

user4@gmail.com

.....

Login

Request page : Here user can choose blood group and enter the location, after submitting the request it is forwarded to the concerned people

Plasma Donor App

Home Register Request

Choose your blood group

Enter the address

Submit the request

Your request is sent to the concerned people.


```

app.py - PlasmaDonorApplication-main - Visual Studio Code
EXPLORER
  PLASMA DONOR APPLICATION-MAIN
    static
    style1.css
    templates
    login.html
    register.html
    request.html
    stats.html
    style.css
  app.py
  DigiCertGlobalRootCA.crt
  Dockerfile
  README.md
  requirements.txt

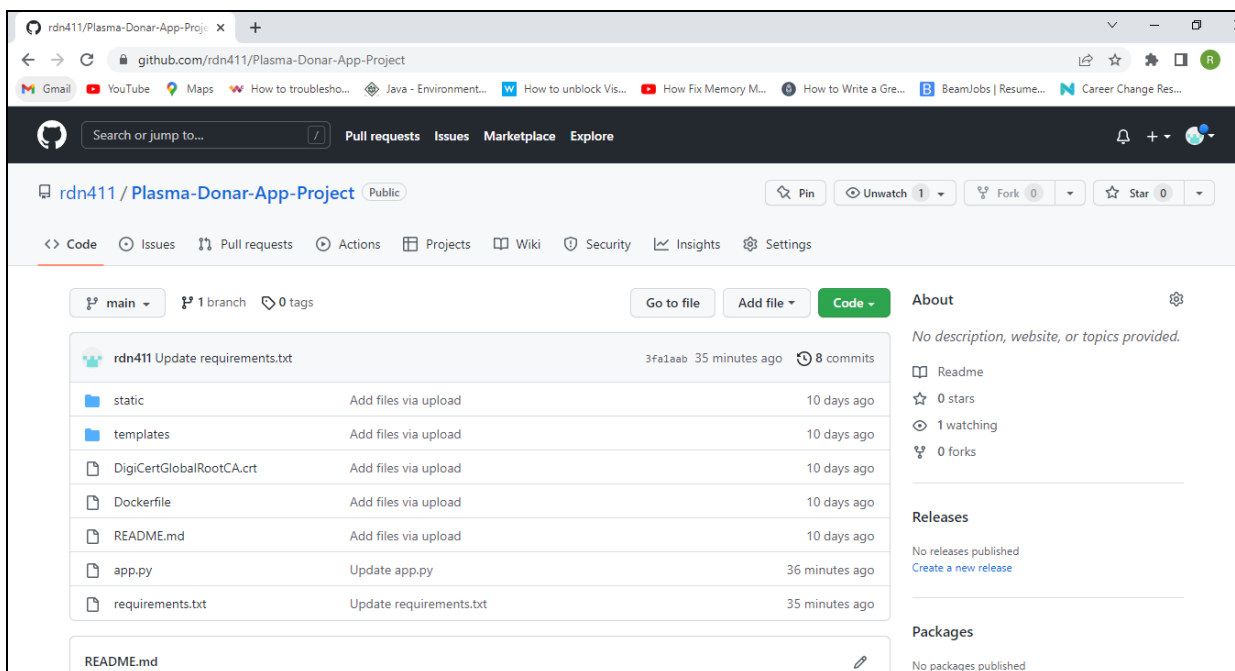
app.py
7 conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=824dfd4d-99de-448d-9991-629c01b3832d.bs21o90108kqblod81cg.databa
8
9 @app.route('/registration')
10 def home():
11     return render_template('register.html')
12
13 @app.route('/register', methods=['POST'])
14 def register():
15     x = [x for x in request.form.values()]
16     print(x)
17     name=x[0]
18     email=x[1]
19     phone=x[2]
20     city=x[3]
21     infect=x[4]
22     blood=x[5]
23     password=x[6]

TERMINAL
192.168.43.146 - - [23/Jul/2022 13:01:45] "GET /static/style1.css HTTP/1.1" 304 -
192.168.43.146 - - [23/Jul/2022 13:01:59] "GET /requester HTTP/1.1" 200 -
192.168.43.146 - - [23/Jul/2022 13:02:00] "GET /static/style1.css HTTP/1.1" 304 -
pune
The Phone is : 4563256666
<Response [401]>
The Phone is : 1234567892
<Response [401]>
The Phone is : 34567899
<Response [401]>
192.168.43.146 - - [23/Jul/2022 13:03:57] "POST /requested HTTP/1.1" 200 -
192.168.43.146 - - [23/Jul/2022 13:03:57] "GET /static/style1.css HTTP/1.1" 304 -
192.168.43.146 - - [23/Jul/2022 15:57:04] "GET /login HTTP/1.1" 200 -
192.168.43.146 - - [23/Jul/2022 15:57:05] "GET /static/style1.css HTTP/1.1" 304 -
Ln 86, Col 17 Spaces: 4 UTF-8 LF Python 3.10.5 64-bit Go Live Prettier

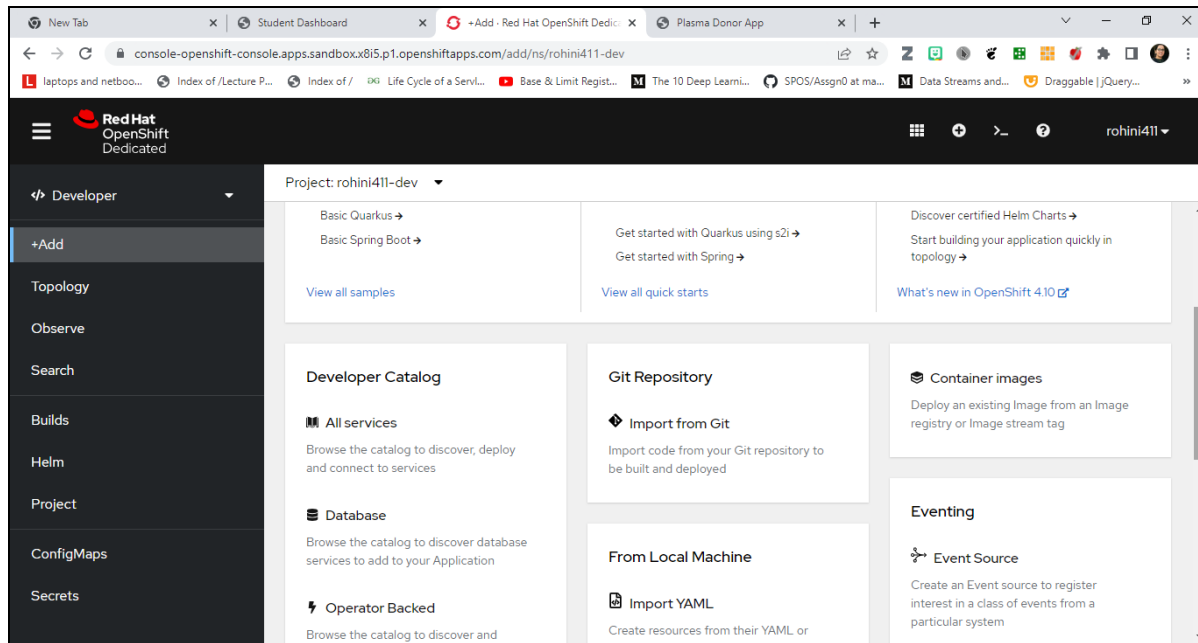
```

The user interface and screenshots of the project deployment on Redhat Open-shift Dedicated dev space are given below

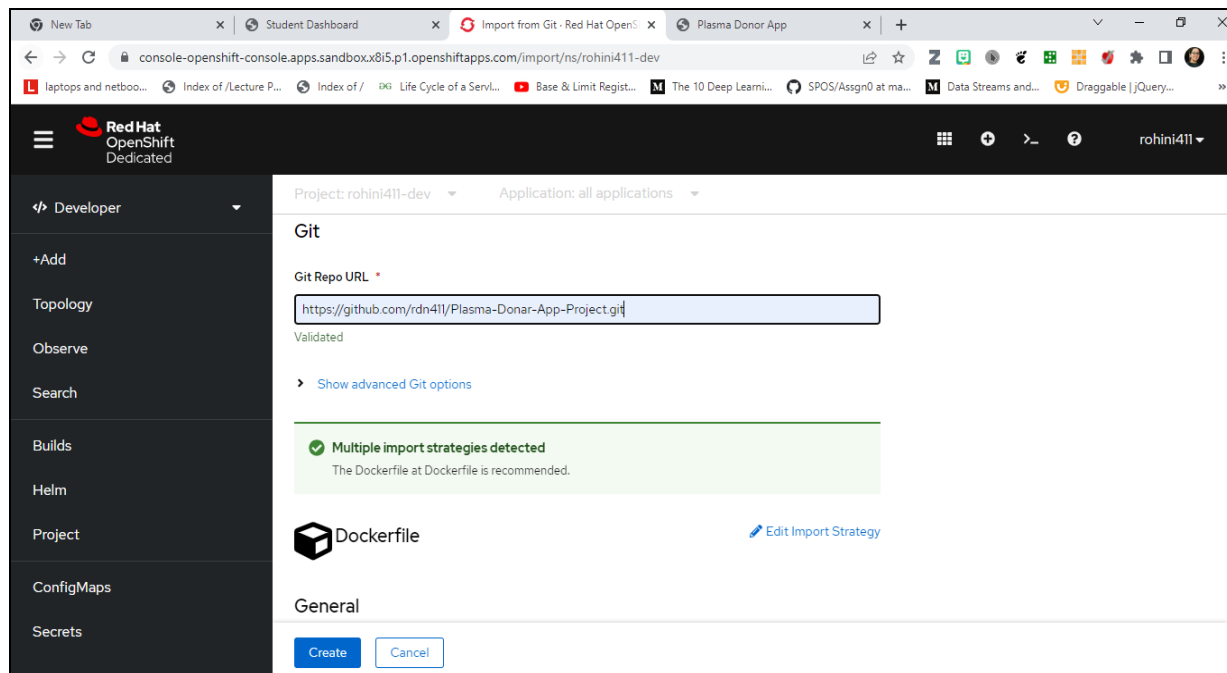
Step-1 To deploy application on Open-shift dedicated dev space, create repository on github and upload all source code files in that repository. This Source code also consist of DigitalCertificate which is downloaded from the IBM DB2 Service.



Step-2 Open and login in Red Hat Open-shift Dedicated console, Go to Developer perspective and click on +Add it shows the following page. Now select Git Repository option from tab

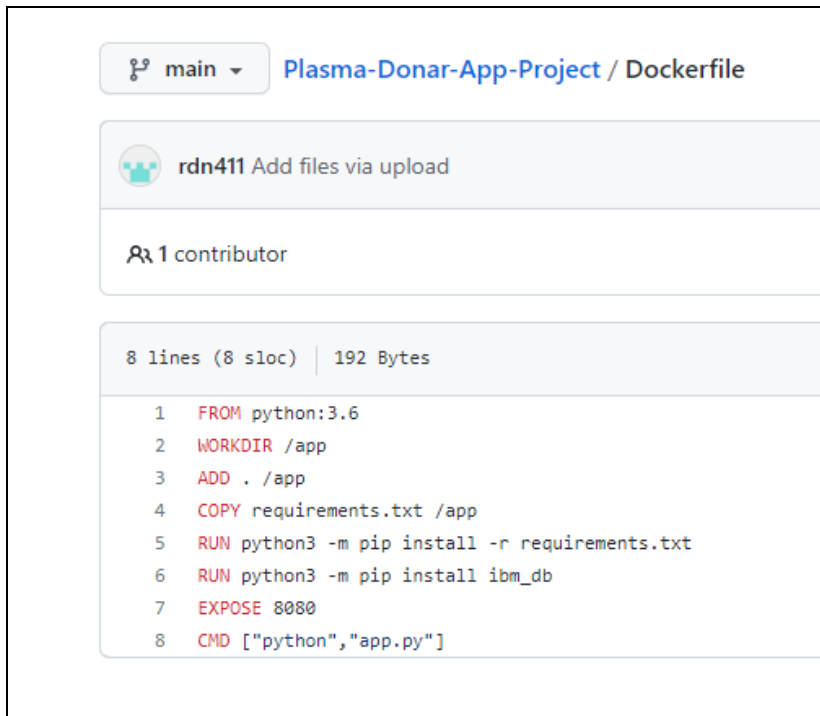


Step-3 Copy and paste your Git repository URL in field and click on **Create**.



Step-4 Now it will be navigated to the Topology tab where we can see our application. It takes some time to deploy the application. During this time build console refers to the Docker file. A Docker file is simply a text file that contains all the commands a user would visit the

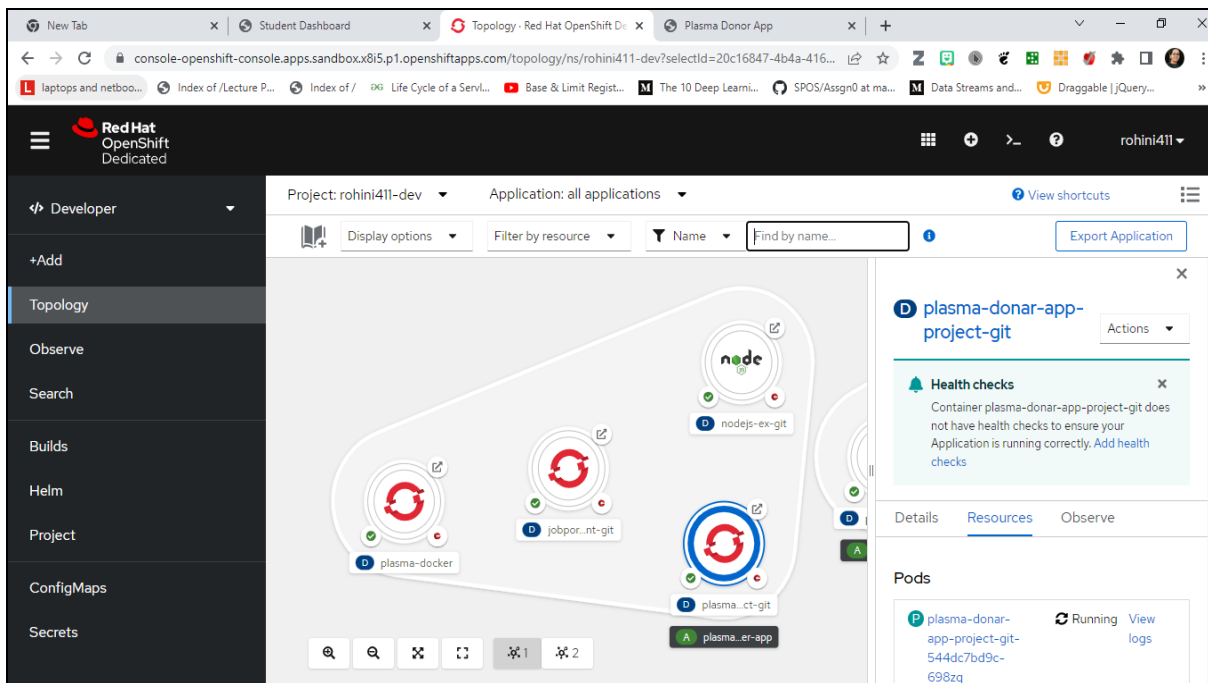
command line to create the image. Docker file must first begin with a FROM instruction, which initializes a new build stage and specifies the base image that subsequent instructions will build upon. This base image is often from a public repository, in this case it is Github.



The screenshot shows a GitHub repository page for 'Plasma-Donar-App-Project / Dockerfile'. It displays the Dockerfile content with 8 lines of code. The code starts with 'FROM python:3.6' and includes instructions for setting the working directory, adding files, copying requirements, installing dependencies, exposing a port, and running the application.

```
1 FROM python:3.6
2 WORKDIR /app
3 ADD . /app
4 COPY requirements.txt /app
5 RUN python3 -m pip install -r requirements.txt
6 RUN python3 -m pip install ibm_db
7 EXPOSE 8080
8 CMD ["python", "app.py"]
```

if everything is fine it shows build is running successfully.

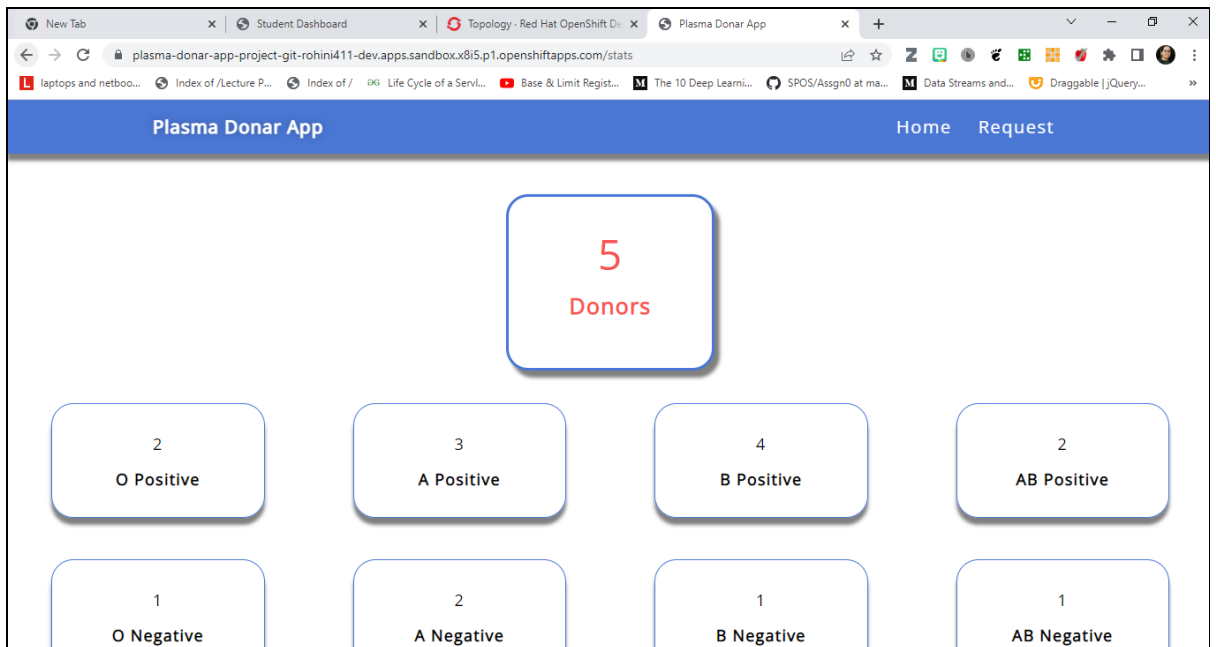


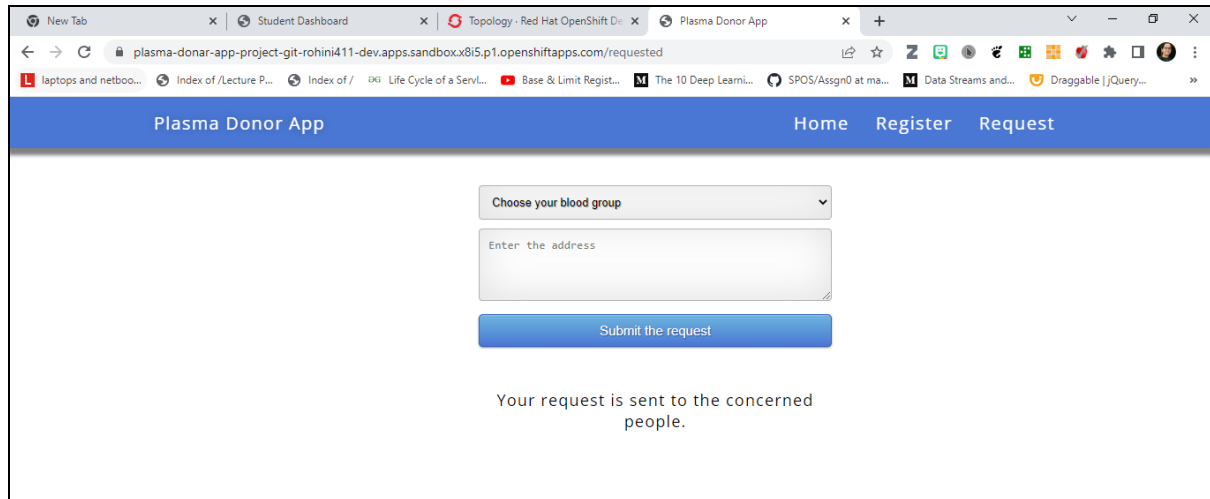
Step-5 Click on the URL of the application, this will open your application in the new tab as shown below. Now we are ready to run the application on Red Hat Open shift dev space, rest

of the procedure is the same as that of local deployment. Here are some screenshots while running our application on Red Hat Open shift Dedicated dev space

The screenshot shows the login page of the Plasma Donor App. The browser's address bar displays the URL: `plasma-donar-app-project-git-rohini411-dev.apps.sandbox.x8i5.p1.openshiftapps.com/login`. The page has a blue header with the app name "Plasma Donor App" and navigation links "Home" and "Register". The main content area contains a login form with two input fields: "Enter UserName" and "Enter Password", followed by a blue "Login" button.

The screenshot shows the registration page of the Plasma Donor App. The browser's address bar displays the URL: `plasma-donar-app-project-git-rohini411-dev.apps.sandbox.x8i5.p1.openshiftapps.com/registration`. The page has a blue header with the app name "Plasma Donor App" and a "Home" link. The main content area contains a registration form with the following fields: "user5", "user5@gmail.com", "4567891223", "Aurangabad", a dropdown menu set to "Infected", another dropdown menu set to "B Positive", and a password field with masked characters "*****". A blue "Register" button is at the bottom.





7. ADVANTAGES & DISADVANTAGES

In this web application we have deployed the app on local-host machine as well as on Red Hat Open shift dedicated.

The **advantages** of the app deploying on cloud platform are given below

- **Flexibility:** Red Hat OpenShift simplifies deployment and management of a hybrid infrastructure, giving you the flexibility to have a self-managed or fully managed service, running on-premise or in cloud and hybrid environments.
- **Container Portability:** Container images built on the OCI industry standard ensure portability between developer workstations and Red Hat OpenShift production environments.
- **Automated installation and updates**
- **Multi-cluster management :** Red Hat Advanced Cluster Management for Kubernetes can easily deploy apps, manage multiple clusters, and enforce policies across clusters at scale.

Disadvantages

- ✍ For Open shift to be perfect, Red Hat has to follow strict life cycle timelines (i.e. as Openstack), rather than postponing a release for a few days, which will make our life easier when we plan the patching
- ✍ The migration between versions in clusters that include numerous operators is sometimes perilous because of the versions' compatibility.

8. APPLICATIONS

This project can be applicable for the Medical field, where a person who are willing to donate the blood to the patients. Through this system it will be easier to find donor for exact blood type and easy to build connection between donor and needy person. To achieve this objective we need to store the person details in a special platform so that we can access it quickly without wasting the time.

9. CONCLUSION

Plasma donor app is developed to manage the blood banks in various locations. This is a Flask application which will take user inputs, update the data in IBM DB2 database and notify the user upon request. In this project we have successfully deployed the app on local as well as cloud platform (Red Hat OpenShift Dedicated) with the help of Docker image.

10. FUTURE SCOPE

We can develop the portable version of the web app using the web application with some extra module which gives detail information about the nearby Blood banks and hospitals.

BIBLIOGRAPHY

https://docs.openshift.com/dedicated/osd_architecture/osd-understanding.html

<https://cloud.ibm.com/docs/Db2onCloud?topic=Db2onCloud-dcw>

<https://www.redhat.com/en/technologies/cloud-computing/openshift/features#:~:text=With%20integrated%20platform%20monitoring%20and,build%20code%20pipelines%20with%20ease.>

<https://www.trustradius.com/products/openshift/reviews?qs=pros-and-cons#reviews>

APPENDIX

Source Code:

Github URL for the code: <https://github.com/rdn411/Plasma-Donar-App-Project.git>

Link for Courses completed during this Externships

Journey To Cloud: Envisioning Your Solution

https://www.credly.com/badges/e7e3f80c-0c0a-4efe-b0e9-53ac07c12261/public_url

App Modernization Basics

https://www.credly.com/badges/01766bb4-554a-45fe-a256-c38883a1f1e6/public_url

Introduction To Containers, Kubernetes, And OpenShift

https://www.credly.com/badges/fce767c1-fd33-4ae5-814c-38d3e2ccde90/public_url

Getting Started With OpenShift and Deploying Microservice With OpenShift

Please find the screenshots for above project on next page

Project Demo Link

<https://drive.google.com/drive/folders/1W7DtUFbGE3KgBd2J2p59RPSotXbw4z7D?usp=sharing>

