

PREPARATION PHASE

Deployment of App in IBM Cloud

Containerize The App

Date	16/11/2022
Team ID	PNT2022TMID26204
Project Name	Personal Expense Tracker Application

1. DOCKER IMAGE CREATION:

STEP 1: To Create a Kubernetes Cluster

The screenshot shows the IBM Cloud Clusters dashboard for a cluster named 'mycluster'. The cluster is in a 'Normal' state and is set to expire in 30 days. The dashboard provides an overview of the cluster's status, including the number of worker nodes (1 of 1), add-on status (0 of 0), master status (Normal), and ingress status (Healthy). Below the overview, the 'Details' section lists the cluster ID, version (1.24.8_1544), infrastructure type (Classic), and zones (Milan 01). It also shows the creation time (17/11/2022, 2:02 pm) and the resource group (Default). An 'Image security enforcement' toggle is set to 'Enable'. A 'Node health' section is visible at the bottom, and a 'Worker node details' link is provided.

STEP 2: Containerize the Flask Application

The screenshot shows a Visual Studio Code editor with a Dockerfile open. The Dockerfile contains the following instructions:

```
1 FROM python:2.7
2 LABEL maintainer="Buvanewari M"
3 RUN apt-get update
4 RUN mkdir /app
5 WORKDIR /app
6 COPY . /app
7 RUN pip install -r requirements.txt
8 EXPOSE 5000
9 ENTRYPOINT [ "python" ]
10 CMD [ "app.py" ]
```

The terminal window at the bottom shows the output of the Docker build process, indicating that the image was successfully built and pushed to the registry.

(The Process will continue in **Upload Image to IBM Container Registry and Deploy in Kubernetes Cluster**)

2. CREATING DOCKER IMAGE FOR FLASK APP

STEP 1: Make a Project folder

STEP 2: Insert the following code into the Dockerfile created earlier

STEP 3: Copy the following into “requirements.txt” file

STEP 4: Test the flask app

STEP 5: Close the server by pressing CTRL + C

STEP 6: Build the Docker image

STEP 7: Run the docker image

STEP 8: Test Again

CODE:

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "welcome to the flask tutorials"

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5001, debug=True)
```

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
EXPOSE 5001
ENTRYPOINT [ "python" ]
CMD [ "demo.py" ]
```

OUTPUT:

