

PREPARATION PHASE

Deployment of App in IBM Cloud

Containerize The App

Date	27 August 2022
Team ID	PNT2022TMID45814
Project Name	Personal Expense Tracker Application

1. DOCKER IMAGE CREATION:

STEP 1: To Create a Kubernetes Cluster

The screenshot shows the IBM Cloud mycluster dashboard. The cluster 'mycluster' is in a 'Normal' state and expires in 30 days. A warning banner indicates that the cluster will be deleted in 30 days and advises backing up data. The dashboard provides a 'Kubernetes dashboard' link and an 'Actions...' menu. The 'Overview' section shows the cluster's status: 1 of 1 nodes, 0 of 0 add-ons, and a 'Normal' master status. The 'Details' section lists the cluster ID (cdqv317f8otoo8svhtdg), version (1.24.8_1544), infrastructure (Classic), and zones (Milan 01). It also shows the creation time (17/11/2022, 2:02 pm) and resource group (Default). The 'Node health' section is visible at the bottom.

STEP 2: Containerize the Flask Application

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

```
1 FROM python:2.7
2 LABEL maintainer="Buvaneswari 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 output shows the following logs:

```
2022-11-17 15:14:08.021 [info] updateSetState idle
2022-11-17 15:14:14.568 [info] Starting extension host with pid 5324 (fork() took 76 ms).
2022-11-17 15:14:38.032 [info] updateSetState checking for updates
2022-11-17 15:14:38.207 [info] updateSetState idle
```

(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:

