

PREPARATION PHASE

Deployment of App in IBM Cloud

Containerize The App

Date	19/11/2022
Team ID	PNT2022TMID26133
Project Name	Nutrition Assistant Application

1. DOCKER IMAGE CREATION:

STEP 1: To Create a Kubernetes Cluster

The screenshot shows the IBM Cloud console interface for a Kubernetes cluster named 'mycluster'. The cluster is in a 'Normal' state and is set to expire in 30 days. The overview section displays the following details:

- Node status:** 1 of 1, Normal
- Add-on status:** 0 of 0, Normal
- Master status:** Normal
- Ingress status:** Healthy

The details section provides further information:

- Cluster ID:** cdqv317I9otoo8svhtdg
- Version:** 1.24.8_1544
- Infrastructure:** Classic
- Zones:** Milan 01
- Created:** 17/11/2022, 2:02 pm
- Resource group:** Default
- Image security enforcement:** Enable

The bottom section shows 'Node health' with a link to 'Worker node details'.

STEP 2: Containerize the Flask Application

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

```
1 FROM python:2.7
2 LABEL maintainer="Buvaneshwari 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] update:setState idle
2022-11-17 15:14:14.568 [info] Starting extension host with pid 5324 (fork() took 76 ms).
2022-11-17 15:14:38.022 [info] update:setState checking for updates
2022-11-17 15:14:38.207 [info] update:setState 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:

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

OUTPUT:

