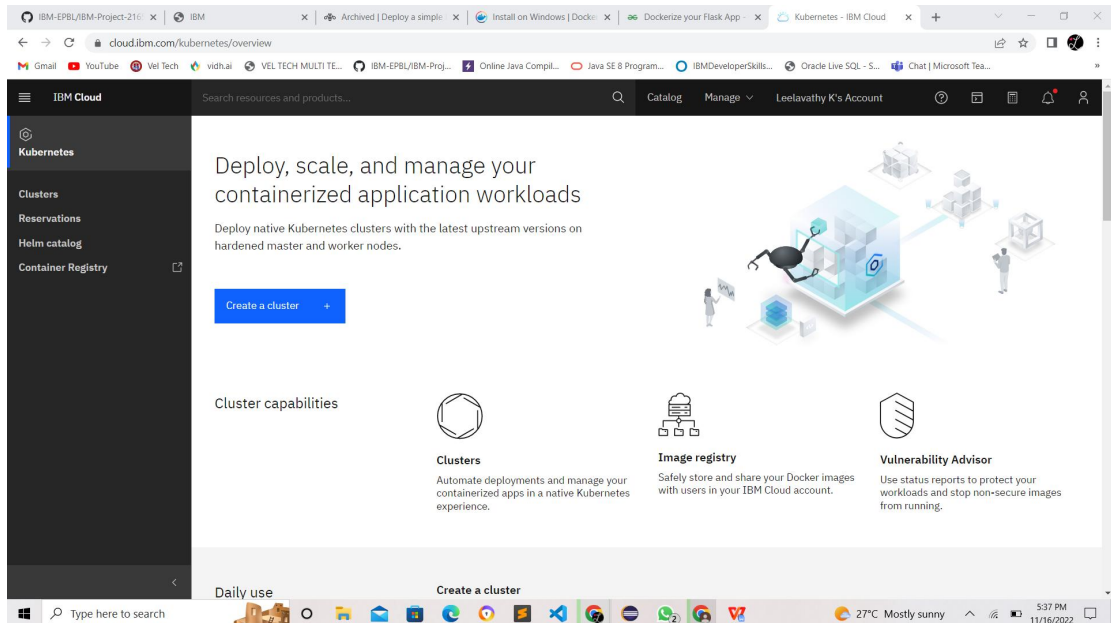


CONTAINERIZE THE APP

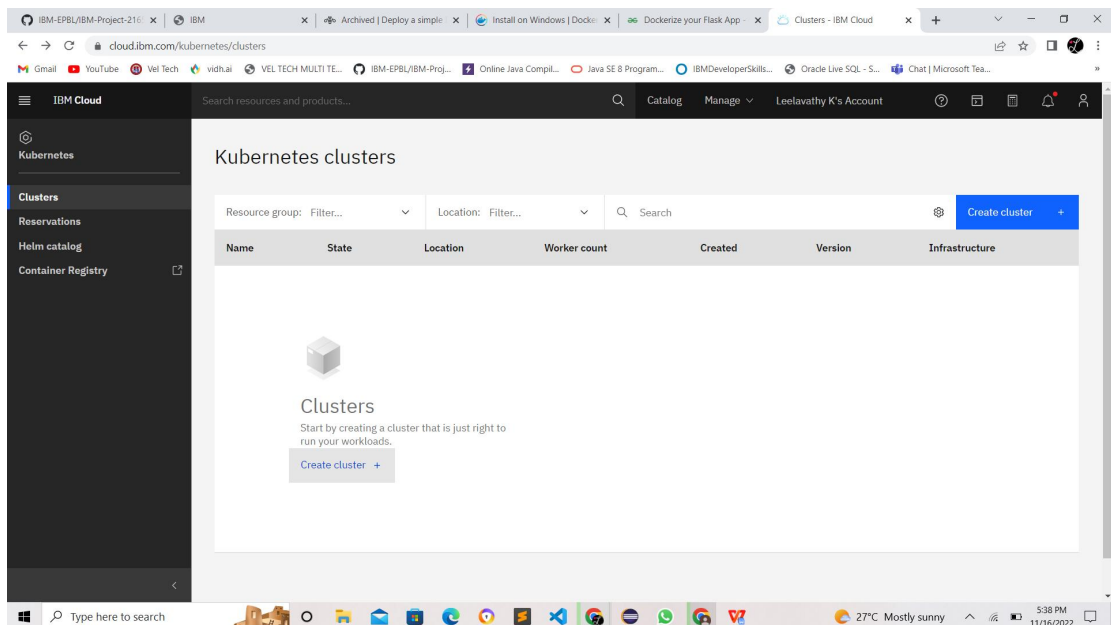
TEAM ID : PNT2022TMID22368

STEP 1: Sign in to your [IBM Cloud Dashboard](#)

STEP 2: Open IBM Kubernetes Service



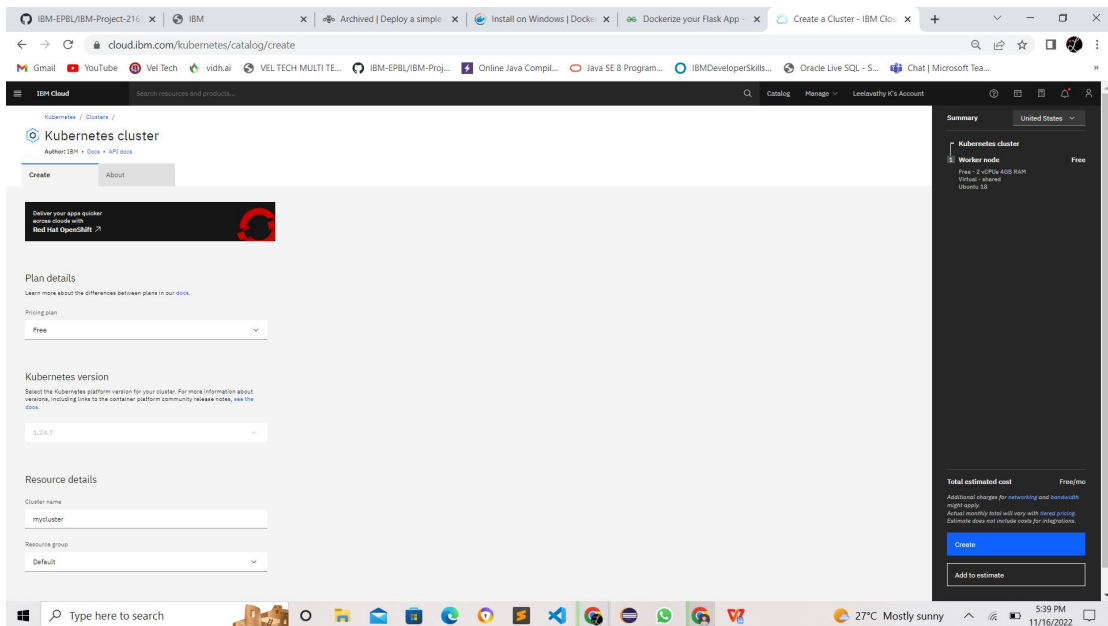
STEP 3: Click Create Cluster



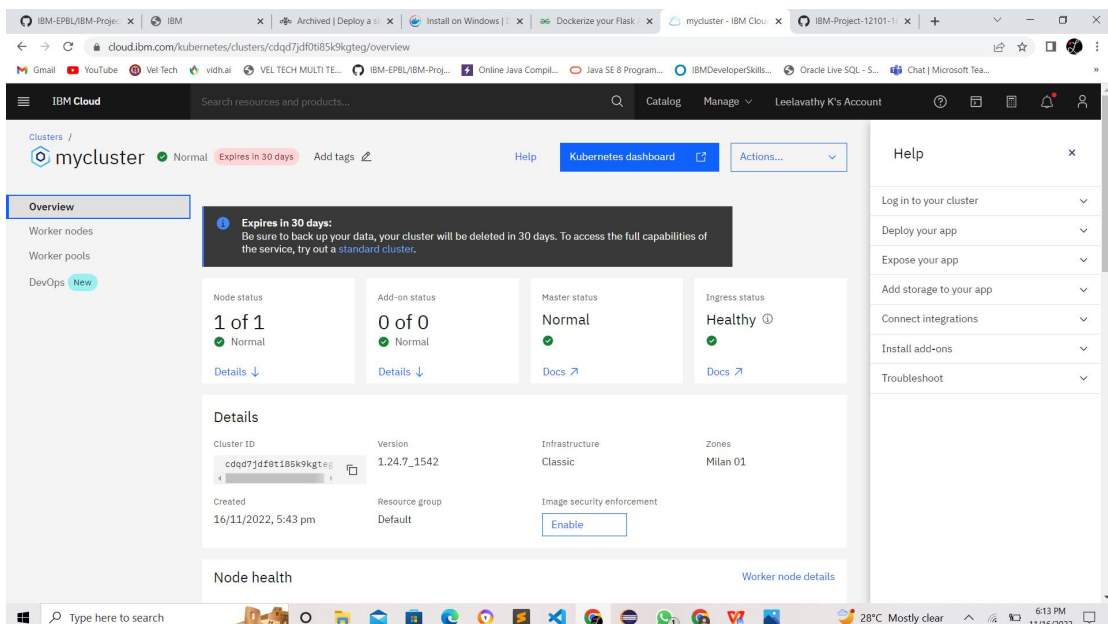
STEP 4: Select the Region where you want to deploy the cluster, type in a name for your cluster, then click Create Cluster.

STEP 5: Select the appropriate cluster type depending on your account.

STEP 6: It takes some time for the cluster to get ready (around 30 minutes).



STEP 7: Once the cluster is ready, click on your cluster's name and you will be redirected to a new page with information about your cluster and worker node.



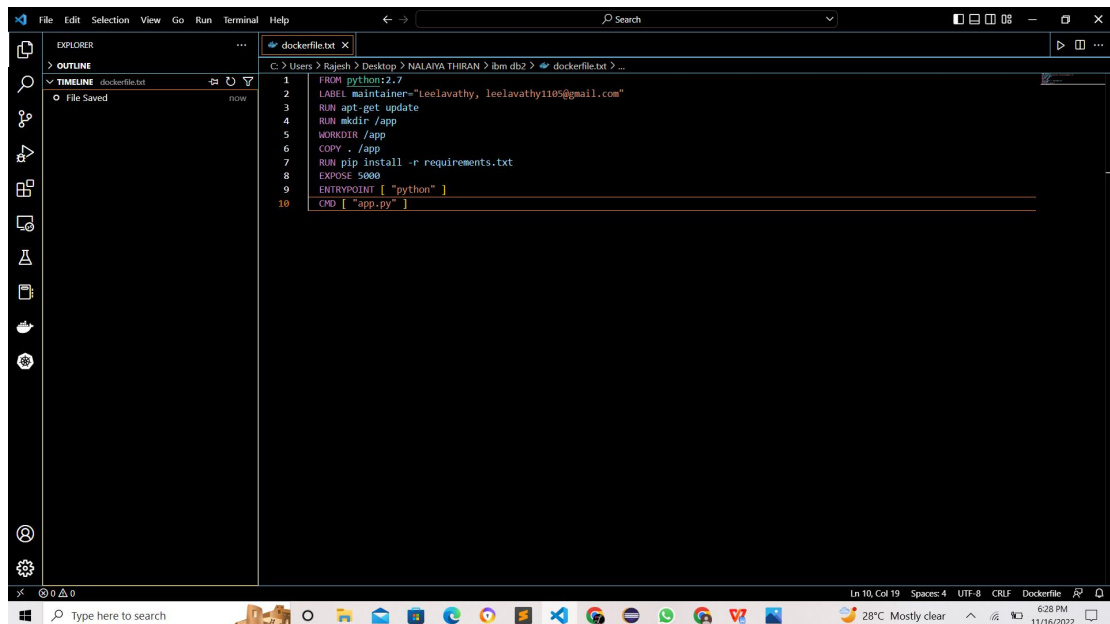
STEP 8: Click on the **Worker Nodes** tab to note the cluster's Public IP.

The screenshot shows the IBM Cloud console interface. The 'Worker nodes' tab is selected, displaying a table of worker nodes. The table has columns for Name, Status, Worker pool, Zone, Private IP, Public IP, and Version. A single node is listed with the name '000000cf', status 'Normal', worker pool 'default', zone 'Milan 01', private IP '10.144.180.104', public IP '169.51.195.208', and version '1.24.7_1543'. The 'Public IP' column is highlighted, indicating the cluster's Public IP.

Name	Status	Worker pool	Zone	Private IP	Public IP	Version
000000cf	Normal	default	Milan 01	10.144.180.104	169.51.195.208	1.24.7_1543

CONTAINERIZE YOUR FLASK APPLICATION

The screenshot shows a code editor interface with a file explorer on the left. The file explorer is titled 'EXPLORER' and shows a project structure. The 'web' directory is expanded, showing files like 'static', 'templates', 'venv', '.dockerignore', '.gitignore', 'app.py', 'deployment.yaml', 'Dockerfile', 'requirements.txt', 'service.yaml', and 'README.md'. The 'Dockerfile' file is highlighted with a red box, indicating it is the focus of the containerization process.



```
1 FROM python:2.7
2 LABEL maintainer="leelavathy, leelavathy1105@gmail.com"
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" ]
```

BUILD AN IMAGE FROM THE DOCKERFILE

Open the terminal and type this command to build an image from your Dockerfile: `docker build -t <image_name>:<tag> .` (note the period to indicate we're in our apps top level directory). For example: `docker build -t app:latest .`

After you build your image successfully, type: `docker run -d -p 5000:5000 app`

This command will create a container that contains all the application code and dependencies from the image and runs it locally.

