

1.What is Docker?

Docker is a tool, it is having single container where we store our code in the form of images in docker container repository. Docker is an open-source platform to build, deploy, run and manage containers.

2.How to Install Docker and Why do we need Docker?

Prerequisite: Docker account, Docker Desktop Application, Pycharm,

Step 1: Create a python file main.py

Step 2: Import Flask in main.py

Step 3: Create a Docker File with no extension.

Step 4: Navigate to cmd from docker directory

Step 5: Create a build command.

```
`docker build . --tag tagname`
```

Step 6: Run the build command

```
`docker run -d -p 5000:5000 python-flask-docker`
```

Step 7: check all the images created by **Docker images**

Step 8: Run Python3 -m flask run -- host =0.0.0.0

Step 9: Check with Docker ip address.

Python.py:

```
from flask import Flask, redirect, url_for, request
```

```
app = Flask(__name__)
```

```
@app.route('/success/name')
```

```
def success(name):
```

```
    return 'welcome %s' % name
```

```
@app.route('/login', methods=['POST', 'GET'])
```

```
def login():
```

```
    if request.method == 'POST':
```

```
        user = request.form['nm']
```

```
        return redirect(url_for('success', name=user))
```

```
    else:
```

```
        user = request.args.get('nm')
```

```
        return redirect(url_for('success', name=user))
```

```
if __name__ == '__main__':
```

```
    app.run(debug=True)
```

Docker provides tooling and a platform to manage the lifecycle of your containers:

- Develop your application and its supporting components using containers.
- The container becomes the unit for distributing and testing your application.

- When you're ready, deploy your application into your production environment, as a container or an orchestrated service. This works the same whether your production environment is a local data center, a cloud provider, or a hybrid of the two.

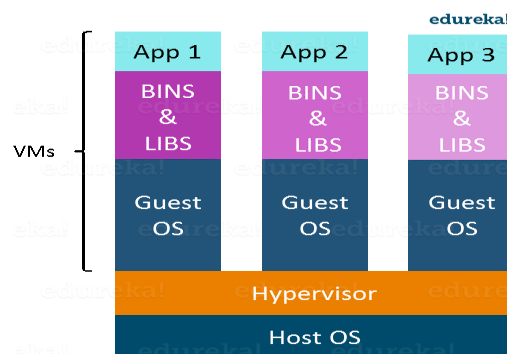
3. What is Virtualization?

Before containerization came into the picture the leading way to isolate, organize applications and their dependencies was to place each and every application in its own virtual machine. These machines run multiple applications on the same physical hardware, and this process is called **Virtualization**.

In own words we can also say:

Virtualization is the technique of importing a Guest operating system on top of a Host operating system. This technique was a revelation at the beginning because it allowed developers to run multiple operating systems in different virtual machines all running on the same host. This eliminated the need for extra hardware resource. The advantages of Virtual Machines or Virtualization are:

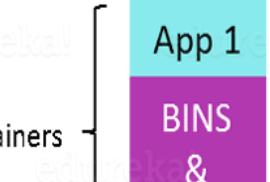
- Multiple operating systems can run on the same machine
- Maintenance and Recovery were easy in case of failure conditions
- Total cost of ownership was also less due to the reduced need for infrastructure.



4. What is Containerization?

Containerization is the technique of bringing virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the

Advantages of Containerization over Virtualization:

- 
- The diagram illustrates the container architecture stack. At the base is the **Host OS** (dark blue). Above it is the **Container Engine** (orange). On top of the Container Engine are three containers, each represented by a vertical stack of two colored boxes: a light blue box for the application and a purple box for binaries and libraries. The containers are labeled **App 1**, **App 2**, and **App 3**. A bracket on the left side of the containers is labeled **Containers**.
- | Layer | Component |
|------------------------|---------------------------------------|
| Host OS | Host OS |
| Container Engine | Container Engine |
| Containers | App 1, App 2, App 3 |
| Application Layer | App 1, App 2, App 3 |
| Binary & Library Layer | BINS & LIBS, BINS & LIBS, BINS & LIBS |

Step 1: Build the Docker file using the below command.

“.” is used to build the Docker file in the present folder

1	docker images
---	---------------

Step 3: Now to create a container based on this image, you have to run the following command:

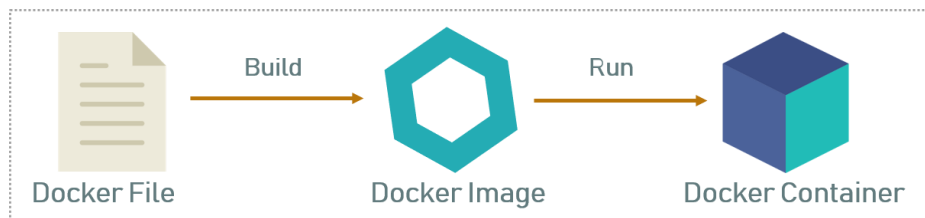
```
1 id
```

Where *-it* is to make sure the container is interactive, *-p* is for port forwarding, and *-d* to run the daemon in the background.

Step 4: Now you can check the created container by using the following command:

```
1 docker ps
```

6. Docker image ,command for build and execute the docker image?



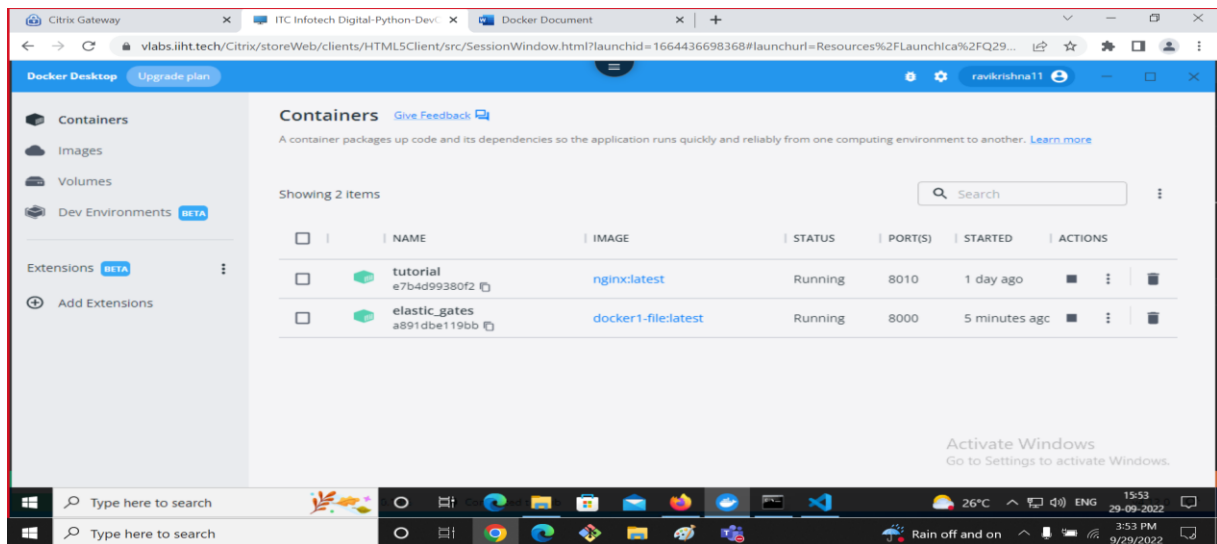
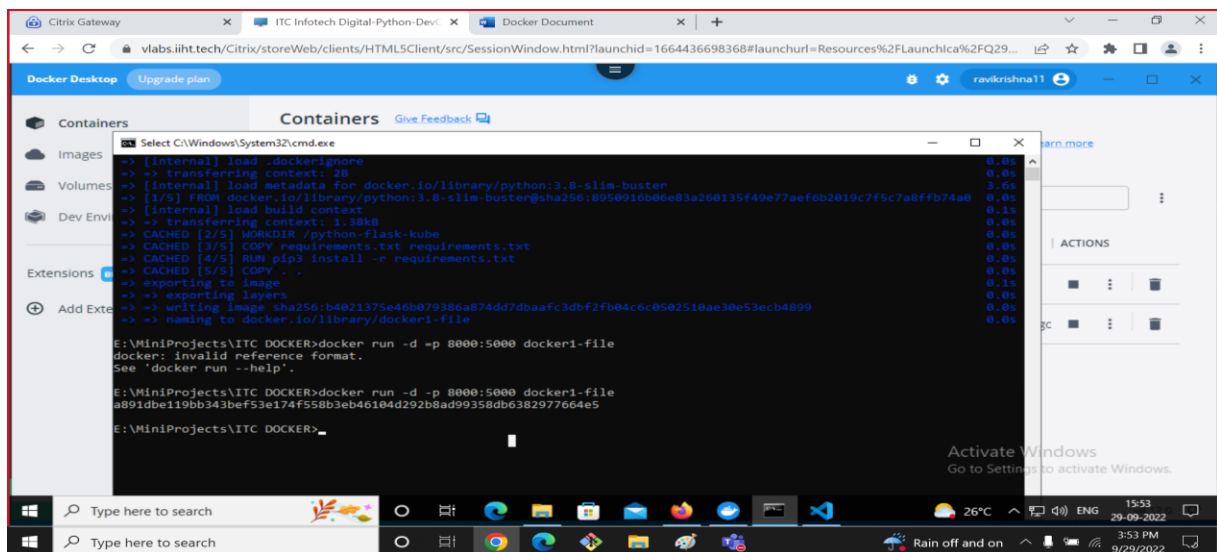
In layman terms, Docker Image can be compared to a template which is used to create Docker Containers. So, these read-only templates are the building blocks of a Container. You can use `docker run` to run the image and create a container.

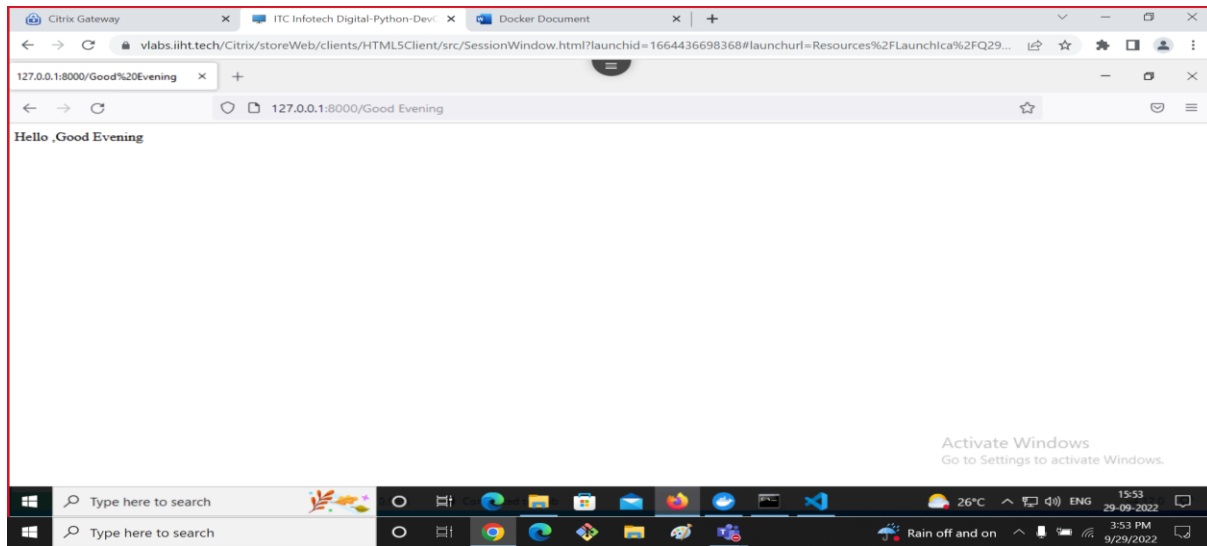
Docker Images are stored in the Docker Registry. It can be either a user's local repository or a public repository like a Docker Hub which allows multiple users to collaborate in building an application.

Build: ``docker build . --tag tagname``

Execute: ``docker run -d -p 5000:5000 python-flask-docker``

Outputs:







kubernetes

1. What is Kubernetes?

Kubernetes is an open-source container management (orchestration) tool. Its container management responsibilities include container deployment, scaling & descaling of containers & container load balancing.

2. Why to use Kubernetes?

We use Kubernetes to containerize the application on a massive scale. If there is a high demand then we can scale the container and if the demand is less we can descale the container.

One of the benefits of Kubernetes is that it makes building and running complex applications much simpler. Here's a handful of the many Kubernetes features:

- Standard services like local DNS and basic load-balancing that most applications need, and are easy to use.
- Standard behaviors (e.g., restart this container if it dies) that are easy to invoke, and do most of the work of keeping applications running, available, and performant.
- A standard set of abstract “objects” (called things like “pods,” “replicasets,” and “deployments”) that wrap around containers and make it easy to build configurations around collections of containers.

1. Automatic Binpacking

Kubernetes automatically packages your application and schedules the containers based on their requirements and available resources while not sacrificing availability. To ensure complete utilization and save unused resources, Kubernetes balances between critical and best-effort workloads.

2. Service Discovery & Load balancing

With Kubernetes, there is no need to worry about networking and communication because Kubernetes will automatically assign IP addresses to containers and a single DNS name for a set of containers, that can load-balance traffic inside the cluster.

3. Storage Orchestration

With Kubernetes, you can mount the storage system of your choice. You can either opt for local storage, or choose a public cloud provider such as GCP or AWS, or perhaps use a shared network storage system such as NFS, iSCSI, etc.

4. Self-Healing

Personally, this is my favorite feature. Kubernetes can automatically restart containers that fail during execution and kills those containers that don't respond to user-defined health checks. But if nodes itself die, then it replaces and reschedules those failed containers on other available nodes.

5. Secret & Configuration Management

Kubernetes can help you deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.

6. Batch Execution

In addition to managing services, Kubernetes can also manage your batch and CI workloads, thus replacing containers that fail, if desired.

7. Horizontal Scaling

Kubernetes needs only 1 command to scale up the containers, or to scale them down when using the CLI. Else, scaling can also be done via the Dashboard (kubernetes UI).

8. Automatic Rollbacks & Rollouts

Kubernetes progressively rolls out changes and updates to your application or its configuration, by ensuring that not all instances are worked at the same instance. Even if something goes wrong, Kubernetes will rollback the change for you.

These were some of the notable features of Kubernetes. Let me delve into the attractive aspects of Kubernetes with a real-life implementation of it and how it solved a major industry worry.

3.How to use Kubernetes?

Step1: Go to powershell run command get-executionpolicy

Step2: If it is Restricted then copy paste

```
Set-ExecutionPolicy Bypass -Scope Process -Force;  
[System.Net.ServicePointManager]::SecurityProtocol =  
[System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object
```

```
System.Net.WebClient).DownloadString('https://community.chocolatey.org/install.ps1'))
```

Step3: Install

```
choco install minikube  
minikube start
```

Step4: Version check

```
minikube --version
```

4.Differences Between Kubernetes and Docker?

Kubernetes	Docker
Kubernetes is an open-source platform used for maintaining and deploying a group of containers	Docker is a tool that is used to automate the deployment of applications in lightweight containers so that applications can work efficiently in different environments.
In practice, Kubernetes is most commonly used alongside Docker for better control and implantation of containerized applications.	With Docker, multiple containers run on the same hardware much more efficiently than the VM environment & productivity of Docker is extremely high.
Applications are deployed as a combination of pods, Deployment, and services.	Apps are deployed in the form of services.
It supports auto-scaling of the container in a cluster.	Docker does not support auto-scaling.
The health check is of two kinds: liveness and readiness.	Health checks are limited to service.
Hard to set up and configure.	Docker's setup and installation are easy.
It does not have extensive documentation but quite less than Docker. But it does include everything from installation to deployment.	Docker documentation is more effective, more extensive, and even more capabilities & it includes everything from installation to deployment & quick-start instructions as well as the more detailed tutorial.
Kubernetes installation is provided to be quite difficult than Docker and even the command for Kubernetes is quite more complex than Docker.	Docker installation is quite easier, by using fewer commands you can install Docker in your virtual machine or even on cloud.

Azure, buffer, intel, Evernote, Shopify Using Kubernetes.	Google, Amazon, ADP, VISA, citizens bank, MetLife companies using Docker
--	---

5.Installing and Configuring Kubernetes in the Devices?

1. `docker build . --tag python-flask-kube`
2. `minikube image load python-flask-kube:latest`
3. `Minikube tunnel`
4. `kubectl create --filename deployment.yaml`
5. `kubectl create --filename service.yaml`
6. `Kubectl get all`
7. `Kubectl get services`

6. Loading the Docker Image into the Kubernetes?

`minikube image load python-flask-kube:latest`

7. Working with python programs in Kubernetes?

Step1: Create `name.py` file

Step2: Create `requirements.txt` file

Step3: Create `deployment.yaml` and `service.yaml` files.

Step4: Run and Goto the external ip with port that is shown in the terminal.