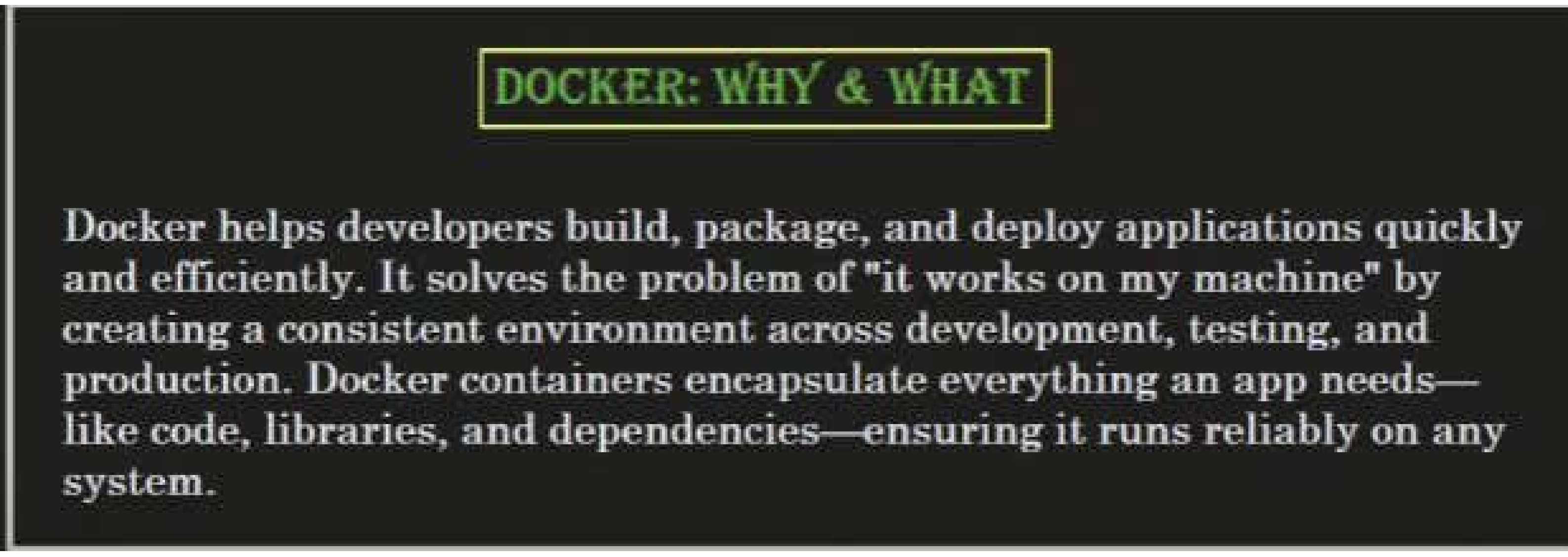
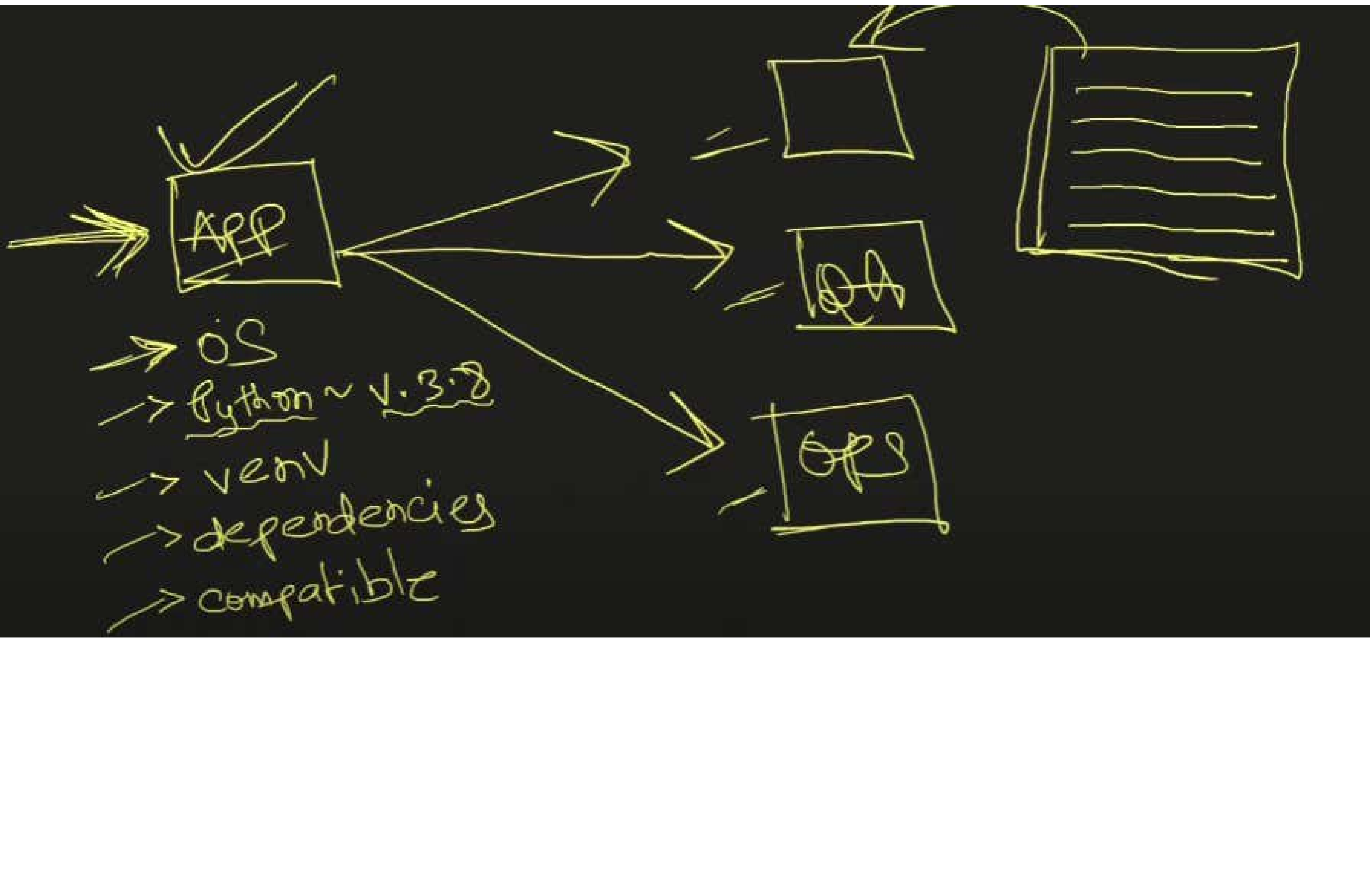
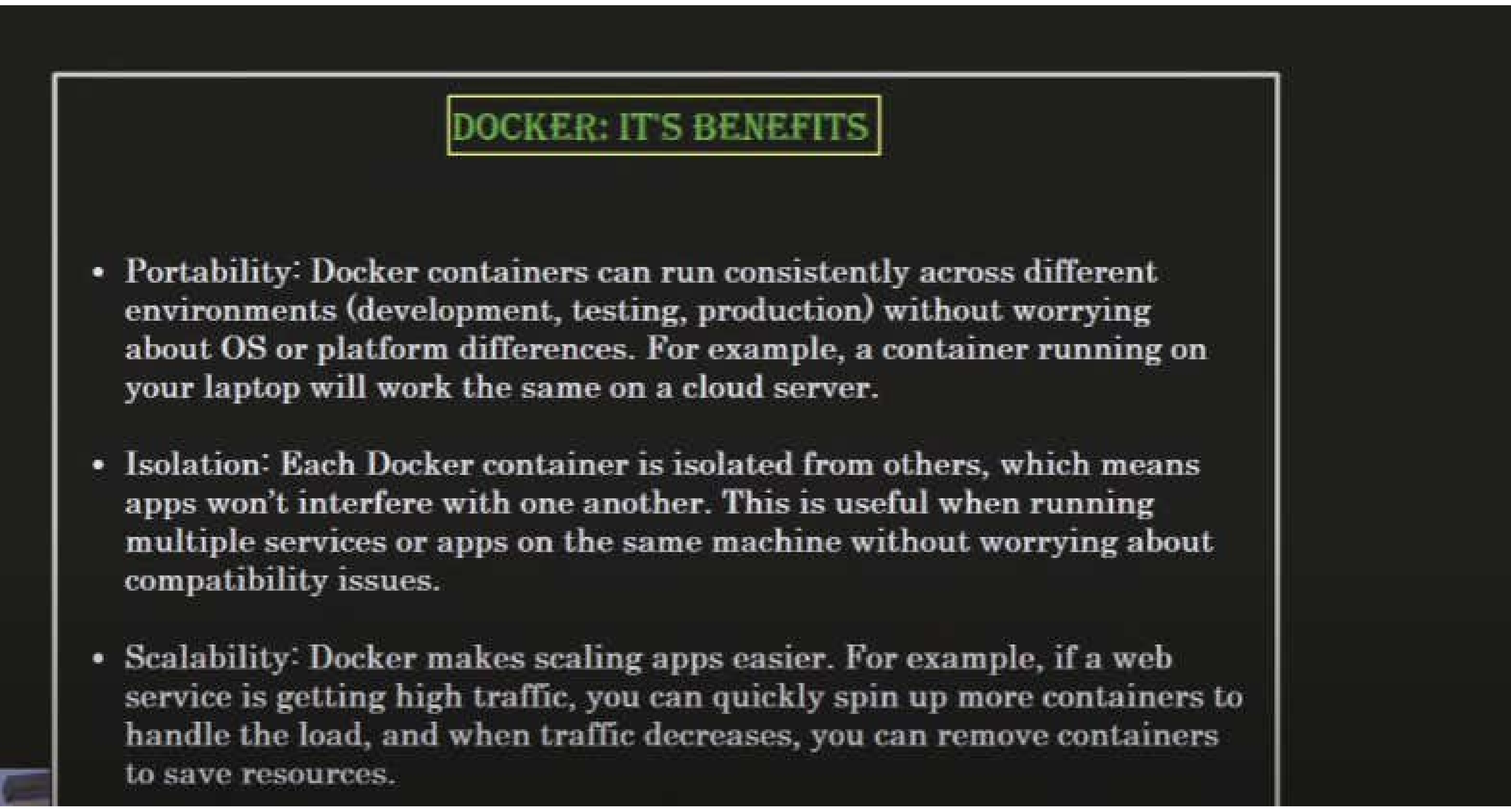


DOCKER: WHY & WHAT

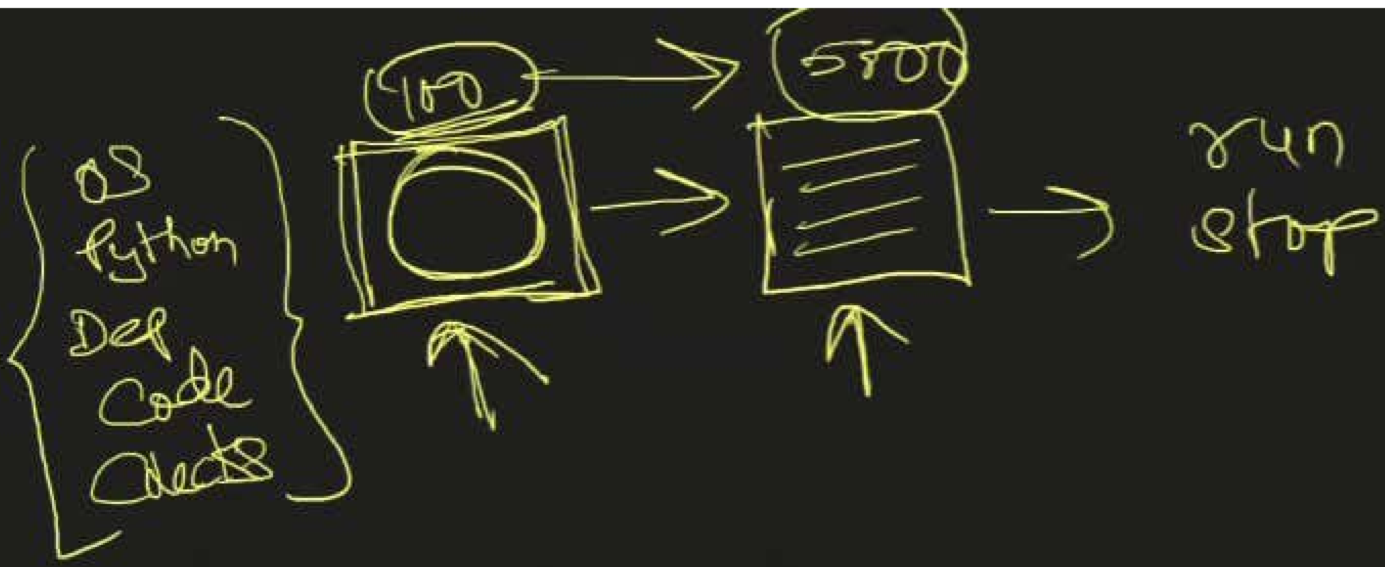
Docker helps developers build, package, and deploy applications quickly and efficiently. It solves the problem of "it works on my machine" by creating a consistent environment across development, testing, and production. Docker containers encapsulate everything an app needs—like code, libraries, and dependencies—ensuring it runs reliably on any system.

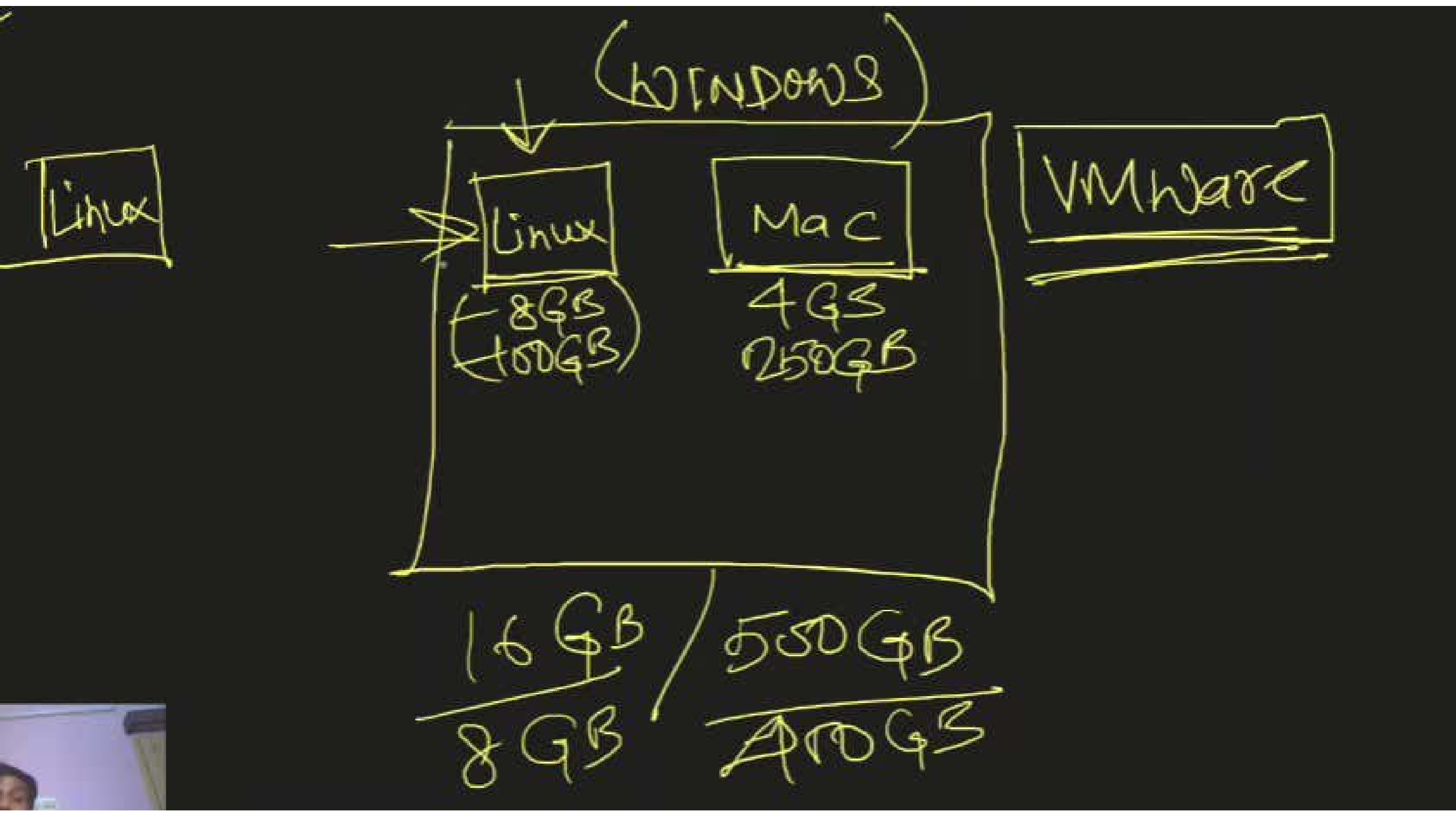

APP
OS
Python ~ V.3.8
venv
dependencies
compatible
QA
OPS



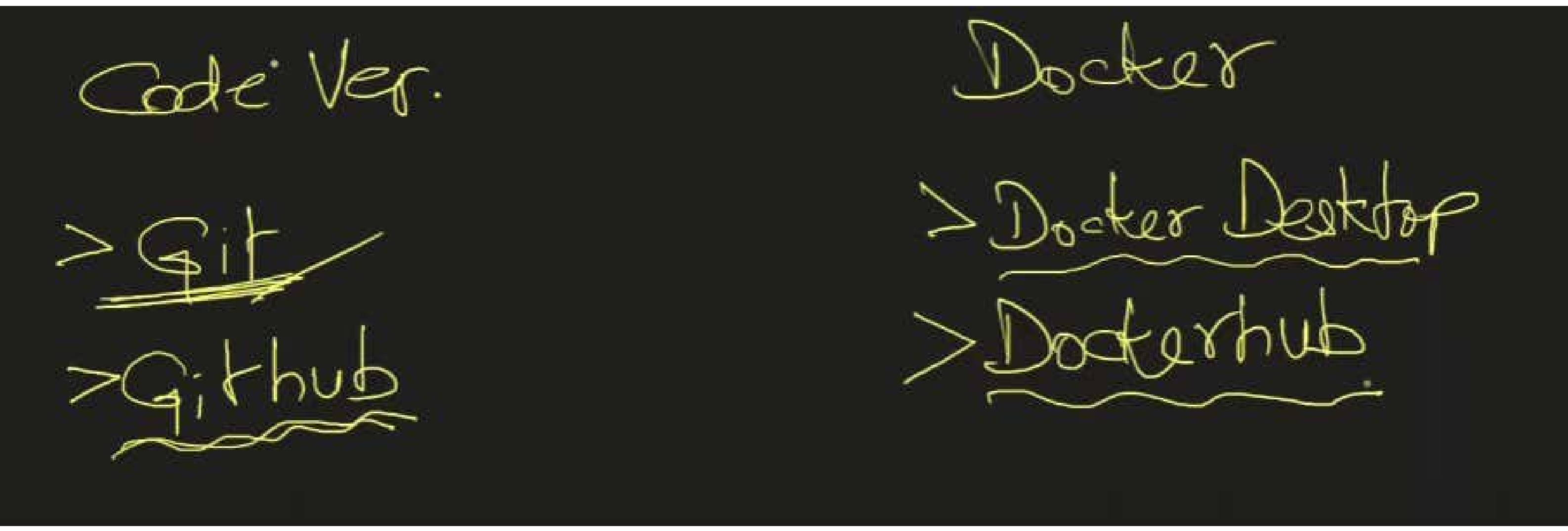
DOCKER: IT'S BENEFITS

- Portability: Docker containers can run consistently across different environments (development, testing, production) without worrying about OS or platform differences. For example, a container running on your laptop will work the same on a cloud server.
- Isolation: Each Docker container is isolated from others, which means apps won't interfere with one another. This is useful when running multiple services or apps on the same machine without worrying about compatibility issues.
- Scalability: Docker makes scaling apps easier. For example, if a web service is getting high traffic, you can quickly spin up more containers to handle the load, and when traffic decreases, you can remove containers to save resources.


OS
Python
Dep
Code
Checks
400
5800
run
stop



(WINDOWS)
Linux
Linux
Mac
VMware
8GB
400GB
4GB
250GB
16GB
8GB
550GB
400GB



Code Ver.

> Git

> Github

Docker

> Docker Desktop

> Dockerhub

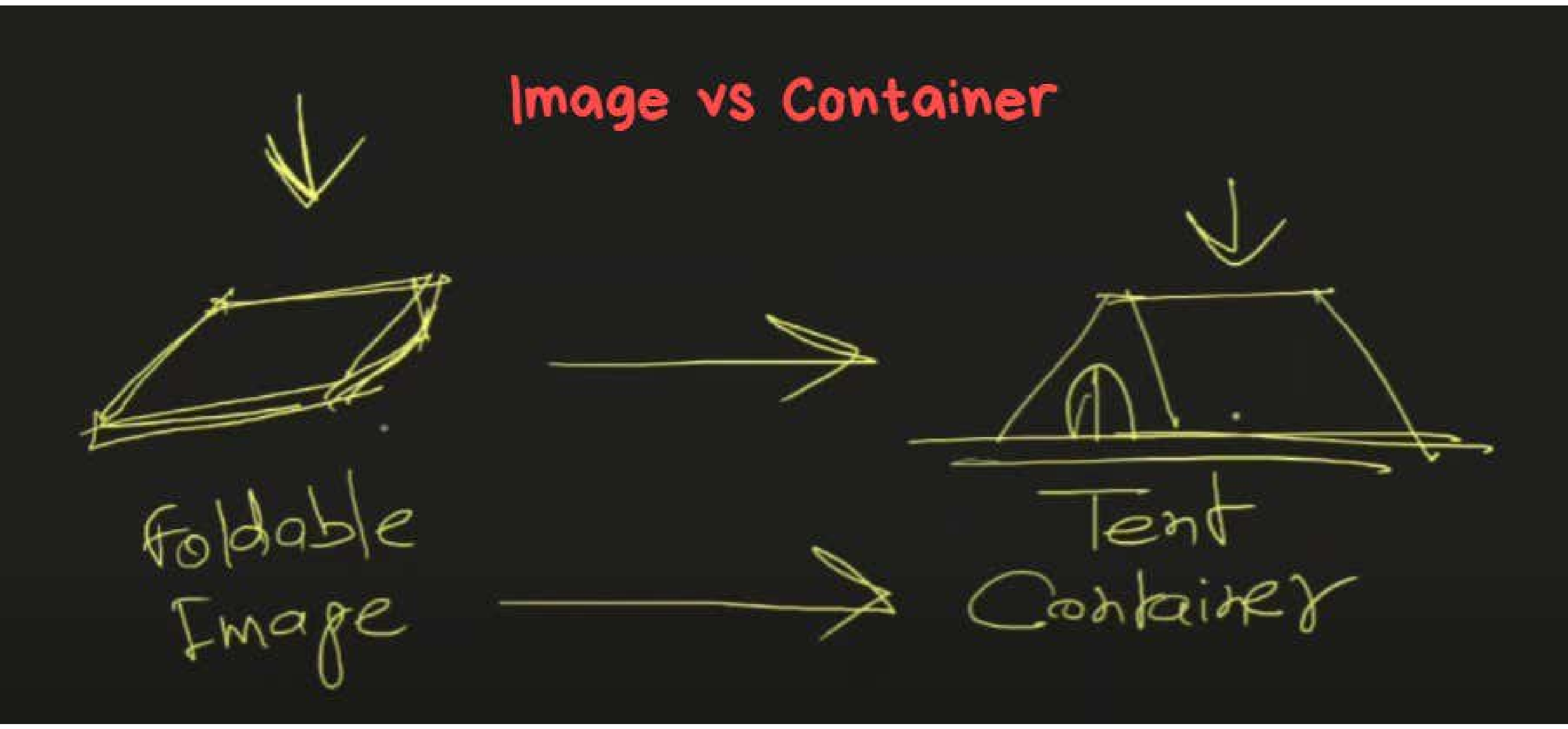
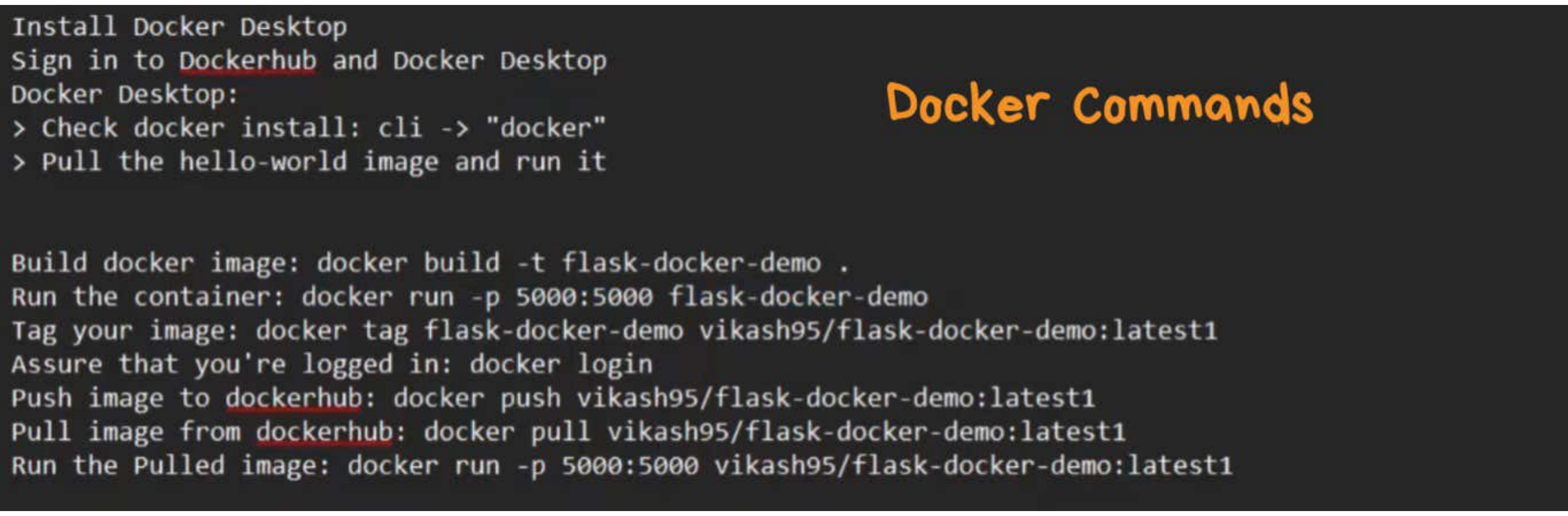


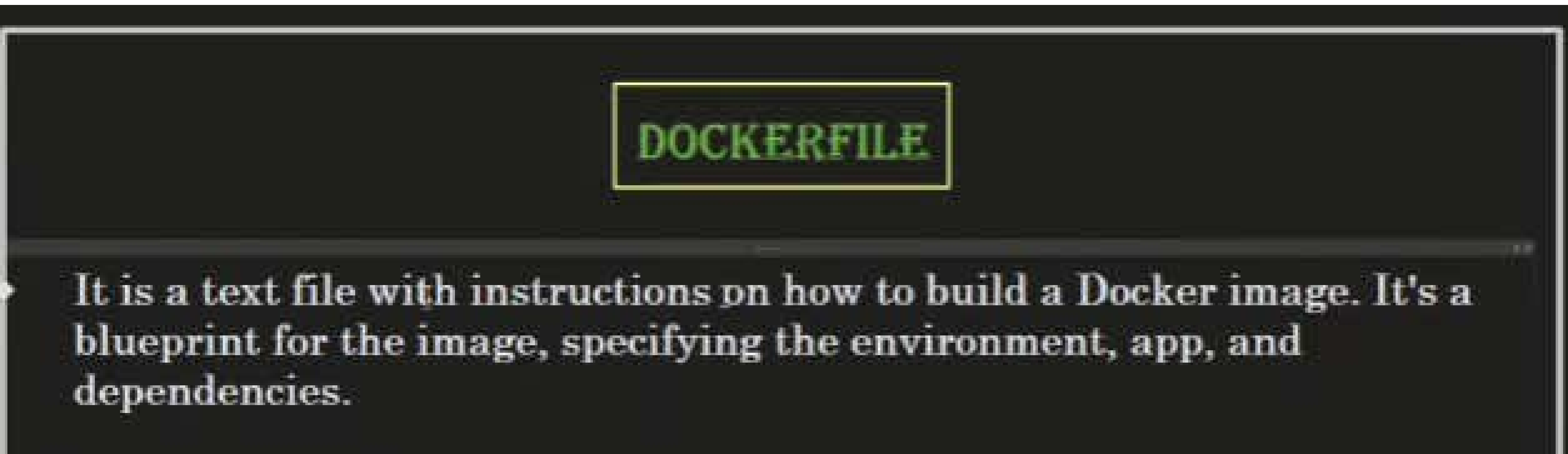
Image vs Container
Foldable Image
Tent Container



Docker Commands

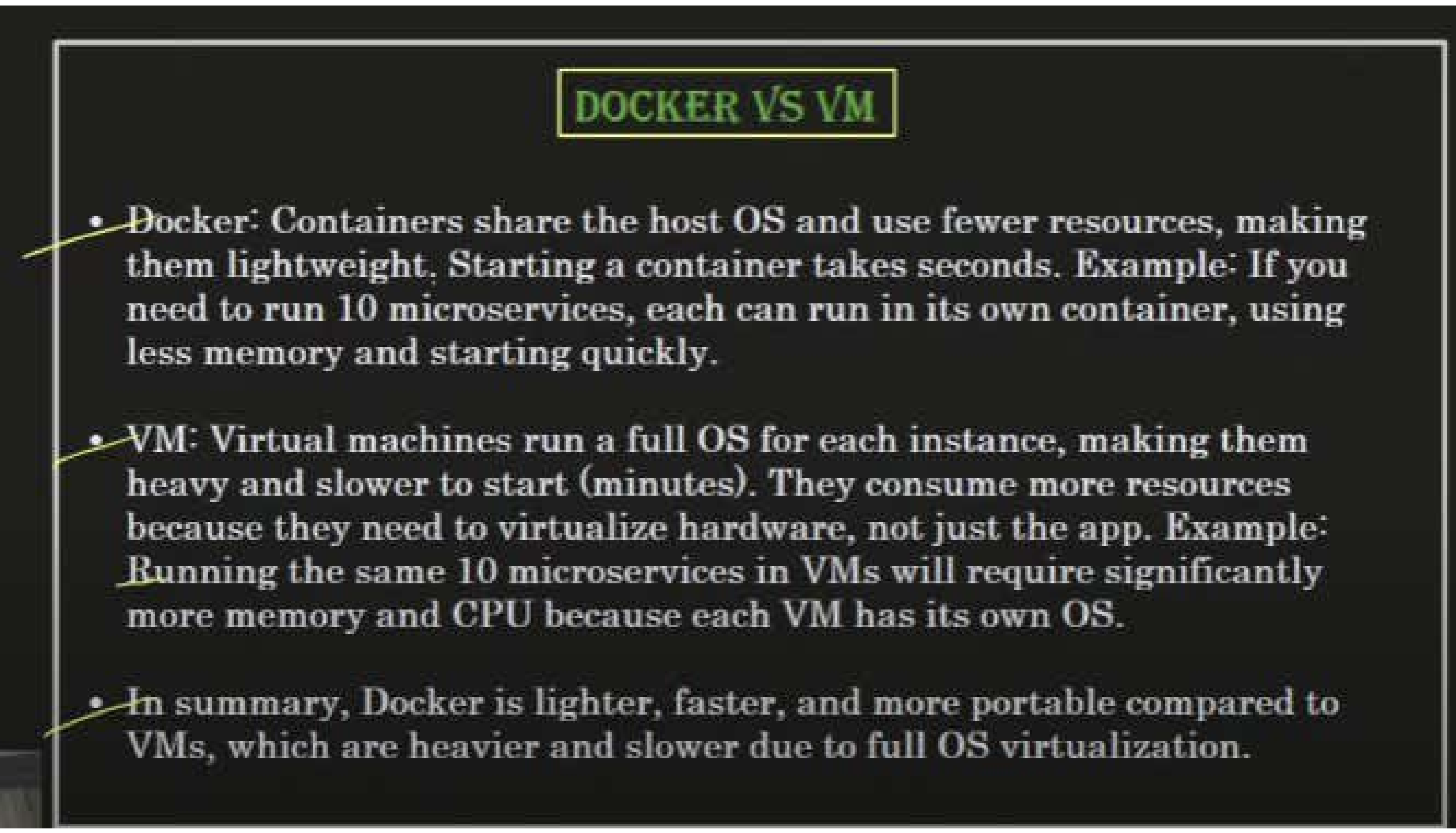
```
Install Docker Desktop
Sign in to Dockerhub and Docker Desktop
Docker Desktop:
> Check docker install: cli -> "docker"
> Pull the hello-world image and run it


Build docker image: docker build -t flask-docker-demo .
Run the container: docker run -p 5000:5000 flask-docker-demo
Tag your image: docker tag flask-docker-demo vikash95/flask-docker-demo:latest1
Assure that you're logged in: docker login
Push image to dockerhub: docker push vikash95/flask-docker-demo:latest1
Pull image from dockerhub: docker pull vikash95/flask-docker-demo:latest1
Run the Pulled image: docker run -p 5000:5000 vikash95/flask-docker-demo:latest1
```



DOCKERFILE

- It is a text file with instructions on how to build a Docker image. It's a blueprint for the image, specifying the environment, app, and dependencies.
- Key Components:
- FROM: Specifies the base image.
- COPY or ADD: Adds files from your host system into the image.
- RUN: Executes commands in the image, such as installing software.
- CMD or ENTRYPOINT: Defines the command that runs when the container starts.
- EXPOSE: Specifies the port the container will listen on.

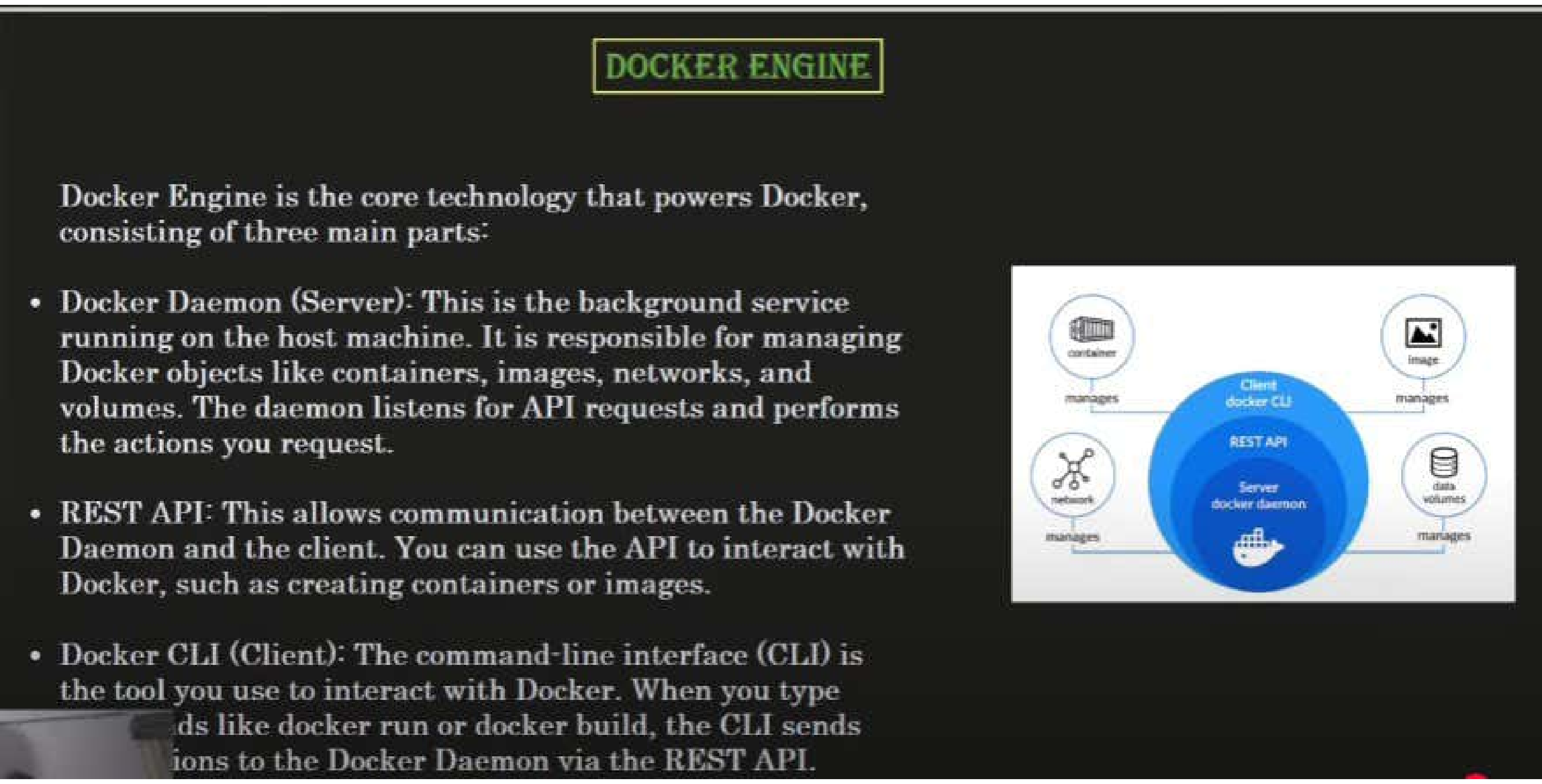


DOCKER VS VM

- Docker: Containers share the host OS and use fewer resources, making them lightweight. Starting a container takes seconds. Example: If you need to run 10 microservices, each can run in its own container, using less memory and starting quickly.

- VM: Virtual machines run a full OS for each instance, making them heavy and slower to start (minutes). They consume more resources because they need to virtualize hardware, not just the app. Example: Running the same 10 microservices in VMs will require significantly more memory and CPU because each VM has its own OS.

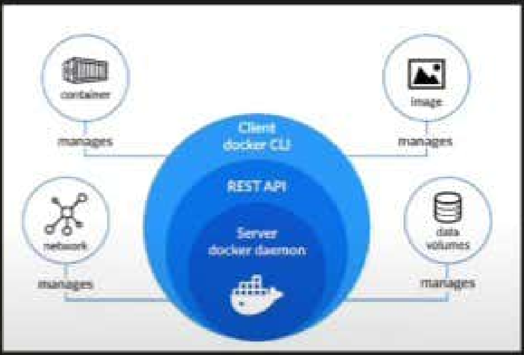
- In summary, Docker is lighter, faster, and more portable compared to VMs, which are heavier and slower due to full OS virtualization.

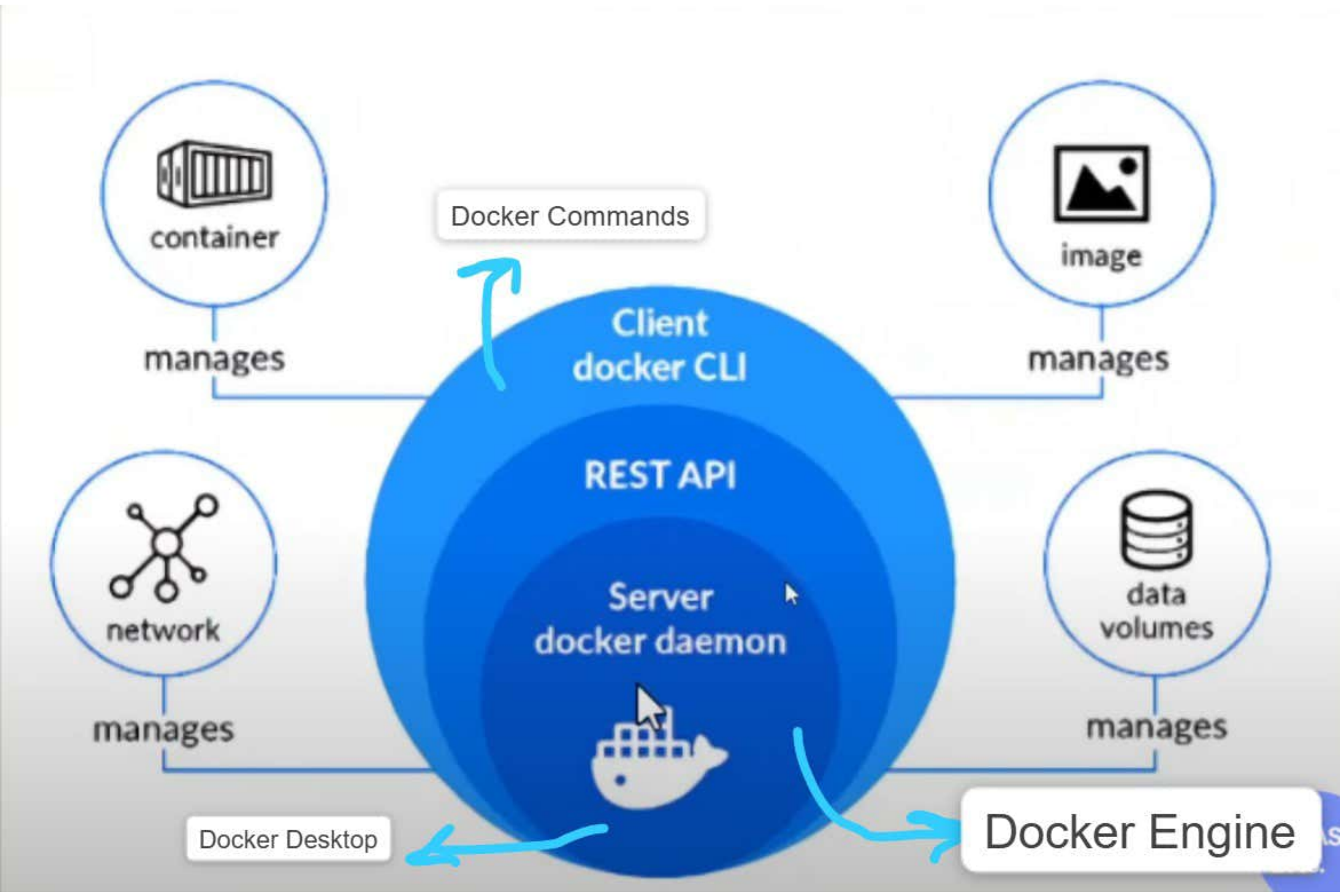


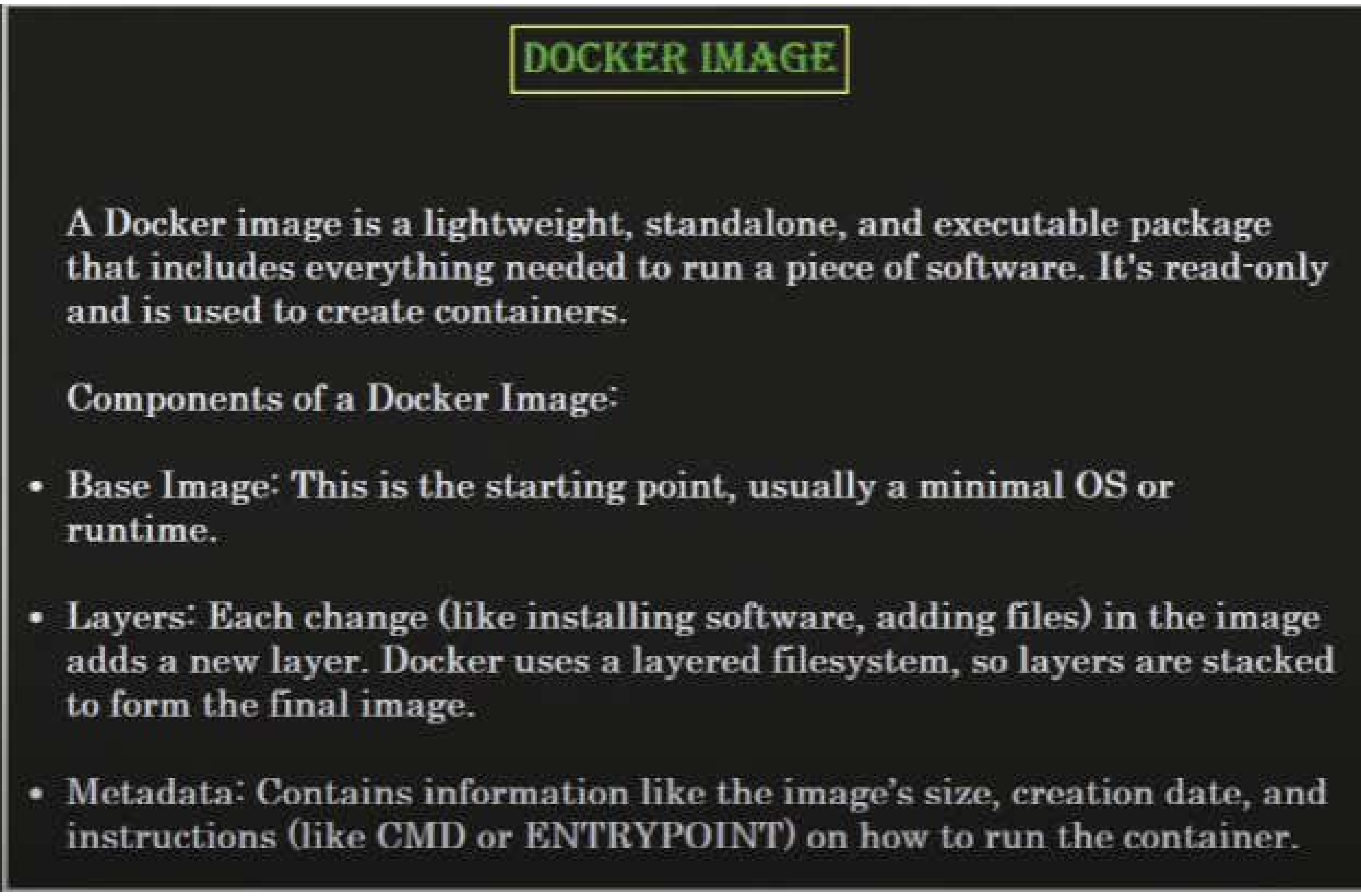
DOCKER ENGINE

Docker Engine is the core technology that powers Docker, consisting of three main parts:

- Docker Daemon (Server): This is the background service running on the host machine. It is responsible for managing Docker objects like containers, images, networks, and volumes. The daemon listens for API requests and performs the actions you request.
- REST API: This allows communication between the Docker Daemon and the client. You can use the API to interact with Docker, such as creating containers or images.
- Docker CLI (Client): The command-line interface (CLI) is the tool you use to interact with Docker. When you type ds like docker run or docker build, the CLI sends ions to the Docker Daemon via the REST API.




container
manages
Docker Commands
image
manages
Client
docker CLI
REST API
network
manages
Server
docker daemon
data
volumes
manages
Docker Desktop
Docker Engine

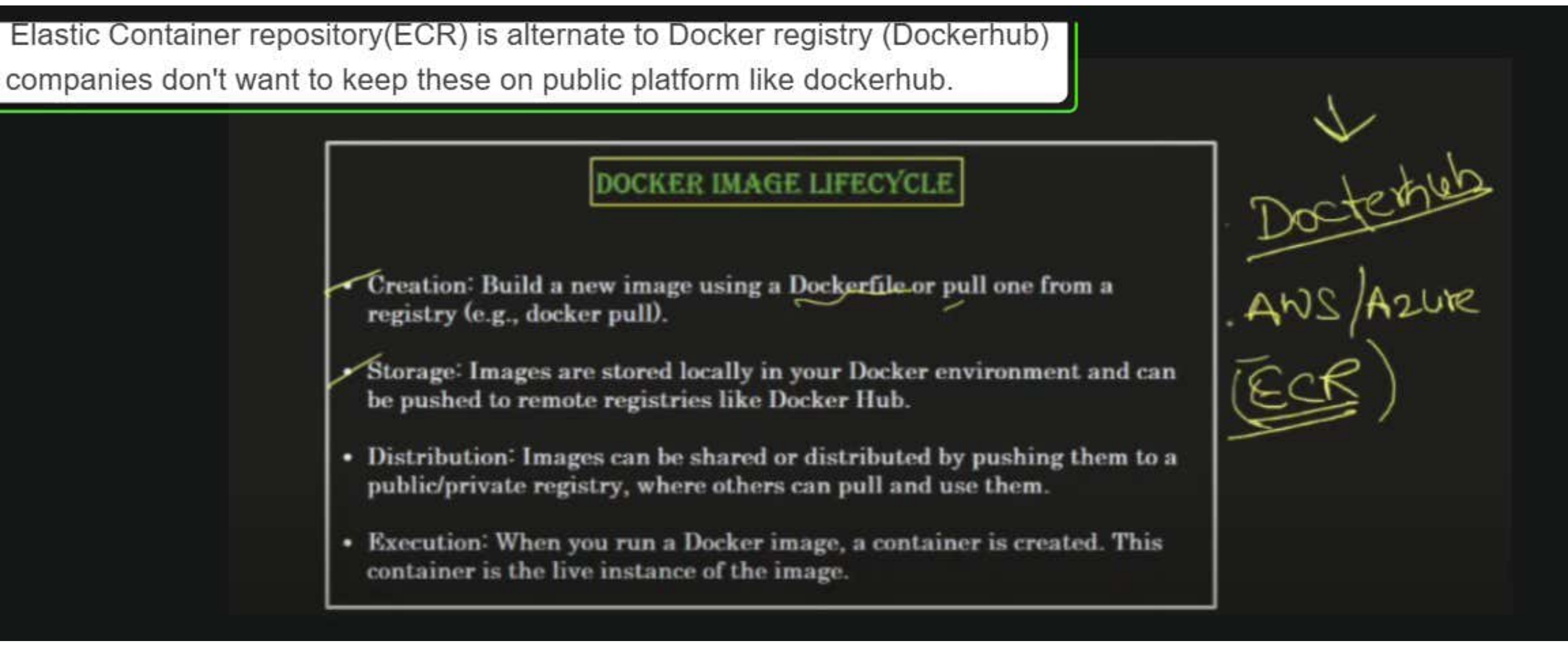


DOCKER IMAGE

A Docker image is a lightweight, standalone, and executable package that includes everything needed to run a piece of software. It's read-only and is used to create containers.

Components of a Docker Image:

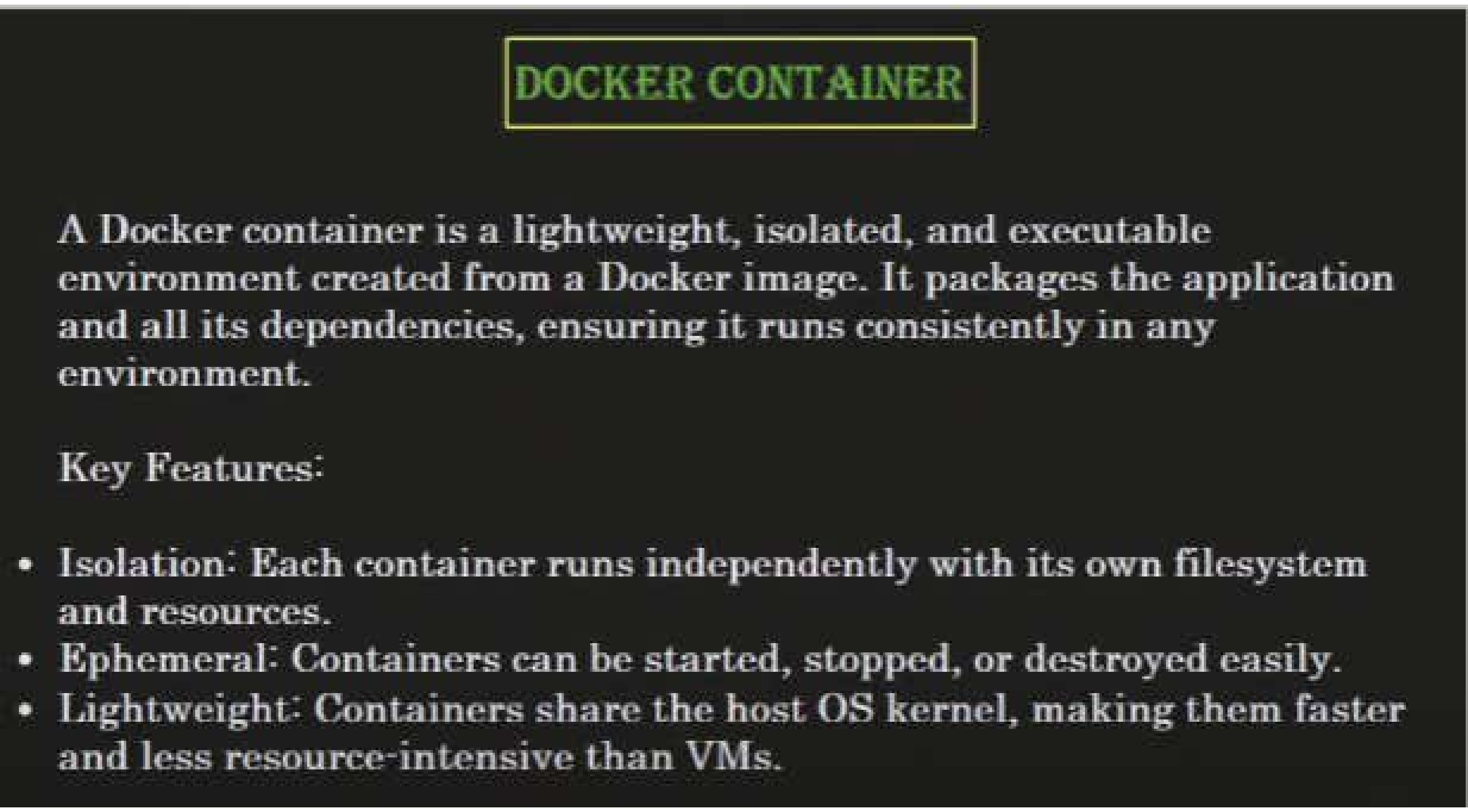
- Base Image: This is the starting point, usually a minimal OS or runtime.
- Layers: Each change (like installing software, adding files) in the image adds a new layer. Docker uses a layered filesystem, so layers are stacked to form the final image.
- Metadata: Contains information like the image's size, creation date, and instructions (like CMD or ENTRYPOINT) on how to run the container.



Elastic Container repository(ECR) is alternate to Docker registry (Dockerhub) companies don't want to keep these on public platform like dockerhub.

DOCKER IMAGE LIFECYCLE

- Creation: Build a new image using a Dockerfile or pull one from a registry (e.g., docker pull).
- Storage: Images are stored locally in your Docker environment and can be pushed to remote registries like Docker Hub.
- Distribution: Images can be shared or distributed by pushing them to a public/private registry, where others can pull and use them.
- Execution: When you run a Docker image, a container is created. This container is the live instance of the image.

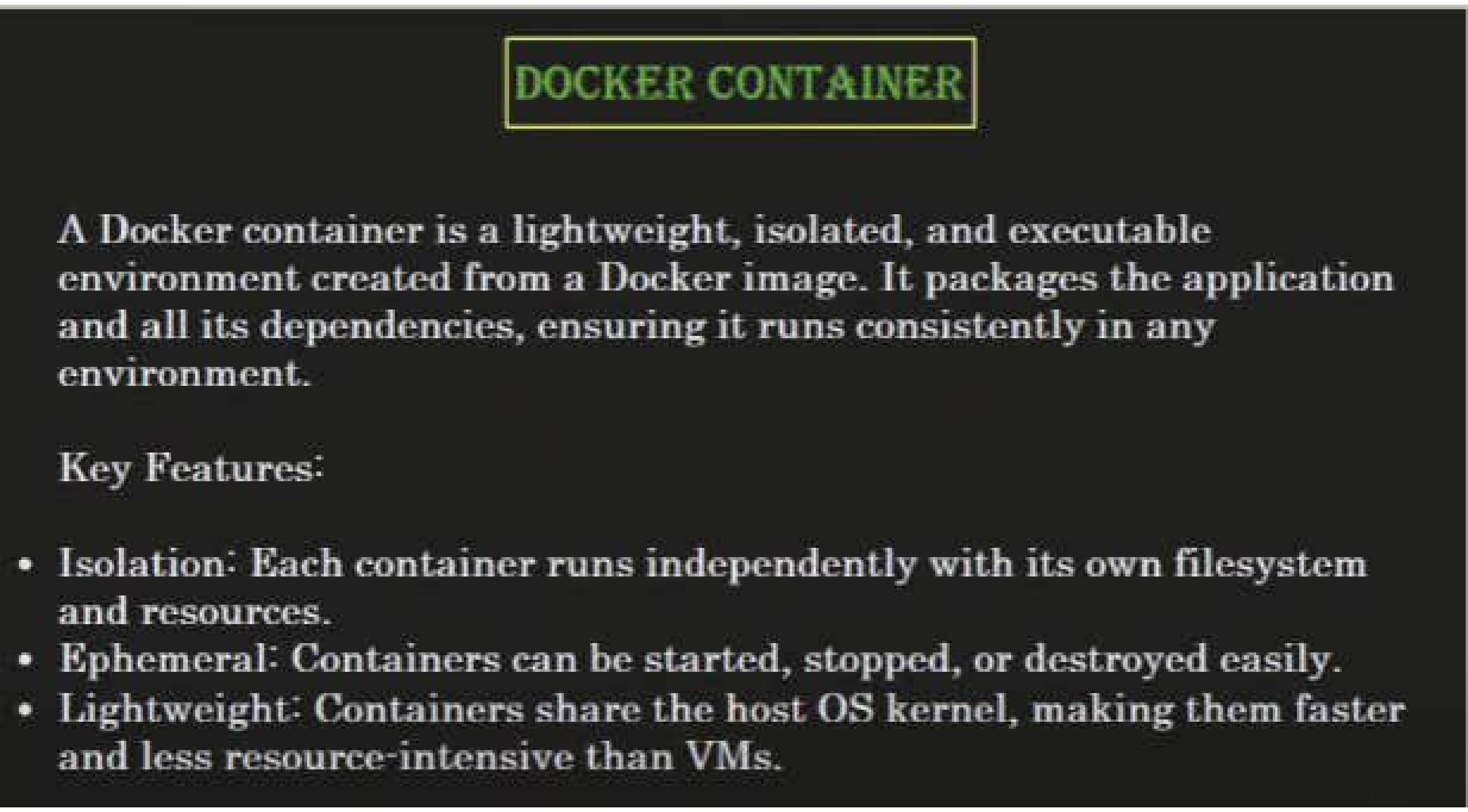


DOCKER CONTAINER

A Docker container is a lightweight, isolated, and executable environment created from a Docker image. It packages the application and all its dependencies, ensuring it runs consistently in any environment.

Key Features:

- Isolation: Each container runs independently with its own filesystem and resources.
- Ephemeral: Containers can be started, stopped, or destroyed easily.
- Lightweight: Containers share the host OS kernel, making them faster and less resource-intensive than VMs.



DOCKER CONTAINER

A Docker container is a lightweight, isolated, and executable environment created from a Docker image. It packages the application and all its dependencies, ensuring it runs consistently in any environment.

Key Features:

- Isolation: Each container runs independently with its own filesystem and resources.
- Ephemeral: Containers can be started, stopped, or destroyed easily.
- Lightweight: Containers share the host OS kernel, making them faster and less resource-intensive than VMs.