Docker 🐳

Docker is an **open-source platform** that lets you **build, package, and run applications** in a consistent and portable way using **containers**.

Docker lets you run your app **in its own little environment**, so it works the same on any computer — your laptop, a server, or the cloud.

What is a Container?

A **container** is like a mini-computer inside your computer.
It has:

- Your app
- Its dependencies (like Python, libraries, tools)
- Configuration settings

All in **one neat package**.
So if it runs on your machine, it'll run the same anywhere — no more "it works on my computer" issues.

Part 11

1. Problem Statement
2. Installation of Docker CLI and Desktop
3. Understanding Images v/s Containers
4. Running Ubuntu Image in Container
5. Multiple Containers
6. Port Mappings
7. Environment Variables
8. Dockerization of Node.js Application
9. Dockerfile
10. Caching Layers
11. Publishing to Hub
12. Docker Compose
13. Services
14. Port Mapping
15. Env Variables

Part 2

1. Docker Networking
2. Bridge
3. Host

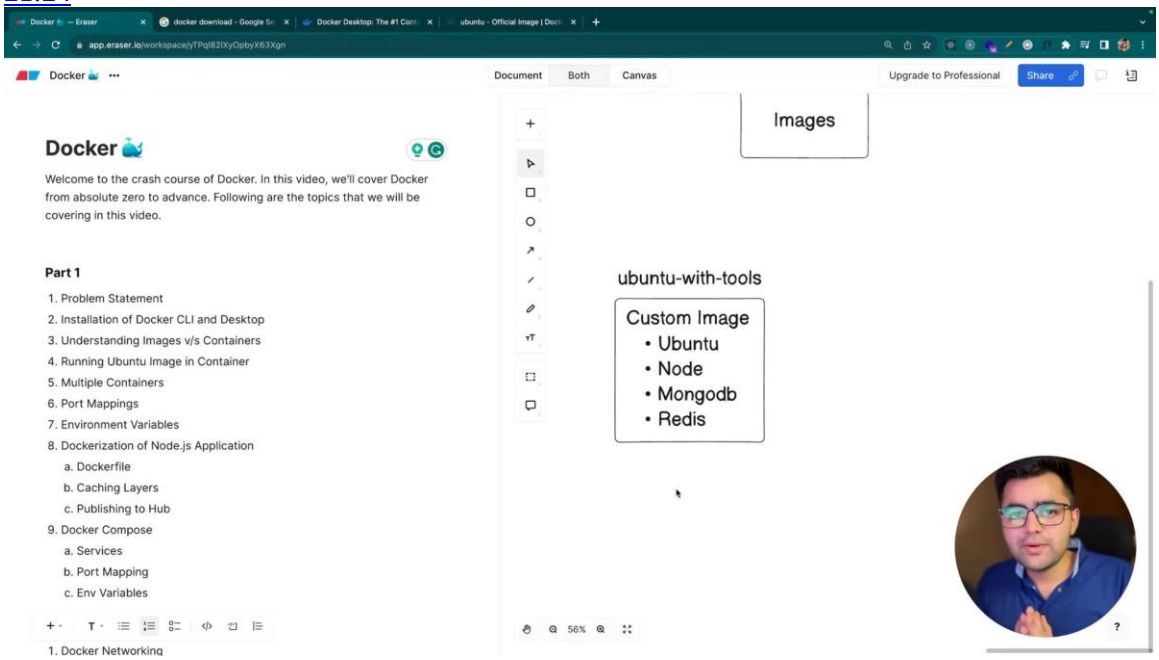
4. Volume Mounting
5. Efficient Caching in Layers
6. Docker Multi-Stage Builds

Bonus: Amazon Elastic Container Service and ECR

It is Having Images and Container to run your applications in Different Environment

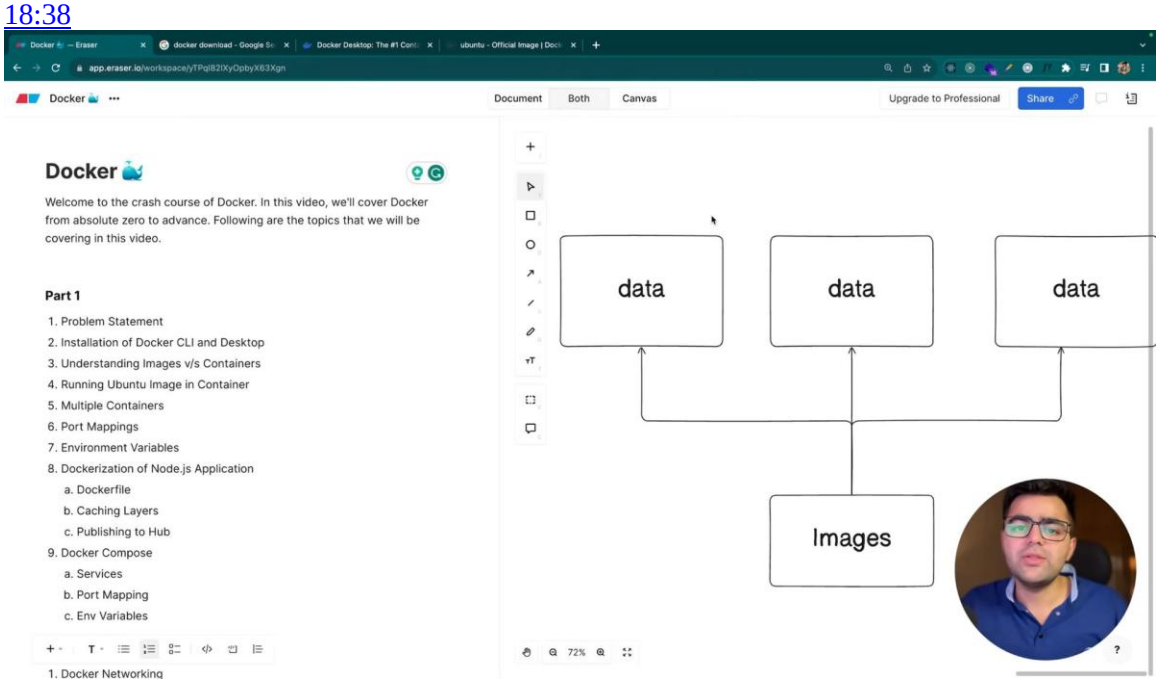
Images are OS that Having Coustomize packages

21:24



Container are To run the Images fro OS that having Customized package it is Isolated from outer system

18:38



Docker Command to Run

Basic Docker Commands

1. **docker run -it ubuntu**
    - **Description**: This command runs an Ubuntu container interactively (-it), which means you get access to a terminal inside the container.
    - **Usage**: It creates and starts a container from the ubuntu image.
2. **docker container ls**
    - **Description**: Lists all running containers.
    - **Usage**: Shows only the containers that are currently running.
3. **docker container ls -a**
    - **Description**: Lists all containers, including both running and stopped ones.
    - **Usage**: Shows containers that have been created, regardless of their state (running or stopped).
4. **docker start container_name**
    - **Description**: Starts a stopped container by its name or ID.
    - **Usage**: If the container was previously stopped, this command will start it again.
5. **docker stop container_name**
    - **Description**: Stops a running container by its name or ID.
    - **Usage**: Terminates a container that is currently running.
6. **docker exec container_name ls**

- **Description**: Executes the ls command inside a running container.
- **Usage**: This command runs ls in the container to list files and directories.

7. **docker exec -it container_name bash**
   - **Description**: Opens an interactive bash session inside a running container.
   - **Usage**: Attaches your terminal to the container and gives you a shell (bash).
8. **ctr+d**
   - **Description**: Used to exit the interactive terminal (bash session) inside the container.
   - **Usage**: Exits from the bash shell session within the container and returns you to your host terminal.
9. **docker images or docker image ls**
   - **Description**: Lists all available Docker images on the host.
   - **Usage**: Shows the locally stored images with details such as repository, tag, and image size.
10. **docker run -it node**
    - **Description**: Runs an interactive Node.js container.
    - **Usage**: Starts a Node.js container from the Node.js image and gives you access to the container terminal.

Port Mapping Commands

11. **docker run -it -p 1025:1025 mailhog/mailhog**
    - **Description**: Runs the Mailhog container and maps port 1025 on your local machine to port 1025 inside the container.
    - **Usage**: Allows access to Mailhog's service from your local machine on port 1025.
12. **docker run -it mailhog/mailhog**
    - **Description**: Runs a Mailhog container without port mapping.
    - **Usage**: This starts the Mailhog container, but you won't be able to access it from your local machine unless you specify port mapping.
13. **docker run -it -p <host_port>:<container_port> <image_name>**
    - **Description**: General syntax for port mapping.
    - **Usage**: This allows you to map ports between the host and the container. For example, -p 1025:1025 maps port 1025 on the host machine to port 1025 inside the container.

Additional Commands for Managing Containers and Images

14. **docker ps**
    - **Description**: Lists all running containers (same as docker container ls).
    - **Usage**: Shows running containers only.

15. **docker ps -a**
    - **Description**: Lists all containers, both running and stopped (same as docker container ls -a).
    - **Usage**: Shows all containers, regardless of whether they are running or stopped.
16. **docker rm container_name**
    - **Description**: Removes a stopped container by its name or ID.
    - **Usage**: Deletes a container from your system, but only if it's stopped.
17. **docker rmi image_name**
    - **Description**: Removes an image from your system.
    - **Usage**: Deletes a local image from the system.
18. **docker pull <image_name>**
    - **Description**: Pulls a Docker image from the Docker Hub or a registry.
    - **Usage**: Downloads an image (like ubuntu, node, mailhog) to your local machine.
19. **docker build -t <image_name> <directory_path>**
    - **Description**: Builds a Docker image from a Dockerfile in a specified directory.
    - **Usage**: Use this command to build custom images from a Dockerfile.
20. **docker-compose up**
    - **Description**: Starts containers as specified in a docker-compose.yml file.
    - **Usage**: Useful for running multi-container applications, like a web app with a database, all from one configuration file.
21. **docker-compose down**
    - **Description**: Stops and removes containers defined in a docker-compose.yml file.
    - **Usage**: Shuts down the environment created by docker-compose up.

Summary of Useful Docker Commands

- **Managing Containers**:
    - docker container ls: List running containers.
    - docker container ls -a: List all containers (running and stopped).
    - docker start <container_name>: Start a stopped container.
    - docker stop <container_name>: Stop a running container.
    - docker exec -it <container_name> bash: Start a bash shell in the container.
- **Managing Images**:
    - docker images: List images on the system.
    - docker pull <image_name>: Download an image.
    - docker rmi <image_name>: Remove an image.

- **Port Mapping**:
    - docker run -p <host_port>:<container_port> <image_name>: Run a container with port mapping.