**DOCKER**

1. How we install?

* First visit **doc.docker.com** website and select the required download for the system.
* Before downloading we have to check whether our system support docker or not , for that we have to check our windows version as well as ckeck wsl enable or not .

1. For checking windows version we press win+R and enter winver (gives the version of the window).
2. For ckecking wsl enable or not press window key and search for enable windows and there we go for the [turn windows feature on or off] and search for wsl (enabled or not).

* If we don’t have wsl then we go to the terminal and run this command

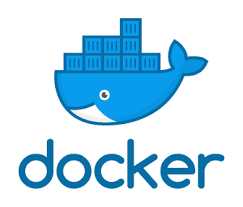
wsl - -install. (this would install wsl for us.)

* Now we check whether our virtualization for windows is on or off .

So we press ctrl+shift+esc (task manager) go to the performance icon and select CPU and check for the virtualization (right hand side) is enabled.

* Now we download the required docker desktop for our system.
* Create account or sign up and go ahead.
* We can also use docker with github account or google account.

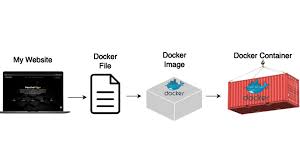
Installation Completed



**WHAT IS DOCKER AND WHY WE USE IT?**

**DOCKER?**

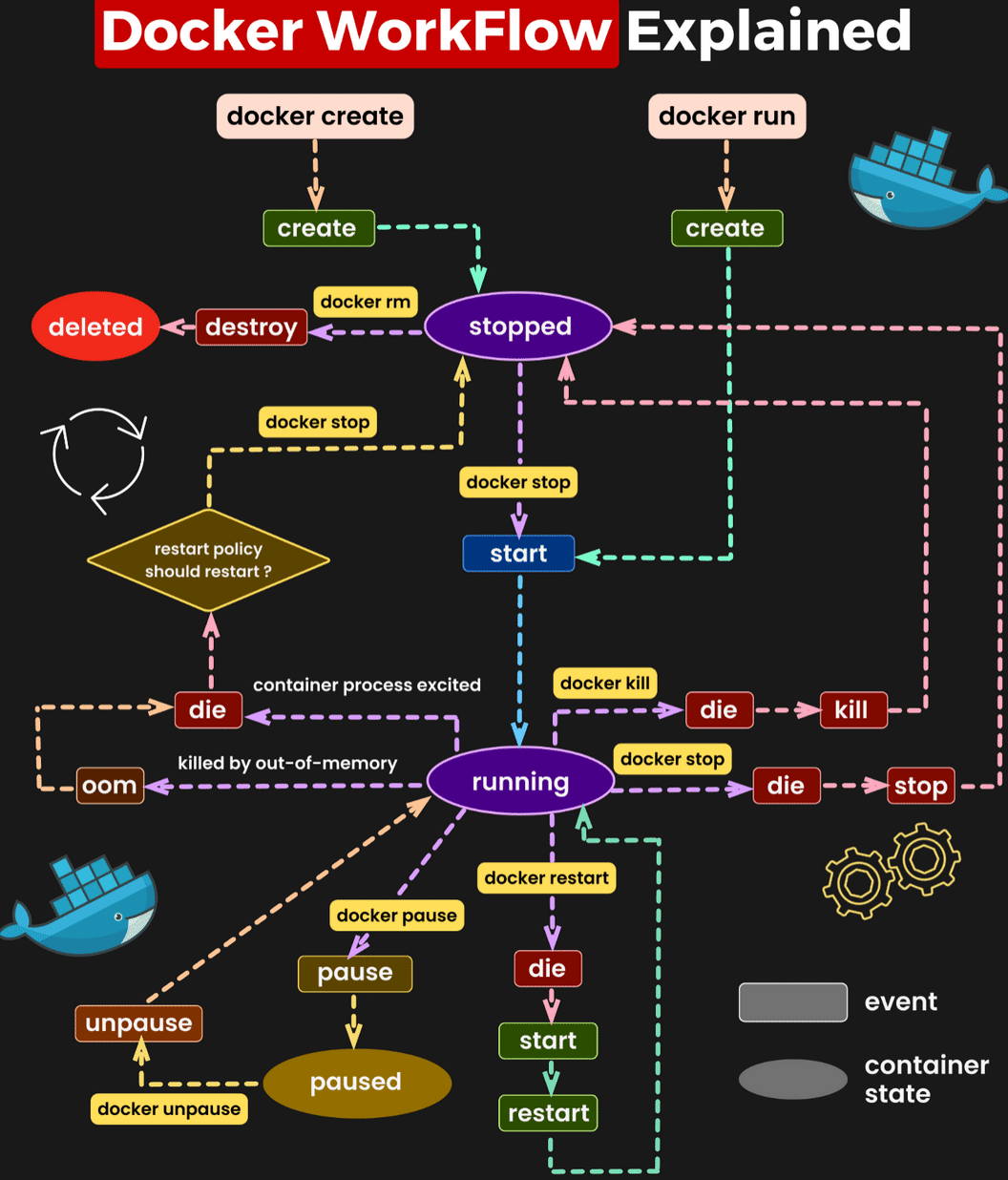
* **Docker** is an open-source platform that makes it easier to build, package, and run applications. It allows developers to bundle their applications with all the necessary dependencies (like libraries, tools, and settings) into a single unit called a **container**.

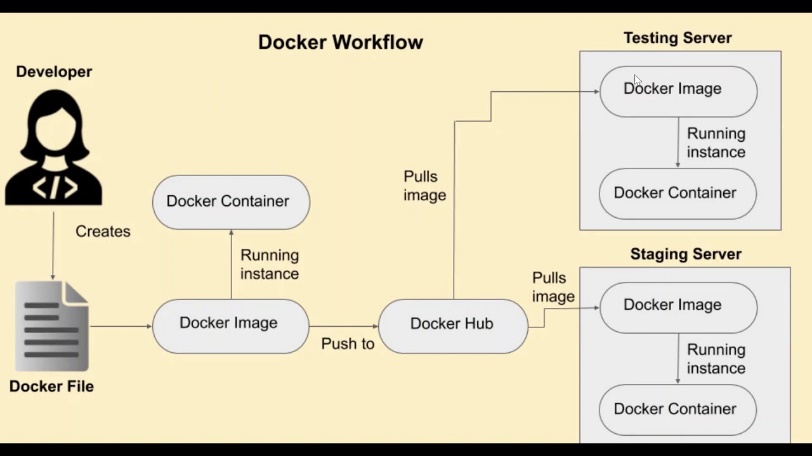


**Explanation:**

* Normally, running an application on different machines can be tricky because each machine might have a different setup or environment. Docker solves this problem by packaging everything the application needs (code, libraries, settings) into a container.
* Containers are isolated from each other and the host system, but they share the operating system's kernel, making them much lighter and faster than virtual machines (VMs).

Think of it as packing up our app into a portable box that can be shipped and run anywhere without worrying about compatibility issues.





We use docker for these following reasons:

We use Docker because it simplifies and improves the development, deployment, and scaling of applications. Here are the key reasons why Docker is widely used:

**1.** **Consistency Across Environments.**

-"It works on my machine!" is a common problem in software development. Docker solves this by ensuring that the app runs the same way on any machine, regardless of differences in the environment (e.g., different operating systems or configurations).

**2.** **Portability**

- Docker containers are \*\*lightweight\*\* and can be easily moved between different environments (development, testing, production) without changes. This makes it ideal for microservices and cloud-native applications.

**3.** **Isolation**

- Containers run isolated from each other and the host system. This means you can run multiple applications on the same machine without conflicts in libraries, settings, or dependencies.

**4.** **Faster Deployment**

- Docker containers start up much faster than virtual machines (VMs), often in seconds, making the development and testing process faster and more efficient.

**5.** **Efficient Resource Use**

- Unlike VMs, containers share the host operating system’s kernel, which makes them lightweight, uses fewer system resources (CPU, RAM), and allows running more containers on a single machine.

**6.** **Scalability**

- Docker makes it easier to scale applications by spinning up multiple containers of the same app. This is particularly useful in cloud environments where you might need to scale services up or down based on demand.

**7**. **Simplified Dependency Management**

- With Docker, you can package your application with all its dependencies into a container. This ensures that everyone running the container has the same dependencies, without worrying about mismatches or missing components.

**8. Version Control for Applications**

- Docker allows you to version your container images, which means you can easily roll back to a previous version if something goes wrong during an update or release.

**9. Better Collaboration**

- Docker makes it easy for development teams to share and collaborate on projects, as developers can distribute the same Docker image that can run anywhere.

**10. CI/CD Integration**

- Docker integrates smoothly with Continuous Integration/Continuous Deployment (CI/CD) pipelines, automating the testing and deployment process.

In summary, Docker is used to streamline the process of developing, deploying, and running applications in consistent, portable, and resource-efficient containers.

**BUILD A SMALL PROJECT USING DOCKER**

1. First we open the docker desktop in our terminal, then we run ‘docker run node’.
2. After that we make a folder anywhere on the local eg. Docker-code

And open it with the terminal and make a directory => mkdir my-app.

Open it with VS Code with “code .”

1. Now in vs code open the terminal and run the “npm init” command then install expressjs in the project ‘npm i express’.
2. Make a file name index.js .in the editor.
3. Now we write a simple app code

= cosnt express = require(‘express’);

const app = express();

app.get("/", (req, res) => {

    res.json({message: "Hello from Docker!🐋"});

    res.end();

});

app.listen(9000 ,() => console.log("Server is running at port 9000"));

1. Now we make a ‘Dockerfile’ in the editor , and write the following command;-

# Use the latest Node.js image

->FROM node:latest

# Copy the entire project to the /home/app directory inside the container

->COPY . /home/app

# Set the working directory inside the container

->WORKDIR /home/app

# Install project dependencies

->RUN npm install

# Expose port 9000

->EXPOSE 9000

# Run the app with node (assuming 'index.js' is the entry file)

->CMD ["node", "index.js"]

Expalnation:

* **FROM node**

: This pulls the latest Node.js image from Docker Hub.

* **COPY . /home/app**: Copies the content of your local project folder to /home/app in the Docker container.
* **WORKDIR /home/app**: Sets /home/app as the working directory for any subsequent commands.
* **RUN npm install**: Installs the project dependencies listed in package.json.
* **EXPOSE 9000**: Tells Docker to expose port 9000 (assuming your application runs on this port).
* **CMD ["node", "index.js"]**: This starts your application by running the index.js file with Node.js.

1. Now in VS code terminal we run the command docker build -t myapp .

This command says that build docker file with tag myapp and . says the current directory(make sure we are in the same directory where we work.)

Its make an image of myapp in the docker desktop.

1. Now in our project folder (docker-code)terminal run the command ‘docker run -it myapp’ . It shows that the server is running on port 9000.
2. But the server shows nothing on the local, because the server is running within the docker. So to make a connection of these two servers we make a port configuration.
3. So next we go to the docker terminal and run the command “docker run -p 9000:9000 myapp(image name or the tag name)”. Here we want to say that docker run the server on the port 9000 which is running on the port 9000 of the local.
4. If we want to upload this on dockerhub then we open the VS code terminal and write the command docker build -t shreyasingh30/mynodeapp (username and the name of the image container). It will build the image of the shreyasingh30/mynodeapp in the docker desktop.
5. Now I want to push it on the docker hub, so give the command “docker push shreyasingh30/mynodeapp”.It will push it on the docker hub.
6. Now I want to run it on my virtual system so for that open the virtual system along with it’s terminal and give the command :-

“docker run -it -p 9000:9000 shreyasingh30/mynodeapp” (username with image name and also give the port configuration).

There might be error showing after running this command like port is already located ,because our image is running on the docker desktop so we can remove all the filles from docker desktop and then we run the command ,it will work now.

Image in executable format is now called containers.

To be Continued…………..