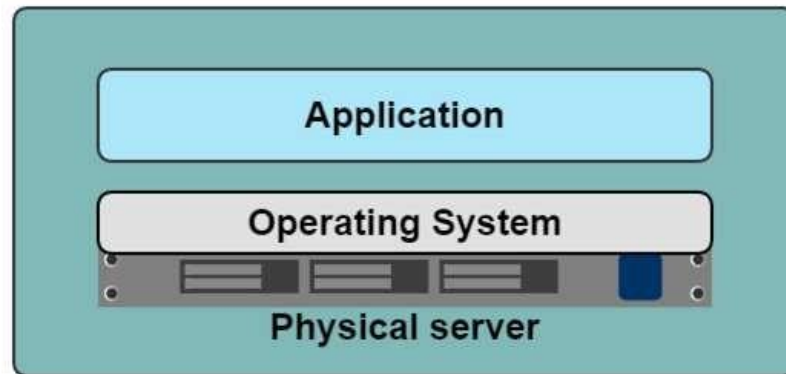


Docker

# A History Lesson

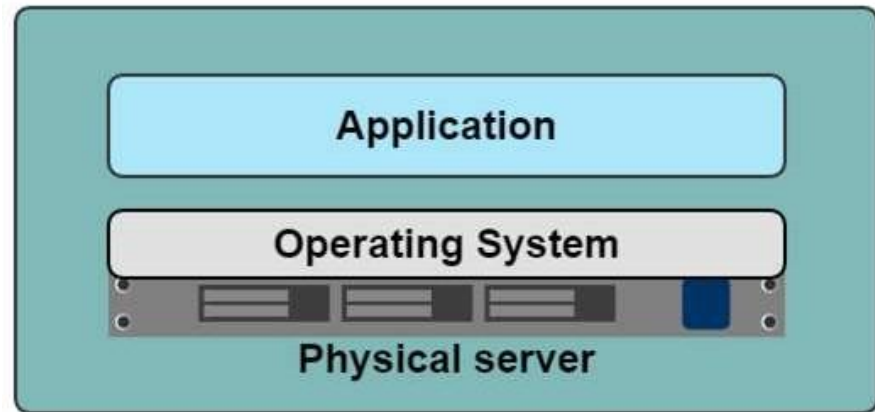
In the Dark Ages

**One application on  
one physical server**



# Historical limitations of application deployment

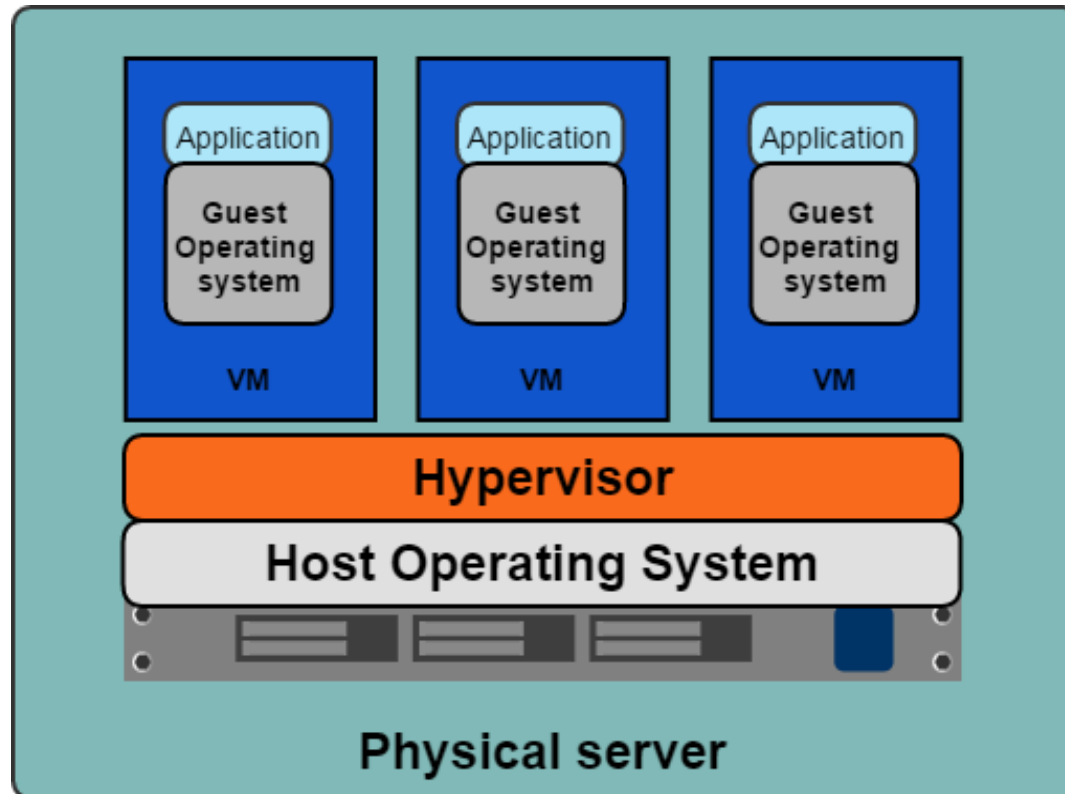
- Slow deployment times
- Huge costs
- Wasted resources
- Difficult to scale
- Difficult to migrate
- Vendor lock in



# A History Lesson

## Hypervisor-based Virtualization

- One physical server can contain multiple applications
- Each application runs in a virtual machine (VM)



# Benefits of VMs

- Better resource pooling
  - One physical machine divided into multiple virtual machines
- Easier to scale
- VMs in the cloud
  - Rapid elasticity
  - Pay as you go model

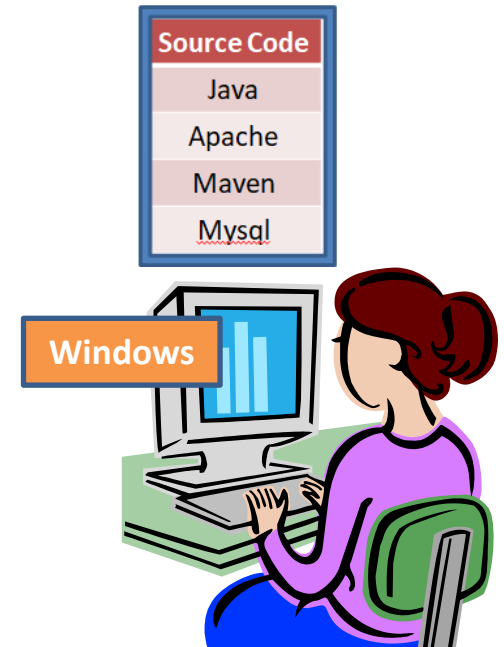
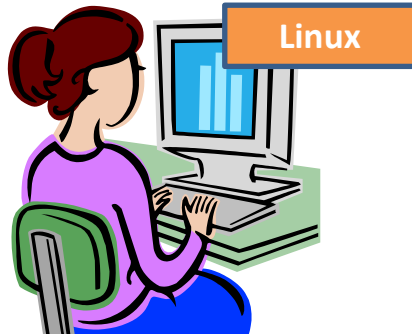


# Limitations of VMs

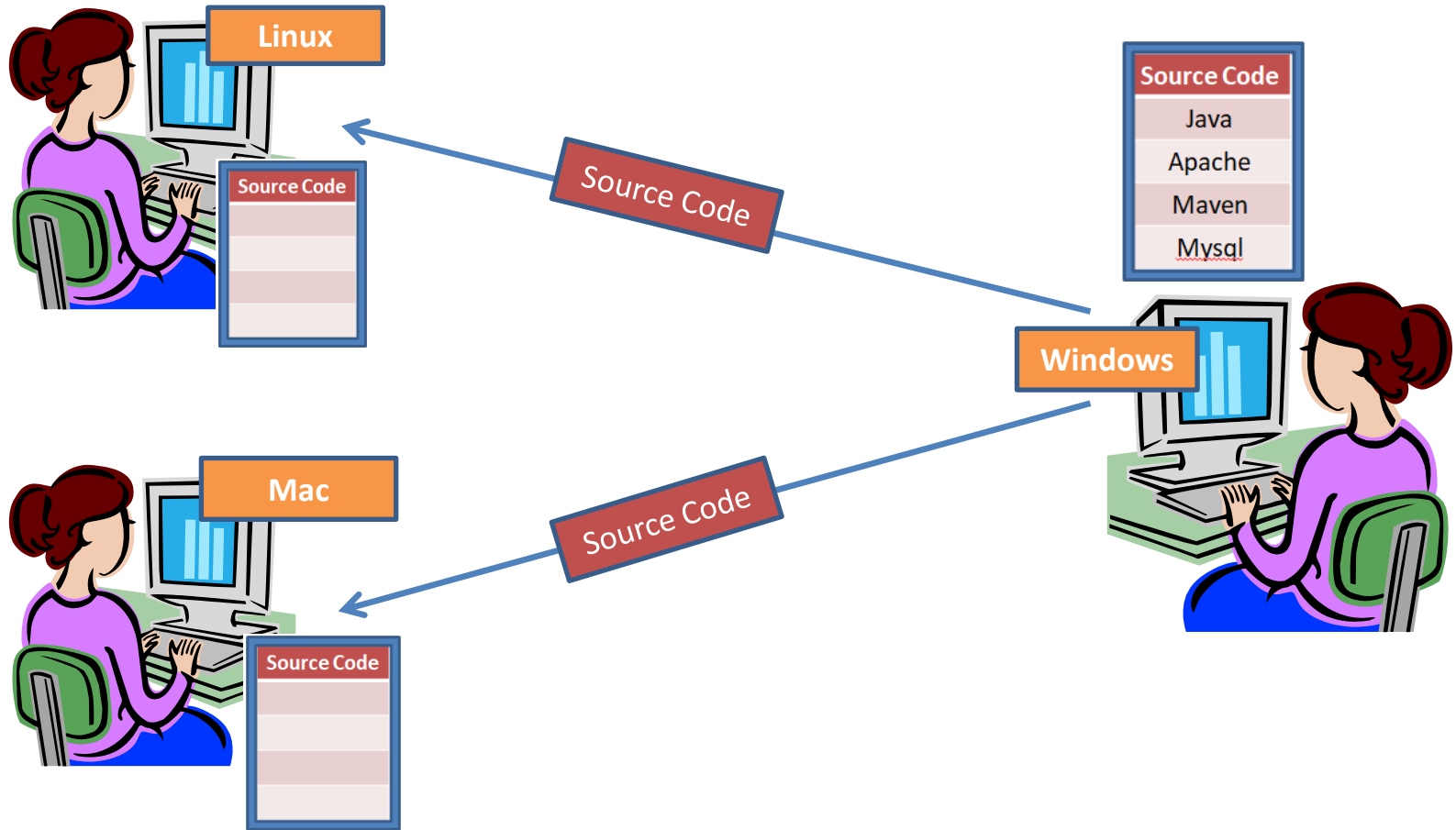
- Each VM stills requires
  - CPU allocation
  - Storage
  - RAM
  - An entire guest operating system
- The more VMs you run, the more resources you need
- Guest OS means wasted resources
- Application portability not guaranteed



# Why Docker?

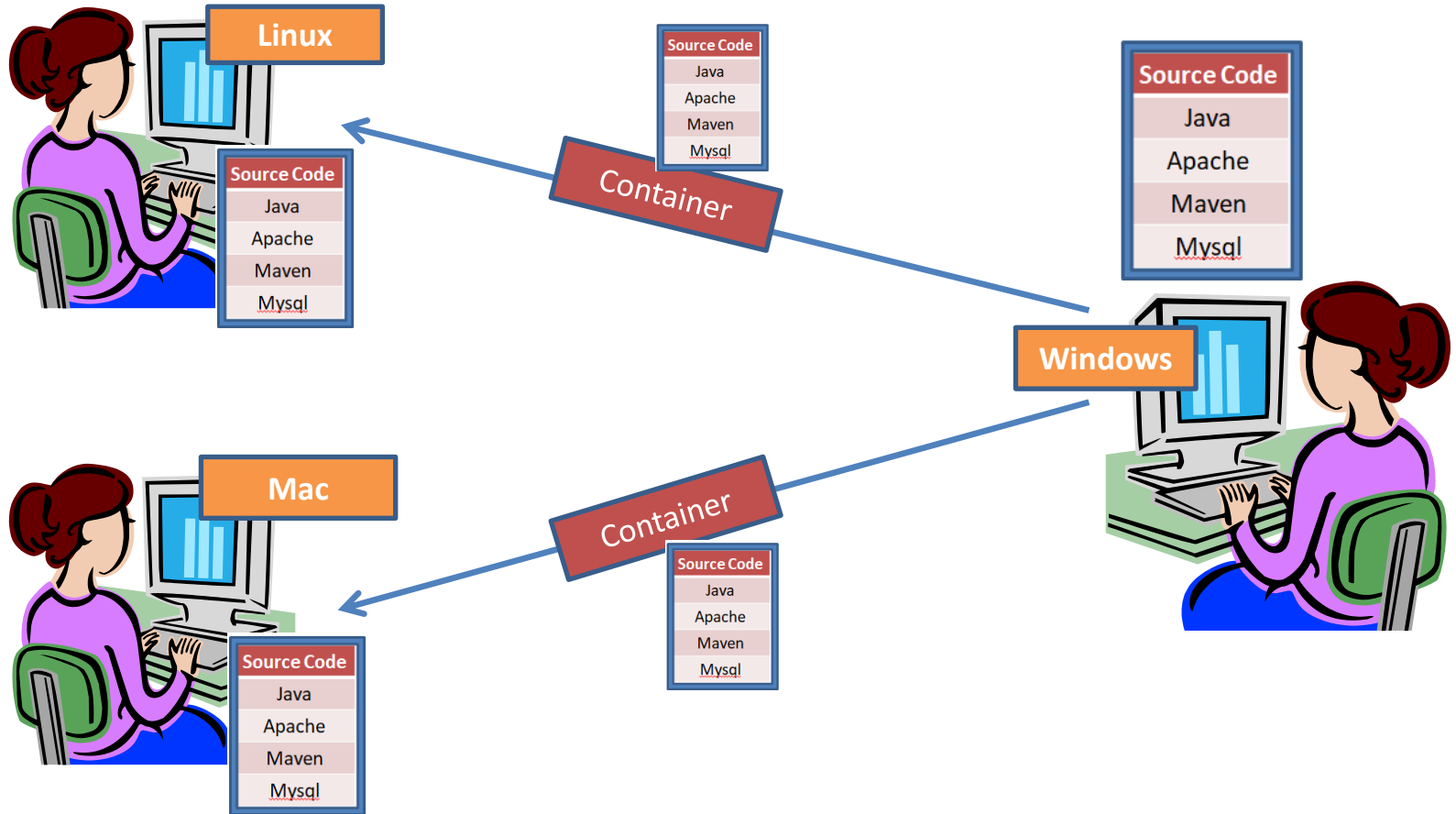


# Why Docker?

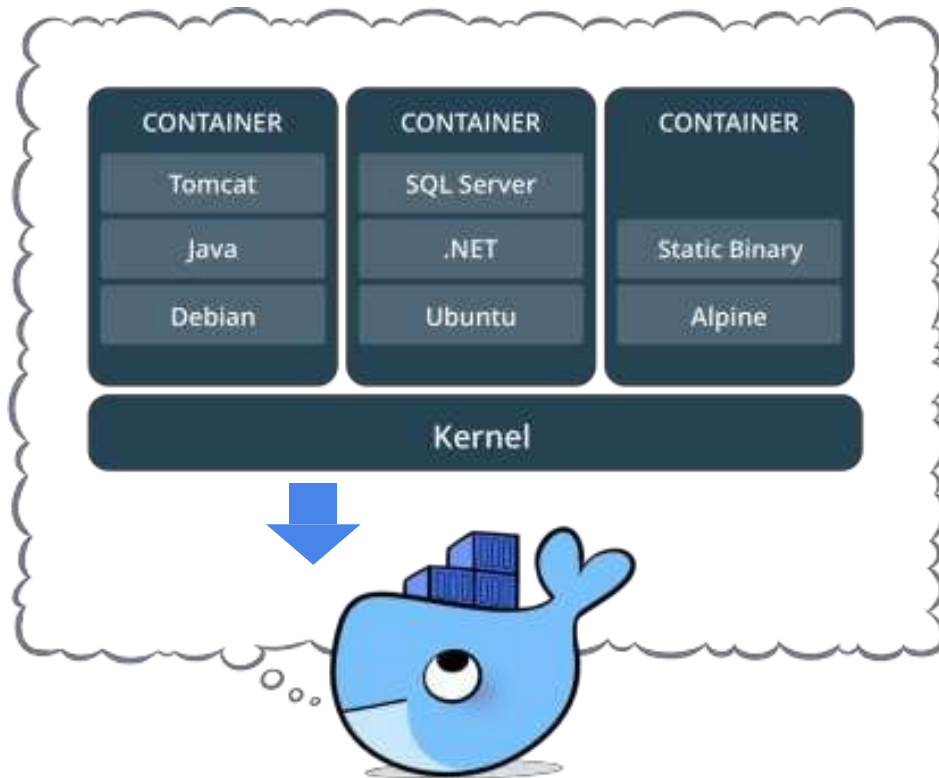




# Containerization



# What is a container?

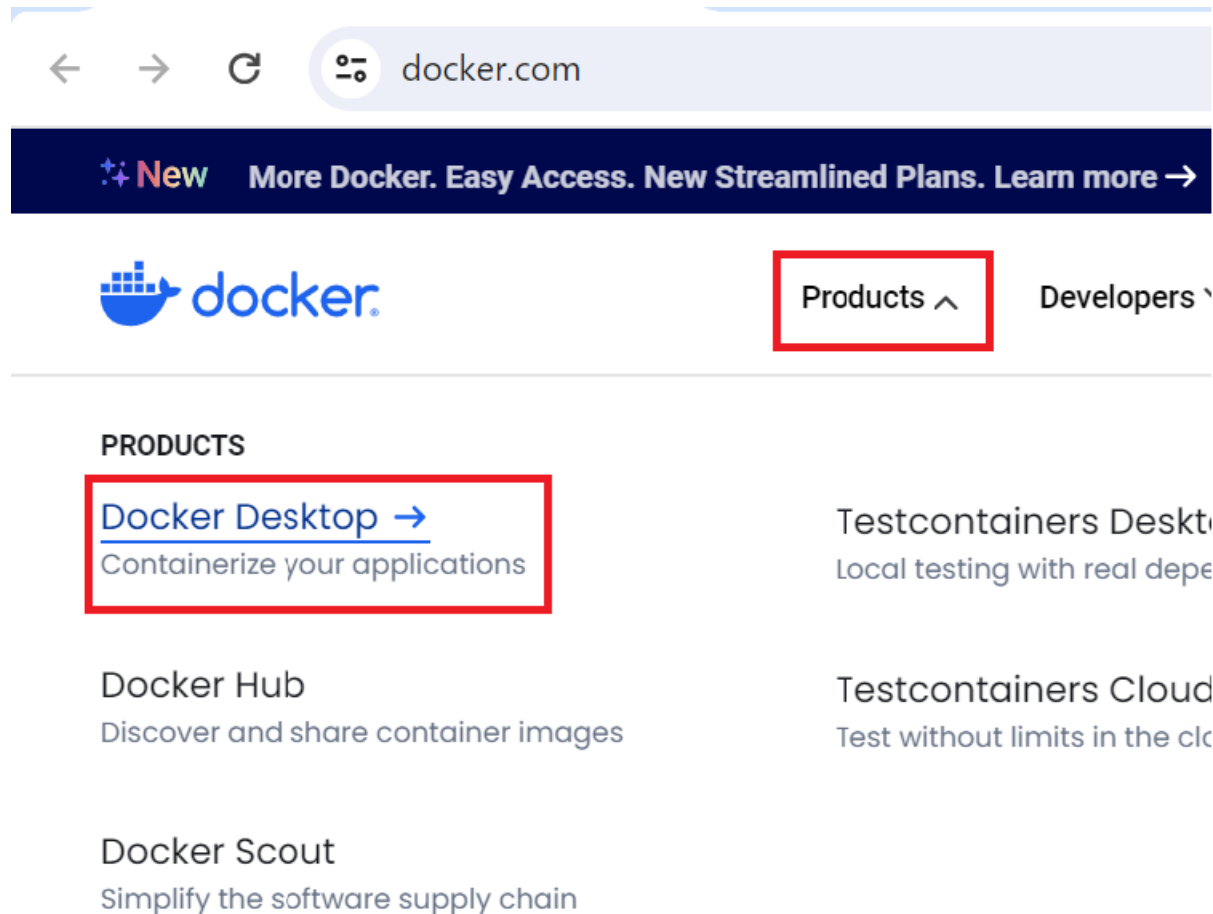


- Standardized packaging for software and dependencies
- Isolate apps from each other
- Share the same OS kernel
- Works with all major Linux and Windows Server

# Docker Installation

- Docker is been created for Linux, and it will run on linux directly
- For windows, we need HyperV/WSL
- For Mac, We need Hyperkit


# Docker Desktop



The screenshot shows the Docker website homepage. At the top, there is a navigation bar with the Docker logo and a 'Products' menu item highlighted with a red box. Below the navigation bar, there is a 'PRODUCTS' section. The first item in this section is 'Docker Desktop', which is also highlighted with a red box. The description for Docker Desktop is 'Containerize your applications'. Other products listed include Docker Hub, Docker Scout, Testcontainers Desktop, and Testcontainers Cloud.

← → ↻ 🔍 docker.com

**New** More Docker. Easy Access. New Streamlined Plans. Learn more →

 **Products** ^ Developers ^

---

**PRODUCTS**

[Docker Desktop →](#)  
Containerize your applications

Docker Hub  
Discover and share container images

Docker Scout  
Simplify the software supply chain

Testcontainers Desktop  
Local testing with real dependencies

Testcontainers Cloud  
Test without limits in the cloud

# Download & Install



Download for Windows - AMD64 ^

Learn more



Download for Windows - ARM64



Download for Mac - Intel Chip



Download for Mac - Apple Silicon



Download for Linux

1.06% / 1000% (10 cores allocated)

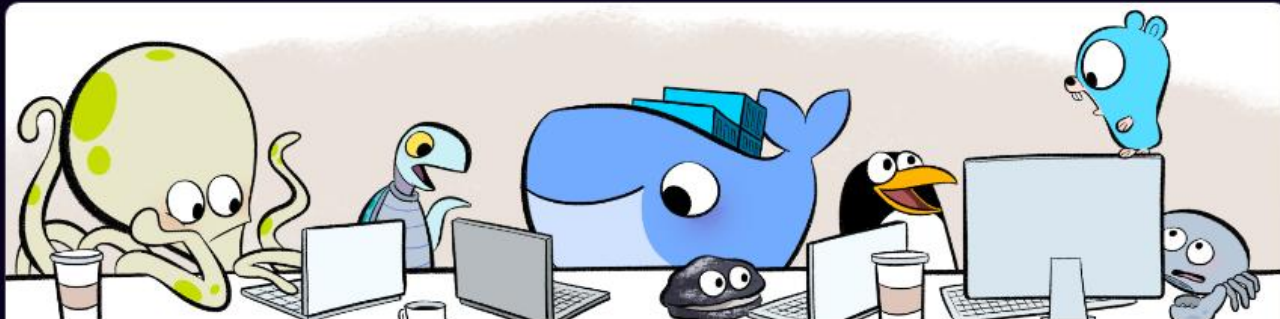
127.45 MB / 15.1 GB



Recycle Bin



Docker Desktop



## Docker Subscription Service Agreement

By selecting **accept**, you agree to the [Subscription Service Agreement](#), the [Docker Data Processing Agreement](#), and the [Data Privacy Policy](#).

Commercial use of Docker Desktop at a company of more than 250 employees OR more than \$10 million in annual revenue requires a paid subscription (Pro, Team, or Business). [See subscription details](#)

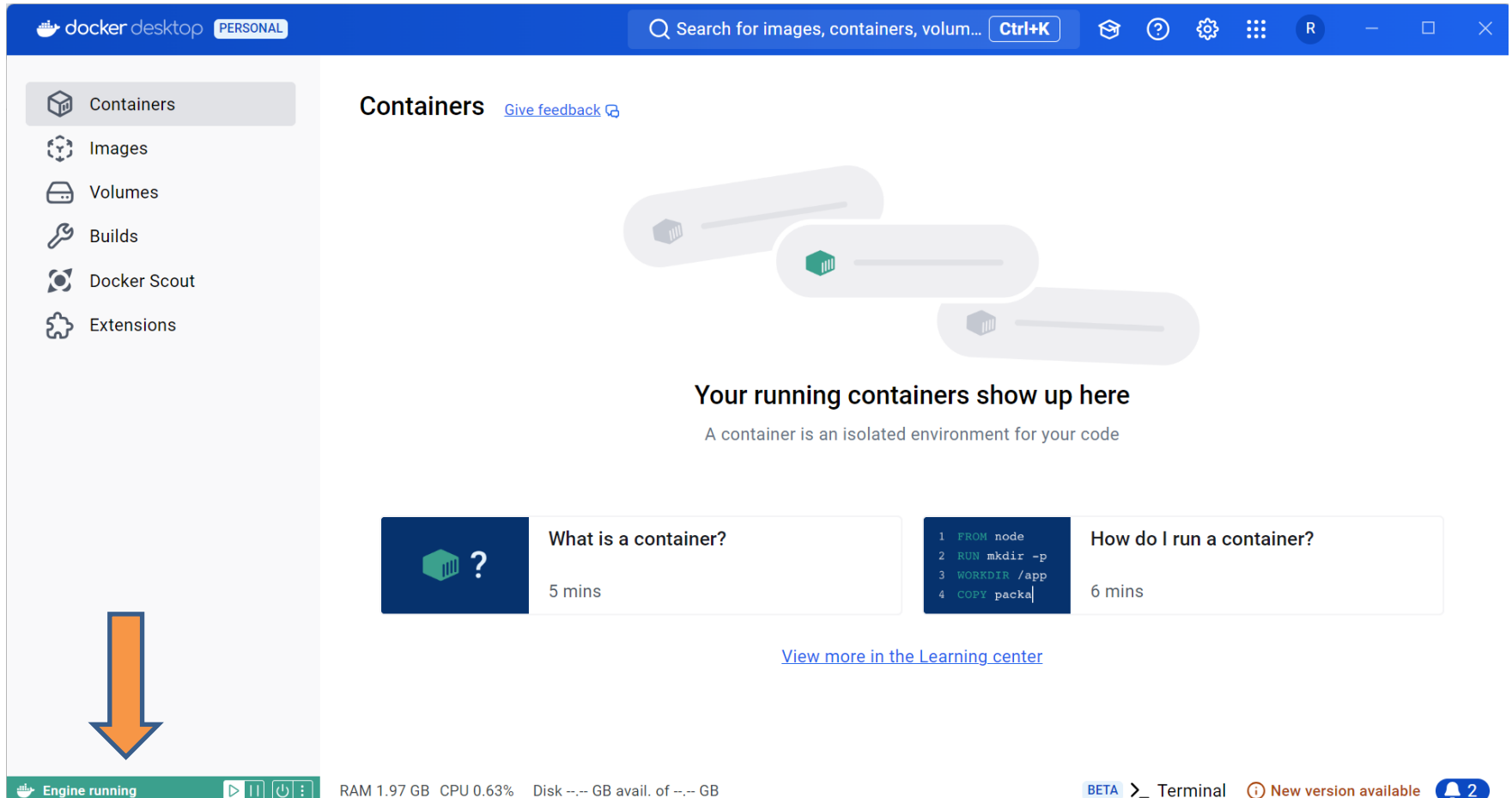
[View Full Terms](#)



**Accept**

**Close**

# Docker Engine is Started



# Confirm it with cmd

```
Command Prompt
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ritit>docker --version
Docker version 27.1.1, build 6312585

C:\Users\ritit>docker info
Client:
 Version:      27.1.1
 Context:      desktop-linux
 Debug Mode:   false
 Plugins:
  buildx: Docker Buildx (Docker Inc.)
   Version:  v0.16.1-desktop.1
   Path:      C:\Program Files\Docker\cli-plugins\docker-
  compose: Docker Compose (Docker Inc.)
```



**DOCKER IMAGE**

# Docker Image

A **Docker image** is a lightweight, stand-alone, and executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, dependencies, and environment configurations.

- **Docker Image:** A blueprint or template for creating containers.
- **Docker Container:** A running instance of a Docker image.

# Docker Image?



Software  
Application



Image

# Image Creation



# Docker Hub



# Pulling an image from Hub

```
F:\Docker>docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
c1ec31eb5944: Pull complete
Digest: sha256:91fb4b041da273d5a3273b6d587d62d518
Status: Downloaded newer image for hello-world:latest
docker.io/library/hello-world:latest

F:\Docker>
```

# Show Image

```
F:\Docker>docker images
REPOSITORY      TAG         IMAGE ID      CREATED        SIZE
hello-world     latest     d2c94e258dcb  16 months ago 13.3kB

F:\Docker>|
```

The screenshot shows the Docker Desktop application window. The left sidebar contains navigation icons for Containers, Images (selected), Volumes, Builds, Docker Scout, and Extensions. The main panel is titled 'Images' and has tabs for 'Local' and 'Hub'. Below the tabs, it shows '0 Bytes / 0 Bytes in use' and '1 images'. A search bar and filter icons are present. A table lists the local images:

| <input type="checkbox"/> | Name                        | Tag    | Status | Created    | Size     | Actions |
|--------------------------|-----------------------------|--------|--------|------------|----------|---------|
| <input type="checkbox"/> | <a href="#">hello-world</a> | latest | Unused | 1 year ago | 13.25 KB |         |
|                          | d2c94e258dcb                |        |        |            |          |         |

# Run an image in cmd

```
F:\Docker>docker run hello-world
```

```
Hello from Docker!
```

```
This message shows that your installation appears to be working correctly.
```

```
To generate this message, Docker took the following steps:
```

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

```
To try something more ambitious, you can run an Ubuntu container with:
```

```
$ docker run -it ubuntu bash
```

```
Share images, automate workflows, and more with a free Docker ID:
```

```
https://hub.docker.com/
```





```
For more examples and ideas, visit:
```

```
https://docs.docker.com/get-started/
```


```
F:\Docker>|
```



# Run an image in GUI

[hello-world](#)  
d2c94e258dcb  latest [In use](#) 1 year ago 13.25 KB  


Run






<

stupefied\_dijkstra

[hello-world:latest](#)

8e9c3637e9b0 

STATUS  
Exited (0) (0 seconds ago)

Logs Inspect Bind mounts Exec Files Stats

2024-09-20 20:29:05

2024-09-20 20:29:05 Hello from Docker!

2024-09-20 20:29:05 This message shows that your installation appears to be working correctly.

2024-09-20 20:29:05

2024-09-20 20:29:05 To generate this message, Docker took the following steps:

2024-09-20 20:29:05 1. The Docker client contacted the Docker daemon.

2024-09-20 20:29:05 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.

2024-09-20 20:29:05 (amd64)

2024-09-20 20:29:05 3. The Docker daemon created a new container from that image which runs the

2024-09-20 20:29:05 executable that produces the output you are currently reading.

2024-09-20 20:29:05 4. The Docker daemon streamed that output to the Docker client, which sent

# Docker Container



The screenshot shows the Docker Desktop interface with the 'Containers' tab selected. The left sidebar contains navigation options: Containers, Images, Volumes, Builds, Docker Scout, and Extensions. The main area displays the 'Containers' tab with a search bar, a toggle for 'Only show running containers', and a table of containers.

Containers [Give feedback](#)

Container CPU usage ⓘ  
*No containers are running.*

Container memory usage ⓘ  
*No containers are running.*

Search ☐ Only show running containers

| Name                                    | Image                             | Status | Port(s) | CPU (%) | Last s |
|---|-----------------------------------|--------|---------|---------|--------|
| <a href="#">determined</a><br>520c4ba24 | <a href="#">hello-world</a>       | Exited |         | N/A     | 15 min |
| <a href="#">stupefied</a><br>8e9c3637e  | <a href="#">hello-world:lates</a> | Exited |         | N/A     | 11 min |

# Container List in cmd

- `docker ps` will show running containers
- `docker ps -a` will list exited containers


```
F:\Docker>docker ps
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES


F:\Docker>docker ps -a
CONTAINER ID    IMAGE    COMMAND    CREATED    STATUS    PORTS    NAMES
8e9c3637e9b0    hello-world:latest    "/hello"    16 minutes ago    Exited (0)    5
520c4ba242e5    hello-world    "/hello"    19 minutes ago    Exited (0)    9

F:\Docker>|
```

# First Image

- Create a folder
- Add a file “Dockerfile” without extension in it
- Add instruction in the dockerfile
- Build the docker image in terminal
  - **docker build -t myfirstimage .**
  - (here the “.” refers current directory)
- To run the docker image
  - **docker run myfirstimage**

 Dockerfile X

 Dockerfile > ...

```
1  #Download base image from hub
2  #light weight linux
3  FROM alpine:latest
4
5  #Execute the command in linux
6  CMD [ "echo", "Welcome to Docker" ]
7
```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

```
PS F:\Docker\Workspace> docker build -t myfirstimage .
[+] Building 4.0s (6/6) FINISHED
=> [internal] load build definition from Dockerfile
=> == transferring dockerfile: 181B
=> [internal] load metadata for docker.io/library/alpine:latest
=> [auth] library/alpine:pull token for registry-1.docker.io
```

```
PS F:\Docker\Workspace> docker run myfirstimage
Welcome to Docker
```

**CREATE AN IMAGE WITH NODEJS**



nodejs.org/en



[Learn](#)

[About](#)

[Download](#)

[Blog](#)

[Docs](#)

[Certification](#) ↗

Discover

[TypeScript in Node.js](#) →

# Run JavaScript Everywhere

Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command

# Pull nodejs image

```
F:\Docker>docker pull node
Using default tag: latest
latest: Pulling from library/node
8cd46d290033: Pull complete
2e6afa3f266c: Pull complete
2e66a70da0be: Pull complete
1c8ff076d818: Pull complete
71a2ad2ab1a1: Pull complete
8ed09065f016: Pull complete
8cc9946ce160: Pull complete
fa87db89e50a: Pull complete
Digest: sha256:cbe2d5f94110cea9817dd8c5809d05df49b4bd1a
Status: Downloaded newer image for node:latest
docker.io/library/node:latest

F:\Docker>
```



# Run node image to start container

- To run nodejs, we need to run the node image
- If we use “docker run node”, It will started and gets exited immediately, since it is not associated with an interactive terminal to work with node.
- So we have use “docker run -it node”
- -i refers interactive
- -t refers TTY (TeleTYpe – basic Input/Output)

# Run node container

```
F:\Docker>docker run node
```

```
F:\Docker>docker run -it node
```

```
Welcome to Node.js v22.9.0.
```

```
Type ".help" for more information.
```

```
> a = 3
```

```
3
```

```
> b = 4
```

```
4
```


```
> console.log(a+b)
```


```
7
```


```
undefined
```


```
> |
```


# Running Container


 Containers

 Images

 Volumes

 Builds



 Docker Scout





 Extensions

## Containers [Give feedback](#)

Container CPU usage ⓘ Container

**0.00% / 800%** (8 CPUs available) **11.77M**

  Only show running container





| <input type="checkbox"/> | Name  | Image                | Status  | Port(s) |
|--------------------------|---|----------------------|---------|---------|
| <input type="checkbox"/> |  <a href="#">laughing_davinci</a><br>1b45645226f6   | <a href="#">node</a> | Exited  |         |
| <input type="checkbox"/> |  <a href="#">strange_mendeleev</a><br>e2aa7a227d57  | <a href="#">node</a> | Running |         |

# Stop a Container

- To stop a docker container use “**ctrl+c**”, or
- open another terminal “**docker stop <container-name/id>**”

```
PS C:\Users\ritit> docker stop strange_mendeleev
strange_mendeleev
PS C:\Users\ritit> |
```

```
F:\Docker>docker run -it node
Welcome to Node.js v22.9.0.
Type ".help" for more information.
> a = 3
3
> b = 4
4
> console.log(a+b)
7
undefined
>
F:\Docker>|
```



|   |   |                      |              |
|---|---|----------------------|--------------|
|  | <a href="#">laughing_davinci</a>  | <a href="#">node</a> | Exited       |
| 1b45645226f6  |  |                      |              |
|  | <a href="#">strange_mendeleev</a>   | <a href="#">node</a> | Exited (137) |
| e2aa7a227d57  |  |                      |              |

# Restart a Container

- “docker start <container name/id>”
  - Container will be in running status

```
F:\Docker\Workspace>docker start strange_mendeleev
strange_mendeleev

F:\Docker\Workspace>
```

|   |   |                      |         |
|---|---|----------------------|---------|
|  | <a href="#">strange_mendeleev</a>   | <a href="#">node</a> | Running |
| e2aa7a227d57  |  |                      |         |

- To run commands in running container
  - docker exec -it <container> <command>



```
F:\Docker\Workspace>docker exec -it strange_mendeleev node
Welcome to Node.js v22.9.0.
Type ".help" for more information.
> |
```

# Named Containers

```
F:\Docker\Workspace>docker run --name mynode node  
F:\Docker\Workspace>
```

|   |   |                      |        |
|---|---|----------------------|--------|
|  | <a href="#">mynode</a>  | <a href="#">node</a> | Exited |
| 9c0608c75c59  |  |                      |        |

```
F:\Docker\Workspace>docker rename mynode mynewnode  
F:\Docker\Workspace>
```

|   |   |                      |        |
|---|---|----------------------|--------|
|  | <a href="#">mynewnode</a>   | <a href="#">node</a> | Exited |
| 9c0608c75c59  |  |                      |        |

# To remove all stopped containers

- `docker container prune`

Docker Repository with official & third party images

# **PUBLISHING DOCKER IMAGES**





hub.docker.com



**New**

More Docker. Easy Access. New Streamlined Plans. Learn more.



Sign In

# Develop faster. Run anywhere.

Docker Hub is the world's easiest way to create, manage, and deliver your team's applications.



node



Trusted Content (1027)

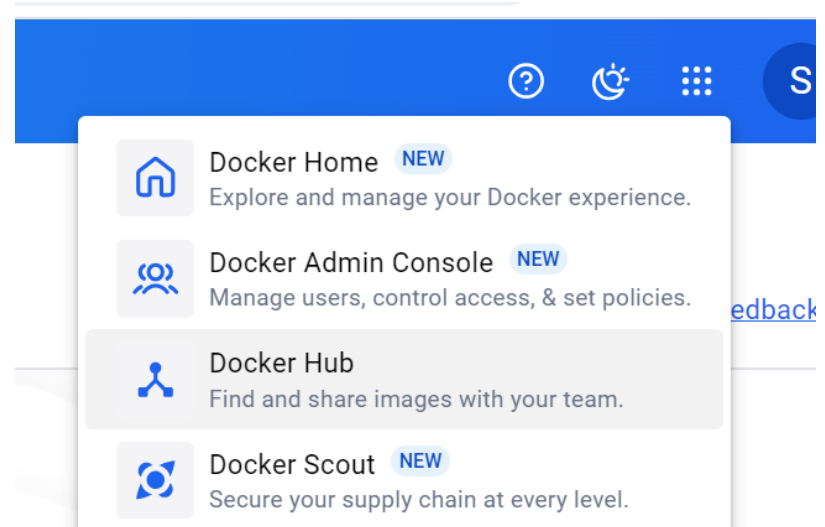
[node](#)

[kindest/node](#)

Trusted content

# Docker Hub

- Signup with docker
- Login with docker
- Click on Docker Hub
- Create a Repository



[Repositories](#) / [Create](#)

## Create repository

Namespace

spradeepagp



Repository Name \*

samples



Short description

sample repository

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

### Visibility

Using 0 of 1 private repositories. [Get more](#)



Public



Appears in Docker Hub search results



Private



Only visible to you

Cancel

Create


# Docker login from cmd

```
PS F:\Docker\Workspace\usecase1> docker login
Log in with your Docker ID or email address to push and
cker ID, head over to https://hub.docker.com/ to create
You can log in with your password or a Personal Access T
security and is required for organizations using SSO. L
ns/


Username: spradeepagp
Password:
Login Succeeded
PS F:\Docker\Workspace\usecase1> █
```

# Push image to repository

[myfirstimage](#)

feb84fcfe749 

latest

spradeepagp/samples 

Created 5 minutes ago

Login Succeeded

```
PS F:\Docker\Workspace\usecase1> docker tag myfirstimage spradeepagp/samples:latest
```

```
PS F:\Docker\Workspace\usecase1> docker push spradeepagp/samples:latest
```



The push refers to repository [docker.io/spradeepagp/samples]

63ca1fbb43ae: Mounted from library/alpine

latest: digest: sha256:19a0d6dcabda818b738aa8b644991be0107ca876c64b956c19cee520c1b470b









```
PS F:\Docker\Workspace\usecase1> █
```

This repository contains 1 tag(s).

| Tag  | OS  | Type  | Pulled        |
|--|---|-------|---------------|
|  latest |  | Image | 3 minutes ago |

[See all](#)

# Delete images from docker desktop

| <input type="text" value="Search"/>   |    |  | <div>Delete</div>      | Space to be reclaimed 0 B |      |
|---|---|---|------------------------|---------------------------|------|
|  | Name  | Tag   | Status                 | Created                   | Size |
| <input type="checkbox"/>  | <a href="#">node</a><br>2ef13a9c33b0                 | latest  | <a href="#">In use</a> | 4 days ago                | 1.1  |
| <input checked="" type="checkbox"/>   | <a href="#">myfirstimage</a><br>feb84fcfe749         | latest  | <a href="#">In use</a> | 15 days ago               | 7.79 |
| <input checked="" type="checkbox"/>   | <a href="#">spradeepagp/samples</a><br>feb84fcfe749  | latest  | <a href="#">In use</a> | 15 days ago               | 7.79 |
| <input type="checkbox"/>  | <a href="#">alpine</a><br>91ef0af61f39             | latest  | <a href="#">In use</a> | 15 days ago               | 7.79 |
| <input type="checkbox"/>  | <a href="#">hello-world</a><br>d2c94e258dcb        | latest  | Unused                 | 1 year ago                | 13.2 |

# Pull the image

| <input type="checkbox"/> | Name  | Tag    | Status                 | Created     | Size     | Actions |  |
|--------------------------|---|--------|------------------------|-------------|----------|---------|--|
| <input type="checkbox"/> | <a href="#">node</a><br>2ef13a9c33b0        | latest | <a href="#">In use</a> | 4 days ago  | 1.11 GB  |         |  |
| <input type="checkbox"/> | <a href="#">alpine</a><br>91ef0af61f39      | latest | <a href="#">In use</a> | 15 days ago | 7.79 MB  |         |  |
| <input type="checkbox"/> | <a href="#">hello-world</a><br>d2c94e258dcb | latest | Unused                 | 1 year ago  | 13.25 KB |         |  |

```
PS F:\Docker\Workspace\usecase1> docker pull spradeepagp/samples:latest
latest: Pulling from spradeepagp/samples
```

|                          |   |        |                        |             |          |  |  |
|--------------------------|---|--------|------------------------|-------------|----------|--|--|
| <input type="checkbox"/> | <a href="#">alpine</a><br>91ef0af61f39              | latest | <a href="#">In use</a> | 15 days ago | 7.79 MB  |  |  |
| <input type="checkbox"/> | <a href="#">spradeepagp/samples</a><br>feb84fcfe749 | latest | Unused                 | 15 days ago | 7.79 MB  |  |  |
| <input type="checkbox"/> | <a href="#">hello-world</a><br>d2c94e258dcb         | latest | Unused                 | 1 year ago  | 13.25 KB |  |  |

# Run the image

```
PS F:\Docker\Workspace\usecase1> docker run -it spradeepagp/samples  
Welcome to Docker  
PS F:\Docker\Workspace\usecase1> █
```



**SPRING BOOT + DOCKERFILE**

# Dockerfile

A **Dockerfile** is a text file that contains instructions to **build a Docker image** — like a recipe that tells Docker how to create an environment for your application.

Think of it like:

*"Hey Docker, here's what you need to do:  
Start from this base image,  
install some things,  
copy my code in,  
and then run my app."*

# Dockerfile

A Dockerfile is a step-by-step script that tells Docker how to build an image. For a **Java app**, here are the **essential components**:

1. Base Image
2. Set Working Directory (Optional but Recommended)
3. Copy the Compiled Java App (JAR File)
4. Define How to Run the App
5. (Optional) Expose Port

## 1. Base Image

This is the environment that your Java app needs to run.

**FROM openjdk:17-jdk-slim**

This line means: "Start from a lightweight Linux image that already has Java 17 installed."

## 2. Set Working Directory (Optional but Recommended)

**WORKDIR /app**

This sets /app as the folder where your app will live inside the container.

### 3. Copy the Compiled Java App (JAR File)

You copy your .jar file into the container. Assuming you already built it using Maven:

```
ARG JAR_FILE=target/*.jar
```

```
COPY ${JAR_FILE} app.jar
```

Or, if you want to hardcode it:

```
COPY target/myapp.jar app.jar
```

### 4. Define How to Run the App

This tells Docker what command to run when the container starts:

```
ENTRYPOINT ["java", "-jar", "app.jar"]
```

Or if you're using CMD:

```
CMD ["java", "-jar", "app.jar"]
```

Both work, but ENTRYPOINT is more consistent for Docker best practices.

## 5. (Optional) Expose Port

- If your Java app listens on a specific port (e.g., 8080 for Spring Boot):
- **EXPOSE 8080**
- Note: This doesn't actually publish the port — it's just documentation. You still need to use -p when running the container:
- **docker run -p 8080:8080 my-app**

# Dockerfile for SpringBoot

# Use Java 17 as base image

FROM openjdk:17-jdk-slim

# Set the working directory inside the container

WORKDIR /app

# Copy the JAR file built by Maven

ARG JAR\_FILE=target/\*.jar

COPY \${JAR\_FILE} app.jar

# Optional: Expose port used by Spring Boot

EXPOSE 8080

# Run the app

ENTRYPOINT ["java", "-jar", "app.jar"]

# Spring boot + Docker

- Create a spring boot application
- Create a jar
  - with Run As -> Maven Install
- Create a Dockerfile
  - in the project root dir
  - with the following content.

```
FROM openjdk:17-jdk-slim
WORKDIR /app
COPY target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```



Package Explorer ×

- WebDemo [boot]
  - src/main/java
  - src/main/resources
  - src/test/java
  - JRE System Library [JavaSE-17]
  - Maven Dependencies
    - target/generated-sources/annot
    - target/generated-test-sources/te
  - src
- target
  - generated-sources
  - generated-test-sources
  - maven-archiver
  - maven-status
  - surefire-reports
  - SBDemoApp1.0.jar
  - SBDemoApp1.0.jar.original
- DockerFile
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml

DockerFile ×

```
1 # Use a base image with Java
2 FROM openjdk:17-jdk-slim
3
4 # Set the working directory
5 WORKDIR /app
6
7 # Copy the JAR file into the container
8 COPY target/*.jar app.jar
9
10 # Expose the port the app runs on
11 EXPOSE 8080
12
13 # Run the application
14 ENTRYPOINT ["java", "-jar", "app.jar"]
15
```

# Workflow with Maven

Open a terminal in the project root location

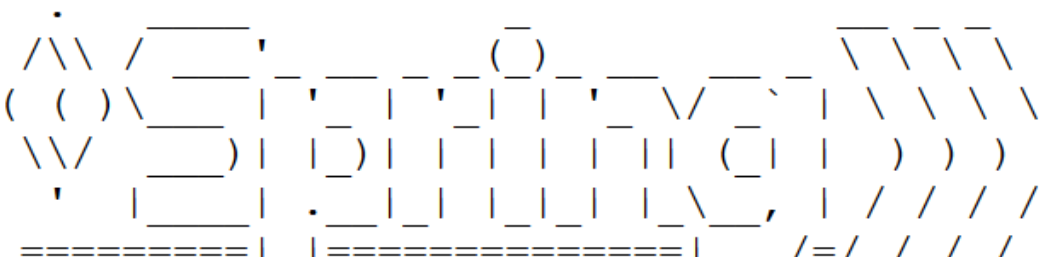
- Package your app:
  - `mvn clean package`
- Build Docker image:
  - `docker build -t spradeepagp/sbdemoapp:latest .`
- Run the container:
  - `docker run -p 8080:8080 spradeepagp/sbdemoapp:latest`
- Access the api:
  - Visit <http://localhost:8080/hello>

# Docker Image Push

- Log in to Docker Hub
  - `docker login`
- Push the Docker Image to Docker Hub
  - `docker push spradeepagp/sbdemoapp:latest`


# Open terminal and type below

```
F:\WebDemo>docker build -t spradeepagp/sbdemoapp:latest .  
[+] Building 27.6s (9/9) FINISHED  
....  
  
F:\WebDemo>docker login  
Authenticating with existing credentials...  
Login Succeeded  
  
F:\WebDemo>docker push spradeepagp/sbdemoapp:latest  
The push refers to repository [docker.io/spradeepagp/sbdemoapp]  
d56ac6ab8fe9: Pushed  
....  
  
F:\WebDemo>docker run -p 8080:8080 spradeepagp/sbdemoapp:latest
```

A large ASCII art logo for "spradeep". It features a central stylized face with a wide smile, composed of various symbols like backslashes, forward slashes, underscores, and parentheses. The name "spradeep" is written in a decorative font above the face. Below the face are several horizontal lines of varying lengths made of equals signs and other symbols, creating a base for the logo.

# Image in local and Hub

|                          |   |        |                        |               |
|--------------------------|---|--------|------------------------|---------------|
| <input type="checkbox"/> | <a href="#">spradeepagp/sbdemoapp</a><br>785b1a76f26a | latest | <a href="#">In use</a> | 12 minutes ag |
| <input type="checkbox"/> | <a href="#">node</a><br>2ef13a9c33b0                  | latest | <a href="#">In use</a> | 4 days ago    |

 **docker hub**

Explore Repositories Organizations Usage

spradeepagp

Search by repository name

All Content

spradeepagp / **sbdemoapp**

Contains: Image • Last pushed: 17 minutes ago

spradeepagp / **samples**

Contains: Image • Last pushed: about 2 hours ago

**SPRING BOOT + DOCKER COMPOSE**

# Docker Compose

**Docker Compose** lets you define and run **multi-container** Docker applications using a **single YAML file**.

- Instead of running multiple docker run commands manually, you just define your whole app in a file, then run:

- **docker-compose up**

- **Typical Java App Use Case**

- Let's say you have:
- A **Spring Boot** app (Java) that runs in one container
- A **MySQL** database in another container
- You can use **Docker Compose** to run both together, networked automatically.

# Docker-compose.yml

(Spring Boot + MySQL)

- version: '3.8'
- 
- services:
- app:
- build: .
- ports:
- - "8080:8080"
- environment:
- SPRING\_DATASOURCE\_URL: jdbc:mysql://mysql:3306/testdb
- SPRING\_DATASOURCE\_USERNAME: root
- SPRING\_DATASOURCE\_PASSWORD: root
- depends\_on:
- - mysql
- 
- mysql:
- image: mysql:8
- ports:
- - "3306:3306"
- environment:
- MYSQL\_ROOT\_PASSWORD: root
- MYSQL\_DATABASE: testdb



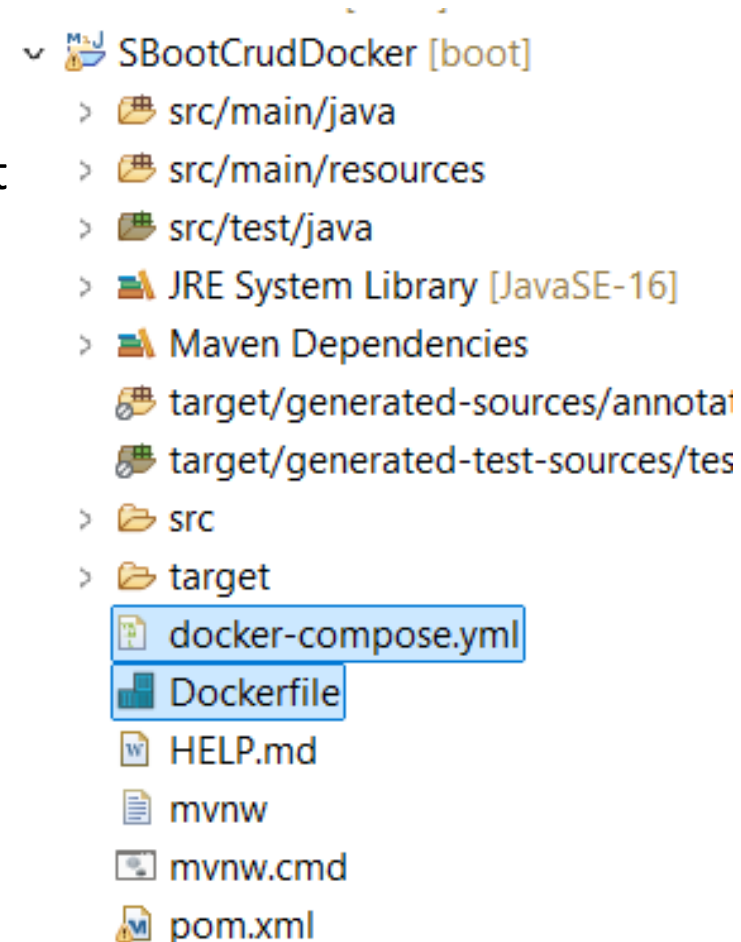
# Breakdown: What Each Section Means

| Section               | Meaning   |
|-----------------------|---|
| <b>version: '3.8'</b> | Compose file format version                                   |
| <b>services:</b>      | List of containers to run                                     |
| <b>app:</b>           | Your Java/Spring Boot application                             |
| <b>build: .</b>       | Build image from Dockerfile in the current folder             |
| <b>ports:</b>         | Map container port to local machine (8080:8080)               |
| <b>environment:</b>   | Inject environment variables (used in application.properties) |
| <b>depends_on:</b>    | Start MySQL before Spring Boot                                |
| <b>mysql:</b>         | MySQL service using official image                            |
| <b>image:</b>         | Download this Docker image from Docker Hub                    |
| <b>MYSQL_*</b>        | MySQL container settings (passwords, database name, etc.)     |

# Spring Boot + MySQL

## Implementation Step by Step:

- Create a spring boot mysql CRUD application
- Create a Dockerfile (No extension) in the root directory
- Create a docker-compose.yml file in the root directory
- Open a terminal in the root directory
  - Create jar file
    - `mvn clean package`
  - Run everything
    - `Docker-compose up --build`
- Press 'v' key to view container in docker desktop



# Pom.xml

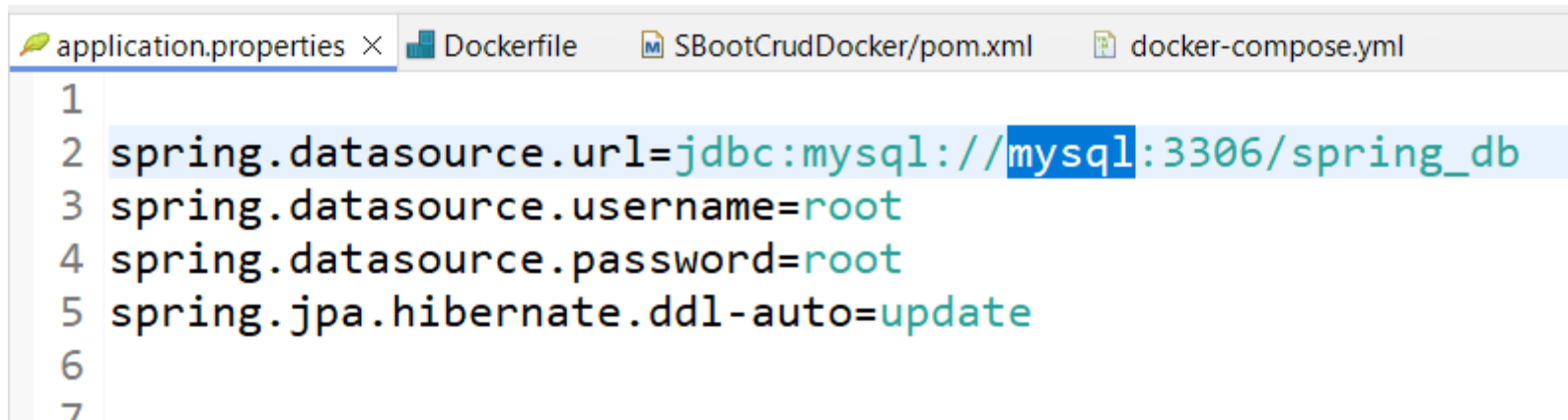
```
SBootCrudDocker/pom.xml × application.properties Dockerfile docker-compose.yml
39         </dependency>
40     </dependencies>
41
42     <build>
43         <finalName>springboot-crud</finalName>
44         <plugins>
45             <plugin>
46                 <groupId>org.springframework.boot</groupId>
47                 <artifactId>spring-boot-maven-plugin</artifactId>
48             </plugin>
49         </plugins>
50     </build>
51
52 </project>
```

# Dockerfile

```
SBootCrudDocker/pom.xml Dockerfile × application.properties docker-compose.yr
1 # Use an official JDK image
2 FROM openjdk:17-jdk-slim
3
4 # Set the working directory
5 WORKDIR /app
6
7 # Copy built jar file
8 COPY target/springboot-crud.jar app.jar
9
10 # Expose the port Spring Boot runs on
11 EXPOSE 8080
12
13 # Run the application
14 ENTRYPOINT ["java", "-jar", "app.jar"]
15
```

# Properties file

- Replace localhost with mysql (container name)



The screenshot shows an IDE window with four tabs: 'application.properties', 'Dockerfile', 'SBootCrudDocker/pom.xml', and 'docker-compose.yml'. The 'application.properties' tab is active, displaying the following content:

```
1  
2 spring.datasource.url=jdbc:mysql://mysql:3306/spring_db  
3 spring.datasource.username=root  
4 spring.datasource.password=root  
5 spring.jpa.hibernate.ddl-auto=update  
6  
7
```


The text on line 2, 'spring.datasource.url=jdbc:mysql://mysql:3306/spring\_db', is highlighted in blue. The word 'mysql' in the URL is also highlighted in blue, indicating the container name used for the database connection.

```




1  docker-compose.yml - The Compose specification establishes a
2  version: '3.8'
3  services:
4    mysql:
5      image: mysql:8
6      container_name: mysql
7      ports:
8        - "3306:3306"
9      environment:
10        MYSQL_DATABASE: mydb
11        MYSQL_ROOT_PASSWORD: root
12      volumes:
13        - mysql_data:/var/lib/mysql
14
15    springboot-app:
16      build: .
17      container_name: springboot-crud
18      depends_on:
19        - mysql
20      ports:
21        - "8080:8080"
22      environment:
23        SPRING_DATASOURCE_URL: jdbc:mysql://mysql:3306/mydb
24        SPRING_DATASOURCE_USERNAME: root
25        SPRING_DATASOURCE_PASSWORD: root
26
27    volumes:
28      mysql_data:
29
```





- Open a terminal in the root directory
  - Create jar file
    - `mvn clean package -DskipTests`
  - Run everything
    - `Docker-compose up --build`
- Press 'v' key to view container in docker desktop
- Run the application container
- Access the endpoints in the postman

# Open and Run in DockerDesktop

**sbootcruddocker**  
G:\EY Training\SpringBootWS\SBootCrudDocker

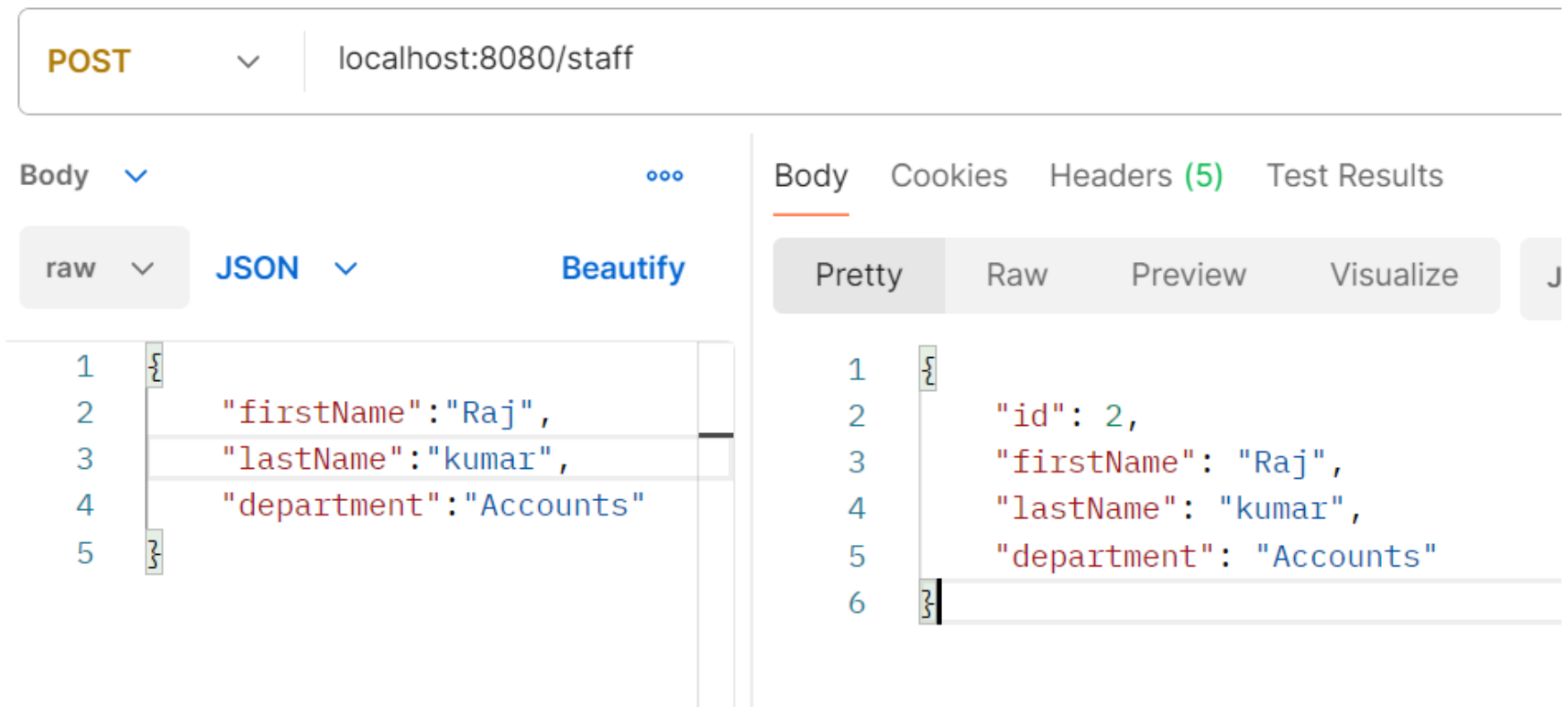
View configurations



|  |   |   |   |  |
|--|---|---|---|--|
|  <b>mysql</b><br><a href="#">mysql:8</a><br><a href="#">3306:3306</a>                         |  | : |  | <pre>xxer.hikari.pool.HikariPool      : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImp l@599f571f 2025-07-21 23:17:20 springboot-crud   2025-07-21T17:47:20.887Z  INFO 1 --- [          main] com.za xxer.hikari.HikariDataSource      : HikariPool-1 - Start completed. 2025-07-21 23:17:21 springboot-crud   2025-07-21T17:47:21.232Z  INFO 1 --- [          main] o.h.b. i.BytecodeProviderInitiator      : HHH000021: Bytecode provider name : bytebuddy 2025-07-21 23:17:22 springboot-crud   2025-07-21T17:47:22.205Z  INFO 1 --- [          main] o.h.e. t.j.p.i.JtaPlatformInitiator      : HHH000490: Using JtaPlatform implementation: [org.hibernate.eng ine.transaction.jta.platform.internal.NoJtaPlatform] 2025-07-21 23:17:22 springboot-crud   2025-07-21T17:47:22.420Z  INFO 1 --- [          main] j.Loca lContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'defa ult' 2025-07-21 23:17:22 springboot-crud   2025-07-21T17:47:22.793Z  WARN 1 --- [          main] JpaBas eConfiguration\$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, datab ase queries may be performed during view rendering. Explicitly configure spring.jpa.open-in-view to disable this warning 2025-07-21 23:17:23 springboot-crud   2025-07-21T17:47:23.257Z  INFO 1 --- [          main] o.s.b. w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path '' 2025-07-21 23:17:23 springboot-crud   2025-07-21T17:47:23.279Z  INFO 1 --- [          main] com.ri t.SbDemo1Application             : Started SbDemo1Application in 6.335 seconds (process running fo r 7.032) 2025-07-21 23:17:30 springboot-crud   2025-07-21T17:47:30.167Z  INFO 1 --- [nio-8080-exec-2] o.a.c. c.C.[Tomcat].[localhost].[/]      : Initializing Spring DispatcherServlet 'dispatcherServlet' 2025-07-21 23:17:30 springboot-crud   2025-07-21T17:47:30.168Z  INFO 1 --- [nio-8080-exec-2] o.s.we b.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet' 2025-07-21 23:17:30 springboot-crud   2025-07-21T17:47:30.172Z  INFO 1 --- [nio-8080-exec-2] o.s.we b.servlet.DispatcherServlet        : Completed initialization in 3 ms</pre> |
|  <b>springboot-crud</b><br><a href="#">sbootcruddocker-sprii</a><br><a href="#">8080:8080</a> |  | : |  |  |



# Access it with postman



**THANK YOU**