05 What is Docker?

Docker is a tool that lets you package your application and all its dependencies into a **container** so it runs the same everywhere.

Let’s imagine you want to run a Spark application.

What do you need to get Spark working on your machine?

Of course, you need to write Spark code using PySpark or Spark SQL.

But the code alone is not enough.

To run Spark, you also need:

- Java
- Spark binaries
- Python
- correct versions of libraries
- environment variables
- proper configuration

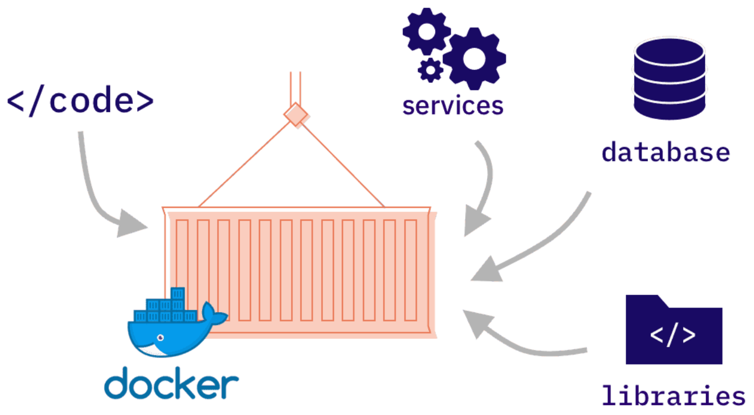
Setting all this up manually is different on Windows, Mac, and Linux, and often leads to “it works on my machine” problems.

Docker solves this by allowing us to bundle **Spark, Java, Python, and all required dependencies** into a single package called a **container**.

When we run this container, Spark runs in a **clean, isolated, and consistent environment**, the same way for everyone.

This is why we are using Docker for Spark setup:

not to learn Docker, but to make Spark setup simple, repeatable, and OS independent.

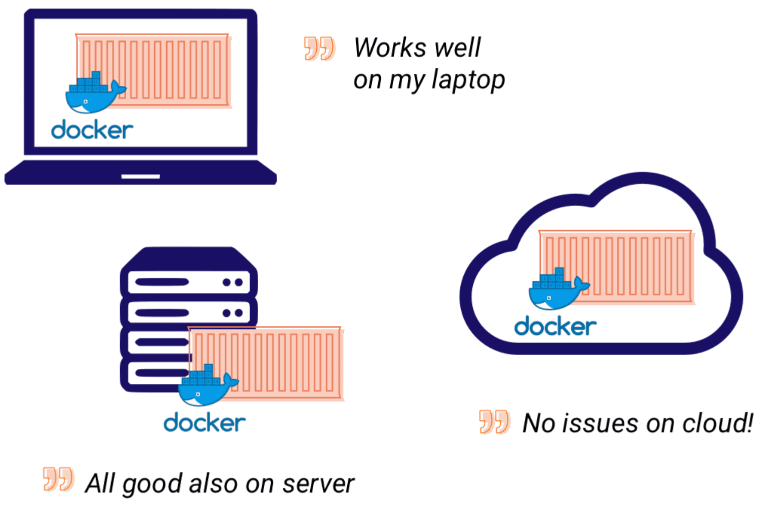


Think of a container like a small, lightweight box that has:

- your code
- your libraries
- your runtime
- your system dependencies

Everything packaged together.

So whether you run it on your laptop, a server, or the cloud, it behaves exactly the same.



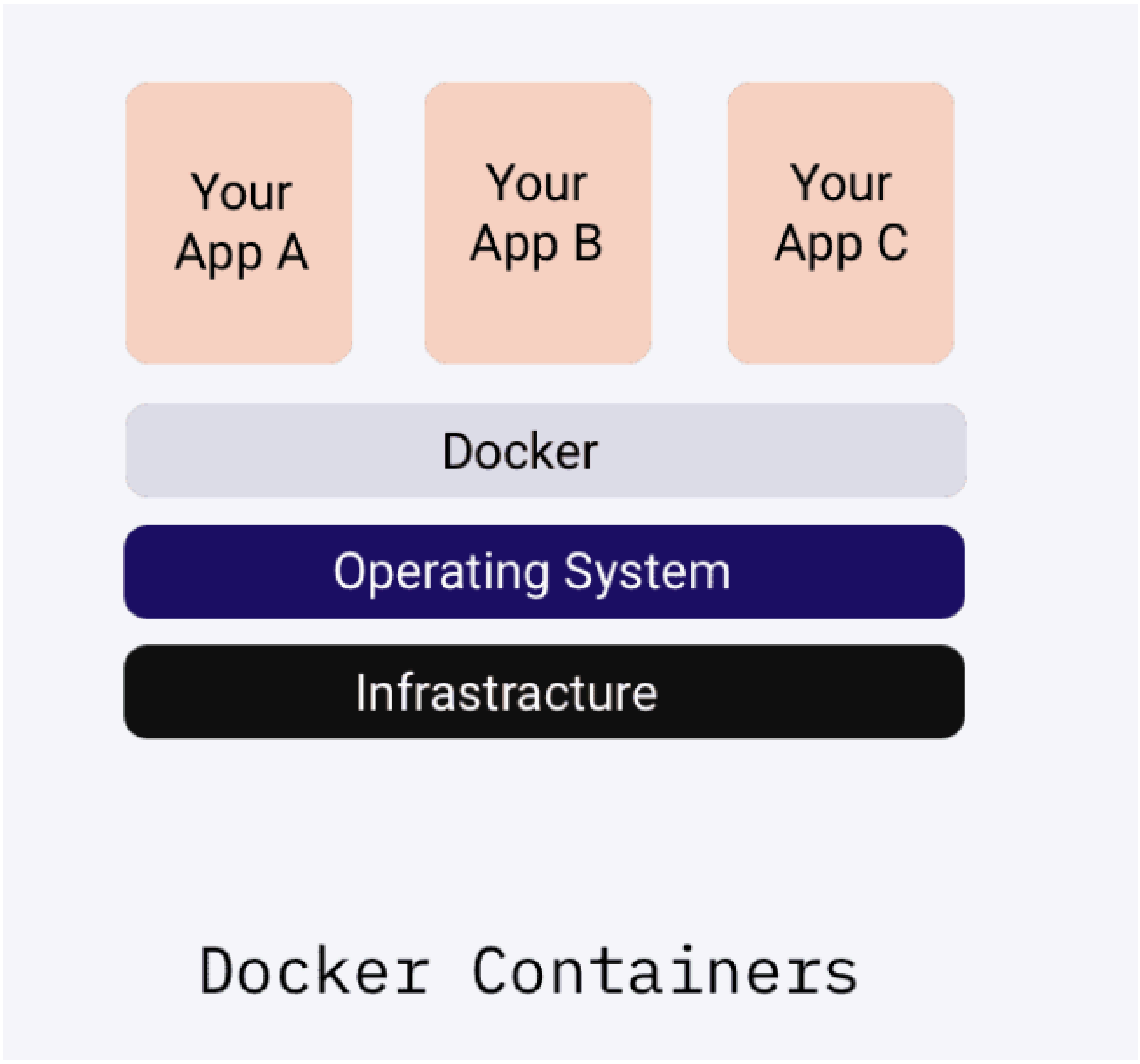
Traditionally:

- everyone installs software differently

- OS issues, version mismatch, “it works on my machine”

Docker solves this by:

- packaging software + dependencies together
- making setup **consistent across machines**



Key Docker concepts

- **Image**

  Think of an image as a **recipe or blueprint**.

It contains everything needed to run an application, but it is **not running yet**.

Example:

Python image, Ubuntu image, Spark image.

- **Container**

  A container is the **running version of an image**.

  When you start an image, it becomes a container.

  👉 Image = class

  👉 Container = object (instance)

- **Dockerfile**

  A Dockerfile is a **step-by-step instruction file** that tells Docker:

  - which base image to use
  - what software to install
  - how to configure the environment

  Docker reads this file and creates an image.

- **Docker Hub**

  Docker Hub is like **GitHub for Docker images**.

  It is a public place where people share ready-made images.

  Instead of building everything from scratch, you can simply download an image and use it.

One-line summary

> Dockerfile builds an image,
>
> image runs as a container,
>
> Docker Hub stores images.

In one line

Docker = Package your app once, run it anywhere.