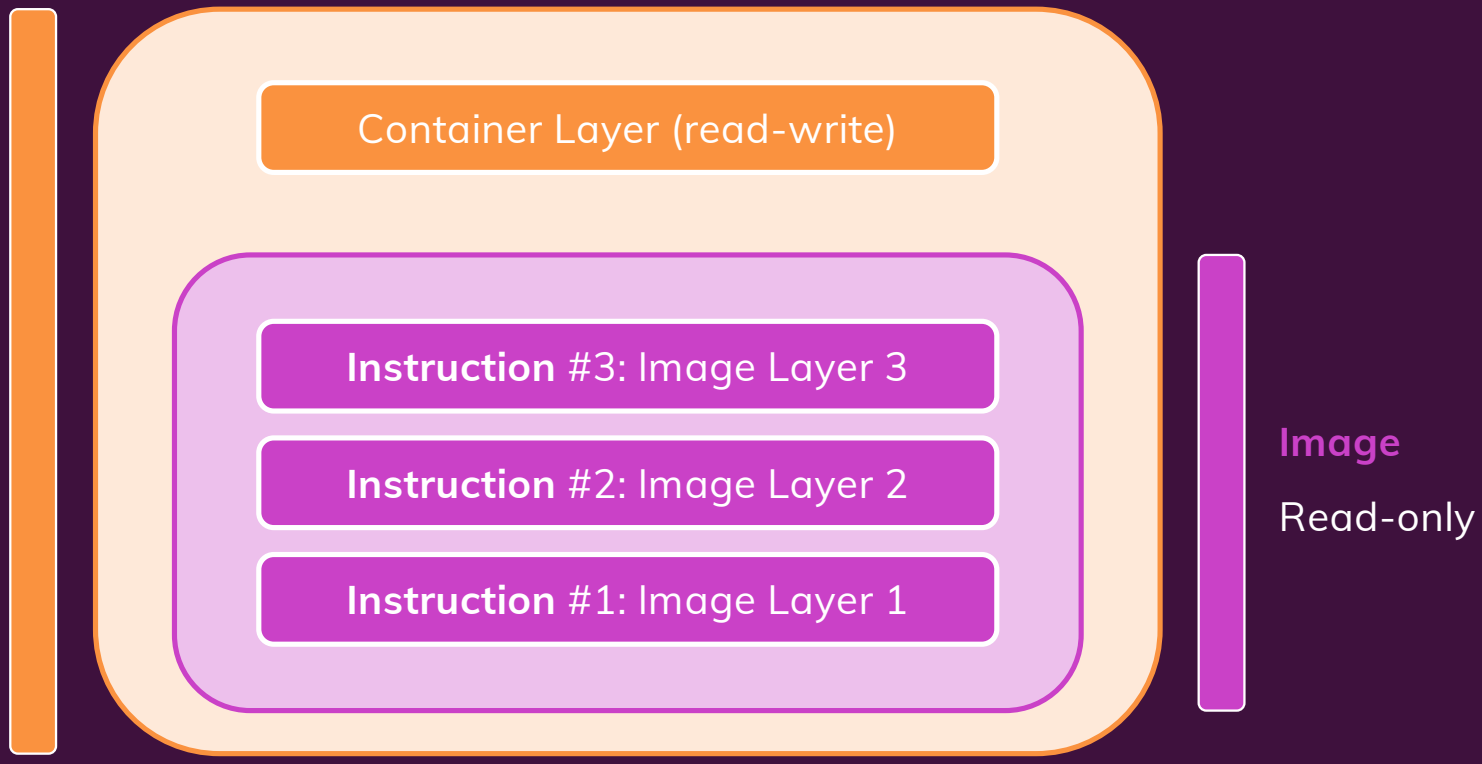
ACADEMIND

Data?

Application *(Code + Environment)*	**Temporary App Data** *(e.g. entered user input)*	**Permanent App Data** *(e.g. user accounts)*
Written & provided by you (= the developer)	Fetched / Produced in running container	Fetched / Produced in running container
Added to image and container in build phase	Stored in memory or temporary files	Stored in files or a database
"Fixed": Can't be changed once image is built	Dynamic and changing, but cleared regularly	Must not be lost if container stops / restarts
Read-only, hence stored in Images	**Read + write, temporary, hence stored in Containers**	**Read + write, permanent, stored with Containers & Volumes**

ACADEMIND

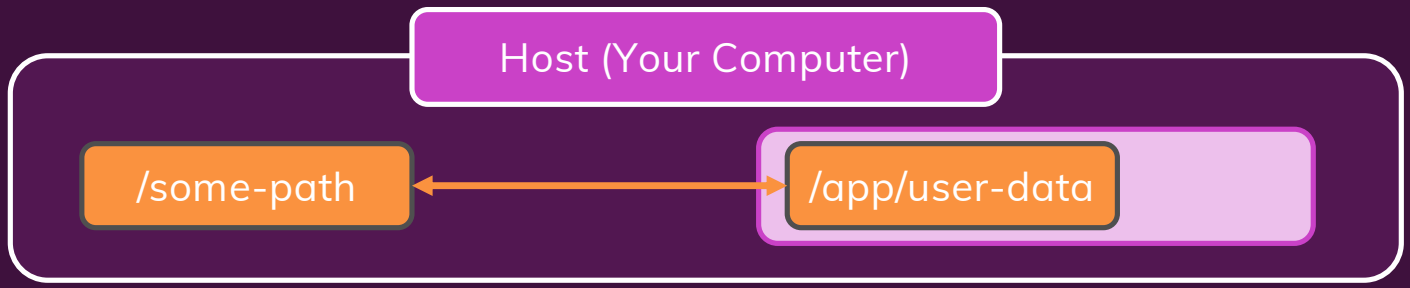
A Container Is Based On An Image



ACADEMIND

Understanding Volumes

Volumes are **folders on your host machine** hard drive which are **mounted** (“made available”, mapped) **into containers**



Volumes **persist if a container shuts down**. If a container (re-)starts and mounts a volume, any data inside of that volume is **available in the container**.

A container **can write** data into a volume **and read** data from it.

ACADEMIND

Two Types of External Data Storages

Volumes
(Managed by Docker)

Bind Mounts
(Managed by you)

Anonymous Volumes

Named Volumes

Docker sets up a folder / path on your host machine, exact location is unknown to you (= dev). Managed via `docker volume` commands.

You define a folder / path on your host machine.

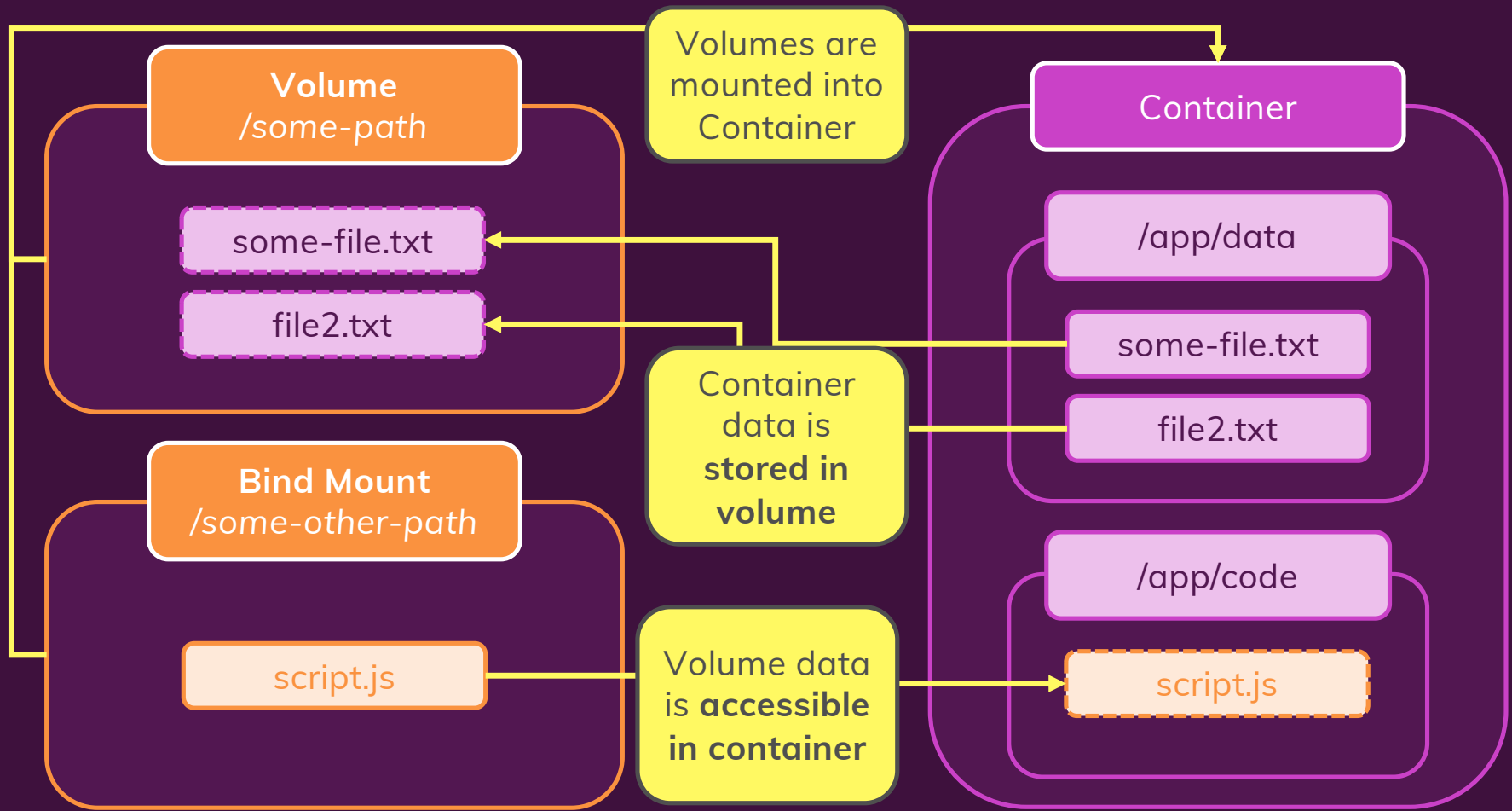
A defined path in the container is mapped to the created volume / mount.
e.g. /some-path on your hosting machine is mapped to /app/data

Great for data which should be persistent but which you don't need to edit directly.

Great for persistent, editable (by you) data *(e.g. source code)*.

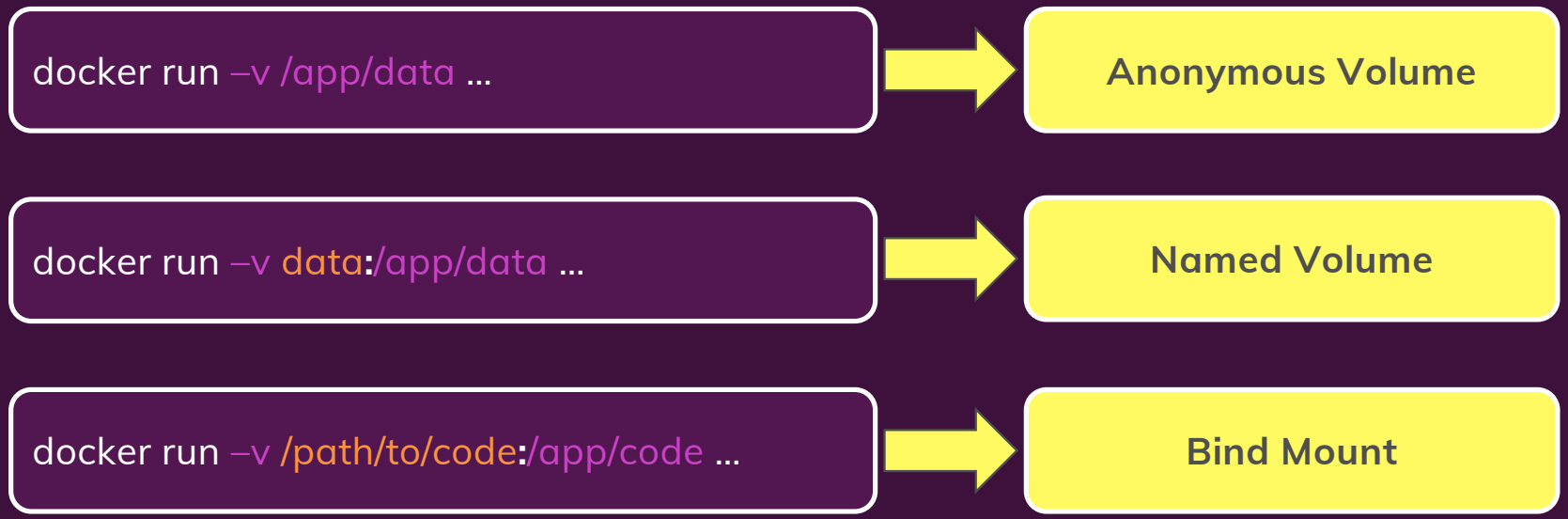
ACADEMIND

Understanding Container / Volume Interaction



ACADEMIND

Volumes & Bind Mounts – Quick Overview



ACADEMIND

Volumes – Comparison

Anonymous Volumes	Named Volumes	Bind Mounts
Created specifically for a single container	Created in general – not tied to any specific container	Location on host file system, not tied to any specific container
Survives container shutdown / restart unless --rm is used	Survives container shutdown / restart – removal via Docker CLI	Survives container shutdown / restart – removal on host fs
Can not be shared across containers	Can be shared across containers	Can be shared across containers
Since it's anonymous, it can't be re-used (even on same image)	Can be re-used for same container (across restarts)	Can be re-used for same container (across restarts)

ACADEMIND

ARGuments & ENVironment Variables

Docker supports build-time ARGuments and runtime ENVironment variables

ARG

Available inside of Dockerfile, NOT accessible in CMD or any application code

Set on image build (**`docker build`**) via `--build-arg`

ENV

Available inside of Dockerfile & in application code

Set via `ENV` in Dockerfile or via `--env` on **`docker run`**

ACADEMIND

Module Summary

Containers can read + write data. **Volumes** can help with data storage, **Bind Mounts** can help with direct container interaction.

Containers can read + write data, but written **data is lost** if the container is removed

Volumes are folders on the host machine, managed by Docker, which are mounted into the Container

Named Volumes survive container removal and can therefore be used to store persistent data

Anonymous Volumes are attached to a container – they can be used to save (temporary) data inside the container

Bind Mounts are folders on the host machine which are specified by the user and mounted into containers – like **Named Volumes**

Build ARGuments and **Runtime ENVironment** variables can be used to make images and containers more **dynamic / configurable**

ACADEMIND

Read-Only, Read-Write & Volumes

Images	Containers
Read-only	**Read & Write**
Once created, you need to re-build them to change something	A running container can store data (e.g. incoming user data)
Application data (e.g. user data) is NOT stored in images	But: Data is lost when a container stops

Solution for persistent data: **Volumes**