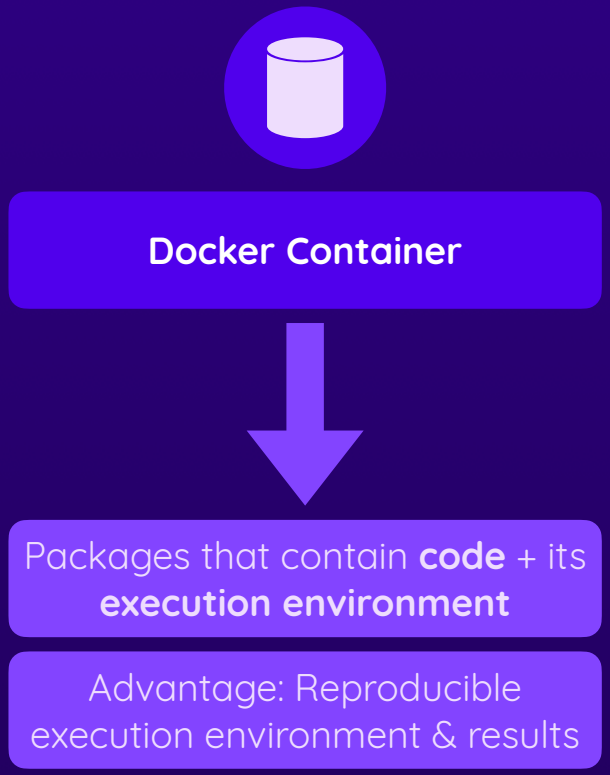
ACADEMIND

Using Containers

Utilizing Docker Containers

- Containers - A Re-Introduction
- Running Jobs in Containers
- Using Service Containers

What Are Containers?



Containers & GitHub Actions

Docker Container

Packages that contain **code** + its **execution environment**

Advantage: Reproducible execution environment & results

still runs on

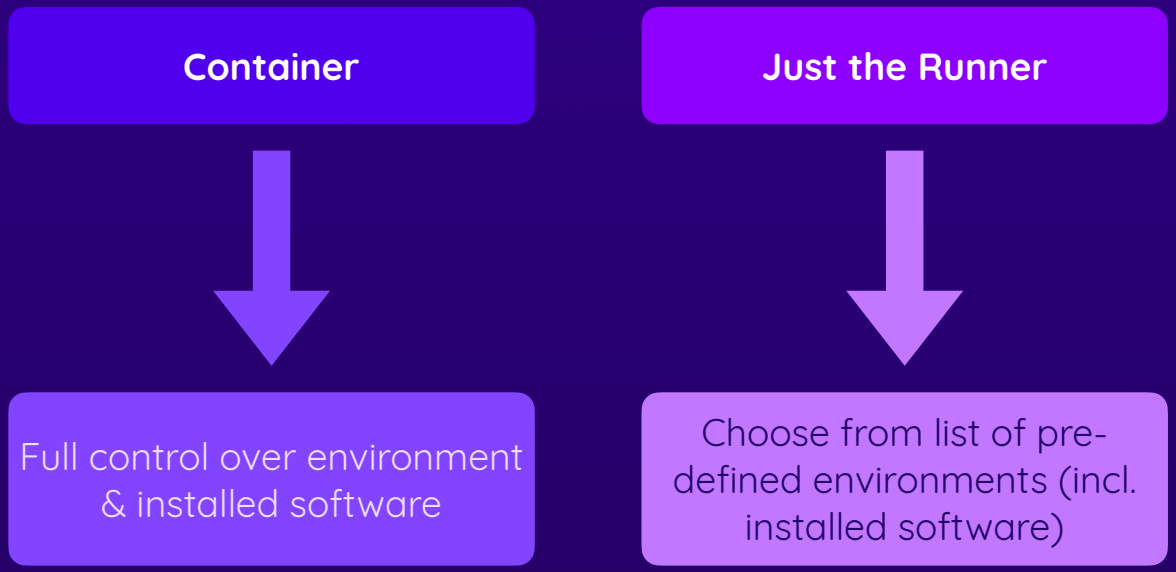
GitHub Actions Runners

Your containerized Job is hosted by the Runner

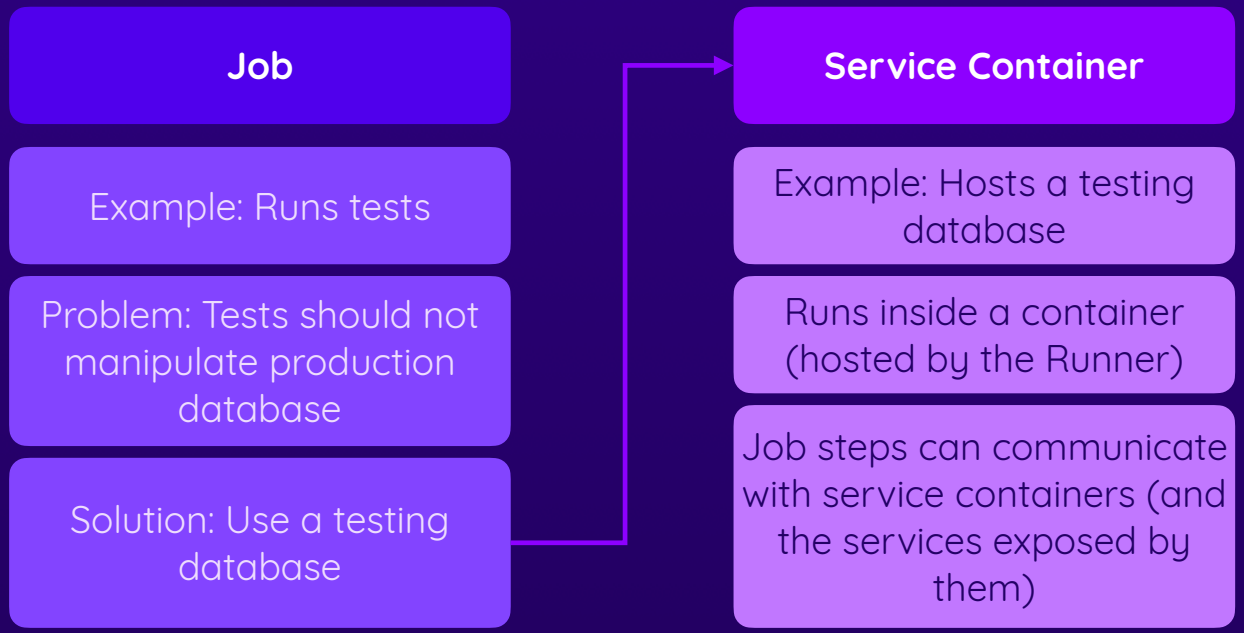
Steps execute inside the container

You can also create **Services**: Utility containers used by your Steps (Example: Testing Database)

Why Use Containers?



Using Service Containers ("Services")



Module Summary

Containers	Containers for Jobs	Service Containers
Packages of code + execution environment	You can run Jobs in pre-defined environments	Extra services can be used by Steps in Jobs
Great for creating re-usable execution packages & ensuring consistency	Build your own container images or use public images	Example: Locally running, isolated testing database
Example: Same environment for testing + production	Great for Jobs that need extra tools or lots of customization	Based on custom images or public / community images

ACADEMIND

Building Custom Actions

Beyond Shell Commands & The Marketplace

- What & Why?
- Different Types of Custom Actions
- Building & Using Custom Actions

ACADEMIND

Why Custom Actions?

Simplify Workflow Steps

Instead of writing multiple (possibly very complex) Step definitions, you can build and use a single custom Action

Multiple Steps can be grouped into a single custom Action

No Existing (Community) Action

Existing, public Actions might not solve the specific problem you have in your Workflow

Custom Actions can contain any logic you need to solve your specific Workflow problems

ACADEMIND

Different Types of Custom Actions

JavaScript Actions

Execute a JavaScript file

Use JavaScript (NodeJS) + any packages of your choice

Pretty straightforward (if you know JavaScript)

Docker Actions

Create a Dockerfile with your required configuration

Perform any task(s) of your choice with any language

Lots of flexibility but requires Docker knowledge

Composite Actions

Combine multiple Workflow Steps in one single Action

Combine `run` (commands) and `uses` (Actions)

Allows for reusing shared Steps (without extra skills)

ACADEMIND

Module Summary

What & Why?	Composite Actions	JavaScript & Docker Actions
Simplify Workflows & avoid repeated Steps	Create custom Actions by combining multiple Steps	Write Action logic in JavaScript (NodeJS) with @actions/toolkit
Implement logic that solves a problem not solved by any publicly available Action	Composite Actions are like “Workflow Excerpts”	Alternatively: Create your own Action environment with Docker
Create & share Actions with the Community	Use Actions (via `uses`) and Commands (via `run`) as needed	Either way: Use inputs, set outputs and perform any logic