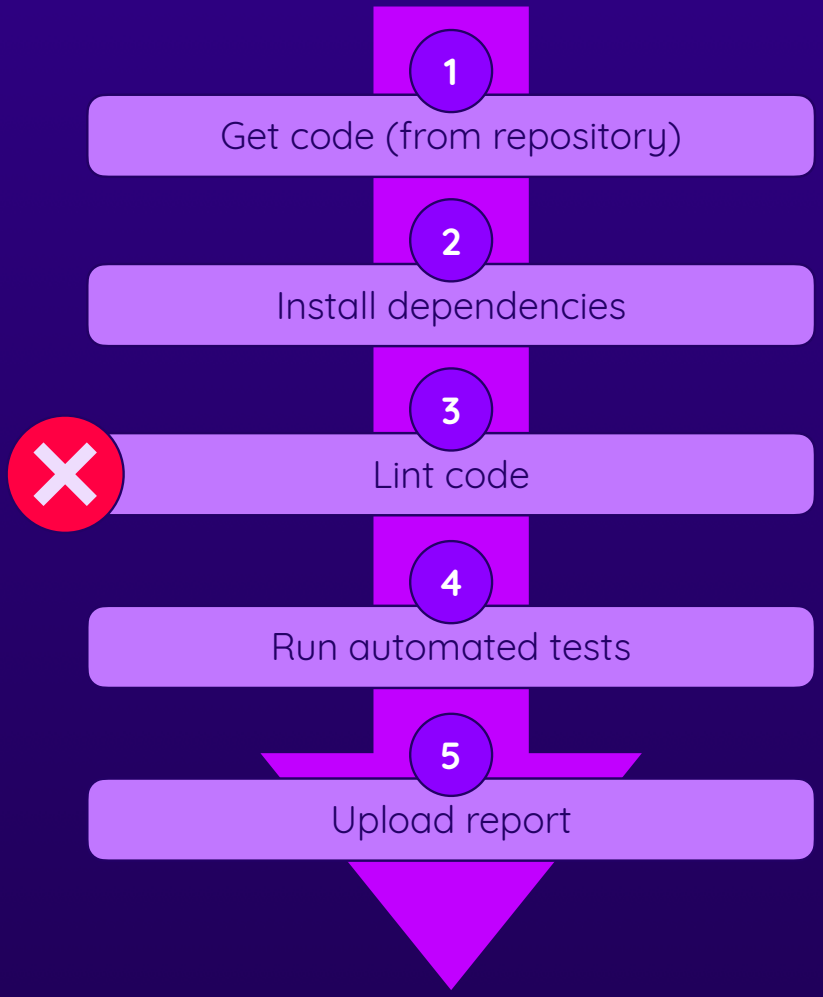
ACADEMIND

Controlling Execution Flow

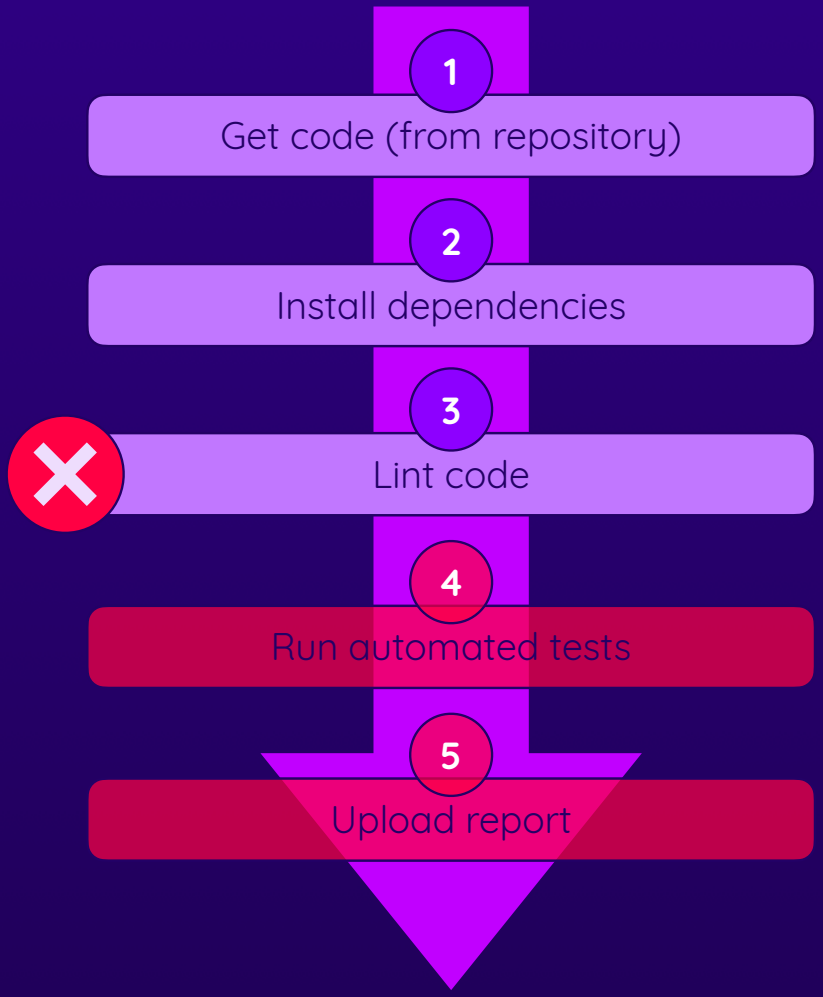
Beyond Step-By-Step Flows

- Running Jobs & Steps Conditionally
- Running Jobs with a Matrix
- Re-Using Workflows

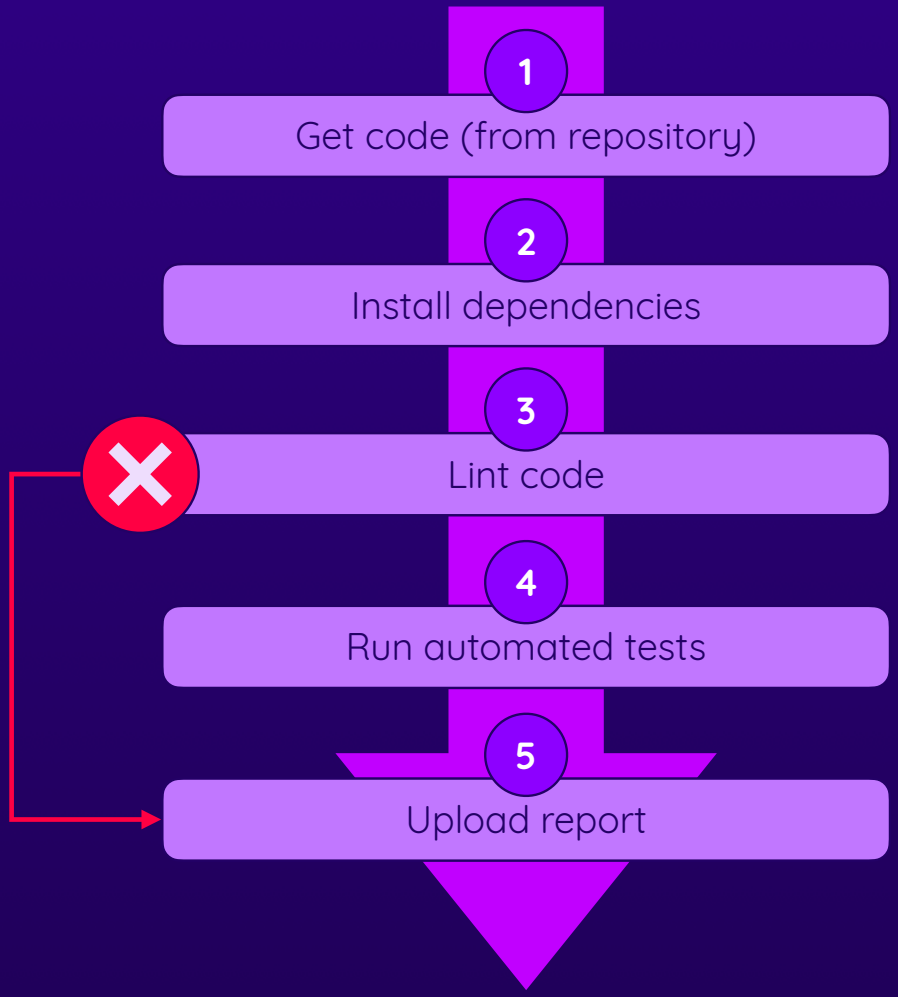
Controlling Execution Flow



Controlling Execution Flow



Controlling Execution Flow



Conditional Jobs & Steps

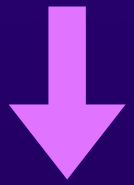
Jobs

Conditional execution via `if` field

Steps

Conditional execution via `if` field

Ignore errors via `continue-on-error` field



Evaluate conditions via Expressions

Special Conditional Functions

failure()

Returns `true` when any previous Step or Job failed

success()

Returns `true` when none of the previous steps have failed

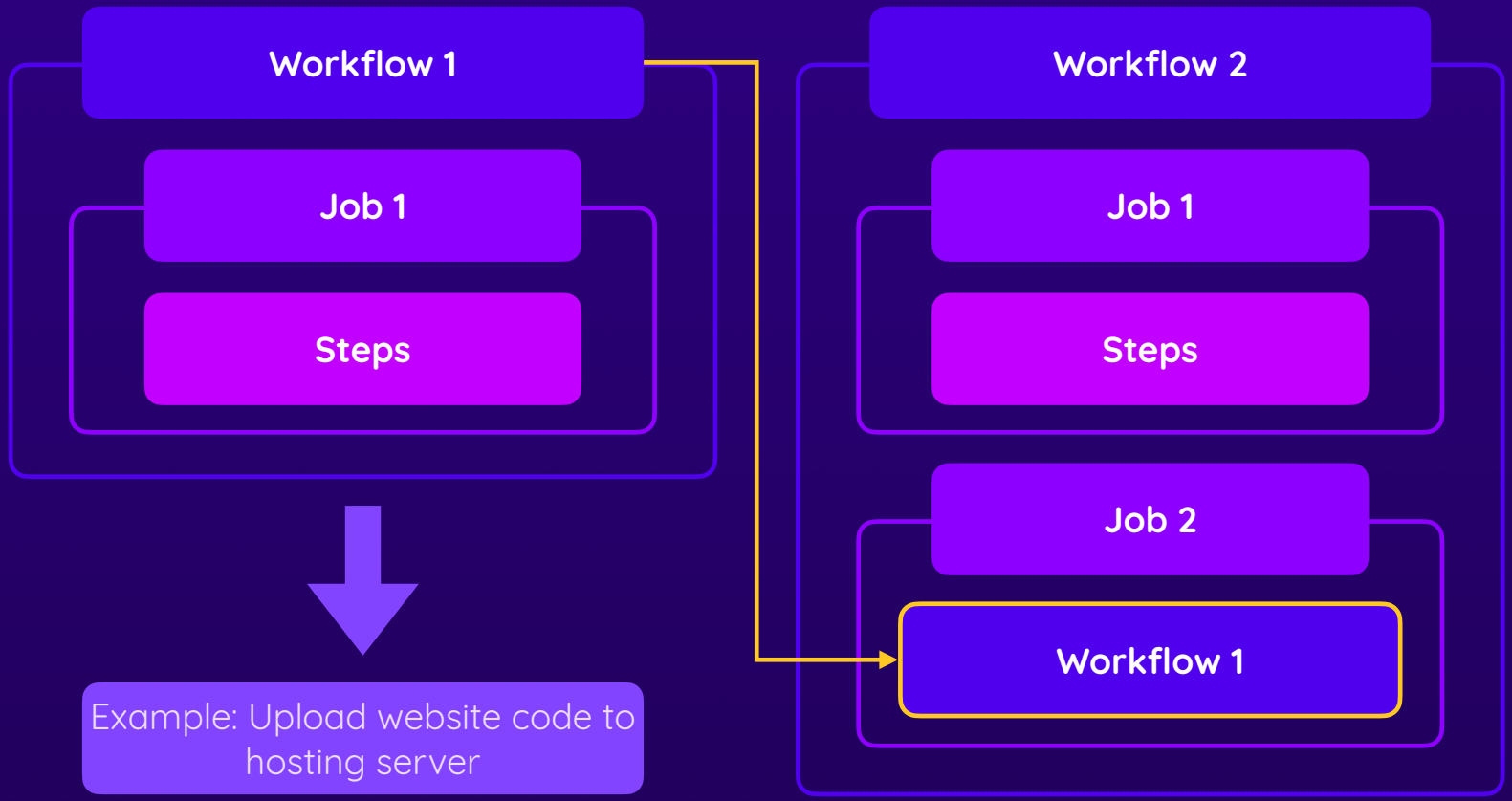
always()

Causes the step to always execute, even when cancelled

cancelled()

Returns `true` if the workflow has been cancelled

Reusable Workflows



Module Summary

Conditional Jobs & Steps	Matrix Jobs	Reusable Workflows
Control Step or Job execution with `if` & dynamic expressions	Run multiple Job configurations in parallel	Workflows can be reused via the `workflow_call` event
Change default behavior with `failure()`, `success()`, `cancelled()` or `always()`	Add or remove individual combinations	Reuse any logic (as many Jobs & Steps as needed)
Use `continue-on-error` to ignore Step failure	Control whether a single failing Job should cancel all other Matrix Jobs via `continue-on-error`	Work with `inputs`, `outputs` and `secrets` as required