**Practical Exercise 11 - Defining Dependencies & Executing Jobs Conditionally**

**Exercise Description**

**In this practical exercise, our goal is to explore in details how to define dependencies between jobs, as well as how to execute jobs conditionally and how to continue the execution despite failures.**

Here are the instructions for the exercise:

1. Create a file named 10-execution-flow.yaml under the .github/workflows folder at the root of your repository.
2. Name the workflow 10 - Controlling the Execution Flow.
3. Add the following triggers with event filters and activity types to your workflow:
   1. workflow\_dispatch: the workflow\_dispatch trigger should receive a single input named pass-unit-tests, of type boolean and with a default value of false.
4. This workflow will contain six jobs:
   1. The first job:
      1. Should be named lint-build
      2. Should execute on ubuntu-latest
      3. Should contain a single step named Lint and build that prints the following message: "Linting and building project"
   2. The second job:
      1. Should be named unit-tests
      2. Should execute on ubuntu-latest
      3. Should contain two steps:
         1. The first step, named Running unit tests, should print the following message: "Running tests..."
         2. The second step, named Failing tests, should exit with a non-zero code and it should execute if and only if the input pass-unit-tests is set to false.
   3. The third job:
      1. Should be named deploy-nonprod
      2. Should execute on ubuntu-latest
      3. Should execute only after both the lint-build and the unit-tests jobs successfully complete
      4. Should contain a single step named Deploying to nonprod that prints the following message: "Deploying to nonprod..."
   4. The fourth job:
      1. Should be named e2e-tests
      2. Should execute on ubuntu-latest
      3. Should execute only after the deploy-nonprod job successfully completes
      4. Should contain a single step named Running E2E tests that prints the following message: "Running E2E tests"
   5. The fifth job:
      1. Should be named load-tests
      2. Should execute on ubuntu-latest
      3. Should execute only after the deploy-nonprod job successfully completes
      4. Should contain a single step named Running load tests that prints the following message: "Running load tests"
   6. The sixth job:
      1. Should be named deploy-prod
      2. Should execute on ubuntu-latest
      3. Should execute only after both the e2e-tests and load-tests jobs successfully complete
      4. Should contain a single step named Deploying to prod that prints the following message: "Deploying to prod..."
5. Commit the changes and push the code. Trigger the workflow from UI, providing varying values for the pass-unit-tests input. Take a few moments to inspect the output of the workflow runs. How did the failing of a single job impact downstream dependent jobs?
6. Now modify the unit-tests job to allow the continuation of the workflow even if this specific job fails. This can be done by adding continue-on-error: true at the job definition level.
7. Commit the changes and push the code. Trigger the workflow from UI, providing varying values for the pass-unit-tests input. Take a few moments to inspect the output of the workflow runs. How did the failing of a single job impact downstream dependent jobs?
8. continue-on-error should be used sparingly, only if there is a specific use-case for it (for example, experimental features). Remove the continue-on-error option from the job definition, commit the changes and push the code.