**GitHub Portfolio: Workflow Learning Journey**

**Objective**

This portfolio tracks my learning process in GitHub Actions. Each section showcases a specific concept or feature, explained with objectives, code examples, and case studies. Screenshots are used to highlight key steps and outputs.

## Workflow: Managing Concurrency

### Objective

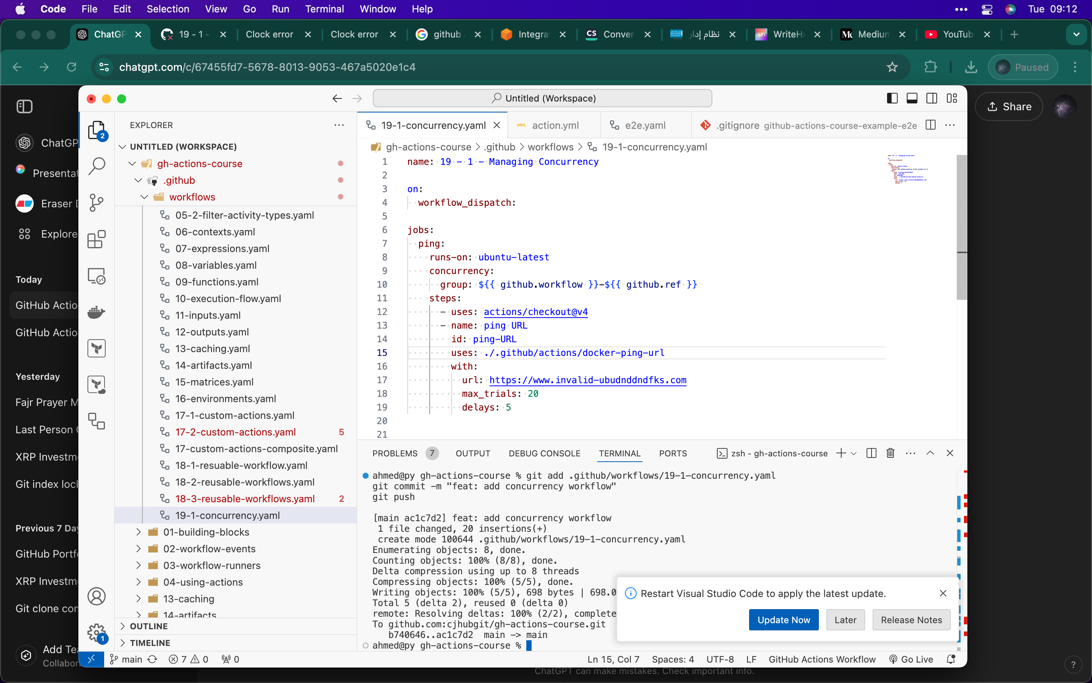
The goal of this workflow is to demonstrate how to manage concurrency in GitHub Actions, ensuring that only one instance of a workflow runs at a time for a specific branch.

### Workflow Structure

#### 1. Trigger Event

The workflow is triggered manually using workflow\_dispatch. This allows the workflow to be started only when you manually trigger it.

****

****

### Workflow Structure

#### 2. Defining Concurrency

The concurrency feature is used to limit workflow runs. This prevents multiple instances of the workflow from running simultaneously.

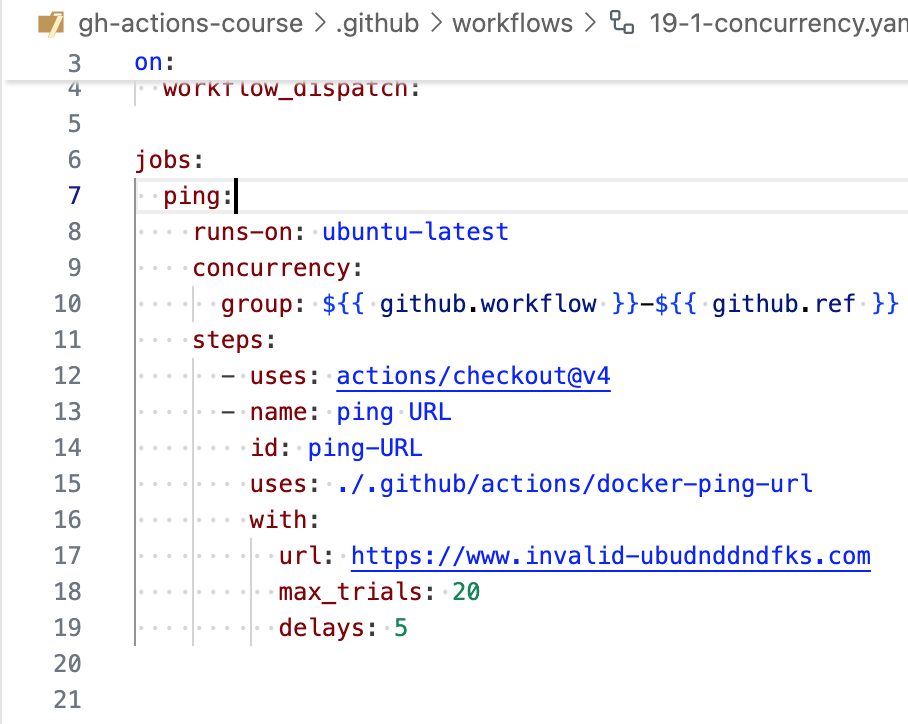
**Explanation:**

* ${{ github.workflow }}: Refers to the name of the workflow file (9-1 Managing Concurrency), ensuring the workflow name is part of the concurrency group.
* ${{ github.ref }}: Refers to the branch or tag reference (e.g., main). This ensures concurrency is scoped to a specific branch or tag.

### Workflow Structure

#### 3. Jobs Section

The ping job demonstrates how to use concurrency while performing a custom task like pinging a URL using a custom Docker action.

Code

Explanation:

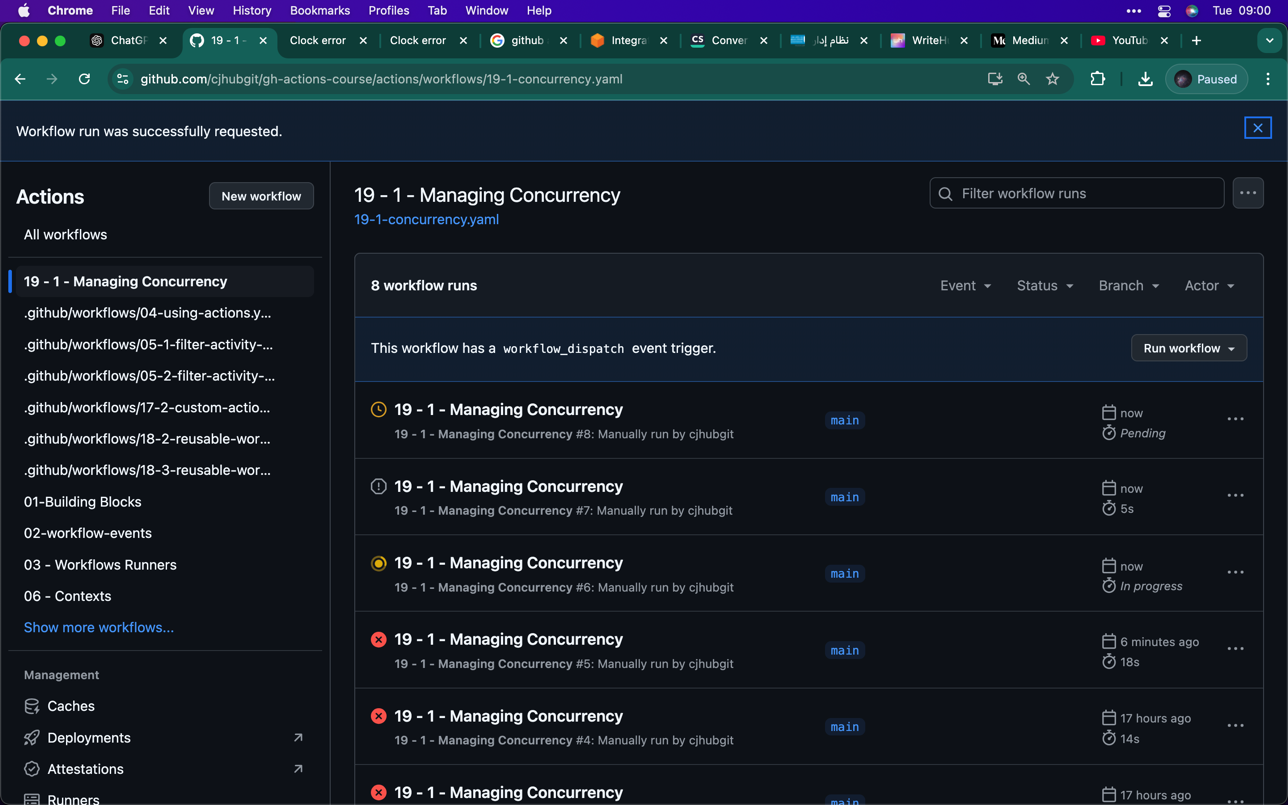
* runs-on: ubuntu-latest: Specifies the operating system where the job runs.
* Concurrency Block: Ensures only one ping job runs at a time for the same workflow and branch.
* Steps:
  + Checkout Code: Retrieves the repository code.
  + Ping URL: Uses a custom Docker action (docker-ping-url) to ping a specified URL.
    - url: The URL to ping (in this case, an intentionally invalid one).
    - max\_trials: The maximum number of retry attempts.
    - delays: The delay (in seconds) between retry attempts.

### Testing the Workflow

1. **Push to the Main Branch**Commit and push a change to the main branch to trigger the workflow.

**Screenshot Opportunity:**Take a screenshot of the GitHub Actions page showing the triggered workflow.

1. **Trigger Another Workflow**Push another change while the first workflow is running to observe concurrency management.



**Testing the Workflow**

In this example, I triggered the workflow 19-1 Managing Concurrency multiple times manually for the main branch. GitHub's concurrency feature took over and ensured that only one workflow runs at a time, while the others were either cancelled or queued. Here's what’s happening:

1. **Cancelled Workflows:**
   * Runs #4, #5, and #6 were cancelled automatically because a newer workflow (#7) was triggered. GitHub doesn’t let multiple workflows from the same concurrency group run simultaneously.
2. **Running Workflow:**
   * Run #7 is the most recent active workflow, so it’s currently running. GitHub allows this workflow to proceed because it’s the latest one in the group.
3. **Pending Workflow:**
   * Run #8 is in a "Pending" state, waiting for Run #7 to finish before it can start. This ensures workflows don’t overlap and cause conflicts.

**Why This Matters:**

This screenshot is a great example of GitHub's concurrency feature in action. It shows how GitHub prevents redundant runs and ensures only the most relevant workflow continues, saving time and resources.