Parallel in Github using Matrix

In the world of software development, efficiency is king. As developers, we constantly seek ways to make our code not just work better, but faster and more efficiently. Enter the realm of Continuous Integration/Continuous Deployment (CI/CD) pipelines, a critical component of modern development practices. Among the myriad of tools and features available, one stands out for its ability to dramatically enhance the speed and adaptability of our workflows: parallel execution with GitHub Actions matrix.

But what exactly is it, and how can it transform your CI/CD pipeline from a single-lane road into a multi-lane highway, speeding up the delivery process?

This article is your roadmap to understanding and implementing parallel execution in your GitHub workflows. We'll dive into the nuts and bolts of how to set up and benefit from this powerful feature, ensuring your development process is not just efficient, but a model of modern engineering excellence. Whether you're building the next big thing or improving an existing project, let's explore how parallel execution with GitHub Actions matrix can elevate your work to new heights.

Parallel execution with GitHub Actions matrix allows efficient and simultaneous processes across multiple services, environments, or configurations within a CI/CD pipeline. This feature dynamically generates jobs based on combinations of given variables, improving the workflow's speed and flexibility without manual duplication.

How It Works:

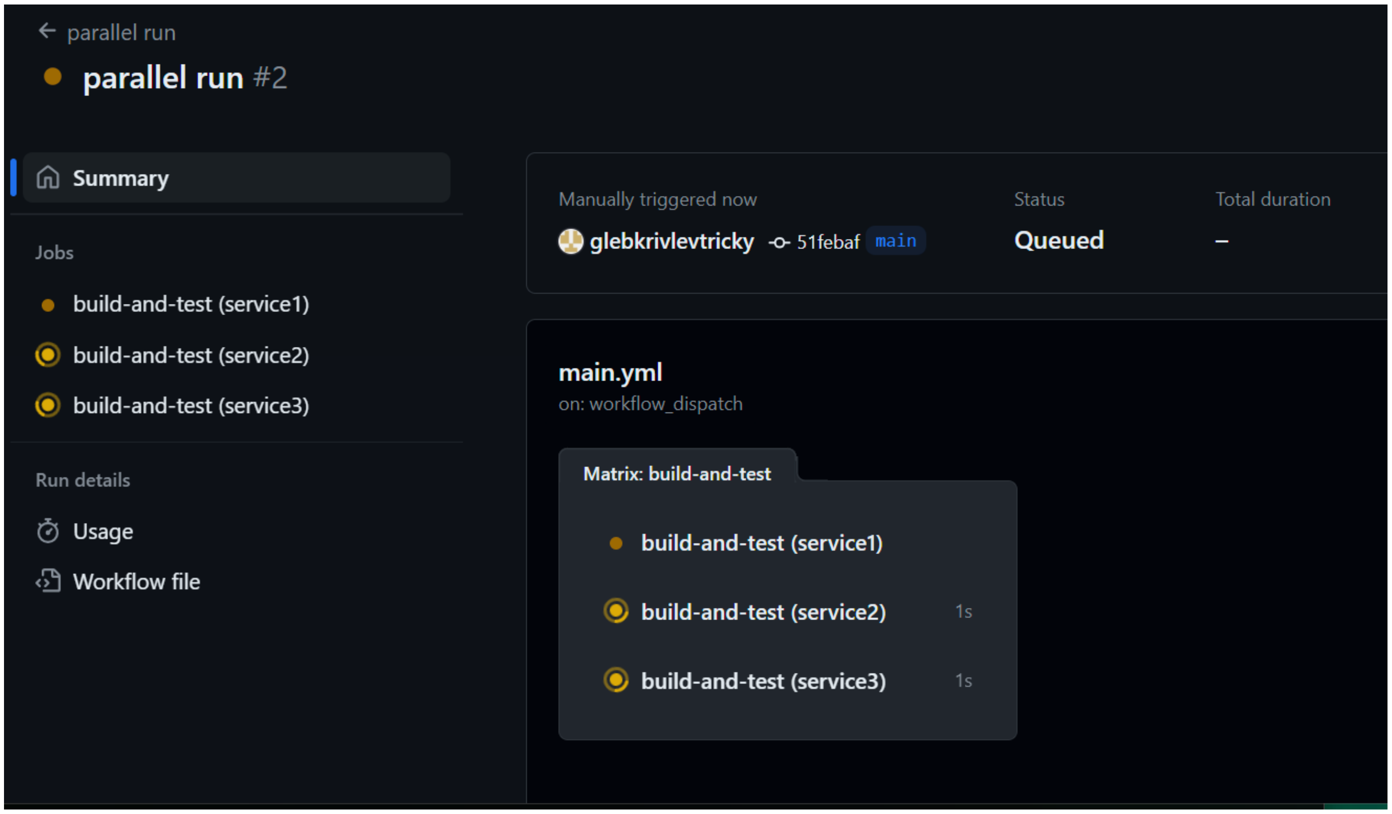
Define a Matrix: Use strategy. Matrix in the workflow YAML to specify variables like service names, environments, or versions.

‘’’

```
strategy:
      matrix:
        your-definition:
```

‘’’

Automatic Parallel Jobs: GitHub Actions create jobs for each matrix combination, running them concurrently.



← parallel run

● parallel run #2

Summary

Jobs

● build-and-test (service1)

● build-and-test (service2)

● build-and-test (service3)

Run details

Usage

Workflow file

Manually triggered now	Status	Total duration
glebkrivlevtricky 51febaf main	Queued	–

main.yml
on: workflow_dispatch

Matrix: build-and-test

● build-and-test (service1)

● build-and-test (service2) 1s

● build-and-test (service3) 1s

TRICKY
Simplifying technology

Benefits:

- Speed: Parallel jobs reduce total runtime, offering quicker feedback loops.
- Scalability: Easily extend processes across more services or configurations by simply adding to the matrix.
- Versatility: Run processes across various conditions (e.g., multiple databases and configurations) in one workflow.
-

Example: Building and testing a web application across different services such as user authentication, payment processing, and data storage simultaneously to ensure compatibility and performance across all services.
‘’’

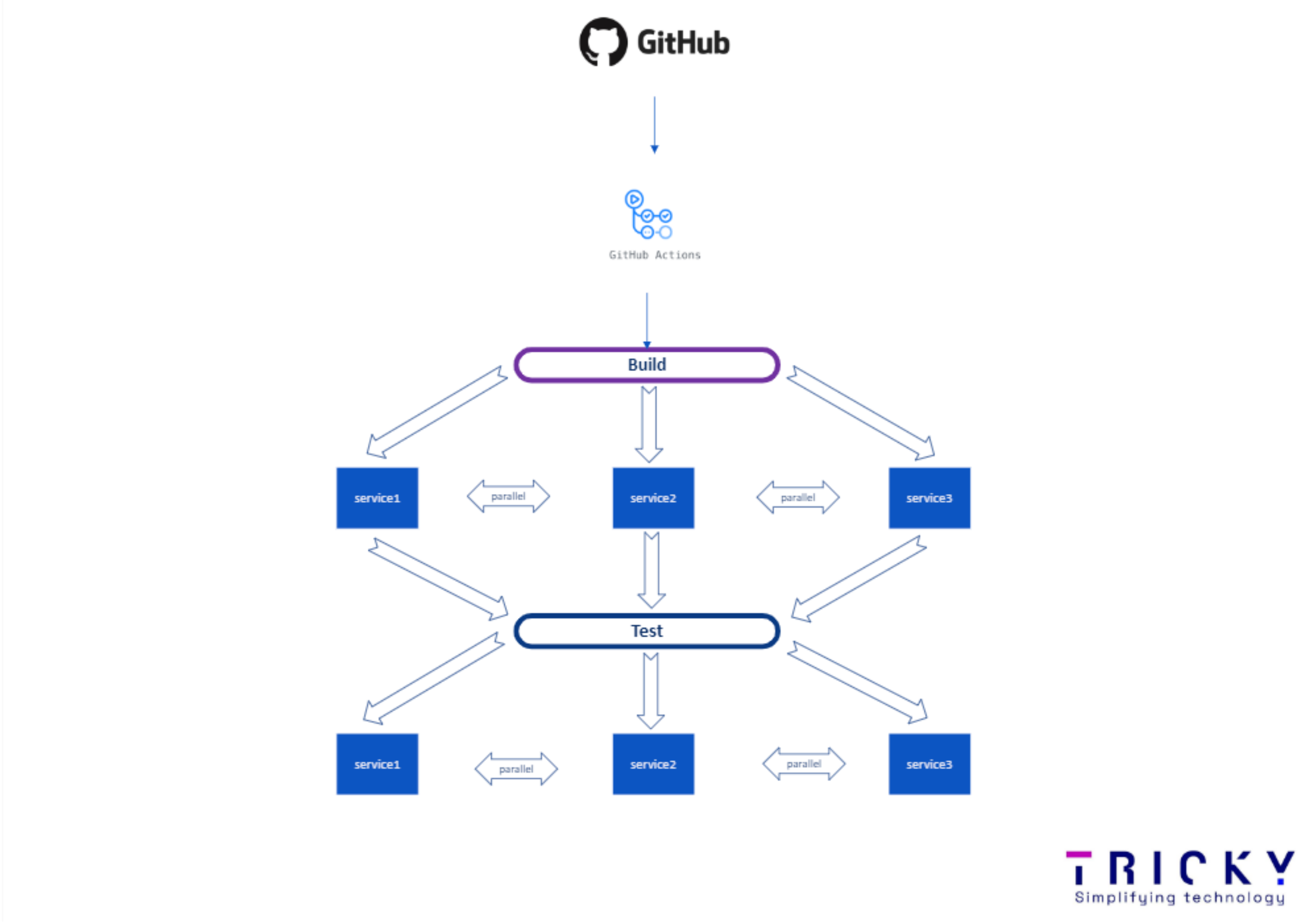
```
jobs:
  build-and-test:
    runs-on: ubuntu-latest
    strategy:
      matrix:
        service: [service1, service2, service3]

    steps:
    - uses: actions/checkout@v2

    - name: Build ${{ matrix.service }}
      run: ./${{ matrix.service }}/build-script.sh ${{ matrix.service }}

    - name: Test ${{ matrix.service }}
      run: ./${{ matrix.service }}/test-script.sh ${{ matrix.service }}
```

‘’’



Best Practices:
Optimize the Matrix: Limit to meaningful combinations to conserve CI resources.
Monitor CI Usage: Keep an eye on the parallel jobs' impact on your CI minutes allocation.

Utilizing the job matrix for parallel execution in GitHub Actions significantly enhances CI/CD efficiency, especially for complex projects requiring tests across multiple services or configurations.