Embedded Test Engineer Tech Challenge

This challenge is designed to test your ability to build a distributed automated test system: it will involve working with tools such as **RabbitMQ**, **Celery**, and whichever your preferred containerized solution (**Docker**, **Podman**, etc) is. We're interested in how you approach the problem, organize your code, and explain your reasoning.

The Challenge

You will:

1. Run a **RabbitMQ server** locally (not inside the container).
2. Write a small **Celery app** with (at least) two tasks:
    - `task_a`: returns `"Hello from Task A"`.
    - `task_b`: returns `"Hello from Task B"`.
3. Create **two containers** running Celery workers:
    - Container 1 should process only `task_a`.
    - Container 2 should process only `task_b`.
4. Write a dispatcher script (`dispatch.py`) that sends one job to each container *concurrently* and prints both results.

Dummy example output when running:

```
$ python dispatch.py
Result from task_a: Hello from Task A
Result from task_b: Hello from Task B
```

Optional Stretch Goal

Do **something interesting** with the task output in processing the result: visualization, structured logging, etc: we leave it up to you.

Deliverables

Please provide a repository containing:

- `celery_app.py`: Celery config + tasks.
- A file for worker containers (`Dockerfile`, `devcontainer.json`)
- `dispatch.py`: script to run tasks.
- *(optional)* `test-config.yml` if you use it for orchestration.
- `README.md`: setup and usage instructions:
    - How to start RabbitMQ.
    - How to build and run the worker containers.

    - How to run `dispatch.py` and see the output.
    - *(optional)* How to try out your stretch goal.
    - *(optional)* Clarify approach / describe the work

Evaluation Criteria

- **Correctness**: RabbitMQ + Celery run as expected, tasks execute.
- **Isolation**: Each container only processes its designated task.
- **Clarity**: Your README is clear and setup is reproducible.
- **Bonus Points**:
- Creativity in how you handle the results
- How to orchestrate RabbitMQ + workers.
- Add structured logging.
- Demonstrate retries or task monitoring.