🧪 Integration Testing with Docker Compose

📘 Overview

Integration testing verifies that multiple components or services in a system work together as expected. When dealing with containerized applications—especially microservices—each service often runs in isolation using Docker. **Docker Compose** makes it easy to spin up all dependent services for integration testing in a consistent and reproducible environment.

🧱 Why Use Docker Compose for Integration Testing?

Docker Compose allows defining multi-container applications in a single YAML file. For integration testing, it provides:

- **Dependency orchestration**: Easily define databases, APIs, message queues, etc.
- **Repeatability**: Same environment across CI/CD and local dev.
- **Isolation**: Test environment doesn't affect production data.
- **Lifecycle management**: Tear down containers after tests complete.

🔬 Integration Test Workflow

Step-by-step Flow:

1. **Start services** with Docker Compose:

`bash docker-compose up -d`

1. **Wait for services to be healthy** (PostgreSQL might take a few seconds).
2. **Run test scripts** (e.g., using `pytest`, `unittest`, or a custom test runner).
3. **Clean up**:

`bash docker-compose down -v`

🚀 Running Tests in CI (e.g., GitHub Actions)

```
jobs:
  test:
    runs-on: ubuntu-latest
    services:
      postgres:
        image: postgres:15
        env:
          POSTGRES_DB: test_db
          POSTGRES_USER: test_user
          POSTGRES_PASSWORD: test_pass
        ports: [5432:5432]

    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v4
        with:
          python-version: 3.10
      - name: Install dependencies
        run: pip install -r requirements.txt
      - name: Run tests
        run: pytest tests/
```

Alternatively, use `docker-compose` in CI:

```
docker-compose up -d
pytest tests/
docker-compose down -v
```

🔄 Tear Down and Cleanup

Always shut down containers and clean up resources:

```
docker-compose down -v --remove-orphans
```

Use a trap or teardown fixture in test runners to automate this.

📌 Summary

Feature	Benefit
Docker Compose	Simplifies multi-service orchestration
Repeatability	Same test environment across platforms
Isolation	No pollution of prod data
CI/CD Friendly	Easily integrated into pipelines

Docker Compose makes integration testing both **scalable and maintainable**, especially for microservices and modern cloud-native apps.