Docker Compose Tutorial – Week 5: Advanced Docker Compose for Real Projects

By Week 5, you should already be comfortable running and managing multi-container environments. This week focuses on **advanced Docker Compose workflows** — including multi-environment setups, healthchecks, scaling, networking, volumes, and secrets management.

Goal

By the end of this week, you will be able to: 1. Structure Compose files for dev, test, and production. 2. Add service healthchecks and dependencies. 3. Use named volumes and shared networks effectively. 4. Manage environment variables and secrets securely. 5. Scale services horizontally.

Lesson 1: Multi-environment Setup

Use override files for different environments. Example structure: project/ ■■■ docker-compose.yml ■■■ docker-compose.dev.yml ■■■ docker-compose.prod.yml ■■■ .env Run commands: docker compose -f docker-compose.yml -f docker-compose.dev.yml up -d docker compose -f docker-compose.yml -f docker-compose.prod.yml up -d

Lesson 2: Healthchecks and Dependencies

Ensure services start only after dependencies are ready. Example: db: image: postgres:16 healthcheck: test: ["CMD-SHELL", "pg_isready -U appuser"] interval: 10s timeout: 5s retries: 5 web: depends_on: db: condition: service_healthy

Lesson 3: Persistent Volumes

Named volumes allow data to persist across container restarts. Example: services: db: image: postgres:16 volumes: - pgdata:/var/lib/postgresql/data volumes: pgdata:

Lesson 4: Networking and Service Discovery

Services on the same Compose network can reach each other by name. Example: backend: image: myapp-backend frontend: image: myapp-frontend environment: - BACKEND_URL=http://backend:5000

Lesson 5: Environment Variables and Secrets

Use .env files or Docker secrets for configuration. Example: db: image: postgres:16 env_file: .env For secrets: secrets: db_password: file: ./secrets/db_password.txt

Lesson 6: Scaling Services

Scale services horizontally for load handling. Command: docker compose up --scale worker=3 -d Docker will create worker_1, worker_2, worker_3.

Lesson 7: Cleaning and Debugging

Common commands: docker compose ps docker compose logs docker compose top docker compose down -v docker system prune -a

Practice Exercise

Extend your setup (Airflow + Feast + Jupyter + Redis) to include: 1. A Postgres service with a persistent volume. 2. Healthchecks for Postgres and Redis. 3. Separate dev and prod override files. 4. Sensitive credentials in .env. Then test: docker compose -f docker-compose.yml -f docker-compose.dev.yml up -d docker compose ps Ensure all services show “healthy” in the STATUS column.