Monolith to Micro-Services: Overall Problem Statement

Status In progress

Timing Jan 15, 2026 to Feb 17, 2026

Owners Soumik Dutta Arnab Bhakta

You have just joined the engineering team at **"ShopStyle,"** a successful e-commerce company. For years, the company has run on a single, massive Java Spring Boot application—**The Monolith**.

While it served the company well in the startup phase, it is now a bottleneck:

- **Scaling is Impossible:** To handle more "Browsing" traffic on Black Friday, we have to scale the entire application (including the heavy Payment and Order modules), wasting money.
- **Fragile Deployments:** Last week, a bug in the "Review" logic crashed the "Checkout" system. Everything went down.
- **Tech Lock-in:** The Data Science team wants to use **Python** for recommendations, but the Monolith forces everyone to use **Java**.

The Mission

Your Team Lead has tasked you with migrating this legacy system to a **Microservices Architecture**.

The Constraint: You cannot just shut down the store and rewrite it. You must use the **Strangler Fig Pattern** to gradually replace pieces of the Monolith while the business keeps running.

Lab Roadmap

This lab is divided into three progressive activities. You must complete them in order.

🛑 Activity 0: Understanding the Monolith

- **The Baseline:** You will deploy the legacy `monolith-app` container.
- **The Problem:** You will inspect the **Shared Database** anti-pattern and see how the User, Product, and Order domains are tightly coupled in a single schema.
- **The Goal:** Understand *why* we are migrating before writing a single line of code.

🔨 Activity 1: Handheld Migration (The Strangler Fig)

- **The Guided Path:** We will walk you through extracting core domains step-by-step.
- **Inventory Service (Java):** You will implement the **Database-per-Service** pattern using PostgreSQL.
- **Product Service (Python):** You will introduce **Polyglot Programming** by running a Python microservice alongside the Java Monolith.
- **Order Service (The Aggregator):** You will build an **Anti-Corruption Layer** that orchestrates transactions across these distributed services.
- **The Result:** A Hybrid architecture where new traffic goes to Microservices, and old traffic goes to the Monolith.

🎓 Activity 2: The Challenge (Extract "Reviews")

- **The Exam:** The training wheels come off.
- **The Task:** You must extract the **Review Service** (Python) on your own.
- **Assessment:** The Autograder will ruthlessly check if you have decoupled it correctly from the Monolith.

Technical Stack

Throughout this lab, you will work with the following technologies:

Component	Technology	Role
Legacy App	Java 17 (Spring Boot)	The Monolith to be strangled.
Microservices	Python (FastAPI) & Java	The new Polyglot services.
Databases	PostgreSQL, MySQL, H2, SQLite	Showing different persistence strategies.
Orchestration	Docker & Docker Compose	Managing the distributed ecosystem.

Prerequisites

Before starting Activity 0, ensure you have:

1. **Docker Desktop** (or Engine) installed and running.
2. **Git** to clone the lab repository.
3. A terminal (Bash, PowerShell, or Zsh).
4. Approximately **4GB** of free RAM (to run the full stack in Activity 1).

Getting Started

The code for all activities is located in your lab directory.

- `~/labdirectory`

Are you ready to break the Monolith? Proceed to **Activity 0**.

END OF DOCUMENT