LAB: Dependency Mapping & Impact Chain Analysis (Case Study)

1. Lab Overview

This lab teaches how to identify hidden dependencies and map impact cascades when a critical function or service fails. Participants will use a case study of a SaaS company and build a dependency map, then trace how a single failure propagates through the organization and outward to customers.

Learning outcomes:

- Understand dependency layers (people, systems, infrastructure, third parties)
- Identify upstream and downstream dependencies
- Recognize failure amplification through chained systems
- Trace consequence pathways across business functions
- Expose single points of failure hidden behind abstractions

2. Case Study Scenario Introduction

You are part of the Resilience & Continuity Working Group for **FlowCast**, a mid-size SaaS company providing workflow automation and data dashboards for enterprise customers. The platform supports 2,300 business clients, primarily in finance, healthcare, and logistics.

FlowCast relies on a **single-region cloud deployment** to reduce cost and complexity. The company is scaling quickly, but resilience has not matured at the same speed as growth. A recent architectural review revealed concerns about dependency fragility, especially during peak usage or infrastructure stress.

3. Organizational Profile (Case Study)

Category	Details
Industry	SaaS – workflow automation & analytics
Size	420 employees
Hosting	Single-region cloud (AWS-like)
Customers	Highly regulated industries (B2B)
Revenue model	Subscription (SaaS)
Business risk	Reliability → churn → contractual penalties
Team structure	Product, SRE/Platform, Data, Customer Success, Compliance

4. Critical Services & Functions (Core Offerings)

Critical Service	Description	Why Critical
Workflow Engine	Executes automation rules and triggers	Central to platform use
Data Analytics Layer	Real-time dashboards for clients	Customer-facing value
Authentication Gateway	Access control & identity	Trust + regulatory
Usage Reporting	Billing + SLA compliance proof	Financial + legal impact

5. Dependency Layers

A. Internal System Dependencies

- Workflow engine relies on message queue service
- Analytics relies on data ingestion + storage layer
- Auth relies on IAM + secrets management
- Usage reporting relies on analytics and logging

B. Infrastructure Dependencies

- Single-region cloud availability zone
- Managed database platform
- Container orchestration platform
- CDN for external dashboard delivery

C. People Dependencies

- SRE team: infrastructure continuity
- Security team: identity & access governance
- Data engineering: ingestion & dashboard uptime
- Customer success: client escalations under incident

D. Third-Party Dependencies

- External identity provider (SSO)
- Payment / billing gateway
- Logging / monitoring SaaS
- Email notification API

6. Exercise 1 – Dependency Identification

Task:
Participants list **all upstream and downstream dependencies** for each critical service.

Guiding prompts:

- What must exist before this service can function?

- What does this service feed into (downstream impact)?
- Who owns continuity of this dependency?
- Is the dependency internal or third-party?

Deliverable: preliminary dependency map per critical service.

7. Exercise 2 – Impact Chain Construction

Task:
Select **one** critical service from the list and trace a failure forward.

Prompts:

- If this fails, what is the first consequence?
- Who or what is affected next?
- At what stage does business impact become visible?
- When does reputational impact begin?

Deliverable: linear impact chain (first-order → second-order → third-order effect).

8. Exercise 3 – Failure Cascade Mapping

Task:
Convert the impact chain into a **cascade diagram** that shows branching impacts.

Include:

- Technical impact (system/platform)
- Business impact (operations/SLAs)
- Customer impact (loss of function or trust)

- Compliance impact (contractual/regulatory)

Goal: show the “blast radius” of a single failure.

9. Exercise 4 – Risk Prioritization by Consequence

Task:

Rank dependencies not by likelihood, but by **consequence severity if they fail**.

Prompts:

- Which dependency is “quietly critical”?
- Where is there no redundancy?
- Which dependency has no workaround?
- Which relies on a single team or tool?

Deliverable: Top 3 high-consequence dependencies with justification.

10. Reflection & Debrief

Discussion questions:

- What surprised you about the depth of dependencies?
- Which parts of the chain were invisible at first?
- Where did business impact show up earlier than expected?
- What failure would customers notice first vs last?
- Which dependency is most fragile from a governance perspective?

Key takeaway:

Dependencies are resilience vulnerabilities when they are not visible, owned, or mapped.