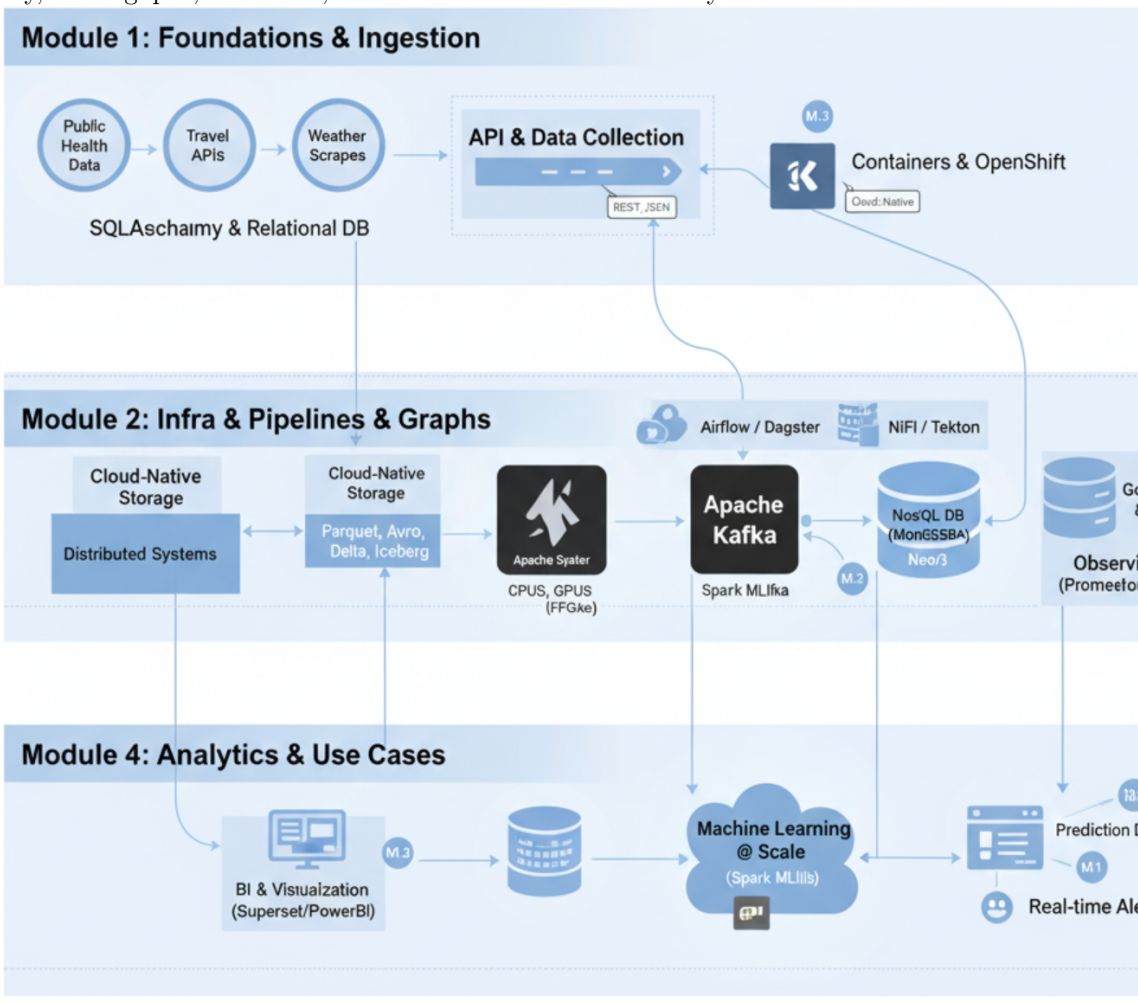
DS-551: Data Engineering at Scale

Fall 2025 Syllabus

Course Description & Pedagogical Approach

Welcome to *Data Engineering at Scale*. This course immerses you in the practice of building and operating systems that collect, move, transform, store, analyze, and publish data at production scale. We organize the semester around the Epidemic Engine—a hypothetical but realistic application that gathers intelligence about potential health events, aggregates heterogeneous data, publishes it in multiple forms, and supports forecasting. Although fictional, the Epidemic Engine mirrors the problems you will face in industry: imperfect data, changing requirements, trade-offs among latency, throughput, and cost, and the need for observability and governance.



Across the term, we will:

- **Collect and integrate** data from APIs, scrapers, and streams.
- **Design and orchestrate** robust pipelines using modern batch and streaming paradigms.
- **Work across data models** (relational, document, graph, geospatial) with an emphasis on interoperability.
- **Operate distributed systems** (containers, clusters, topics, jobs) and reason about failure, back-pressure, and recovery.
- **Publish and monitor** outputs with logs, metrics, and alerts to support decision-making.
- **Evaluate trade-offs** in consistency, availability, performance, governance, and ethics.

The goal is not to memorize tools but to develop **documentation-driven engineering habits** that make you adaptable as technologies evolve.

Instructional Format, Course Pedagogy, and Approach to Learning

The course is a fast-paced, project-based exploration of contemporary data engineering. The Epidemic Engine provides a unifying context so that lectures, assignments, and projects compose into a coherent whole. With few exceptions, the components you build will integrate into that system.

- **Class meetings.** Most meetings begin with a short **in-class assignment** (closed-book, no GenAI) that assumes you completed the assigned reading(s) or AI-prep materials. We then focus lecture time on the hardest concepts, trade-offs, and live demonstrations. Slides and references are posted afterward.
- **Assignments.** Completion-based assignments are tied directly to the class meeting where they are introduced. They verify preparation, reinforce key ideas, and often scaffold into projects. Assignments cannot be made up later, as they are relevant to that class session. To encourage flexibility, the lowest **20%** of assignment grades are automatically dropped.
- **Projects.** Three cumulative projects add capabilities to the Epidemic Engine. Teams document design decisions, deployment choices, observability, and operational run-books.
- **Discussions.** Weekly sections led by the TA emphasize Q&A, debugging, design critiques, and short guided exercises.
- **Recordings.** Class sessions are recorded on a best-effort basis for enrolled students; recordings may not capture every interaction and are not guaranteed.

Attendance is **mandatory**, as in-class assignments and discussions are central to verifying understanding. Active contributions—questions, comments, and insights during lecture—may be noted as **Check+** and can positively influence the assignments category.

Prerequisites

- DS 310 (or equivalent)
- Strong Python programming skills
- Exposure to SQL and DDL
- A GitHub account

Learning Outcomes

By the end of the course, students will be able to:

- Explain the complexity of modern data ecosystems for enterprise-scale applications.
- Apply core concepts and evaluate tools that address data engineering challenges.
- Distinguish conceptual understanding from implementation details of specific tools.
- Adapt to new technologies by effectively reading and applying official documentation.

Books & Tools

- **Required text:** Martin Kleppmann, *Designing Data-Intensive Applications*.
- **Supplemental readings:** Selected articles, docs, and whitepapers (posted in Blackboard Ultra).
- **Primary environment:** Python; GitHub; OpenShift; Kafka; Spark; NiFi; Tekton; supporting CLI and SDKs.
- **Costs:** No student-paid cloud resources are expected.

Courseware

- **Blackboard Ultra** is the **source of truth** for the course: announcements, deadlines, lecture materials, office-hours calendar, and links to submission portals.

- **Gradescope (via Blackboard)** is used for submissions and feedback; access it by clicking the assignment link in Blackboard (direct login is optional).
- **Piazza** is for Q&A and logistical updates; post public questions when possible so answers benefit the class.
- **GitHub** hosts course repositories for assignments and projects.

Assessment & Grading

Grade Distribution

Category	Weight
Assignments	40%
Projects (3)	50%
Peer Evaluation	10%

How Grading Works

- **Assignments:** Completion-based using Check / Check+ / Missing. Lowest 20% automatically dropped. Check+ indicates strong preparation or substantive in-class engagement (e.g., asking questions, contributing to discussion).
- **Projects:** Three projects with increasing scope and weight:
  - **Project 1 (25%)** — API Collection & Integration: schema design, ingestion service, validation tests, deployment manifest.
  - **Project 2 (35%)** — Streaming Pipelines: Kafka + NiFi/Tekton pipelines, stress testing, back-pressure handling, run-book documentation.
  - **Project 3 (40%)** — Deployment at Scale: end-to-end Epidemic Engine on OpenShift with observability, scaling, and a live demo.
  - Projects are graded by rubric on clarity, correctness, reproducibility, observability, documentation, and teamwork. Peer evaluations and contribution evidence (commits, PRs, reviews, issues) may adjust individual project scores.
- **Peer Evaluations:** Team feedback affects final grades on collaborative projects.

Timing & Late Work

- **Assignments:** Due in class. No make-ups; missed assignments fall under the 20% auto-drop.

- **Projects:** Follow milestone dates precisely. Certain deliverables (e.g., proposal/final) may allow 48h late at −10%—see each spec.

GenAI Policy (for DS-551)

GenAI is a **productivity enhancer**, but the work is still **yours**. You must be able to **demonstrably prove your understanding** of any material where GenAI provided assistance.

Context	Examples (non-exhaustive)	Notes
Encouraged	Brainstorm data models and pipeline sketches; draft manifests; summarize docs; rubber-duck debugging	Cite all meaningful AI assistance in your usage log.
Allowed with care	Generate scaffolding code; produce config/templates; refactor for clarity; propose observability metrics	You are responsible for correctness, licensing, and being able to explain every decision.
Prohibited	Any **individual verification** (in-class assignments, oral/written quizzes) or activity explicitly marked no-AI	Violations are academic misconduct.

Academic Integrity & Licensing

We follow BU's Academic Conduct Code. Collaboration is encouraged where specified, but all submissions must cite collaborators, data sources, libraries, and GenAI assistance. Respect open-source licensing—**no license no reuse**.

How to Succeed

1. Prepare before class: complete readings and AI-prep; expect a short in-class assignment.
2. Treat assignments as project scaffolding: build reusable components.
3. Contribute in lecture: substantive questions and comments are valued and can elevate your assignment credit.
4. Document decisions and trade-offs as you go; automate where possible.
5. Use office hours and Piazza actively; ask focused questions and share minimal reproducible examples.
6. Build team habits: small PRs, code reviews, and clear commit messages.

Support & Accessibility

- **Disability accommodations:** Office for Disability Services (617-353-3658, access@bu.edu). Share your letter privately with the instructor.
- **Student support:** ERC tutoring, writing resources, and BU Student Wellbeing services (see Blackboard for links).

Regrades

Submit regrade requests within **7 days** of score release via **Gradescope (accessed through Blackboard)**. Scores may go up, down, or remain unchanged; decisions are final.

Course Modules (High-Level Overview)

Module 1 — Foundations & Setup

Git/Python workflow refresh · APIs & collection · Relational integration · Containers & OpenShift basics

Module 2 — Architectures & Infrastructure

Distributed systems fundamentals · Cloud-native storage formats (Parquet, Avro, Delta, Iceberg) · Hardware trade-offs (CPU/GPU/FPGA) · Event-driven architectures · MapReduce & Spark foundations

Module 3 — Pipelines & Orchestration

Pipeline tools (NiFi, Tekton) · Orchestration frameworks (Dagster, Airflow) · NoSQL & graph databases · Kafka and streaming patterns · Serverless & functions

Module 4 — Advanced Topics & Integration

ML at scale (Spark MLlib, GPU acceleration) · Geospatial systems (PostGIS, GIS pipelines) · BI & visualization (tool-agnostic) · Data governance & lineage · Security & ethics · Observability & monitoring · Scaling & resource management · Case studies & wrap-up