1. Solution Architecture:

For the logical architecture, we'll have the following components:

* Microservices:
* Attendance Service:
  + Responsible for handling attendance-related operations.
  + Exposes REST endpoints for swipe in/out and total hours calculation.
  + Subscribes to events from the Event Service for updating attendance records.
* Event Service:
  + Responsible for handling events and streaming to Kafka.
  + Receives swipe in/out events from REST calls and publishes them to Kafka topics.
  + Subscribes to Kafka topics for processing events and updating attendance records in the Attendance Service.
* Azure Services**: (Initially: Apache Kafka for event streaming, Docker containers for DB)**
* Azure Event Hubs:
  + Used for event streaming and message processing.
  + Provides a Kafka-compatible interface for publishing and consuming events.
* Azure SQL Database:
  + Stores employee information such as employee ID, name, and department.
  + Accessed by the Attendance Service for querying employee data.
* Azure Cosmos DB:
  + Used as the event store for storing swipe in/out events.
  + Provides scalability and fault tolerance for event storage.
* API Gateway:
  + Utilize Azure API Management for managing APIs, enforcing security policies, and monitoring API usage
* Monitoring:
  + Utilize Azure Monitor for logging, monitoring, and alerting on system performance and health.
* CI/CD Pipeline:
  + Utilize Azure DevOps for continuous integration and continuous deployment pipelines for automated build, test, and deployment processes.

For the physical architecture, we'll deploy these components on Azure cloud using Azure Kubernetes Service (AKS) for container orchestration.

Detailed Design

* Microservices:
  + Attendance Service:
* Responsibilities:
* Handling attendance-related operations (swipe in/out, total hours calculation).
* Exposing REST endpoints for external communication.
* Interacting with event service for event processing.
* API Contracts:
  + POST /attendance/swipe/in
  + POST /attendance/swipe/out
  + GET /attendance/total-hours/{employeeId}
* Event Structures:
  + SwipeInEvent: { employeeId, timestamp }
  + SwipeOutEvent: { employeeId, timestamp }
* Data Models:
  + Employee: { employeeId, name, department }
  + AttendanceRecord: { employeeId, swipeInTimestamp, swipeOutTimestamp }
* Interactions:
  + Handles incoming swipe in/out requests, updates attendance records, and publishes events to Kafka.
  + Queries read store for total hours calculation.
  + Event Service:
* Responsibilities:
  + Handling events and event streaming.
  + Publishing events to Kafka topics.
* Event Structures:
  + SwipeInEvent: { employeeId, timestamp }
  + SwipeOutEvent: { employeeId, timestamp }
  + EODTotalHoursEvent: {employeeId, totalHours, status }
  + Interactions:
    - * + Subscribes to REST endpoints for swipe in/out.
        + Publishes events to Kafka topics.
  + Data Stores:
    1. Read Store (Local Database):
       - * Database: MySQL for local development.
         * Tables:

Employee: { employeeId (PK), name, department }

AttendanceRecord: { employeeId (FK), swipeInTimestamp, swipeOutTimestamp }

* + 1. Event Store (Local Kafka):
       - Message Broker: Apache Kafka for local development.
       - Topics:
         * SwipeInTopic
         * SwipeOutTopic
         * EODTotalHoursTopic

Sequence Diagram:

