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.
* Azure Services**: (Initially: Apache Kafka for event streaming instead of Azure Event hubs, MySQL instead of Azure SQL Database, Cassandra instead of Azure Cosmos DB, Kubernetes instead of AKS, 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:
    - Handles attendance-related operations such as total hours calculation and updating attendance records.
    - Exposes REST endpoints for querying attendance information.
    - Subscribes to swipe events from the event service via Kafka for processing.
  + API Contracts:
    - GET /attendance/total-hours/{employeeId}: Retrieves total hours worked by an employee.
    - GET /attendance/absentees/{date}: Retrieves a list of employees absent on a specific date.
    - GET /attendance/half-day/{date}: Retrieves a list of employees who worked half-day on a specific date.
    - GET /attendance/present/{date}: Retrieves a list of employees who worked full-day on a specific date.
* Event Structures (for internal processing, not exposed via API):
* SwipeInEvent: { employeeId, timestamp }
* SwipeOutEvent: { employeeId, timestamp }
* Data Models:
* Employee: { employeeId, name, department }
* AttendanceRecord: { employeeId, date, swipeInTimestamp, swipeOutTimestamp }
* Interactions:
  + Subscribes to swipe events (SwipeInEvent, SwipeOutEvent) from the event service via Kafka for processing.
  + Calculates total hours worked and updates attendance records in the MySQL database.
  + Provides endpoints for querying attendance information.
* Event Service:
* Responsibilities:
  + - * Handles events and event streaming.
      * Publishes events to Kafka topics.
* Event Structures:
  + - * SwipeInEvent: { employeeId, timestamp }
      * SwipeOutEvent: { employeeId, timestamp }
* Interactions:
  + - * Receives swipe events from external sources (e.g., devices) or other systems.
      * Publishes swipe events (SwipeInEvent, SwipeOutEvent) to Kafka topics.
* Data Stores:
* Attendance Service (MySQL):
* Tables:
* Employee: { employeeId (PK), name, department }
* AttendanceRecord: { employeeId (FK), date, swipeInTimestamp, swipeOutTimestamp }
* Event Service (Cassandra):
* Tables:
* SwipeEvents: { employeeId (PK), eventTimestamp (CK), eventType, PRIMARY KEY (employeeId, eventTimestamp) }
* Kafka:
* Topics:
  + - * Swipe-in-topic
      * swipe-out-topic

Sequence Diagram:

