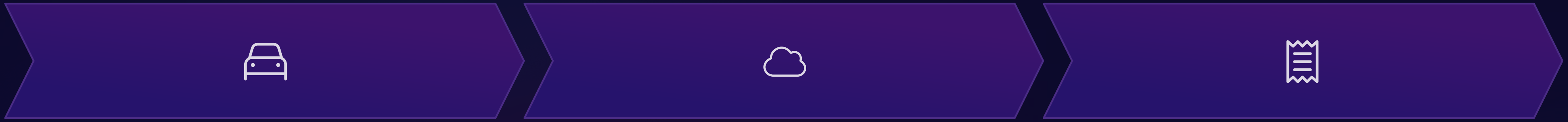
Cloud-Based Parking Lot Management System: Implementation Guide

This document provides comprehensive instructions for building and deploying a cloud-based parking lot management system on AWS. The system will handle vehicle entry and exit operations, including license plate recognition, time tracking, and fee calculation based on parking duration. Two deployment approaches are outlined: serverless architecture and EC2 instance hosting.

by Dor Kleiman

Scenario



Vehicle Entry

Camera captures license plate

Cloud Processing

Store data and generate ticket

Vehicle Exit

Calculate fee and process payment

The parking lot management system utilizes cloud infrastructure to process vehicle entry and exit. When a vehicle enters, a camera recognizes the license plate and sends this information to our cloud service. The system records the entry time, plate number, and parking lot identifier, then generates a unique ticket ID. Upon exit, the system calculates the parking duration and fee based on the recorded entry time and a rate of $10 per hour, prorated in 15-minute increments.

API Endpoints for Parking Management

You will implement two HTTP endpoints:

- POST /entry?**plate=**123-123-123&**parkingLot=**382
    - Returns ticket id
- POST /exit?**ticketId=**1234
    - Returns the license plate, total parked time, the parking lot id and the charge
    - Fees are calculated in 15-minute increments, at a rate of $10/hour

Implementation Task: Deployment Options

Build a system that would track and compute cars entry & exit from parking lots, as well as compute their charge. The system should be deployed to AWS in one of two ways:

As a serverless solution

as covered in Lesson 4

Deployed on an EC2 instance

as covered in Lesson 3

Notes:

- You may use any technology stack you like (Node.js, Python, C#, JVM, etc).
- Data persistence is left at your discretion, we'll cover persistence in the cloud in Lesson 5.

Project Deliverables

1. **API Endpoint Code:** Implement the /entry and /exit endpoints with proper business logic.

2. **Deployment Script:** Create bash, pulumi, terraform, other IaC, or custom code for GCP or AWS deployment.

3. **Code Repository:** Push completed code to GitHub or equivalent with documentation.

4. **Security Compliance:** Never include GCP or AWS access keys in your submission.

   <u>Inclusion of access keys in the submission will automatically reduce 25% of the grade.</u>