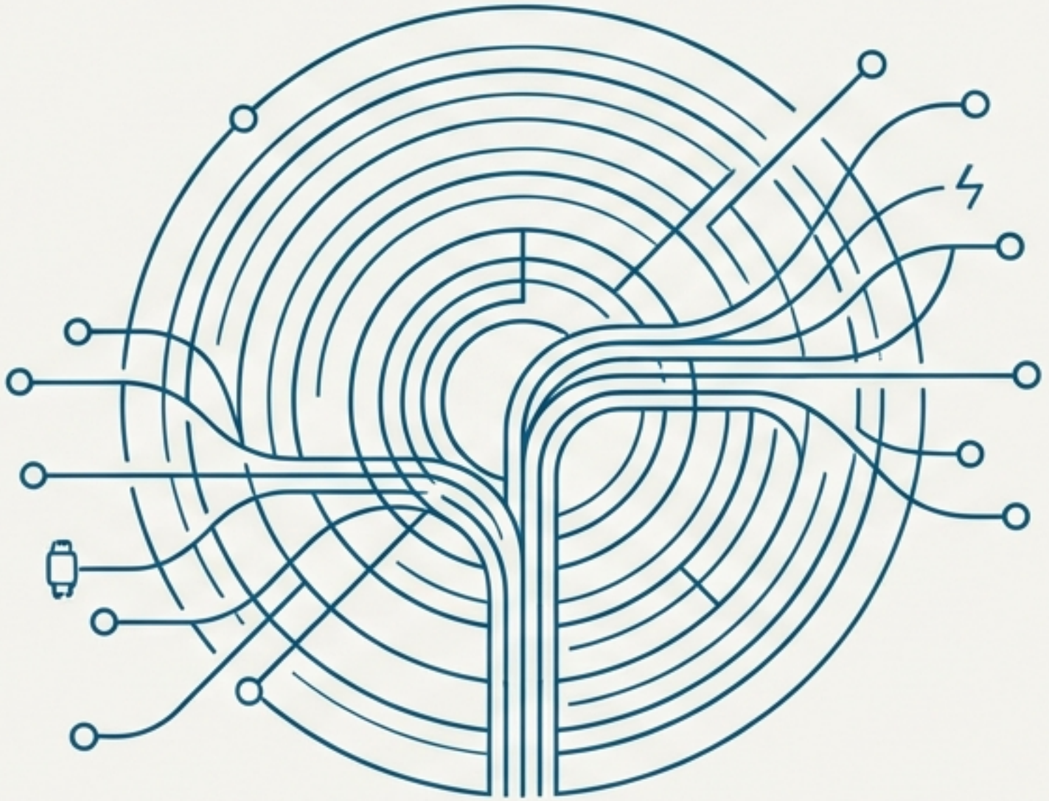
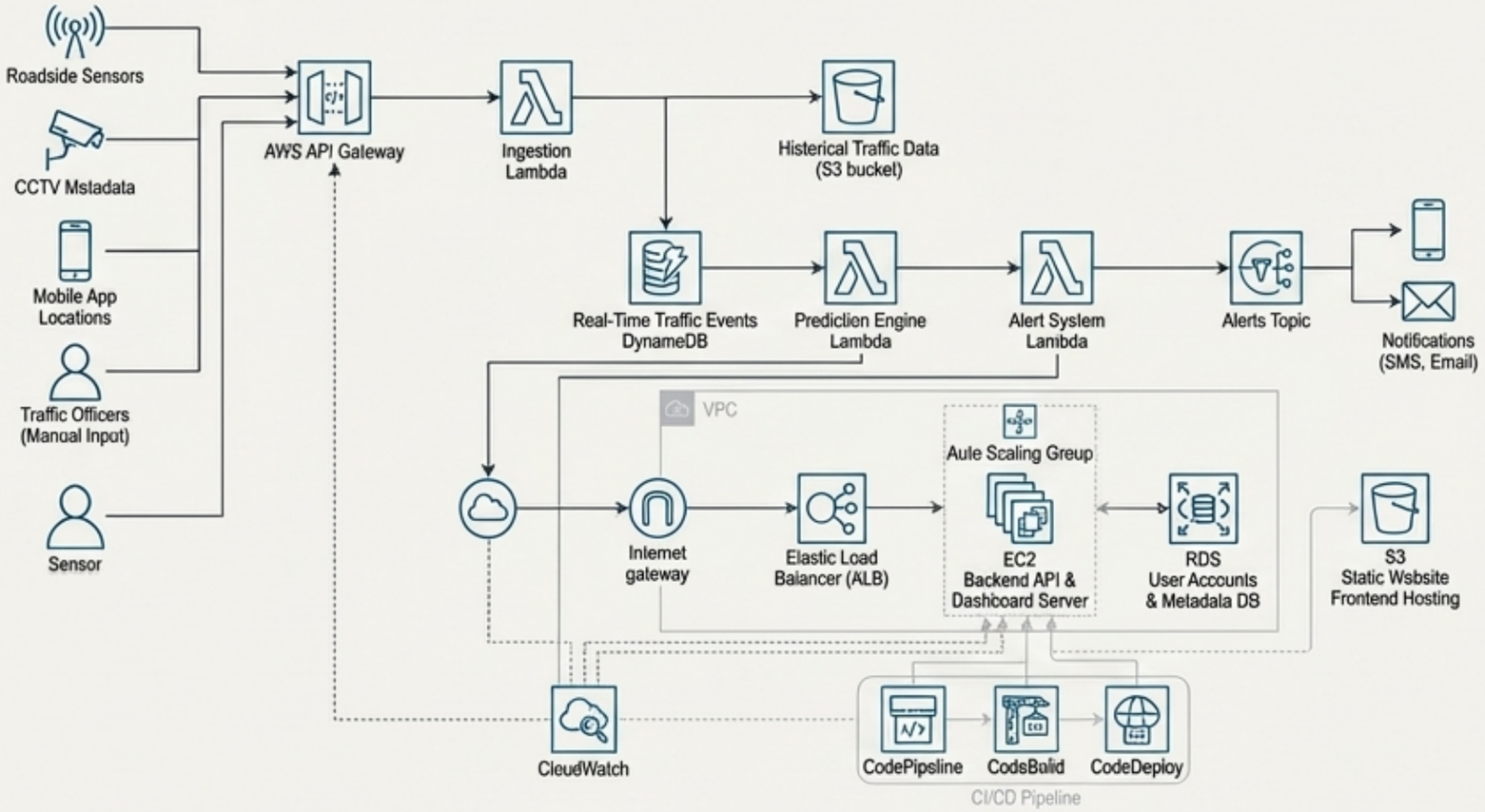
An Architectural Deep Dive

Smart City Traffic Prediction & Alert System



Built with AWS Academy Allowed Services.

This is the complete system for real-time traffic prediction and alerting.



- **Goal:** To ingest diverse, real-time traffic data from city-wide sources.
- **Function:** Predict potential congestion and incidents using a machine learning core.
- **Outcome:** Proactively alert citizens and city operators to improve traffic flow and safety.

Over the next few slides, we will deconstruct this architecture piece by piece, following the journey of data from ingestion to alert.

Part 1: Data Ingestion & Processing

The system begins by sensing the city's traffic in real time.

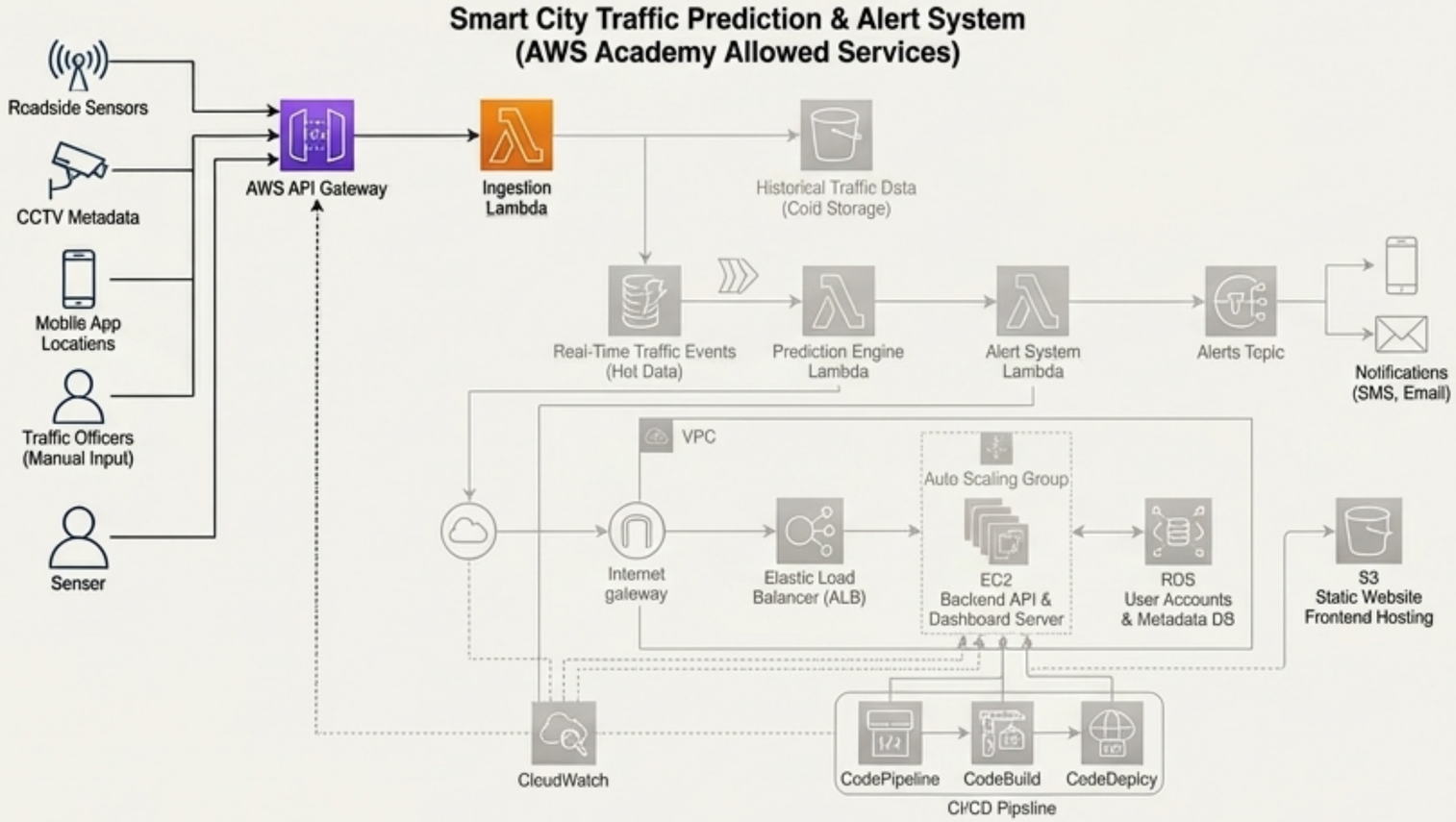
Diverse Data Sources

The system gathers data from a wide array of inputs:

- **Roadside Sensors**: Automated traffic flow and speed data.
- **CCTV Metadata**: Computer vision analysis of traffic camera feeds.
- **Mobile App Locations**: Aggregated, anonymized location data from user devices.
- **Traffic Officers (Manual Input)**: On-the-ground reports of accidents or blockages.
- **Sensor**: Generic sensor input for flexibility.

The Unified Front Door: AWS API Gateway

- **What it is**: A managed service for creating, publishing, and securing APIs.
- **Why it's here**: It provides a single, scalable, and secure entry point for all incoming data, regardless of the source. It simplifies data ingestion and protects the backend services.



Part 1: Data Ingestion & Processing

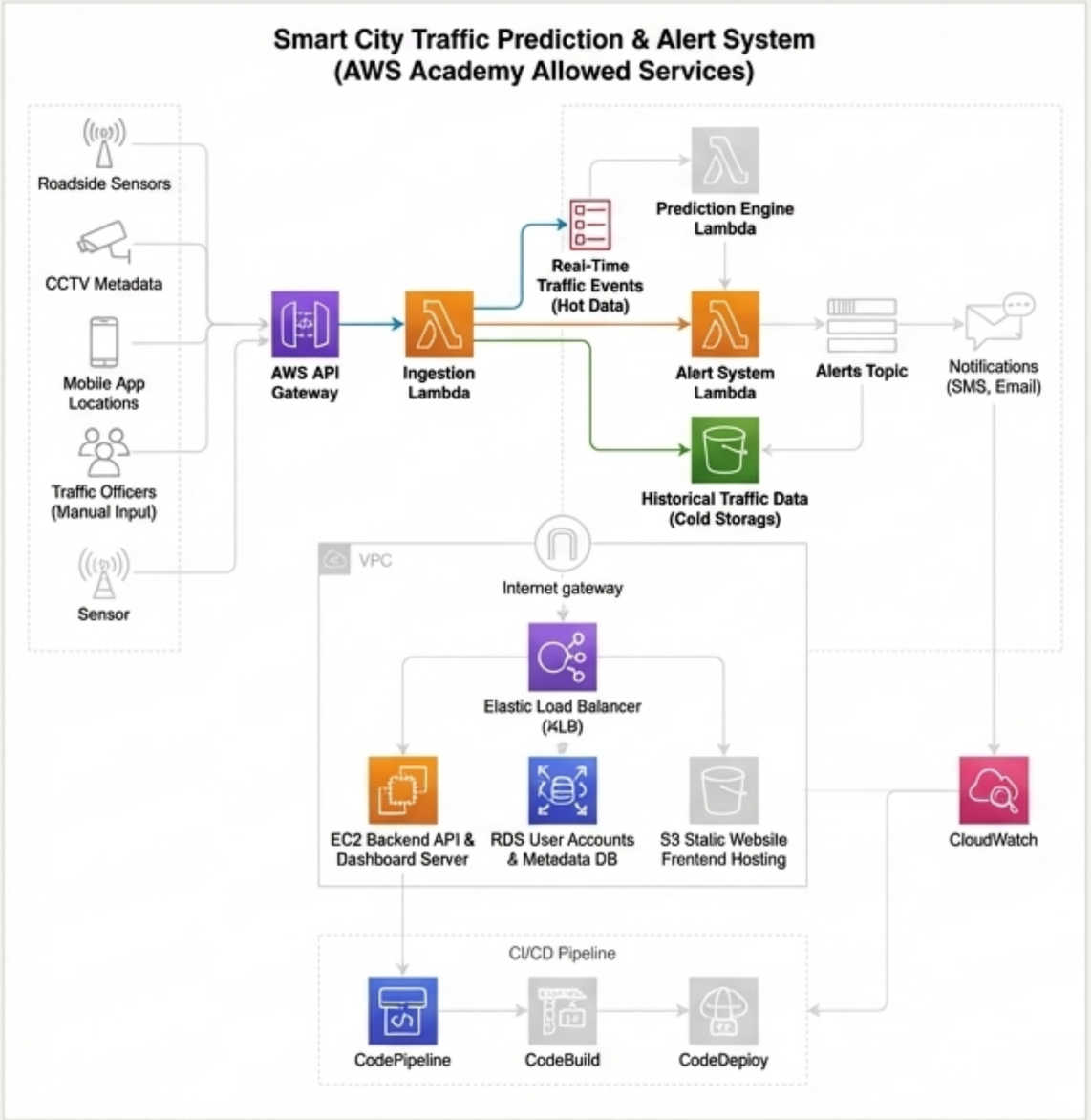
Incoming data is immediately sorted for its intended purpose.

The Initial Handler: Ingestion Lambda

- How it works: Triggered by the API Gateway, this serverless function acts as a lightweight, intelligent router. It inspects the incoming data and determines its path.

Data is split into two paths based on urgency:

1. **Hot Data: Real-Time Traffic Events**
   - **Purpose:** Data needing immediate analysis for real-time predictions (e.g., a sudden drop in sensor speed).
   - **Storage:** A fast, accessible data store designed for low-latency processing.
2. **Cold Storage: Historical Traffic Data**
   - **Purpose:** All incoming data is archived for long-term use, such as training future machine learning models and historical analysis.
   - **Storage:** An S3 bucket, providing durable, cost-effective storage.

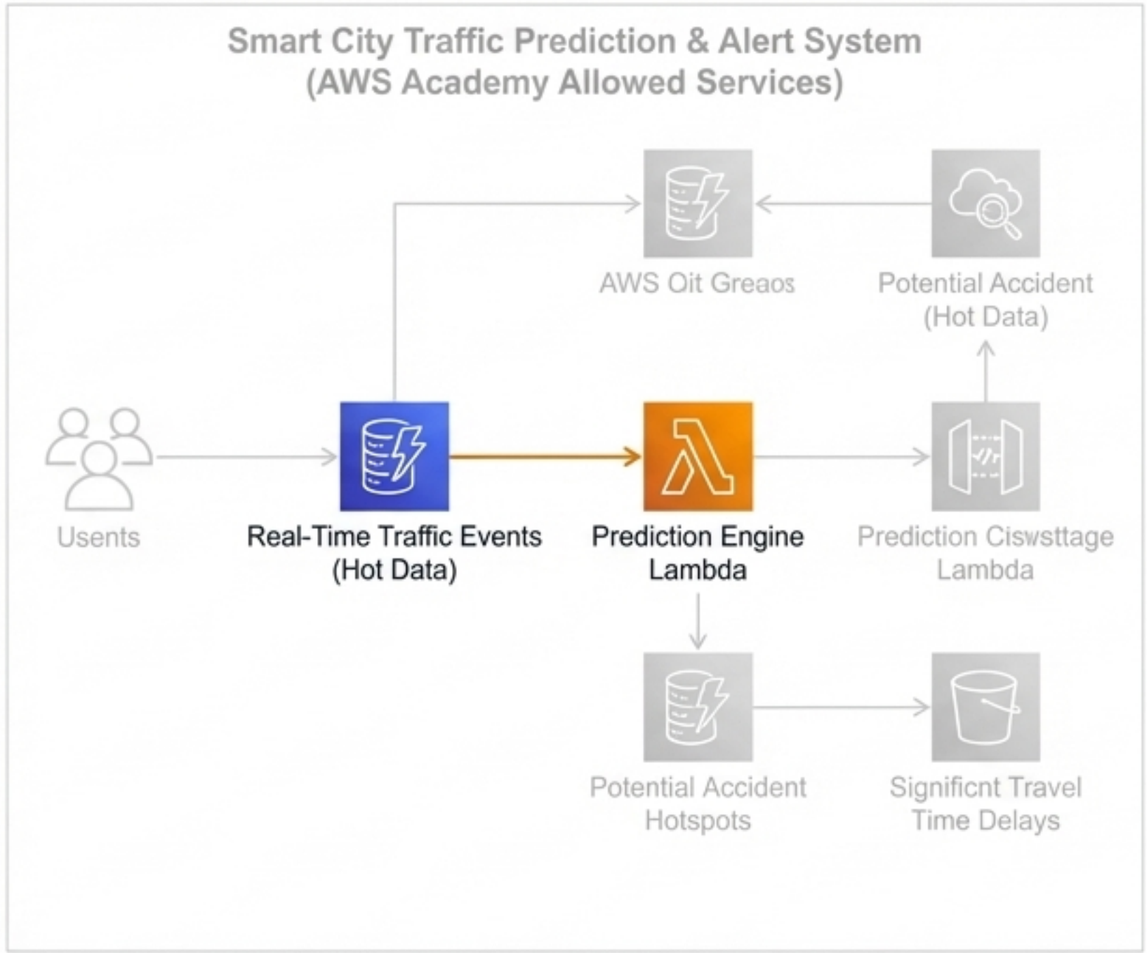


Part 2: The Prediction & Alerting Core

The system's brain analyzes real-time data to predict future traffic.

The Intelligence Core: Prediction Engine Lambda

- **What it is:** A serverless compute function that contains the system's predictive logic.
- **Why it's here:** This is where raw data becomes insight. It continuously monitors the 'Hot Data' stream to identify patterns that indicate a future problem.
- **How it works:** The Lambda is triggered by new events in the real-time data store. It runs a pre-trained machine learning model or a set of complex algorithms to forecast events like:
  - Impending traffic jams
  - Potential accident hotspots
  - Significant travel time delays



Part 2: The Prediction & Alerting Core

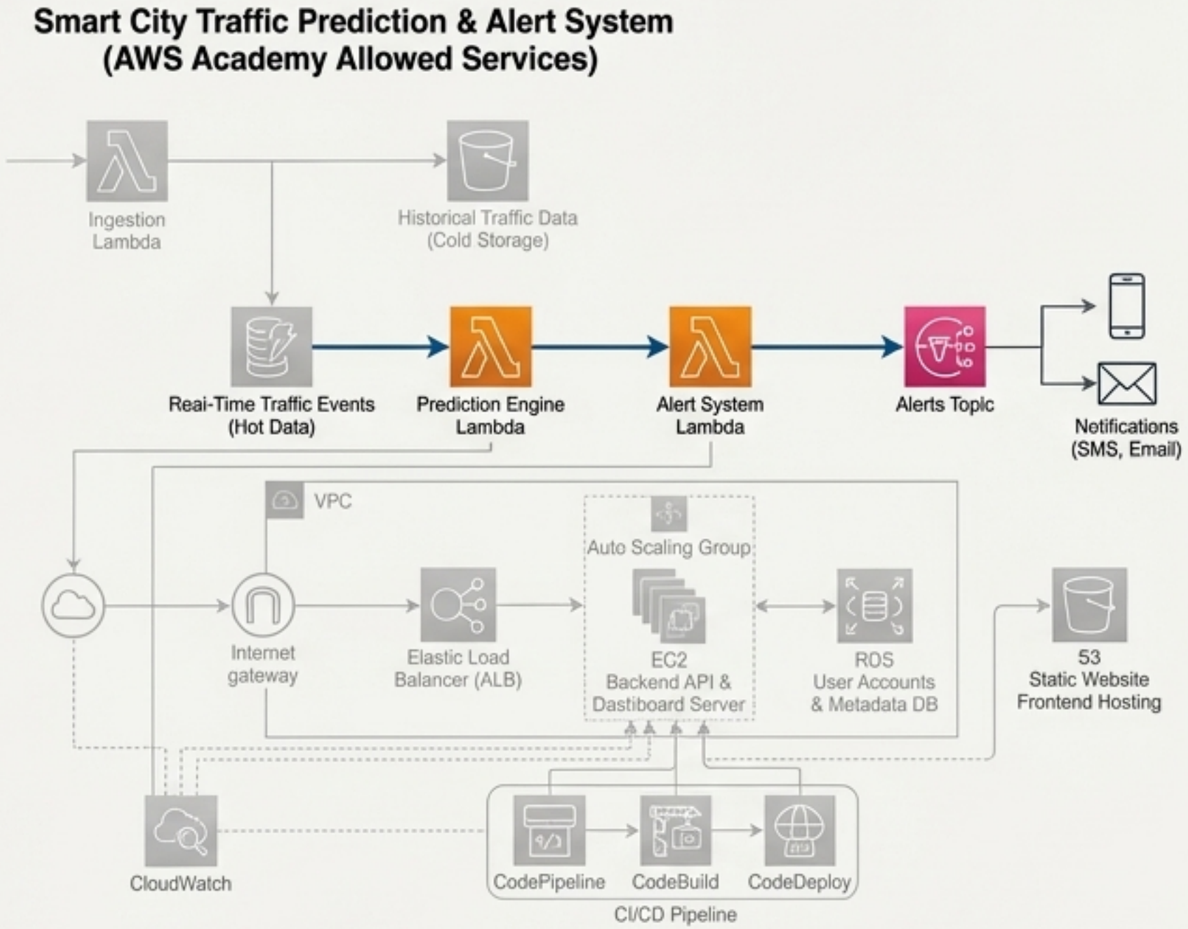
When a problem is predicted, an alert is prepared and broadcast.

1. Generating the Alert: Alert System Lambda

- **Trigger:** The Prediction Engine Lambda invokes this function when it identifies a high-probability event.
- **Function:** This Lambda's job is to format the prediction into a human-readable alert. It assesses the severity, determines the affected area, and crafts the message content.

2. Broadcasting the Message: Alerts Topic (Amazon SNS)

- **What it is**: Amazon Simple Notification Service (SNS) is a managed pub/sub messaging service.
- **How it works**: The Alert System Lambda publishes the formatted alert to a specific 'Alerts Topic.' This decouples the alert generation from the delivery, allowing multiple different systems to subscribe and react to the same alert simultaneously.



Part 2: The Prediction & Alerting Core

Critical alerts are delivered instantly to citizens and operators.

Closing the Loop

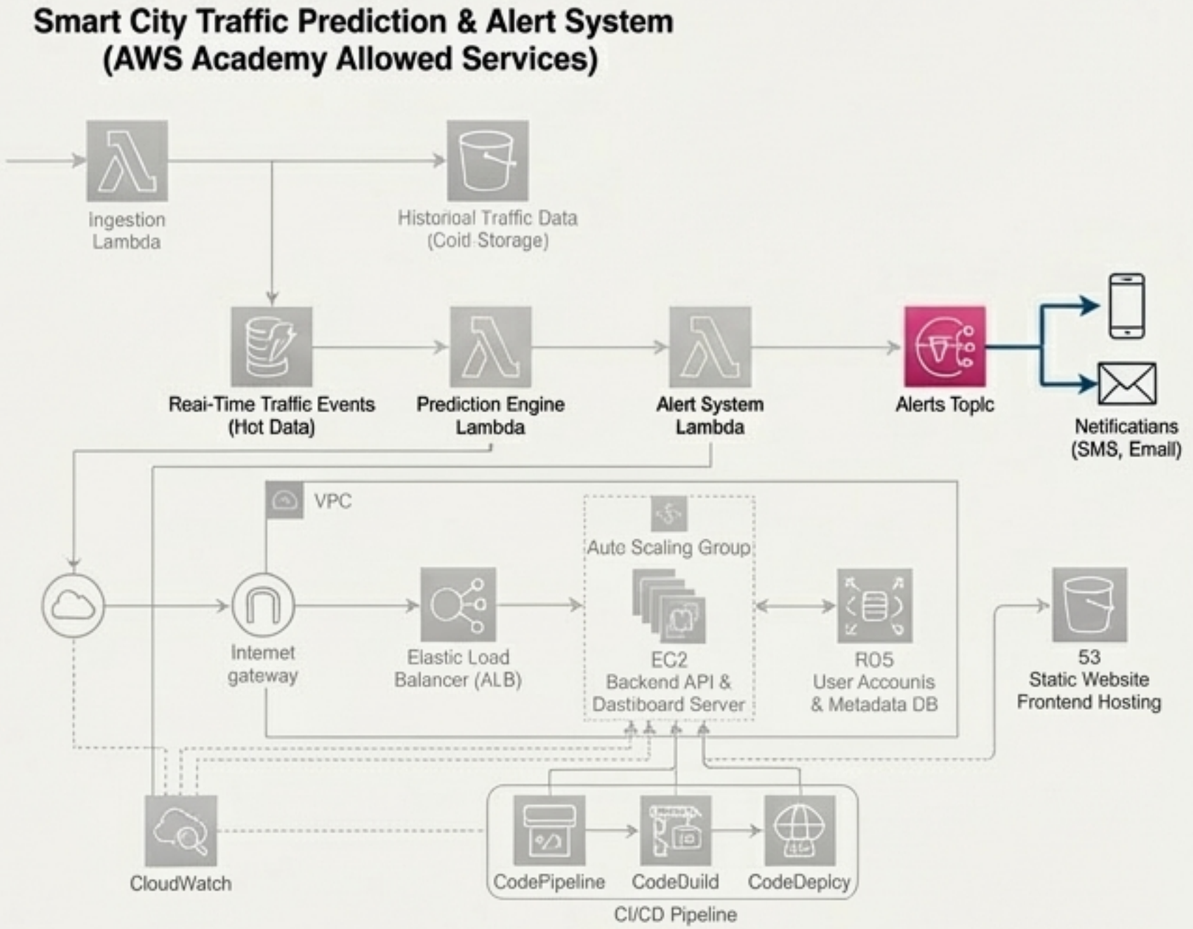
- The SNS ‘Alerts Topic’ pushes notifications to all subscribed endpoints.

Multi-Channel Delivery

- This architecture supports reaching different audiences through the most effective channels:
  - SMS: For immediate alerts to drivers on the road.
  - Email: For detailed reports to traffic management operators or city officials.
  - (Future-proofing): Other subscribers could easily be added, such as a mobile app push notification service or an API endpoint for digital road signs.

The Result

- A seamless flow from a sensor event to a notification on a user’s device in near real-time.



Part 3: The Human Interface

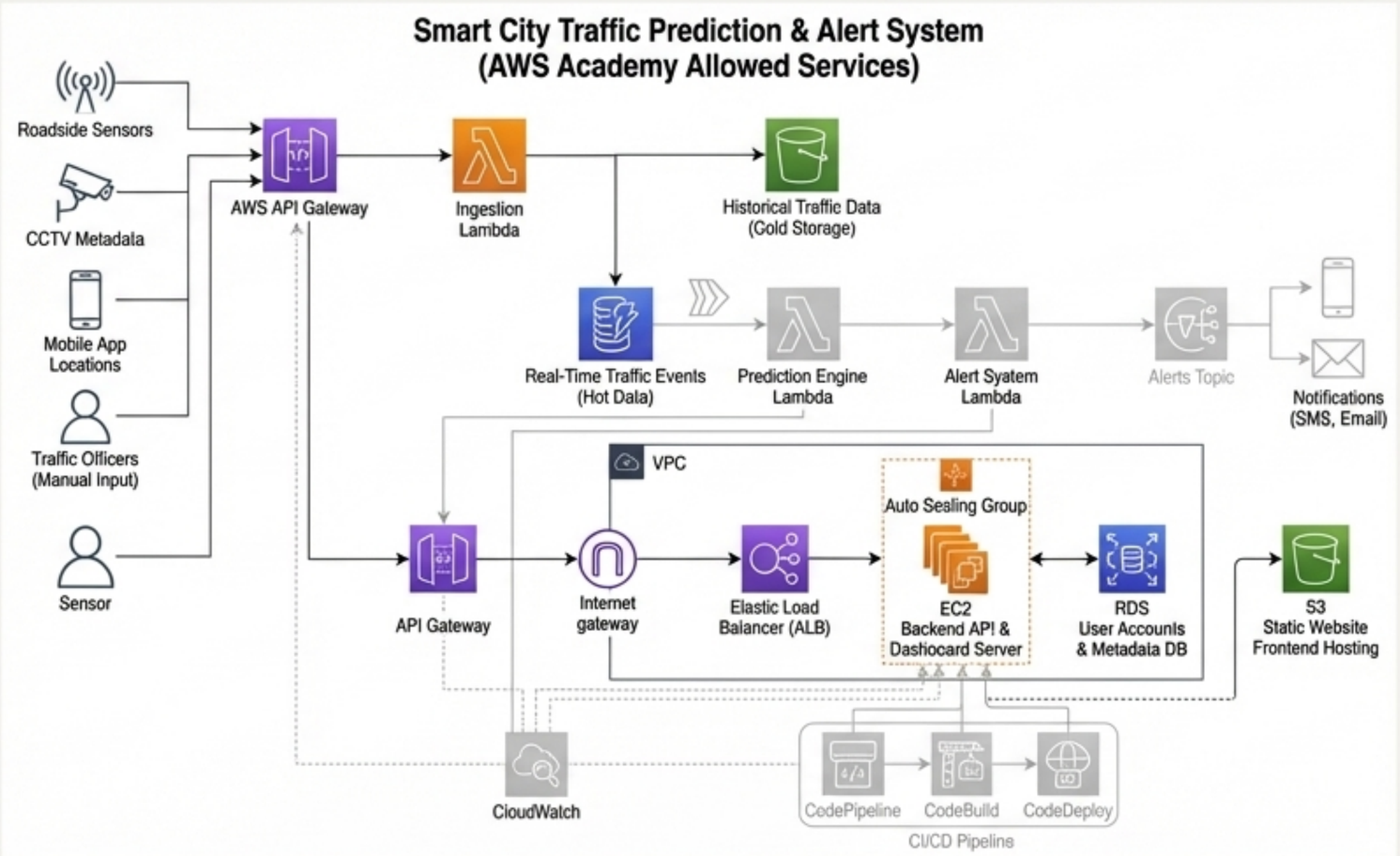
Beyond alerts, a dashboard provides visualization and control.

The Need for a Dashboard

- City operators and power users need a way to visualize historical trends, manage system settings, and explore data beyond simple alerts.

A Standard Three-Tier Web Architecture

- **Frontend (S3):** The user interface is a static web application hosted on Amazon S3 for global scalability and low cost.
- **API Layer (API Gateway):** The same API Gateway used for sensor data also exposes endpoints for the dashboard to securely fetch data.
- **Backend (EC2 & RDS):** A resilient, scalable backend processes requests and retrieves data from a relational database.

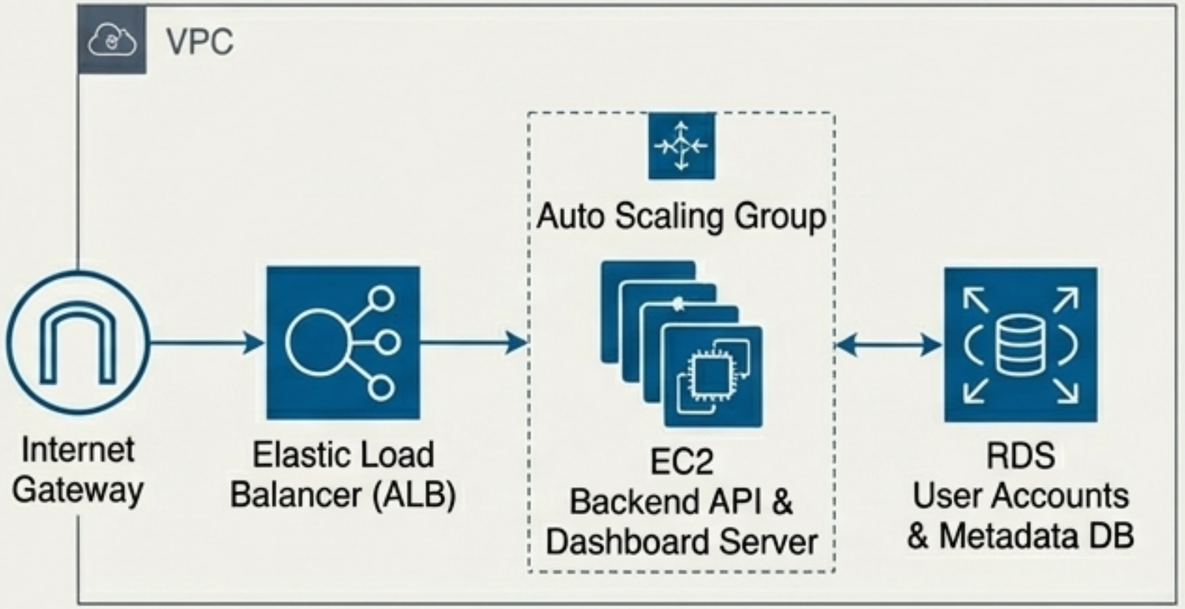


Part 3: The Human Interface

The backend is engineered for high availability and performance.

Key Components of the Backend

- **VPC (Virtual Private Cloud)**: Provides a logically isolated section of the AWS Cloud, ensuring network security.
- **Internet Gateway**: Allows communication between the VPC and the internet.
- **Elastic Load Balancer (ALB)**: Automatically distributes incoming application traffic across multiple EC2 instances, preventing any single server from being a bottleneck.
- **Auto Scaling Group**: Automatically adds or removes EC2 instances based on traffic demand, ensuring performance while optimizing costs.
- **RDS (Relational Database Service)**: A managed database that stores user accounts and application metadata. AWS manages patching, backups, and failover for high durability.



Part 4: The Operational Foundation

Automation ensures rapid and reliable system updates.

The CI/CD Pipeline

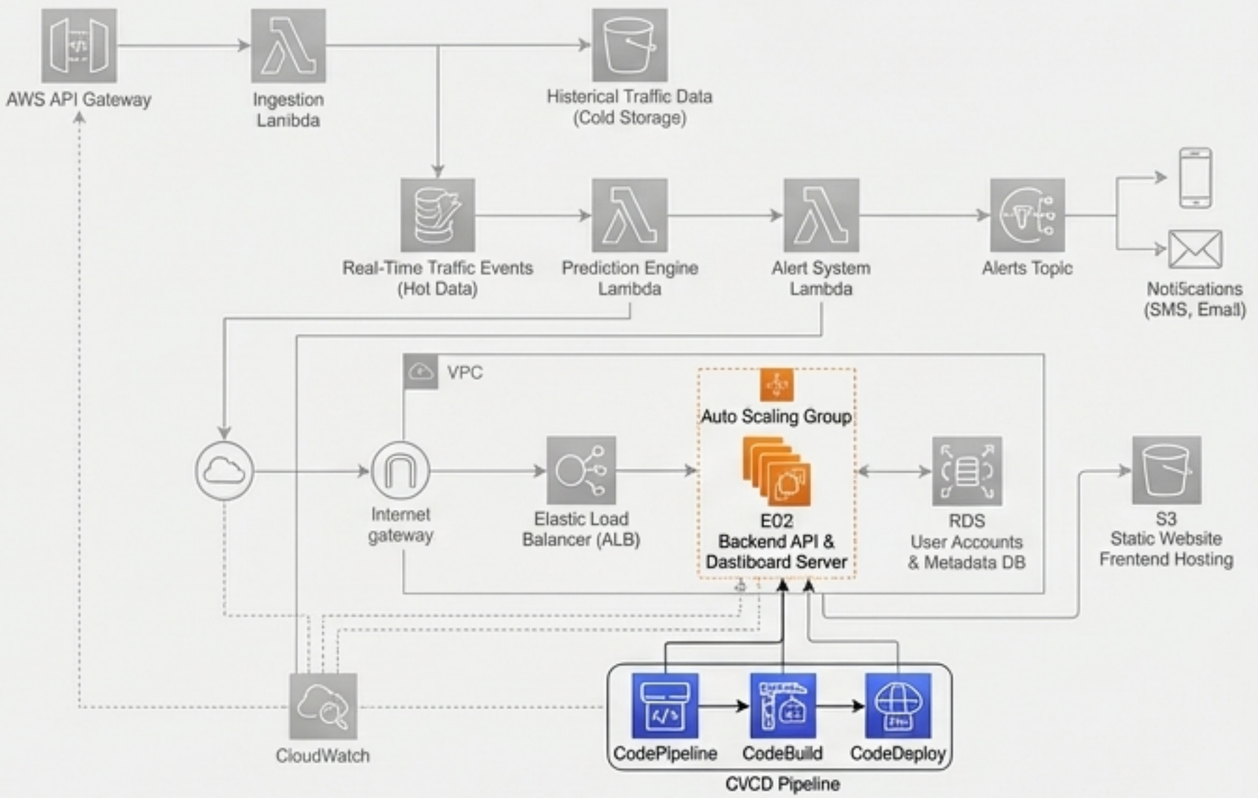
A fully automated process for software delivery that allows for safe and frequent updates to the “Backend API & Dashboard Server” running on EC2.

The Automation Workflow

- 1. **Source (CodePipeline):** The pipeline starts when a developer commits new code to a repository. CodePipeline orchestrates the entire workflow.
- 2. **Build (CodeBuild):** CodeBuild compiles the source code, runs tests, and produces software packages that are ready to be deployed.
- 3. **Deploy (CodeDeploy):** CodeDeploy automates the deployment of the new code to the EC2 instances in the Auto Scaling Group, often using strategies like rolling updates to avoid downtime.

Benefit

This automation removes manual error, accelerates release speed, and improves developer productivity.



Part 4: The Operational Foundation

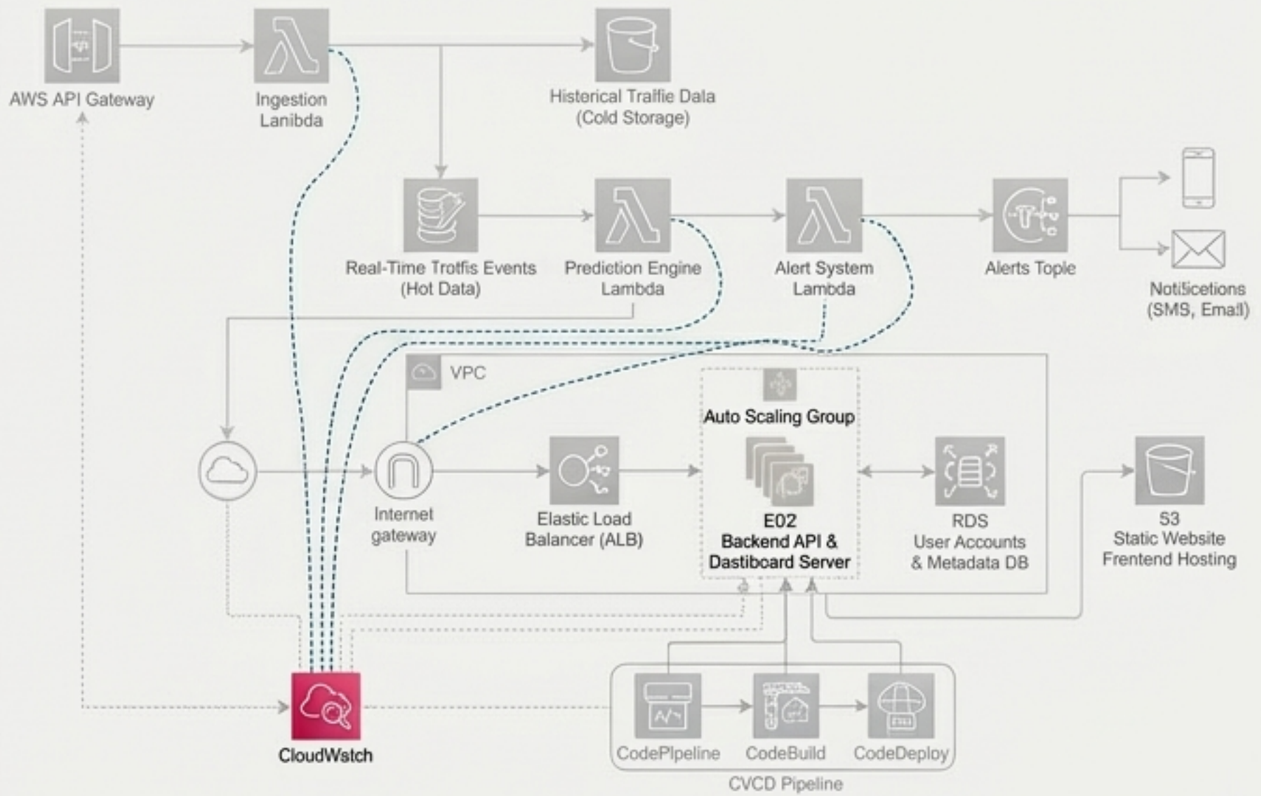
Comprehensive monitoring provides full-system observability.

The Eyes and Ears of the System: Amazon CloudWatch

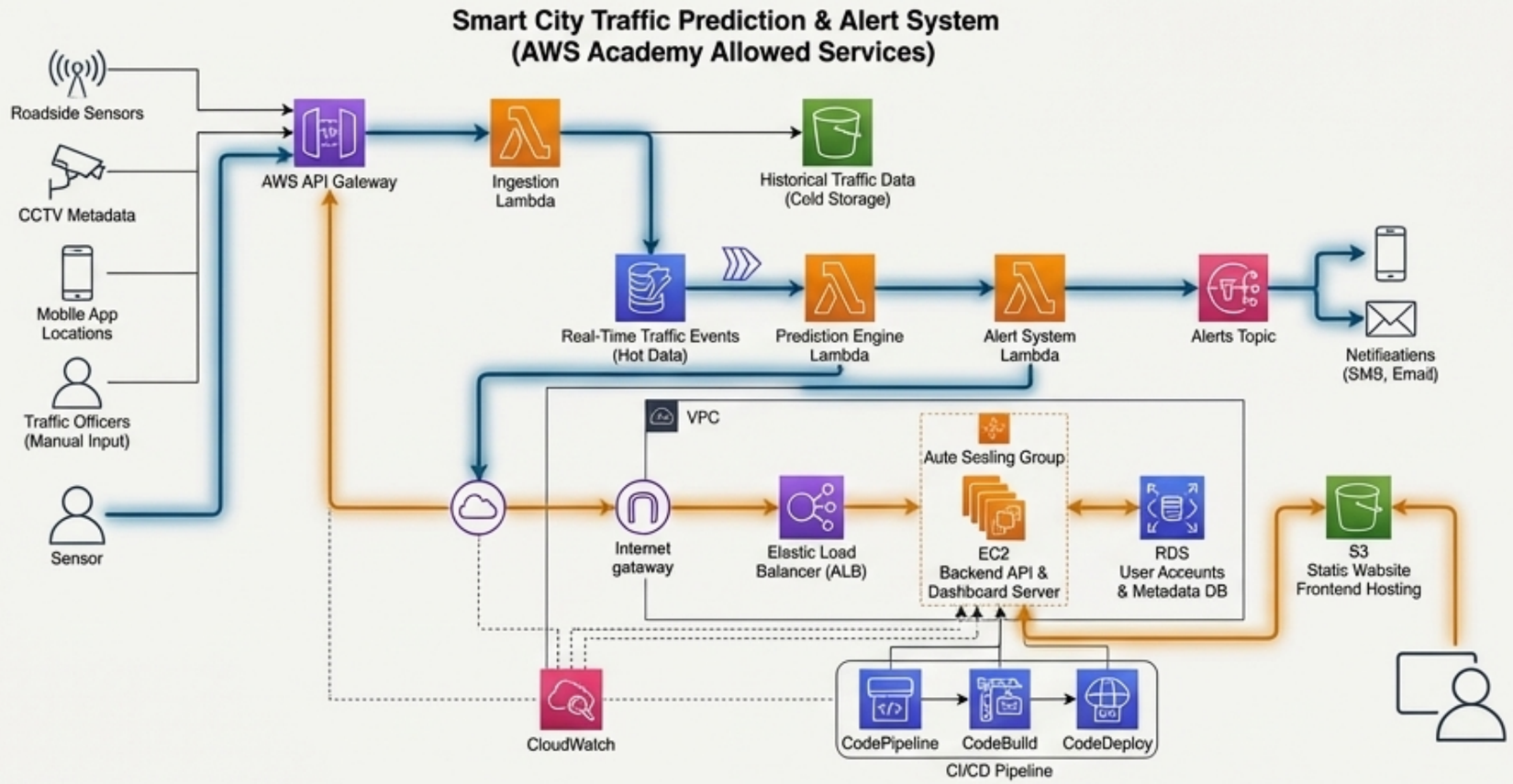
- **What it is:** A monitoring and observability service that collects data from all running AWS resources and applications.
- **Why it's essential:** In a complex, distributed system, CloudWatch provides a single place to understand system health, diagnose problems, and react to operational events.

Key Functions in this Architecture

- **Logs:** Collects logs from Lambda functions and EC2 servers for debugging.
- **Metrics:** Tracks performance metrics like Lambda invocation counts, EC2 CPU utilization, and RDS database connections.
- **Alarms:** Operators can set alarms that trigger automatically if a metric crosses a threshold (e.g., alert if the Prediction Engine Lambda error rate spikes).



The integrated system works in concert to create a smarter city.



Real-Time Processing Pipeline

A serverless, event-driven flow for low-latency prediction and alerting.

Interactive User Dashboard

A highly available, scalable web application for deep analysis and control.

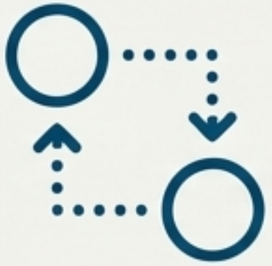
Automated Operational Foundation

Robust CI/CD and monitoring practices ensure the system is reliable, secure, and easy to maintain.

Key Principles of a Modern Cloud Architecture

Serverless-First

Using managed services like AWS Lambda, API Gateway, and SNS for the core processing pipeline reduces operational overhead, scales automatically, and follows a pay-for-what-you-use model.



Decoupled Components

Services are loosely coupled using APIs (API Gateway) and messaging (SNS). This makes the system more resilient; a failure in one component is less likely to cascade and bring down the entire system.

Scalability and Elasticity

The system is designed to handle variable loads. The serverless components scale inherently, and the web backend uses an Auto Scaling Group to match resources to traffic demand automatically.

Automation and Observability

With a full CI/CD pipeline (CodePipeline, etc.) and comprehensive monitoring (CloudWatch), the system is built for rapid, safe evolution and proactive operational management.

Thank You

Q & A