NYCPS Transportation Management System - AWS GovCloud Architecture (RFP R1804)

## **Status** : Draft

**Section Tracker**

| Section # | Section Name | Status |
| --- | --- | --- |
| 1 | Introduction | Draft |
| 2 | System Overview | Draft |
| 3 | System Architecture | Draft |
| 4 | Throughput & Scalability | NotStarted |
| 5 | Data Model & Management | Draft |
| 6 | Network & Security | NotStarted |
| - | Anything Else? | NotStarted |
| - | Anything that needs to be pulled back from the Archive Section? | NotStarted |

## 1. Introduction

This document outlines the technical architecture for the NYCPS Transportation Management System, designed to meet the functional requirements specified in the Solution Design and the stringent non-functional requirements (performance, scalability, security, availability, compliance) outlined in RFP R1804. The architecture prioritizes deployment within AWS GovCloud (US) regions, leveraging managed services where possible to enhance reliability, scalability, and security while adhering to necessary compliance standards (e.g., FedRAMP High, ITAR, CJIS, HIPAA as applicable within GovCloud).

New Section - Add an Exec Summary with these details

* (Maybe) Mission & Vision 1 liner
* How is the document organized - add a table with section and purpose
* Mention this is high level design , and more detailed info with links are added to Appendix X

### 1.1 Abbreviations, Acronyms and Definitions

| Abbreviations | Description |
| --- | --- |
| SMBS | School Bus Management System |
| AWS Gov Cloud | **AWS GovCloud (US)** is designed for US government agencies and regulated industries with specific features:  • Compliance:  ○ ITAR compliance \* FedRAMP certified  ○ DoD SRG Impact Levels 2, 4, and 5  ○ HIPAA, PCI-DSS compliant  • Key Characteristics:  ○ Physically located in US  ○ Operated by US citizens  ○ Accessible to US entities only  ○ Separate and isolated from standard AWS regions  ○ Supports most AWS services  • Use Cases:  ○ Government workloads  ○ Healthcare data  ○ Financial services  ○ Defense applications  ○ Sensitive data processing |
| AWS ECS Cluster Container | **An Amazon ECS cluster** is a logical grouping of tasks or services. Key components include:  • Infrastructure capacity combining:  ○ Amazon EC2 instances  ○ Serverless (AWS Fargate)  ○ On-premises VMs/servers  • Core elements:  ○ Tasks - Groups of containers that run together  ○ Services - Applications maintaining specified number of tasks  ○ Network (VPC and subnet)  ○ Optional namespace for service communication  ○ CloudWatch Container Insights for monitoring  ECS clusters are region-specific and help separate resources. They enable efficient container management, scheduling, and deployment of microservices architectures while providing seamless integration with other AWS services like ELB and Auto Scaling. |
| AWS ECS EC2 Container | **With ECS on EC2 launch** type, you:  • Have complete control over EC2 instances in your cluster  • Run containers on EC2 instances you manage  • Are responsible for provisioning, patching, and scaling servers  • Can choose specific EC2 instance types and customize the OS  • Use Auto Scaling Groups to manage EC2 capacity  • Need to install ECS container agent (included in ECS-optimized AMI)  Your containers are defined in task definitions that specify parameters like CPU, memory, ports, and IAM roles. ECS handles container scheduling and placement across your EC2 instances based on your resource requirements. |
| AWS Router 53 | **Amazon Route 53** is a highly available and scalable cloud DNS web service. Key features include:  • Domain registration and management  • DNS routing with multiple options:  ○ Latency-based routing  ○ Geolocation routing  ○ IP-based routing  ○ Weighted routing  • Health checking and DNS failover  • Route 53 Resolver for VPC DNS resolution  • Integration with other AWS services\* DNS Firewall for filtering outbound DNS traffic |
| Amazon API Gateway | Amazon API Gateway is a fully managed service for creating, publishing, and managing APIs. Key features:  • API Types:  ○ REST APIs  ○ HTTP APIs (faster and lower cost)  ○ WebSocket APIs  • Capabilities:  ○ Request/response transformation  ○ API versioning  ○ Security (IAM, Lambda authorizers)  ○ Throttling and caching  ○ SDK generation  ○ API documentation  • Benefits:  ○ Serverless architecture  ○ Pay-per-use pricing  ○ Automatic scaling  ○ Edge caching  ○ Monitoring and logging  ○ Global deployment options |
| Amazon Managed Streaming Kafka | Amazon Managed Streaming for Apache Kafka (Amazon MSK) is a fully managed service that helps you build and run applications using Apache Kafka for streaming data processing. Key benefits include:  • No server management required - AWS handles provisioning, configuration, and maintenance  • High availability with multi-AZ deployment  • Built-in security with IAM integration  • Easy scalability to support load changes  • Native integration with AWS services like Lambda, Glue Schema Registry, and Amazon Managed Service for Apache Flink\* Pay-for-what-you-use pricing model  Amazon MSK eliminates the operational overhead of managing Kafka infrastructure, letting you focus on building streaming applications. |
| EKS | **Amazon Elastic Kubernetes Service** (EKS) is a fully managed Kubernetes service provided by AWS. It simplifies the deployment, scaling, and management of containerized applications by automating the setup and operation of Kubernetes clusters. Key features include:  • Integration with AWS Services: EKS seamlessly integrates with services like EC2, S3, IAM, and VPC for networking, storage, and security.  • Flexible Deployment: Applications can run on AWS infrastructure (EC2 or Fargate) or on-premises using EKS Anywhere.  • Ease of Use: Reduces operational overhead by automating updates, scaling resources, and patching nodes |
| Promtail | **Promtail** is a lightweight log collection agent that:  • Reads logs from local files, systemd journals, or application containers.  • Enriches logs by attaching metadata (e.g., pod name, namespace, labels).  • Sends logs to a Grafana Loki instance for storage and analysis. |
| Grafana Loki | **Grafana Loki** is an open-source log aggregation system. Key features include:  • Log storage: Logs are compressed and stored in chunks, with only metadata (labels) indexed.  • Scalability: Loki scales horizontally by distributing logs across multiple ingesters, ensuring high availability and efficient handling of heavy log traffic. It prevents data loss during node failures and redirects logs to healthy nodes.  • Log Compression: Logs are compressed into chunks, reducing storage space requirements and improving retrieval times. Indexed logs allow for quick querying based on timestamps and labels  • Integration with Grafana: Loki seamlessly integrates with Grafana, allowing visualization of logs alongside metrics and traces in a unified dashboard |
| Prometheus | Prometheus is an open-source systems monitoring and alerting toolkit that collects and stores time-series data, offering a powerful and flexible solution for monitoring cloud-native applications and infrastructure. |
| Grafana | Grafana is an open-source, interactive web application used for monitoring and visualizing data, allowing users to create customizable dashboards, charts, and graphs from various data sources |
| Aurora | **Amazon Aurora** is a cloud-native relational database offering key features:  • Performance:  ○ 5x throughput of MySQL, 3x of PostgreSQL  ○ Compatible with MySQL and PostgreSQL  ○ Auto-scaling storage up to 128 TiB  • High Availability:  ○ Data replicated across 3 Availability Zones  ○ 6 copies of data  ○ Automated failover  ○ Up to 15 read replicas  • Cost-effective:  ○ 1/10th cost of commercial databases  ○ Pay for what you use  ○ Serverless option available  • Management:  ○ Fully managed by AWS  ○ Automated patching, backup, and recovery  ○ Built-in monitoring and security |
| nosql | AWS offers these main NoSQL database services:  • **Amazon DynamoDB:**  ○ Serverless key-value and document database  ○ Single-digit millisecond latency  ○ Automatic scaling  ○ Global tables capability  • **Amazon DocumentDB:**  ○ MongoDB-compatible document database  ○ Flexible schema and indexing  ○ Millions of requests per second  ○ Best for content management, catalogs, user profiles  **MongoDB Atlas** is a fully managed cloud database service provided by MongoDB  ○ Fully managed cloud database service  ○ Multi-cloud and global deployment support  ○ Automatic scaling and performance optimization  ○ Automated backups with point-in-time recovery  You can use these databases independently or combine them with SQL databases for optimal performance based on your workload needs. |
| Redis ElastiCache | **Amazon ElastiCache for Redis** is a fully managed, in-memory data store and cache service provided by AWS  • Provides microsecond latency for real-time applications  • Provides up to 80x faster read performance compared to disk-based databases  • Offers auto-scaling capabilities  • Reduce database query costs by >50% |
| S3 | **Amazon Simple Storage Service** (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance  • Low cost Data storage and backup  • High durability, availability and Security |
| AWS CloudWatch | **Amazon CloudWatch** is a monitoring and management service that provides:  • Real-time monitoring of AWS resources and applications  • Collection and tracking of:  ○ Metrics  ○ Logs  ○ Events  ○ Application performance data  • Key capabilities:  ○ Create custom dashboards  ○ Set automated alarms  ○ Collect data from on-premises, hybrid, and cloud infrastructure  ○ Take automated actions based on predefined thresholds  ○ Monitor complete stack (applications, infrastructure, network)  ○ Access via API, CLI, SDKs, or AWS Console  CloudWatch helps you gain system-wide visibility, optimize resource utilization, and reduce mean time to resolution (MTTR) for operational issues. |
| DNS Logs | **Route 53 DNS** query logging provides:  • Information logged:  ○ Domain/subdomain requested  ○ Date and time  ○ DNS record type (A, AAAA)  ○ Route 53 edge location  ○ DNS response codes  • Logs can be sent to:  ○ CloudWatch Logs (default)  ○ Amazon S3  ○ Kinesis Data Firehose  • Features:  ○ Logs available within minutes of queries  ○ Log group must be in US East (N. Virginia)  ○ Only logs queries forwarded to Route 53  Helps monitor DNS traffic and troubleshoot issues |
| AWS Cloudtrail Logs | **AWS CloudTrail provides:**  • Event logging of AWS account activity:  ○ Management events (API calls, console actions)  ○ Data events (S3 object-level activities)  ○ Network activity events  ○ Insights events for unusual API activities  • Key features:  ○ 90-day event history by default  ○ Logs delivered to S3 bucket  ○ Optional CloudWatch Logs integration  ○ Multi-region and multi-account logging  ○ Immutable audit trail  ○ Security analysis and compliance auditing  ○ Resource change tracking  CloudTrail helps answer "who did what, where, and when?" by recording user activity and API calls across AWS services. |
| AWS VPC Flow Logs | **Amazon Virtual Private Cloud (VPC)**.  1. Captured Information:  ○ Source and destination IP addresses  ○ Source and destination ports  ○ Protocol  ○ Number of packets and bytes transferred  ○ Time interval during which the flow was observed  ○ Action (accepted or rejected traffic)  2. Logging Options: VPC Flow Logs can be published to: [2]  **○ Amazon CloudWatch Logs**  ○ Amazon S3  ○ Amazon Kinesis Data Firehose  3. Scope: Flow logs can be created at different levels:  ○ VPC level  ○ Subnet level  ○ Network interface level  4. Use Cases:  ○ Diagnosing overly restrictive security group rules  ○ Monitoring traffic reaching your instances  ○ Determining traffic direction (inbound/outbound)  ○ Enhancing network security analysis  5. Performance Impact: VPC Flow Logs do not affect network performance or latency as they are collected outside the network traffic path. [3]  6. Limitations: Some types of traffic are not logged, such as:  ○ Traffic to Amazon DNS servers  ○ Windows instance license activation  ○ Instance metadata  ○ DHCP traffic  ○ Traffic to reserved IP addresses for the VPC router  7. Security and Compliance:  ○ Helps in monitoring and auditing network traffic for compliance requirements  ○ Assists in identifying potential security issues or unauthorized access attempts  8. Cost Considerations: There are charges associated with using VPC Flow Logs. For current pricing details, please refer to the AWS documentation.  9. Integration: VPC Flow Logs can be integrated with other AWS services for further analysis and visualization of network traffic patterns.  When implementing VPC Flow Logs, it's important to consider your specific security and monitoring needs, and to configure the logs according to the principle of least privilege. Always test your configurations in a non-production environment before deploying to production systems. |
| AWS VPC Flow Logs | **AWS Shield** is a managed DDoS protection service that safeguards applications running on AWS. It comes in two tiers:  • AWS Shield Standard:  ○ Free, automatic protection against common Layer 3 and 4 DDoS attacks  ○ Always-on detection and automatic inline mitigations  ○ Protects AWS services like CloudFront and Route 53  • AWS Shield Advanced:  ○ Enhanced protection against sophisticated Layer 3-7 attacks  ○ Protects EC2, ELB, CloudFront, Global Accelerator, and Route 53  ○ Includes 24/7 access to AWS Shield Response Team  ○ Provides cost protection against DDoS-related spikes  Offers real-time attack visibility through **CloudWatch** |
| AWS WAF | **AWS WAF (Web Application Firewall)** helps protect web applications from common exploits. Key features include:  • Web traffic filtering based on:  ○ IP addresses  ○ HTTP headers and body  ○ Custom URIs  ○ SQL injection patterns  ○ Cross-site scripting  • Can be deployed on:  ○ Application Load Balancer  ○ Amazon API Gateway  ○ CloudFront  ○ AppSync for GraphQL APIs  • Includes Bot Control to manage bot traffic  • Offers managed rules that are automatically updated  • Allows creation of custom security rules  • Enables centralized rule management across multiple websites |
| AWS GuardDuty | **Amazon GuardDuty** is a threat detection service that continuously monitors your AWS environment for:  • Malicious activity and unauthorized behavior  • Account compromise and credential theft attempts  • Data exfiltration and potential ransomware events  • Unauthorized cryptomining\* Malware in EC2 instances and container workloads  • Suspicious database login patterns  • Malicious files in S3 buckets  Key features:  • Uses AI/ML and threat intelligence  • Analyzes AWS CloudTrail, VPC Flow Logs, and DNS logs  • Integrates with AWS Organizations for multi-account monitoring  • No additional security software needed  • Continuous monitoring without performance impact  Accurate account-level threat detection |
| AWS Detective | **Amazon Detective** automatically collects log data from your AWS resources and uses machine learning (ML), statistical analysis, and graph theory to build a dataset that you can use to conduct more efficient security investigations.  Key features:  • Determine potential security issues through a unified view of user and resource interactions.  • Save time and effort with graph models that automatically summarize security-related relationships running on AWS.  • Investigate and respond to security findings with streamlined visualizations.  • Accelerate security investigations with generative AI insights to more quickly comprehend threats |
| AWS Event Bridge | **Amazon EventBridge** is a serverless event bus service that helps you build event-driven applications. Key features include:  • Integration with 200+ AWS services and SaaS providers  • EventBridge Pipes for point-to-point integrations  • EventBridge Scheduler for task automation  • Built-in filtering and transformation capabilities  • JSON-based event structure  • Multiple targets including Lambda, SQS, SNS, Kinesis  • Support for event-driven architecture  • SaaS provider integration over private AWS network\* Pay-per-use pricing model  • Real-time event routing and delivery  EventBridge extends beyond AWS services to bring external SaaS data into your AWS environment, eliminating the need for polling or custom webhooks. |
| Amazon Simple Notification Service | **Amazon SNS** is a fully managed pub/sub messaging service that enables:  • Application-to-application messaging  • Application-to-person notifications via:  ○ Mobile push notifications  ○ SMS  ○ Email  ○ HTTP/HTTPS endpoints  • Integration with AWS services like SQS, Lambda, and CloudWatch  • Standard and FIFO topic types  • Message filtering and data protection\* Pay-as-you-go pricing with no upfront costs  • Secure message delivery using access control and encryption  • High availability and durability across multiple AZs  SNS helps build distributed applications and microservices architectures while e |
| AWS Secret Manager Data Encryption | **AWS Secret Manager** encrypts data using:  • Encryption in transit:  ○ TLS-protected channels between AWS internal systems  ○ Secure data transfer from EC2 instances to AWS  ○ Protected telemetry data collection  • Encryption at rest options:  ○ AWS owned CMK (default)  ○ Customer managed CMK  ○ AWS KMS keys for customized encryption  Systems Manager integrates with AWS KMS for key management and uses envelope encryption to protect sensitive data. All communication between components is encrypted using TLS to ensure data security during transmission. |
| React utilizing JS | **React** is a JavaScript library for building user interfaces, particularly for web applications, that uses a component-based architecture to create interactive and dynamic UIs efficiently |
| Next.js | **Next.js** is a Web Framework built on React. It allows for seamless integration between the API End and the Frontend Webpage. NextJS is one of the most used React Web Frames, and provide optimizations and many useful Utilities for building Web Pages of all sizes and complexities  **Server Side Render** and **HTML streaming** provides substantial performance gains  **Static Site Generation** allows for Prebuilding of pages to make simple pages load near instantly |
| SpringBoot / Java | Java Backend Framework known for its ability to streamline the development of robust applications through simplified Configuration, Increased productivity, microservice support, large ecosystem and community |
| Apache HTTP Server | **Apache HTTP Server** is a Free open source Cross-platform web server  Can support Various Languages  Most Popular Web Servers in the world  Known for its Security Features and large community that work to improve security |
| React-Native / JS | **React-Native** isa JavaScript framework developed by Meta Platforms (formerly Facebook Inc.) that enables developers to build native mobile apps for Android and iOS using a single codebase, leveraging the React framework and native platform capabilities |
| Amplify / Javascript | **AWS Amplify** is a development framework and hosting service that offers:  • Development Features:  ○ Libraries and UI components  ○ CLI for backend setup  ○ Authentication via Cognito  ○ Storage with S3  ○ APIs (GraphQL/REST)  ○ Analytics and push notifications  • Hosting Capabilities:  ○ Full-stack web application hosting  ○ Built-in CI/CD workflows  ○ Support for React, Angular, Vue, Gatsby  ○ Serverless architecture  ○ Automatic HTTPS  • Key Benefits:  ○ Quick setup and deployment  ○ Scalable infrastructure  ○ Pay for what you use  ○ Multiple storage bucket support  Built-in security features |

New Section - Start System Overview Here

* Needs to include the conceptual Flow Diagram with some basic interfacing like DBs and message bus
* Separation of FrontEnd, Backend, Security and External Integration
* Design Consideration - Embed the Architecture Pattern section
* Scope (maybe?) and summary?

## 2. System Overview

TODO HOLD : //Might not need this section if we doing the solution overview on another doc.

TODO - Check if the tech contents of this [AWS Microservice Workflow](https://docs.google.com/document/d/1PINjhjuAF72zLTZ3RNF13rFjrgTyNNCYqJ7FpnZmoto/edit?tab=t.0#heading=h.tuh56a8qk98o) can be added here

**Web Portal**

Users will access the Web portal through Route 53 a Domain Registration and management AWS service. To login, users will be directed to the DOE Identity provider SSO, and provided a token for the Sentry Web Portal. They will be routed through our AWS shield and AWS Web Application Firewall, which offers DDoS protection, Traffic filtering based on Ip Addresses, HTTP Headers/Body, Custom URIs, SQL Injection patterns and Cross Site Scripting. AWS CloudWatch will be leveraged to provide real-time monitoring of our Web server, collecting and tracking Metrics, logs, events and performance data. Displaying such data on custom Dashboards and triggering Automated Alarms if any configured criteria is met.

Once through, the user will be passed through the Network Load Balancer to the Application Load Balancers, and finally to the Apache Web Service hosting the Web Portal's Application. The Web Service will be housed on a Public Subnet, within an ECS Cluster Containers EC2. From here any requests that require the backend will be routed to the Parallel Private Subnet ECS Cluster Containers EC2, where our Backend jobs will be running pushing and pulling data to and from our Databases. Jobs will push data to an Elastic Kubernetes Service housing our Prometheus DB for Logging and Grafana for Reporting/Dashboards. Both tools are Open source reducing overall cost.

Data will be pushed from our Web Backend to Aurora for RDS. When the backend fetches data, it will first check the Redis ElastiCache for matching recent queries that have already been saved to the in memory data store before going to Aurora. This caching layer helps improve the application performance by retrieving frequently accessed data quickly, reducing database load and latency.

**Driver Mobile App**

Users will authenticate with DOE IDP to get into the driver application, routed through our NLB then the ALB before passing through to the Public Subnets Amplify Service. Amplify is a dev framework and hosting services that offers Serverless Architecture for Applications running in AWS. Once on Amplify, users are rooted through the Web Application Firewall which offers DDoS protection, Traffic filtering based on IP Addresses, HTTP Headers/Body, Custom URIs, SQL Injection patterns and Cross Site Scripting. After Security, users requests will hit the AWS API Gateway for all Rest, HTTP and Websock API routing to the Backend server running on the Private Subnet within an EC2.

Dynamic Data/Events Data will be pushed to Dynamo DB and Static Data/Metadata will be pushed to Aurora. Redis ElastiCache will be used for Retrieve requests first, if no relevant cached queries, then it will query directly to DynamoDB or Aurora. Both DBs will have an ETL to S3 cold storage for historical data / Archives.

Drivers will be provided the predetermined Routes from the Backend but once they start their routes, they will make a call to a third party Routing API to get Real Time Turn by Turn navigation to each stop.

Scanners will be used to scan Barcodes/RFID of students ID card and send that data to the Backend servers for parents to be notified that their child was picked up and what bus they were picked up by.

**Parent Mobile App**

Users will authenticate with DOE IDP to get into the driver application, routed through our NLB then the ALB before passing through to the Public Subnets Amplify Service. Amplify is a dev framework and hosting services that offers Serverless Architecture for Applications running in AWS. Once on Amplify, users are rooted through the Web Application Firewall which offers DDoS protection, Traffic filtering based on IP Addresses, HTTP Headers/Body, Custom URIs, SQL Injection patterns and Cross Site Scripting. After Security, users requests will hit the AWS API Gateway for all Rest, HTTP and Websock API routing to the Backend server running on the Private Subnet within an EC2.

The Parent App will provide the ability to select a bus route and stream the buses location, provide ETA when the bus will be at the Pickup / Drop off location. Parents will be able to adjust Pickup/drop off locations of their children which will push data to Aurora.

**GPS**

There will be a separate/Built in GPS tracking device that will stream the current buses location to the AWS Iot Core Service, which will pass the location Data directly to DynamoDB and AWS Location Services which is a managed package that provides Route tracking and Geofencing. This will allow for Real Time tracking of Buses, and data to compare the true routes vs the Predestined Routes created by DOE OPT users. This data will be passed to Amazon EventBridge to trigger Simple Notification Service if a route is deviated or a bus leaves their Geofenced location (NYC) during a route.

**Security Monitors**

We will collect VPC Flow Logs at the VPC, Subnet and network interface level to capture Source/Destination IP Addresses, Ports, Protocol, Number of packets and bytes transferred, Time, and action. CloudTrail will be linked to each Ec2 for Event Logging of AWS Activity, including API Calls, Console actions, Data Events, and insights events for unusual API Activities. DNS Logging will provide Domain/Subdomain Requests, Date/Time, DNS Record, Route 53 Edge Location, DNS response Codes. All of these tools will pass to S3 for event storage, Guard Duty for threat detection and Amazon Detective for ML statistical analysis. These tools together will pass information to EventBridge and then to Simple Notification Service/Lambda depending on the Event/Alert/activity.While Guard Duty detects potential threats and generates findings/alerts, the Amazon Detective will be used by the Security operations team to triage and investigate these findings to gain contextual understanding and apply any mitigation when required. The security operations team will also be setting up automated mitigation for specific serious network compromise scenarios using lambda to isolate compromised assets from the rest of the network, to reduce blast radius and subsequent forensic investigation.

**Reporting / Tracking**

### 2.1 Design Principles

* GovCloud Native: Utilize AWS services available within the GovCloud (US) regions.
* Security by Design: Implement security controls at every layer (network, compute, data, application). Assume a Zero Trust posture.
* High Availability & Resilience: Design for fault tolerance using multi-AZ deployments and managed services with built-in HA. Implement robust Disaster Recovery (DR).
* Add High Throughput
* Scalability & Elasticity: Employ auto-scaling, serverless components, and elastic managed services to handle variable loads (daily peaks, seasonal changes) efficiently.
* Modularity (Microservices): Utilize a microservices architecture to promote independent development, deployment, scaling, and fault isolation of components.
* Event-Driven: Leverage asynchronous messaging and event streams for decoupling components and enabling real-time processing.
* Infrastructure as Code (IaC): Manage infrastructure provisioning and configuration through code for consistency, repeatability, and auditability.
* Managed Services Preference: Utilize AWS managed services (AuroraDB, DynamoDB, MSK, SNS, Lambda, EKS etc.) to reduce operational overhead and leverage AWS expertise.

New Section - Add System Architecture Here

WIP - Contents applicable for system architecture

* System diagram
* Network/Storage
* Security
* Private/public/security subnet
* Various Components
* Data Model High Level (Should be a separate section for deeper diver later\_

## 3. System Architecture

Sentry SBMS architecture is a cloud-native, event-driven microservices architecture deployed across multiple Availability Zones (AZs) within an AWS GovCloud (US) region. It is a 7 layered application platform which consists of

* **Presentation Layer:** Mobile applications (iOS/Android) and Web applications (React/Angular/Vue hosted on S3/CloudFront) interact with backend APIs.
* **Core Services Layer:** Lambda functions and containerized services (on ECS) process events and data streams. The Routing Engine may use dedicated EC2 instances for intensive computation.
* **Event Processing Layer:** GPS devices communicate via MQTT/HTTPS to AWS IoT Core or directly to API Gateway endpoints. Data flows through streaming services (MSK).
* **APIs & Communication Layer:** RESTful APIs and HTTPS exposed via API Gateway serve front-end applications and internal service communication. SNS handles asynchronous messaging and notifications.
* **Data Services Layer:** A mix of databases (AuroraDB, DynamoDB, ElastiCache) and object storage (S3) is used based on data type and access patterns.
* **Monitoring & Logging Layer:** Promtail, Grafana Loki for Application Logs, Prometheus for Metrics & Alerts , Grafana for visualization . AWS CloudWatch, DNS Logs, AWS Cloudtrail Logs, AWS VPC Flow Logs are additional logging infrastructure for security and other infrastructure specific logging
* **Security Layer:**

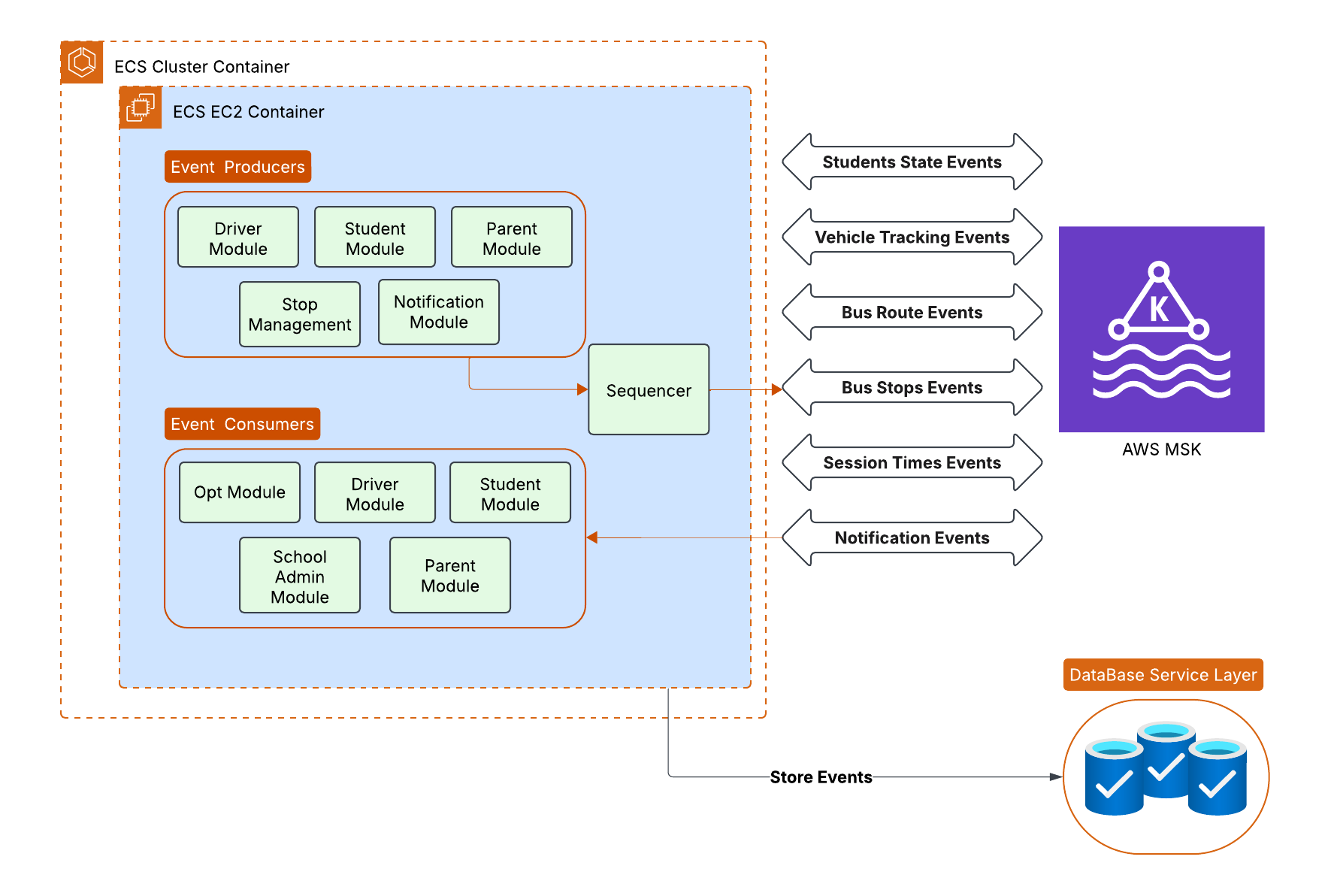
### 3.1 Presentation Layer

//Kenny to add

### 3.2 Core Services Layer

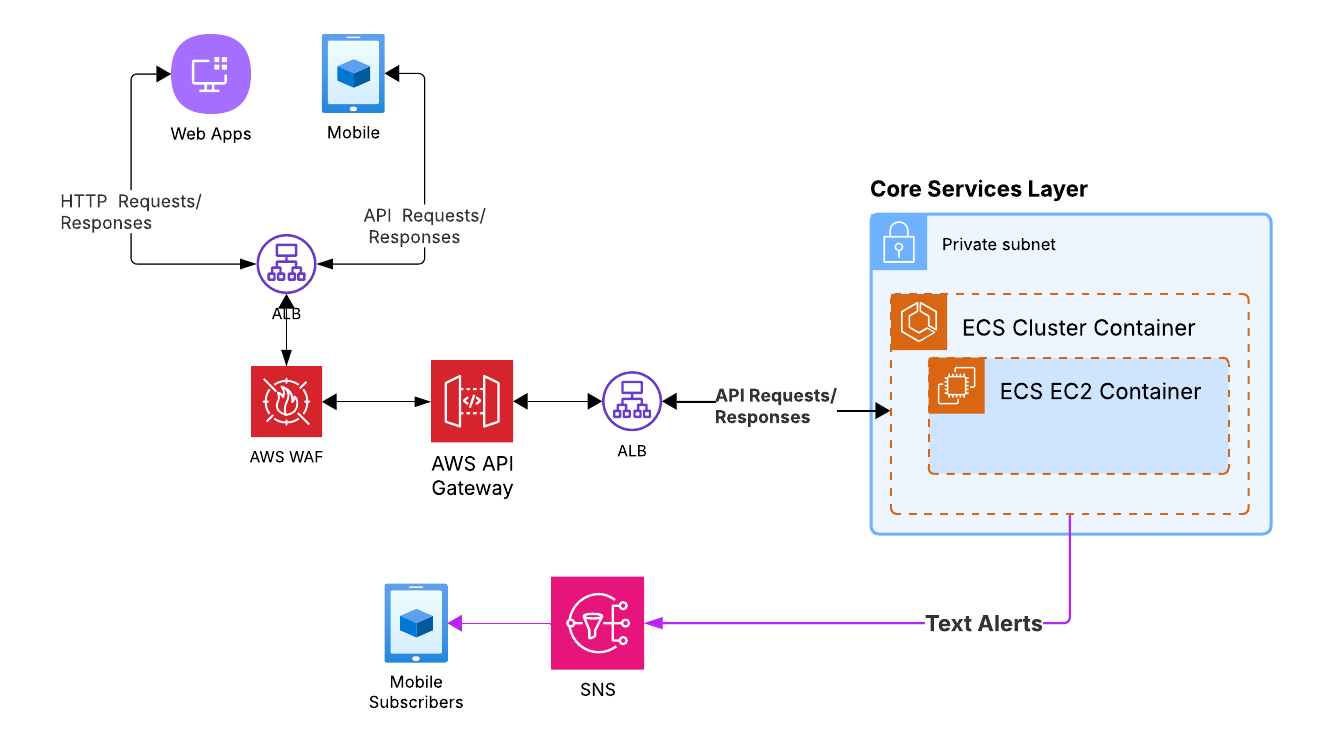
//HOLD until we get the Dynamic Routing Infra diagram

### 3.3 Event Processing Layer



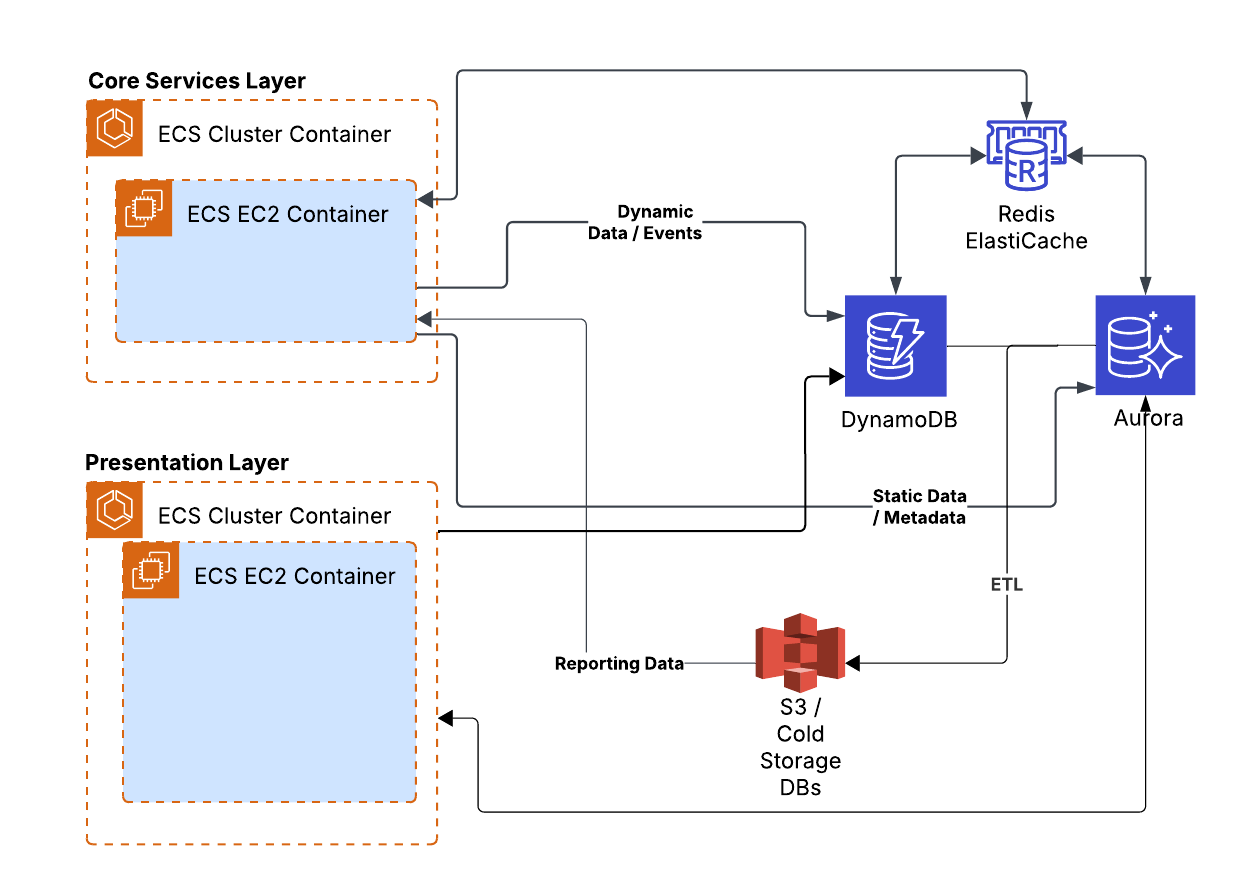
**Figure 3.3 Event Processing Diagram**

### 3.4 APIs & Communication Layer



**Figure 3.4 APIs & Communication Diagram**

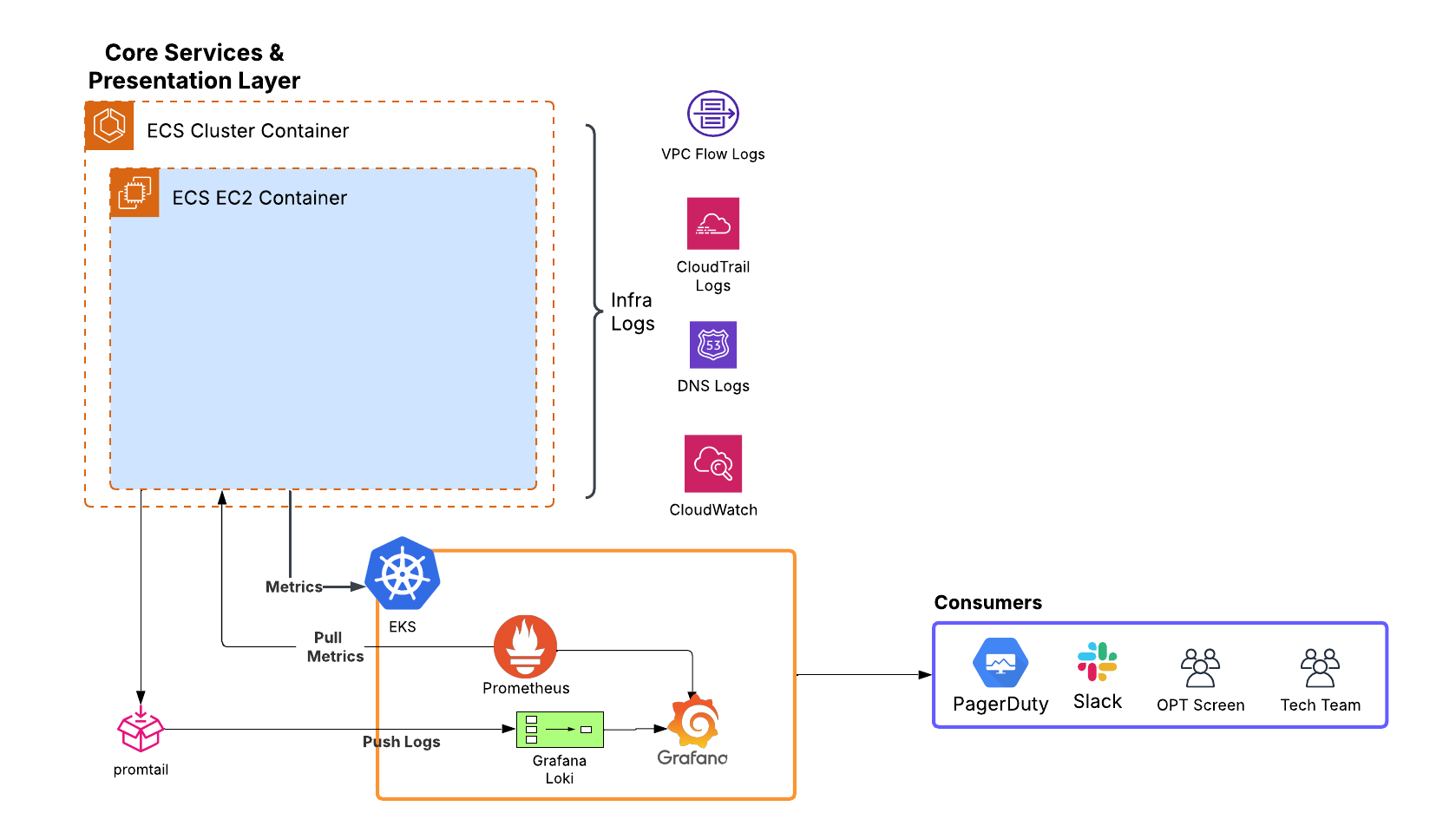
### 3.5 Data Services Layer



**Figure 3.5 Data Services Diagram**

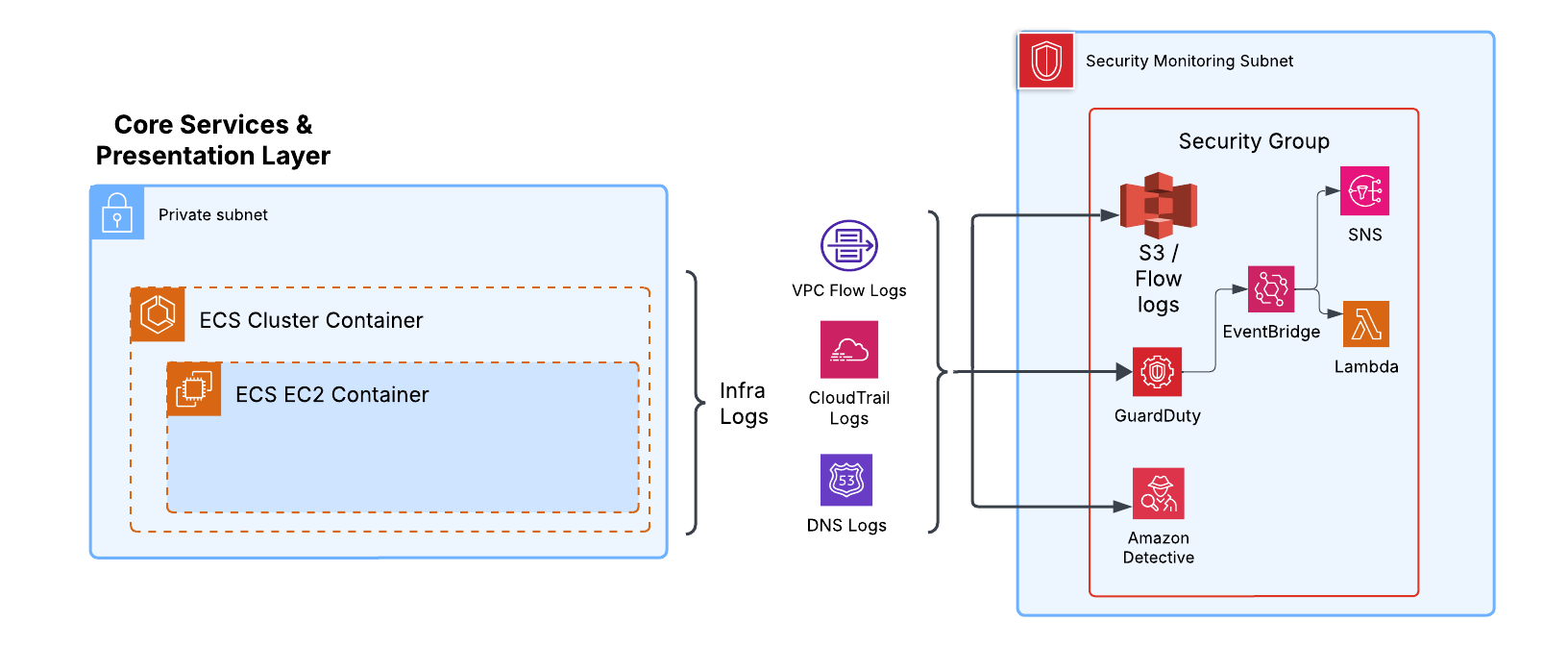
### 

### 3.6 Monitoring & Logging Layer



**Figure 3.6 Monitoring & Logging Diagram**

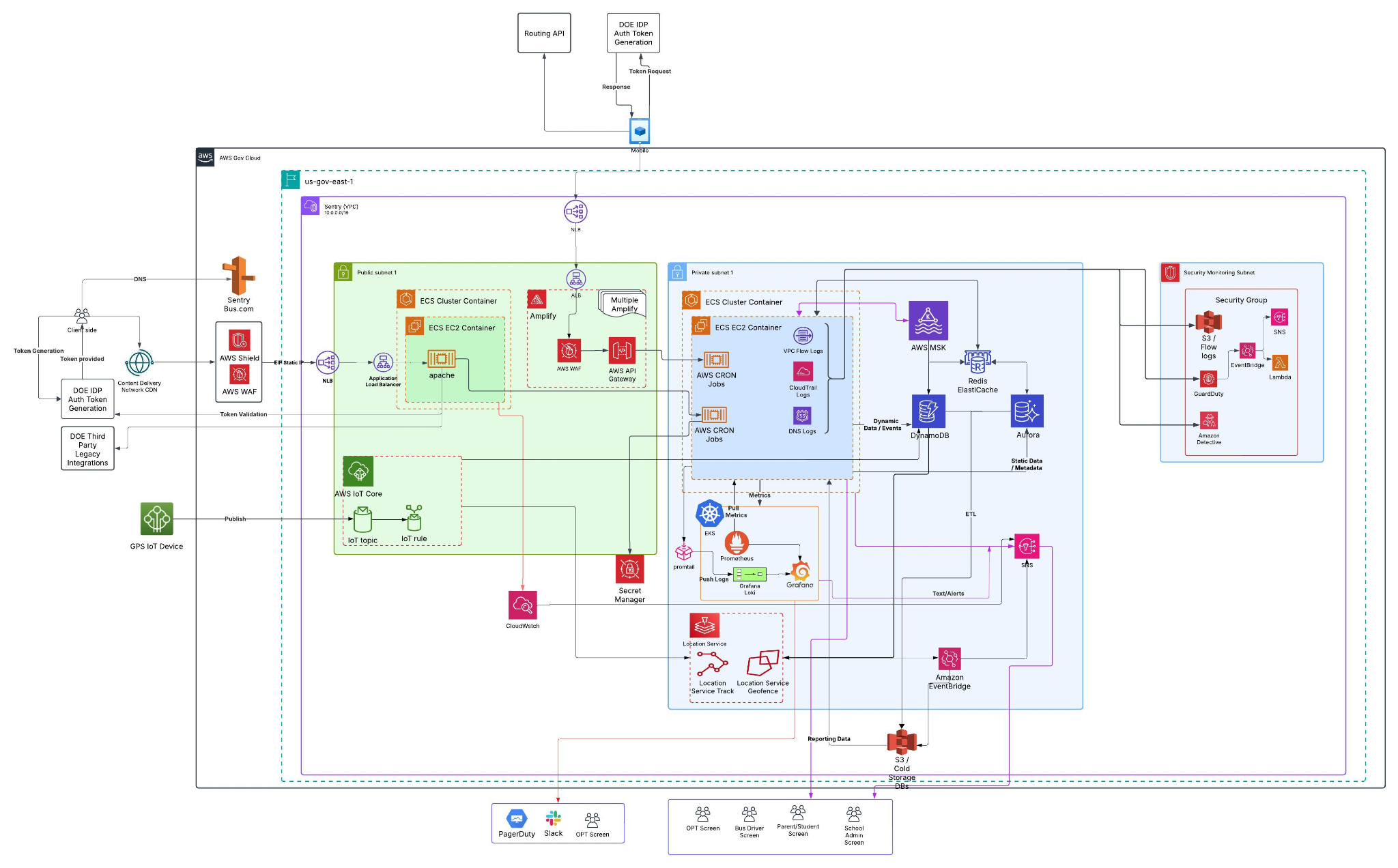
### 3.7 Security Layer

****

**Figure 3.7 Security Layer Diagram**

### 3.8 Full Architecture Diagram

* [Link to Full Architecture Diagram](http://drive.google.com/uc?id=1PTwLBljPVsrdClP9BaTyImQAwnLRcaD3)



**Figure 3.8 Full Architecture Diagram**

## 4. Throughput & Scalability

//Add anticipated Event numbers etc.

## 5. Data Model & Management

* Storage: Tiered storage using S3 Standard, S3 Intelligent-Tiering, S3 Glacier, and S3 Glacier Deep Archive to meet access needs and 7-year retention requirements cost-effectively.
* Databases: Use appropriate databases per component needs (RDS for relational/spatial, DynamoDB for key-value/high-throughput, ElastiCache for caching). Regular backups and point-in-time recovery configured.
* Data Privacy: Implement controls aligned with Solution Design, including encryption, RBAC, audit logging, and data minimization principles.
* Data Ownership & Export: Mechanisms (e.g., APIs, data exports to S3) provided for NYCPS to access and extract their data on demand, reaffirming DOE data ownership.

New Section - Add Data Model Section

## 6. Networking & Security (AWS GovCloud)

* VPC Structure: Multiple VPCs may be used (e.g., Prod, Dev, Test). Each VPC spans multiple AZs. Use of public subnets restricted to necessary components (e.g., Load Balancers, NAT Gateways). Private subnets host core application/database resources. VPC Endpoints (Gateway and Interface) used for private access to AWS services (S3, DynamoDB, Kinesis, etc.).
* Segmentation: Security Groups (stateful) and Network ACLs (stateless) enforce strict ingress/egress rules based on the principle of least privilege between application tiers and components.
* Connectivity: AWS Direct Connect or Site-to-Site VPN established between AWS GovCloud VPC and NYCPS data centers for secure, private access to internal resources/databases.
* Edge Security: CloudFront for caching and DDoS protection for web applications. AWS WAF deployed with API Gateway and CloudFront to filter malicious requests (SQLi, XSS). AWS Shield Advanced for enhanced DDoS protection.
* Encryption:
  + In Transit: TLS 1.2+ enforced for all external and internal API Gateway endpoints, CloudFront distributions, Load Balancers, and direct service communications.
  + At Rest: Server-side encryption enabled for S3 (SSE-S3 or SSE-KMS), EBS volumes, RDS instances/snapshots, DynamoDB tables, SQS queues, SNS topics using AWS KMS with customer-managed keys (CMKs) where appropriate for enhanced control and auditability.
* Identity & Access Management (IAM):
  + Strict use of IAM Roles for EC2 instances, Lambda functions, ECS/EKS tasks (via IAM Roles for Service Accounts - IRSA - in EKS). Avoid long-lived access keys.
  + Fine-grained IAM Policies adhering to least privilege.
  + Multi-Factor Authentication (MFA) enforced for all human access to the AWS Management Console and API, especially for privileged accounts.
  + Federated identity management via SAML 2.0 integration with NYCPS's Identity Provider (e.g., ADFS, Azure AD) for console/API access if feasible, otherwise stringent controls on IAM users.
* Secrets Management: AWS Secrets Manager used to store and rotate database credentials, API keys, and other secrets securely. Applications retrieve secrets at runtime via IAM roles.
* Monitoring & Logging:
  + CloudTrail enabled in all regions, logging to a central, secured S3 bucket with log file validation enabled.
  + CloudWatch Logs collected from all services (Lambda, Fargate, EC2, RDS, etc.). CloudWatch Alarms configured for critical metrics (CPU, memory, latency, error rates, queue depths).
  + AWS Config used to track resource configurations and compliance.
  + Security Hub aggregated findings from GuardDuty, Inspector, Macie (if used), Config, and partner integrations.
  + GuardDuty enabled for intelligent threat detection.
  + Inspector used for vulnerability assessments on EC2 instances (if used).

# Anything below is Archived

## 3. Overall Architecture Pattern

The proposed architecture is a cloud-native, event-driven microservices architecture deployed across multiple Availability Zones (AZs) within an AWS GovCloud (US) region.

* Data Ingestion: GPS devices communicate via MQTT/HTTPS to AWS IoT Core or directly to API Gateway endpoints. Data flows through streaming services (MSK).
* Processing: Lambda functions and containerized services (on ECS) process events and data streams. The Routing Engine may use dedicated EC2 instances for intensive computation.
* Data Storage: A mix of databases (AuroraDB, DynamoDB, ElastiCache) and object storage (S3) is used based on data type and access patterns.
* APIs & Communication: RESTful APIs exposed via API Gateway serve front-end applications and internal service communication. SNS/SQS/SES handle asynchronous messaging and notifications.
* Frontend: Mobile applications (iOS/Android) and Web applications (React/Angular/Vue hosted on S3/CloudFront) interact with backend APIs.

## 4. Component-to-AWS Service Mapping

| Functional Component | Primary AWS GovCloud Service(s) | Rationale & Notes |
| --- | --- | --- |
| GPS Data Ingestion & Processing | IoT Core (optional, for MQTT), API Gateway + Lambda/Fargate, Kinesis Data Streams or MSK, S3 (raw data), DynamoDB (real-time location), RDS PostgreSQL w/ PostGIS (processed tracks, geofences), Lambda/Fargate (ETA calc, geofence logic), CloudWatch (device status) | IoT Core provides scalable MQTT ingestion if devices support it. Kinesis/MSK handles high-throughput streaming data. Lambda/Fargate enables scalable, event-driven processing. DynamoDB offers low-latency reads/writes for real-time location. RDS/PostGIS handles complex spatial queries & history. |
| Ridership Tracking Module | API Gateway + Lambda/Fargate, DynamoDB or RDS PostgreSQL, S3 (for ID scan images if needed) | API endpoints handle scan/manual entry events. DynamoDB or RDS store ridership records, depending on query patterns. Lambda/Fargate process events, validate against rosters (from Student Mgmt). |
| Dynamic Routing Engine | EC2/ECS/EKS (potentially w/ GPU instances), RDS PostgreSQL w/ PostGIS, ElastiCache (Redis/Memcached), AWS Location Service (if available/suitable) or self-managed ESRI ArcGIS on EC2/RDS, SQS/Step Functions (async tasks), Lambda (rule triggers) | EC2/Containers needed for complex, potentially long-running routing algorithms. RDS/PostGIS stores road network, stops, routes, constraints. ElastiCache speeds up access to frequently used data (e.g., traffic, routes). AWS Location or ESRI for mapping/geocoding/traffic. SQS/Step Functions for orchestration. |
| Notification & Communication | SNS (Push, SMS), SES (Email), Amazon Connect (optional, for Robocalls/IVR), Lambda/Fargate (notification logic), API Gateway (for internal communication APIs) | SNS/SES provide scalable, managed notification delivery. Connect offers programmable voice capabilities. Lambda/Fargate implement business logic for triggering and formatting notifications based on events from other systems. |
| User Modules (Frontend) | S3 + CloudFront (Web Apps), Native iOS/Android SDKs, Amplify (optional, for mobile/web dev acceleration) | S3/CloudFront provide scalable, secure, low-latency hosting for web frontends. Native SDKs for mobile apps. Amplify can streamline frontend development and backend integration. |
| User Modules (Backend APIs) | API Gateway, Lambda / Fargate (ECS/EKS), Cognito or SAML integration via backend, IAM | API Gateway provides secure, scalable API endpoints. Lambda/Fargate host backend microservices logic. Cognito handles user authentication/pools if direct DOE integration isn't used. IAM secures service-to-service communication. |
| Student Management Backend | API Gateway + Lambda/Fargate, RDS PostgreSQL or DynamoDB, Glue (for ETL from NYCPS systems if needed), SFTP/DataSync (for file-based integration if needed) | Standard API-driven microservice pattern. Choice of DB depends on data model complexity and query patterns. Glue/SFTP/DataSync facilitate integration with potentially legacy NYCPS systems. |
| Reporting & Analytics | S3 (Data Lake), Glue (ETL/Catalog), Athena, Redshift, QuickSight, Kinesis Data Analytics (optional, real-time KPIs), Lambda (scheduled report generation) | S3 provides a scalable data lake. Glue handles ETL and cataloging. Athena for ad-hoc queries on S3. Redshift for performant data warehousing. QuickSight for dashboards/visualizations. Kinesis Data Analytics for streaming analytics. Lambda for automated report generation. |
| Device Management & Inventory | DynamoDB or RDS PostgreSQL, IoT Core (Device Shadow/Registry), Systems Manager (potentially for config), API integration with MDM & Ticketing | DynamoDB/RDS stores inventory data. IoT Core can track device state/metadata if devices connect via MQTT. Systems Manager might assist with certain configurations. Requires integration points. |
| Ticketing System Integration | API Gateway + Lambda/Fargate | Acts as a secure facade/adapter layer to mediate API calls between AWS environment and the NYCPS/Vendor ticketing systems. |

## 

## 7. Deployment & Operations (DevSecOps)

* Infrastructure as Code (IaC): AWS CloudFormation, AWS CDK, or Terraform used to define and provision all infrastructure resources. Templates stored in version control (AWS CodeCommit or GitHub/GitLab).
* CI/CD Pipeline: AWS CodePipeline orchestrating CodeCommit (source), CodeBuild (build/test, including SAST/DAST scans), and CodeDeploy or ECS/EKS deployment strategies (blue/green, canary) for automated, secure deployments across environments (Dev, Test, Staging, Prod).
* Monitoring: Leverage CloudWatch dashboards, alarms, and logs. Integrate with centralized logging/monitoring solutions if used by NYCPS. Potentially use APM tools (e.g., Datadog, Dynatrace, configured appropriately for GovCloud) for deeper application insights.

## 8. High Availability & Disaster Recovery (DR)

* High Availability (HA):
  + Deploy critical components (API Gateway, Lambda, Fargate/ECS/EKS clusters, RDS, ElastiCache, DynamoDB) across multiple AZs (typically 3) within the primary GovCloud region.
  + Use Elastic Load Balancing (ALB/NLB) to distribute traffic across AZs.
  + Configure RDS Multi-AZ deployments for automatic failover.
  + Leverage DynamoDB global tables (if cross-region active-active is needed and available) or rely on inherent multi-AZ replication.
* Disaster Recovery (DR):
  + Establish a DR strategy (e.g., Pilot Light, Warm Standby) in a second AWS GovCloud (US) region.
  + Regularly back up data (using AWS Backup, RDS snapshots, DynamoDB backups) and replicate backups/snapshots to the DR region.
  + Use IaC to provision infrastructure in the DR region quickly.
  + Utilize Route 53 health checks and DNS failover mechanisms.
  + Regularly test the DR plan (at least annually).
  + Design to meet RFP RPO/RTO targets (RPO=0 for GPS, RPO<=1hr for Routing/Notifications; RTO=0 for GPS, RTO<=15min for Routing/Notifications). Achieving RTO=0/RPO=0 for GPS likely requires an active-active or hot standby approach across AZs/regions for critical ingestion/processing components.

## 9. Compliance

* The architecture leverages services within AWS GovCloud (US), designed to host sensitive data and regulated workloads, meeting standards like FedRAMP High, ITAR, CJIS, DoD SRG IL4/5.
* Specific configurations (encryption, logging, IAM, network controls) align with security requirements from NYCPS, NYC3, OTI, DIIT, FERPA, CIPA, HIPAA (as applicable within GovCloud).
* Regular audits and security testing (as required by RFP) will validate ongoing compliance.