**Project Overview**

This project is designed to detect and recognize vehicle number plates in real-time using a robust pipeline. The system leverages AWS services for scalable data processing, advanced AI models for object detection, and a consumer-based architecture for data visualization and storage. The project is divided into three main components:

1. **Producer**: Streams video data to AWS Kinesis Video Streams.
2. **Consumer #1**: Processes the streamed data to perform object detection and tracking.
3. **Consumer #2**: Visualizes and stores the processed data for analysis and reporting.

**Detailed Pipeline**

**1. Producer (Video Stream Input)**

The producer streams video data into AWS Kinesis Video Streams. Here's how it works:

* **AWS Kinesis Setup**: Kinesis Video Streams is configured to handle live or offline video feeds.
* **EC2 Instance(**Amazon EC2 (Elastic Compute Cloud)**)**: Its basically a CPU, EC2 instance are used for GPUs and CPUs   
  A lightweight t2.small EC2 instance runs the video producer, which uses the Amazon Kinesis Video Streams Producer SDK.
* **GStreamer Pipeline**: A video file (or live feed) is sent to Kinesis via the GStreamer pipeline. This includes encoding the video in the correct format for efficient streaming.

Amazon EC2's T2 instances are designed to provide a baseline level of CPU performance with the ability to burst above the baseline when needed. The primary differences between the t2.small and t2.large instances are in their vCPU count, memory allocation, baseline CPU performance, and pricing.

**Instance Specifications:**

| **Instance Type** | **vCPUs** | **Memory (GiB)** | **Baseline CPU Performance** | **CPU Credits Earned per Hour** | **Maximum CPU Credit Balance** | **On-Demand Price per Hour\*** |
| --- | --- | --- | --- | --- | --- | --- |
| t2.small | 1 | 2.0 | 20% | 12 | 288 | $0.023 |
| t2.large | 2 | 8.0 | 60% (30% per vCPU) | 36 | 864 | $0.0928 |

**2. Consumer #1 (Object Detection and Tracking)**

This consumer processes the video stream to detect and track vehicles and number plates.

* **EC2 Instance**: A more powerful t2.xlarge instance is used to handle computationally intensive tasks.
* **AI Models**:
  + **Object Detection**: YOLO (via the ultralytics library) detects vehicles and number plates in the frames.
  + **Tracking**: SORT (Simple Online and Realtime Tracking) is used to maintain tracking of moving objects across frames.
* **Data Handling**:
  + Detected number plates are stored in **DynamoDB**.
  + Plate images or metadata are uploaded to **S3**.

**Storing Processed Images in S3 bucket**

* **What Happens?**
  + When YOLO detects a number plate in a video frame, the relevant portion of the frame is cropped or saved as an image.
  + This image is then uploaded to an S3 bucket for permanent storage.
* **Why Use S3?**
  + Provides durable and reliable storage for these images, ensuring they are safe and accessible for future use.
  + Scales automatically to handle a large number of images.
* **Example Workflow**:
  + YOLO detects a number plate and generates an image (e.g., vehicle-plate-123.jpg).
  + The image is uploaded to a specific path in an S3 bucket.
* **Integration with AWS**:
  + **DynamoDB**: Stores structured information (e.g., vehicle details, timestamps).
  + **SQS**: Facilitates communication between processing units (queues for data transfer).
  + **Lambda**: Executes event-based tasks like invoking downstream processing for detected plates.

**Consumer #2 (Visualization and Reporting)**

This consumer focuses on visualizing the processed data and presenting it for further analysis.

* **Visualization Tools**:
  + The main\_plot.py script generates real-time visualizations of the tracked vehicles and recognized number plates.
  + Processed frames are displayed, highlighting detected number plates and tracking information.
* **AWS Integration**:
  + **S3**: Stores visualized frames and processed outputs.
  + **SQS**: Receives messages about completed detections and triggers visualization tasks.
* **User Interaction**:
  + Users can view real-time dashboards and logs.
  + Reports can be generated from the stored data for further insights.

**What is YOLO?** YOLO is a state-of-the-art object detection algorithm known for its speed and accuracy. Unlike traditional methods, which process images in multiple steps, YOLO predicts bounding boxes and object classes in a single pass.

**DynamoDB**

**What is DynamoDB?** Amazon DynamoDB is a fully managed NoSQL database service that provides fast and scalable data storage. It is ideal for applications that require low-latency data access, low latency data means data which takes minimum time to travel from source to destination.

**How DynamoDB Works in Your Project:**

1. **Storing Data**:
   * Each detected number plate and associated metadata (e.g., timestamp, vehicle details) are stored as a record in DynamoDB.
2. **Table Schema**:
   * Primary Key: Number plate ID or a combination of plate number and timestamp.
   * Attributes: Plate image URL, detection time, vehicle type, etc.
3. **Integration with Other AWS Services**:
   * **Lambda**: Reads and writes data to DynamoDB during event processing.
   * **S3**: Stores actual images, and DynamoDB stores references (e.g., S3 URLs).

**Why DynamoDB?**

* **Fast Queries**: Supports millisecond response times, even for large datasets.
* **Scalability**: Automatically scales to handle any workload.
* **Event-Driven**: Can trigger actions based on data changes.

**Amazon SQS (Simple Queue Service)**

**What is SQS?** SQS is a fully managed message queuing service that allows decoupling of application components. It ensures reliable communication between different parts of the system.

**How SQS Works in Your Project:**

1. **Message Producer**:
   * The detection pipeline (YOLO and SORT) sends messages to an SQS queue whenever a new number plate is detected.
2. **Message Consumer**:
   * Downstream processes (e.g., visualization scripts or Lambda functions) read messages from the queue and take action.
3. **Message Format**:
   * Includes metadata like number plate, timestamp, vehicle type, and S3 URL of the image.

**Types of Queues Used**:

* **FIFO Queue**:
  + Ensures messages are processed in the exact order they are sent.
  + Useful for sequential data processing (e.g., tracking vehicles in real time).

**Why SQS?**

* **Reliability**: Guarantees message delivery and durability.
* **Decoupling**: Separates the detection pipeline from storage and visualization components.
* **Scalability**: Automatically handles varying message volumes.

**AWS Lambda**

**What is Lambda?** AWS Lambda is a serverless compute service that lets you run code in response to events without provisioning or managing servers.

**How Lambda Works in Your Project:**

1. **Triggered by Events**:
   * Example: A new image is uploaded to S3, triggering a Lambda function to process the number plate.
2. **Processing Logic**:
   * Reads metadata from DynamoDB.
   * Analyzes or enhances the detected data (e.g., runs OCR on the plate for validation).
3. **Outputs**:
   * Updates DynamoDB with additional information (e.g., OCR results).
   * Notifies other components via SQS or SNS (Simple Notification Service).

**Why Lambda?**

* **Event-Driven**: Automatically triggers functions when specific events occur (e.g., new data in S3 or DynamoDB).
* **Scalability**: Automatically scales with demand, ensuring quick response times.
* **Cost-Effective**: Only pay for the compute time used.

**How They Work Together**

* **YOLO**: Detects number plates in video frames and sends data to DynamoDB and SQS.
* **DynamoDB**: Stores structured data for each detection (e.g., plate numbers, timestamps).
* **SQS**: Acts as a message broker, ensuring smooth communication between different services (e.g., detection and visualization).
* **Lambda**: Processes events, such as analyzing new data in S3, and updates DynamoDB or triggers downstream tasks.

**Complete Flow Diagram**

1. **Video Producer**:
   * Captures the video feed and streams it to AWS Kinesis.
2. **Kinesis Video Stream**:
   * Handles the video stream and passes it to Consumer #1.
3. **Object Detection and Tracking** (Consumer #1):
   * YOLO detects number plates.
   * SORT tracks moving vehicles.
   * Results (images and metadata) are stored in S3 and DynamoDB.
4. **Event Notifications**:
   * S3 triggers Lambda for further processing.
   * SQS queues messages for visualization.
5. **Visualization and Reporting**:
   * Consumer #2 visualizes data in real-time.
   * Reports can be generated for analysis.

**How Components Work Together**

1. **AWS Kinesis**:
   * Acts as a bridge between the producer and the processing components.
2. **EC2 Instances**:
   * Run heavy workloads like YOLO and SORT (detection and tracking).
3. **S3**:
   * Stores processed data (images, frames) and triggers further workflows.
4. **DynamoDB**:
   * Maintains a structured record of detections.
5. **Lambda**:
   * Automates OCR and metadata updates.
6. **SQS**:
   * Ensures smooth communication between processes.
7. **Visualization Scripts**:
   * Provide a user-friendly interface to monitor and analyze results.

**Example Use Case in Action**

* **Scenario**: Monitoring traffic at a toll booth.

1. Live video feeds from cameras are streamed to AWS Kinesis.
2. YOLO detects vehicles and their number plates in real-time.
3. Data (plate images and numbers) are saved to S3 and DynamoDB.
4. A dashboard visualizes detected vehicles as they pass the toll booth.
5. Reports are generated to track traffic patterns or identify flagged vehicles.

**Applications**

* **Traffic Monitoring**: Automates the recognition of number plates for traffic management.
* **Toll Collection**: Detects vehicles and links them to payment systems.
* **Parking Management**: Tracks vehicles entering and exiting parking lots.
* **Law Enforcement**: Identifies and tracks stolen or flagged vehicles.