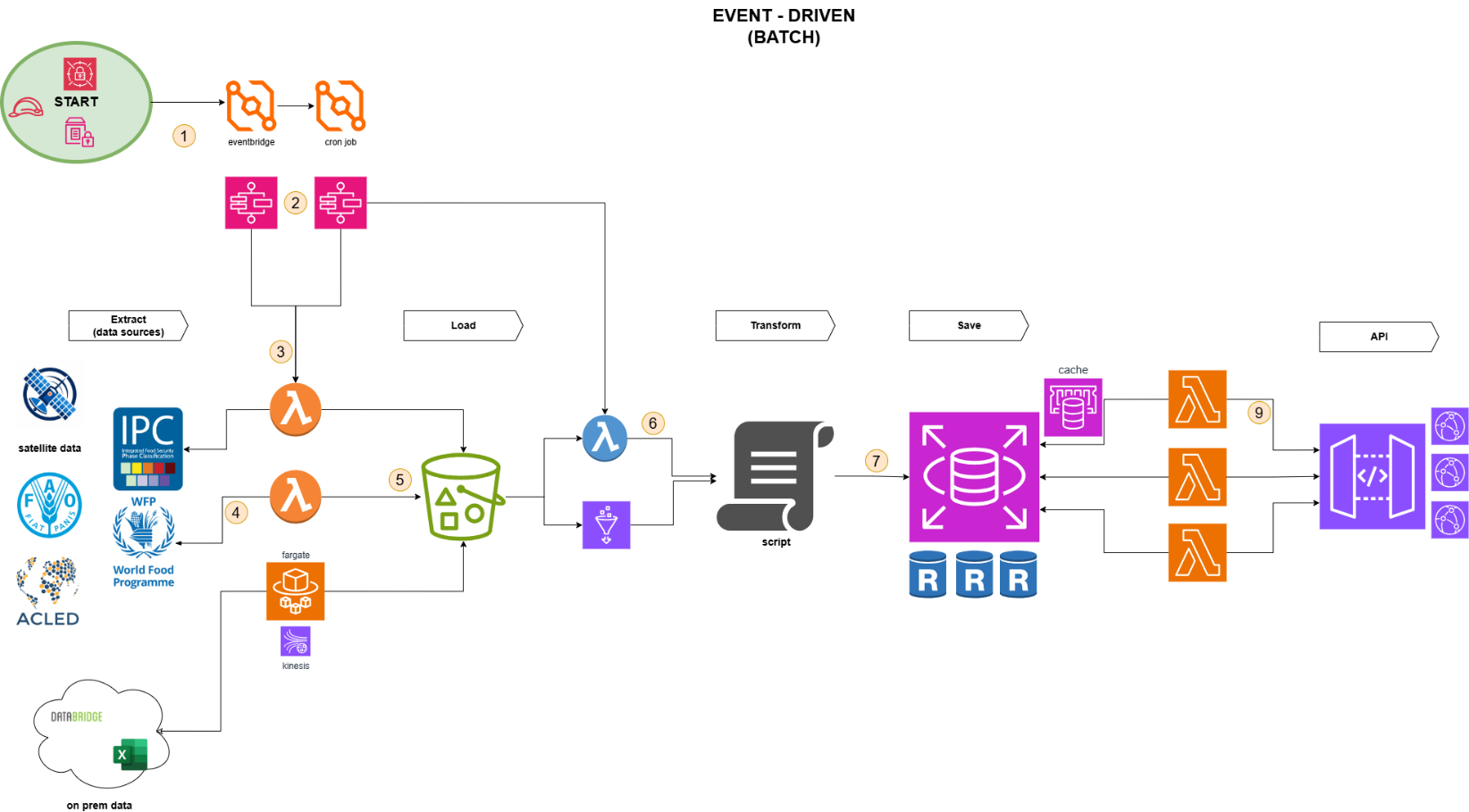
# System Design for Food Security Monitoring

## System Architecture

The system follows a batch-based processing pipeline with AWS-managed services for scalability. The architecture consists of the following key components:

<https://drive.google.com/file/d/1Je8fk01ZWs_SUMLokhMh42P44I4y_2mo/view?usp=sharing>



## Technical Explanation (AWS Services)

### Event Triggering & Scheduling

* **AWS EventBridge, Cron Jobs**
  + Automate the periodic execution of data extraction and processing jobs.
* **Step Functions**
  + Manage workflow

### Data Extraction & Ingestion

* **AWS Lambda**
  + Serverless functions to pull data from satellite feeds, economic indicators, and conflict data.
* **AWS S3**
  + Store extracted raw data for further processing.
* **AWS Fargate**
  + Facilitates long running data fetches.

### Data Transformation & Processing

* **AWS Lambda & Processing Scripts**
  + Process and transform the data, normalizing and filtering it.
* **AWS RDS / Amazon Aurora**
  + Store transformed data in a relational database.

### Api Layer

* **AWS Lambda**
  + Serverless functions to handle API requests.
* **AWS API Gateway**
  + To expose REST API for external consumption and external users.

## Data Processing Pipeline

The system follows a batch processing model with periodic updates:

1. EventBridge triggers AWS Step function that manages the workflow
2. Lambda functions to pull new data.
3. Data is extracted from external sources (satellite, IPC, ACLED, etc.).
4. Extracted data is stored on S3.
5. Another Lambda function from the Step Function workflow loads the data into the processing pipeline.
6. Transformation scripts process and clean the data.
7. The transformed data is stored in AWS RDS**.**
8. The API Gateway serves client applications with real-time alerts.
9. API queries retrieve food security alerts from the database.
   1. Improvements: ElastiCache (Redis) caches frequent queries to improve response time.
10. The API Gateway serves client applications with real-time alerts.
11. Improvements: CloudFront sits in front of the API Gateway making it highly available across the world.
12. AWS CloudFront sits in front of the API Gateway making it highly available across the world

## Security Considerations

* **IAM Roles & Permissions**
  + Used to manage access for AWS services.
* **VPC & Private Subnets**
  + Database and caching services are kept inside a private VPC.
* **Encryption**
  + API Gateway TLS (HTTPS).
  + ACM to manage certificates
* **WAF & Rate Limiting**
  + Protect APIs against DDoS attacks.

## API Design

* REST APIusingAWS API Gateway
* Endpoints structure:
  + /alerts → fetch food security alerts.
  + /download → download food security data.
* Response Format: JSON

## Deployment Strategy

* CI/CD azure-pipeline or CodePipeline & CodeBuild
* Infrastructure as Code (IaC) with AWS Terraform
* Automated testing ensure code passes before it is deployed

**Conclusion**

This architecture ensures a scalable, secure, and high-performance system for food security monitoring. It can efficiently handle large-scale data and provide real-time alerts via APIs.

**Notes:**

**Real-Time Ingestion**Depending on data rates, to handle the real-time data ingestion, AWS Kinesis can be used instead with the data sources always pushing data to Kinesis as opposed to data always going to be fetched. This would require knowledge from the external API sources. As data is only provided daily, this would be a hard stretch.

**GraphQL**Depending on the complexity of the data provided at the API endpoints, GraphQL can be used for complex data.