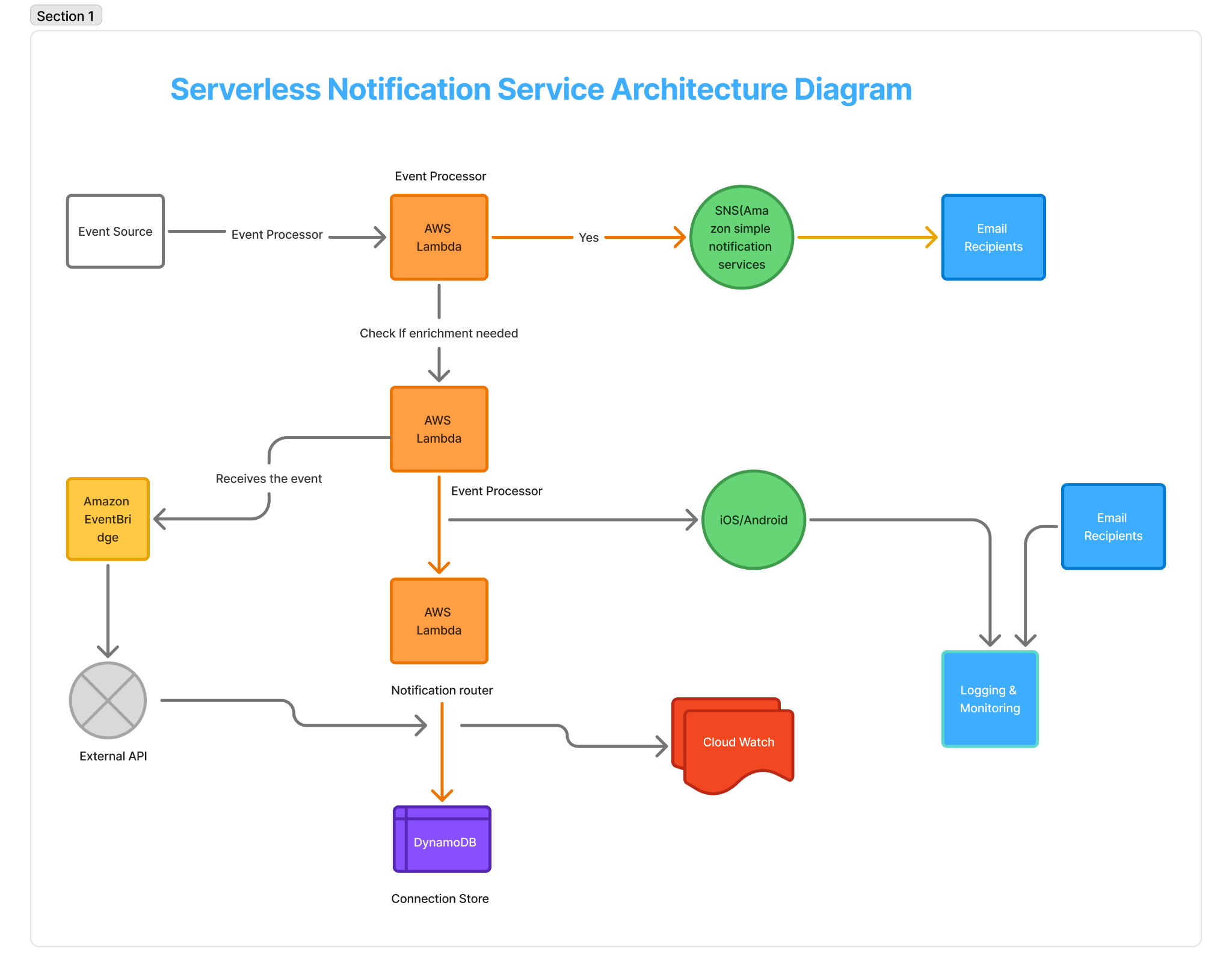
**Serverless Notification Service Architecture**

****

**1. Event Source**  
  
This is where the event lifecycle begins. An event source could be any external system or application that emits events, such as a user action in a web app, a backend service update, or a third-party webhook.  
  
**2. Event Processor**  
The first AWS Lambda function receives and parses the incoming event. It is responsible for basic validation and determines whether the event contains all necessary information. If enrichment is required, it routes the event to a process that can gather the missing data.  
  
**3. Enrichment Check**  
Here, the system decides if additional information is needed before proceeding. If data is incomplete, the event is sent through Amazon EventBridge, which then routes the request to an external API.   
**4. Amazon EventBridge & External API**  
Amazon EventBridge acts as a rule-based event bus. It handles the routing of the event to an external API for data enrichment. The external API could provide user details, localization data, or any other required metadata. The enriched data is then returned for continued processing.  
  
**5. Event Processor (Post-Enrichment)**  
Once the data is enriched, another Lambda function processes the complete event. This step ensures that the enriched event is formatted correctly and is now ready to be routed to the appropriate notification channel.  
  
**6. Notification Router (Lambda)**

This Lambda function acts as the central dispatcher. Based on event type and user preferences, it routes notifications to the right channel—such as email (via SNS), mobile push notifications, or real-time WebSocket messages. It also logs the notification status to a database.

**7. Amazon SNS (Simple Notification Service)**  
SNS is used to publish and fan-out messages to multiple subscribers. It handles sending notifications to users via email or other supported endpoints. This decouples the messaging system and allows asynchronous delivery of email alerts.  
  
**8. iOS / Android (Mobile Push Notifications)**  
For mobile users, the system sends push notifications through services like Firebase Cloud Messaging (FCM) for Android or Apple Push Notification Service (APNS) for iOS. These notifications keep users updated in real-time, even when the app isn't open.  
  
**9. Email Recipients (via SES or SNS)**  
Users receive dynamic and localized email notifications through Amazon SES or via SNS integrations. The system uses email templates with placeholders to personalize content and support multilingual communication.  
  
**10. DynamoDB**  
DynamoDB stores data such as user notification preferences, active WebSocket connections, and delivery logs. This provides a fast, scalable way to manage stateful data and helps track message delivery status across all channels.  
  
**11. CloudWatch**  
Amazon CloudWatch provides monitoring for Lambda functions, API calls, and other AWS services. It logs metrics like invocation counts, error rates, and performance stats, which are crucial for ensuring system health and optimization.  
  
**12. Logging & Monitoring**  
This component ensures all events and processes are logged for auditability and debugging. It can include CloudWatch Logs, structured log aggregation, and alerting tools. Monitoring gives visibility into failures and ensures reliable system operation.