**Serverless Notification Service Architecture**

**1. Event Processing & Routing**

**Services Used:**

* **Amazon EventBridge**: Event routing hub.
* **AWS Lambda**: For all logic — enrichment, routing, and delivery.
* **Amazon DynamoDB**: User preferences, device tokens, WebSocket connections.
* **Amazon SNS**: Push notifications.
* **Amazon SES**: Email notifications.
* **Amazon API Gateway (WebSocket API)**: Real-time notifications.

**Flow:**

1. **Event Ingestion**: Event from client/API → **API Gateway** → **EventBridge**.
2. **Event Filtering & Routing**: **EventBridge Rule** routes event to a Lambda function.
3. **Enrichment Lambda**:
   * Checks if enrichment is needed.
   * Retrieves additional data from:
     + **DynamoDB** (user/device info, preferences).
     + **External APIs** (e.g., user profile service).
4. **Notification Router Lambda**:
   * Based on enriched event, determines delivery channels.
   * Triggers appropriate downstream functions for delivery.

**2. Real-Time WebSocket Notifications**

**Implementation:**

* **API Gateway WebSocket API**: Manages WebSocket connections.
* **DynamoDB Table**: Stores mapping of user IDs to connection IDs.
* **Lambda Function**:
  + Retrieves user’s WebSocket connection ID.
  + Uses **@connections** API to push messages to the browser.

**3. Mobile Push Notifications**

**Implementation:**

* **Amazon SNS**:
  + One platform application per OS (iOS/Android).
  + Device tokens stored in DynamoDB during registration.
* **Lambda Function**:
  + Reads user device tokens from DynamoDB.
  + Publishes message to SNS for push delivery.

**4. Email Notifications**

**Implementation:**

* **Amazon SES**: Sends emails.
* **Email Templating**:
  + Templates stored in **Amazon S3** or embedded in code.
  + Supports **Handlebars** or similar templating engines.
  + Localization handled via language files or keys.
* **Lambda Function**:
  + Loads the appropriate language template.
  + Injects dynamic placeholders (e.g., {{username}}, {{action}}).
  + Sends via **SES SendEmail API**.

**5. Event Enrichment**

**Mechanism:**

* **Lambda: Event Enrichment**:
  + Checks for required fields in the event.
  + If missing:
    - Pulls from **DynamoDB** or external services (via HTTPS call).
  + Rewrites/enriches the event object.
* **Conditional Enrichment**:
  + Can be rule-based (e.g., via EventBridge rule metadata or event flags).

**6. Logging & Monitoring**

**Services Used:**

* **Amazon CloudWatch Logs**: All Lambda logs for debugging.
* **CloudWatch Metrics**: Custom metrics (e.g., notification success/failure).
* **CloudWatch Alarms**: Alert on anomalies (e.g., high failure rates).
* **AWS X-Ray**: Traces through Lambdas and API calls for end-to-end visibility.
* **EventBridge Archive** (Optional): Store raw incoming events for replay/debugging.

**Complete Flow (Simplified)**

1. **Event Received** → API Gateway → EventBridge.
2. **EventBridge Rule** → Enrichment Lambda.
3. **Enrichment Lambda** → adds user preferences/device/channel info.
4. → Routed to Notification Lambda.
5. **Notification Lambda** decides:
   * Web → API Gateway WebSocket
   * Mobile → SNS
   * Email → SES with localized template
6. **CloudWatch/X-Ray** logs all actions and errors.

