Serverless Notification Service Architecture using AWS

1. Event Origination:

- **The process begins when an Event Producer within your application or system generates an event that requires a notification to be sent to users. This could be anything from a new order being placed, a task being assigned, a message being received, or any other relevant activity.**
- **This event is then sent to an Event Bridge / Message Queue. Think of this as a central hub that receives all the notification-related events. Using a service like AWS EventBridge or a message queue like Amazon SQS provides several benefits:**
    - **Decoupling: The Event Producer doesn't need to know anything about the notification service's internal workings or how the notifications will be delivered.**
    - **Scalability: The message queue can handle a large volume of incoming events, even during peak times.**
    - **Reliability: Events are persisted in the queue, ensuring they are not lost if downstream services are temporarily unavailable.**
    -

2. Event Routing and Initial Processing:

- **The Event Router (Lambda function) is triggered whenever a new event arrives in the Event Bridge / Message Queue. This is the brain of the notification routing logic.**
- **The Event Router examines the content of the incoming event. Based on the event type, the intended recipients, or other relevant data within the event, it makes decisions about which notification channels are appropriate (e.g., WebSocket for real-time updates in the web application, push notifications for mobile users, and/or email for asynchronous communication).**
- **The Event Router then directs the event (or a relevant representation of it) to the specific handler functions for each chosen notification channel.**
-

3. Optional Event Enrichment:

- **Sometimes, the initial event might not contain all the necessary information required to construct a meaningful notification. In such cases, the Event Router might determine that Enrichment is Needed.**
- **If enrichment is required, the event is sent to the Event Enrichment (Lambda function). This function can perform tasks like:**

    - **Fetching additional user details from a database.**
    - **Retrieving context-specific information related to the event.**
    - **Formatting the data into a more user-friendly format for the notification.**
-
-
- **Once the event is enriched with the necessary data, it is sent back to the Event Router for further processing and routing to the appropriate notification handlers.**

4. WebSocket Notifications (Real-Time Web Application Updates):

- **If the Event Router determines that a WebSocket notification should be sent, it directs the relevant information to the WebSocket Handler (Lambda function).**
- **The WebSocket Handler needs to know which users are currently connected via WebSocket. This information is typically stored and managed in a database like DynamoDB (WebSocket Connections). When a user connects to the web application, a record of their connection ID is stored here.**
-
- **The WebSocket Handler retrieves the active connection IDs for the intended recipients from DynamoDB.**
- **It then uses the API Gateway (WebSocket) to send the notification message to the connected User Device(s) in real-time over the established WebSocket connection.**

5. Push Notifications (Mobile Devices):

- **When the Event Router identifies the need for a mobile push notification, it sends the relevant data to the Push Notification Handler (Lambda function).**
- **To send push notifications, the system needs to know the unique device tokens for each user's mobile device. These tokens are obtained when the user installs and registers the mobile application with the operating system's push notification service (APNs for iOS, FCM for Android). These tokens are stored in DynamoDB (Device Tokens). The Push Notification Registration process (indicated by the arrow from "User Device") is how these tokens are initially saved.**
-
- **The Push Notification Handler retrieves the appropriate device tokens from DynamoDB for the intended recipients.**
- **It then interacts with the respective platform's push notification service:**
    - **APNs (Apple Push Notification service) for sending notifications to iOS devices.**
    -
    - **FCM (Firebase Cloud Messaging) for sending notifications to Android devices.**
    -
- **These services then deliver the push notification to the user's User Device.**
-

6. Email Notifications (Asynchronous Communication):

- **If the Event Router decides that an email notification is necessary, it forwards the relevant information to the Email Handler (Lambda function).**
-
- **The system might maintain a record of users who have subscribed to receive email notifications in DynamoDB (Email Subscriptions). The Email Registration process (arrow from "User Device") would handle these subscriptions. The Email Handler might consult this data to determine the correct email addresses.**
- **The Email Handler then uses a third-party email sending service like SES (Amazon Simple Email Service), SendGrid, or another similar provider to compose and send the email notification to the intended recipients.**
    -