Phase 7: Integration & External Access

Goal:

Enable Salesforce to securely communicate with external systems (hotel API) while ensuring data security, scalability, and compliance with Salesforce governor limits

---

1. Named Credentials (Used)

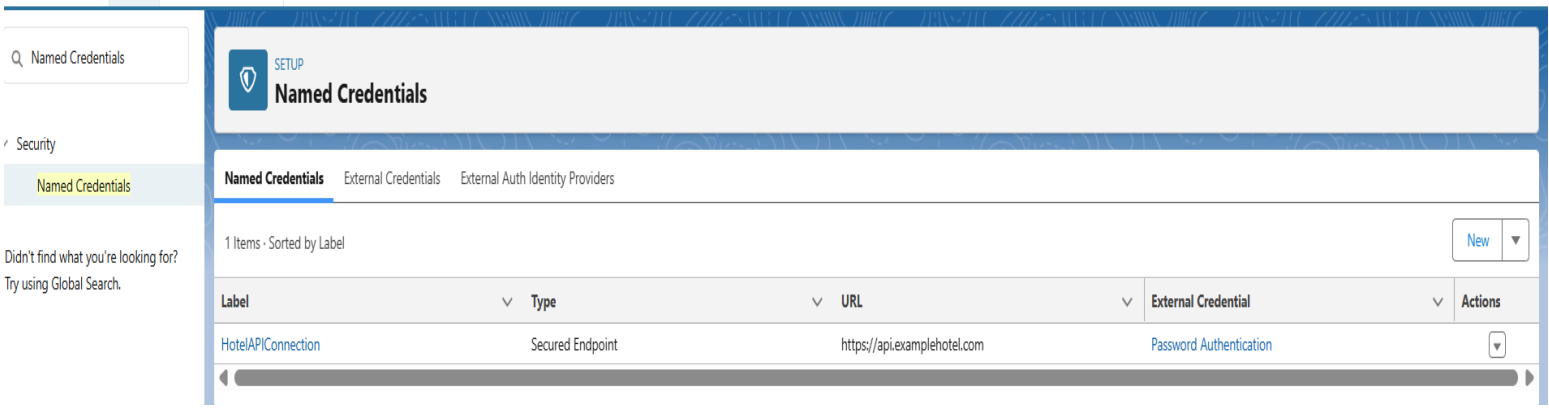
Purpose:

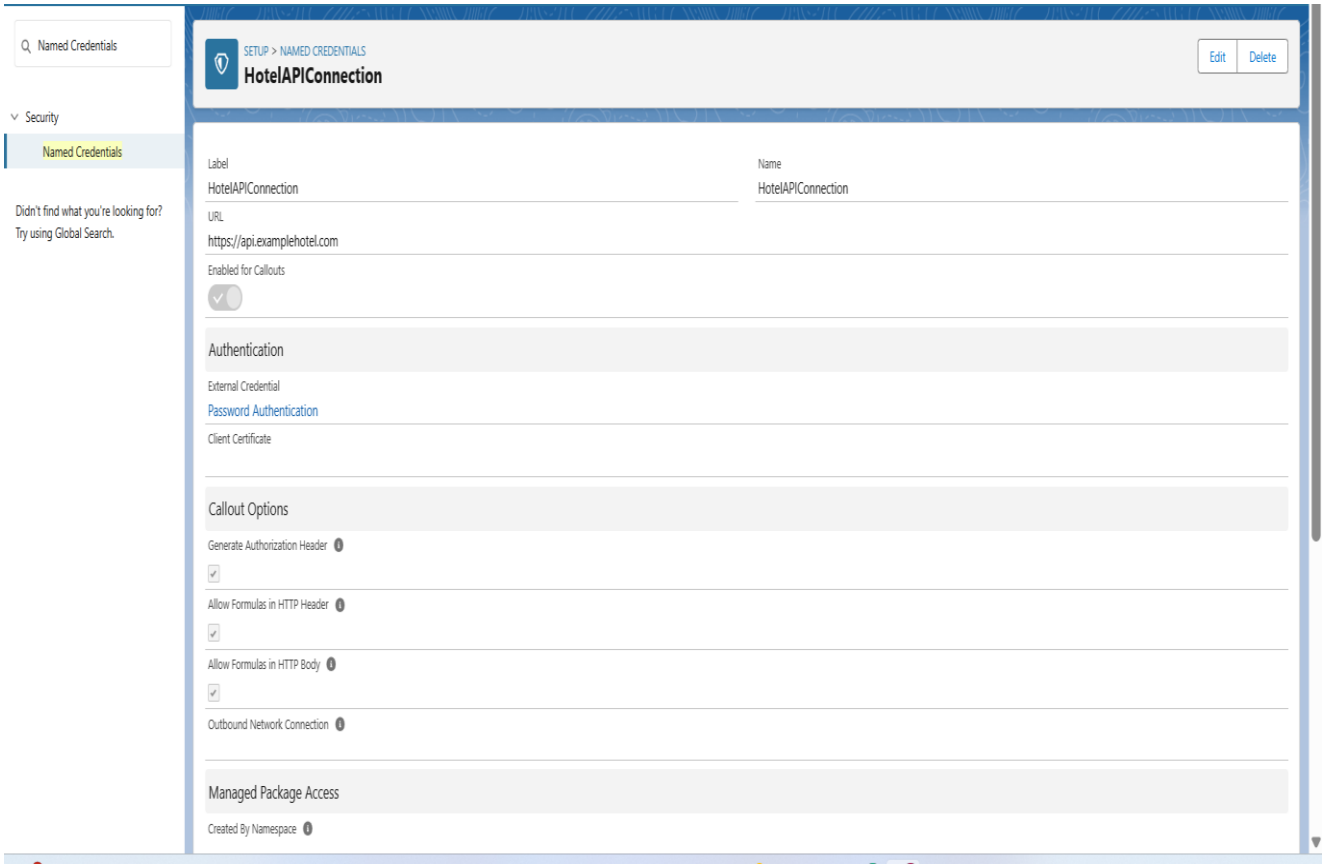
Securely store the base URL and authentication settings for the hotel API without hardcoding credentials inside Apex.

Steps Followed:

- Setup → Named Credentials → New.
- Added **Label:** HotelAPI and **URL:** https://api.hotel.com (replace with actual API URL).
- Chose **Identity Type:** Anonymous and **Authentication Protocol:** No Authentication, since using a demo API without login credentials.
- Saved the configuration.

- **Why Important:**
  - Avoids hardcoding API endpoints in Apex.
  - Centralized management of authentication.
  - Secure and easy to update if endpoint changes.





---

2. External Services (Not Used)

Purpose:

- Allows Salesforce to consume APIs described by OpenAPI/Swagger without writing code.

Reason for Skipping:

- The hotel API does not provide a Swagger/OpenAPI definition.
- We directly integrated via **Named Credentials + Apex Callout**.

---

3. Web Services (REST/SOAP) (Used – REST)

Purpose:

- **Salesforce supports integrating with external systems via REST or SOAP.**
- **REST → lightweight, JSON-based (used in this project).**

Implementation:

- **Created an Apex class to send REST callouts to the hotel API.**
- **Example: Fetch hotel data using req.setEndpoint('callout:HotelAPI/listHotels').**

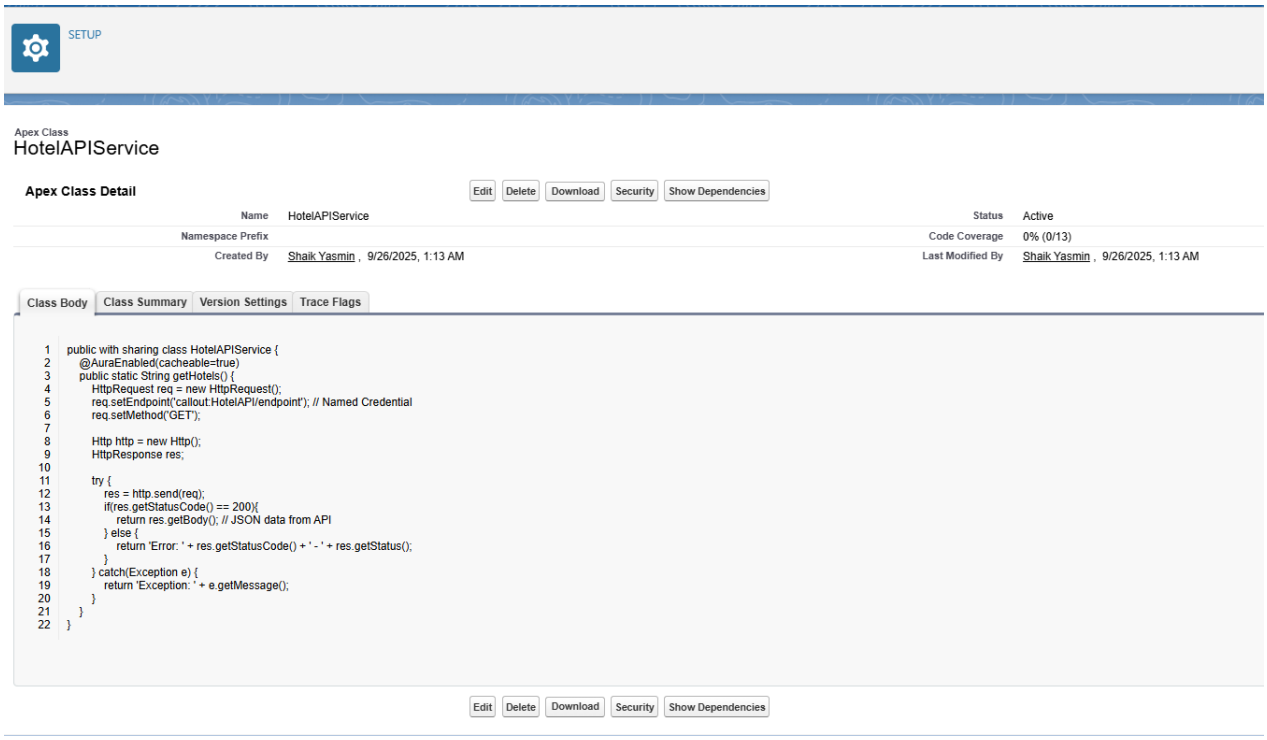
Code Skeleton (Apex):

```
public with sharing class HotelAPIService {
    @AuraEnabled(cacheable=true)
    public static String getHotels() {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:HotelAPI/listHotels');
        req.setMethod('GET');

        Http http = new Http();
        try {
            HttpResponse res = http.send(req);
            if(res.getStatusCode() == 200){
                return res.getBody(); // JSON response
            } else {
                return 'Error: ' + res.getStatusCode();
            }
        } catch(Exception e){
            return 'Exception: ' + e.getMessage();
```

```
    }
  }
}
```

Why Important:

- **Enables Salesforce to send HTTP requests (GET/POST) to external systems.**
- **Fetches real-time hotel data for display in Lightning Web Components.**
- 

---

4. Callouts (Used)

Purpose:

- **Allow Salesforce to send HTTP requests (GET/POST) to the hotel API for real-time data.**

Implementation:

- **Added Named Credential HotelAPI to store API URL and authentication.**

- **Created Apex class HotelAPIService to make callouts:**

HttpRequest req = new HttpRequest();

req.setEndpoint('callout:HotelAPI/listHotels');

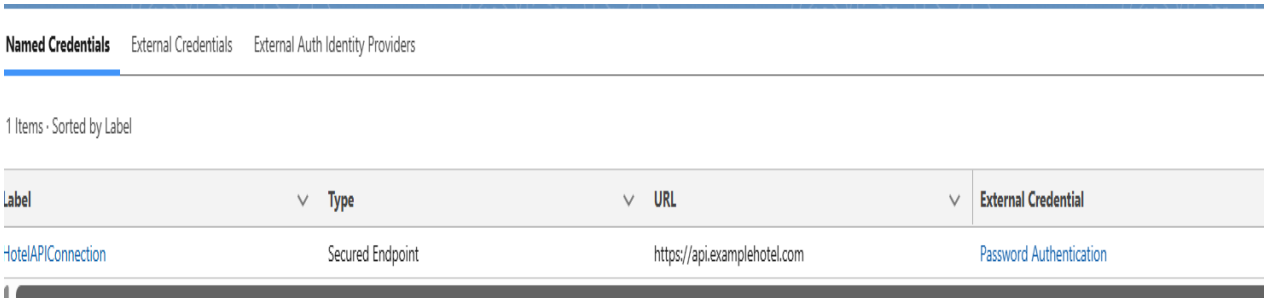
req.setMethod('GET');

HttpResponse res = new Http().send(req);

- **Parsed JSON response and displayed hotel data in LWC.**

Why Important:

- **Enables Salesforce to fetch live hotel details without hardcoding credentials.**



---

5. Platform Events (Used)

Purpose:

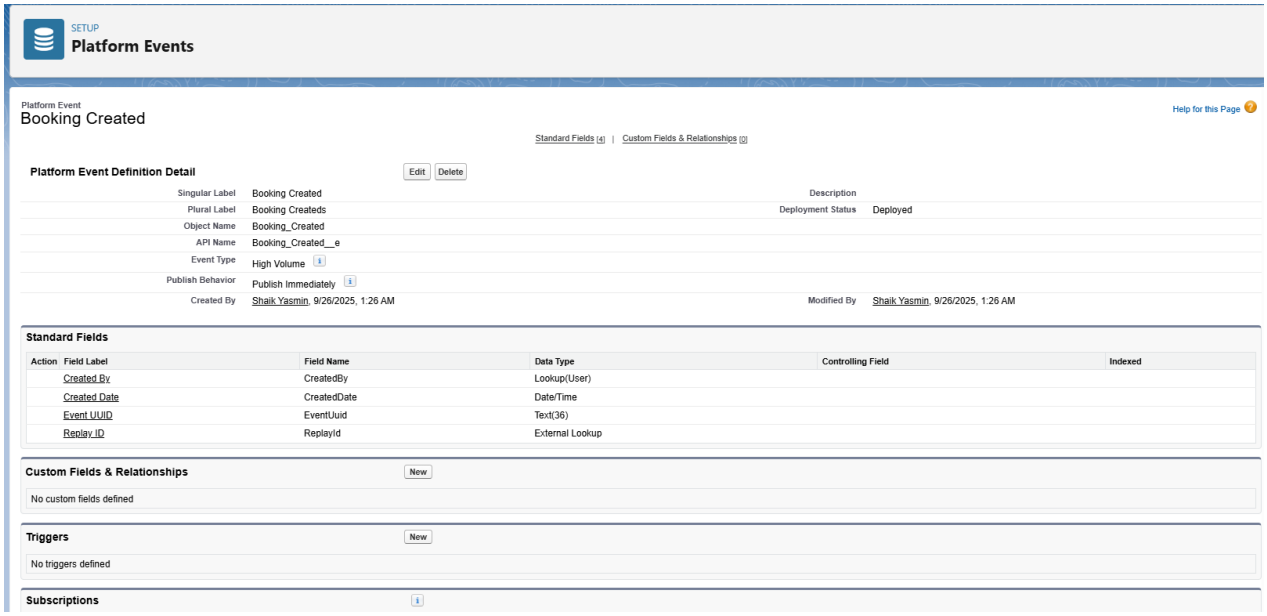
- Support **real-time notifications** inside Salesforce and external systems.

Implementation:

- Created BookingCreated__e platform event.
- Published event after a hotel booking is confirmed.
- Subscribed (via Apex trigger / external listener) to react instantly.

Why Important:

- **Decouples booking logic** from notifications.
- Enables **instant alerts** (e.g., send confirmation or fraud check).



1**. Customer Events**

Purpose:

Notify when a customer creates/updates a booking.

Implementation:

Created Customer_Booking__e event.

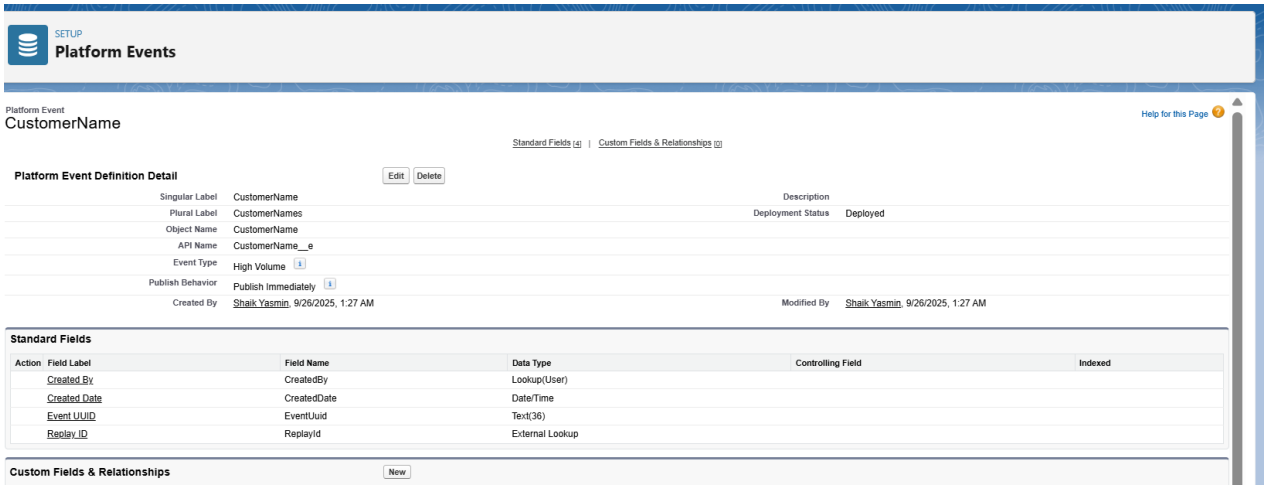
Published when customer booking is confirmed.

External system (or Salesforce flow) can subscribe to trigger notifications.

Why Important:

Provides real-time booking updates to other systems.

Improves customer experience with instant confirmation.



2. Hotel Events

Purpose:

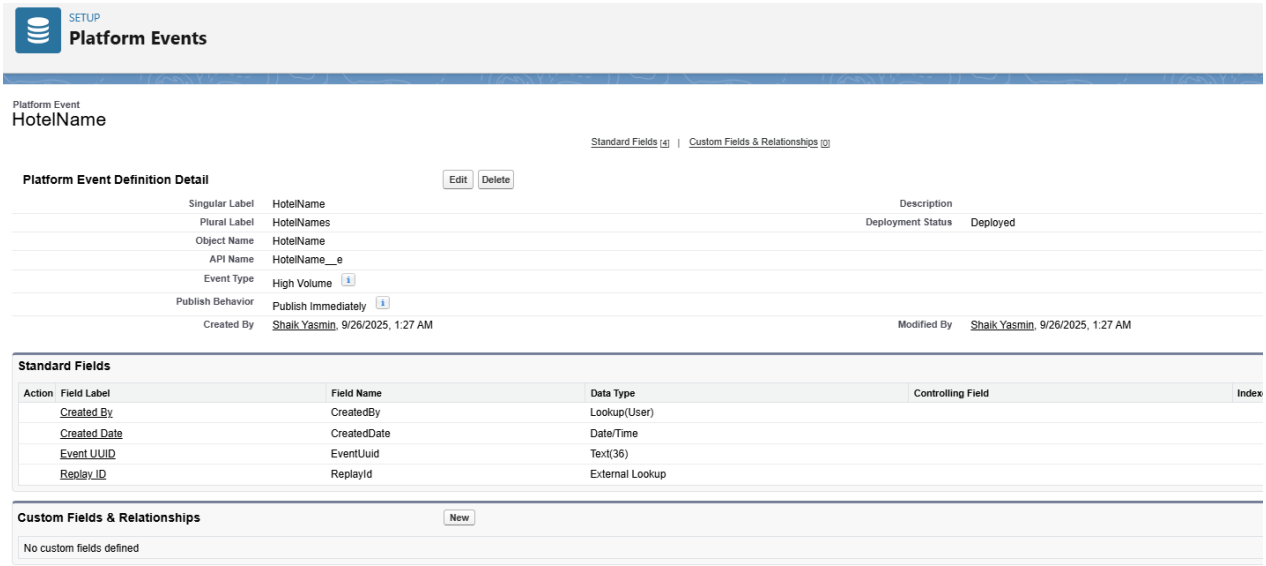
- Notify when **hotel availability or booking status changes**.

Implementation:

- Created Hotel_Update__e event.
- Published when hotel confirms/cancels a room.
- External listeners (fraud check, analytics, or partner systems) get notified.

Why Important:

- Keeps **hotel system and Salesforce synced**.
- Enables **instant alerts** for room allocation, cancellations, etc.



3. Room Events

Purpose:

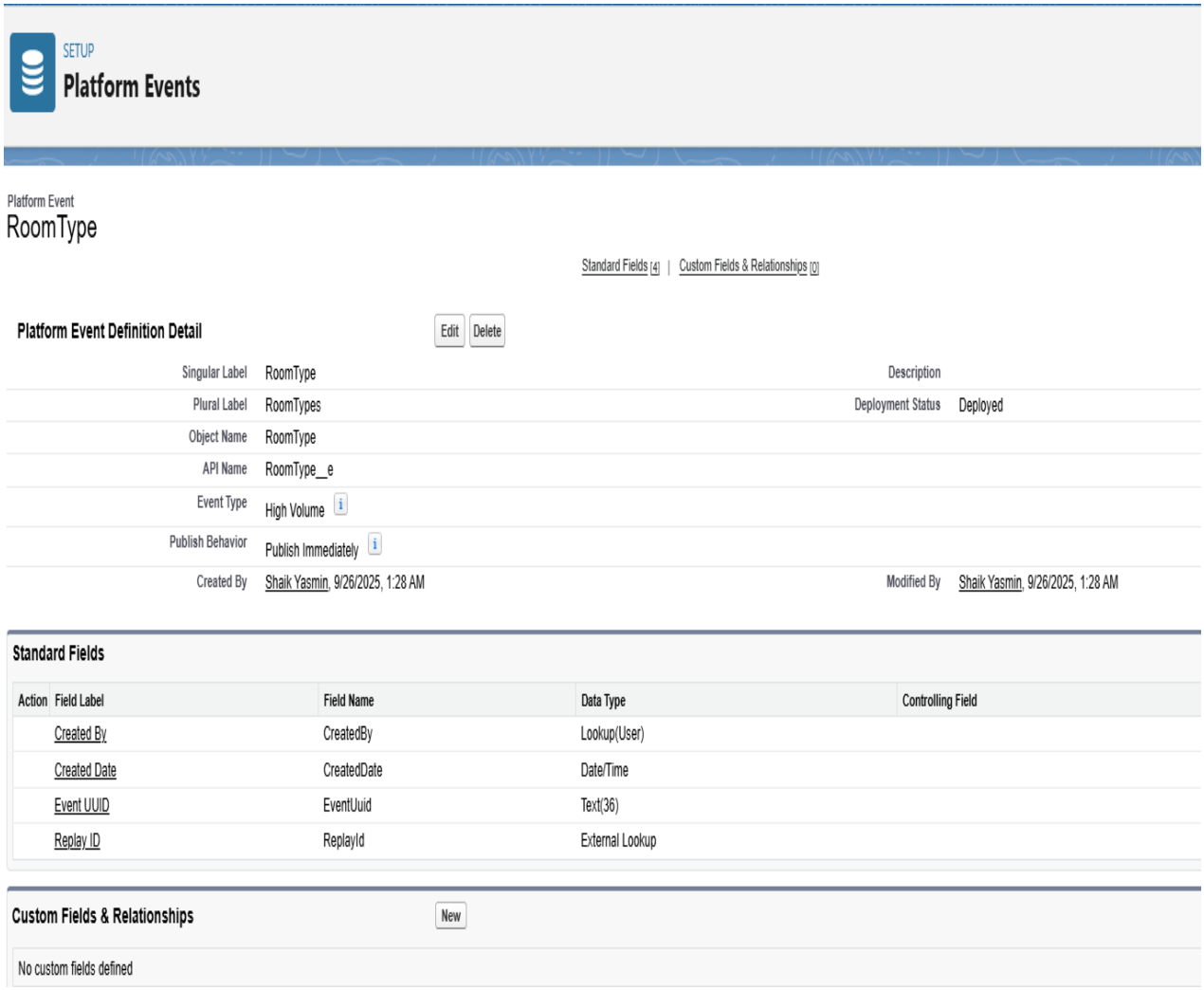
- Notify when **room status changes** (booked, available, cleaned).

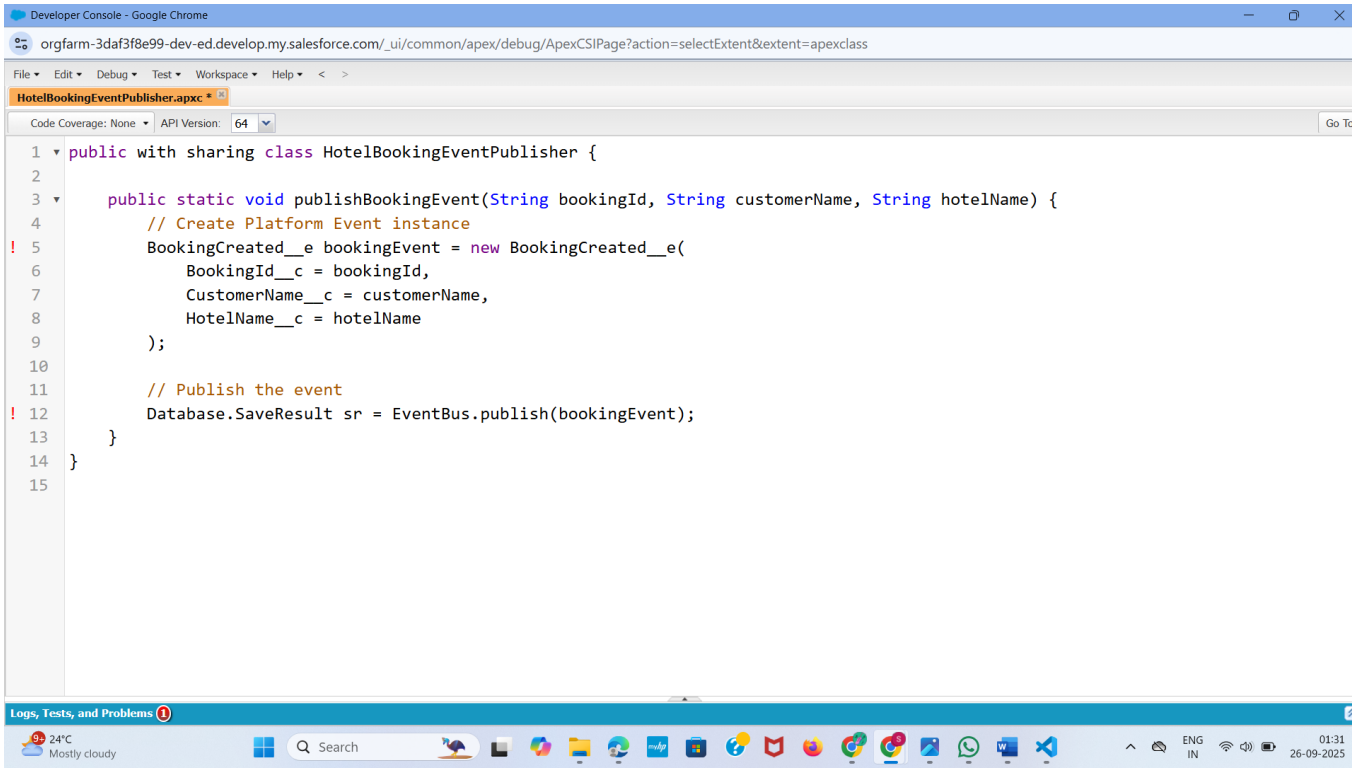
Implementation:

- Created Room_Status__e event.
- Published when a room is **booked, released, or updated**.

Why Important:

- Helps maintain **accurate availability in real-time**.
- Supports **hotel operations** like cleaning, check-in, and check-out.





6. Salesforce Connect (Not Used)

Purpose:

- Connects external databases (Oracle, SAP, AWS) as virtual objects in Salesforce without storing data locally.

Reason for Skipping:

- The hotel API provides **API access only**, not a database connection.

---

7. API Limits

Purpose:

- Salesforce imposes limits to ensure system performance.

Important Limits:

- REST API calls per 24 hours → based on license (e.g., 15,000 for Enterprise Edition).
- Callout timeout → 120 seconds max.

- Concurrent callouts → up to 10 per Apex transaction.

Why Important:

- Helps design integrations without exceeding limits.
- Ensures API quota is not exhausted while fetching hotel data.

---

8. OAuth & Authentication (Not Used – No Auth)

Purpose:

- Standard method for connecting Salesforce to third-party services.

Reason for Skipping:

- Hotel API is a demo and does not require credentials.
- Chose **No Authentication** in Named Credentials.

Note: In real projects, APIs usually require **OAuth2 or Username-Password** for security.

---

9. Remote Site Settings (Used)

Purpose:

- Allow Salesforce to call external services securely.

Steps Followed:

- Using **Named Credentials**, Remote Site Settings are automatically handled.
- If calling the API endpoint directly, you must manually add the **base URL** in **Setup → Remote Site Settings**.

Why Important:

- Prevents unauthorized external callouts.
- Adds a layer of security by whitelisting allowed domains