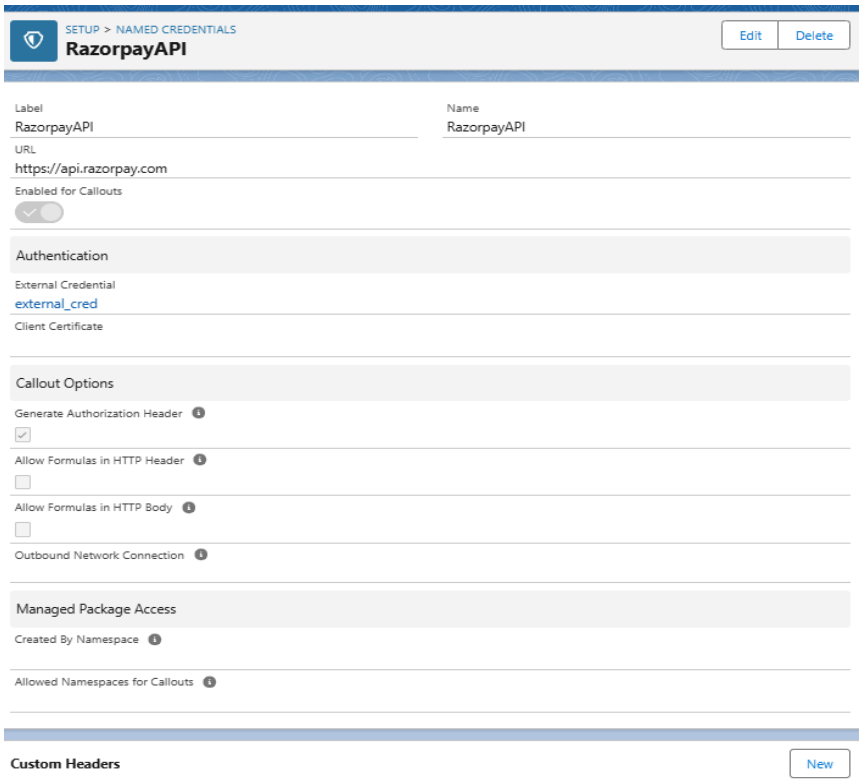
Project Title – “Smart Subscription Tracker”

Phase 7: Integration & External Access

Named Credentials

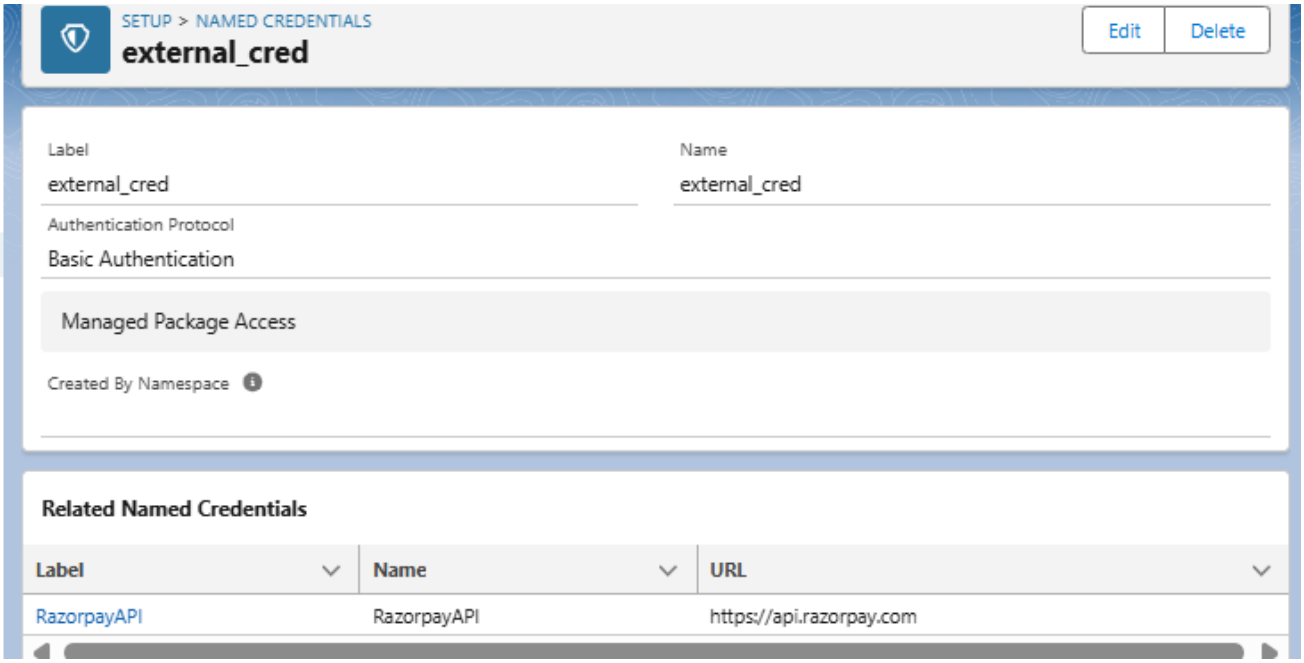
- Setup → Quick Find → **Named Credentials** → **New Named Credential**.
- Fill these fields (example for the Razorpay scenario below):
- **Label / Name**: e.g., RazorpayAPI (Name used in Apex).
- **URL**: https://api.razorpay.com (the base URL).
- **Identity Type**: Named Principal (or Per User).
- **Authentication Protocol**: Password Authentication (or OAuth 2.0 if you created an Auth Provider).
- If Password Auth: fill **Username** and **Password** (or API key / secret) and check **Generate Authorization Header** if you want Salesforce to automatically add Basic Auth.
- Save.



External credentials -

Step-by-step: create an External Credential (UI)

1. Setup → Quick Find → Named Credentials → open that page. Click the External Credentials tab → New.
2. In the New External Credential screen fill:
    - Label and API Name (unique).
    - Authentication Protocol: choose Basic, OAuth 2.0, JWT, AWS SigV4, Custom, or No Authentication depending on your provider.
    - Optionally add a Description.
    - Save.



Callouts

- Configured **Remote Site Settings** to allow Salesforce to communicate with external billing/payment systems securely.
- Designed an **Apex callout class (`BillingService`)** that uses `HttpRequest` and `HttpResponse` to fetch customer details from the billing API.
- Implemented error handling in the callout code to capture API failures and return meaningful exceptions.
- Demonstrated callouts for scenarios such as **subscription expiry reminders** or **payment status checks**, where Salesforce can send/receive data from external services.
- Verified callouts using the **Execute Anonymous window**, ensuring the API endpoints were reachable and returning expected responses.

- Kept the design flexible so future integrations (e.g., Stripe, Razorpay, or notification services like Slack/WhatsApp) can be added with minimal changes.

```
public class SimpleBillingCallout {
    public static String getCustomerInfo(String customerId) {
        try {
            HttpRequest req = new HttpRequest();
            req.setEndpoint('callout:RazorpayAPI/v1/customers/' + customerId); // Named Credential
            req.setMethod('GET');
            req.setHeader('Accept', 'application/json');

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

            if(res.getStatusCode() >= 200 && res.getStatusCode() < 300) {
                return res.getBody(); // Success, return response body
            } else {
                return 'Error: ' + res.getStatusCode() + ' - ' + res.getBody();
            }
        } catch (Exception ex) {
            return 'Exception: ' + ex.getMessage();
        }
    }
}
```

```
//SimpleBillingCallout
String result = SimpleBillingCallout.getCustomerInfo('cust_123'); // replace with actual customer ID
System.debug(result);
```

🚨 Platform Events

1) Open Platform Events in Setup

1. In Salesforce, click the **Setup** gear (top right) → **Setup**.
2. In Quick Find, type **Platform Events** → click **Platform Events**.

2) Create the Platform Event

1. Click **New Platform Event**.
2. Fill these values:
   - **Label:** Subscription Expiry Event
   - **Plural Label:** Subscription Expiry Events
   - **API Name:** Subscription_Expiry_Event__e (Salesforce will suggest this)
   - **Description:** Event for subscription notifications: expiry reminders, payment failures, retries.
   - **Publish Behavior: After Commit** ← **choose this** (recommended — publishes only after the DB transaction succeeds).
3. Click **Save**.

3) Add the fields (one by one)

After saving you'll be on the Platform Event detail page. Click **Fields & Relationships** → **New** for each of the fields below — use the exact types and settings suggested.

Field A — Subscription_Id__c

- **Data Type:** Text → **Next**
- **Field Label:** Subscription Id
- **Length:** 18 (Salesforce Id length)
- **Field Name (API):** Subscription_Id__c
- **Required:** No (optional)
- **Help Text:** Salesforce Subscription record Id (18 chars).
- **Save.**

Field B — External_Customer_Id__c

- **Data Type:** Text → **Next**
- **Label:** External Customer Id
- **Length:** 50
- **API Name:** External_Customer_Id__c
- **Help Text:** ID used by the billing system (Stripe/Razorpay).
- **Save.**

Field C — End_Date__c

- **Data Type:** Date → **Next**
- **Label:** End Date
- **API Name:** End_Date__c
- **Help Text:** Subscription end date.
- **Save.**

Field D — Event_Type__c

Two options: Picklist (strict values) or Text (simple). For the easiest admin control, use **Text**; if you prefer enforced values (ExpiryReminder, PaymentFailed), use **Picklist**.
(Text option — easiest)

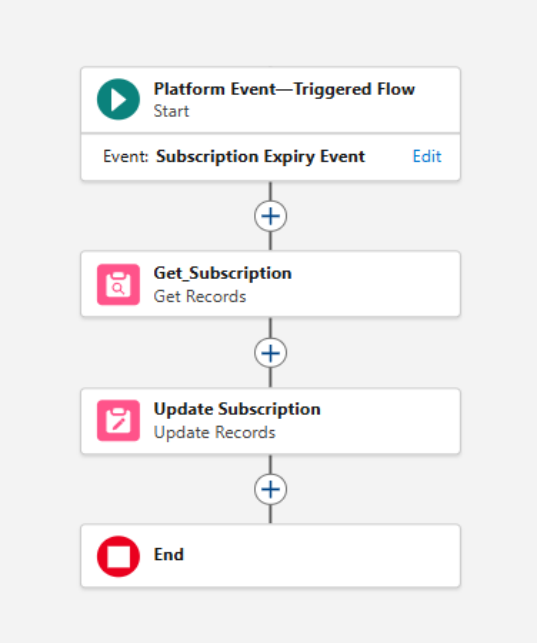
- **Data Type:** Text → **Next**
- **Label:** Event Type
- **Length:** 30
- **API Name:** Event_Type__c
- **Help Text:** e.g., ExpiryReminder, PaymentFailed
- **Save.**

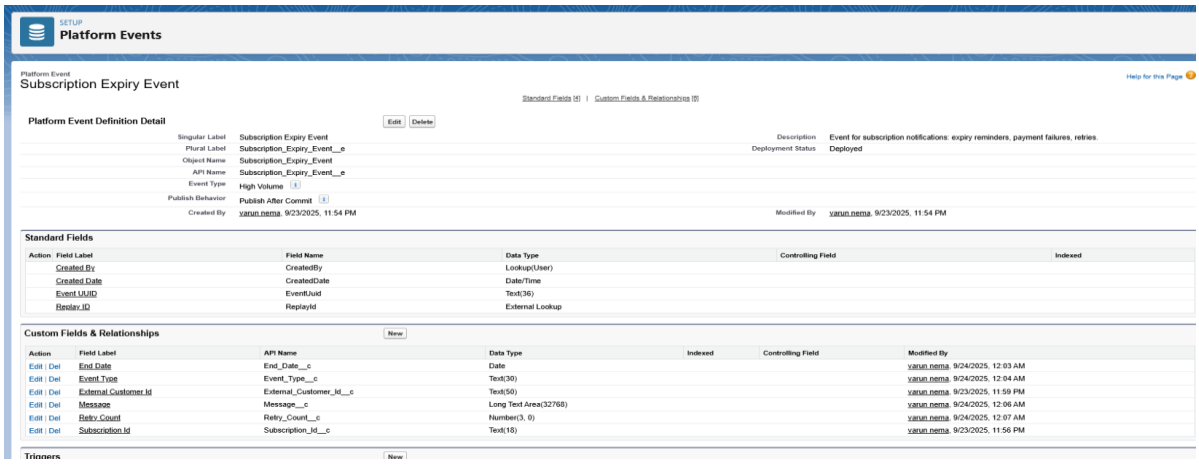
Field E — Message__c

- **Data Type:** Long Text Area → **Next**
- **Label:** Message
- **Length:** 1000 (or larger if you expect long payloads)
- **Visible Lines:** 3 (UI display only)
- **API Name:** Message__c
- **Help Text:** Extra info or debugging notes.
- **Save.**

Field F — Retry_Count__c

- **Data Type:** Number → **Next**
- **Label:** Retry Count
- **Length:** 3 (max digits)
- **Decimal Places:** 0
- **API Name:** Retry_Count__c
- **Help Text:** Number of retry attempts for processing.
- **Save.**





SETUP
Platform Events

Platform Event
Subscription Expiry Event

Help for this Page

Standard Fields [4] | Custom Fields & Relationships [9]

Platform Event Definition Detail Edit Delete

Singular Label	Subscription Expiry Event	Description	Event for subscription notifications: expiry reminders, payment failures, retries.
Plural Label	Subscription_Expiry_Event__e	Deployment Status	Deployed
Object Name	Subscription_Expiry_Event		
API Name	Subscription_Expiry_Event__e		
Event Type	High Volume		
Publish Behavior	Publish After Commit		
Created By	varun nema, 9/23/2025, 11:54 PM	Modified By	varun nema, 9/23/2025, 11:54 PM

Standard Fields

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
	Created By	CreatedBy	Lookup(User)		
	Created Date	CreatedDate	Date/Time		
	Event UUID	EventUuid	Text(36)		
	Replay ID	ReplayId	External Lookup		

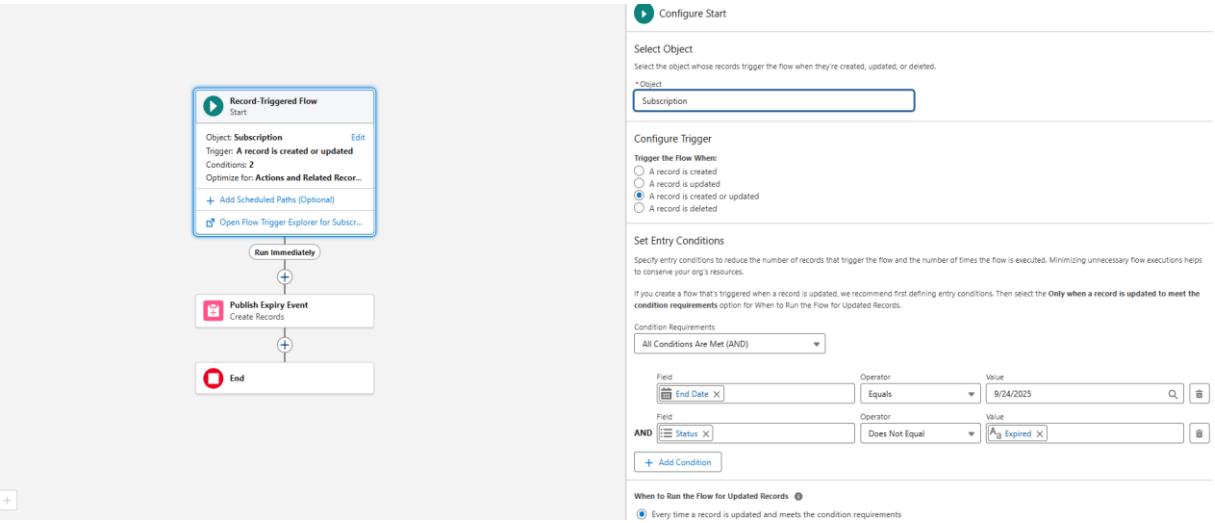
Custom Fields & Relationships New

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit | Del	End Date	End_Date__c	Date			varun nema, 9/24/2025, 12:03 AM
Edit | Del	Event Type	Event_Type__c	Text(30)			varun nema, 9/24/2025, 12:04 AM
Edit | Del	External Customer Id	External_Customer_Id__c	Text(50)			varun nema, 9/23/2025, 11:59 PM
Edit | Del	Message	Message__c	Long Text Area(32768)			varun nema, 9/24/2025, 12:06 AM
Edit | Del	Retry Count	Retry_Count__c	Number(3, 0)			varun nema, 9/24/2025, 12:07 AM
Edit | Del	Subscription Id	Subscription_Id__c	Text(18)			varun nema, 9/23/2025, 11:56 PM

Triggers New

📌 *Publish Event*

- Open Flow Builder
  Setup → Quick Find → Flows → New Flow → choose Record-Triggered Flow → Create.
- Create a Date formula resource (Cutoff = Today + 7)
- Left panel → Manager → New Resource.
- Resource Type: Formula.
- API Name: CutoffDate
- Data Type: Date
- Formula:
- $Flow.CurrentDate + 7
- Save the resource.
- Configure the Start (Trigger)
- Object: Subscription__c
- Trigger: A record is created or updated
- Condition Requirements: Conditions are met (AND)
  - Condition 1: End_Date__c Less or Equal {!CutoffDate}
  - Condition 2: Status__c Not Equal Expired
- When to Run the Flow for Updated Records: Only when a record is updated to meet the condition requirements (recommended — avoids repeats)
- Optimize the Flow for: Actions and Related Records
- Click Done.
- Add Create Records (publish the event)
- Click the + from the Start node → Create Records → Label: Publish Expiry Event.
- How Many Records to Create: One
- Create Record Of: Subscription_Expiry_Event__e
- Set Field Values (map each):
  - Subscription_Id__c = {!$Record.Id}
  - External_Customer_Id__c = {!$Record.External_Customer_Id__c}
  - End_Date__c = {!$Record.End_Date__c}
  - Event_Type__c = ExpiryReminder (type the text literal)
  - Message__c = 7-day reminder (type the text literal)
  - Retry_Count__c = 0 (optional)
- Click Done.
- *Connect & Save*
- *Make sure Start → Publish Expiry Event is connected.*
- *Click Save → give it a name (e.g., RTF_Publish_Expiry_Event) → Save.*
- *Activate*
- *Click Activate.*


Record-Triggered Flow
Start
Object: Subscription
Edit
Trigger: A record is created or updated
Conditions: 2
Optimize for: Actions and Related Recor...
Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Subscr...
Run Immediately
Publish Expiry Event
Create Records
End
Configure Start
Select Object
Select the object whose records trigger the flow when they're created, updated, or deleted.
* Object
Subscription
Configure Trigger
Trigger the Flow When:
A record is created
A record is updated
A record is created or updated
A record is deleted
Set Entry Conditions
Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.
If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the Only when a record is updated to meet the condition requirements option for When to Run the Flow for Updated Records.
Condition Requirements
All Conditions Are Met (AND)
Field
Operator
Value
End Date
Equals
9/24/2025
AND
Status
Does Not Equal
Expired
Add Condition
When to Run the Flow for Updated Records
Every time a record is updated and meets the condition requirements