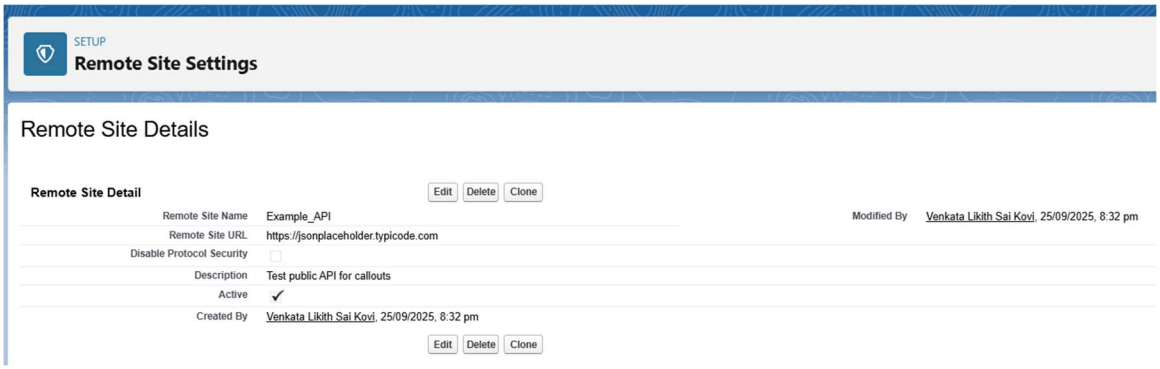
Phase 7 — Integration & External Access

Contents

A. Remote Site Settings (Quick Allow List)

Purpose: If you will call external endpoints with simple HTTP requests (and are not using Named Credentials), you must register the external domain in Remote Site Settings.

1. Click the Gear icon → Setup.
2. In Quick Find box type 'Remote Site Settings' → Click 'Remote Site Settings'.
3. Click New Remote Site.
4. Fill the form exactly:
   - Remote Site Name: Example_API
   - Remote Site URL: https://jsonplaceholder.typicode.com
   - Active: checked
   - Description: Test public API for callouts
5. Click Save.

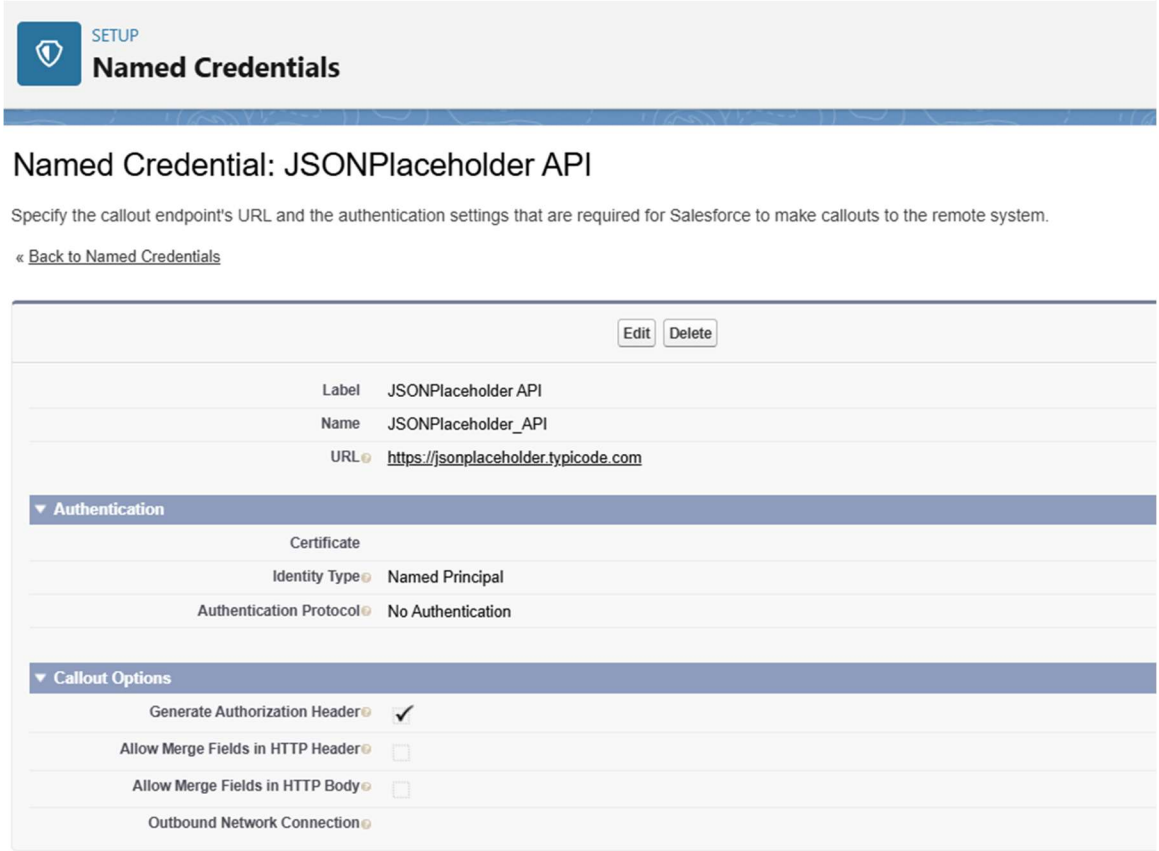


B. Named Credentials (Preferred for Secure Callouts)

Purpose: Named Credentials centralize endpoint URL and authentication. Use them instead of Remote Site Settings whenever possible — they simplify authentication and secure credentials.

Exact steps to create a Named Credential (no auth example):

1. Setup → Quick Find → Named Credentials → Click Named Credentials.
2. Click New Named Credential.
3. Fill values:
   - Label: JSONPlaceholder API
   - Name: JSONPlaceholder_API
   - URL: https://jsonplaceholder.typicode.com
   - Identity Type: Named Principal
   - Authentication Protocol: No Authentication
   - Allow Merge Fields in HTTP Header: unchecked
4. Click Save.



C. Connected App & Auth Provider (for OAuth 2.0)

Purpose: Use a Connected App to allow external systems to authenticate with Salesforce, or to connect Salesforce to an external OAuth provider via Named Credential + Auth Provider.

Create a Connected App (for external OAuth client):

1. Setup → Quick Find → App Manager → Click New Connected App (top-right).

2. Fill the basics:
   - Connected App Name: HelpingHands Integration App
   - API Name: HelpingHands_Integration_App
   - Contact Email: your-email@example.com

3. Under 'API (Enable OAuth Settings)'

check the box: 'Enable OAuth Settings'.
   - Callback URL: https://login.salesforce.com/services/oauth2/success
   - Selected OAuth Scopes: Add 'Full access (full)', 'Perform requests on your behalf at any time (refresh_token, offline_access)'

4. Save the Connected App. Wait ~2–10 minutes for the app to be active.

5. After saving, you will get Consumer Key (Client Id) and Consumer Secret (Client Secret) on the page — copy them for external use.

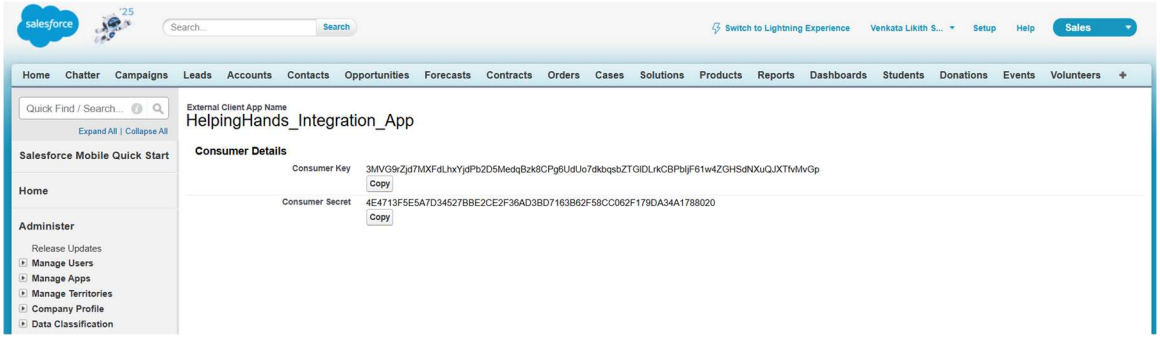
Create an Auth. Provider (if you want Salesforce to use an OAuth provider like Google or Custom OAuth):

1. Setup → Quick Find → Auth. Providers → New.

2. Provider Type: select the provider (e.g., Google) or choose 'OpenID Connect'/'OAuth 2.0' for custom providers.

3. Fill name and the client Id/secret if required.

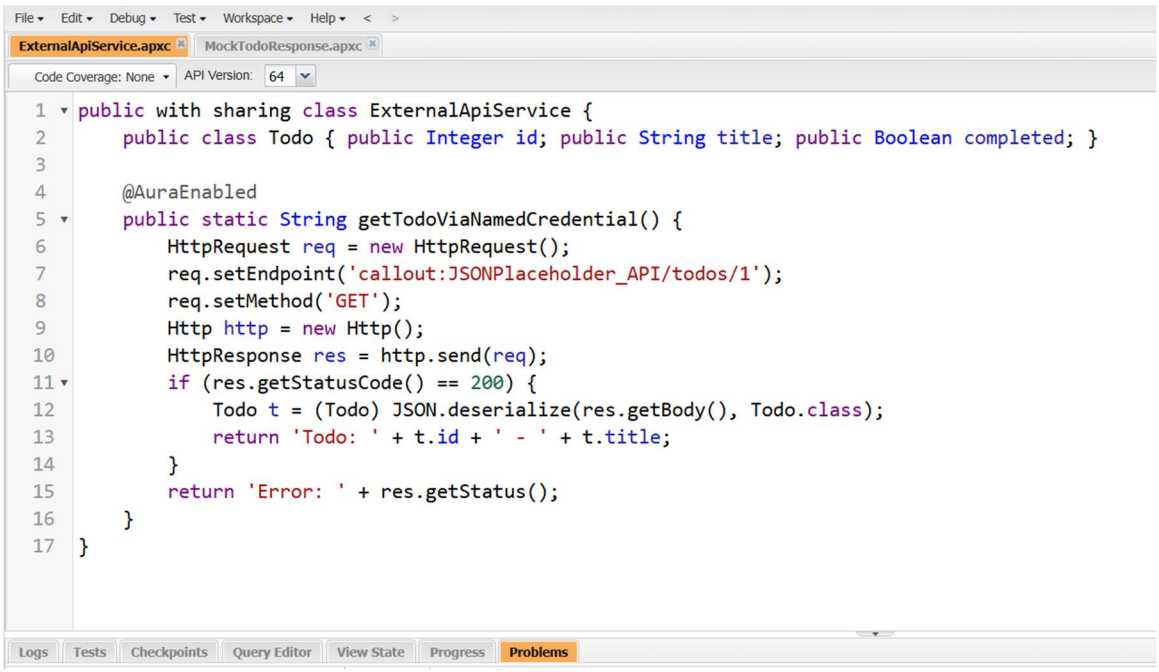
4. Save.



D. Apex HTTP Callouts and Testing (Developer Console)

Purpose: This section shows a complete pattern for doing a callout from Apex using a Named Credential and how to test it safely with HttpCalloutMock so tests run without issues.

Step D.1 — Create an Apex class that performs the callout:(MockTodoResponse.apxc)

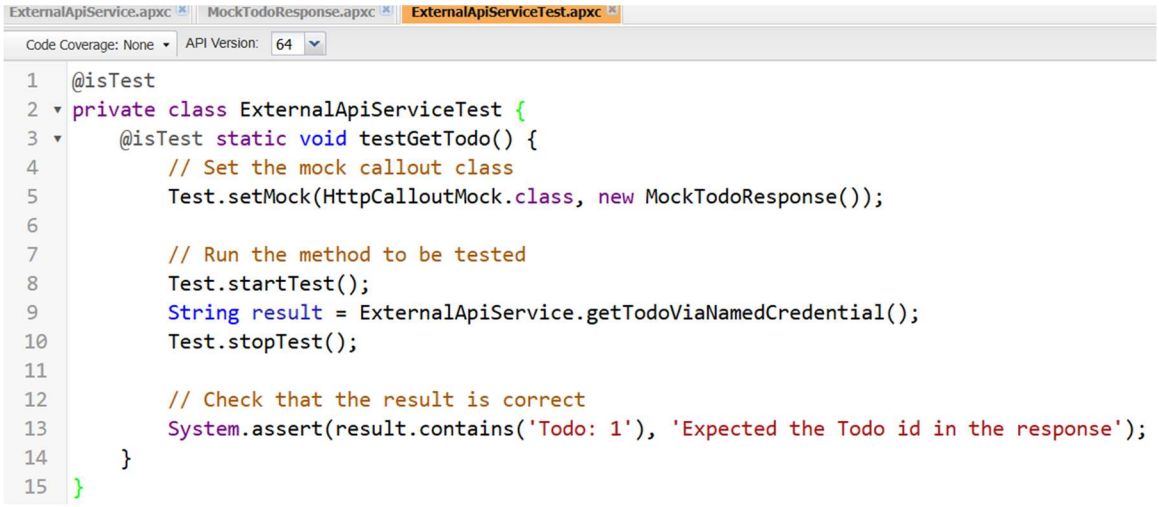



```
public with sharing class ExternalApiService {
    public class Todo { public Integer id; public String title; public Boolean completed; }

    @AuraEnabled
    public static String getTodoViaNamedCredential() {
        HttpRequest req = new HttpRequest();
        req.setEndpoint('callout:JSONPlaceholder_API/todos/1');
        req.setMethod('GET');
        Http http = new Http();
        HttpResponse res = http.send(req);
        if (res.getStatusCode() == 200) {
            Todo t = (Todo) JSON.deserialize(res.getBody(), Todo.class);
            return 'Todo: ' + t.id + ' - ' + t.title;
        }
        return 'Error: ' + res.getStatus();
    }
}
```

Step D.2 — Create an HttpCalloutMock for tests(MockTodoResponse.apxc)

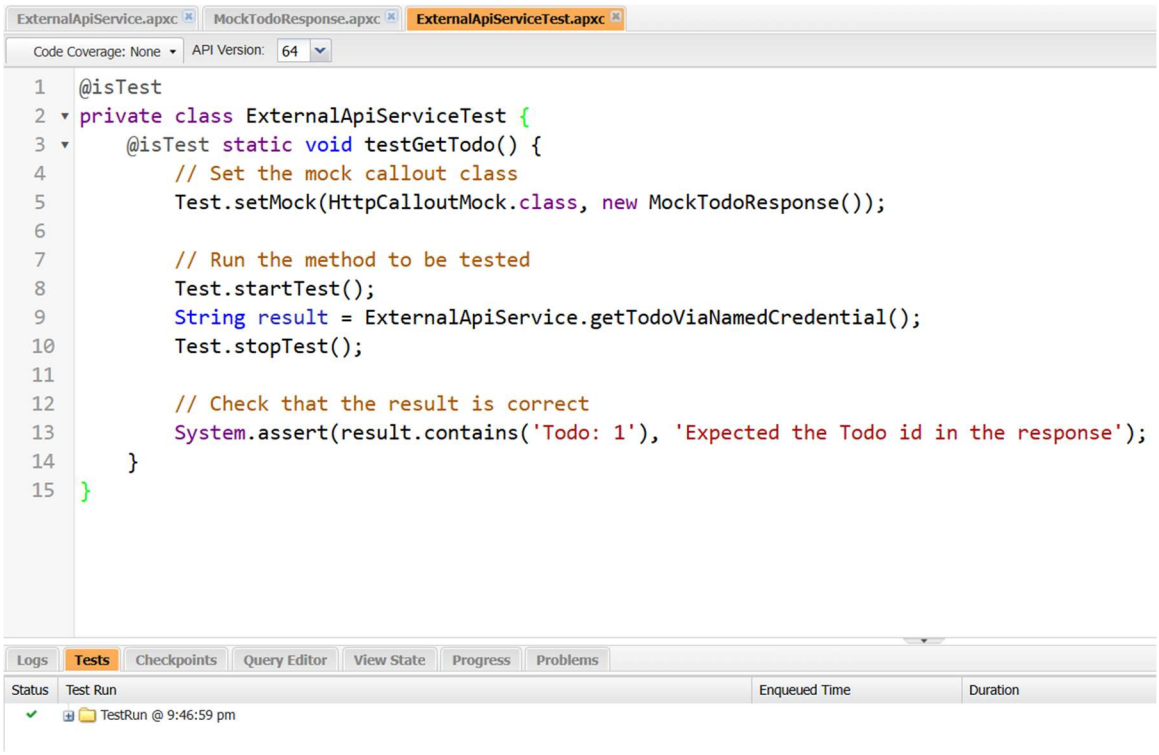
Step D.3 — Create a Test Class that uses the mock and validates behavior:



Step D.4.

open Developer Console → File → New → Apex Class (MockTodoResponse) → Save; then create ExternalApiService class and save;

Then create ExternalApiServiceTest class and save. Run the Apex test via Setup → Apex Test Execution or Developer Console → Test → New Run.



E. External Services (Register an OpenAPI / Swagger)

Purpose: External Services lets you import a REST API described by an OpenAPI (Swagger) schema and then use the generated actions in declarative tools (Flows).

Steps to Register the External Service

Your guide doesn't provide the required API specification, so I've created a simple one for you to use.

1. Go to **Setup**, use the Quick Find box to search for External Services, and open it.
2. Click the **New External Service** button.
3. On the first screen, select **From API Specification** and click **Next**.
4. Fill out the registration details:
    - **External Service Name**: JSONPlaceholder_Service

- **Named Credential**: Select the JSONPlaceholder_API you created earlier.
- For the **Service Schema**, choose the option Complete JSON. This will open a text box.

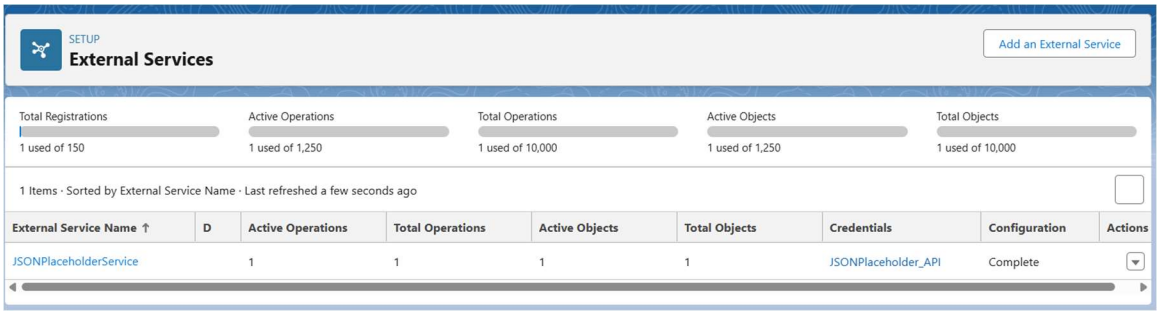
5. JSON

```
{
 "openapi": "3.0.0",
 "info": {
  "title": "JSONPlaceholder Todos API",
  "version": "1.0.0"
 },
 "paths": {
  "/todos/{id}": {
   "get": {
    "summary": "Get a single todo item by ID",
    "parameters": [
     {
      "name": "id",
      "in": "path",
      "required": true,
      "schema": {
       "type": "integer"
      }
     }
    ],
    "responses": {
     "200": {
```

```
        "description": "A single todo item",
        "content": {
          "application/json": {
            "schema": {
              "$ref": "#/components/schemas/Todo"
            }
          }
        }
      }
    }
  }
},
"components": {
  "schemas": {
    "Todo": {
      "type": "object",
      "properties": {
        "userId": { "type": "integer" },
        "id": { "type": "integer" },
        "title": { "type": "string" },
        "completed": { "type": "boolean" }
      }
    }
  }
}
```

```
}
```

6. Save



F. Salesforce Connect — External Data Source & External Objects

Purpose: Salesforce Connect lets you surface external data as external objects without copying data into Salesforce.

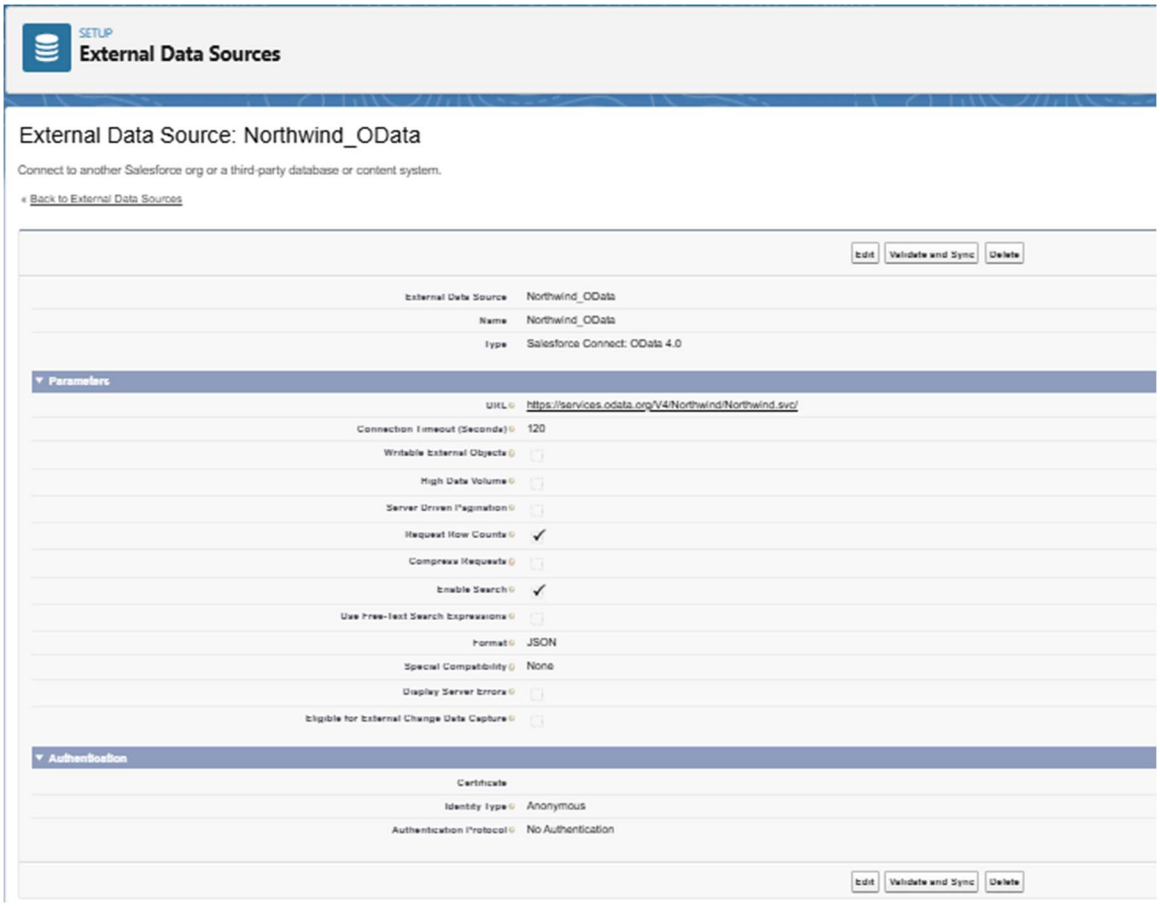
1. Setup → Quick Find → External Data Sources → New External Data Source.

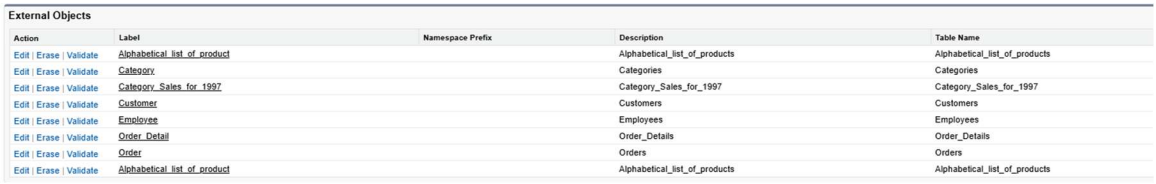
2. Fill values:
   - External Data Source Name: Northwind_OData
   - Type: OData 4.0 (or 2.0 depending on the endpoint)
   - URL: https://services.odata.org/V4/Northwind/Northwind.svc/
   - Identity Type: Anonymous (or Named Principal with Named Credential if auth required)

3. Click Save and then click 'Validate and Sync'.

4. In the Sync UI, select the Entities you want to sync (Products, Orders, etc.) and click 'Sync'.

5. Salesforce will create External Objects (e.g., Product__x) that map to external entities.

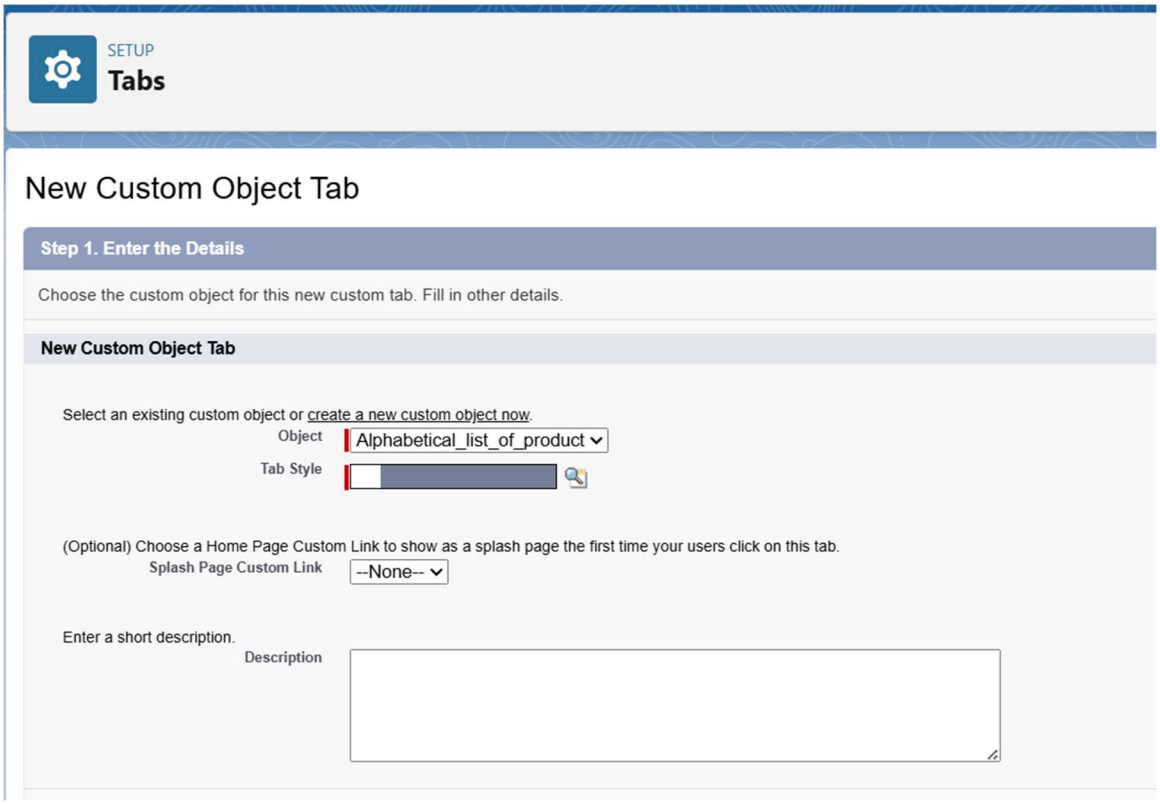




Test Example:

Click the new External Object tab (create a Tab via Setup → Tabs → New → External Object Tab) or browse via App Launcher to view records.

You can also create external lookup relationships between standard/custom objects and external objects.



Custom Object Tabs

New | What Is This?

Action	Label	Tab Style
Edit | Del	Alphabetical_list_of_products	Airplane
Edit | Del	Donations	Flag
Edit | Del	Donation Summaries	Laptop
Edit | Del	Events	Building Block
Edit | Del	Mentors	Hexagon
Edit | Del	Students	Desk
Edit | Del	Volunteer Events	Books
Edit | Del	Volunteers	Bell
Edit | Del	Volunteer Tasks	Chalkboard

G. Platform Events — Publish & Subscribe

Purpose: Platform Events deliver custom event messages within Salesforce (or to external systems). They are great for near real-time integration patterns.

Create a Platform Event: exact clicks:

1. Setup → Quick Find → Platform Events → Click Platform Events.
2. Click 'New Platform Event'.
3. Fill:
   - Label: Volunteer_Update
   - Plural Label: Volunteer_Updates
   - Object Name: Volunteer_Update__e
   - Publish Behavior: Publish After Commit (recommended)
4. Click Save.

Add fields to the Platform Event: e.g., VolunteerId__c (Text), Update_Type__c (Text), Notes__c (Long Text).

Publish a Platform Event via Apex (Developer Console):

Subscribe to Platform Events with an Apex trigger:

VolunteerUpdateEventTrigger.apxt *

Code Coverage: None API Version: 64

```
trigger VolunteerUpdateEventTrigger on Volunteer_Update__e (after insert) {
    for (Volunteer_Update__e ev : Trigger.New) {
        System.debug('Platform Event received: ' + ev);
    }
}
```

5. Publish an Event to Test It

Now I manually publish one event to see if your trigger is working.

1. In the **Developer Console**, click **Debug** > **Open Execute Anonymous Window**.
2. In the window that pops up, paste the following code. (The VolunteerId__c is just a placeholder, you don't need to change it).

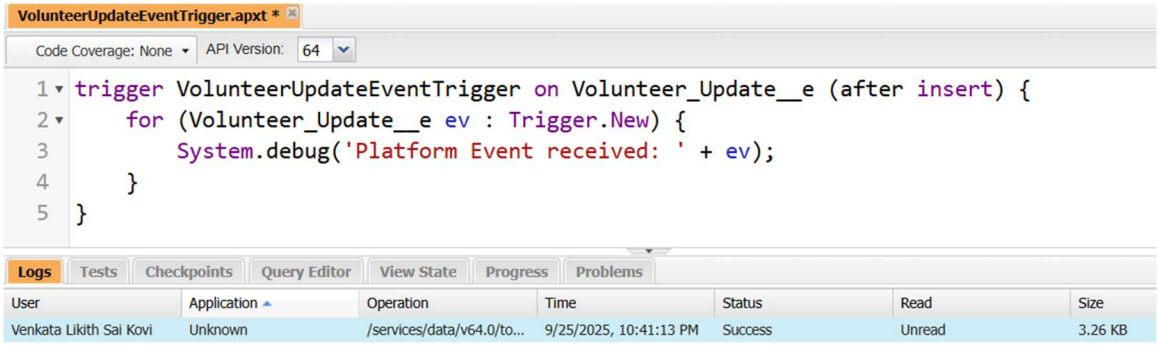
Apex

```
Volunteer_Update__e ev = new Volunteer_Update__e(

  VolunteerId__c = 'a0B...',
```

```
    Update_Type__c = 'Assigned',
    Notes__c = 'Auto-assigned task'
);
Database.SaveResult sr = EventBus.publish(ev);
```

6. Check the **Open Log** box at the bottom and click **Execute**.



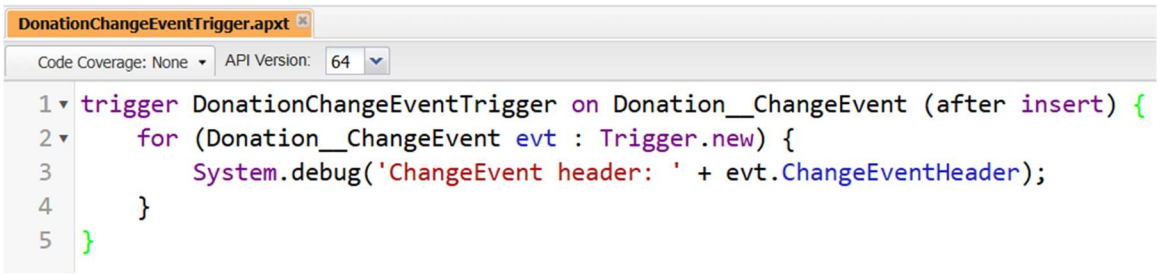
H. Change Data Capture (CDC) — Enable & Subscribe

Purpose: CDC publishes change events for record lifecycle changes (create/update/delete/undelete).

Enable CDC for Donation__c :

1. Setup → Quick Find → Change Data Capture → Click 'Change Data Capture'.
2. From the list of objects, check 'Donation' (Donation__c) and click Save.
3. Now Salesforce will produce Donation__ChangeEvent events whenever Donation__c records change.

Subscribe in Apex by creating a trigger for the change event (Developer Console → New → Apex Trigger):



Test: Update a Donation record. Then check Debug Logs or the trigger's System.debug output to confirm the change event fired.

DonationChangeEventTrigger.apxt

Code Coverage: None API Version: 64

```
1 trigger DonationChangeEventTrigger on Donation__ChangeEvent (after insert) {
2     for (Donation__ChangeEvent evt : Trigger.new) {
3         System.debug('ChangeEvent header: ' + evt.ChangeEventHeader);
4     }
5 }
```

Logs | Tests | Checkpoints | Query Editor | View State | Progress | Problems

User	Application	Operation	Time	Status	Read	Size
Venkata Likith Sai Kovi	Unknown	QueueableHandler	9/25/2025, 10:57:32 PM	Success	Unread	4.57 KB
Venkata Likith Sai Kovi	Browser	/aura	9/25/2025, 10:57:31 PM	Success	Unread	25.17 KB
Venkata Likith Sai Kovi	Unknown	common.api.soap.Direc...	9/25/2025, 10:57:31 PM	Success	Unread	516 bytes

I. API Limits & Monitoring

Purpose: Understand API usage and limits so you design efficient integrations.
How to check API usage in Salesforce (click-by-click):

1. Setup → Quick Find → System Overview → Click System Overview. Review 'API Requests, Last 24 Hours' and other stats.

2. To get programmatic view of limits, you can call the REST endpoint /services/data/vXX.0/limits (use Workbench or Postman with Salesforce authentication).

Quick steps to call the REST limits endpoint (Workbench method):

1. Go to https://workbench.developerforce.com → Login with your dev org credentials.

2. Navigate to Utilities → REST Explorer.
3. Use GET /services/data/v54.0/limits (replace v54.0 with your API version).
4. Review the JSON result which shows Concurrent Async, Daily API Calls, etc.