

# Phase 7 — Integration & External Access

---

## Contents

- A. Remote Site Settings
- B. Named Credentials (preferred) & Authentication
- C. Connected App & Auth Provider (OAuth flow)
- D. Apex Callouts (HTTP) + HttpCalloutMock test
- E. External Services (Register OpenAPI)
- F. Salesforce Connect (External Data Source & External Objects)
- G. Platform Events (publish & subscribe)
- H. Change Data Capture (enable & subscribe)
- I. API Limits & Monitoring

## 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.

The screenshot shows the 'Remote Site Settings' page in Salesforce. At the top, there's a 'SETUP' button and a 'Remote Site Settings' link. Below that is a section titled 'Remote Site Details'. A table titled 'Remote Site Detail' contains the following information:

Remote Site Detail		<a href="#">Edit</a> <a href="#">Delete</a> <a href="#">Clone</a>
Remote Site Name	Example_API	Modified By <a href="#">Venkata Likith Sai Kovil</a> 25/09/2025, 8:32 pm
Remote Site URL	<a href="https://jsonplaceholder.typicode.com">https://jsonplaceholder.typicode.com</a>	
Disable Protocol Security	<input type="checkbox"/>	
Description	Test public API for callouts	
Active	<input checked="" type="checkbox"/>	
Created By	<a href="#">Venkata Likith Sai Kovil</a> , 25/09/2025, 8:32 pm	

At the bottom of the table, there are three buttons: 'Edit', 'Delete', and 'Clone'.

## 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.

The screenshot shows the Salesforce setup interface for creating a new named credential. The top navigation bar has a blue 'SETUP' button and the title 'Named Credentials'. Below the header, a sub-header reads 'Named Credential: JSONPlaceholder API'. A note below says 'Specify the callout endpoint's URL and the authentication settings that are required for Salesforce to make callouts to the remote system.' A back-link '« Back to Named Credentials' is present. The main form contains fields for 'Label' (JSONPlaceholder API), 'Name' (JSONPlaceholder\_API), and 'URL' (<https://jsonplaceholder.typicode.com>). An 'Edit' and 'Delete' button are at the top right. A 'Certificate' section is collapsed. The 'Authentication' section is expanded, showing 'Identity Type' (Named Principal) and 'Authentication Protocol' (No Authentication). The 'Callout Options' section is also expanded, showing checked boxes for 'Generate Authorization Header' (with a checkmark), 'Allow Merge Fields in HTTP Header' (unchecked), 'Allow Merge Fields in HTTP Body' (unchecked), and 'Outbound Network Connection' (unchecked).

## 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.

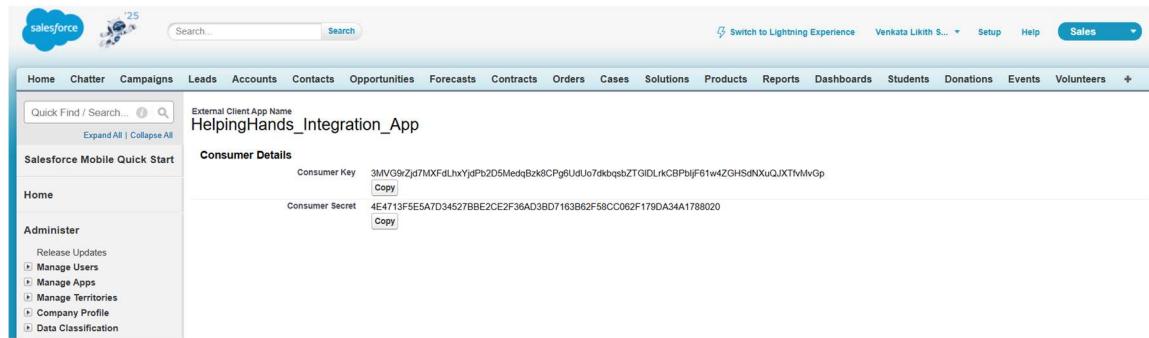
[Create an Auth. Provider \(if you want Salesforce to use an OAuth provider like Google or Custom OAuth\):](#)

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.

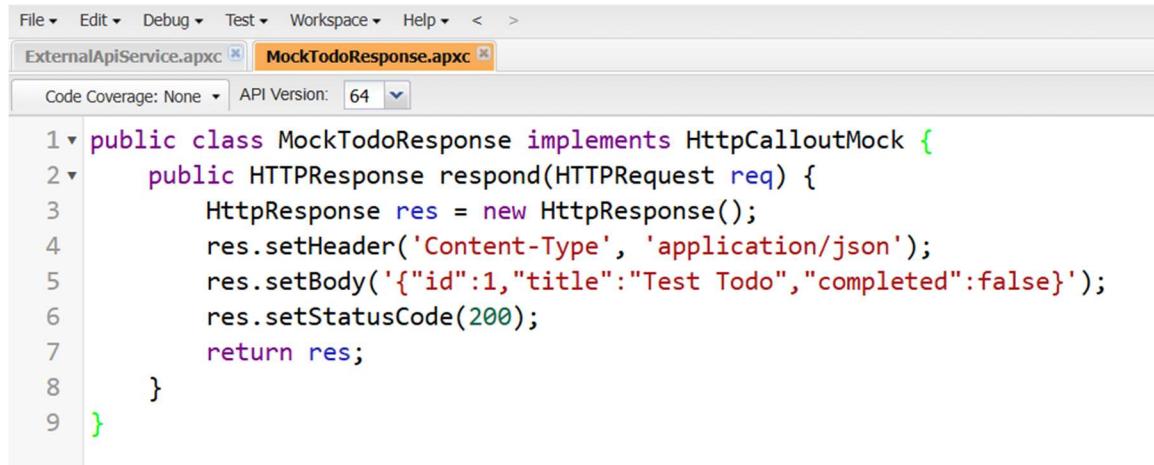
**Step D.1** — Create an Apex class that performs the callout:(MockTodoResponse.apxc)

A screenshot of the Salesforce Developer Console. The top menu bar includes 'File', 'Edit', 'Debug', 'Test', 'Workspace', 'Help', and tabs for 'ExternalApiService.apxc' and 'MockTodoResponse.apxc'. A dropdown for 'Code Coverage' is set to 'None'. An 'API Version' dropdown is set to '64'. The main code editor area contains the following Apex code:

```
1 public with sharing class External ApiService {
2     public class Todo { public Integer id; public String title; public Boolean completed; }
3
4     @AuraEnabled
5     public static String getTodoViaNamedCredential() {
6         HttpRequest req = new HttpRequest();
7         req.setEndpoint('callout:JSONPlaceholder_API/todos/1');
8         req.setMethod('GET');
9         Http http = new Http();
10        HttpResponse res = http.send(req);
11        if (res.getStatusCode() == 200) {
12            Todo t = (Todo) JSON.deserialize(res.getBody(), Todo.class);
13            return 'Todo: ' + t.id + ' - ' + t.title;
14        }
15        return 'Error: ' + res.getStatus();
16    }
17 }
```

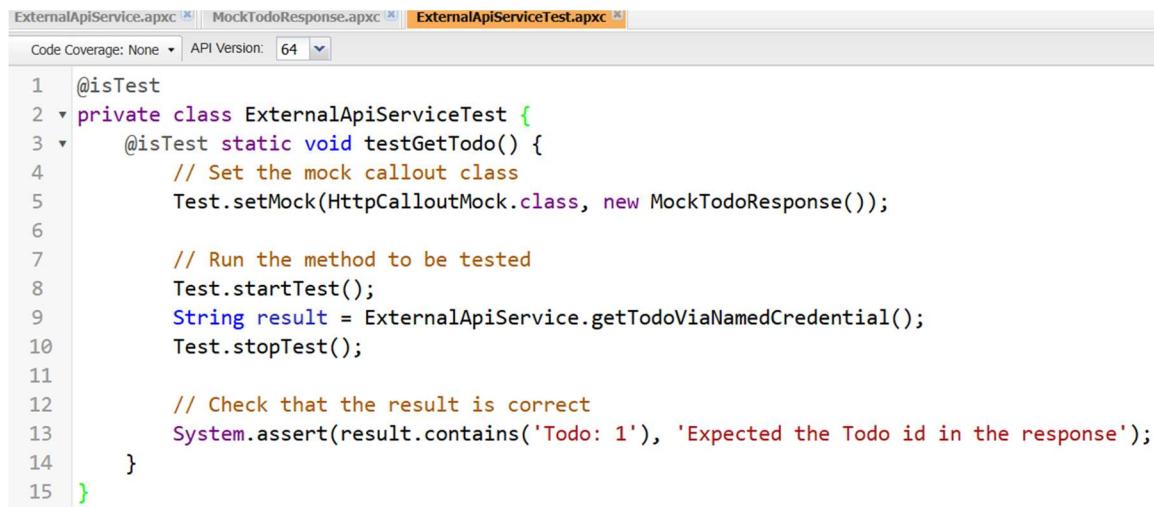
The status bar at the bottom shows tabs for 'Logs', 'Tests', 'Checkpoints', 'Query Editor', 'View State', 'Progress', and 'Problems', with 'Problems' being the active tab.

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



```
File ▾ Edit ▾ Debug ▾ Test ▾ Workspace ▾ Help ▾ < >
External ApiService.apxc MockTodoResponse.apxc
Code Coverage: None API Version: 64
1 public class MockTodoResponse implements HttpCalloutMock {
2     public HttpResponse respond(HTTPRequest req) {
3         HttpResponse res = new HttpResponse();
4         res.setHeader('Content-Type', 'application/json');
5         res.setBody('{"id":1,"title":"Test Todo","completed":false}');
6         res.setStatusCode(200);
7         return res;
8     }
9 }
```

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



```
External ApiService.apxc MockTodoResponse.apxc External ApiServiceTest.apxc
Code Coverage: None API Version: 64
1 @isTest
2 private class External ApiServiceTest {
3     @isTest static void testGetTodo() {
4         // Set the mock callout class
5         Test.setMock(HttpCalloutMock.class, new MockTodoResponse());
6
7         // Run the method to be tested
8         Test.startTest();
9         String result = External ApiService.getTodoViaNamedCredential();
10        Test.stopTest();
11
12        // Check that the result is correct
13        System.assert(result.contains('Todo: 1'), 'Expected the Todo id in the response');
14    }
15 }
```

## Step D.4.

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

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

The screenshot shows the Mule Studio IDE interface. At the top, there are three tabs: 'External ApiService.apxc', 'MockTodoResponse.apxc', and 'External ApiServiceTest.apxc' (which is currently selected). Below the tabs, there are dropdown menus for 'Code Coverage' (set to 'None') and 'API Version' (set to '64').

```

1  @isTest
2  private class External ApiServiceTest {
3  @isTest static void testGetTodo() {
4      // Set the mock callout class
5      Test.setMock(HttpCalloutMock.class, new MockTodoResponse());
6
7      // Run the method to be tested
8      Test.startTest();
9      String result = External ApiService.getTodoViaNamedCredential();
10     Test.stopTest();
11
12     // Check that the result is correct
13     System.assert(result.contains('Todo: 1'), 'Expected the Todo id in the response');
14 }
15 }
```

At the bottom, there is a 'Tests' tab in the navigation bar. The 'Logs' tab is also visible. The 'Tests' tab shows a table with one row:

Status	Test Run	Enqueued Time	Duration
✓	TestRun @ 9:46:59 pm		

## 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": {  
          "content": {  
            "application/json": {  
              "schema": {  
                "type": "object",  
                "properties": {  
                  "id": {  
                    "type": "integer",  
                    "description": "The ID of the todo item."  
                  },  
                  "title": {  
                    "type": "string",  
                    "description": "The title of the todo item."  
                  },  
                  "body": {  
                    "type": "string",  
                    "description": "The body of the todo item."  
                  },  
                  "completed": {  
                    "type": "boolean",  
                    "description": "A flag indicating whether the todo item is completed."  
                  }  
                }  
              }  
            }  
          }  
        }  
      }  
    }  
  }  
}
```

```
"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

The screenshot shows the Salesforce External Services setup page. At the top, there are five progress bars: Total Registrations (1 used of 150), Active Operations (1 used of 1,250), Total Operations (1 used of 10,000), Active Objects (1 used of 1,250), and Total Objects (1 used of 10,000). Below the bars, a message says "1 Items - Sorted by External Service Name - Last refreshed a few seconds ago". A table lists one item:

External Service Name ↑	D	Active Operations	Total Operations	Active Objects	Total Objects	Credentials	Configuration	Actions
JSONPlaceholderService	1	1	1	1	1	JSONPlaceholder_API	Complete	▼

## 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.

The screenshot shows the 'External Data Sources' setup page in Salesforce. At the top, there's a 'SETUP' icon and the title 'External Data Sources'. Below the title, it says 'External Data Source: Northwind\_OData'. A note says 'Connect to another Salesforce org or a third-party database or content system.' with a link '« Back to External Data Sources'. On the right, there are three buttons: 'Edit', 'Validate and Sync', and 'Delete'.

**External Data Source:** Northwind\_OData

**Name:** Northwind\_OData

**Type:** Salesforce Connect: OData 4.0

**Parameters**

- URL:** <https://services.odata.org/V4/Northwind/Northwind.svc/>
- Connection Timeout (Seconds):** 120
- Writable Internal Objects:**
- High Data Volume:**
- Server Driven Pagination:**
- Request Row Counts:**
- Compress Requests:**
- Enable Search:**
- Use Free-Text Search Expressions:**
- Format:** JSON
- Special Compatibility:** None
- Display Server Errors:**
- Eligible for External Change Data Capture:**

**Authentication**

- Certificate:**
- Identity type:** Anonymous
- Authentication Protocol:** No Authentication

**External Objects**

Action	Label	Namespace Prefix	Description	Table Name
Edit   Erase   Validate	Alphabetical_list_of_product	Alphabetical_list_of_products	Alphabetical_list_of_products	Alphabetical_list_of_products
Edit   Erase   Validate	Category	Categories	Categories	Categories
Edit   Erase   Validate	Category_Sales_for_1997	Category_Sales_for_1997	Category_Sales_for_1997	Category_Sales_for_1997
Edit   Erase   Validate	Customer	Customers	Customers	Customers
Edit   Erase   Validate	Employee	Employees	Employees	Employees
Edit   Erase   Validate	Order_Detail	Order_Details	Order_Details	Order_Details
Edit   Erase   Validate	Order	Orders	Orders	Orders
Edit   Erase   Validate	Alphabetical_list_of_product	Alphabetical_list_of_products	Alphabetical_list_of_products	Alphabetical_list_of_products

## 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.

 SETUP  
Tabs

## New Custom Object Tab

**Step 1. Enter the Details**

Choose the custom object for this new custom tab. Fill in other details.

**New Custom Object Tab**

Select an existing custom object or [create a new custom object now](#).

Object	Alphabetical_list_of_product
Tab Style	

(Optional) Choose a Home Page Custom Link to show as a splash page the first time your users click on this tab.

Splash Page Custom Link	--None--
-------------------------	----------

Enter a short description.

Description	<input type="text"/>
-------------	----------------------

**Custom Object Tabs**

[New](#) [What Is This?](#)

Action	Label	Tab Style
Edit   Del	<a href="#">Alphabetical_list_of_products</a>	Airplane
Edit   Del	<a href="#">Donations</a>	Flag
Edit   Del	<a href="#">Donation Summaries</a>	Laptop
Edit   Del	<a href="#">Events</a>	Building Block
Edit   Del	<a href="#">Mentors</a>	Hexagon
Edit   Del	<a href="#">Students</a>	Desk
Edit   Del	<a href="#">Volunteer Events</a>	Books
Edit   Del	<a href="#">Volunteers</a>	Bell
Edit   Del	<a href="#">Volunteer Tasks</a>	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.

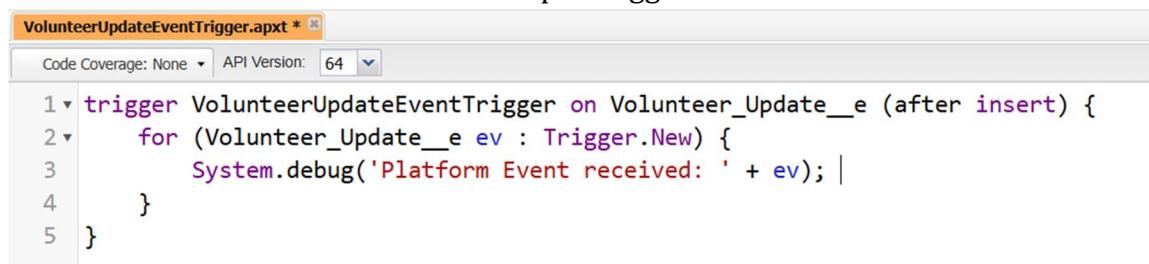
Create a Platform Event: exact clicks:

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
1 trigger VolunteerUpdateEventTrigger on Volunteer_Update_e (after insert) {
2     for (Volunteer_Update_e ev : Trigger.New) {
3         System.debug('Platform Event received: ' + ev);
4     }
5 }
```

## 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**.

```

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

```

User	Application	Operation	Time	Status	Read	Size
Venkata Likith Sai Kovi	Unknown	/services/data/v64.0/to...	9/25/2025, 10:41:13 PM	Success	Unread	3.26 KB

## 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):

```

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

```

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

The screenshot shows the Salesforce developer console. At the top, there is a header bar with tabs for 'Logs', 'Tests', 'Checkpoints', 'Query Editor', 'View State', 'Progress', and 'Problems'. The 'Logs' tab is selected. Below the header, there is a code editor window containing the following trigger code:

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

Below the code editor is a table titled 'Logs' showing three entries:

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.Direct...	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.

The screenshot shows the REST Explorer interface in the Salesforce Workbench. The top navigation bar includes links for workbench, info, queries, data, migration, and utilities. A sub-header indicates the user is VENKATA LIKITH SAJ KOVI AT HELPING HANDS FOUNDATION ON API 62.0. Below this, a red banner says "Try the Salesforce APIs for Postman." The main area is titled "Choose an HTTP method to perform on the REST API service URI below:" with a radio button selected for "GET". There are also buttons for POST, PUT, PATCH, DELETE, HEAD, Headers, Reset, and Up. A text input field contains the URL "/services/data/v62.0". To the right of the input field is an "Execute" button. Below the input field, there are links for "Expand All", "Collapse All", and "Show Raw Response". A large list of API endpoints is displayed as a tree structure, starting with metadata, eclair, folders, jsonform, appMenu, iot, analytics, smartdatadiscovery, composite, parameterizedSearch, fingerprint, scheduling, domino, serviceTemplates, recent, dedupe, query, ai, consent, digitalwallet, compactLayouts, knowledgeManagement, actions, support, tooling, prechatForms, contactTrading, chatter, payments, tabs, quickActions, queryAll, commerce, wave, search, identity, theme, nouns, event, connect, licensing, limits, process, jobs, match, localizedValue, mobile, emailConnect, tokenizer, async, externalServices, and subjects. At the bottom of the page, a footer note states "Requested in 0.079 sec" and "Workbench 62.0.0".