

Phase 7: Integration & External Access

1. Named Credentials

Purpose

- Securely store authentication information for external APIs.
- Simplifies API callouts by avoiding hardcoding credentials in Apex code.
- Manages authentication protocols such as OAuth 2.0, Password Authentication, or Named Principal.
- Enhances security by controlling access centrally.

Key Components

- **Label & Name:** User-friendly label and unique API name for reference in Apex.
- **URL:** Base URL of the external service.
- **Identity Type:** Defines who is making the callout:
 - Named Principal – All users share the same authentication.
 - Per User – Each Salesforce user uses their own credentials.
- **Authentication Protocol:** Type of authentication used for the API:
 - Password Authentication
 - OAuth 2.0
 - AWS Signature
 - JWT Bearer Token
- **External Credential (Optional):** Can link to external credentials stored securely.
- **Callout Enabled:** Must be checked to allow callouts using this credential.
- **Usage in Apex:**

```
HttpRequest req = new HttpRequest();  
req.setEndpoint('callout:WeatherAPI_NC/v1/current.json?q=London');  
req.setMethod('GET');  
HttpResponse res = new Http().send(req);  
System.debug(res.getBody());
```

The screenshot displays the Salesforce Setup interface. On the left, the 'Setup' menu is visible with 'Security' expanded and 'Named Credentials' selected. The main content area shows the configuration for a Named Credential named 'WeatherAPI_NC'. The configuration includes fields for Label, Name, and URL, all set to 'WeatherAPI_NC'. The URL is 'https://api.weatherapi.com'. The 'Enabled for Callouts' toggle is turned on. Under the 'Authentication' section, 'External Credential' is set to 'WeatherAPI_EC' and 'Client Certificate' is empty. Under the 'Callout Options' section, 'Generate Authorization Header' is checked and 'Allow Formulas in HTTP Header' is unchecked. 'Edit' and 'Delete' buttons are located at the top right of the configuration panel.

Field	Value
Label	WeatherAPI_NC
Name	WeatherAPI_NC
URL	https://api.weatherapi.com
Enabled for Callouts	Yes
Authentication Protocol	External Credential
External Credential	WeatherAPI_EC
Client Certificate	
Generate Authorization Header	Yes
Allow Formulas in HTTP Header	No

2. External Services

Purpose

- Connect Salesforce to external APIs **declaratively**.
- Automatically generate **actions for Flows**.
- Simplifies integration for admins and non-developers.
- Supports both **REST and SOAP APIs**.

Key Features

- Declarative API integration with no Apex required.
- Uses **Swagger/OpenAPI schema** to define API structure.
- Actions are **automatically created** in Salesforce.
- Works with **Named Credentials** for secure authentication.
- Can be used in **Screen Flows, Scheduled Flows, and Record-Triggered Flows**.
- Supports reusability and governance of API calls.

Steps:

- Setup → External Services → New External Service
- Provide Name, Description, Service Schema (Swagger/OpenAPI URL)
- Select Named Credential
- Salesforce auto-generates actions usable in Flow

The screenshot shows the Salesforce Setup interface. The left sidebar has a search bar with "named" entered, and the "Named Credentials" link is highlighted under the "Security" section. The main content area displays the configuration for the "WeatherAPI_EC" Named Credential. It shows the Label as "WeatherAPI_EC", the Name as "WeatherAPI_EC", and the Authentication Protocol as "Custom". Below this is a "Managed Package Access" section. The "Related Named Credentials" section shows a table with one entry: "WeatherAPI_NC" with the URL "https://api.weatherapi.com". The "Principals" section shows a table with one entry: "WeatherAPI_Principal" with the authentication type "Unknown".

Label	Name
WeatherAPI_EC	WeatherAPI_EC

Authentication Protocol: Custom

Managed Package Access

Created By Namespace ⓘ

Label	Name	URL
WeatherAPI_NC	WeatherAPI_NC	https://api.weatherapi.com

S...	Parameter Name	A...	Aut...	Actions
1	WeatherAPI_Principal	1	Unknown	⌵

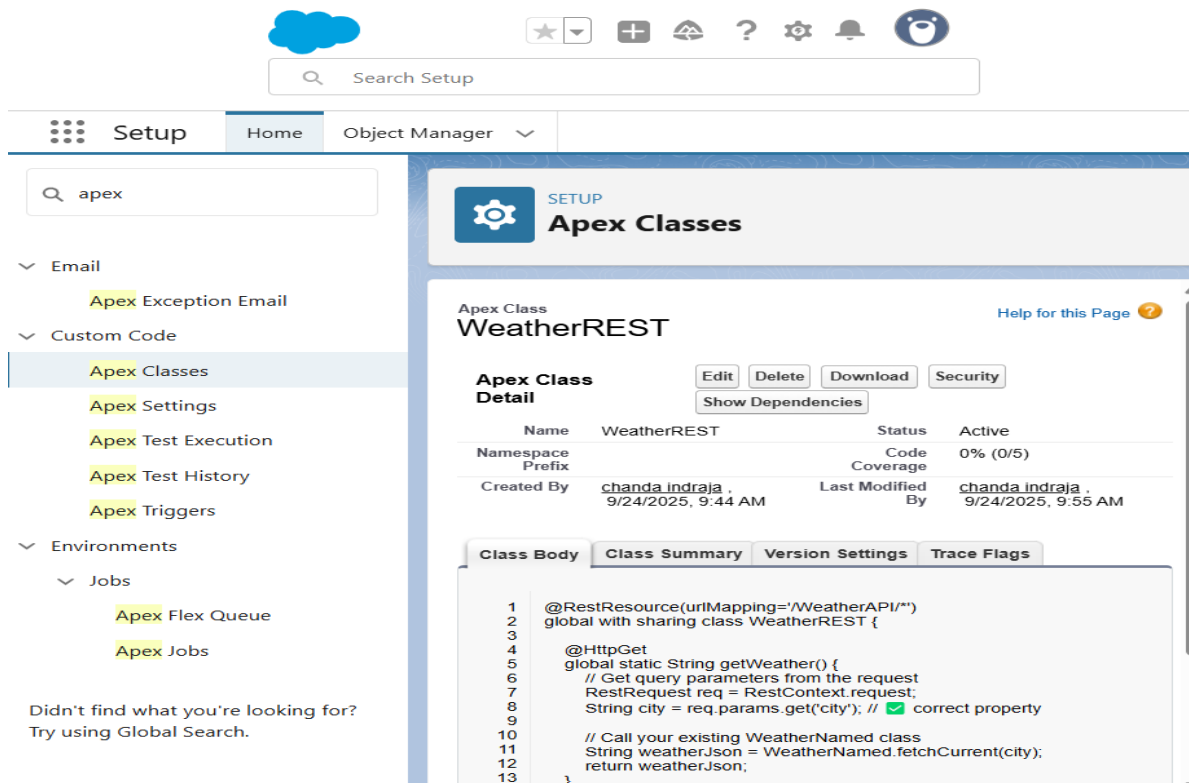
3. Web Services (REST/SOAP)

Purpose

- Expose Salesforce data to external systems.
- Consume external web services within Salesforce.
- Enable integration with external applications using standard protocols.
- Supports both **REST (lightweight)** and **SOAP (structured)** protocols.

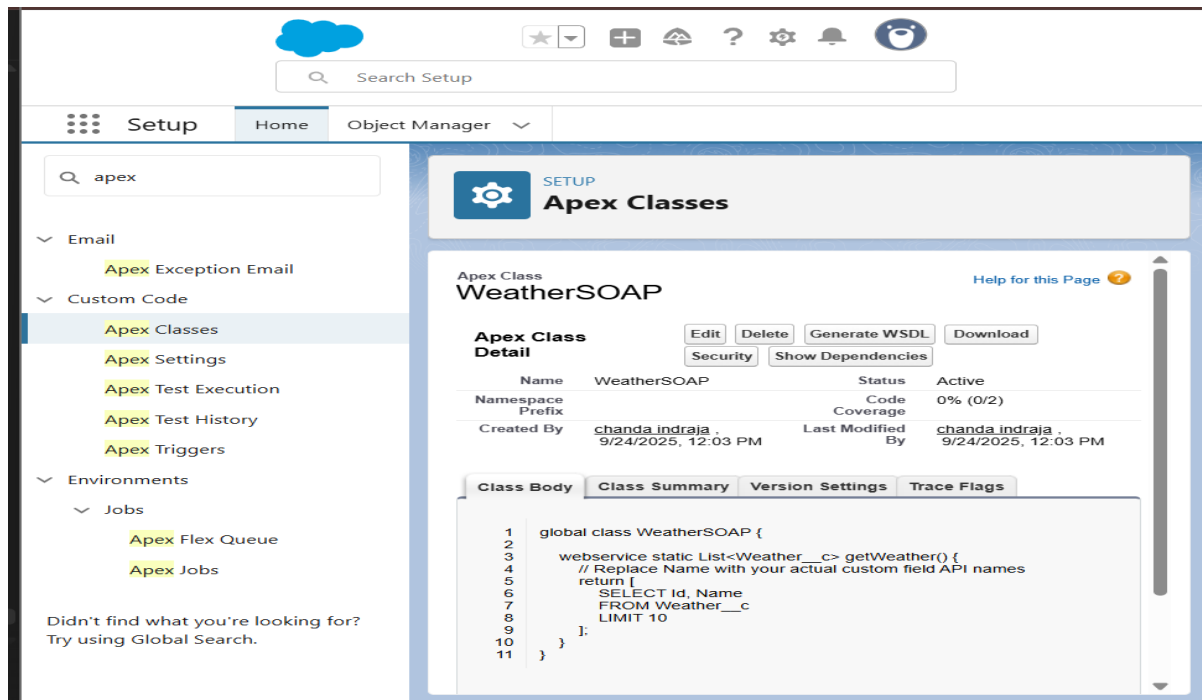
Key Features

- **REST Services**
 - Lightweight, stateless, uses HTTP methods (GET, POST, PUT, DELETE).
 - Supports JSON and XML formats.
 - Easier to use for modern web and mobile apps.
- **SOAP Services**
 - Standardized protocol using XML.
 - Strongly typed operations, supports WSDL.
 - Preferred for enterprise-level integrations.



The screenshot displays the Salesforce Setup interface. The left sidebar shows the navigation menu with 'Setup' selected. Under 'Custom Code', 'Apex Classes' is highlighted. The main content area shows the 'Apex Classes' page for a class named 'WeatherREST'. The class is active, created by 'chanda.indraja' on 9/24/2025 at 9:44 AM, and has 0% code coverage. The 'Class Body' tab is selected, showing the following code:

```
1 @RestResource(urlMapping='/WeatherAPI/*')
2 global with sharing class WeatherREST {
3
4     @HttpGet
5     global static String getWeather() {
6         // Get query parameters from the request
7         RestRequest req = RestContext.request;
8         String city = req.params.get('city'); // correct property
9
10        // Call your existing WeatherNamed class
11        String weatherJson = WeatherNamed.fetchCurrent(city);
12        return weatherJson;
13    }
14 }
```



4. Callouts

Purpose

- Enable Salesforce to **communicate with external APIs** or web services.
- Retrieve or send data from/to external systems.
- Used in integrations, third-party services, or external data synchronization.

Key Features

- Supports **HTTP methods**: GET, POST, PUT, DELETE.
- Can use **Named Credentials** for secure authentication.
- Can also use **Remote Site Settings** for older setups.
- Supports **synchronous** (immediate response) and **asynchronous** (future methods, batch Apex) callouts.
- Can handle **REST and SOAP APIs**.
- Use **Named Credentials** instead of Remote Site Settings for security.
- Always handle exceptions with `try-catch`.

5. Platform Events in Salesforce

Purpose

- Enable **event-driven architecture** inside Salesforce.
- Send and receive custom notifications between Salesforce processes or external systems.
- Useful for real-time updates and decoupled integrations.

Key Features

- **Custom Event Objects** with fields like any other object.
- Events are **published** and **subscribed** asynchronously.
- Supports **Publish-Subscribe model**:
 - **Publisher**: Creates and publishes events.
 - **Subscriber**: Triggers, Flows, or external systems that react to events.
- Can be used in **Apex, Flows, and External Systems (Streaming API)**.
- Durable events: Can retain events for up to **24 hours**.

The screenshot shows the Salesforce Setup interface for Platform Events. The left sidebar contains a navigation menu with options like MuleSoft, Anypoint Platform Setup, Einstein, Einstein Platform, Einstein Bots, Einstein AI, Custom Code, Platform Cache, Integrations, Platform Events (selected), Security, Platform Encryption, Encryption Settings, and Key Management. The main content area is titled 'Platform Events' and includes a 'Platform Event Summary' section with fields for Singular Label, Plural Label, Object Name, API Name, Event Type, Publish Behavior, and Created By. Below this is a 'Standard Fields' table with columns for Action, Field Label, Field Name, Data Type, Controlling Field, and Indexed. The 'Custom Fields & Relationships' section includes a 'New' button and a table with columns for Action, Field Label, API Name, Data Type, Indexed, Controlling Field, and Modified By. The 'Triggers' section also has a 'New' button.

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		

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit / Del	City	City__c	Text(3)			chanda.indraja, 9/24/2025, 10:39 AM
Edit / Del	Condition	Condition__c	Text(18)			chanda.indraja, 9/24/2025, 10:40 AM
Edit / Del	Temperature	Temperature__c	Number(18, 0)			chanda.indraja, 9/24/2025, 10:39 AM

The screenshot shows the Salesforce Execution Log. The log entry is for a debug message: '23:11:50:044 USER_DEBUG [8] DEBUG Event published: true'. A dialog box titled 'Log line #35 details' is open, displaying the same log entry: '23:11:50:044 USER_DEBUG [8] DEBUG Event published: true'. The dialog box has an 'OK' button.

6. Change Data Capture (CDC)

Purpose

- Track **create, update, delete, and undelete events** on Salesforce records in real time.
- Enable **asynchronous integration** with internal or external systems.
- Helps in **synchronizing Salesforce data** with other systems.
- Works with **any standard or custom object**, including `Weather__c`.

Key Features

- Real-time tracking of **record changes**.
- Automatically generates **Change Event objects** for tracked objects.
- Supports **Apex Triggers, Flows, and External Subscribers**.
- Captures **all field changes** automatically.
- Can be integrated with **Streaming API** or external systems.
- Works with **standard objects** and **custom objects**.

The screenshot displays the Salesforce Setup interface. The left sidebar shows the navigation menu with 'Setup' selected. Under 'Custom Code', 'Apex Triggers' is highlighted. The main content area shows the 'Apex Triggers' configuration for 'WeatherChangeTrigger'. The trigger is active and has 0% code coverage. The trigger code is displayed in the 'Apex Trigger' tab, showing a trigger on 'Weather__ChangeEvent' (after insert) that logs record IDs and field values (City, Temperature, Condition) using System.debug. The code is as follows:

```
1 trigger WeatherChangeTrigger on Weather__ChangeEvent (after insert) {
2   for (Weather__ChangeEvent event : Trigger.New) {
3
4     // Access changed record IDs
5     List<Id> changedRecordIds = event.ChangeEventHeader.getRecordIds();
6     System.debug('Record changed: ' + changedRecordIds);
7
8     // Access your custom fields
9     System.debug('City: ' + event.get('City__c'));
10    System.debug('Temperature: ' + event.get('Temperature__c'));
11    System.debug('Condition: ' + event.get('Condition__c'));
12
13    // Optional: publish Platform Event
14    Weather__IndateEvent e = new Weather__IndateEvent(e.get('City__c'), e.get('Temperature__c'), e.get('Condition__c'));
```

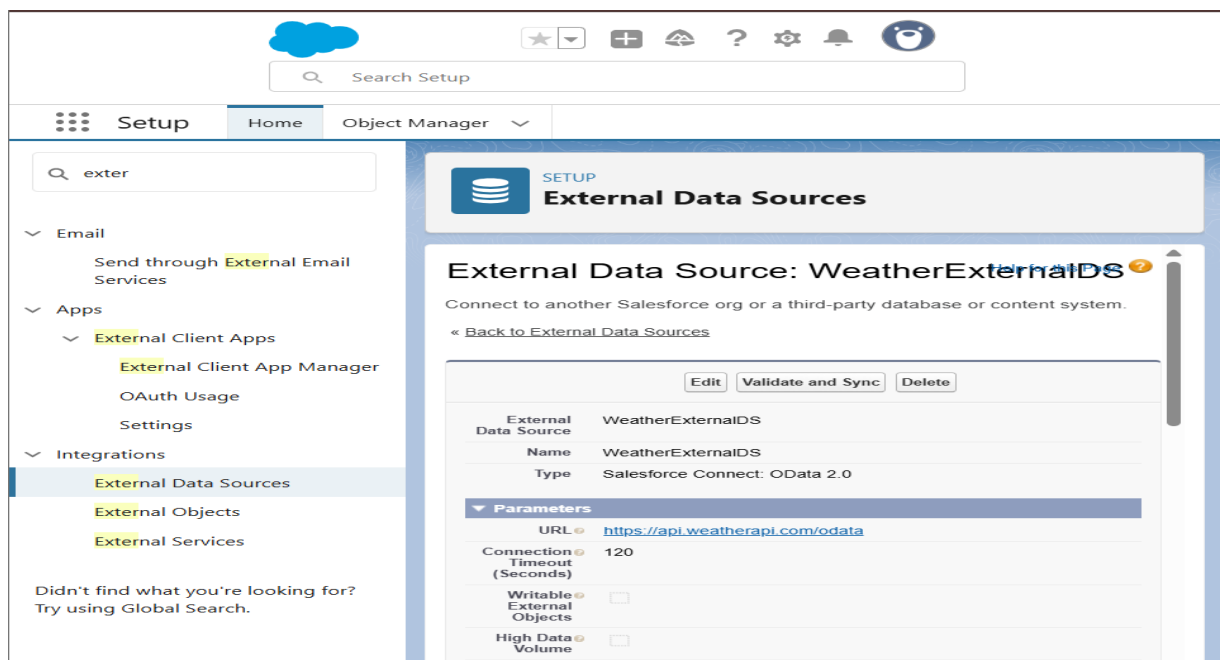
7. Salesforce Connect

Purpose

- Integrate external data sources directly into Salesforce without storing the data in Salesforce.
- Enables **real-time access** to external objects via **OData, REST, or custom adapters**.
- Ideal for **large datasets** or external systems where replication is not practical.

Key Features

- Supports **OData 2.0 and 4.0** protocols.
- Creates **External Objects** in Salesforce representing external tables.
- Provides **read, create, update, and delete** access depending on external system capabilities.
- Works with **Lightning, Visualforce, and Apex**.
- Can combine **external objects with standard Salesforce objects** using **relationship**



8. API Limits

Purpose

- Salesforce enforces **limits on API calls** to protect system resources.
- Helps track and manage usage for integrations and automation.

Key Points

- **API Request Limits** vary by **edition, license type, and user**.

- Types of API calls:
 - REST API
 - SOAP API
 - Bulk API
 - Streaming API
 - Tooling API
- Monitor usage via: **Setup** → **System Overview** → **API Usage**, or **REST Limits resource**.
- **Exceeded limits** return `REQUEST_LIMIT_EXCEEDED` errors.

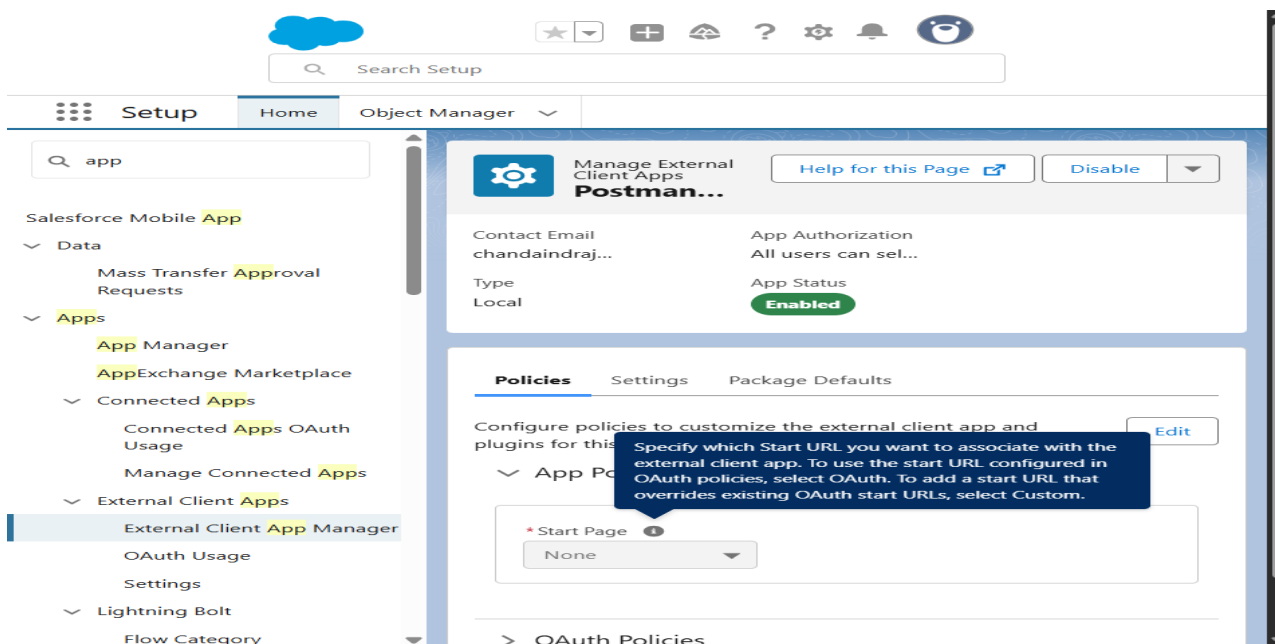
9. OAuth & Authentication

Purpose

- Secure integration with Salesforce or external systems.
- Provides **token-based authentication** without storing passwords.
- Supports **single sign-on (SSO)** and **API access**.

Key Features

- Supports **OAuth 2.0 flows**:
 - Authorization Code
 - Username-Password
 - JWT Bearer
 - Refresh Token
- Provides **access token** for API calls.
- Tokens can expire; refresh tokens allow re-authentication without user intervention.



10.Remote Site Settings

Purpose

- Allows Salesforce to **perform HTTP or HTTPS callouts** to external systems.
- Required when using **Apex HTTP callouts** to URLs that are **not secured via Named Credentials**.
- Ensures **security by allowing only trusted URLs** to be accessed from Salesforce.

Key Features

- Lets Salesforce **call external web services or APIs**.
- Provides a **centralized place** to manage external URLs.
- Required for **REST/SOAP API integration**, webhooks, or external system data fetch.
- Works with **all Apex HTTP callouts**, including GET, POST, PUT, DELETE.
- Supports **HTTPS** endpoints for secure communication.
- Can be used in combination with **Named Credentials** for advanced authentication.

The screenshot displays the Salesforce user interface for the 'Remote Site Settings' page. The top navigation bar includes the Salesforce logo, a search bar labeled 'Search Setup', and various utility icons. The left sidebar shows the 'Setup' menu with 'Remote Site Settings' highlighted under the 'Security' section. The main content area is titled 'Remote Site Settings' and contains a 'Remote Site Details' section for a specific site named 'WeatherAPI_RS'. This section includes fields for the Remote Site Name, Remote Site URL, a checkbox for 'Disable Protocol Security', a description, an 'Active' checkbox, and the 'Created By' field. Action buttons for 'Edit', 'Delete', and 'Clone' are provided for the site details.

Remote Site Detail		Edit	Delete	Clone
Remote Site Name	WeatherAPI_RS	Modified By chanda.indraja, 9/24/2025, 11:15 AM		
Remote Site URL	https://api.weatherapi.com			
Disable Protocol Security	<input type="checkbox"/>			
Description				
Active	<input checked="" type="checkbox"/>			
Created By	chanda.indraja, 9/24/2025, 11:15 AM	Edit	Delete	Clone