

Introduction.....	4
How to.....	5
Prerequisite.....	6
What is Integration.....	6
Types of Authentication.....	9
No Authentication & API Key Authentication.....	11
No Authentication.....	11
API Key Authentication.....	14
OpenCage Data API.....	14
Basic Authentication.....	19
Introduction to Connected Application.....	25
How Salesforce manage Access Token Expiration Time.....	26
Introduction to OAuth2.0.....	28
OAuth2.0 Authorization Code Flow (Web Server Flow).....	31
oAuth 2.0 Step1 - Prepare authorization code URL & Get Authorization Code....	32
oAuth 2.0 Step2 - Prepare toke URL & get the access token.....	35
Test the Access Token.....	37
Assignments.....	38
OAuth 2.0 User Agent flow (Implicit Flow).....	49
OAuth 2.0 Authorization Code Flow with PKCE - Proof Key for Code Exchange.....	52
OAuth 2.0 Resource Owner Flow (Username & Password) - Not recommended.....	63
OAuth 2.0 Client Credentials Flow for Server-to-Server Integration.....	65
Visual Flow walkthrough.....	68
Step2 - Get Access Token.....	68
OAuth 2.0 Device Flow for IoT Integration - Using Salesforce for Demo.....	69
Configure the Connected Application.....	69
STEP#1 - Device Requests Authorization.....	70
STEP#2 - VERIFY THE DIVIDE CODE (Done by User).....	71
Step#3 - Get the Access Token.....	73

OAuth 2.0 JWT Bearer Flow for Server-to-Server Integration.....	74
Useful JSON Class methods.....	82
What is Apex REST.....	85
Introduction to Apex Rest.....	85
Methods that can be used with Apex Rest.....	86
RestContext Class.....	87
RestContext Class.....	87
RestRequest Class.....	87
Authentication Mechanism.....	87
A Complete Scenario.....	90
Write your First Apex Rest API & Test using Workbench.....	92
Get Method Using Apex Rest @HttpGet.....	92
Test Your API using the Postman API Testing Tool.....	92
Post Method Using Apex Rest @HTTPPost.....	93
Delete Method using Apex Rest @HttpDelete.....	93
Patch Method using Apex Rest @HTTPPatch.....	94
Assignments.....	94
Consume 3rd Party APIs.....	101
What are all the 3rd Party APIs that we will work on in this course.....	101
OpenCage Geocoder.....	102
Implement Geocoder API using Apex.....	102
Assignment.....	102
Introduction to Freshdesk and test using Postman.....	102
Integrate Freshdesk with Salesforce - Part 1.....	103
Integrate Freshdesk with Salesforce - Part 2.....	103
Assignment.....	103
Introduction to QuickBooks.....	105
→ Authenticate Quickbooks from Salesforce.....	107
Make API Callouts to QuickBooks.....	108
Implement the Refresh Token method.....	109
QB_Customers Apex Class.....	123

Assignments.....	127
Prerequisites -	127
#1.....	127
#1.1 -	128
#2.....	130
Google Calendar Integration.....	133
Create Connected APP.....	133
Login to Google Console.....	133
Create a New Project.....	134
Create Credentials in the Google Console.....	139
Test Google API using Postman.....	142
Create Event From Salesforce to Google Calendar.....	143
Assignments.....	143
Introduction to LinkedIn.....	145
Implement LinkedIn Integration with Salesforce.....	146
Assignments.....	146
More Apis to Work upon.....	148
Named Credentials in Salesforce.....	148
Create Permission Set.....	149
Provide “User External Credentials” Object Access to Permission Set.....	150
Implement Named credentials in Salesforce.....	151
Create External Credentials.....	151
Create Principals related to External Credentials.....	152
Provide Access to Principals at the Permission Set Level.....	153
Create Named Credentials.....	154
Configure the External Credentials at the User Level (Optional).....	156
Use Named Credentials in Apex Class.....	158
Authentication (Auth.) Provider in Salesforce.....	159
Google Email as an Authentication Provider.....	160
Configure the External Credentials at the User Level (Optional).....	164
Use Auth. Provider to Login into Salesforce (Social Sign-On).....	169

Assignment.....	170
Salesforce JWT Authentication using Apex Class.....	171
JWT Structure.....	171
Prerequisites.....	172
Steps involved for JWT.....	172
Create a JWT Header.....	172
Create the JWT Claims.....	173
Get the Access Token using Custom Apex Class and a Private Key.....	178
Convert .CRT to Salesforce Keystore JKS file.....	181
Upload the .jks file to Salesforce.....	181
Salesforce JWT Authentication using Named Credentials.....	183
Introduction to Streaming API.....	185
How does push technology work?.....	185
Examples of push technology.....	186
Publish-Subscribe Model.....	186
Important Terms.....	187
Introduction to Platform Events.....	187
Types of Platform Events.....	188
What do we have in this course?.....	189
Create Your 1st Platform Event & Publish it using Apex.....	189
Publishing behaviour of platform event.....	191
Publish the event using Apex Class.....	192
Publish the Event Using Lighting Flow Builder.....	193
Use Case.....	193
Publish Event using Salesforce API.....	193
Use UUID to publish the Platform Event.....	196
Track the Success & Failure Platform Event.....	196
Subscribe to Standard Platform Event using Web Component (BatchApexErrorEvent).....	199
Subscribe to the Platform Event.....	199
Unsubscribe the Platform Event.....	200

Handle the Error Message if there are any while subscribing to the event.....	200
Assignment.....	202
Subscribe to Platform Event using Node.js.....	202
Introduction to Change Data Capture.....	204
Enable Change Data Capture.....	205
Prepare Subscription Channel.....	205
Structure of Change Data Capture.....	206
Apex Trigger Code.....	207
Assignment.....	208
Outbound Message.....	208
Example.....	209
Structure of Notification Acknowledgement.....	209
Outbound in Action.....	210
Apex Code to generate the Random Contact Records.....	210
Apex Code to generate Random Opportunities.....	212
METADATA API IN SALESFORCE.....	214
List Metadata Using Metadata API.....	214
Create Object using API.....	217
Create Fields using API.....	217
Create Lookup Fields using Metadata API.....	218
Create a Picklist Field using Metadata API.....	219
Delete Custom Metadata Record using metadata API.....	221
Complete Apex Class.....	221
Bulk API.....	228
Key Features of Bulk API:.....	228
Use Cases for Bulk API:.....	228
Example Scenarios:.....	228
Integration Interview Questions.....	232
General Integration Questions.....	232
Specific Integration Scenarios.....	232
Tools and Middleware.....	232

Salesforce-Specific Tools.....	232
Authentication and Security.....	233
Troubleshooting and Monitoring.....	233
Advanced Topics.....	233
Integration Use Cases.....	233
Apex REST.....	234
Platform Events.....	234
Unit Testing for Integration.....	234
Auth Providers and Named Credentials.....	235
Concepts:.....	235
Configuration:.....	235
Integration:.....	236
Security:.....	236
Troubleshooting:.....	236
Bonus:.....	236
Bulk API.....	236
Metadata API.....	236
Integration with Other APIs.....	237
Salesforce Integration Fundamentals.....	237
General Concepts.....	237
Integration Techniques.....	237
Security Considerations.....	237
Troubleshooting.....	237
Scenario-Based Questions.....	238

Introduction

- Introduction to the Course
- Introduction to Author

How to

- How to get help
 - Q & A Section (preferred)
- How to succeed with the course
 - Make your Notes
 - Use Your laptops/desktops - Learn & Apply
 - Maintain your own Code Repo -
Assignment/Homework/Interview questions
 - Ask Questions
 - Be curious about the things
 - Track Your Progress
 - Compete with yourself
 - Persistence.
 - Take a break and come back
 - Make sure you come back
- How to get the materials used in this course
- How to set required objects
 - Provide the link to the GitHub Repo with the Deploy Button
- Useful Resources for the Integration
 - An article with all the links
- Leave a useful review
 - with proper comment

Prerequisite

- Salesforce Development
- Salesforce Admin
- SOQL Queries
- Data Model
- Your Brand New Salesforce Developer Org.

<https://www.postman.com/downloads/>

Pre-Request Script

<https://learning.postman.com/docs/writing-scripts/intro-to-scripts/#execution-order-of-scripts>

Chai.js for multiple asserts

<https://www.chaijs.com/api/bdd/>

What is Integration

Application Programming Interface is the full form of API

API is used to communicate with two or more systems and exchange data between each other.

Client → API → Server (Authentic, Authorization, Response)

API works on Client Service Architecture where there will always be min one client and a minimum of a Server.

Let's take an example of a website here

<https://www.bbc.com/news>

B B C Sign in Home News Sport Reel Worklife Travel Future Culture ... Q Search BBC

NEWS

Home | War in Ukraine | Coronavirus | Climate | Video | World | Asia | UK | Business | Tech | Science | Entertainment & Arts | Health | World News TV | More ▾

Kim Jong Un shows off missiles to Russia's Shoigu

The tour comes amid accusations that Pyongyang is supplying Russia with arms for the Ukraine war.

⌚ 7h | Asia

- How to negotiate with world's most secretive country
- Preparing N Korea's traumatised defectors for new lives
- Inside North Korea: "We are stuck, waiting to die"



Sinéad O'Connor's death not treated as suspicious

Police say she was found "unresponsive" and "pronounced dead at the scene" in London at 11:18 BST.

⌚ 9m | Entertainment & Arts



Taiwan practises repelling a Chinese invasion

Taiwan has learned from Russia's invasion of Ukraine and is moving to fortify vulnerable points.

⌚ 4h | Asia



How a diplomat's downfall leaves China red-faced

As mystery swirls over Qin Gang's removal, questions have arisen over what this means for Chinese diplomacy.

⌚ 7h | China



Aung San Suu Kyi moved out of jail to house arrest

Myanmar's civilian leader is serving a 33-year jail sentence after being ousted in a military coup.

⌚ 4h | Asia



Jewish group upset at Holocaust scenes in India film

The Simon Wiesenthal Center has asked Amazon Prime to remove Bollywood film Bawaal from its platform.

⌚ 1h | India

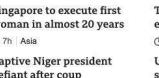


Singapore to execute first woman in almost 20 years

⌚ 7h | Asia

Captive Niger president defiant after coup

⌚ 1h | Africa

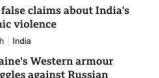


The false claims about India's ethnic violence

⌚ 15h | India

Ukraine's Western armour struggles against Russian defences

⌚ 4h | Europe



Relegated Leicester City delight fans on Asia tour

⌚ 10h | Business

Barbie toy maker looks to more movies as sales fall

⌚ 8h | Business



Can India become a global chip powerhouse?

⌚ 15h | India



LIVE Is population growth fuelling climate change? Your questions answered

⌚ 10h | World

Women's World Cup

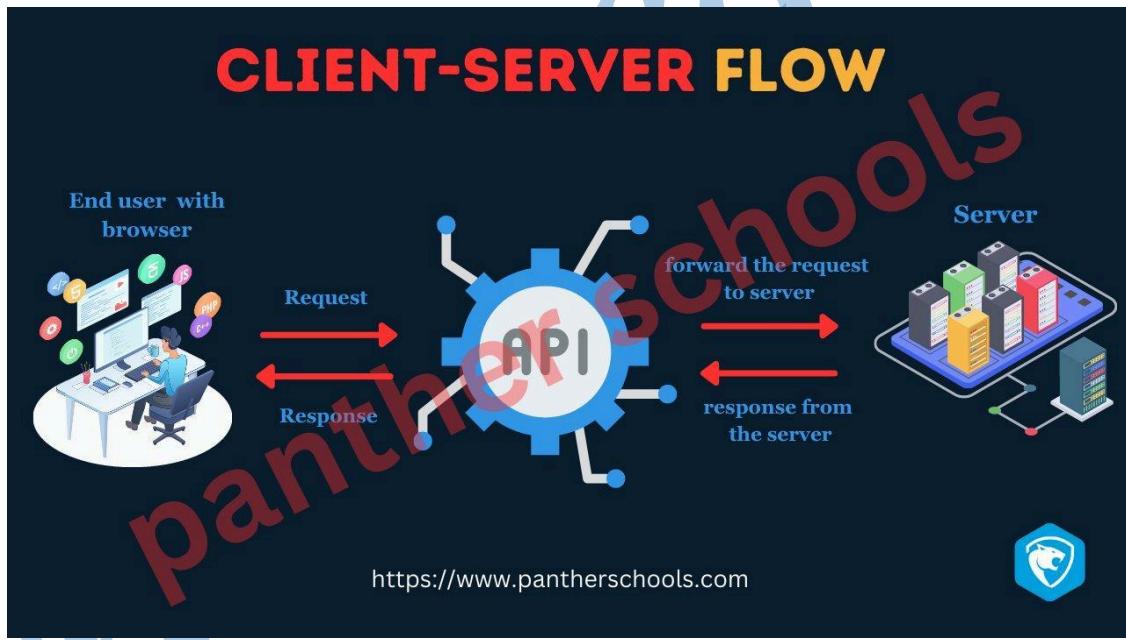
Let's discuss an API with the restaurant example

Image taken from <https://www.apispreadsheets.com/>



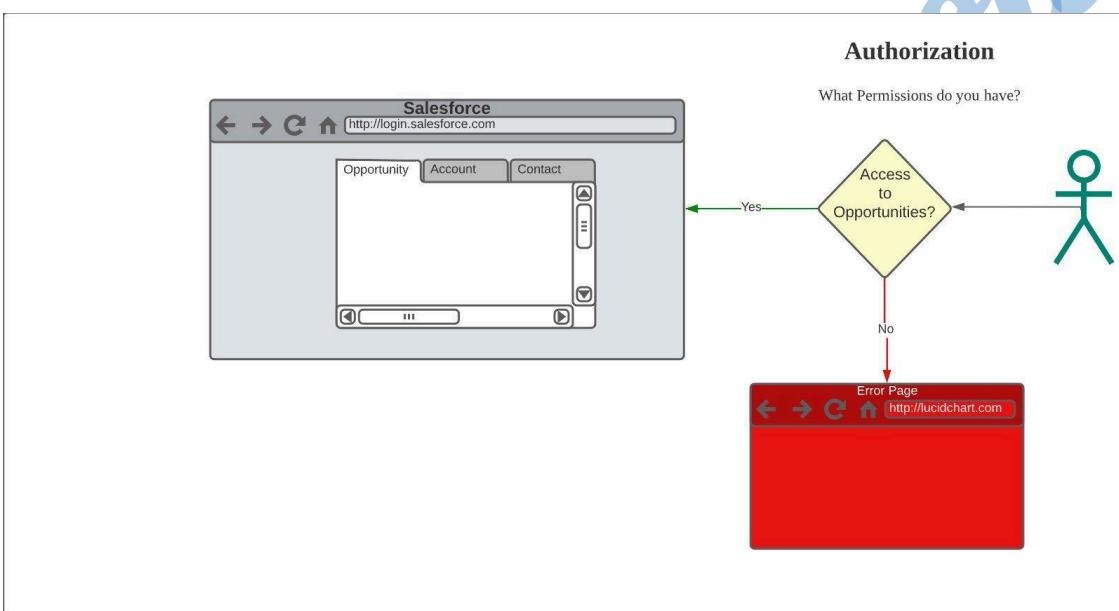
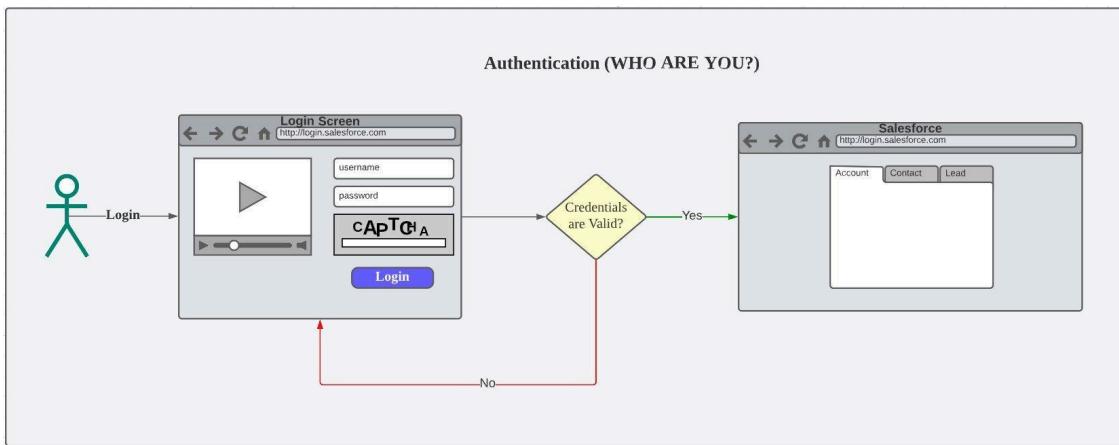
Mobile Application
Zomato
Swiggy
UI (Front End)
Items (Restaurant List)
Food Options
Images
Offer

Restaurants



Types of Authentication

- Understanding the concept of Authentication & Authorization
 - Authentication [*Who are you?*]
 - Authorization [*What Permissions have been given to you?*]



- Type of Authentication in Salesforce
 - No Authentication (Not Related to Salesforce)
 - Basic Authentication
 - Freshdesk
 - API Key-based Authentication
 - OpenCage Geocoder
 - Exchange Rate
 - OAuth 1.0 [3-Legged Authentication]
 - Twitter
 - QuickBooks [Now Using OAuth 2.0]
 - Not Included in the Course
 - OAuth 2.0 - Industry Wide used

- Web Server Flow
 - User Agent Flow
 - Username & Password Flow [Deprecated For Salesforce]
 - Not Included
 - Refresh Token Flow
 - JWT Bearer Flow for Server-to-Server Integration
 - Client Credentials Flow for Server-to-Server Integration
 - Device Flow for IoT Integration
 - SAML Bearer Assertion Flow for Previously Authorised Apps
 - Not Included
 - PKCE?
- o OAuth 2.1
- Future
 - <https://oauth.net/2.1/>

No Authentication & API Key Authentication

No Authentication

If you want to access a resource that is available to the public and does not require any kind of authentication then you can use No Authentication in Salesforce.

```
public with sharing class PS_CurrencyAPIService {  
  
    public static void getExchangeRates(){  
        /* Callouts Only */  
        /* Step1 - Prepare the Request */  
        HttpRequest httpReq = new HttpRequest();  
        /* Step1.1 - Send the Endpoint */
```

```
httpReq.setEndPoint('https://open.er-api.com/v6/latest/USD');

/* Step 1.2 - Set the Headers */
httpReq.setHeader('Content-Type', 'application/json');
httpReq.setHeader('Accept', 'application/json'); // JSON, XML, Text,
HTML

/* Step1.3 - Set the Method */
httpReq.setMethod('GET'); // GET, POST, PUT, PATCH, DELETE

/* Step2 - Send the Request */
Http htt = new Http();
try {
    HttpResponse httpRes = htt.send(httpReq);
    /* Step3 - Print the Information */
    /*
        getStatus(), getStatusCode()
        getBody() getXmlStreamReader() --> SOAP, getBodyAsBlob()
    */
    String responseBody = httpRes.getBody();
    Integer statusCode = httpRes.getStatusCode();
    String status = httpRes.getStatus();
    System.System.debug('The Response from Currency API - '+
status );
    System.System.debug('The Response from Currency API - '+
responseBody );
    System.System.debug('The Response from Currency API - '+
statusCode );
}catch(System.CalloutException calloutEx){
    System.debug('System.CalloutException .... '+
calloutEx.getStackTraceString());
    if(String.valueOf(calloutEx).startsWith('System.CalloutException:'))
```

```

        Unauthorised endpoint')){

            // Remote Site missing Error -
            System.debug(' CalloutException ');

        }

    }catch(System.Exception ex){

        System.debug('Exception Executed ... '+ ex.getStackTrace());
        if(String.valueOf(ex).startsWith('System.CalloutException:

        Unauthorised endpoint')){

            // Remote Site missing Error -
            System.debug(' Executed ');

        }

    }

}

```

Examples

- <https://www.ipify.org/>
 - API Document
- Exchange Rates API - <https://open.er-api.com/v6/latest/USD>
- Public APIs - <https://api.publicapis.org/entries>
- Coindesk - <https://api.coindesk.com/v1/bpi/currentprice.json>
- Dogs Api - <https://dog.ceo/api/breeds/image/random>
- Random Joke API - https://official-joke-api.appspot.com/random_joke
- Random User API - <https://randomuser.me/api/>

Note: - You can get the list of APIs here - <https://apipheny.io/free-api/>

REST -

JSON, XML, Text, Raw, etc

SOAP API

Server to Server [Metadata manipulation]

XML

HttpRequest - To prepare the Request with all the information

HTTP - send(HttpRequest req) →

HttpResponse - Holds the response that is being sent by the Server

API Key Authentication

API Key-based authentication is the easiest way of doing the authentication for any API that supports this method to authenticate the API request.

- Most of the time we need to pass the API Key in the URL parameters with the endpoint itself.
- However, sometimes we need to pass the API Key in the header as well.

Examples

- <https://exchangeratesapi.io/> API takes the Key as part of the header
- Exchange Rate API
 - https://v6.exchangerate-api.com/v6/YOUR_API_KEY/latest/USD
 - o <https://v6.exchangerate-api.com/v6/ef48df7536db9dd806a40181/latest/USD>

OpenCage Data API

<https://api.opencagedata.com/geocode/v1/json?key=1badb7d9b4ad4f889c54d05f6bd0b3d7>

URL Construct - https://www.example.com/uri-endpoint?key=API_KEY

```
public with sharing class OpenCageGeocoderService {

    public static void reverseGeoCoding(String accountId){ // argument
        /* Step0 - Get The Latitude and Longitude for given account */
        Account accRecord = [SELECT Id,
            Location__Latitude__s, Location__Longitude__s
            FROM
                Account
            WHERE
                Id =: accountId
                AND Location__Latitude__s != null
                AND Location__Longitude__s != null
            LIMIT 1
        ];
        /* Step0.1 - Prepare the Query Params */
        String queryParams =
            accRecord.Location__Latitude__s+'+'+accRecord.Location__Longitude__s;
        // As Per API Document
        System.debug(queryParams);

        // URLEncode of that complete Query parameter
        // EncodingUtil.urlEncode();

        /* Callouts Only */
        /* Step1 - Prepare the Request */
        HttpRequest httpReq = new HttpRequest();
        /* Step1.1 - Send the Endpoint */
        httpReq.setEndPoint(System.Label.OPENCAGE_API_URL+'?key='
            +System.Label.OPENCAGE_API_KEY+'&q='+queryParams+'&pretty=1');

    }
}
```

```

/* Step 1.2 - Set the Headers */
httpReq.setHeader('Content-Type', 'application/json');
httpReq.setHeader('Accept', 'application/json'); // JSON, XML, Text,
HTML

/* Step1.3 - Set the Method */
httpReq.setMethod('GET'); // GET, POST, PUT, PATCH, DELETE

/* Step2 - Send the Request */
Http htt = new Http();
try {
    HttpResponse httpRes = htt.send(httpReq);
    /* Step3 - Print the Information */
    /*
        getStatus(), getStatusCode()
        getBody() getXmlStreamReader() --> SOAP, getBodyAsBlob()
    */
    String responseBody = httpRes.getBody(); // String
    Integer statusCode = httpRes.getStatusCode();
    // GET, 200
    if(statusCode == 200){
        // main logic
        // JSON Class inside System Namespace
        // deserialize, dereializeUntyped, dereializeStrict
        OpenCageReverseResponseWrapper wrapper
        = (OpenCageReverseResponseWrapper)
        System.JSON.deserialize(responseBody,
        OpenCageReverseResponseWrapper.class);
        if(wrapper?.results?.size() > 0){
            // The Compiler
            // The Run Time Environment ( Machine )
            // wrapper?.results?.size() > 0
    }
}

```

```
// if wrapper is not null
    // Yes -
        // wrapper.results is not null
            // YES -
                // wrapper.results.size()
                    // YES

OpenCageReverseResponseWrapper.results rslt
= wrapper.results.get(0);
// Update Account Record
accRecord.BillingStreet = rslt?.components?.road;
accRecord.BillingCity = rslt?.components?.city;
accRecord.BillingState = rslt?.components?.state;
accRecord.BillingPostalCode = rslt?.components?.postcode;
accRecord.BillingCountry = rslt?.components?.country;

accRecord.ShippingStreet = rslt?.components?.road;
accRecord.ShippingCity = rslt?.components?.city;
accRecord.ShippingState = rslt?.components?.state;
accRecord.ShippingPostalCode =
rslt?.components?.postcode;
accRecord.ShippingCountry = rslt?.components?.country;
update accRecord;
}

}else{
    // error handling code...
}

}catch(System.CalloutException calloutEx){
    System.debug('System.CalloutException .... '
+ calloutEx.getStackTraceString());
    if(String.valueOf(calloutEx)
.startsWith('System.CalloutException: Unauthorised endpoint')){
```

```
// Remote Site missing Error -
System.debug(' CalloutException ');
}

}catch(System.Exception ex){
    System.debug('Exception Executed ... '+ ex.getStackTraceString());
    if(String.valueOf(ex)
        .startsWith('System.CalloutException: Unauthorised endpoint')){
        // Remote Site missing Error -
        System.debug(' Executed ');
    }
}
}

public class OpenCageReverseResponseWrapper{
    public results[] results;
    public class results {
        public components components;
        public Integer confidence;
        public String formatted;
        public geometry geometry;
    }
    public class components {
        public String city;    //Hanover
        public String city_district; //Vahrenwald-List
        public String continent; //Europe
        public String country; //Germany
        public String country_code; //de
        public String county; //Region Hannover
        public String house_number; //2
        public String office; //Design Offices
        public String political_union; //European Union
        public String postcode; //30165
    }
}
```

```
    public String road; //Philipsbornstraße
    public String state; //Lower Saxony
    public String state_code; //NI
    public String suburb; //Vahrenwald
}
public class geometry {
    public Double lat; //52.387783
    public Double lng; //9.7334394
}
}
```

Basic Authentication

In basic authentication, username & password are passed over the internet in the encoded form to the target system, and then the target system decodes to get the information. Validates the information in the system if the information is correct then returns the response.

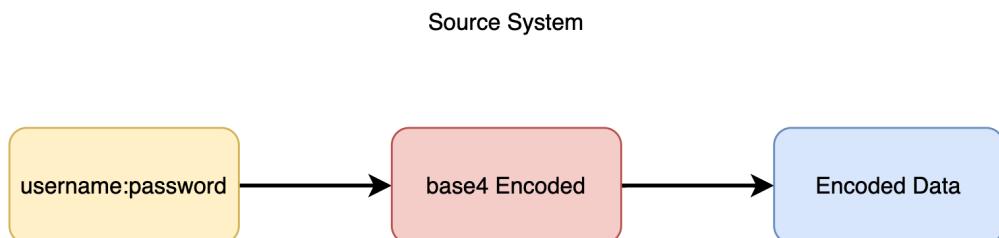
HTTPS/LTS should be used with the Basic Authentication without this additional security enhancement; the basic authentication should not be used to protect sensitive and valuable information.

The authentication will be sent in the header after encoding to Base64
Encode

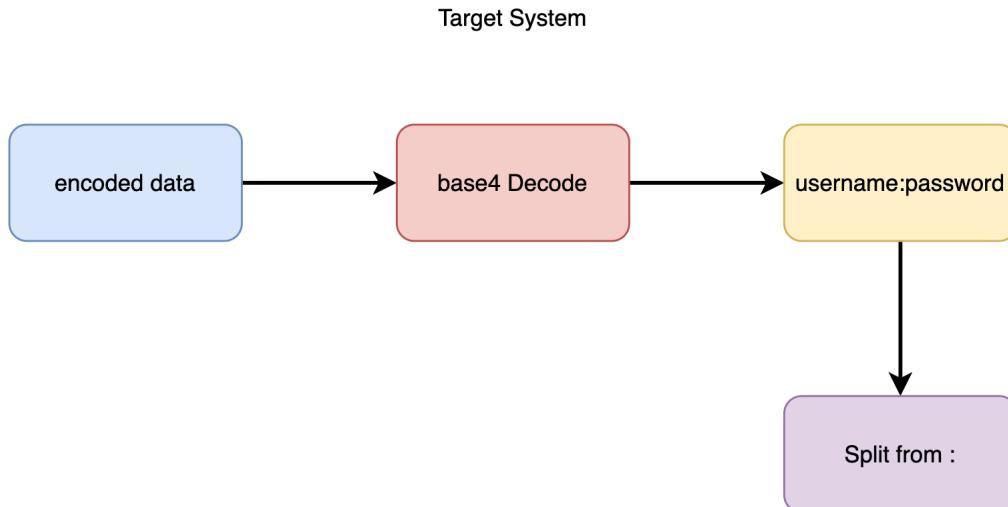
Steps

1. Send the base64 encoded value in the header. The encoding string should be in *username:password* format
2. The target system will do the *Base64Decode(encoded string)*
3. Split the information from : and get the username and password and validate the same in the system

Note: - Step 3 is as per my understanding



myUsername:myPassword → bXlVc2VybmFtZTpteVBhc3N3b3Jk



bXlVc2VybmFtZTpteVBhc3N3b3Jk → myUsername:myPassword

Online Tool

<https://www.base64encode.org/>

Apex Encoding

```
Blob header = Blob.valueOf(username + ':' + password);
String authHeader = 'Basic ' + EncodingUtil.base64Encode(header);
```

https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apexref/apex_classes_restful_encodingUtil.htm

Zendesk - <https://www.zendesk.com/register/>

Freshdesk

Kayako

```
public with sharing class PS_ZendeskTicketUtils {  
  
    /*  
     * Create a ticket using the wrapper  
     */  
    PS_ZendeskTicketUtils.TicektWrapper wrapper = new  
    PS_ZendeskTicketUtils.TicektWrapper();  
    wrapper.body = 'Dynamic body';  
    wrapper.subject = 'Testing from Salesforce Apex!';  
    wrapper.priority = 'urgent';  
    wrapper.name = 'Amit Singh';  
    wrapper.email = 'asingh@example.org';  
    PS_ZendeskTicketUtils.createTicket(wrapper);  
}  
  
public class TicektWrapper {  
    public String body;  
    public String subject;  
    public String priority;  
    public String name;  
    public String email;  
}  
  
public static void createTicket(TicektWrapper wrapper){ // Enhance it  
    /*  
     * Create a Case in Salesforce  
     */  
    Case -->  
    Case is Created in Salesforce
```

```
Zendesk
Salesforce Case should be updated with the Zendesk ticket Id
*/
String header = System.Label.Zendesk_Username
+':'+System.Label.Zendesk_APITOKEN;
Blob headerValue = Blob.valueOf(header);

String requestBody = '{'+
    ' "ticket": {'+
        ' "comment": {'+
            ' "body": "'"+wrapper.body+"'",+
        ' },'+

        ' "priority": "'"+wrapper.priority+"'",+
        ' "subject": "'"+wrapper.subject+"'",+
        ' "requester": {'+
            ' "locale_id": 8,'+
            ' "name": "'"+wrapper.name+"'",+
            ' "email": "'"+wrapper.email+"'",+
        ' },'+

        ' }'+

    '}'+

}';

System.debug('requestBody \n '+ requestBody);

HttpRequest httpReq = new HttpRequest();

httpReq.setEndpoint('https://pantherschools.zendesk.com/api/v2/tickets');
httpReq.setMethod('POST');
httpReq.setBody(requestBody); // String --> XML/JSON
httpReq.setHeader('Content-Type','application/json');
httpReq.setHeader('Accept','application/json');
httpReq.setHeader('Authorization','Basic
```

```
' + EncodingUtil.base64Encode( headerValue ) );\n\nHttp http = new Http();\ntry{\n    HttpResponse httpRes = http.send(httpReq);\n    if(httpRes.getStatusCode() == 201){\n        System.debug('SUCCESS \n '+httpRes.getBody());\n        // Success\n    } else {\n        // 400\n        System.debug('ERROR \n '+ httpRes.getBody());\n        // error\n    }\n} catch(System.CalloutException ex){\n    //\n} catch(System.Exception ex){\n\n}\n}\n}
```

```
trigger CaseTrigger on Case (after insert) {\n    CaseTriggerHandler.handleAfterInsert(Trigger.New); //List<Case>\n}
```

```
public with sharing class CaseTriggerHandler {\n\n    public static void handleAfterInsert(List<Case> newRecords) {\n        if(System.isFuture() || System.isBatch()){\n            return;\n        }\n    }\n}
```

```
for(Case c : newRecords){ // make a SQOL Query to get the Contact  
Details  
    PS_ZendeskTicketUtils.TicektWrapper wrapper = new  
    PS_ZendeskTicketUtils.TicektWrapper();  
    wrapper.body      = c.Description;  
    wrapper.subject   = c.Subject;  
    wrapper.priority = c.Priority.toLowerCase(); // Allowed values are  
"urgent", "high", "normal", or "low".  
    wrapper.name      = 'Amit Singh';  
    wrapper.email     = 'asingh@example.org';  
    // Converting the Object into String  
    makeCallout( JSON.serialize(wrapper) );  
}  
}  
  
@future(callout = true) // THIS is not the best Solution  
private static void makeCallout(String params){  
    // Convert the String in object(class)  
    PS_ZendeskTicketUtils.TicektWrapper wrapper =  
(PS_ZendeskTicketUtils.TicektWrapper)JSON.deserialize(params,  
PS_ZendeskTicketUtils.TicektWrapper.class);  
    PS_ZendeskTicketUtils.createTicket(wrapper);  
}  
}
```

Introduction to Connected Application

- A connected app is a framework that enables any external application to integrate with Salesforce using Standard or custom APIs OR can use any protocols like SAML, OAuth, OpenID Connect &, etc.
- OR if there is a connected application created within the external system salesforce can integrate and consume the external System API and work with the data.
- With the help of a connected app, we get very important information like client id & client secret
 - What are client id and client secrets, and why are they kept hidden? (SB)
- Furthermore, from the connected app, we can control what level of information the external system can access.
 - User
 - Profile
 - Location
 - And Many More
- What is the life of tokens generated and what ways to set life to it?

Resource form Salesforce Connected Application -

https://help.salesforce.com/s/articleView?id=sf.connected_app_overview.htm&type=5

Use Cases for Connected Application - SFDC

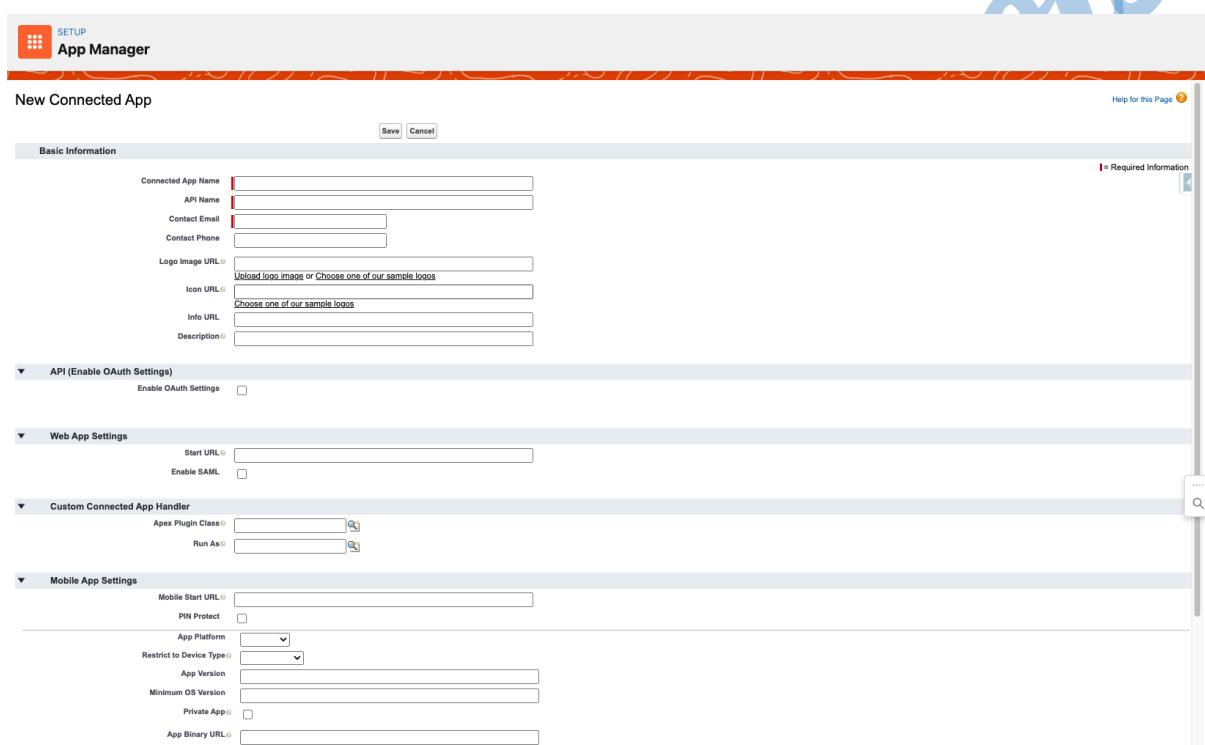
- Access or Manipulate Salesforce Data
 - E-Commerce
 - Inventory Management
 - Order Management System
 - SAP
 - Oracle
 - MuleSoft
- Enable SSO to your salesforce environment

- Manage access to a third-party application
- Provide Authorization for External API Gateways

Create a Connected Application

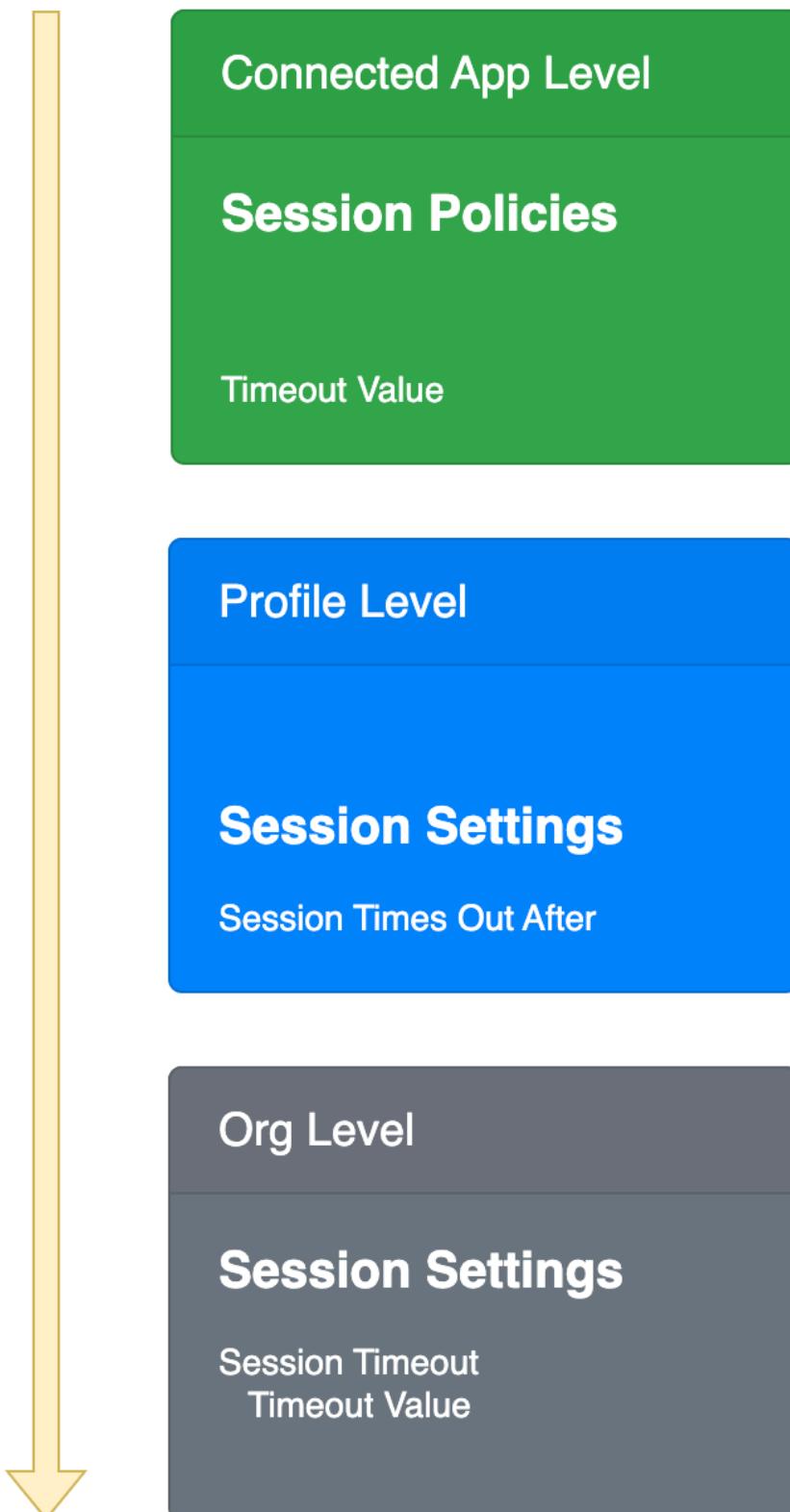
Redirect URLs

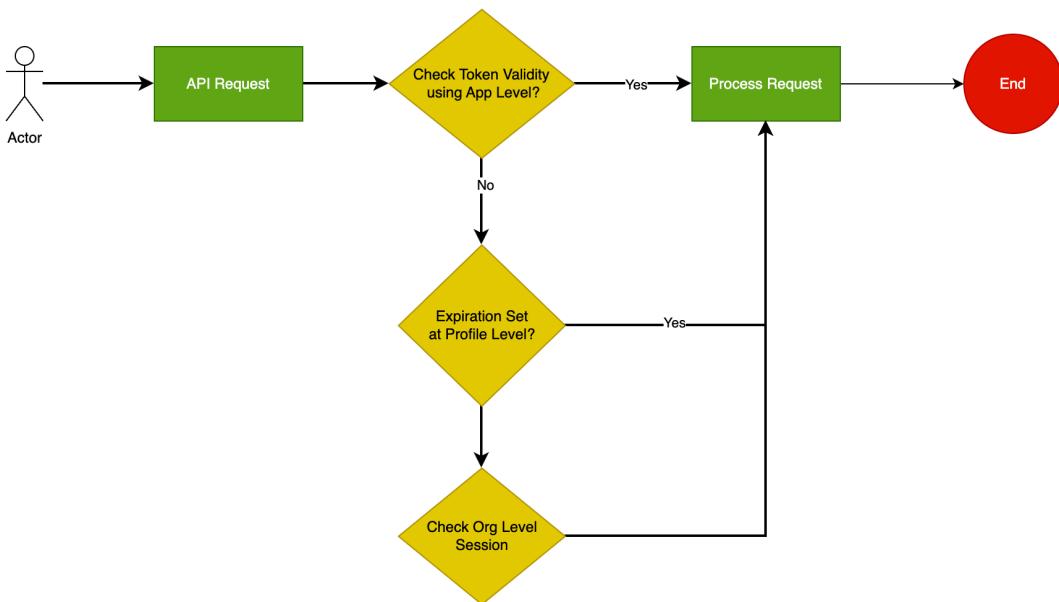
- <https://login.salesforce.com/services/oauth2/success>
- <https://test.salesforce.com/services/oauth2/success>



The screenshot shows the Salesforce App Manager interface for creating a new connected app. The page is titled "New Connected App" under the "Basic Information" section. It includes fields for "Connected App Name", "API Name", "Contact Email", "Contact Phone", "Logo Image URL", "Icon URL", "Info URL", and "Description". Below this, there are sections for "API (Enable OAuth Settings)" (with an unchecked checkbox for "Enable OAuth Settings"), "Web App Settings" (with fields for "Start URL" and "Enable SAML"), "Custom Connected App Handler" (with fields for "Apex Plugin Class" and "Run As"), and "Mobile App Settings" (with fields for "Mobile Start URL", "PIN Protect", "App Platform", "Restrict to Device Type", "App Version", "Minimum OS Version", "Private App", and "App Binary URL"). At the top right, there are "Save" and "Cancel" buttons, and a "Help for this Page" link.

How Salesforce manage Access Token Expiration Time





Introduction to OAuth2.0

Introduction to oAuth 2.0 Authorization Framework

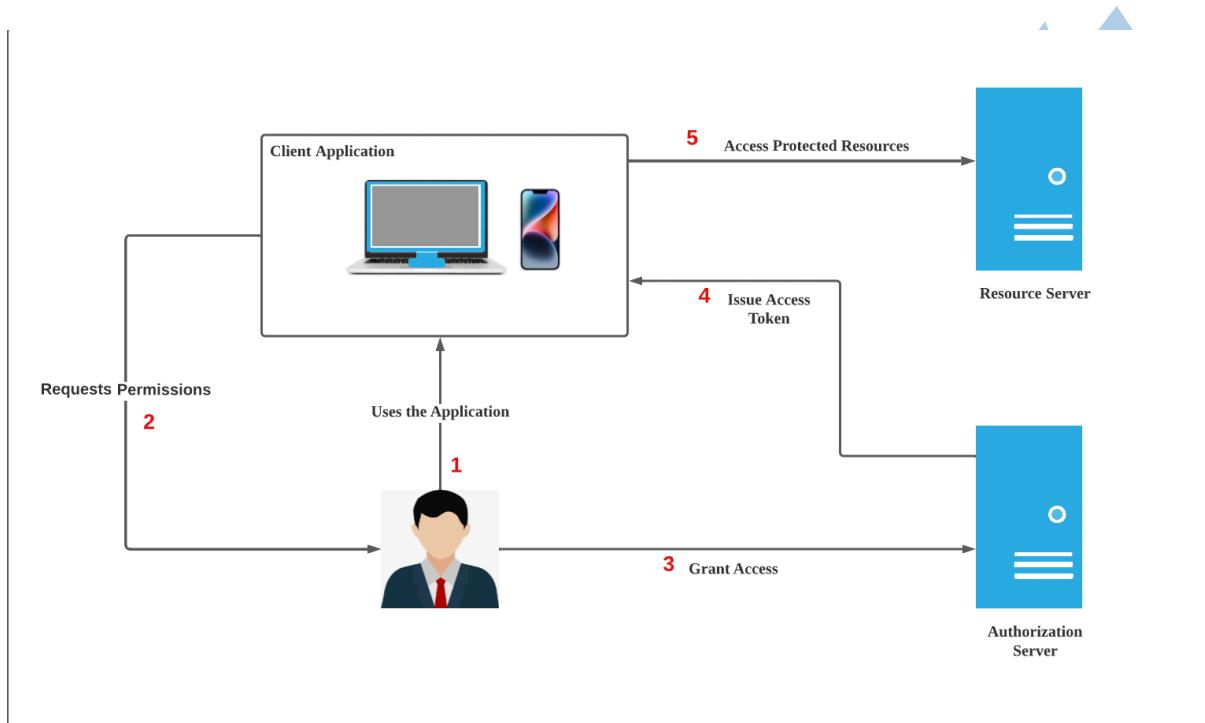
- The **OAuth 2.0 authorization framework** is a protocol that allows a user to grant a third-party web or application access to the user's protected resources, without necessarily revealing their long-term credentials or even their identity.
- OAuth introduces an authorization layer and separates the role of the client from that of the resource owner.
- In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server and is issued a different set of credentials than those of the resource owner

Roles in oAuth 2.0

An OAuth 2.0 flow has the following roles:

- Resource Owner: Entity that can grant access to a protected resource. Typically, this is the end-user.

- Resource Server: Server hosting the protected resources. This is the API you want to access.
- Client: Application requesting access to a protected resource on behalf of the Resource Owner.
- Authorization Server: Server that authenticates the Resource Owner and issues access tokens after getting proper authorization.



<https://tbid.digital.salesforce.com/oauth2/v1/authorize/callback>

Grant Types in oAuth 2.0

OAuth 2.0 defines four flows to get an access token and these flows are called grant types. Deciding which one is suited for your case depends mostly on your application type.

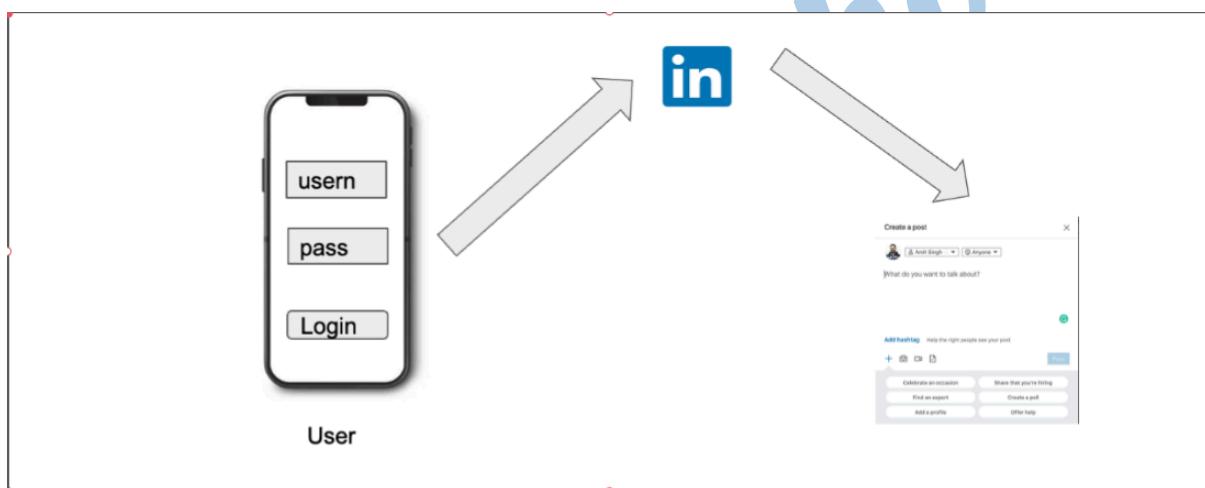
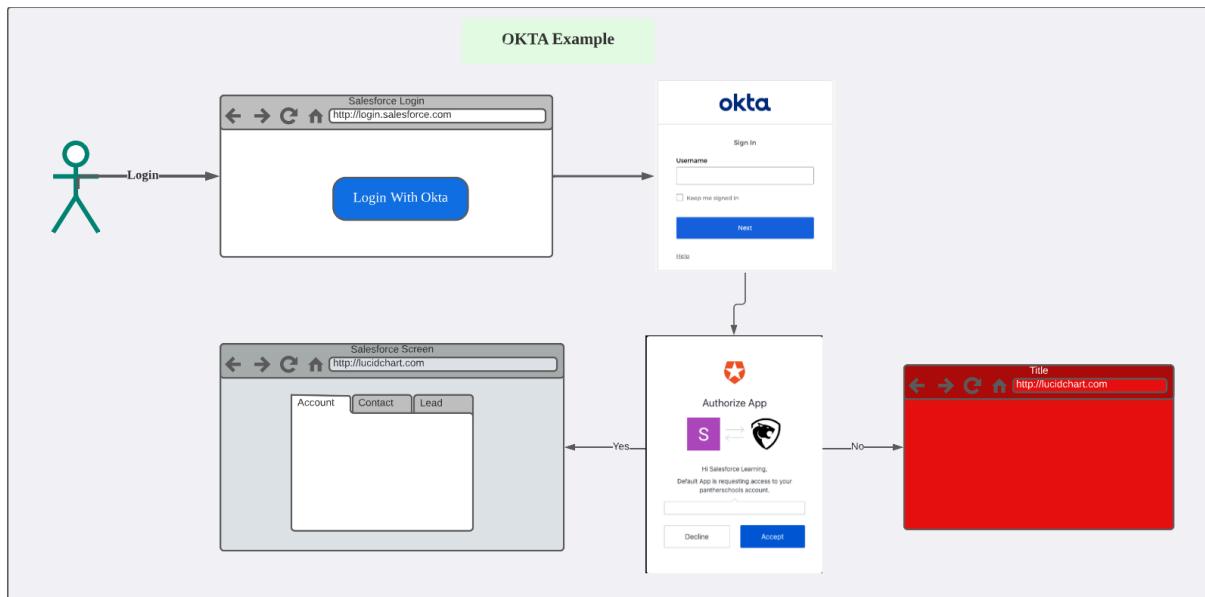
- *Authorization Code Flow*: used by Web Apps executing on a server. This is also used by mobile apps, using the Proof Key for Code Exchange (PKCE) technique.
- *Implicit Flow with Form Post*: used by JavaScript-centric apps (Single-Page Applications) executing on the user's browser.

- *Resource Owner Password Flow*: used by highly trusted apps.
recommended *Client Credentials Flow* instead
- *Client Credentials Flow*: used for machine-to-machine communication.

Which OAuth 2.0 Flow Should I Use?

- Is the Client the Resource Owner? – Client Credentials Flow
- Is the Client a web app executing on the server? - Authorization Code Flow OR Authorization Code Flow with Proof Key for Code Exchange (PKCE)
- Is the Client trusted with user credentials? - Resource Owner Password Flow - Never Use this
- Is the Client a Single-Page App? - Authorization Code Flow with Proof Key for Code Exchange (PKCE) and the Implicit Flow with Form Post - User-Agent Flow
- Is the Client a Native/Mobile App? - Authorization Code Flow with Proof Key for Code Exchange (PKCE)

Example

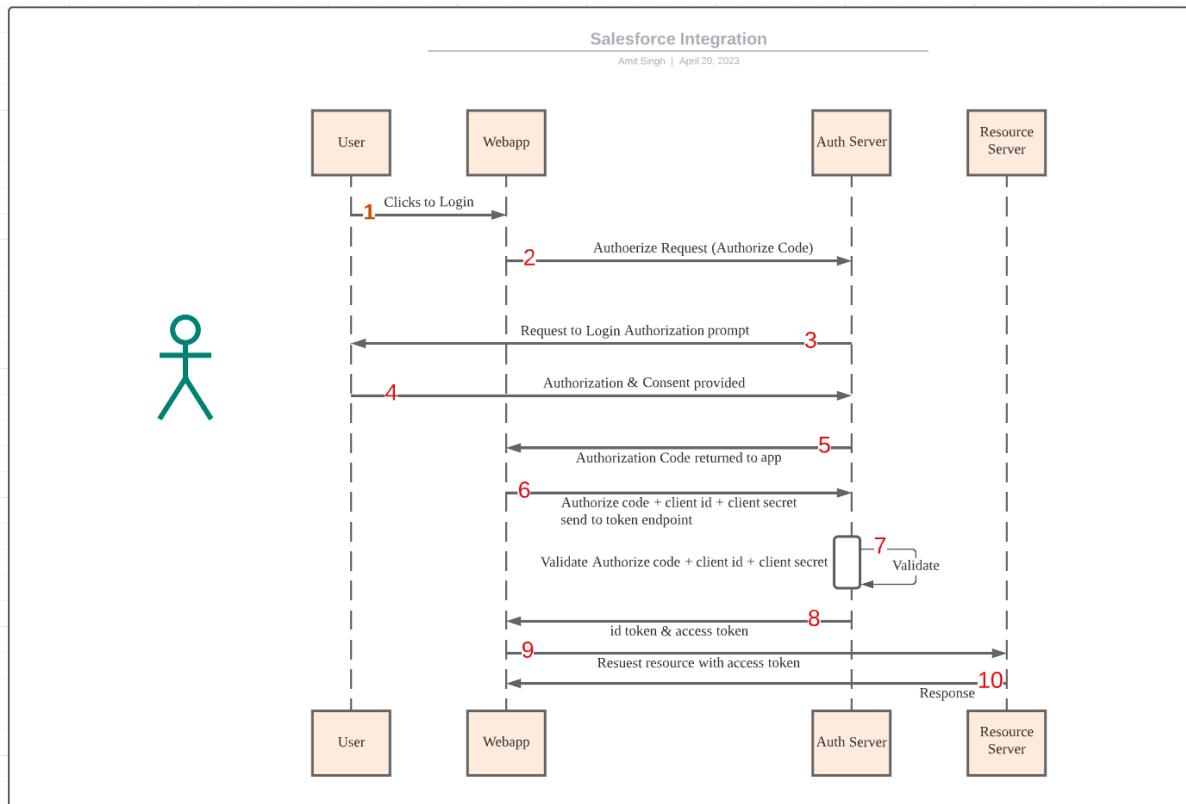


Showcase the demo using the <https://auth0.com/> application

OAuth2.0 Authorization Code Flow (Web Server Flow)

Prerequisite

- Postman tool must need to be installed in your ORG -
RECOMMENDED
- OR Advanced REST client Google Chrome Extension
- OR install Thunder Client in VS Code



oAuth 2.0 Step1 - Prepare authorization code URL & Get Authorization Code

Parameters needed to prepare the URL

- Authorization endpoint
- response_type – **code** for Auth Code flow and **token** for implicit flow
- state
- Redirect URI(redirect_uri)
- scope - While integrating with Salesforce the scopes are controlled from the connected application.
- client_id

Note: - We can get the client id & client secret from the connected application that we have created in the previous videos

Example Details

- Key - 3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91U48_

- Secret - 647FC2EB318E4752B4FF4B0CA98E74ACD83955D542

The Authorization endpoint is going to look like the below for the Salesforce Org.

- For Production or Developer Org -
<https://login.salesforce.com/services/oauth2/authorize>
- For Sandbox - <https://test.salesforce.com/services/oauth2/authorize>

You can find the relevant scope related to Salesforce from the given link -

https://help.salesforce.com/s/articleView?id=sf.remoteaccess_oauth_tokens_scopes.htm&type=5

Below is how your URL should look

https://login.salesforce.com/services/oauth2/authorize?client_id=3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91U48_Sf1bupeXUaxLs5nwz3Y0RP&redirect_uri=https://login.salesforce.com/services/oauth2/success&response_type=code&state=abc12434534

→ Now copy the URL that you have prepared in the above step, open any browser tab, paste the link, and hit enter

If prompted to login then please do log in and then approve the permissions from the permission screen



Allow Access?

Udemy Integration is asking to:

- Access the identity URL service
- Manage user data via APIs
- Manage user data via Web browsers
- Access Connect REST API resources
- Access Visualforce applications
- Access unique user identifiers
- Access custom permissions
- Access Analytics REST API resources
- Access Analytics REST API Charts Geodata resources
- Manage hub connections
- Manage Pardot services
- Access Lightning applications
- Access content resources
- Manage Customer Data Platform Ingestion API data
- Manage Customer Data Platform profile data
- Perform ANSI SQL queries on Customer Data Platform data
- Access chatbot services
- Perform segmentation on Customer Data Platform data
- Manage Customer Data Platform Identity Resolution
- Access Headless Forgot Password API
- Manage Customer Data Platform Calculated Insight data
- Perform requests at any time

Do you want to allow access for
sfdcpanther+hindi@gmail.com? ([Not you?](#))

Deny

Allow

To revoke access at any time, go to your personal settings.

→ Now, it will take you to the redirect URI and you will see the response below where the code is given in the URL parameters along with the state

<https://login.salesforce.com/services/oauth2/success?code=aPrxUHjncL9lYc5vP5LCztq.hn9edWI0TMbP0MozNli6OyyCWFNeiwK7xAHi8ja45lpD3pISA%3D%3D&state=abc12434534>

→ Make a note of the value of the code parameter

oAuth 2.0 Step2 - Prepare token URL & get the access token

To get the token we need to make use of any web service testing tool like Postman, Advance Rest Client &, etc. For our demo purpose, we will use Postman.

Below is the information that we need to keep handy for getting the access token

- Token endpoint
- code – we have got from the first step
- client_id
- client_secret
- redirect_uri
- grant_type = authorization_code

→ The very first step is to get the Token Endpoint and for salesforce depending upon the type of environment you can use any one of the below

- For Production OR Developer Org -
<https://login.salesforce.com/services/oauth2/token>
- For Sandbox OR Scratch ORG -
<https://test.salesforce.com/services/oauth2/token>

- Content-Type needs to be sent as a header in the request and the value must be application/x-www-form-urlencoded
- Method must need to be POST to get the access token

The screenshot shows the Postman interface for a POST request to <https://login.salesforce.com/services/oauth2/token>. The 'Headers' tab is selected, showing a single entry for 'Content-Type' with the value 'application/x-www-form-urlencoded'. The 'Method' dropdown at the top is set to 'POST'.

- Now, prepare the body with the above information

As we talked about, the content type of the request should be application/x-www-form-urlencoded so we need to pass the body in the same format. This is how the body will look like.

- After preparing the body once you click on send you will see either a success or error response.

The screenshot shows the Postman interface for the same POST request. The 'Body' tab is selected, and the 'form-data' radio button is selected. The body contains five key-value pairs: 'grant_type' (value: 'authorization_code'), 'redirect_uri' (value: 'https://login.salesforce.com/services/oauth2/success'), 'client_id' (value: '3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91U...'), 'client_secret' (value: '647FC2EB318E4752B4FF4B0CA98E74ACD83955D5...'), and 'code' (value: 'aPrxUHjncl9!Yc5vP5LCztq.huT8yBTgesVAodFaiMtlvS...'). The status bar at the bottom shows 'Status: 200 OK'.

Note: - You must need a URL to encode your authorization code before using Postman. You can use the <https://www.urldecoder.org/> tool to decode

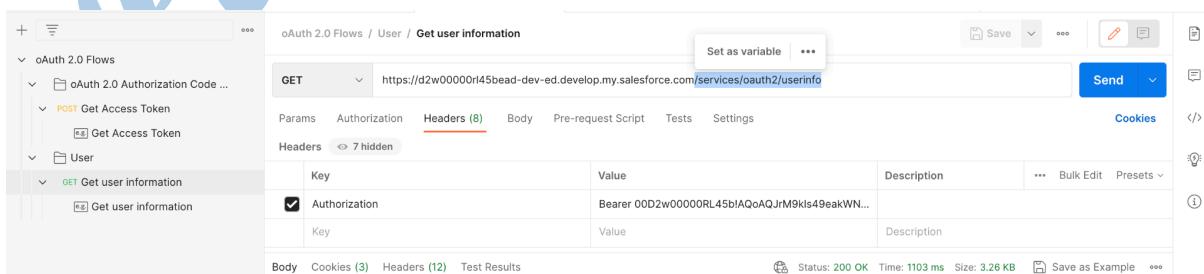
the Value. The URL decode is not required while achieving the same user code.

Success Response

```
{  
  "access_token": "ACCESS TOKEN HERE",  
  "refresh_token": "REFRESH TOKEN HERE",  
  "scope": "refresh_token web api id full",  
  "id_token": "ID TOKEN HERE",  
  "instance_url":  
    "https://yourdomain-dev-ed.develop.my.salesforce.com/",  
  "id":  
    "https://login.salesforce.com/id/00D2w00000RL45bEAD/0052w00000G3Z1  
pAAF",  
  "token_type": "Bearer",  
  "issued_at": "1682002671407"  
}
```

Test the Access Token

→ Once you have got the access token you can execute the GET method for <https://yourdomain-dev-ed.develop.my.salesforce.com/services/oauth2/userinfo> endpoint to get the current user information.



The screenshot shows the Postman application interface. On the left, there's a sidebar with a tree view of OAuth 2.0 Flows, including 'oAuth 2.0 Flows' and 'User'. Under 'User', there are two items: 'GET Get user information' and 'Get user information'. The main workspace shows a 'Get user information' request. The 'Headers' tab is selected, showing a table with one row: 'Authorization' with value 'Bearer 00D2w00000RL45bIAQoAQJrM9kls49ekWN...'. Other tabs like 'Params', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible. At the bottom, there are buttons for 'Body', 'Cookies (3)', 'Headers (12)', 'Test Results', and status information: 'Status: 200 OK', 'Time: 1103 ms', 'Size: 3.26 KB', 'Save as Example', and 'Copy'.

Custom Metadata to Store the Credentials

The screenshot shows the 'Custom Metadata Types' page in the Salesforce setup. A specific type named 'Salesforce Config' is selected. The details pane shows the following configuration:

- Singular Label:** Salesforce Config
- Plural Label:** Salesforce Configs
- Object Name:** sfdc_Config
- API Name:** sfdc_Config_mdt
- Created By:** Amit Singh, 2/29/2024, 11:27 PM
- Description:** Public
- Protection Level:** Record Size 2,474
- Modified By:** Amit Singh, 2/29/2024, 11:27 PM

The 'Standard Fields' section lists fields like Created By, Developer Name, Master Label, Last Modified By, Namespace Prefix, and Is Protected.

The 'Custom Fields' section lists fields such as access_token_c, auth_url_c, client_id_c, client_secret_c, environment_c, expires_in_c, expires_in_time_c, instance_url_c, page_name_c, refresh_token_c, scope_c, and token_url_c, each with its API name, data type, field manageability, indexed status, controlling field, and modified by user.

Assignments

1. Create the Custom Object for the following Information for Zendesk API
 - a. Agent → Maps the Users from Zendesk
 - b. Ticket → Maps with custom Ticket Object or you can also leverage the Case Standard Object by creating some custom fields like Zendesk Ticket Id as External Id
 - i. Create a Lookup field on Case with the Agent Object and add the Lookup filter so that only users whose role is Agent can only be selected on Case Record.
 - c. requester → Maps to Contact Standard Salesforce Object but you might need to create some custom fields on Salesforce
- d. **Assignments to Completed**
 - i. Create a Batch Apex that will run everyday at 12:00 AM and will fetch all the Users from Zendesk and Upsert the Agent Object in Salesforce will all these details.
 - ii. Whenever a Case is Created in Salesforce, the same case will be created as a ticket in Zendesk and update the zendesk Id back to Case custom field.

1. If the Agent is populated on the Case record while creating, then the Zendesk ticket should be assigned to that Agent at the time of creation.
 - iii. If the agent was not populated and the case was updated to populated the Agent Field then make the API call to update the Zendesk Ticket and assign the Zendesk ticket to this updated Agent.
2. Create a Custom metadata that will store the Country Name, ISO Codes, Currency Code, Currency Symbol, latitude, longitude and The flag emoji
 - a. You can get the details using [this](#) link.
 - i. <https://cdn.jsdelivr.net/npm/country-flag-emoji-json@2.0.0/dist/index.json>
 - ii. <https://github.com/lukes/ISO-3166-Countries-with-Regional-Codes/blob/master/all/all.csv>
 - b. Now, develop an Apex Class that will be responsible for deploying any custom metadata in the bulk
 - c. Develop a Custom Lightning Web Component, that will display all the Custom Metadata Records in the Picklist and by default, the value should be blank.
 - i. Once the user selects any custom metadata, it should fetch all the records and display them in the editable table format.
 - ii. The table should have the ability to add a new row which will give the ability to add a new row
 - iii. On the click on the button “Create/Update Records” it should update the record in Custom Metadata.
3. Create a Custom Metadata to store the information about the Google API and the fields will be similar to the below screenshot.
 - a. **Note:-** this is for your reference only.

- b. Create an Apex Class which will be the response for Authorization, Getting the Access Token and Getting the refreshing token if the token is expired. Followings are the expected methods
- authorizeUrl()
 - getAccessToken()
 - checkTokenValidity()
 - refreshToken()
- c. Create a VF page which will have a single button and will be used as the callback or return_url and get the access token using this VF Page
- d. Verify that the token details are available in the Custom Metadata.

```
public with sharing class PS_SalesforceTokenUtils {
    public PageReference getAuthCode(){
        /*
        TODO: Get the Custom Metadata Record
    }
}
```

```
*/



    sfdc_Config__mdt config =
sfdc_Config__mdt.getInstance(System.Label.PS_SalesforceTokenLabel)
; // Use Custom Label

    if(config != null){

        String orgUrl = config.Environment__c == 'Production'
? 'https://login.salesforce.com' : 'https://test.salesforce.com';

        System.System.debug( orgUrl );



        String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+'/apex/'+config.Page
Name__c;
        System.System.debug( redirect_uri );


        String authorizeUrl =
orgUrl+config.authurl__c+'?client_id='+config.clientid__c

+'&redirect_uri=' + redirect_uri +'&response_type=code';
        System.System.debug( authorizeUrl );


        return new PageReference(authorizeUrl);
    } else {
        return null;
    }
}

public void getAccessToken(){

    String code =
ApexPages.currentPage().getParameters().get('code');
```

```

        System.debug(code);

        sfdc_Config_mdt config =
sfdc_Config_mdt.getInstance(System.Label.PS_SalesforceTokenLabel)
; // Use Custom Label

        if(config != null){

            String orgUrl = config.Environment__c == 'Production'
? 'https://login.salesforce.com' : 'https://test.salesforce.com';
            String tokenUrl = orgUrl+config.tokenUrl__c;
            String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+'/apex/'+config.Page
Name__c;
            System.System.debug( redirect_uri );

            String requestBody =
'code='+code+'&grant_type=authorization_code&client_id='
+config.clientid__c+'&client_secret='+config.clientsecret__c+'&red
irect_uri='+redirect_uri;

            System.debug(requestBody);

            HttpRequest httpReq = new HttpRequest();
            httpReq.setEndpoint(tokenUrl);
            httpReq.setMethod('POST');
            httpReq.setBody(requestBody);

httpReq.setHeader('Content-Type','application/x-www-form-urlencoded');
            httpReq.setHeader('Accept','application/json');

            Http http = new Http();

```

```

try{
    HttpResponse httpRes = http.send(httpReq);
    if(httpRes.getStatusCode() == 200){
        PS_SalesforceTokenWrapper wrapper =
(PS_SalesforceTokenWrapper)System.JSON.deserialize(httpRes.getBody()
(), PS_SalesforceTokenWrapper.class);
        /*
                TODO: Deploy the Custom Metadata
                ! How to to Deploy
                Metadata is a NameSpace
                CustomMetadata - Class
                CustomMetadataValue - Class
                Operations - Class
                DeployCallback - Interface
        */
        String fullName =
'sfdc_Config.'+System.Label.PS_SalesforceTokenLabel;
        String label =
System.Label.PS_SalesforceTokenLabel;
        Map<String, Object> fieldWithValuesMap = new
Map<String, Object>();
        fieldWithValuesMap.put('accesstoken__c',
wrapper.access_token);

//fieldWithValuesMap.put('expires_in__c',wrapper.);
        fieldWithValuesMap.put('expires_in_time__c',
System.now().addHours(2) );
        fieldWithValuesMap.put('instanceurl__c',
wrapper.instance_url);
        fieldWithValuesMap.put('refreshtoken__c',
wrapper.refresh_token);
    }
}

```

```

fieldWithValuesMap.put('scope__c',wrapper.scope);

fieldWithValuesMap.put('clientid__c',config.clientid__c);

fieldWithValuesMap.put('clientsecret__c',config.clientsecret__c);

System.debug('SUCCESS \n '+httpRes.getBody());

CreateUpdateMetadataUtils.createUpdateMetadata(fullName, label,
fieldWithValuesMap);

ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.CONFIRM, 'Successfull!'));

// Success

} else {
// 400
System.debug('ERROR \n '+ httpRes.getBody());
ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, httpRes.getBody() ));

// error
}

}catch(System.CalloutException ex){
ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));

}catch(System.Exception ex){
ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));

}

}

}

public static Boolean isValid(sfdc_Config_mdt config){
Boolean isValid = true;

```

```

        if(config.expires_in_time__c < System.now()){
            isValid = false;
        }
        return isValid;
    }

    public static Map<String, Object> refreshToken(sfdc_Config__mdt config){
        String orgUrl = config.Environment__c == 'Production' ?
'https://login.salesforce.com' : 'https://test.salesforce.com';
        String tokenUrl = orgUrl+config.tokenUrl__c;

        String requestBody =
'grant_type=refresh_token&client_id='+config.clientid__c
+'&client_secret='+config.clientsecret__c+'&refresh_token=' +config
.refreshtoken__c;

        System.debug(requestBody);
        HttpRequest httpReq =
PS_CalloutUtils.prepareRequest(tokenUrl, 'POST', requestBody, 'applic
ation/json', 'application/x-www-form-urlencoded');
        Map<String, Object> fieldWithValuesMap = new Map<String,
Object>();
        try{
            HttpResponse httpRes = (new Http()).send(httpReq);
            if(httpRes.getStatusCode() == 200 ||
httpRes.getStatusCode() == 201 ){
                PS_SalesforceTokenWrapper wrapper =
(PS_SalesforceTokenWrapper)System.JSON.deserialize(httpRes.getBody
(), PS_SalesforceTokenWrapper.class);
                fieldWithValuesMap.put('accesstoken__c',

```

```

        wrapper.access_token);

            fieldWithValuesMap.put('expires_in_time__c',
System.now().addHours(2) );



//CreateUpdateMetadataUtils.createUpdateMetadata(fullName, label,
fieldWithValuesMap);

}else{



}

}catch(System.CalloutException ex){

}catch(System.Exception ex){


}

return fieldWithValuesMap;
}

}

```



```

public with sharing class PS_AccountUtils {

    // PS_AccountUtils.createAccount('Ravi Grover', 'Hot');

    public static void createAccount(String name, String rating){
        sfdc_Config__mdt config =
sfdc_Config__mdt.getInstance(System.Label.PS_SalesforceTokenLabel); // Use
Custom Label

        if(config != null){

            Boolean isValid = PS_SalesforceTokenUtils.isValid(config);
            Map<String, Object> fieldWithValuesMap = new Map<String, Object>();
            String accessToken = config.accesstoken__c;
            if(!isValid){

                // config.accesstoken__c ~= Token Expired
                // Get the Access Token using Refresh Token
                fieldWithValuesMap = PS_SalesforceTokenUtils.refreshToken(config);
            }
        }
    }
}

```

```

accessToken = (String)fieldWithValuesMap.get('accesstoken__c');

}

String endpoint =
config.instanceurl__c+'/services/data/v60.0/sobjects/Account';

String requestBody = '{'+
'    "Name": "'+name+'",
'    "Rating": "'+rating+'",
'    "Phone": "9876543210",
'    "Industry": "Education",
'    "Active__c": "Yes"
'};

HttpRequest httpReq =
PS_CalloutUtils.prepareRequest(endpoint,'POST',requestBody,'application/json',
'application/json');

httpReq.setHeader('Authorization', 'Bearer '+accessToken);

try{
    HttpResponse httpRes = (new Http()).send(httpReq);
    if(httpRes.getStatusCode() == 200 || httpRes.getStatusCode() == 201){
        System.debug('Success \n '+ httpRes.getBody());
    }else{
        System.debug('ERROR \n '+ httpRes.getBody());
    }
}catch(System.CalloutException ex){
    //ApexPages.addmessage(new
    ApexPages.message(ApexPages.severity.ERROR, ex.getMessage()));
}catch(System.Exception ex){
    //ApexPages.addmessage(new
    ApexPages.message(ApexPages.severity.ERROR, ex.getMessage()));
}

/*
    TODO: Update The Custom Metadata with New Values
    ! Update the new Access Token
*/

```

```
if(fieldWithValuesMap.size() >0 ){
    String fullName =
'sfdc_Config'+System.Label.PS_SalesforceTokenLabel;
    String label = System.Label.PS_SalesforceTokenLabel;
    CreateUpdateMetadataUtils.createUpdateMetadata(fullName, label,
fieldWithValuesMap);
}
}
}
}
```

```
public with sharing class PS_CalloutUtils {

    public static HttpRequest prepareRequest(String endPoint,
String method, String body, String accept, String contentType){
        HttpRequest httpReq = new HttpRequest();
        httpReq.setEndpoint(endPoint);
        httpReq.setMethod(method);
        httpReq.setHeader('Content-Type', contentType);
        if(!String.isBlank(accept)){
            httpReq.setHeader('Accept',accept);
        }
        if(!String.isBlank(body)){
            httpReq.setBody(body);
        }
        return httpReq;
    }
}
```

```
/*
? Use Metadata Namespace to deploy/update the Custom Metadata
```

Record.

```
    TODO: Using Metadata.DeployCallback interface
*/
public class CreateUpdateMetadataUtils implements
Metadata.DeployCallback {
    public static final String JOB_ID = 'a0I0o00001MbaBJ';

    public void handleResult(Metadata.DeployResult result,
Metadata.DeployCallbackContext context) {
        if (result.status == Metadata.DeployStatus.Succeeded) {
            /*
                TODO : Fire the Platform Event or Log the Success
            */
            System.debug(' success : 😍 ' + result);
        } else {
            /*
                TODO : Fire the Platform Event or Log the Error
Message
            */
            System.debug(' fail : 😢 ' + result);
        }
    }

    public static void createUpdateMetadata(String fullName, String
label, Map<String, Object> fieldWithValuesMap){

        /* Step1 - Create the Custom Metadata Object from Metadata
NameSpace */
        Metadata.CustomMetadata customMetadata =  new
Metadata.CustomMetadata();
        customMetadata.fullName = fullName; //
sfdc_Config.PS_SalesforceToken
```

```
// API Name - Custom metadata Record
customMetadata.label      = label; // 'Salesforce Token'

for(String key : fieldWithValuesMap.keySet()){

    /* Step2 - Create the CustomMetadataValue Object from
Metadata NameSpace */
    Metadata.CustomMetadataValue customField = new
Metadata.CustomMetadataValue();

    /* Step2.1 - Prepare the field values */
    customField.field = key;
    customField.value = fieldWithValuesMap.get(key);

    /* Step2.2 - Add the field values to the custom
metadata */
    customMetadata.values.add(customField);
}

/* Step3 - Create the DeployContainer Object from Metadata
NameSpace */
Metadata.DeployContainer mdContainer = new
Metadata.DeployContainer();

/* Step4 - Add the metadata under container */
mdContainer.addMetadata(customMetadata);

/*
Step5
TODO: Deploy the metadata if test is not running using
Operations
*/
```

```

        Id jobId =
    Metadata.Operations.enqueueDeployment(mdContainer, new
CreateUpdateMetadataUtils());
    }
}

```

```

<apex:page id="thePage" lightningStylesheets="true"
showHeader="true" controller="PS_SalesforceTokenUtils" >
<script>
    window.onload = function(){
        let code  =  '{!$CurrentPage.parameters.code}';
        if( code ){
            fetchAccessToken(); // Action Function
        }
    }
</script>
<apex:form id="theForm" >
    <apex:outputPanel id="errorMessage">
    </apex:outputPanel>
    <apex:pageMessages ></apex:pageMessages>
    <apex:actionstatus id="theStatus">
        <apex:facet name="start">
            <div class="waitingSearchDiv" id="el_loading"
                style="background-color: #fbfbfb;
height:100%;opacity:0.65;width:100%;">
                <div class="waitingHolder" style="top: 100px;
width: 91px;">
                    

```

```
<span class="waitingDescription">Loading...</span>
</div>
</div>
</apex:facet>
</apex:actionstatus>

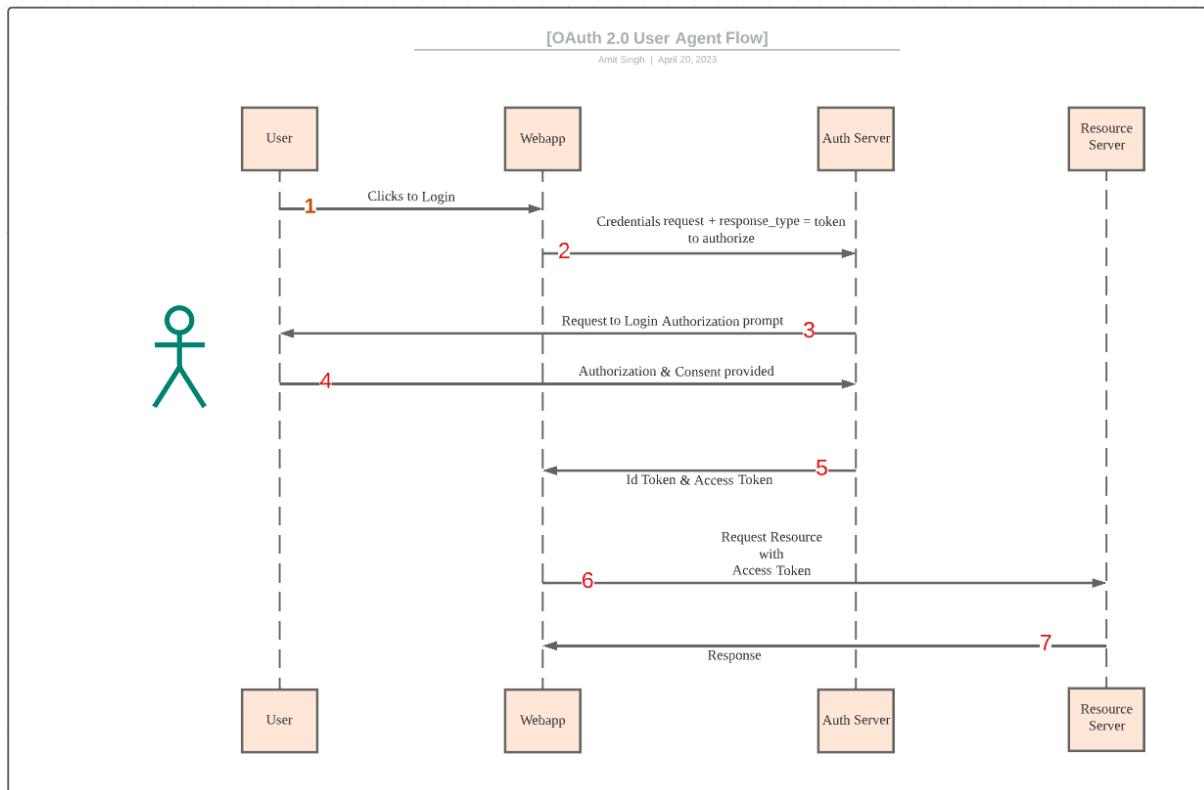
<apex:actionFunction name="fetchAccessToken" status="theStatus
errorMessage"
                      action="{!!getAccessToken}"
reRender="theForm" />

<apex:pageBlock >
    <apex:pageBlockButtons >
        <apex:commandButton value="Authorise Salesforce
Account" status="theStatus errorMessage"
                      action="{!!getAuthCode}" />
    </apex:pageBlockButtons>
</apex:pageBlock>

</apex:form>
</apex:page>
```

OAuth 2.0 User Agent flow (Implicit Flow)

User-agent flow is used when you are developing a Single Page Web Application, Web Application, or Mobile Application. Using this method you will get the access token in a single step and the access token will be available in the URL parameter itself.



The access token URL is going to be the same as Step #1 for the oAuth 2.0 Web Server flow. **The only change is that for the response_type we need to use the token instead of code.**

Note: - Please change the token URL based on your Salesforce Environment.

HERE IS THE SAMPLE URL

https://login.salesforce.com/services/oauth2/authorize?client_id=3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91U48_Sf1bupeXUaxLs5nwz3Y0RP.DD61jD8Q5p_EV.CZpxU&redirect_uri=https://login.salesforce.com/services/oauth2/success&response_type=token&state=abc12434534

Once you will hit the above URL in the browser it will prompt you to approve the access.



Allow Access?

Udemy Integration is asking to:

- Access the identity URL service
- Manage user data via APIs
- Manage user data via Web browsers
- Access Connect REST API resources
- Access Visualforce applications
- Access unique user identifiers
- Access custom permissions
- Access Analytics REST API resources
- Access Analytics REST API Charts Geodata resources
- Manage hub connections
- Manage Pardot services
- Access Lightning applications
- Access content resources
- Manage Customer Data Platform Ingestion API data
- Manage Customer Data Platform profile data
- Perform ANSI SQL queries on Customer Data Platform data
- Access chatbot services
- Perform segmentation on Customer Data Platform data
- Manage Customer Data Platform Identity Resolution
- Access Headless Forgot Password API
- Manage Customer Data Platform Calculated Insight data
- Perform requests at any time

Do you want to allow access for
sfdcpanther+hindi@gmail.com? ([Not you?](#))

Deny

Allow

To revoke access at any time, go to your personal settings.

→ After the successful authorization the access token will be present in the URL Parameters like below after the **# tag**.

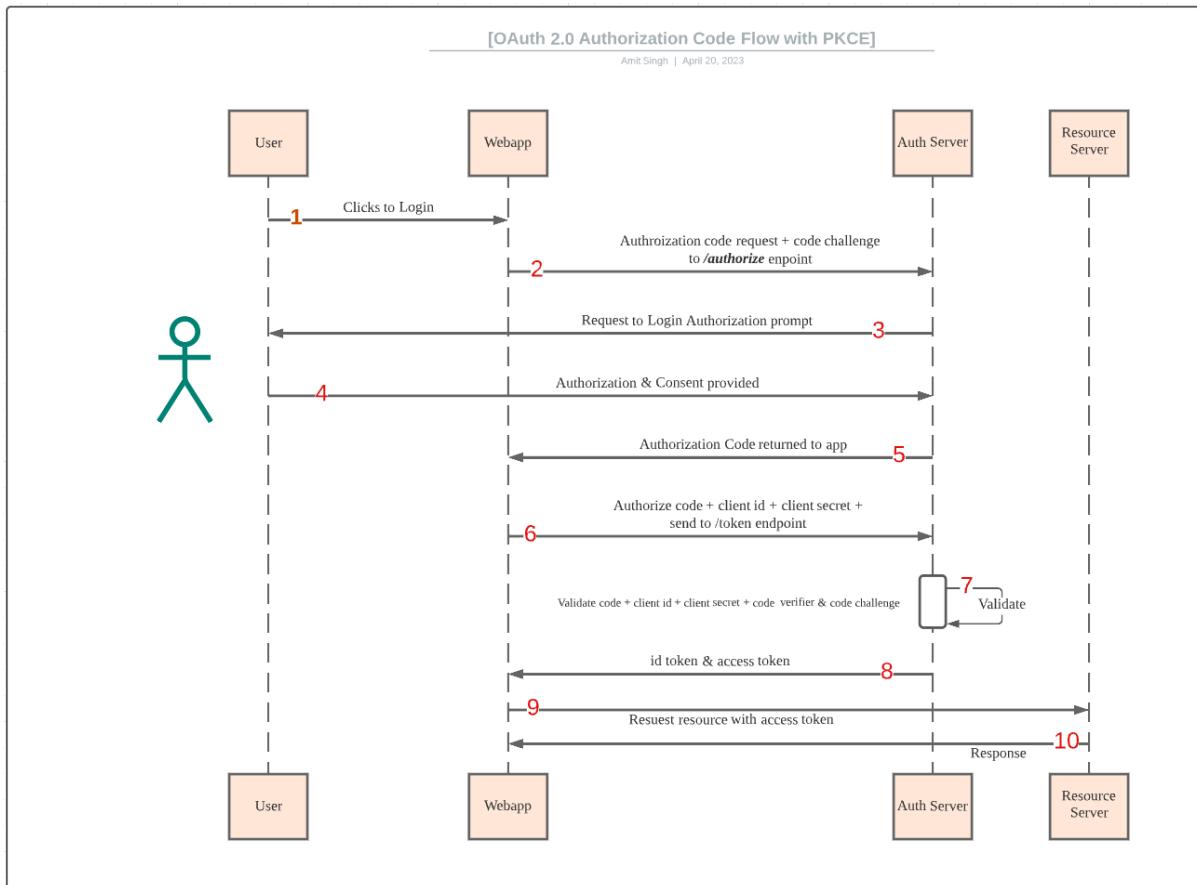
https://login.salesforce.com/services/oauth2/success#access_token=RZRxoSlls.tu3xJ6jMMh3uTpaBPCDZQRMQXep4HBj_5W9kUMXfRAInpEo0&refresh_token=5Aep861ZBQbtA4s3JUR520XzBQ4LMqGwxbjcxlDNMgettaiyx6xSN253LZqMHoqtZTCMyQIep96DaQ2eFD5fvH&instance_url=https%3A%2F%2Fd2w00000rl45bead-dev-ed.develop.my.salesforce.com&id=https%3A%2F%2Flogin.salesforce.com%2Fid%2F00D2w00000RL45bEAD%2F0052w00000G3ZlpAAF&issued_at=1682004309466&signature=jdCpXEz6qVwXjjGIkH8Sbzy5UOzTLcaHRXSgdBPTrIE%3D&state=abc12434534&scope=id+api+web+full+refresh_token&token_type=Bearer

Note:- The access token will be URL encoded so you have to URL decode before you start using the Access Token

OAuth 2.0 Authorization Code Flow with PKCE - Proof Key for Code Exchange

When client secrets cannot be secured using any server at the application, especially for the mobile application or single page web application.

OAuth code with PKCE (**Proof Key for Code Exchange**) is the most secure flow for mobile applications. Use this approach for native or single-page applications that cannot store a client secret.



Steps

- The very first step is to generate the Code Challenge and the code verifier by the application that is trying to get the access token.
 - **Note:** - For practice purposes, we can use an online tool to generate the code challenge and code verifier
 - <https://referbruv.com/utilities/pkce-generator-online/>
 - <https://tonyxu-io.github.io/pkce-generator/>
 - For Apex you can take a look at the below link
 - <https://gist.github.com/amitastreat/b3e38a66faa5e7449c463f0f6e9854fd>
 - https://help.salesforce.com/s/articleView?id=release-note_s.rn_security_pkce.htm&release=246&type=5
- After generating the request, now the application will make an HTTPS request to get the Authorization Code
- If the user is not logged in, the user is prompted to log in and provide

the consent

- Now, the resource server validates basic details like client ID and stores code challenge.
- If the validation is successful, the server generates an authorization code and gives it back to the app as a response.
- In this step (#2), the application will send the access token request using the **/token** endpoint and in the request body the app will pass the client ID, client secret, authorization code, code verifier, and redirect URI
- The Server will validate all the details and if the details are valid the access token will be returned along with other details.

How to Generate code_verifier & code_challenge?

→ **code_verifier** is a random string that will be sent in the access token call

→ **code_challenge** is a **SHA256 of code_verifier** which must be **base64 Url Encode**. This will be sent with an authorization call.

So, here are the steps

- Generate a random string and use that as code_verifier
- Generate the SHA256 of this random string (Crypto Class)
- Now, do the base64 Url Encode (EncodingUtils)

Step #1 - Generate the Authorization URL and get the Auth Code

Connected Application Details

- **Key** - 3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_
- **Secret** - 647FC2EB318E4752B4FF4B0CA98E74ACD83955D542

The Authorization endpoint is going to look like the below for the Salesforce Org.

- For Production or Developer Org -
[https://**login**.salesforce.com/services/oauth2/authorize](https://login.salesforce.com/services/oauth2/authorize)
- For Sandbox -
[https://**test**.salesforce.com/services/oauth2/authorize](https://test.salesforce.com/services/oauth2/authorize)

Sample PKCE Authorize URL -

[https://**login**.salesforce.com/services/oauth2/authorize?client_id=3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91Uwz3Y0RP.DD61jD8Q5p_EV.CZpxU&response_type=code&redirect_uri=https://login.salesforce.com/services/oauth2/success&code_challenge_method=S256&code_challenge=Hlx7ayeiGuPjKGjF7gG1HQWDxQzy52WY9n5reAcHCRw](https://login.salesforce.com/services/oauth2/authorize?client_id=3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91Uwz3Y0RP.DD61jD8Q5p_EV.CZpxU&response_type=code&redirect_uri=https://login.salesforce.com/services/oauth2/success&code_challenge_method=S256&code_challenge=Hlx7ayeiGuPjKGjF7gG1HQWDxQzy52WY9n5reAcHCRw)

If you are prompted to log in or approve, please do the needful.



Allow Access?

Udemy Integration is asking to:

- Access the identity URL service
- Manage user data via APIs
- Manage user data via Web browsers
- Access Connect REST API resources
- Access Visualforce applications
- Access unique user identifiers
- Access custom permissions
- Access Analytics REST API resources
- Access Analytics REST API Charts Geodata resources
- Manage hub connections
- Manage Pardot services
- Access Lightning applications
- Access content resources
- Manage Customer Data Platform Ingestion API data
- Manage Customer Data Platform profile data
- Perform ANSI SQL queries on Customer Data Platform data
- Access chatbot services
- Perform segmentation on Customer Data Platform data
- Manage Customer Data Platform Identity Resolution
- Access Headless Forgot Password API
- Manage Customer Data Platform Calculated Insight data
- Perform requests at any time

Do you want to allow access for
amitsingh@integration.org? ([Not you?](#))

[Deny](#)

[Allow](#)

To revoke access at any time, go to your personal settings.

Schools

Step #1 - Get the Access token using Postman

Access Token Endpoints

The very first step is to get the Token Endpoint and for salesforce depending upon the type of environment you can use any one of the below

- For Production OR Developer Org -

<https://login.salesforce.com/services/oauth2/token>

- For Sandbox OR Scratch ORG -

<https://test.salesforce.com/services/oauth2/token>

The screenshot shows the Postman interface for a POST request to <https://login.salesforce.com/services/oauth2/token>. The 'Body' tab is selected with 'x-www-form-urlencoded' chosen. The parameters are:

Key	Value	Description
code	aPrx9pB8PA1X2QPXhMe4v8ZXBK25MhLUJVUHbft4e...	
code_verifier	OJzljdehZCO8IQjv3D15SHD4t2ni28BJqHUctYEnqk60...	
client_id	((client_id))	
client_secret	((client_secret))	
redirect_uri	((redirect_uri))	
grant_type	authorization_code	

The response status is 200 OK.

Access Token Details

The screenshot shows the Postman interface with a tree view of OAuth 2.0 Flows and a detailed view of a POST request for a GET Access Token under a PKCE Flow. The request URL is https://login.salesforce.com/services/oauth2/token. The body is set to x-www-form-urlencoded with parameters: code, code_verifier, client_id, and client_secret. The response is a 200 OK status with a JSON payload containing tokens and other metadata.

```
public with sharing class PS_PKCEAuthHelper {
```

```
    /**
     * code_challenge - SHA 256 of code_verifier and must be base64 Url
     * encoded (Crypto)
     * First Call
     * code_verifier - Random String - UUID Class (
     System.UUID.getRandomString().toString() ); in Token Call
     * Second Call
     */

    /**
     * 1. Generate code_verifier
     * 2. SHA-256 of code verifier
     * 3. baseUrlEndoc of SHA-256 of code verifier
     * 4. Custom Setting / Custom Object to store code_verifier &
     code_challenge, an Active_c
     * 5. Get the data (code_verifier) from Custom Setting / Custom Object
```

```

* 6. Expire the code_verifier by setting Active__c checkbox to false
*/
public static List<String> getDetails(){

    String code_verifier = System.UUID.randomUUID().toString();
    Blob sha256        = Crypto.generateDigest('SHA-256',
Blob.valueOf(code_verifier));
    String code_challenge = SFDC_BASE64_URL_ENCODE(sha256);

    System.debug('**** code_verifier **** \n '+code_verifier);
    System.debug('**** code_challenge **** \n '+code_challenge);
    return new List<String>{code_verifier, code_challenge};
}

private static String SFDC_BASE64_URL_ENCODE(Blob input){
    if(input == null) {
        return null;
    }
    return EncodingUtil.base64Encode(input)
        .replace('/', '_')
        .replace('+', '-')
        .replaceAll('=+$', "");
}
}

```

```

public with sharing class PS_SalesforceTokenUtils {

```

```
public PageReference getAuthCode(){

/*
    TODO: Get the Custom Metadata Record
*/

sfdc_Config__mdt config =
sfdc_Config__mdt.getInstance(System.Label.PS_SalesforceTokenLabel); // Use Custom Label

if(config != null){

    List<String> codeVerifierChallenge =
PS_PKCEAuthHelper.getDetails();
    //codeVerifierChallenge.get(0); code_verifier
    //codeVerifierChallenge.get(1); code_challenge

    String orgUrl = config.Environment__c == 'Production' ?
'https://login.salesforce.com' : 'https://test.salesforce.com';

    System.System.debug( orgUrl );

    String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+'/apex/'+config.PageName__c;
    System.System.debug( redirect_uri );

    String authorizeUrl =
orgUrl+config.authurl__c+'?client_id='+config.clientid__c
        +'&redirect_uri=' + redirect_uri +'&response_type=code'
        +'&code_challenge=' + codeVerifierChallenge.get(1);
    System.System.debug( authorizeUrl );

    // Insert the Custom Object code_verifier, code_challenge
}
```

```

PKCEHelper__c helper = new PKCEHelper__c();
helper.Active__c = true;
helper.User__c = UserInfo.getUserId();
helper.code_challenge__c = codeVerifierChallenge.get(1);
helper.code_verifier__c = codeVerifierChallenge.get(0);
insert helper;

return new PageReference(authorizeUrl);
} else {
    return null;
}
}

public void getAccessToken(){

String code = ApexPages.currentPage().getParameters().get('code');
System.debug(code);
sfdc_Config__mdt config =
sfdc_Config__mdt.getInstance(System.Label.PS_SalesforceTokenLabel); //
Use Custom Label
if(config != null){
    // Get the Custom Object code_verifier, code_challenge
    PKCEHelper__c helper = [SELECT Id, Name, code_verifier__c
                           FROM
                           PKCEHelper__c
                           WHERE
                           User__c =: UserInfo.getUserId()
                           AND Active__c = True
                           LIMIT 1
];
String orgUrl = config.Environment__c == 'Production' ?

```

```
'https://login.salesforce.com' : 'https://test.salesforce.com';

    String tokenUrl = orgUrl+config.tokenUrl__c;
    String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+'/apex/'+config.PageName__c;
    System.System.debug( redirect_uri );

    String requestBody =
'code='+code+'&grant_type=authorization_code&client_id='
+config.clientid__c+'&client_secret='+config.clientsecret__c
+'&redirect_uri='+redirect_uri+'&code_verifier='+helper.code_verifier__c;

    System.debug(requestBody);

    HttpRequest httpReq = new HttpRequest();
    httpReq.setEndpoint(tokenUrl);
    httpReq.setMethod('POST');
    httpReq.setBody(requestBody);

httpReq.setHeader('Content-Type','application/x-www-form-urlencoded');
    httpReq.setHeader('Accept','application/json');

    Http http = new Http();
    try{
        HttpResponse httpRes = http.send(httpReq);
        if(httpRes.getStatusCode() == 200){
            PS_SalesforceTokenWrapper wrapper =
(PS_SalesforceTokenWrapper)System.JSON.deserialize(httpRes.getBody(),
PS_SalesforceTokenWrapper.class);
            /*

```

```

    TODO: Deploy the Custom Metadata
    ! How to Deploy
    Metadata is a NameSpace
        CustomMetadata - Class
        CustomMetadataValue - Class
        Operations - Class
        DeployCallback - Interface
    */
    String fullName =
'sfdc_Config'+System.Label.PS_SalesforceTokenLabel;
    String label = System.Label.PS_SalesforceTokenLabel;
    Map<String, Object> fieldWithValuesMap = new Map<String,
Object>();
    fieldWithValuesMap.put('accesstoken__c',
wrapper.access_token);
    //fieldWithValuesMap.put('expires_in__c',wrapper.);
    fieldWithValuesMap.put('expires_in_time__c',
System.now().addHours(2) );
    fieldWithValuesMap.put('instanceurl__c',
wrapper.instance_url);
    fieldWithValuesMap.put('refreshtoken__c',
wrapper.refresh_token);
    fieldWithValuesMap.put('scope__c',wrapper.scope);
    fieldWithValuesMap.put('clientid__c',config.clientid__c);

fieldWithValuesMap.put('clientsecret__c',config.clientsecret__c);

    System.debug('SUCCESS \n '+httpRes.getBody());
    CreateUpdateMetadataUtils.createUpdateMetadata(fullName,
label, fieldWithValuesMap);
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.CONFIRM,'Successful!'));

```

```

        helper.Active__c = false;
        // Success
    } else {
        // 400
        System.debug('ERROR \in '+ httpRes.getBody());
        ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, httpRes.getBody() ));
        // error
    }
}catch(System.CalloutException ex){
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));
}catch(System.Exception ex){
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));
}
}

public static Boolean isValid(sfdc_Config__mdt config){
    Boolean isValid = true;
    if(config.expires_in_time__c < System.now()){
        isValid = false;
    }
    return isValid;
}

public static Map<String, Object> refreshToken(sfdc_Config__mdt
config){
    String orgUrl = config.Environment__c == 'Production' ?
'https://login.salesforce.com' : 'https://test.salesforce.com';
    String tokenUrl = orgUrl+config.tokenUrl__c;

```

```

String requestBody =
'grant_type=refresh_token&client_id='+config.clientid__c
+'&client_secret='+config.clientsecret__c+'&refresh_token='+config.refreshToken__c;

System.debug(requestBody);
HttpRequest httpReq =
PS_CalloutUtils.prepareRequest(tokenUrl,'POST',requestBody,'application/json',
'application/x-www-form-urlencoded');

Map<String, Object> fieldWithValuesMap = new Map<String,
Object>();
try{
    HttpResponse httpRes = (new Http()).send(httpReq);
    if(httpRes.getStatusCode() == 200 || httpRes.getStatusCode() ==
201 ){
        PS_SalesforceTokenWrapper wrapper =
(PS_SalesforceTokenWrapper)System.JSON.deserialize(httpRes.getBody(),
PS_SalesforceTokenWrapper.class);
        fieldWithValuesMap.put('accesstoken__c',
wrapper.access_token);
        fieldWithValuesMap.put('expires_in_time__c',
System.now().addHours(2) );
        //CreateUpdateMetadataUtils.createUpdateMetadata(fullName,
label, fieldWithValuesMap);
    }else{
}
}catch(System.CalloutException ex){
}catch(System.Exception ex){
}

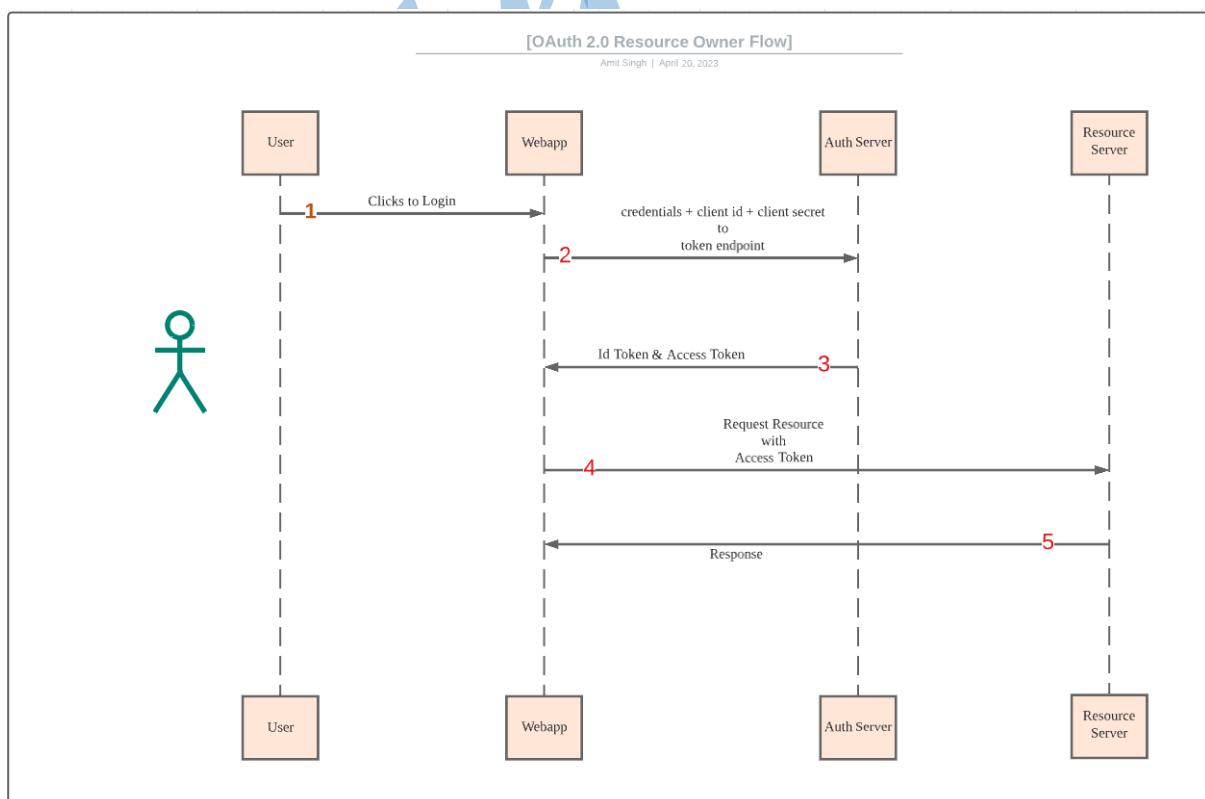
```

```
    }  
    return fieldWithValuesMap;  
}  
}  
}
```

OAuth 2.0 Resource Owner Flow (Username & Password) - Not recommended

This flow is used in a very rare case and is used only when the client and resource owner are the same person or company. As this flow uses the client credentials this has to be used very carefully.

Below is the flow illustrating the same



Sample Request

Before we start with this flow, there are a few things that need to be there with us so that we can access them easily.

- Token Endpoint
- **Client ID & Client Secret** - You can get this from the connected application
- Username
- Password
- **Security Token for the user** - If you do not have a security token you can use the
https://help.salesforce.com/s/articleView?id=sf.user_security_token.htm&type=5 link to reset the security token

Token Endpoint - The Token endpoint will depend upon your Salesforce environment. You can use any of the below where it is applicable

Dummy Connected Application Details

- **Key** - 3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_61jD8Q5p_EV.CZpxU
- **Secret** - 647FC2EB318E4752B4FF4B0CA98EED807258FD5B5B7A2F

The Token endpoint is going to look like the below for the Salesforce Org.

- For Production or Developer Org -
<https://login.salesforce.com/services/oauth2/token>
- For Sandbox OR Scratch ORG -
<https://test.salesforce.com/services/oauth2/token>

The content type header must need to be present and the value should be application/x-www-form-urlencoded.

Content-Type: application/x-www-form-urlencoded

As we are using the resource owner flow we need to use the **password as grant_type** and prepare the body like below.

Note: - While using the password you just need to use the security token and the password.

For example, if your password is ABCD123456789 & Your Security token is yqnQoe345BmTsdBKb3gWHUHS then you need to pass ABCD123456789yqnQoe345BmTsdBKb3gWHUHS as password for the password field.

Key	Value	Description
grant_type	password	
client_id	3MVG9n_HvETGhr3COGSBZ2RGw0cXRyKyG6_2F91...	
client_secret	647FC2EB318E4752B4FF4B0CA98E74ACD83955D5...	
username	({{username}})	
password	({{password}}){{security_token}}	

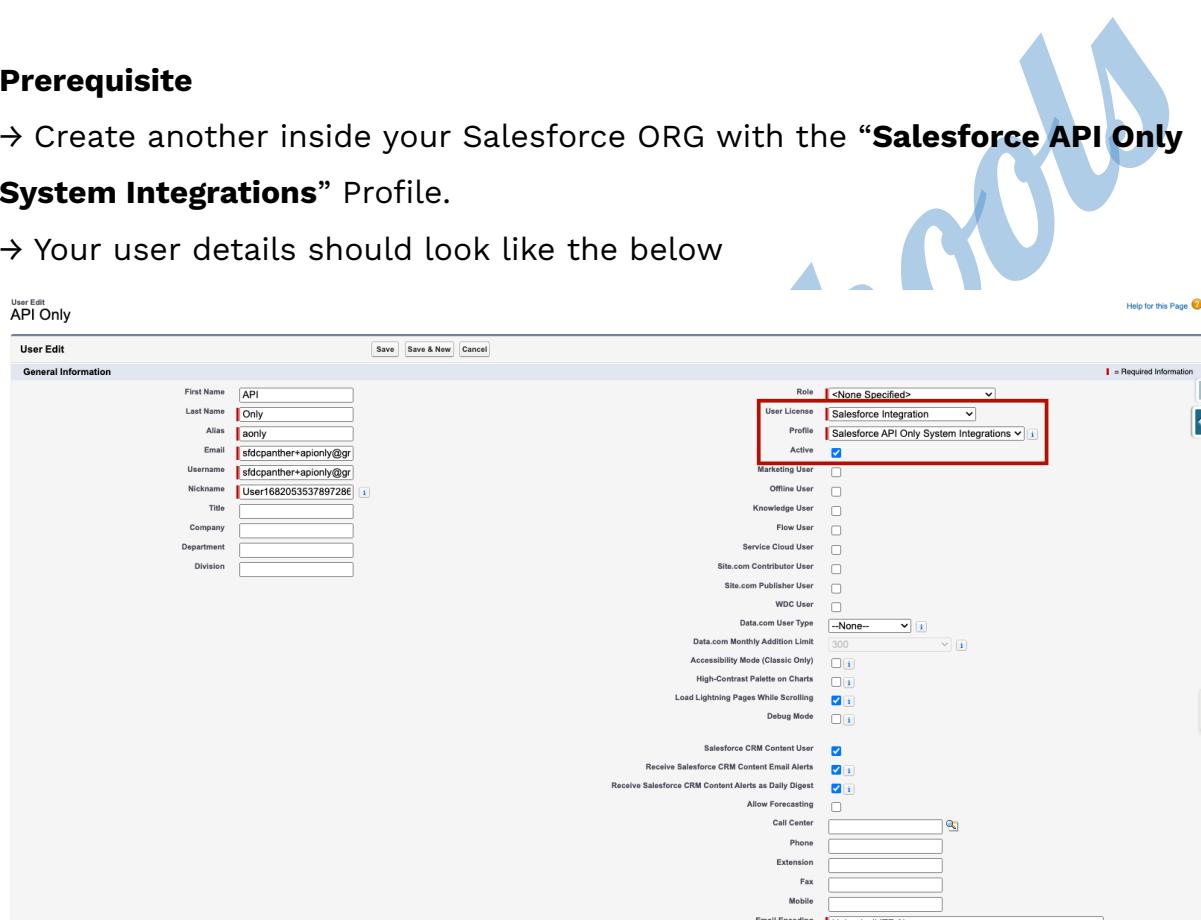
OAuth 2.0 Client Credentials Flow for Server-to-Server Integration

→ In the scenarios where you want to integrate one system into another system and where you do not want the users to get involved or use any business users' credentials you can use the OAuth 2.0 client credentials flow.

- To use the client credentials flow, a user will be pre-configured in the connected application and this user will be the API Only user.
- You can use **this method as an alternative method in a more secure way for Username & Password Flow.**

Prerequisite

- Create another inside your Salesforce ORG with the “**Salesforce API Only System Integrations**” Profile.
- Your user details should look like the below



User Edit
API Only

Save Save & New Cancel

General Information

First Name	API
Last Name	Only
Alias	aonly
Email	sfdcpanther+apionly@gr
Username	sfdcpanther+apionly@gr
Nickname	User168205353789728e
Title	
Company	
Department	
Division	

Role: <None Specified>

User License: Salesforce Integration

Profile: Salesforce API Only System Integrations

Active:

Marketing User:

Offline User:

Knowledge User:

Flow User:

Service Cloud User:

Site.com Contributor User:

Site.com Publisher User:

WDC User:

Data.com User Type: –None–

Data.com Monthly Addition Limit: 300

Accessibility Mode (Classic Only):

High-Contrast Palette on Charts:

Load Lightning Pages While Scrolling:

Debug Mode:

Salesforce CRM Content User:

Receive Salesforce CRM Content Email Alerts:

Receive Salesforce CRM Content Alerts as Daily Digest:

Allow Forecasting:

Call Center:

Phone:

Extension:

Fax:

Mobile:

Email Encoding: Unicode (UTF-8)

Step1 - Configure the Connected App for the Client Credentials Flow

- Login to **Salesforce**
- Navigate to **App Manager**
- Click on the dropdown next to your Connected Application & click on View
- Once you are on the Connected Application detail page, Now click on the **Edit** button

→ Enable **Enable Client Credentials Flow** checkbox & Save the changes

Connected App Name
Udemy Integration

Back to List: Custom Apps

Help for this Page

API (Enable OAuth Settings)

Consumer Key and Secret

Selected OAuth Scopes Access the identity URL service (id, profile, email, address, phone)
Manage user data via APIs (api)
Manage user data via REST API (rest)
Full access (full)
Perform actions at any time (refresh_token, offline_access)

Callback URL https://ogn.salesforce.com/api/oauth2/authSuccess
https://test.salesforce.com/services/oauth2/success

Enable for Device Flow

Require Secret for Web Server Flow

Require Secret for Refresh Token Flow

Enable Client Credentials Flow **Enable Client Credentials Flow**

Enable Authorization Code and Credentials Flow

Invalidate All Tokens

Token Valid for 0 Hour(s)

Include Custom Attributes

Include Custom Permissions

Enable Single Logout Single Logout disabled

Initial Access Token for Dynamic Client Registration

Initial Access Token

Custom Connected App Handler

Apex Plugin Class
Run As

Configure a Connected App for the OAuth 2.0 Client Credentials Flow

→ Now click on the “**Manage**” button to open the Manage page of the Connected Application
→ Then click on the “**Edit Policies**” Button

Connected App Name
Udemy Integration

Back to List: Custom Apps

Help for this Page

Edit **Delete** **Manage**

Version 1.0

→ Select “**Admin approved users are pre-authorized**” from the dropdown for “**Permitted Users**”
→ Select the API users that you have created as a prerequisite as **Run As**
→ Save the changes
→ Refer to the screenshot

Connected App
Udemy Integration

Help for this Page ⓘ

Connected App Edit

Connected App Edit

Basic Information

Start URL: [] ⓘ Version: 1 Description:

OAuth Policies

Permitted Users: Admin approved users are pre-authorized ⓘ

IP Relaxation

Refresh Token Policy:

Enforce IP restrictions: ⓘ

Refresh token is valid until revoked (radio button selected)

Immediately expire refresh token

Expire refresh token if not used for [] Day(s) ⓘ

Expire refresh token after [] Day(s) ⓘ

Session Policies

Timeout Value: [None] ⓘ High assurance session required: []

Custom Connected App Handler

Apex Plugin Class: [] ⓘ Run As: [] ⓘ

User Provisioning Settings

Enable User Provisioning: [] ⓘ

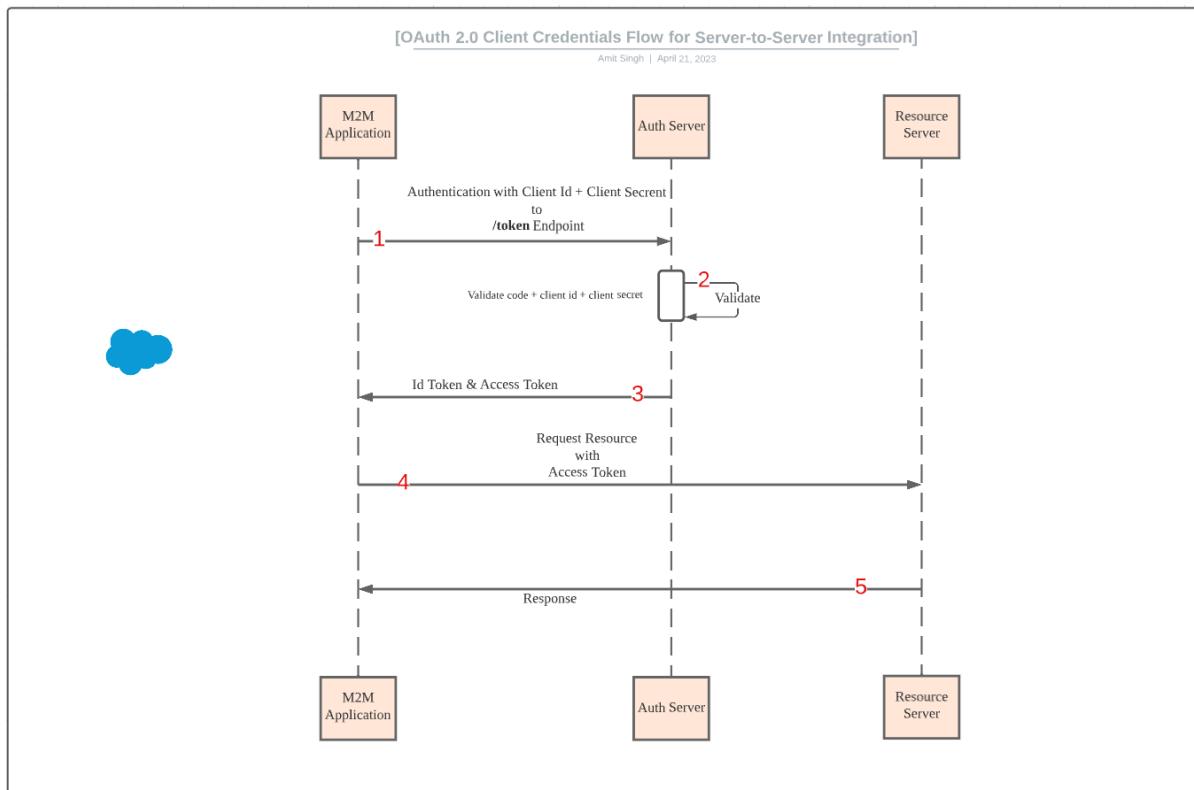
Client Credentials Flow

Run As: API Only ⓘ

Save Cancel



Visual Flow walkthrough



Step2 - Get Access Token

IMPORTANT → To get the access token we need to use the Custom Domain (**YOUR_MY_DOMAIN_URL**)

Here is the SAMPLE URL -

<https://YOUR-DOMAIN-dev-ed.develop.my.salesforce.com/services/oauth2/token>

→ We need to have the Client ID and Client Secret handy that will be used in the request body.

→ In the header, pass Content-Type as application/x-www-form-urlencoded

The screenshot shows the Postman interface with a successful API call to Salesforce's OAuth 2.0 endpoint. The request method is POST, the URL is <https://d2w00000rl45bead-dev-ed.develop.my.salesforce.com/services/oauth2/token>, and the body contains the following parameters:

Key	Value	Description
<input checked="" type="checkbox"/> client_id	<code>{{client_id}}</code>	
<input checked="" type="checkbox"/> client_secret	<code>{{client_secret}}</code>	
<input checked="" type="checkbox"/> grant_type	<code>client_credentials</code>	

The response status is 200 OK, and the JSON response body is:

```

1 "access_token": "00D2w00000RL45b!AQoAQ0hmGHRYoonW77TC2k_is07ozim02RVFR5p2RGQaGietlsXkjBhicIqwTr9ni8E5ajqExf8w9jW_L76kdbTsTxdu30V.",
2 "signature": "w02PGd/lm/G4bpe2N+FLLx8ewvRwQMYZEz39EyR1Jgw=",
3 "scope": "api id",
4 "instance_url": "https://d2w00000rl45bead-dev-ed.develop.my.salesforce.com",
5 "id": "https://login.salesforce.com/id/00D2w00000RL45bEAD/0052w00000H1Hv4AAE",
6 "token_type": "Bearer",
7 "issued_at": "Follow link (cmd + click)"
8
9

```

OAuth 2.0 Device Flow for IoT Integration - Using Salesforce for Demo

When you want to integrate with the devices such as Smart TVs, appliances, and other IoT devices we can use the OAuth 2.0 device flow.

For example, a customer uses a Bluetooth device to control their house lights while they're away for the evening. You can create a connected app for the Bluetooth device to enable this flow.

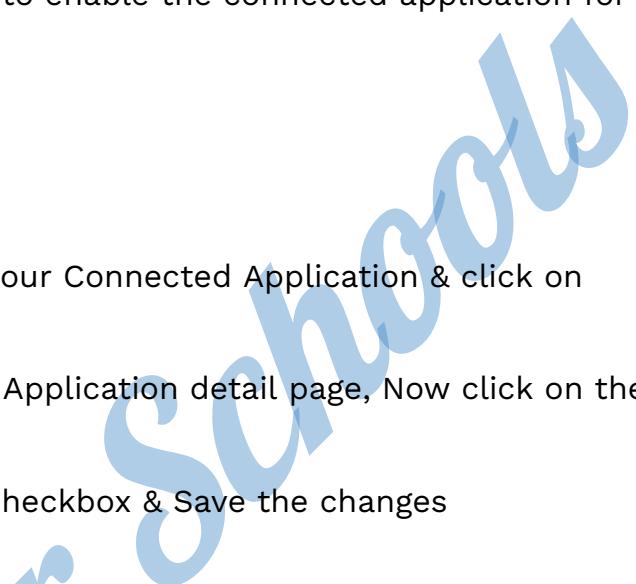
1. The user opens the Bluetooth app on their mobile device and clicks Turn On Lights.
2. The connected app posts a request to the Salesforce token endpoint.
3. Salesforce verifies the request and returns a human-readable user code, verification URL, and device code.
4. The Bluetooth app displays the device code and instructs the user to enter it at the specified verification URL. The app also begins polling the Salesforce token endpoint for authorization.
5. The user clicks the link to the verification URL and enters the code.
6. The user then authorises the app to access their protected data, in this case, their home's location.

7. Salesforce sends **Access and Refresh Tokens** to the connected app.
8. The Bluetooth app can access the user's home location and turn on the lights.

Configure the Connected Application

To use the divide ID flow we need to enable the connected application for using this flow.

- Login to **Salesforce**
- Navigate to **App Manager**
- Click on the dropdown next to your Connected Application & click on View
- Once you are on the Connected Application detail page, Now click on the **Edit** button
- Enable **Enable for Device Flow** checkbox & Save the changes



Connected App Name
Udemy Integration

Basic Information

Connected App Name:	Udemy Integration
API Name:	Udemy_Integration
Contact Email:	contentlearning@gmail.com
Contact Phone:	
Logo Image URL:	Upload logo image or Choose one of our sample logos
Icon URL:	Choose one of our sample logos
Info URL:	
Description:	

API (Enable OAuth Settings)

Enable OAuth Settings

Enable for Device Flow

Callbacks URL:
http://login.salesforce.com/services/oauth2/success
https://test.salesforce.com/services/oauth2/success

Use digital signatures:

Selected OAuth Scopes

Available OAuth Scopes	Selected OAuth Scopes
Access Analytics (TEST) API Charts Geodata resources (eclair_api) Access Chatter REST API resources (chatter_api) Access Connect REST API resources (chatter_api) Access Headless Forget Password API (forgot_password) Access Lightning applications (lightning) Access Visualforce applications (visualforce) Access chatbot services (chatbot_api) Access content resources (content) Access custom permissions (custom_permissions) Access unique user identifiers (openid)	Access the identity URL service (id, profile, email, address, phone) Full access Manage user data via APIs (api) Manage user data via Web browsers (web) Perform requests at any time (refresh_token, offline_access)

Add
Remove

Require Secret for Web Server Flow:

Require Secret for Refresh Token Flow:

Enable Client Credentials Flow:

Enable Authorization Code and Credentials Flow:

Introspect All Tokens:

Configure ID Token:

STEP#1 - Device Requests Authorization

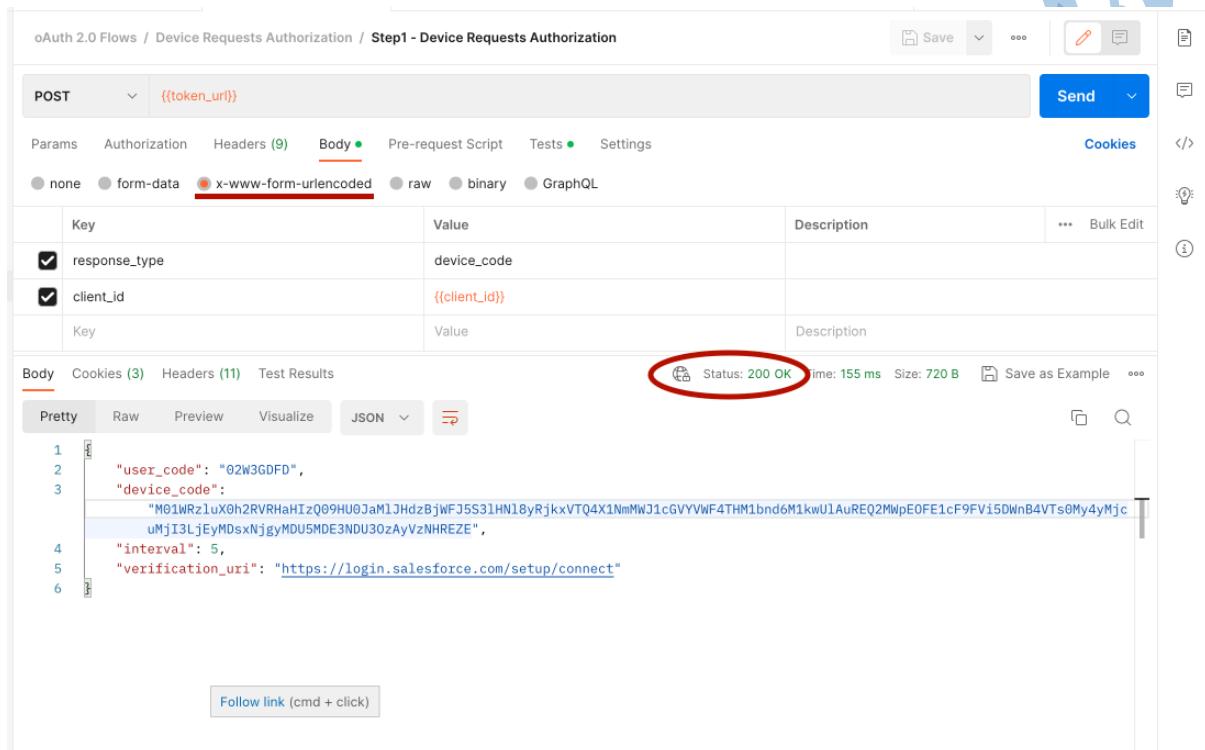
The very first step is to get the device code, in this step, the application will make a POST HTTP request to **/token** endpoint.

We will use the Postman for testing purposes

Token Endpoint - <https://login.salesforce.com/services/oauth2/token>

Content-Type: application/x-www-form-urlencoded

Request body - response_type=devide_code&client_id=YOUR_CLIENT_ID



The screenshot shows a Postman request to the Salesforce OAuth2 token endpoint. The request method is POST, and the URL is {{token_url}}. The 'Body' tab is selected, showing the following parameters:

Key	Value	Description
response_type	device_code	
client_id	{{client_id}}	

The response status is 200 OK, and the JSON response body is:

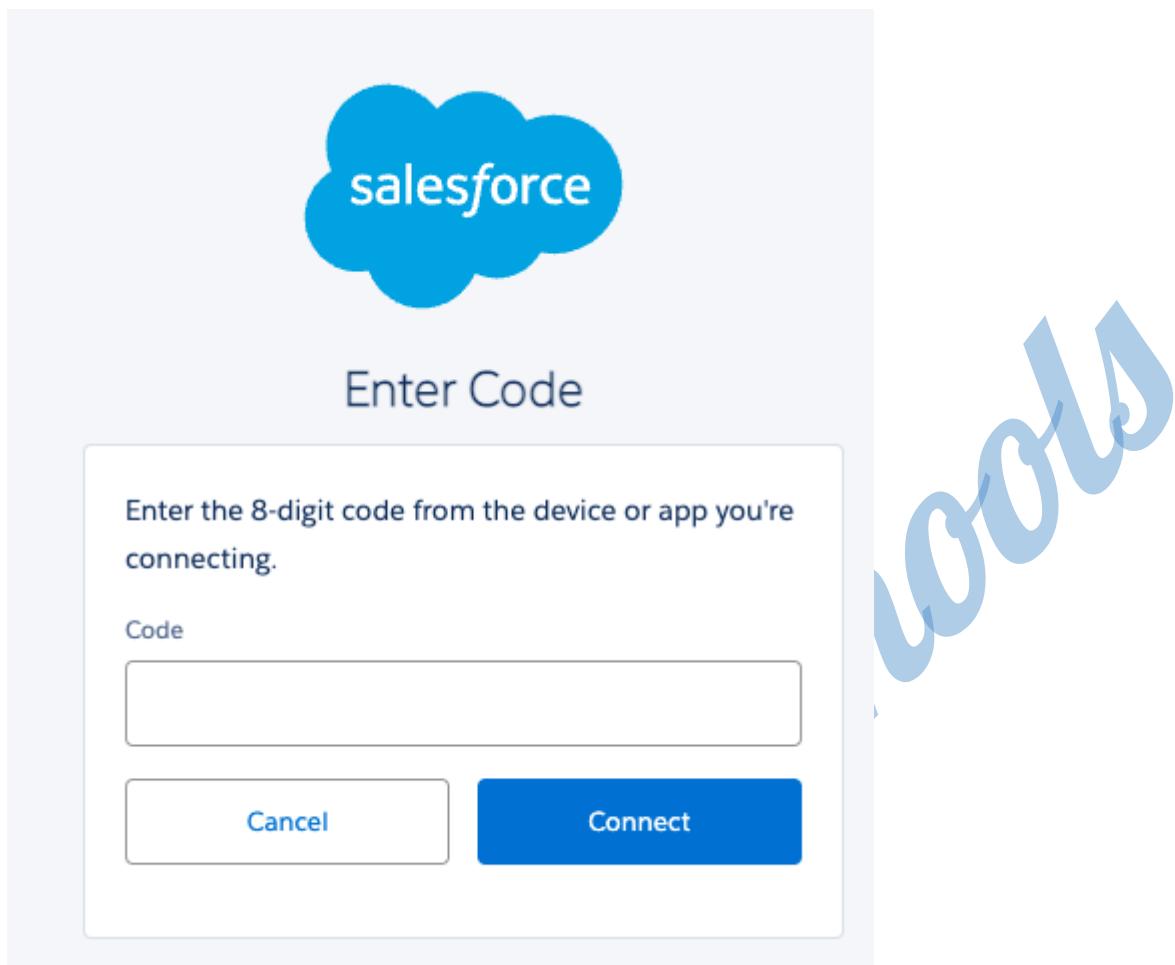
```
1 "user_code": "02W3GDFD",
2 "device_code":
3 "M01WRz1uX0h2RVRHaH1zQ09HU0JaM1JHdzBjWFJ5S31HN18yRjkxVTQ4X1NmMwJ1cGVYVWF4THM1bnd6M1kwU1AuREQ2MwpE0FE1cF9FVi5DWnB4VTs0My4yMjc
4 "interval": 5,
5 "verification_uri": "https://login.salesforce.com/setup/connect"
```

STEP#2 - VERIFY THE DIVIDE CODE (Done by User)

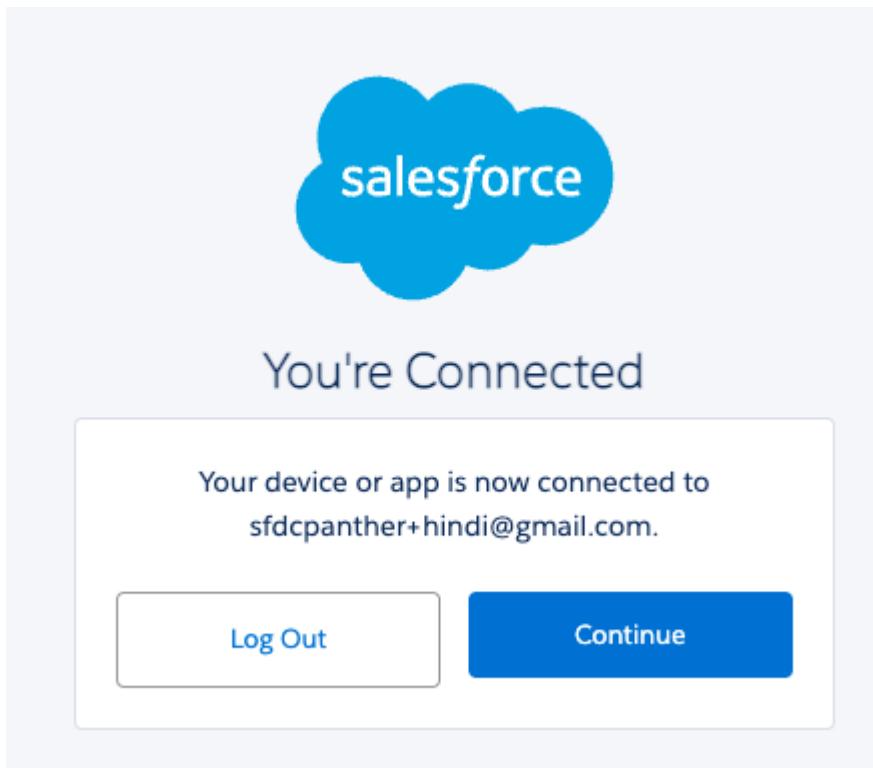
In the very first step, you will get the verification_uri & the user_code, the user needs to navigate to that URI and then provide the code. Below is the Sample URL -

<https://login.salesforce.com/setup/connect>

You will see the screen below



Once the user provides the device code and the server has validated the device code you will see the screen below



Step#3 - Get the Access Token

Once the user has approved the access, now the IOT device is ready to make the access token call. To get the access token device will request the **/token** endpoint

Token Endpoint - <https://login.salesforce.com/services/oauth2/token>

Content-Type: application/x-www-form-urlencoded

Request body -

grant_type=devide&client_id=YOUR_CLIENT_ID&code=THE_DEVIDE_CODE_FOR_M_SETP1

The screenshot shows a Postman collection named "oAuth 2.0 Flows / Device Requests Authorization / Step2 - Get Access Token". The request is a POST to {{token_url}}. The "Body" tab is selected, showing "form-data" with three fields: "grant_type" (value: device), "client_id" (value: {{client_id}}), and "code" (value: {{device_code}}). A red box highlights the "x-www-form-urlencoded" option under "Body" and another red box highlights the "Status: 200 OK" message at the bottom.

OAuth 2.0 JWT Bearer Flow for Server-to-Server Integration

In the context of the OAuth 2.0 JWT bearer token flow, the client initiates the process by sending a signed JWT to the Salesforce OAuth token endpoint.

JWT Structure

- Headers** – These contain the algorithm that will be used to sign the request `{"alg": "RS256"}`
- Payload** – This contains claims information which is an object containing information about the user and additional data. Claims are set using the parameters- `{"Iss, aud, sub, exp"}`
- Signature** – The signature consists of 3 parts and the structure is

given below

```
<headerbase64encodedurl>.  
<claimsbase64encodedclaims>.  
<signature(uses an algorithm like RS 256)>
```

Salesforce receives the JWT and validates the digital signature to confirm the token's authenticity.

If the application is authorized, Salesforce then issues an access token to the client, allowing them to access protected resources on behalf of the user.

Prerequisites

- [Create a Private Key and Self-Signed Digital Certificate](#)
- [Create a Connected App in Your Org](#) - For JWT token purposes we will use a completely different Connected Application that will use a digital signature.

Steps involved

- Create a JWT
 - JWT header
 - JWT Claims
- Combine the JWT Header and JWT claims
- Get the private Key
- Sign & Base64UrlEncode the Private Key
- Get the access token
- Access Protected Resources

→ Create a JWT Header

- Create a JWT header - `{"alg": "RS256"}`

- Base64url encodes the JWT header and the result should be equal to
eyJhbGciOiJSUzI1NiJ9

→ Create the JWT Claims

to construct the JWT claims we need the following parameters and prepare the JSON

- **iss** - The issuer and it will contain the Client ID
- **aud** - The login URL of your salesforce org.
- **sub** - the Salesforce username
- **exp** - The current date time in UNIX format and add the minutes when you wanted to expire the token

Here is the example JSON file

```
{  
  "iss": "3MVG990xTyEMCQ3gNp2PjkqeZKxnmAiG1xV4oHh9AKL_rSK.",  
  "sub": "my@email.com",  
  "aud": "https://login.salesforce.com",  
  "exp": "1333685628"  
}
```

Now do the base64UrlEncode to the JWT claims data.

→ Combine the JWT header and Claims

Now, combine the JWT header & claims and store them in a variable. The concatenation should be with a dot (.) in between the header and claims like below

eyJhbGciOiJSUzI1NiJ9.**eyJzdWliOiJzZmRjcGFudGhlcitoaW5kaUBnbWFpbC5jb20iLCJpc3MiOilzTVZHOU5fSHZFVEdocjNDT0dTQloyUkd3MFVDdGhuZG1EY1drbGJkSGRuMDhMMElkWszZ01wOFA3YzZpTnNUbmhNXzYxWnV2UDNyckFxT2Nw**

SGk1liwiZXhwIjoxNjgyMDgxOTA5NTIwLCJhdWQiOiJodHRwczovL2xvZ2luLnNh
bGVzZm9yY2UuY29tIn0

→ Get the Private Key

we have already created the private key “**server.key**” as part of the prerequisite, upload the file in your salesforce org so that we can use it in Apex Class. To use in Postman we will open the key file in a notepad and can use the value.

→ Sign & Base64UrlEncode the Private Key

The private key must need to be signed using any of the supported algorithms however the recommendation is to use **rsa-sha256** for signing the key. You can use crypto methods of Salesforce Apex Class and some online tools like jwt.io for testing purposes.

→ Append the **Base64UrlEncoded** key with jwt header & claim and use dot(.) as a separator

Below is the complete assertion that will look like. You will see the clear differentiation in the colours for

- header - `eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9`
- claims -
`eyJpc3MiOilzTVZHOU5fSHZFVEdocjNDT0dTQloyUkd3MFVDdGhuZG1EY1
drbGJkSGRuMDhMMElkWszZ01wOFA3YzZpTnNUbmhNXzYxWnV2UDNy
ckFxT2NwSGk1liwic3Viljoic2ZkY3BhbnRoZXIraGluZGlAZ21haWwuY29tliw
iYXVkljoiaHR0cHM6Ly9sb2dpbi5zYWxlczVcmNlLmNvbSIsImV4cCI6MTY
4Mja4NTM3OTM1MH0`
- signature -
`ZWd7CnYg3—pZKjb44TrCtKc4UCTh1`GHb0kOTWOPmZsjl6Lya7_GXa7-g
1ULTb05ckLVYURuP7zURUpLkEjNqlILSlYI6K20v6wgpFdziXbkAYznJRgCX`

Lt6Vz_pwPeK6Z24I9xtuSefuR0LPDNkrKF80npTP6m_XFpYPfl5jf1yO7SEiv
rB8l1IJUstunEGjtQ5mIOmm_9oSiJbGUHFq9TlbdMLnDabZks7u-K3l5fxb-
dam28gz8w1N6K4rzVYSppZwbgNXaljz9sAMn-8P6dRYwTFy4TKlU62ePN
x18uB8Jq3gvjLSRqALcyk4evCRQpePn0IKVunVXBA36tg

eyJhbGciOiJSUzI1NilsInR5cCI6IkpxVCJ9.eyJpc3MiOiIzTVZHOU5fSHZFVEdocjN
DT0dTQloyUkd3MFVDdGhuZG1EY1drbGJkSGRuMDhMMElkWszZ01wOFA3Yzz
pTnNUbmhNXzYxWnV2UDNyckFxT2NwSGk1liwic3Viljoic2ZkY3BhbnRoZXIraGlu
ZGlAZ21haWwuY29tliwiYXVkljoiaHR0cHM6Ly9sb2dpbi5zYWxlc2ZvcmNlLmNvb
SIslmV4cCI6MTY4MjA4NTM3OTM1MH0.ZWd7ChYg3—pZKjb44TrCtKc4UCTh1`G
Hb0kOTWOPmZsjl6Lyae7_GXa7-g1ULTb05ckLVYURuP7zURUpLkEjNqlILSlYI6K
20v6wgpFdziXbkAYznJRGcXLt6Vz_pwPeK6Z24I9xtuSefuR0LPDNkrKF80npTP6
m_XFpYPfl5jf1yO7SEivrB8l1IJUstunEGjtQ5mIOmm_9oSiJbGUHFq9TlbdMLnDab
Zks7u-K3l5fxb-dam28gz8w1N6K4rzVYSppZwbgNXaljz9sAMn-8P6dRYwTFy4T
KlU62ePNx18uB8Jq3gvjLSRqALcyk4evCRQpePn0IKVunVXBA36tg

Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiIzTVZH0W5fSHZFVEdocjNDT0dTQloyUkd3MFVDDgHuZG1EY1drbGJKSGRuMDhMME1kbWszZ01wOFA3YzZpTnNUbmhNXzYxWnV2UDNyckFxT2NwSGk1Iiwc3ViIjoic2ZkY3BhbnRoZXIraGluZGLAZ21haWwuY29tIiwiYXVkIjoiaHR0cHM6Ly9sb2dpbi5zYWxlclZvcmlNlLmNvbSISImV4cCI6MTY4MjA4NTM3OTM1MH0.ZWd7CnYg3--pZKjb44TrCtKc4UCh1`GHb0kOTWOPmZsj16Lya_e7_GXa7-g1ULTb05ckLVYURuP7zURUpLkEjNql1ILS1YI6K20v6wgpFdziXbkAYznJrgCXLt6Vz_pwPeK6Z24I9xtuSefuR0LPDNkrKF80npTP6m_XFpYPf15jf1y07SEivrB8l1IJUstunEGjtQ5mIOMM_9oSiJbGUHFq9T1bdMLnDabZks7u-K315fxb-dam28gz8w1N6K4rzVYSppZwbgNXaljz9sAMn-8P6dRYwTFy4TK1U62ePNx18uB8Jq3gvjLSRqALcyk4evCRQpePn0IKVunVXBAA36tg
```

Warning: Looks like your JWT signature is not encoded correctly using base64url (<https://tools.ietf.org/html/rfc4648#section-5>). Note that padding ("=") must be omitted as per <https://tools.ietf.org/html/rfc7515#section-2>

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "RS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "iss": "MVG9N_HvETGhr3COGSB2RGw0UCthndm0cWk1bdHdn08L0Idmk3gMp8P7c6iNsTnhM_61ZuvP3rrAq0cpHi5",
  "sub": "sfdcpanther+hindi@gmail.com",
  "aud": "https://login.salesforce.com",
  "exp": 1682085379350
}
```

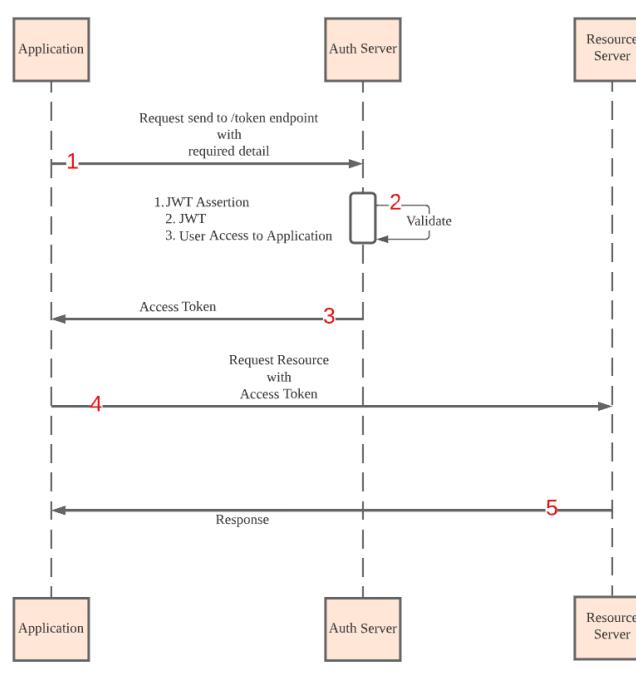
VERIFY SIGNATURE

```
RSASHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  Public Key in SPKI, PKCS #1,
  X.509 Certificate, or JWK string format.

-----BEGIN RSA PRIVATE KEY-----
...
MIIEpaIBAAKCAQEaoi7/+mBiouquiI6L5w2+1cz8nq1x1z8pf609vLBQHvSYnhG+
)
```

[OAuth 2.0 JWT Bearer Flow for Server-to-Server Integration]

Amit Singh | April 21, 2023



→ Get the access token

The Token endpoint will look like the below for the Salesforce Org.

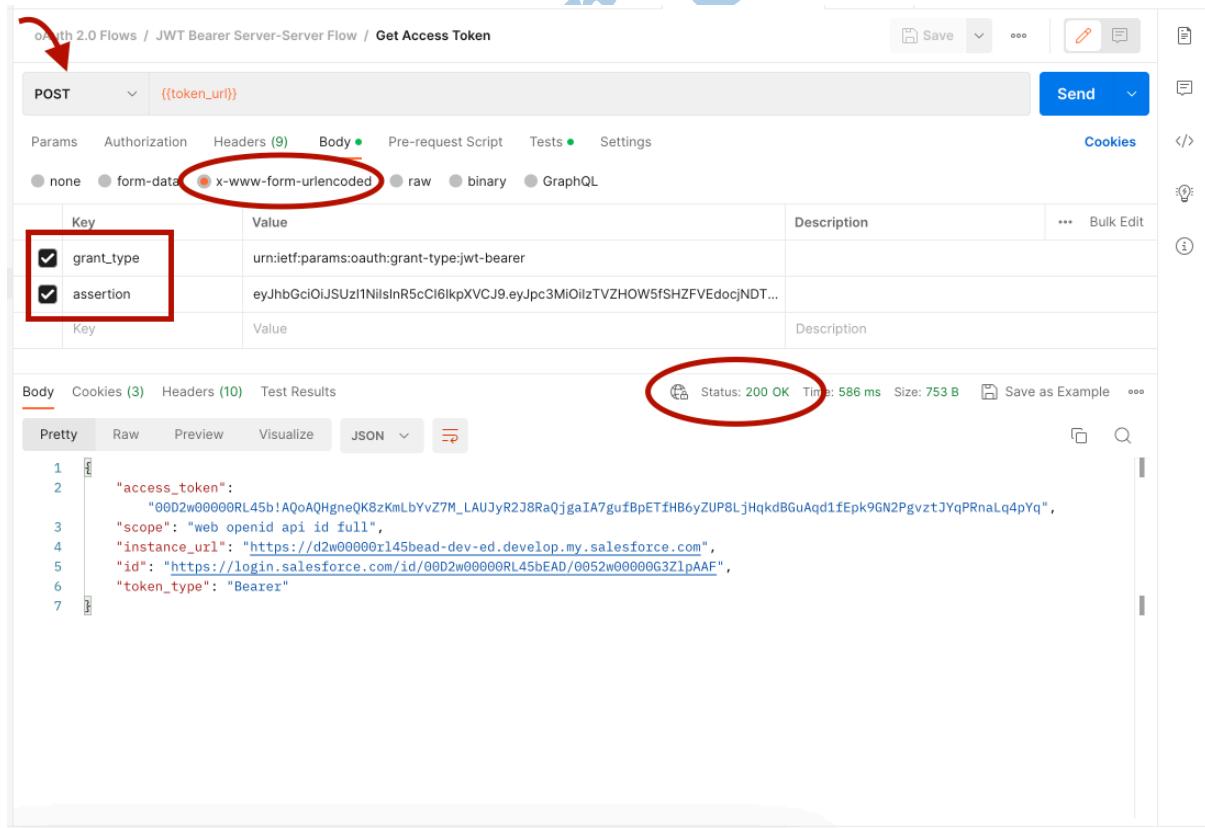
- For Production or Developer Org -
<https://login.salesforce.com/services/oauth2/token>
- For Sandbox OR Scratch ORG -
<https://test.salesforce.com/services/oauth2/token>

The content type must be application/x-www-form-urlencoded

The body would be

grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion=COMPLETETE_JWT_ASSERTION

Here is the sample request from Postman



The screenshot shows a Postman request for an access token. The URL is {{token_url}}. The method is POST. The content type is set to x-www-form-urlencoded. The body contains two parameters: grant_type (set to urn:ietf:params:oauth:grant-type:jwt-bearer) and assertion (set to a long JWT string). The response status is 200 OK, and the response body is a JSON object containing the access token, scope, instance URL, id, and token type.

Key	Value	Description	... Bulk Edit
grant_type	urn:ietf:params:oauth:grant-type:jwt-bearer		
assertion	eyJhbGciOiJSUzI1NlslnR5cCl6IkpxVCJ9eyJpc3MiOilzTVZHOU5fSHZFVEdocjNDT...		

Body Status: 200 OK Time: 586 ms Size: 753 B Save as Example

```
1 "access_token":  
2 "00D2w000000RL45b!AQoAQHgneQK8zM_LAUJyR2J8RaQjgaIA7gufBpETfHB6yZUP8LjHqkdBGuAqd1fEpk9GN2PgvztJYqPRnaLq4pYq",  
3 "scope": "web openid api id full",  
4 "instance_url": "https://d2w00000rl45bead-dev-ed.my.salesforce.com",  
5 "id": "https://login.salesforce.com/id/00D2w00000RL45bEAD/0052w00000G3ZlpAAF",  
6 "token_type": "Bearer"  
7
```

Here is the complete code for the JWT using Apex

```
public class JWT {

    public String iss      { get; set; }
    public String sub      { get; set; }
    public String aud      { get; set; }
    public Long exp       { get; set; }

    public JWT(String iss, String sub, String aud, Long exp){
        this.iss = iss;
        this.sub = sub;
        this.aud = aud;
        this.exp = exp;
    }

    private static final String JWT_HEADER = '{"alg":"RS256"}';

    public static String retrievePrivateKey(){
        ContentVersion base64Content = [SELECT VersionData FROM
ContentVersion where Title='server' ORDER BY Title LIMIT 1];
        String keyContents = base64Content.VersionData.toString();
        keyContents = keyContents.replace('-----BEGIN RSA PRIVATE
KEY-----', '');
        keyContents = keyContents.replace('-----END RSA PRIVATE
KEY-----', '');
        keyContents = keyContents.replace('\n', '');
        return keyContents;
    }

    public static void run(){}
```

```

        DateTime rightNow = System.now();
        rightNow = rightNow.addMinutes(3);
        Long expiry = rightNow.getTime();

        JWT jwt = new JWT(
            '3MVG9n_HvETGhr3C0GSBZ2RGw0UCthndmDcWk1bdHdn08L0Idmk3gMp8P7c6iNsTn
            hM_61ZuvP3rrAqOcpHi5',
            'sfdcpanther+hindi@gmail.com', 'https://login.salesforce.com',
            expiry
        );

        String base64UrlJWTHeader = base64UrlEncode( JWT_HEADER );
        String body = JSON.serialize(jwt);
        String base64UrlJWTClaims = base64UrlEncode( body );

        String combinedHeaderClaim =
base64UrlJWTHeader+'.'+base64UrlJWTClaims;
        System.debug('combinedHeaderClaim \n
'+combinedHeaderClaim);

        String jwtAssertion =
shaSignPrivateKey(combinedHeaderClaim);
        System.debug('jwtstr \n '+ jwtAssertion);

        getAccessToken(jwtAssertion);

    }

    public static String shaSignPrivateKey(String
combinedHeaderClaim){

```

```
// base64Decode the Private key Content
Blob privateKey = EncodingUtil.base64Decode(
retrievePrivateKey() );

// Sign the private Key using rsa-sha256 Algorithm
Blob signature = Crypto.sign('rsa-sha256',
Blob.valueOf(combinedHeaderClaim), privateKey);

// Add the signature into the assertion
combinedHeaderClaim += '.' +
SFDC_BASE64_URLENCODE(signature);

return combinedHeaderClaim;
}

public static String base64UrlEncode(String header){
String base64UrlEncode = SFDC_BASE64_URLENCODE(
Blob.valueOf(header) );
return base64UrlEncode;
}

private static String SFDC_BASE64_URLENCODE(final Blob input){
if(input == null) {
return null;
}
return EncodingUtil.base64Encode(input)
.replace('/', '_')
.replace('+', '-')
.replaceAll('=+$', '');
}

private static void getAccessToken(String assertionValue){
HttpRequest req = new HttpRequest();

req.setEndpoint('https://login.salesforce.com/services/oauth2/toke
```

```

n?grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion
= '+assertionValue);

    req.setMethod('POST');

    req.setHeader('Content-Type',
'application/x-www-form-urlencoded');

    Http http = new Http();
    HTTPResponse res = http.send(req);
    System.debug(res.getBody());
    String response = res.getBody();
}

}

```

Useful JSON Class methods

There are various methods of JSON class in Salesforce that we would be using frequently throughout the course.

- **serialize**(Object o)
- **deserialize**(String str)
- **serializePretty**(Object object)
- **deserializeUntyped**(jsonString)

Note:- You can get the list of all the -

https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apexref/apex_class_System_Json.htm

```

/**
 * @description      :
 * @author           : Amit Singh - PantherSchools
 * @group            :

```

```
* @last modified on : 03-17-2024
* @last modified by : Amit Singh - PantherSchools
*/
public with sharing class PS_ZendeskTicketInputWrapper {

    public ticket ticket;

    public class ticket {
        public String subject;
        public String priority;
        public String type;
        public comment comment;
        public String assignee_id;
        public requester requester;
    }
    public class comment {
        public String body;
    }

    public class requester {
        public Integer locale_id;
        public String name;
        public String email;
    }
}
```

```
PS_ZendeskTicketInputWrapper wrapper = new
PS_ZendeskTicketInputWrapper();
```

```
PS_ZendeskTiketInputWrapper.requester requester = new  
PS_ZendeskTiketInputWrapper.requester();  
requester.locale_id = 8;  
requester.Name = 'Amit Sng';  
requester.Email = 'asingh@zmail.com';  
  
PS_ZendeskTiketInputWrapper.comment comment = new  
PS_ZendeskTiketInputWrapper.comment();  
comment.body = 'JSON Serialize Demo';  
  
PS_ZendeskTiketInputWrapper.ticket ticket = new  
PS_ZendeskTiketInputWrapper.ticket();  
ticket.subject = 'JSON Serialize Demo';  
ticket.priority = 'high';  
ticket.type = 'incident';  
//ticket.assignee_id = '';  
ticket.comment = comment;  
ticket.requester = requester;  
  
wrapper.ticket = ticket;  
  
String requestBody = System.JSON.serialize(wrapper);  
System.debug(requestBody);
```



```
PS_ZendeskTiketInputWrapper wrapper = new PS_ZendeskTiketInputWrapper();  
  
PS_ZendeskTiketInputWrapper.requester requester = new  
PS_ZendeskTiketInputWrapper.requester();  
requester.locale_id = 8;  
requester.Name = 'Amit Sng';  
requester.Email = 'asingh@zmail.com';
```

```
PS_ZendeskTiketInputWrapper.comment comment = new  
PS_ZendeskTiketInputWrapper.comment();  
comment.body = 'JSON Serialize Demo';  
  
PS_ZendeskTiketInputWrapper.ticket ticket = new  
PS_ZendeskTiketInputWrapper.ticket();  
ticket.subject = 'JSON Serialize Demo';  
ticket.priority = 'high';  
ticket.type = 'incident';  
//ticket.assignee_id = '';  
ticket.comment = comment;  
ticket.requester = requester;  
  
wrapper.ticket = ticket;  
  
String requestBody = System.JSON.serializePretty(wrapper);  
System.debug(requestBody);
```

```
PS_ZendeskTiketInputWrapper wrapper = new PS_ZendeskTiketInputWrapper();  
  
PS_ZendeskTiketInputWrapper.requester requester = new  
PS_ZendeskTiketInputWrapper.requester();  
requester.locale_id = 8;  
requester.Name = 'Amit Singh';  
requester.Email = 'asingh@zmail.com';  
  
PS_ZendeskTiketInputWrapper.comment comment = new  
PS_ZendeskTiketInputWrapper.comment();  
comment.body = 'JSON Serialize Demo';  
  
PS_ZendeskTiketInputWrapper.ticket ticket = new  
PS_ZendeskTiketInputWrapper.ticket();  
ticket.subject = 'JSON Serialize Demo';
```

```

ticket.priority = 'high';
ticket.type      = 'incident';
//ticket.assignee_id = '';
ticket.comment = comment;
ticket.requester = requester;

wrapper.ticket = ticket;

String requestBody = System.JSON.serializePretty(wrapper, true);
System.debug(requestBody);

```

What is Apex REST

- Introduction to Apex Rest
- How to Expose Apex Class as REST or SOAP Web Service
- RestContext Class
- Authentication Mechanism

Introduction to Apex Rest

- Apex Rest is used to expose the Salesforce Apex Classes as web services so that the external application can communicate with Salesforce and exchange information.
- If the standard APIs are not the best fit for the requirement then only we will go for the custom web services. For **Example** - getting the data of multiple objects into a single call OR getting the data from an Order Management system and processing/creating the Order within Salesforce

```
{
  "customer": {
    "Name": "",
    "Email": ""
  }
}
```

```
"CustomerNumber": "",  
    "Phone": "",  
},  
    "BillingAddress": {  
  
},  
    "ShippingAddress": {  
  
},  
    "orderHeader": {  
        "totalAmount": "",  
        "orderNumber": "",  
        "currency": "",  
        "tax": "",  
        "quantity": "",  
        "headerLines": {  
            "productName": "",  
            "productCode": "",  
            "quantity": "",  
            "unitPrice": "",  
            "sellingPrice": ""  
        }  
    }  
}
```

How to Expose Apex Class as REST or SOAP Web Service

```
@RestResource(urlmapping='/v1/Account/*')  
global class AccountManager {  
  
}
```

Methods that can be used with Apex Rest

- **Get @httpGet** - Method used to get the information from Salesforce
- **Post @HttpPost** - Method used to create the records inside Salesforce using the information received as a Body from the external application
- **Patch @HttpPatch** - When doing a partial update, they can use HTTP PATCH.
- **Put @HttpPut** - When there is a need to replace an existing Resource entirely, can use PUT
- **Delete @HttpDelete** - When you want to delete a resource from salesforce.

RestContext Class

RestContext Class

is the main class that contains information about both the request and response to the request. This class contains the instance of the RestRequest & RestResponse classes.

RestRequest Class

is a class that contains information about the request like headers, method, request body, parameters requestURL, etc. **RestRequest response = RestContext.request**

RestResponse Class contains information about the response like the response body, header, and status code. **RestResponse response = RestContext.response**

Authentication Mechanism

The Apex rest web services can be consumed using any of the below authentications

- OAuth 2.0 Flows
- Session Id
- Public Web Services

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 03-17-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
public with sharing class PS_AccountService {  
  
    public static void formatString(String stringToFormat){  
        String formattedString = stringToFormat.toString();  
    }  
}
```

```
/**  
 * @description      :  
 */
```

```

* @author : Amit Singh - PantherSchools
* @group :
* @last modified on : 03-17-2024
* @last modified by : Amit Singh - PantherSchools
**/


/**
* Org Base Url - Instance Url +
services/apexrest/v1/PS_AccountManager/
*
https://integration-org2-dev-ed.develop.my.salesforce.com/services
/apexrest/v1/PS_AccountManager/
*/
@RestResource(urlMapping = '/v1/PS_AccountManager/*')
global with sharing class PS_AccountManager {

/**
* @httpGet - GET
* @httpPost - POST
* @httpPut - PUT
* @httpPatch - PATCH
* @delete - DELETE
*/



/** List all the Salesforce Accounts - GET */
@httpGet
global static List<Account> getAccountList(){

    RestRequest req = RestContext.request;
    System.debug(System.JSON.serializePretty(req));

    RestResponse res = RestContext.response;

```

```
System.debug(System.JSON.serializePretty(res));

    return [SELECT Id, Name, Industry, Phone, Fax FROM Account
LIMIT 10];

}

/** Create an Account - POST */

@httpPost
global static Account createAccount(){ // method arguments

    /** Get the information about Request */
    RestRequest req = RestContext.request;
    System.debug(System.JSON.serializePretty(req));

    /** get the request body */
    String requeBody = req.requestBody?.toString(); // Convert
Blob into String format
    System.debug( requeBody );

    RestResponse res = RestContext.response;
    System.debug(System.JSON.serializePretty(res));
    //PS_AccountService.formatString(null);

    if(String.isBlank(requeBody)){
        res.statusCode = 400;
        res.responseBody = Blob.valueOf('{"message": "Request
Body can not be blank!"}');
        //return res;
    }
}
```

```

        Account acc = new Account();
        acc.Name      = UUID.randomUUID().toString();
        acc.Phone     = '98876655623';

        return acc;
    }
}

```

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 03-17-2024
 * @last modified by : Amit Singh - PantherSchools
 */

@RestResource(urlMapping = '/v1/PS_AccountManager/*/Contact')
global with sharing class PS_ContactManager {

    @httpGet
    global static List<Contact> listContacts(){
        return [SELECT Id, Name, Email, Phone FROM Contact LIMIT
10];
    }
}

```

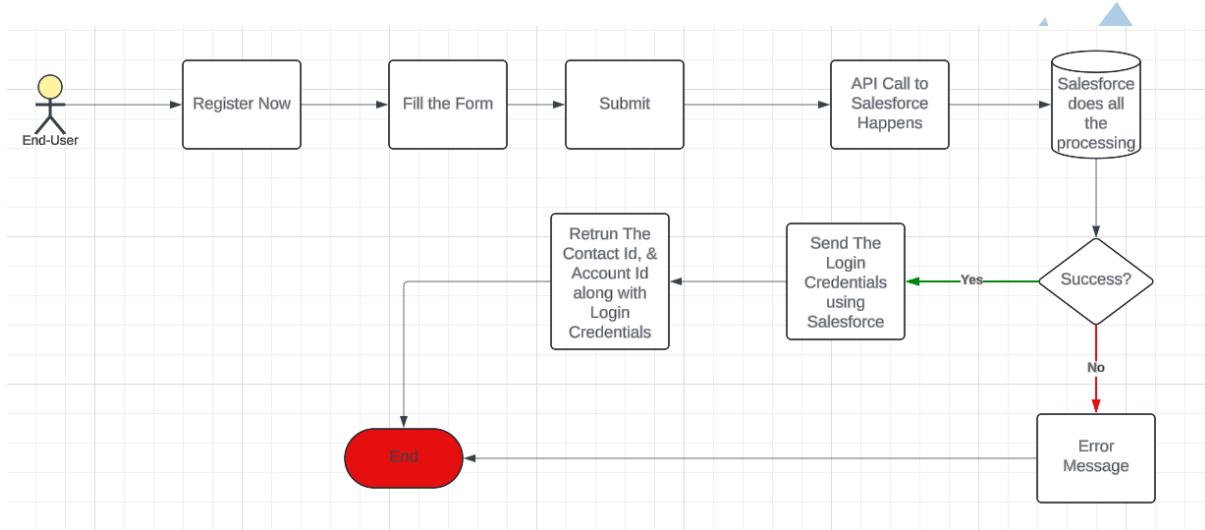
A Complete Scenario

Panther Schools has a web portal where users can do the below activities

- **Register themself and enjoy the free & paid courses** - Once the registration happens the information is sent to Salesforce and

Account & Contacts are Created inside Salesforce. Also, the credentials are being emailed to the subscribers.

- **Note:** - The Contact ID & Salesforce Username are being returned to PantherSchools so that they can use them for reference purposes.
- Hint:- @HttpPost



- **Login** - The user can log in and enjoy the free courses or paid courses that they have subscribed for and also scroll through the existing courses.
 - The web portal will use its login mechanism
- **Register For the Course** - When the User registers for the course the details about the courses like which course has been registered, what the price, the status of the payment, payment information, and then the Contact ID of the user is sent to Salesforce so that Salesforce can create the appropriate record and return success or error message.
 - Use @HttpPost to send the data
 - Booking → Contact
 - Payment Record
 - Booking Confirmation Email
 - Email Template
- **Delete Account** - The user can also delete his/her profile which will

also lead to cleaning up the data from Salesforce.

- @HttpDelete
- **Update Profile** - If the member/subscriber updates the information in the web portal it should flow to Salesforce for updating the information. Assume anything that is missing.
 - @HttpPatch
 - Contact ID will be provided in the URL
- **View Profile** - The information that is being displayed on the My Profile page in the web portal will eventually come from Salesforce
 - @HttpGet
 - Contact ID will be provided in the URL
- **Course Detail Page** - The information about the course like description, instructor, start & end date, and much more information will also come from Salesforce where the web portal is going to make the API call and get the relevant information.
 - @HttpGet
 - Course Object
 - Scheduled Session
 - Instructor
 - Location Course
 - Course Images

Write your First Apex Rest API & Test using Workbench

Use Case - Send a welcome message to the external application that is trying to consume our web service. Once the web service is developed we will test it using Workbench.

Get Method Using Apex Rest @HttpGet

Use Case View Profile - The Panther schools want to give their subscribers the ability to view their profile while they are logged into the portal and can also see the relevant information like Billing OR Shipping Address.

The web developer is responsible for developing the front end and as a developer, **you need to expose a web service that will accept the contactId in the URL parameters** and send the information back to the web portal in the JSON format.

Below is the information that needs to be returned to the portal application

- **Contact** - First Name, Last Name, Email, Phone, Title, Mailing Address
- **Account** - Name, Industry, Rating, Billing Address, Shipping Address
- **Cases** - CaseNumber, Subject, Status, Description, Priority, CreatedDate, CloseDate

Test Your API using the Postman API Testing Tool

What all we need to test the API using Postman

1. Access Token
2. URL Endpoint to the Apex REST Service
3. Postman Installed in your system

Post Method Using Apex Rest @HTTPPost

Use Case - Register the member in the web portal

The web portal needs to send the information to Salesforce using the Custom Apex REST API. As a developer you need to develop a custom Web Service that accepts the below information, creates the account & contact, and returns the contact Id as a response.

- First Name
- Last Email
- Phone
- Title
- Account Name
- Industry
- Rating
- Address

If there is any error while creating the records in Salesforce then return the error message in JSON format.

Delete Method using Apex Rest @HttpDelete

Use Case - Delete the customer account/contact from Salesforce

As the PantherSchools web portal can let the members delete their accounts so once the members delete their account from the web portal the portal does an API call to Salesforce and the Account, Contact & Cases related to that Contact should get deleted from Salesforce.

As a Salesforce developer, you have been asked to develop a custom Apex REST web service that accepts the contactId in the URL params and deletes the relevant information.

Patch Method using Apex Rest @HTTPPatch

Use Case - Update Profile

Whenever the subscribers/members of the web portal change their profile the portal will make an API call to update the data in Salesforce. Your web service should accept the required attributes and update the information in Salesforce.

Assignments

- **Register Course** - When any member is registering for the course the portal will make an API call and will send the necessary information like course ID, contact ID, Amount, and the status of the payment. Your API should accept all these, create a booking within Salesforce and return either a success or an error response.
- Course Detail Page - The portal wants to fetch the details about the course from the salesforce directly so that they are up to date with the course details. Develop a web service to return the course information based on the provided course ID. The response should include the following but not be limited to
 - Course Title
 - Fee Structure
 - Start & End Date
 - Description
 - Instructor Details
 - Type of Course (In-Person or virtual)
 - If it is an in-person training then return to the Physical Address Location for the training
 - Available Sheets

Q- Can we have multiple methods with the same @Http? (SB - Question for students)

→ Enhancement

After the successful rollout of the first page, Panther Schools wants to enhance the portal to have the below functionality and you have been asked to develop the custom rest APIs for the following

- **Submit Case** - The member of the portal can submit the cases if they are having any issues from the portal itself and the information would go to Salesforce directly and return the case number. Also, an email should be sent to the customer with the relevant information. Below is the information that your web service should accept
 - Subject
 - Description
 - Case Reason
 - ContactId
- **Case Details** - The web portal members should be able to see the complete details about the case including but not limited to
 - Case Number
 - Subject
 - Status
 - Priority
 - Description
 - Created Date
 - Close Date
- **FAQ** - The management wants to have a FAQ page in the portal and the information would come from Salesforce. The Salesforce admin has already set up the Knowledge Articles and the articles are published. As a developer you need to develop a web service that will return all the Knowledge articles that are published.
- **Course List Page** - Currently the web portal is using its database to display the list of the courses the business has noticed that there is some despondency in the data and they are complaining that the data

is not up to date. So business has decided to go with that now the courses will be fetched from Salesforce and then will get listed in the portal.

- As a developer, you need to develop a RESTful service in Salesforce that should return the list of all the active courses and the start date is greater than today.

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 03-31-2024  
 * @last modified by : Amit Singh - PantherSchools  
 ***/  
  
/**  
 * Org Base Url - Instance Url +  
 services/apexrest/v1/PS_ContactManager/  
*  
https://integration-org2-dev-ed.develop.my.salesforce.com/services  
/apexrest/v1/PS_ContactManager/  
**/  
  
@RestResource(urlMapping = '/v1/PS_ContactManager/*')  
global with sharing class PS_ContactManager {  
  
    @httpGet  
    global static List<Contact> listContacts(){  
        return [SELECT Id, Name, Email, Phone FROM Contact LIMIT  
10];  
    }  
}
```

```

/*
For Account Record -
    Name,
    Industry, Rating
    AnnualRevenue, Phone

For Contact Record -
    FirstName, LastName, Email, Title, Phone
*/

@httpPost
global static ResponseWrapper processData(InputWrapper input){
    List<Contact> existingContacts = [SELECT Id, FirstName,
    LastName, Account.Name
                                         FROM
                                         Contact
                                         WHERE
                                         Email =:
    input.email
                                         ];
}

ResponseWrapper wrapper = new ResponseWrapper();

if(existingContacts?.size()>0){
    // do nothing
    wrapper.contactRecord = existingContacts.get(0);
    wrapper.status = 'Existing';
    wrapper.message = 'Contact Already Exists with the Same
Email';
}else{
    Account accountRecord = new Account(Name =

```

```
    input.accountName);
        insert accountRecord;

    Contact contactRecord = new Contact(
        FirstName = input.firstName,
        LastName = input.lastName,
        Email = input.email
    );
    contactRecord.AccountId = accountRecord.Id;

    insert contactRecord;

    wrapper.contactRecord = contactRecord;
    wrapper.accountRecord = accountRecord;
    wrapper.status = 'New';
    wrapper.message = 'Data Inserted';
}

return wrapper;
}

/*
@httpPost
global static ResponseWrapper processData(Account
accountRecord, Contact contactRecord){ // method arguments
    System.debug(System.LoggingLevel.DEBUG, accountRecord);
    System.debug(System.LoggingLevel.DEBUG, contactRecord);
    List<Contact> existingContacts = [SELECT Id, FirstName,
LastName, Account.Name FROM Contact WHERE Email =:
contactRecord.Email];
    ResponseWrapper wrapper = new ResponseWrapper();
}
```

```
if(existingContacts?.size()>0){
    // do nothing
    wrapper.contactRecord = contactRecord;
    wrapper.accountRecord = accountRecord;
    wrapper.status = 'Existing';
    wrapper.message = 'Contact Already Exists with the Same
Email';

}else{
    // Create Account
    insert accountRecord;
    // Create Contact
    contactRecord.AccountId = accountRecord.Id;
    insert contactRecord;

    wrapper.contactRecord = contactRecord;
    wrapper.accountRecord = accountRecord;
    wrapper.status = 'New';
    wrapper.message = 'Data Inserted';
}

return wrapper;
}

*/
}

global class InputWrapper {
    global String accountName;
    global String firstName;
    global String lastName;
    global String email;
}

global class ResponseWrapper {
    global Account accountRecord;
```

```
    global Contact contactRecord;
    global String status; // New, Existing
    global String message; // Success, Error
}
}
```

```
/***
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           : PantherSchools
 * @last modified on : 03-31-2024
 * @last modified by : Amit Singh - PantherSchools
 **/


/**
 * Org Base Url - Instance Url +
services/apexrest/v1/PS_AccountManager/
*
https://integration-org2-dev-ed.develop.my.salesforce.com/services
/apexrest/v1/PS_AccountManager/
*/
@RestResource(urlMapping = '/v1/PS_AccountManager/*')
global with sharing class PS_AccountManager {

    /**
     * @httpGet - GET
     * @httpPost - POST
     * @httpPut - PUT
     * @httpPatch - PATCH
     * @httpDelete - DELETE
    */
}
```

```

@httpDelete
global static void deleteRecords(){
    String accountId =
RestContext.request.requestURI.substringAfterLast('/');
    Database.delete(accountId);
}

/** List all the Salesforce Accounts - GET */
// /v1/PS_AccountManager/001Hu000038SwuZIAW/Contacts

@httpGet
global static Account getAccountList(){

    RestRequest req = RestContext.request;
    System.debug(System.JSON.serializePretty(req));

    RestResponse res = RestContext.response;
    System.debug(System.JSON.serializePretty(res));
    // /v1/PS_AccountManager/001Hu000038vjRiiAI
    /**
     * 1 - Get the Account ID from URL
     * 1.1 - get last index of /
     * 1.2 - use the substring method of your String Apex
Class
    * 1.1.1
    * - subStringAfterLast method of your String class
    * 2 - Query the Account with all the Information
    */
    String accountId = req.requestURI.substringAfterLast('/');

    Account accRecord = [SELECT
        Id, Name, Rating, Phone, Fax,

```

```
Industry, BillingAddress, ShippingAddress
    FROM
        Account
    WHERE
        Id =: accountId
    LIMIT 1
];
return accRecord;

}

/** Create an Account - POST */

@httpPost
global static Account createAccount(){ // method arguments

    /** Get the information about Request */
    RestRequest req = RestContext.request;
    System.debug(System.JSON.serializePretty(req));

    /** get the request body */
    String requeBody = req.requestBody?.toString(); // Convert
Blob into String format
    System.debug( requeBody ); // JSON Format

    RestResponse res = RestContext.response;
    System.debug(System.JSON.serializePretty(res));

    if(String.isBlank(requeBody)){
        res.statusCode = 400;
        res.responseBody = Blob.valueOf('{"message": "Request
Body can not be blank!"}');
    }
}
```

```
//return res;  
}  
  
Account acc = new Account();  
acc.Name      = UUID.randomUUID().toString();  
acc.Phone     = '98876655623';  
  
return acc;  
}  
  
public class AccountWrapper {  
    public String id;  
    public String name;  
    public BillingAddress billigAddress;  
}  
  
public class BillingAddress {  
    public String city;  
}  
}
```

Consume 3rd Party APIs

**What are all the 3rd Party APIs that we will work on
in this course**

Below is the list of all the external applications that we will work on during this course

- OpenCage Geocoder
- Zendesk

- **Freshdesk** - Assignment
- ~~LinkedIn Assignment~~
- **QuickBooks Online**
 - <https://developer.intuit.com/app/developer/homepage>
- Google Calendar API
- Xero API -
<https://developer.xero.com/documentation/api/accounting/overview>
- JIRA API -
<https://developer.atlassian.com/server/jira/platform/rest-apis/>

OpenCage Geocoder

Opencage geocoder is a platform that provides the ability to get the address using Latitude & Longitude and vice versa.

What we will cover

- Signup for a Free [OpenCage Account](#)
- Get the API and go through with the documentation
- Test the API Using Postman

Implement Geocoder API using Apex

For the demo purpose, we will do the forward geocoding.

Steps to Integrate OpenCage geocoder API within Salesforce

- Add Remote Site Setting
- Create Custom Labels to Store the Endpoint and API Key
- Develop the Apex Class and Make the Request
- Parse the JSON response to an Apex Class
- Debug the response

Assignment

Implement the Reverse Geocoding using OpenCage Geocoder.

Scenario - For some of the Sales repo who work in the USA working hours they are used to provide the Latitude and Longitude of the account instead of the full account address. So you need to develop a solution that will take the Lat & Long as input and then update the account address fields accordingly.

Introduction to Freshdesk and test using Postman

Freshdesk is a ticketing system that is used to create cases in the platform for customer support functionality. PantherSchools is migrating from Freshdesk to Salesforce Service Cloud so they need the ability to integrate Freshdesk with Salesforce where they can send the data to Freshdesk or get the data from Freshdesk.

What we will cover

- Signup for a Free Freshdesk Account
- Read the API Documentation and learn about authentication
- Test the API Using Postman

Integrate Freshdesk with Salesforce - Part 1

Steps that we will follow to integrate QuickBooks with Salesforce

- Add Remote Site Setting

- Create a Custom Label to store the API Key & Password
- Create a Contact in Freshdesk
- Create a Ticket in Freshdesk

Integrate Freshdesk with Salesforce - Part 2

- List All the tickets from the Freshdesk to Salesforce
- List All the tickets from the Freshdesk to Salesforce related to a Contact

Assignment

- Read the API
- Test from Post for creating the contact
- Use Admin Booster JSON 2 Apex online tool to get the apex class of your JSON Request
- Now, modify the JSON String to make it Dynamic
- Create a new Lightning Web Component **fr_createFreshdeskContact**, create required inputs, and test the flow
 - Email
 - Subject
 - Priority
 - Status
 - Description

Once the user clicks on the Create ticket button it should call the Freshdesk API and if the response is successful then do the below items

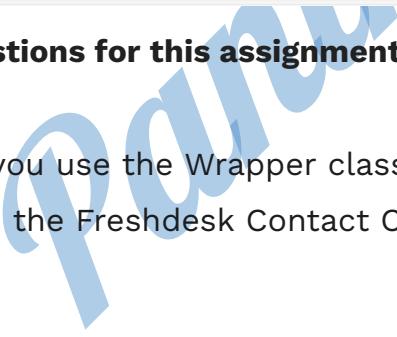
1. Create a custom object in salesforce **Name it Freshdesk Ticket and the object name should be fr_Ticket**
2. Get the fields from <https://developers.freshdesk.com/api/#tickets> and use the appropriate data type.

3. Create a **field on Contact Record Label it Freshdesk ID and api name**

fr_id

4. Create a lightning aura or web component that will take the below input and when you click on save it should create the ticket in Freshdesk and Salesforce as well. Check the mockup

Here are the field details that you can create in your custom object



Custom Object Definition Detail

		Edit Delete																																																																																																																																																													
Singular Label	Freshdesk Ticket																																																																																																																																																														
Plural Label	Freshdesk Tickets																																																																																																																																																														
Object Name	fr_Ticket																																																																																																																																																														
API Name	fr_Ticket_c																																																																																																																																																														
Description																																																																																																																																																															
Enable Reports <input type="checkbox"/>																																																																																																																																																															
Track Activities <input type="checkbox"/>																																																																																																																																																															
Allow in Chatter Groups <input type="checkbox"/>																																																																																																																																																															
Allow Sharing <input checked="" type="checkbox"/>																																																																																																																																																															
Allow Bulk API Access <input checked="" type="checkbox"/>																																																																																																																																																															
Allow Streaming API Access <input checked="" type="checkbox"/>																																																																																																																																																															
Track Field History <input type="checkbox"/>																																																																																																																																																															
Enable Licensing <input type="checkbox"/>																																																																																																																																																															
Deployment Status Deployed																																																																																																																																																															
Allow Search <input checked="" type="checkbox"/>																																																																																																																																																															
Help Settings Standard salesforce.com Help Window																																																																																																																																																															
Modified By Admin User, 10/21/2020, 10:40 AM																																																																																																																																																															
Created By Admin User, 10/21/2020, 9:27 AM																																																																																																																																																															
Standard Fields																																																																																																																																																															
<table border="1"> <thead> <tr> <th>Action</th> <th>Field Label</th> <th>Field Name</th> <th>Data Type</th> <th>Controlling Field</th> <th>Indexed</th> </tr> </thead> <tbody> <tr> <td>Edit Del</td> <td>Created By</td> <td>CreatedBy</td> <td>Lookup(User)</td> <td></td> <td></td> </tr> <tr> <td>Edit Del</td> <td>Currency</td> <td>CurrencyIsoCode</td> <td>Picklist</td> <td></td> <td></td> </tr> <tr> <td>Edit Del</td> <td>Last Modified By</td> <td>LastModifiedBy</td> <td>Lookup(User)</td> <td></td> <td></td> </tr> <tr> <td>Edit Del</td> <td>Owner</td> <td>Owner</td> <td>Lookup(User,Group)</td> <td></td> <td>✓</td> </tr> <tr> <td>Edit Del</td> <td>Ticket #</td> <td>Name</td> <td>Text(80)</td> <td></td> <td>✓</td> </tr> </tbody> </table>					Action	Field Label	Field Name	Data Type	Controlling Field	Indexed	Edit Del	Created By	CreatedBy	Lookup(User)			Edit Del	Currency	CurrencyIsoCode	Picklist			Edit Del	Last Modified By	LastModifiedBy	Lookup(User)			Edit Del	Owner	Owner	Lookup(User,Group)		✓	Edit Del	Ticket #	Name	Text(80)		✓																																																																																																																							
Action	Field Label	Field Name	Data Type	Controlling Field	Indexed																																																																																																																																																										
Edit Del	Created By	CreatedBy	Lookup(User)																																																																																																																																																												
Edit Del	Currency	CurrencyIsoCode	Picklist																																																																																																																																																												
Edit Del	Last Modified By	LastModifiedBy	Lookup(User)																																																																																																																																																												
Edit Del	Owner	Owner	Lookup(User,Group)		✓																																																																																																																																																										
Edit Del	Ticket #	Name	Text(80)		✓																																																																																																																																																										
Standard Fields Help ?																																																																																																																																																															
Custom Fields & Relationships																																																																																																																																																															
<table border="1"> <thead> <tr> <th>Action</th> <th>Field Label</th> <th>API Name</th> <th>Data Type</th> <th>Indexed</th> <th>Controlling Field</th> <th>Modified By</th> </tr> </thead> <tbody> <tr> <td>Edit Del</td> <td>Agent Id</td> <td>responder_id_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:35 AM</td> </tr> <tr> <td>Edit Del</td> <td>Company Id</td> <td>company_id_c</td> <td>Number(18, 0)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:27 AM</td> </tr> <tr> <td>Edit Del</td> <td>Created At</td> <td>created_at_c</td> <td>Date/Time</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:37 AM</td> </tr> <tr> <td>Edit Del</td> <td>Customer Id</td> <td>requester_id_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:34 AM</td> </tr> <tr> <td>Edit Del</td> <td>Customer Name</td> <td>name_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:32 AM</td> </tr> <tr> <td>Edit Del</td> <td>Customer Phone</td> <td>phone_c</td> <td>Phone</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:33 AM</td> </tr> <tr> <td>Edit Del</td> <td>Deleted</td> <td>deleted_c</td> <td>Checkbox</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:28 AM</td> </tr> <tr> <td>Edit Del</td> <td>Description</td> <td>description_c</td> <td>Rich Text Area(32768)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:28 AM</td> </tr> <tr> <td>Edit Del</td> <td>Due By</td> <td>due_by_c</td> <td>Date/Time</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:29 AM</td> </tr> <tr> <td>Edit Del</td> <td>Due in Freshdesk</td> <td>fr_due_by_c</td> <td>Date/Time</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:30 AM</td> </tr> <tr> <td>Edit Del</td> <td>Email</td> <td>email_c</td> <td>Email</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:29 AM</td> </tr> <tr> <td>Edit Del</td> <td>Escalated</td> <td>fr_escalated_c</td> <td>Checkbox</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:31 AM</td> </tr> <tr> <td>Edit Del</td> <td>Facebook Id</td> <td>facebook_id_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:30 AM</td> </tr> <tr> <td>Edit Del</td> <td>Priority</td> <td>Priority_c</td> <td>Number(10, 0)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:33 AM</td> </tr> <tr> <td>Edit Del</td> <td>Source</td> <td>Source_c</td> <td>Number(10, 0)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:35 AM</td> </tr> <tr> <td>Edit Del</td> <td>Status</td> <td>Status_c</td> <td>Number(10, 0)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:36 AM</td> </tr> <tr> <td>Edit Del</td> <td>Subject</td> <td>Subject_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:36 AM</td> </tr> <tr> <td>Edit Del</td> <td>Ticket Id</td> <td>id_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:31 AM</td> </tr> <tr> <td>Edit Del</td> <td>Twitter Id</td> <td>twitter_id_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:38 AM</td> </tr> <tr> <td>Edit Del</td> <td>Type</td> <td>Type_c</td> <td>Text(255)</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:37 AM</td> </tr> <tr> <td>Edit Del</td> <td>Updated At</td> <td>updated_at_c</td> <td>Date/Time</td> <td></td> <td></td> <td>Admin User, 10/21/2020, 9:38 AM</td> </tr> </tbody> </table>						Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By	Edit Del	Agent Id	responder_id_c	Text(255)			Admin User, 10/21/2020, 9:35 AM	Edit Del	Company Id	company_id_c	Number(18, 0)			Admin User, 10/21/2020, 9:27 AM	Edit Del	Created At	created_at_c	Date/Time			Admin User, 10/21/2020, 9:37 AM	Edit Del	Customer Id	requester_id_c	Text(255)			Admin User, 10/21/2020, 9:34 AM	Edit Del	Customer Name	name_c	Text(255)			Admin User, 10/21/2020, 9:32 AM	Edit Del	Customer Phone	phone_c	Phone			Admin User, 10/21/2020, 9:33 AM	Edit Del	Deleted	deleted_c	Checkbox			Admin User, 10/21/2020, 9:28 AM	Edit Del	Description	description_c	Rich Text Area(32768)			Admin User, 10/21/2020, 9:28 AM	Edit Del	Due By	due_by_c	Date/Time			Admin User, 10/21/2020, 9:29 AM	Edit Del	Due in Freshdesk	fr_due_by_c	Date/Time			Admin User, 10/21/2020, 9:30 AM	Edit Del	Email	email_c	Email			Admin User, 10/21/2020, 9:29 AM	Edit Del	Escalated	fr_escalated_c	Checkbox			Admin User, 10/21/2020, 9:31 AM	Edit Del	Facebook Id	facebook_id_c	Text(255)			Admin User, 10/21/2020, 9:30 AM	Edit Del	Priority	Priority_c	Number(10, 0)			Admin User, 10/21/2020, 9:33 AM	Edit Del	Source	Source_c	Number(10, 0)			Admin User, 10/21/2020, 9:35 AM	Edit Del	Status	Status_c	Number(10, 0)			Admin User, 10/21/2020, 9:36 AM	Edit Del	Subject	Subject_c	Text(255)			Admin User, 10/21/2020, 9:36 AM	Edit Del	Ticket Id	id_c	Text(255)			Admin User, 10/21/2020, 9:31 AM	Edit Del	Twitter Id	twitter_id_c	Text(255)			Admin User, 10/21/2020, 9:38 AM	Edit Del	Type	Type_c	Text(255)			Admin User, 10/21/2020, 9:37 AM	Edit Del	Updated At	updated_at_c	Date/Time			Admin User, 10/21/2020, 9:38 AM
Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By																																																																																																																																																									
Edit Del	Agent Id	responder_id_c	Text(255)			Admin User, 10/21/2020, 9:35 AM																																																																																																																																																									
Edit Del	Company Id	company_id_c	Number(18, 0)			Admin User, 10/21/2020, 9:27 AM																																																																																																																																																									
Edit Del	Created At	created_at_c	Date/Time			Admin User, 10/21/2020, 9:37 AM																																																																																																																																																									
Edit Del	Customer Id	requester_id_c	Text(255)			Admin User, 10/21/2020, 9:34 AM																																																																																																																																																									
Edit Del	Customer Name	name_c	Text(255)			Admin User, 10/21/2020, 9:32 AM																																																																																																																																																									
Edit Del	Customer Phone	phone_c	Phone			Admin User, 10/21/2020, 9:33 AM																																																																																																																																																									
Edit Del	Deleted	deleted_c	Checkbox			Admin User, 10/21/2020, 9:28 AM																																																																																																																																																									
Edit Del	Description	description_c	Rich Text Area(32768)			Admin User, 10/21/2020, 9:28 AM																																																																																																																																																									
Edit Del	Due By	due_by_c	Date/Time			Admin User, 10/21/2020, 9:29 AM																																																																																																																																																									
Edit Del	Due in Freshdesk	fr_due_by_c	Date/Time			Admin User, 10/21/2020, 9:30 AM																																																																																																																																																									
Edit Del	Email	email_c	Email			Admin User, 10/21/2020, 9:29 AM																																																																																																																																																									
Edit Del	Escalated	fr_escalated_c	Checkbox			Admin User, 10/21/2020, 9:31 AM																																																																																																																																																									
Edit Del	Facebook Id	facebook_id_c	Text(255)			Admin User, 10/21/2020, 9:30 AM																																																																																																																																																									
Edit Del	Priority	Priority_c	Number(10, 0)			Admin User, 10/21/2020, 9:33 AM																																																																																																																																																									
Edit Del	Source	Source_c	Number(10, 0)			Admin User, 10/21/2020, 9:35 AM																																																																																																																																																									
Edit Del	Status	Status_c	Number(10, 0)			Admin User, 10/21/2020, 9:36 AM																																																																																																																																																									
Edit Del	Subject	Subject_c	Text(255)			Admin User, 10/21/2020, 9:36 AM																																																																																																																																																									
Edit Del	Ticket Id	id_c	Text(255)			Admin User, 10/21/2020, 9:31 AM																																																																																																																																																									
Edit Del	Twitter Id	twitter_id_c	Text(255)			Admin User, 10/21/2020, 9:38 AM																																																																																																																																																									
Edit Del	Type	Type_c	Text(255)			Admin User, 10/21/2020, 9:37 AM																																																																																																																																																									
Edit Del	Updated At	updated_at_c	Date/Time			Admin User, 10/21/2020, 9:38 AM																																																																																																																																																									
Custom Fields & Relationships Help ?																																																																																																																																																															

Questions for this assignment

Did you use the Wrapper class to Parse the Response that you are getting from the Freshdesk Contact Creation?

Introduction to QuickBooks

QuickBooks is an Accounting software that can help small or medium-sized companies manage the following

- Manage Expenses and Track Expenses
- Payroll & Time Tracking Management
- Online Banking

There are two different types of QuickBooks software

- QuickBooks Desktop (POS)
- QuickBooks Online (QBOO)

Setup QuickBooks Account

To set up the QuickBooks account follow the below steps

- Get the free developer account using - the <https://developer.intuit.com/app/developer/homepage> link
- Setup the default sandbox for demo usage
- Read the API Documentation and generate the Access Token
- Test the Sample API using Postman

QuickBooks Online and Payments

The screenshot shows the 'Give your app a name' step of the developer portal. It includes a text input field with 'My app' typed in, a section for selecting scopes (with options for accounting and payment), and buttons for 'Back' and 'Create app'.

Give your app a name
Keep the name length within 80 characters. Names exceeding this length are truncated when displayed.
What's your app name? My app
Select a scope ⓘ
Select scopes for the APIs you would like to access. Scopes may be added as needed. Scopes cannot be removed once assigned to an app.
 com.intuit.quickbooks.accounting
 com.intuit.quickbooks.payment (US only)
Back Create app

The screenshot shows the 'Keys & OAuth' section of the Intuit Developer dashboard. On the left, there's a sidebar with various settings like 'Get Started', 'Development Settings', and 'Keys & credentials' (which is highlighted with a red box). The main area has a heading 'Keys' with a sub-instruction: 'Use these keys to set up OAuth for your environments. You can learn more here and try the flow in the OAuth 2.0 Playground.' Below this are 'Client ID' and 'Client Secret' input fields, each with a 'Rotate Secret' link. A red arrow points from the top right towards these fields. Further down, there's a section for 'Sandbox Companies' with a note about automatically created companies and a 'View Sandbox companies' link. Another red box highlights the 'Add URI' button in the 'Redirect URLs' section, which lists four entries with their respective links and delete icons.

Get QuickBooks Access Token

Quickbooks uses OAuth 2.0 for authentication which means it uses 2 Step process to get the access token.

You can follow these links to get the URL for

- For sandboxes and testing environments:

https://developer.api.intuit.com/.well-known/openid_sandbox_configuration

- For production apps:

https://developer.api.intuit.com/.well-known/openid_configuration

Step1 - Get the authorization code by preparing the URL and hitting the URL

to the browser

Url -

Step2 - Get the access code from the Postman (for testing purposes) & use the code in Apex or any other language

Body -

Below is the QuickBooks URL to make the API calls after getting the access token

- Production Base URL: <https://quickbooks.api.intuit.com>
- Sandbox Base URL: <https://sandbox-quickbooks.api.intuit.com>

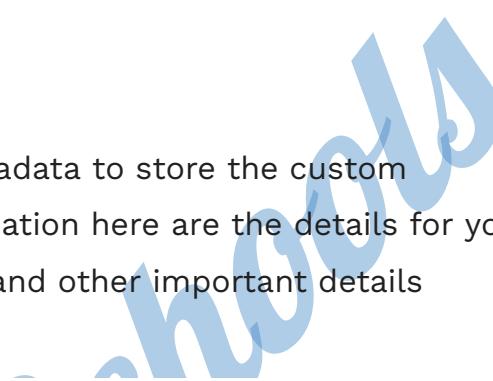
→ **Authenticate Quickbooks from Salesforce**

Here are the steps that we will follow to generate the access token for QuickBooks using Apex Code and VF Page.

1. Add Remote Site Settings - **Mandatory for all the Outbound APIs**
 - a. Below is the URL that we need to add as a Remote Sandbox.
There are a total 3 URLs
 - i. **Production Base URL:** <https://quickbooks.api.intuit.com>
 - ii. **Sandbox Base URL:**
<https://sandbox-quickbooks.api.intuit.com>
 - iii. **Token Url** - <https://oauth.platform.intuit.com>
2. Create a Custom Metadata to store the QuickBooks access token and endpoint details
 - a. You can get all the URLs -
<https://developer.intuit.com/app/developer/qbo/docs/api/accounting/most-commonly-used/account>

3. Create a VF Page and use that VF page as a redirect URI
4. Create a custom label to store the MasterLabel of the custom metadata record that we have created
5. Prepare the URL to get the authorization code
 - a. **We did this in the previous Lecture**
6. Develop an apex class to get the access token and deploy the custom metadata record.

As we are going to create the custom metadata to store the custom metadata in Salesforce after the authentication here are the details for your reference so that we can store our token and other important details



Custom Metadata Type
Quickbooks

Standard Fields (0) | Custom Fields (2) | Validation Rules (0) | Page Layouts (1)

[Edit](#) [Delete](#) [Manage Quickbooks](#)

Singular Label	Plural Label	Description
Quickbooks	Quickbooks	Visibility: Public
qb_Metadata	qb_Metadata	Protection Level:
API Name	qb_Metadata_mdt	Record Size: 5.020
Created By	Admin User, 10/21/2020, 11:14 AM	Modified By: Admin User, 10/23/2020, 7:20 AM

Standard Fields

Action	Field Label	API Name	Field Name	Data Type	Indexed
Edit Del	Created By	access_token_c	CreatedBy	Lookup(User)	
Edit Del	Custom Metadata Record Name	auth_url_c	DeveloperName	Text(40)	
Edit Del	Label	Client_Id_c	MasterLabel	Text(40)	
Edit Del	Last Modified By	Client_Secret_c	LastModifiedBy	Lookup(User)	
Edit Del	Namespace Prefix	Company_Info_c	NamespacePrefix	Text	
Edit Del	Protected Component	Create_Bill_c	IsProtected	Checkbox	

Custom Fields

Action	Field Label	API Name	Data Type	Field Manageability	Indexed	Controlling Field	Modified By
Edit Del	access_token	access_token_c	Long Text Area(32768)	Upgradable			Admin User, 10/21/2020, 9:31 PM
Edit Del	auth_url	auth_url_c	URL(255)	Upgradable			Admin User, 10/22/2020, 10:31 AM
Edit Del	Client Id	Client_Id_c	Text(255)	Upgradable			Admin User, 10/21/2020, 11:14 AM
Edit Del	Client Secret	Client_Secret_c	Text(255)	Upgradable			Admin User, 10/21/2020, 11:14 AM
Edit Del	Company Info	Company_Info_c	URL(255)	Upgradable			Admin User, 10/21/2020, 9:54 PM
Edit Del	Create Bill	Create_Bill_c	URL(255)	Upgradable			Admin User, 10/21/2020, 10:28 PM
Edit Del	Create Customer	Create_Customer_c	URL(255)	Upgradable			Admin User, 10/21/2020, 9:54 PM
Edit Del	Create Estimate	Create_Estimate_c	URL(255)	Upgradable			Admin User, 10/21/2020, 11:56 PM
Edit Del	Create Invoice	Create_Invoice_c	URL(255)	Upgradable			Admin User, 10/21/2020, 9:55 PM
Edit Del	Create Payment	Create_Payment_c	URL(255)	Upgradable			Admin User, 10/21/2020, 9:53 PM
Edit Del	Create Vendor	Create_Vendor_c	URL(255)	Upgradable			Admin User, 10/22/2020, 12:50 AM
Edit Del	Customer Url	Customer_Url_c	URL(255)	Upgradable			Admin User, 10/21/2020, 9:54 PM
Edit Del Replace	Environment	Environment_c	Picklist	Upgradable			Admin User, 10/21/2020, 9:39 PM
Edit Del	expires_in	expires_in_c	Number(10, 0)	Upgradable			Admin User, 10/21/2020, 9:32 PM
Edit Del	expires_in_time	expires_in_time_c	Date/Time	Upgradable			Admin User, 10/21/2020, 9:33 PM
Edit Del	minversion	minversion_c	Text(4)	Upgradable			Admin User, 10/21/2020, 11:16 AM
Edit Del	PageName	PageName_c	Text(255)	Upgradable			Admin User, 10/23/2020, 5:56 AM
Edit Del	Production Base URL	Prod_Base_URL_c	URL(255)	Upgradable			Admin User, 10/21/2020, 11:15 AM
Edit Del	realmId	realmId_c	Text(255)	Upgradable			Admin User, 10/21/2020, 9:32 PM
Edit Del	refresh token	refresh_token_c	Long Text Area(32768)	Upgradable			Admin User, 10/21/2020, 9:31 PM
Edit Del	Sandbox Base URL	Sandbox_Base_URL_c	URL(255)	Upgradable			Admin User, 10/21/2020, 11:15 AM
Edit Del	token_type	token_type_c	Text(255)	Upgradable			Admin User, 10/21/2020, 9:32 PM
Edit Del	token_url	token_url_c	URL(255)	Upgradable			Admin User, 10/22/2020, 10:34 AM

Below is how the metadata record will look like

Make API Callouts to QuickBooks

Below are the operations that we are going to perform here in the part of the course

- Create a Customer to Quickbooks
 - Create an Invoice record to QuickBooks related to the created contact
 - Create Payment record to Quickbooks

Implement the Refresh Token method

As we know that QuickBooks tokens expire after a certain amount of time it is very important to refresh the access token if the token is expired.

We can get the complete document about the refresh token from this link -
<https://developer.intuit.com/app/developer/qbo/docs/develop/authentication-and-authorization/oauth-2.0>

Here's an example request

POST /oauth2/v1/tokens/bearer HTTP/1.1

Accept: application/json

Authorization: Basic UTM0dVBvRDIwanp2OUdxNxE1dmlMemppcTlwM1d2

NzRUdDNReGkwZVNTTDhFRWwx0g6VEh0WEJlR3dheEtZSlVNaFhzeGxma1l

XaFg3ZlFlRzFtN2szTFRwbw==

Content-Type: application/x-www-form-urlencoded

Body: grant_type=refresh_token&

refresh_token=Q311488394272qbajGfLBwGmVsbF6VoNpUKaI05oL49aXLVJUB

Access Token PAGE

```
<apex:page id="thePage" lightningStylesheets="true"
showHeader="true" controller="PS_QBTokenUtil" >
<script>
    window.onload = function(){
        let code  =  '{!$CurrentPage.parameters.code}';
        let realmId  =  '{!$CurrentPage.parameters.realmId}';
        if( code && realmId ){
            fetchAccessToken(); // Action Function
        }
    }
</script>
<apex:form id="theForm" >
    <apex:outputPanel id="errorMessage">
    </apex:outputPanel>
    <apex:pageMessages ></apex:pageMessages>
    <apex:actionstatus id="theStatus" >
        <apex:facet name="start" >
            <div class="waitingSearchDiv" id="el_loading"
                style="background-color: #fbfbfb;
height:100%;opacity:0.65;width:100%;">
                <div class="waitingHolder" style="top: 100px;
```

```
width: 91px;">

    <span class="waitingDescription">Loading...</span>
</div>
</div>
</apex:facet>
</apex:actionstatus>

<apex:actionFunction name="fetchAccessToken" status="theStatus
errorMessage"
action="{!!getAccessToken}"
reRender="theForm" />

<apex:pageBlock >
    <apex:pageBlockButtons >
        <apex:commandButton value="Authorize QuickBooks
Account" status="theStatus errorMessage"
action="{!!authorize}" />
    </apex:pageBlockButtons>
</apex:pageBlock>

</apex:form>
</apex:page>
```

Access Token Apex Class

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 04-07-2024  
 * @last modified by : Amit Singh - PantherSchools  
 ***/  
  
public with sharing class PS_QBTokenUtil {  
    /** Prepare the Auth Url */  
    public PageReference authorize(){  
        /** Get the config from the Custom Metadata */  
        qb_Metadata__mdt config =  
            qb_Metadata__mdt.getInstance('QBToken');  
        if(config !=null){  
            //  
            https://integration-org2-dev-ed--c.develop.vf.force.com/apex/PS_Qu  
ickBooksToken  
            String redirect_uri =  
                System.URL.getOrgDomainURL().toExternalForm()+'apex/'+config.Page  
Name__c;  
            System.System.debug( redirect_uri );  
            //  
            https://integration-org2-dev-ed.develop.my.salesforce.com/apex/PS_  
QuickBooksToken  
            /*  
             https://appcenter.intuit.com/connect/oauth2?  
client_id=Q3ylJatCvnkYqVKLmkxxxxxxxxxxxxxkYB36b5mws7HkKUEv9aI&re  
sponse_type=code&  
scope=com.intuit.quickbooks.accounting&
```

```
redirect_uri=https://www.mydemoapp.com/oauth-redirect&

state=security_token%3D138r5719ru3e1%26url%3Dhttps://www.mydemoapp
.com/oauth-redirect
 */

String authorizeUrl =
config.auth_url__c+'?client_id='+config.Client_Id__c+'&response_ty
pe=code'
        +'&scope=openid profile email
com.intuit.quickbooks.accounting com.intuit.quickbooks.payment'

+'&redirect_uri='+redirect_uri+'&state=' +UserInfo.getOrganizationI
d()+redirect_uri;

System.System.debug( authorizeUrl );
return new PageReference(authorizeUrl);
}else{
    // Show the Error
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.WARNING, 'Config not found'
));
    return null;
}
}

/** Get the Access Token and Store into Metadata */
public void get_accessToken(){
    //
https://integration-org2-dev-ed--c.develop.vf.force.com/apex/PS_Qu
ickBooksToken?

//code=AB11712461984Hf0QnBcF20ni0WQjxnRlnnH1xNowVdBkpk0Hd
```

```

//&state=00DHu000003NXeOMAwhttps%3A%2F%2Fintegration-org2-dev-ed.d
evelop.my.salesforce.com%2Fapex%2FPS_QuickBooksToken
//&realmId=9341452084525746

    /**
     * Get the config from the Custom Metadata */
    qb_Metadata__mdt config =
qb_Metadata__mdt.getInstance('QBToken');
    if(config !=null){
        String code      =
ApexPages.currentPage().getParameters().get('code');
        String realmId =
ApexPages.currentPage().getParameters().get('realmId');

        String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+'/apex/'+config.Page
Name__c;
        System.System.debug( redirect_uri );

        HttpRequest httReq = new HttpRequest();
        httReq.setMethod('POST');
        httReq.setHeader('Content-Type',
'application/x-www-form-urlencoded');
        httReq.setHeader('Accept', 'application/json');
        httReq.setEndpoint(config.token_url__c);
        /*
            grant_type=authorization_code&
code=L3114709614564VSU8JSEiPkXx1xhV8D9mv4xbv6sZJycibMUI&
redirect_uri=https://www.mydemoapp.com/oauth-redirect
        */
    }
}

```

```
        String tokenBody =
'grant_type=authorization_code&code=' + code + '&redirect_uri=' + redirect_uri
+ '&client_id=' + config.Client_Id_c + '&client_secret=' + config.Client_Secret_c;
        httReq.setBody(tokenBody);

        try {
            HttpResponse httpRes = (new Http()).send(httReq);
            if(httpRes.getStatusCode() == 200 ||
httpRes.getStatusCode() == 201){
                // Parse the Response
                /**
                 * {
                    "token_type": "bearer",
                    "expires_in": 3600,
                    "refresh_token": "Q311488394272qabajGfLBwGmVsbF6VoNpUKaI05oL49aXLVJU
B",
                    "x_refresh_token_expires_in":15551893,
                    "access_token": "eJlb"
                }
                */
                Map<String, Object> responseMap = (Map<String, Object>)JSON.deserializeUntyped(httpRes.getBody());
                String access_token =
(String)responseMap.get('access_token');
                String refresh_token =
(String)responseMap.get('refresh_token');
                Integer expires_in =
(Integer)responseMap.get('expires_in');
```

```
// Prepare the Map for Custom Metadata
String fullName = 'qb_Metadata.QBToken';
String label     = 'QBToken';
Map<String, Object> fieldWithValuesMap = new
Map<String, Object>();
fieldWithValuesMap.put('access_token__c',
access_token);

fieldWithValuesMap.put('expires_in__c', expires_in);
fieldWithValuesMap.put('expires_in_time__c',
System.now().addSeconds(expires_in) );
fieldWithValuesMap.put('refresh_token__c',
refresh_token);
fieldWithValuesMap.put('realmId__c', realmId);

// Deploy/Update the Custom Metadata

CreateUpdateMetadataUtils.createUpdateMetadata(fullName, label,
fieldWithValuesMap);
ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.CONFIRM, 'Successful!'));
}else{
System.debug(httpRes.getBody());
ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, httpRes.getBody() ));
}
}catch (CalloutException ex) {
System.debug(ex);
ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));
}catch (Exception ex) {
```

```

        System.debug(ex);
        ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));
    }
}else{
    // Show the Error
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.WARNING, 'Config not found'
));
}
}

public void refesh_token(qb_Metadata__mdt config){
    // Do Refresh Logic
}
}

```

Apex Class with Refresh Token

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 04-20-2024
 * @last modified by : Amit Singh - PantherSchools
 */
public with sharing class PS_QBTokenUtil {
    /** Prepare the Auth Url */
    public PageReference authorize(){
        /** Get the config from the Custom Metadata */
        qb_Metadata__mdt config = qb_Metadata__mdt.getInstance('QBToken');
        if(config !=null){
            //
https://integration-org2-dev-ed--c.develop.vf.force.com/apex/PS\_QuickBook

```

```

sToken

        String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+' /apex/' +config.PageName__c
;

        System.System.debug( redirect_uri );
        //

https://integration-org2-dev-ed.develop.my.salesforce.com/apex/PS_QuickBo
oksToken

/*
https://appcenter.intuit.com/connect/oauth2?

client_id=Q3ylJatCvnkYqVKLmkxxxxxxxxxxxxxkYB36b5mws7HkKUEv9aI&response_
type=code&
scope=com.intuit.quickbooks.accounting&
redirect_uri=https://www.mydemoapp.com/oauth-redirect&

state=security_token%3D138r5719ru3e1%26url%3Dhttps://www.mydemoapp.com/oa
uth-redirect
*/

```



```

String authorizeUrl =
config.auth_url__c+'?client_id='+config.Client_Id__c+'&response_type=code
'

        +'&scope=openid profile email
com.intuit.quickbooks.accounting com.intuit.quickbooks.payment'
        +'&redirect_uri=' + redirect_uri + '&state=' + UserInfo.getOrganizationId() + red
irect_uri;

        System.System.debug( authorizeUrl );
        return new PageReference(authorizeUrl);
}else{
        // Show the Error
        ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.WARNING, 'Config not found' ));

```

```

        return null;
    }

}

/** Get the Access Token and Store into Metadata */
public static void getAccessToken(){

    //
https://integration-org2-dev-ed--c.develop.vf.force.com/apex/PS_QuickBook
sToken?

//code=AB11712461984Hf0QnBcF20ni0WQjxnRlnnH1xNowVdBkpk0Hd

//&state=00DHu000003NXeOMAWhttps%3A%2F%2Fintegration-org2-dev-ed.develop.
my.salesforce.com%2Fapex%2FPS_QuickBooksToken
//&realmId=9341452084525746

/** Get the config from the Custom Metadata */
qb_Metadata__mdt config = qb_Metadata__mdt.getInstance('QBToken');
if(config !=null){
    String code      =
ApexPages.currentPage().getParameters().get('code');
    String realmId =
ApexPages.currentPage().getParameters().get('realmId');

    String redirect_uri =
System.URL.getOrgDomainURL().toExternalForm()+'/apex/'+config.PageName__c
;
    System.System.debug( redirect_uri );

HttpRequest httReq = new HttpRequest();
httReq.setMethod('POST');
httReq.setHeader('Content-Type',
'application/x-www-form-urlencoded');
    httReq.setHeader('Accept', 'application/json');
    httReq.setEndpoint(config.token_url__c);
/*

```

```

grant_type=authorization_code&
code=L3114709614564VSU8JSEiPkXx1xhV8D9mv4xbv6sZJycibMUI&
redirect_uri=https://www.mydemoapp.com/oauth-redirect
*/
String tokenBody =
'grant_type=authorization_code&code=' + code + '&redirect_uri=' + redirect_uri
+ '&client_id=' + config.Client_Id__c + '&client_secret=' + config.Client_Secret
__c;
httReq.setBody(tokenBody);

try {
    HttpResponse httpRes = (new Http()).send(httReq);
    if(httpRes.getStatusCode() == 200 ||
httpRes.getStatusCode() == 201){
        // Parse the Response
        /**
         * {
            "token_type": "bearer",
            "expires_in": 3600,
            "refresh_token": "Q311488394272qbajGfLBwGmVsbF6VoNpUKaI05oL49aXLVJUB",
            "x_refresh_token_expires_in": 15551893,
            "access_token": "eJlb"
        }
    */
    Map<String, Object> fieldWithValuesMap =
prepareMetadata(httpRes.getBody(), '');
    String fullName = 'qb_Metadata.QBToken';
    String label     = 'QBToken';
    // Deploy/Update the Custom Metadata
CreateUpdateMetadataUtils.createUpdateMetadata(fullName, label,
fieldWithValuesMap);
ApexPages.addmessage(new

```

```

ApexPages.message(ApexPages.severity.CONFIRM, 'Successfull!'));
    }else{
        System.debug(httpRes.getBody());
        ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, httpRes.getBody() ));
    }
}catch (CalloutException ex) {
    System.debug(ex);
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));
}catch (Exception ex) {
    System.debug(ex);
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.ERROR, ex.getMessage() ));
}
}else{
    // Show the Error
    ApexPages.addmessage(new
ApexPages.message(ApexPages.severity.WARNING, 'Config not found' ));
}
}

public static Boolean isValid(qb_Metadata__mdt config){
    Boolean isValid = true;
    if(config.expires_in_time__c <= System.now()){
        isValid = false;
    }
    return isValid;
}

public static Map<String, Object> refreshToken(qb_Metadata__mdt
config){

    String tokenUrl = config.token_url__c;

```

```
String requestBody =
'grant_type=refresh_token&client_id='+config.client_id__c
+'&client_secret='+config.client_secret__c+'&refresh_token='+config.refre
sh_token__c;

System.debug(requestBody);
HttpRequest httpReq =
PS_CalloutUtils.prepareRequest(tokenUrl, 'POST', requestBody, 'application/j
son', 'application/x-www-form-urlencoded');
Map<String, Object> fieldWithValuesMap = new Map<String,
Object>();
try{
    HttpResponse httpRes = (new Http()).send(httpReq);
    if(httpRes.getStatusCode() == 200 || httpRes.getStatusCode()
== 201 ){
        fieldWithValuesMap =
prepareMetadata(httpRes.getBody(), '');
    }else{
        }
}catch(System.CalloutException ex){
    }catch(System.Exception ex){
        }
    return fieldWithValuesMap;
}

public static Map<String, Object> prepareMetadata(String requestBody,
String realmId){
    Map<String, Object> responseMap = (Map<String,
Object>)JSON.deserializeUntyped(requestBody);
    String access_token  = (String)responseMap.get('access_token');
    String refresh_token = (String)responseMap.get('refresh_token');
```

```
Integer expires_in    = (Integer)responseMap.get('expires_in');

Map<String, Object> fieldWithValuesMap = new Map<String,
Object>();
fieldWithValuesMap.put('access_token__c', access_token);
fieldWithValuesMap.put('expires_in__c', expires_in);
fieldWithValuesMap.put('expires_in_time__c',
System.now().addSeconds(expires_in) );
fieldWithValuesMap.put('refresh_token__c', refresh_token);
if(!String.isBlank(realmId)){
    fieldWithValuesMap.put('realmId__c', realmId);
}
return fieldWithValuesMap;
}
}
```

QB_Items Apex Class

```
/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 04-20-2024
 * @last modified by : Amit Singh - PantherSchools
 */
```

```

public with sharing class QB_Items {

    // QB_Items.testCreateItem();
    public static void testCreateItem(){
        List<Product2> products = [Select Id, Name FROM Product2 WHERE
QBExternalID__c = null ];
        // System.CalloutException: You have uncommitted work pending.
        Please commit or rollback before calling out
        /**
         * Callout
         * Callout
         * Callout
         * Callout
         * DML - Works

         * Callout
         * DML
         * Callout - Fails System.CalloutException: You have uncommitted
work pending. Please commit or rollback before calling out
        */
        for(Product2 prod: products){
            createItemInQB(prod); // Not a Best Practice, but will do for
POC
        }
        update products;
    }

    public static Product2 createItemInQB(Product2 productRecord){
        /** Get the config from the Custom Metadta */
        List<qb_Metadata_mdt> configList = [SELECT Id, DeveloperName,
MasterLabel, Environment__c, Prod_Base_URL__c, Sanbdox_Base_URL__c,
access_token__c, auth_url__c,
expires_in__c, expires_in_time__c, minorversion__c,
realmId__c, refresh_token__c,
token_type__c, token_url__c
    
```

```

        FROM qb_Metadata_mdt
        WHERE DeveloperName = 'QBToken'
        LIMIT 1]; // Token Details Are
Stored Here...

if(configList?.size() > 0){
    qb_Metadata_mdt config = configList.get(0);

    String accessToken = config.access_token_c;

    Boolean isValid = PS_QBTokenUtil.isValid(config);
    Map<String, Object> fieldWithValuesMap;

    if(isValid == false){ // Token is Expired
        // Refresh the Token
        fieldWithValuesMap = PS_QBTokenUtil.refreshToken(config);
        accessToken =
(String)fieldWithValuesMap.get('access_token_c'); // null/blank
    }

    HttpRequest httReq = new HttpRequest();
    httReq.setMethod('POST');
    httReq.setHeader('Content-Type', 'application/json');
    httReq.setHeader('Authorization', 'Bearer '+accessToken);
    httReq.setHeader('Accept', 'application/json');

    String endpoint =
config.Prod_Base_URL_c+'/v3/company/'+config.realmId_c+'/item?minorvers
ion='+config.minorversion_c;
    if(config.Environment_c.equals('Sandbox')){
        endpoint =
config.Sanbdox_Base_URL_c+'/v3/company/'+config.realmId_c+'/item?minorv
ersion='+config.minorversion_c;
    }

    httReq.setEndpoint(endpoint);
}

```

```

String requestBody = '{'+
    "TrackQtyOnHand": false, '+
    "Name": "'+productRecord.Name+'", '+
    "QtyOnHand": 180, '+
    "IncomeAccountRef": {'+
        "name": "'+System.Label.IncomeAccountRefName+'", '+
        "value": "'+System.Label.IncomeAccountRefValue+'"+
    }, '+
    "AssetAccountRef": {'+
        "name": "'+System.Label.AssetAccountRefName+'", '+
        "value": "'+System.Label.AssetAccountRefValue+'"+
    }, '+
    "InvStartDate": "2025-01-01", '+
    "Type": "Service", '+
    "ExpenseAccountRef": {'+
        "name": "'+System.Label.ExpenseAccountRefName+'", '+
        "value": "'+System.Label.ExpenseAccountRefValue+'"+
    }'+

}';

httReq.setBody(requestBody);
try {
    HttpResponse httpRes = (new Http()).send(httReq);
    if(httpRes.getStatusCode() == 200 ||
httpRes.getStatusCode() == 201 ){
        Map<String, Object> responseMap =
(Map<String, Object>)System.JSON.deserializeUntyped(httpRes.getBody());
        Object item = responseMap.get('Item'); // {}
        Map<String, Object> itemMap = (Map<String, Object>)item;
        System.debug(' itemMap \n
'+System.JSON.serializePretty(itemMap));
        productRecord.QBExternalID__c =
(String)itemMap.get('Id');
        productRecord.Synced_With_Quickbooks__c = true;
    }
}

```

```

        //upsert productRecord QBExternalID__c;

    }else{
        System.debug(' httpRes getBody '+httpRes.getBody());
        System.debug(' httpRes getStatusCode
'+httpRes.getStatusCode());
    }
}catch(CalloutException ex) {
    System.debug('Exception Executed '+ex);
}catch(Exception ex) {
    System.debug('Not Callout Exception Executed '+ex);
}

/**
 * Update the Custom metadata record with the latest values...
 */
if(fieldWithValuesMap?.size() > 0){
    System.debug('Updating the Custom Metadata... ');
    String fullName = 'qb_Metadata.QBToken';
    String label      = 'QBToken';
    // Deploy/Update the Custom Metadata
    CreateUpdateMetadataUtils.createUpdateMetadata(fullName,
label, fieldWithValuesMap);
}
}else{
    // Enhance this...
}
return productRecord;
}
}

```

QB_Customers Apex Class

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 04-20-2024  
 * @last modified by : Amit Singh - PantherSchools  
 **/  
  
public with sharing class QB_Customers {  
    // QB_Customers.testCreateCustomer();  
    public static void testCreateCustomer(){  
        QB_CustomerInput customer = new QB_CustomerInput();  
        customer.CompanyName = 'PantherSchools.com';  
        customer.DisplayName = 'PantherSchools.com';  
        customer.FamilyName = 'Singh';  
        customer.FullyQualifiedNome = 'PantherSchools.com-Amit-Singh';  
        customer.GivenName = 'Amit';  
        customer.MiddleName = '';  
        customer.Notes = 'Testing from Apex Class';  
        customer.Suffix = 'Jr.';  
        customer.Title = 'Mr';  
  
        /* Prepare PrimaryEmailAddr */  
        QB_CustomerInput.PrimaryEmailAddr email = new  
        QB_CustomerInput.PrimaryEmailAddr();  
        email.Address = 'asingh@zmail.com';  
        /*  
         "PrimaryEmailAddr": {  
             "Address": "{$randomEmail}"  
         }  
        */  
        customer.PrimaryEmailAddr = email;  
  
        /* Prepare PrimaryPhone */  
        QB_CustomerInput.PrimaryPhone phone = new  
        QB_CustomerInput.PrimaryPhone();
```

```

        phone.FreeFormNumber = '987653210';
        customer.PrimaryPhone = phone;

        /* Prepare BillAddr */

        createCustomer(customer);
    }

public static void createCustomer(QB_CustomerInput customer){

    /** Get the config from the Custom Metadta */
    List<qb_Metadata__mdt> configList = [SELECT Id, DeveloperName,
    MasterLabel, Environment__c, Prod_Base_URL__c, Sanbdox_Base_URL__c,
                                         access_token__c, auth_url__c,
                                         expires_in__c, expires_in_time__c, minorversion__c,
                                         realmId__c, refresh_token__c,
                                         token_type__c, token_url__c
                                         FROM qb_Metadata__mdt
                                         WHERE DeveloperName = 'QBToken'
                                         LIMIT 1]; // Token Details Are
    Stored Here...

    if(configList?.size() > 0){
        qb_Metadata__mdt config = configList.get(0);
        System.debug(config.access_token__c);
        String accessToken = config.access_token__c;
        System.debug(accessToken);
        Boolean isValid = PS_QBTokenUtil.isValid(config);
        Map<String, Object> fieldWithValuesMap;

        if(isValid == false){ // Token is Expired
            // Refresh the Token
            fieldWithValuesMap = PS_QBTokenUtil.refreshToken(config);
            accessToken =
                (String)fieldWithValuesMap.get('access_token__c'); // null/blank
        }
    }
}

```

```

//System.debug(JSON.serializePretty(fieldWithValuesMap));

HttpRequest httReq = new HttpRequest();
httReq.setMethod('POST');
httReq.setHeader('Content-Type', 'application/json');
httReq.setHeader('Authorization', 'Bearer '+accessToken);
httReq.setHeader('Accept', 'application/json');

String endpoint =
config.Prod_Base_URL__c+'/v3/company/'+config.realmId__c+'/customer?minor
version='+config.minorversion__c;
if(config.Environment__c.equals('Sandbox')){
    endpoint =
config.Sanbdox_Base_URL__c+'/v3/company/'+config.realmId__c+'/customer?mi
norversion='+config.minorversion__c;
}
System.debug(endpoint);
httReq.setEndpoint(endpoint);

String requestBody = System.JSON.serializePretty(customer);
System.debug(requestBody);
httReq.setBody(requestBody);
try {
    HttpResponse httpRes = (new Http()).send(httReq);
    if(httpRes.getStatusCode() == 200 ||
httpRes.getStatusCode() == 201 ){
        Map<String, Object> responseMap =
(Map<String, Object>)System.JSON.deserializeUntyped(httpRes.getBody());
        String customerId = (String)responseMap.get('Id');
        System.debug('Successfully Inserted Customer '+
customerId);
        /*
            Update the information in Salesforce
            OR Return the Response from here back to the
        */
    }
}

```

```

calling class...
    */

    }else{
        System.debug(' httpRes getBody '+httpRes.getBody());
        System.debug(' httpRes getStatuscode
'+httpRes.getStatusCode());
    }
}catch(CalloutException ex) {
    System.debug('Exception Executed '+ex);
}catch(Exception ex) {
    System.debug('Not Callout Exception Executed '+ex);
}
/***
 * Update the Custom metadata record with the latest values...
 */
if(fieldWithValuesMap?.size() > 0){
    System.debug('Updating the Custom Metadata... ');
    String fullName = 'qb_Metadata.QBToken';
    String label     = 'QBToken';
    // Deploy/Update the Custom Metadata
    CreateUpdateMetadataUtils.createUpdateMetadata(fullName,
label, fieldWithValuesMap);
}
}else{
    // Enhance this...
}
}

}

```

QB_CustomerInput Apex Class

```
/**
```

```

* @description      :
* @author           : Amit Singh - PantherSchools
* @group            :
* @last modified on : 04-20-2024
* @last modified by : Amit Singh - PantherSchools
*/
public with sharing class QB_CustomerInput {
    public String FullyQualifiedName;
    public PrimaryEmailAddr PrimaryEmailAddr;
    public String DisplayName;
    public String Suffix;
    public String Title;
    public String MiddleName;
    public String Notes;
    public String FamilyName;
    public PrimaryPhone PrimaryPhone;
    public String CompanyName;
    public BillAddr BillAddr;
    public String GivenName;
    public class PrimaryEmailAddr {
        public String Address;
    }
    public class PrimaryPhone {
        public String FreeFormNumber;
    }
    public class BillAddr {
        public String CountrySubDivisionCode;
        public String City;
        public String PostalCode;
        public String Line1;
        public String Country;
    }
}

```

Assignments

Prerequisites -

To work on the Assignments, You need to create the following objects

- [QuickBooks Customer](#)
- [QuickBooks Invoice](#)
- QuickBooks Invoice Line
- [QuickBooks Payment](#)

You can get the fields from the QuickBooks API OR You can refer to Github to See the fields

- [Customer](#)
- [Invoice Line](#)
- [Invoice](#)
- [payment](#)

#1

- Create the following fields on the Product2 Object
 - **Synced With QuickBooks** A checkbox field
 - **Sync with QuickBooks** a checkbox field
 - **QB External Id** - A Text field which is marked as External Id
- Create a Scheduled Apex which will Query All the Product Records which have **Sync with QuickBooks** field as true and create those records in QuickBooks as Items. You can use [Items API](#) and achieve the same.
 - When the Item is created in QuickBooks then update the following fields
 - Synced With QuickBooks to True which will indicate that the record is synced.
 - QB External ID - field with the unique ID returned by the QuickBooks
- Also, Develop a Logic so that if a product record is updated and the **Sync with QuickBooks field has been changed to True from False then**

create the Same Product as Items in QuickBooks.

- When the Item is created in QuickBooks then update the following fields
 - Synced With QuickBooks to True which will indicate that the record is synced.
 - QB External ID - field with the unique ID returned by the QuickBooks
- Below is the sample cUrl Request

```
curl --location  
'https://sandbox-quickbooks.api.intuit.com/v3/company/realmID/item  
?minorversion=70' \  
--header 'Accept: application/json' \  
--header 'Content-Type: application/json' \  
--header 'Authorization: Bearer eyJlbmMiOiJBMTI4Q0JD  
LU' \  
--data '{  
    "TrackQtyOnHand": false,  
    "Name": "Table",  
    "QtyOnHand": 180,  
    "IncomeAccountRef": {  
        "name": "Sales of Product Income",  
        "value": "79"  
    },  
    "AssetAccountRef": {  
        "name": "Inventory Asset",  
        "value": "81"  
    },  
    "InvStartDate": "2025-01-01",  
    "Type": "Service",  
    "ExpenseAccountRef": {  
        "name": "Cost of Goods Sold",  
        "value": "80"  
    }  
}'
```

```
}
```

```
'
```

#1.1 -

- Create a new custom object in Salesforce and Label it as **Quickbooks Invoice** and the object name should be **qb_Invoice**. You can get all the fields from [Here](#).
- Create a Lightning Web/Aura Component which will take input from the user to create an Invoice record into the Quickbooks system and if the Invoice is created save the invoice in the salesforce system as well.
- You can hardcode the customer to associate the Invoice with like shown below image (**Not Recommended**) OR
 - Create a Lookup field for searching the **QuickBooks Customers**. You can use “[Lightning Record Picker](#)” for the same (**Recommended**)
 - And Create the Invoice for the Selected Customer
 - If the Invoice is created then Create the Invoice record back with the details coming back from QuickBooks
- When an Invoice is created using **QuickBooks Invoice** Object all the required items are added. And now the Invoice is updated when the **“Send To QuickBooks”** field is changed to True from False then
 - Check if there are at least 1 Line Item is added
 - If there is 1 or more than 1 line item added then create the Invoice on QuickBooks
 - If the Invoice is created then Update the Invoice record back with the details coming back from QuickBooks

Item Name
type here...

Amount
type here...

Customer
Amy's Bird Sanctuary

Customer RefNo
1

Create Invoice

As we have to store the details within Salesforce, here are the fields that you need to create for the **QuickBooks Invoice object**

Note:- Please create only the fields that you want to store on Salesforce.

Singular Label	Quickbooks Invoice	Description				
Plural Label	Quickbooks Invoices	Enable Reports <input type="checkbox"/>				
Object Name	qb_Invoice	Track Activities <input type="checkbox"/>				
API Name	qb_Invoice_c	Allow in Chatter Groups <input type="checkbox"/>				
		Allow Sharing <input checked="" type="checkbox"/>				
		Allow Bulk API Access <input checked="" type="checkbox"/>				
		Allow Streaming API Access <input checked="" type="checkbox"/>				
		Track Field History <input type="checkbox"/>				
		Enable Licensing <input type="checkbox"/>				
		Deployment Status Deployed				
		Allow Search <input checked="" type="checkbox"/>				
		Help Settings Standard salesforce.com Help Window				
		Modified By Admin User, 10/22/2020, 9:47 PM				
<hr/>						
Standard Fields						
Action	Field Label	Field Name	Data Type	Controlling Field	Indexed	
Edit Del	Created By	CreatedBy	Lookup(User)			
Edit Del	Currency	CurrencyIsoCode	Picklist			
Edit Del	Invoice #	Name	Text(80)		✓	
Edit Del	Last Modified By	LastModifiedBy	Lookup(User)			
Edit Del	Owner	Owner	Lookup(User,Group)		✓	
<hr/>			Standard Fields Help ?			
Custom Fields & Relationships			Custom Fields & Relationships Help ?			
Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit Del	ApplyTaxAfterDiscount	ApplyTaxAfterDiscount_c	Checkbox			Admin User, 10/22/2020, 10:34 PM
Edit Del	Balance	Balance_c	Currency(16, 2)			Admin User, 10/22/2020, 11:03 PM
Edit Del	Billing Email	BillEmail_c	Email			Admin User, 10/22/2020, 11:21 PM
Edit Del	Billing Location	BillAddr_c	Geolocation			Admin User, 10/23/2020, 12:17 AM
Edit Del	Create Time	CreateTime_c	Text(255)			Admin User, 10/23/2020, 12:28 AM
Edit Del	Customer Id	CustomerId_c	Text(255)			Admin User, 10/22/2020, 11:15 PM
Edit Del	CustomerMemo	CustomerMemo_c	Text Area(255)			Admin User, 10/22/2020, 10:52 PM
Edit Del	Customer Name	CustomerName_c	Text(255)			Admin User, 10/22/2020, 11:10 PM
Edit Del	Deposit	Deposit_c	Currency(10, 2)			Admin User, 10/22/2020, 10:58 PM
Edit Del	DocNumber	DocNumber_c	Text(10)			Admin User, 10/22/2020, 10:43 PM
Edit Del	domain	domain_c	Text(5)			Admin User, 10/22/2020, 10:00 PM
Edit Del	Due Date	DueDate_c	Date			Admin User, 10/22/2020, 10:30 PM
Edit Del	Email Status	EmailStatus_c	Text(255)			Admin User, 10/23/2020, 12:12 AM
Edit Del	Invoice Id	InvoiceId_c	Text(255) (External ID)	✓		Admin User, 10/23/2020, 12:36 AM
Edit Del	Last Updated Time	LastUpdatedTime_c	Text(255)			Admin User, 10/23/2020, 12:33 AM
Edit Del	Print Status	PrintStatus_c	Text(255)			Admin User, 10/22/2020, 10:06 PM
Edit Del	SalesTermRef	SalesTermRef_c	Text(10)			Admin User, 10/22/2020, 10:12 PM
Edit Del	Shipping City	ShipCity_c	Text(255)			Admin User, 10/22/2020, 11:25 PM
Edit Del	Shipping Postal Code	ShipPostalCode_c	Text(10)			Admin User, 10/22/2020, 11:45 PM
Edit Del	Shipping State	ShipState_c	Text(10)			Admin User, 10/23/2020, 12:08 AM
Edit Del	Shipping Street	ShipStreet_c	Text(255)			Admin User, 10/22/2020, 11:31 PM
Edit Del	Total Amount	TotalAmt_c	Currency(16, 2)			Admin User, 10/22/2020, 10:26 PM
Edit Del	Transaction Date	TxnDate_c	Date			Admin User, 10/22/2020, 9:54 PM

Custom Object
Quickbooks Invoice Line

Help for this Page 

[Standard Fields \(4\)](#) | [Custom Fields & Relationships \(1\)](#) | [Validation Rules \(0\)](#) | [Page Layouts \(1\)](#) | [Field Sets \(0\)](#) | [Compact Layouts \(1\)](#) | [Buttons, Links, and Actions \(0\)](#) | [Record Types \(0\)](#) | [Object Limits \(10\)](#)

Custom Object Definition Detail

Singular Label	Quickbooks Invoice Line	Edit	Delete
Plural Label	Quickbooks Invoice Lines		
Object Name	qb_Invoice_Line		
API Name	qb_Invoice_Line__c		
Description			
Enable Reports <input type="checkbox"/>			
Track Activities <input type="checkbox"/>			
Allow in Chatter Groups <input type="checkbox"/>			
Allow Sharing <input checked="" type="checkbox"/>			
Allow Bulk API Access <input checked="" type="checkbox"/>			
Allow Streaming API Access <input checked="" type="checkbox"/>			
Track Field History <input type="checkbox"/>			
Enable Licensing <input type="checkbox"/>			
Deployment Status Deployed			
Allow Search <input type="checkbox"/>			
Help Settings Standard salesforce.com Help Window			
Modified By Admin User: 10/23/2020, 12:43 AM			

Created By Admin User: 10/23/2020, 12:43 AM

Standard Fields

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
Created By	Created By	CreatedBy	Lookup(User)		
Edit	Currency	CurrencyIsoCode	Picklist		
Edit	InvoiceLine #	Name	Text(80)		<input checked="" type="checkbox"/>
Last Modified By	Last Modified By	LastModifiedBy	Lookup(User)		

[Standard Fields Help](#) ?

Custom Fields & Relationships

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit Del	Quickbooks Invoice	qb_Invoice__c	Master-Detail(Quickbooks Invoice)	<input checked="" type="checkbox"/>		Admin User: 10/23/2020, 1:09 AM

[Custom Fields & Relationships Help](#) ?

Related Lookup Filters

No related lookup filters defined.

Validation Rules

[New](#)

[Validation Rules Help](#) ?

#2

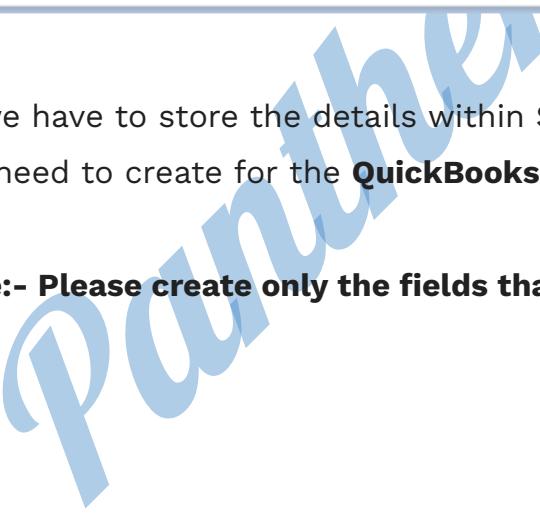
- Create a new custom object in Salesforce and Label it as **Quickbooks Payment** and the object name should be **qb_Payment**. Get all the fields from [Here](#).
- Create a Lightning Web/Aura Component which will take input from the user to create the Payment record into the Quickbooks system and if the Payment is created save the Payment in the salesforce system as well.
- You can hardcode the customer to associate the payment with as shown in the picture (Not Recommended)
 - Create a Lookup field for searching the QuickBooks Customers and Invoices. You can use “[Lightning Record Picker](#)” for the same (**Recommended**)
 - And Create the Payment for the Selected Customer & Invoice
 - If the Invoice is created then Create the Payment record back with the details coming back from QuickBooks and Link the Payment with the Invoice Record.
- When a Payment is created using **QuickBooks Payment** Object is created with the “**Send To QuickBooks**” field to set to True

- Create the Payment on QuickBooks and relate to the Customer and Payment
 - If the Payment is created then Update the Payment record back with the details coming back from QuickBooks
- When a Payment record is updated with the “**Send To QuickBooks**” field to set to True
 - Create the Payment on QuickBooks and relate to the Customer and Payment
 - If the Payment is created then Update the Payment record back with the details coming back from QuickBooks

Below is the Sample **JSON** request for the same

```
curl --location
'https://sandbox-quickbooks.api.intuit.com/v3/company/realmId/payment?minorversion=70' \
--header 'Content-Type: application/json' \
--header 'Authorization: Bearer eyJlbmMiOiJBMTI4Q0JDLUhTMjU2 \
--data '{
  "TotalAmt": 500.0,
  "CustomerRef": {
    "value": "61" // Customer Id
  },
  "CurrencyRef": {
    "value": "USD"
  },
  "Line": [
    {
      "Amount": 500.0,
      "LinkedTxn": [
        {
          "TxnId": "145", // Invoice Id
        }
      ]
    }
  ]
}'
```

```
"TxnType": "Invoice"  
}  
]  
}  
]  
}'
```



 Quickbooks Payment

Payment Amount

Customer

Customer RefNo

Create Payment

As we have to store the details within Salesforce, here are the fields that you need to create for the **QuickBooks Payment object**

Note:- Please create only the fields that you want to store on Salesforce.

Custom Object
Quickbooks Payment

[Help for this Page](#)

Standard Fields (8) | Custom Fields & Relationships (12) | Validation Rules (0) | Page Layouts (1) | Field Sets (0) | Connected Layouts (1) | Search Layouts (0) | Buttons, Links, and Actions (0) | Record Types (0) | Apex Sharing Reasons (0) | Apex Sharing Recalculation (0) | Object Limits (10)

Custom Object Definition Detail

Singular Label	Quickbooks Payment	Description
Plural Label	Quickbooks Payments	Enable Reports
Object Name	qb_Payment	Track Activities
API Name	qb_Payment__c	Allow in Chatter Groups
		Allow Sharing
		Allow Bulk API Access
		Allow Streaming API Access
		Track Field History
		Enable Licensing
		Deployment Status
		Allow Search
		Help Settings
		Modified By

Created By Admin User, 10/23/2020, 1:14 AM

Standard Fields

Action	Field Label	Field Name	Data Type	Controlling Field	Indexed
Edit	Created By	CreatedBy	Lookup(User)		
Edit	Currency	CurrencyIsoCode	Picklist		
Edit	Last Modified By	LastModifiedBy	Lookup(User)		
Edit	Owner	Owner	Lookup(User,Group)		✓
Edit	Payment #	Name	Text(80)		✓

Custom Fields & Relationships

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit Del	CreateTime	CreateTime__c	Text(255)			Admin User, 10/23/2020, 2:23 AM
Edit Del	Currency	Currency__c	Text(255)			Admin User, 10/23/2020, 2:43 AM
Edit Del	Currency_Code	CurrencyCode__c	Text(4)			Admin User, 10/23/2020, 3:02 AM
Edit Del	Customer Id	CustomerId__c	Text(90)			Admin User, 10/23/2020, 1:20 AM
Edit Del	Customer Name	CustomerName__c	Text(255)			Admin User, 10/23/2020, 1:24 AM
Edit Del	domain	domain__c	Text(10)			Admin User, 10/23/2020, 2:11 AM
Edit Del	LastUpdatedTime	LastUpdatedTime__c	Text(255)			Admin User, 10/23/2020, 2:27 AM
Edit Del	Payment Id	Id__c	Text(255) (External ID)	✓		Admin User, 10/23/2020, 2:17 AM
Edit Del	PaymentProcess	ProcessPayment__c	Checkbox			Admin User, 10/23/2020, 2:04 AM
Edit Del	Total Amount	TotalAmt__c	Currency(16, 2)			Admin User, 10/23/2020, 1:28 AM
Edit Del	Transaction Date	TxnDate__c	Date			Admin User, 10/23/2020, 2:36 AM
Edit Del	Unapplied Amount	UnappliedAmt__c	Currency(16, 2)			Admin User, 10/23/2020, 1:34 AM

Google Calendar Integration

Google Calendar is used to create online meetings and send invites to people.

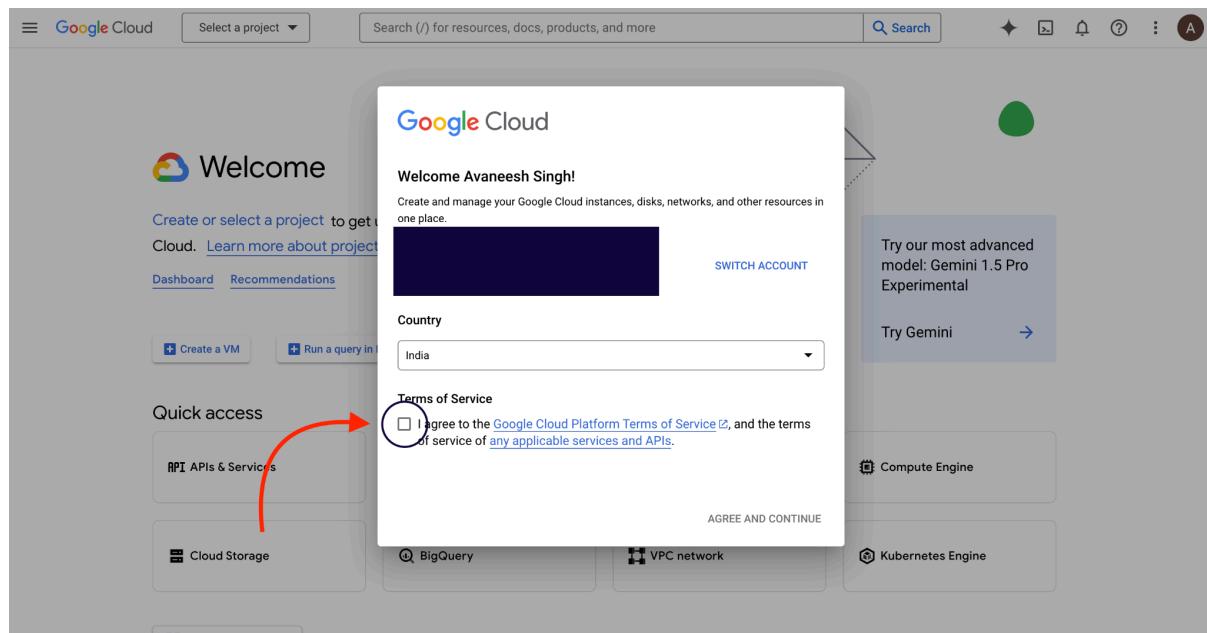
- Create a VF page in Salesforce
- Read the API Document and find out the authentication scheme. You can get the authentication document [Here](#).
- Setup connected application in Google Console
- Read more about Google Calendar API [Here](#).
- Know more about [Google OAuth scope](#)
- Create a Test Event/Calendar from the Google Console

Create Connected APP

Login to Google Console

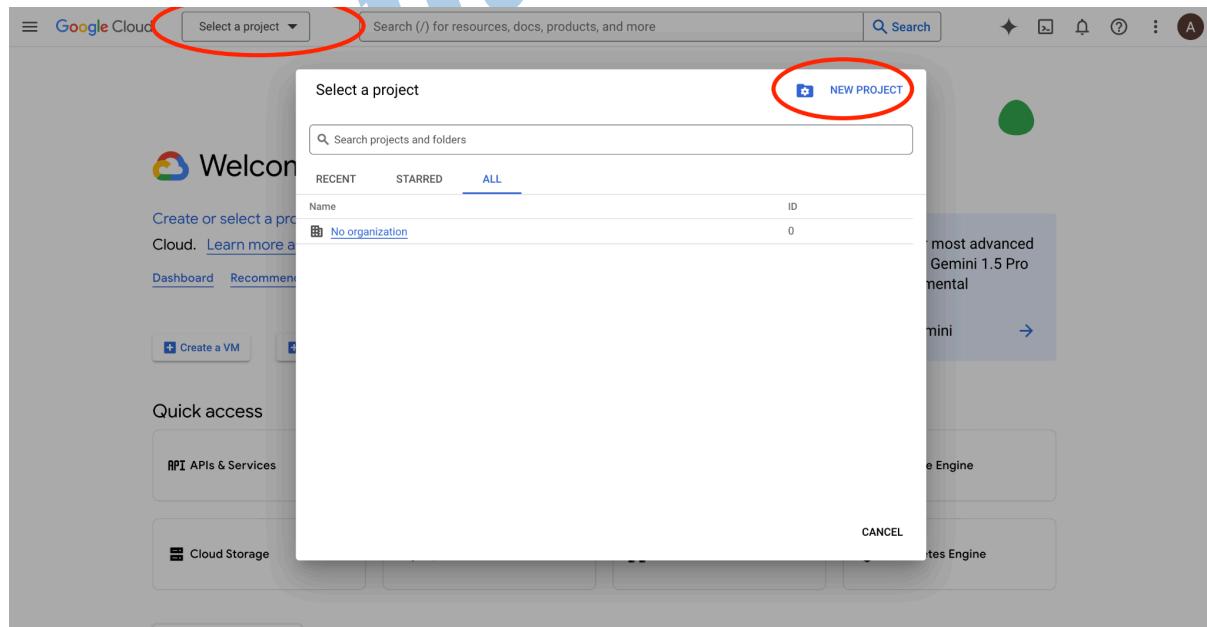
- Login to Google Console using [Welcome – Google Cloud console](#) Link

- If you are logging in for the first time then you will be asked to accept the Terms & Conditions, Please check the checkbox and click on **Agree and Continue**



Create a New Project

Click on the **Select a Project** Dropdown and then Click the **New Project** button



Give the **Project Name** and Click on **Create**. You will get the Notification

when your project is ready. When the Project is ready Click on **Select Project** Button

The screenshot shows the Google Cloud 'New Project' creation interface. At the top, there is a quota warning: "You have 12 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)". Below it, the 'Project name *' field contains "PantherSchools". A red arrow points to this field. The 'Location *' field shows "No organization" with a "BROWSE" button. A red circle highlights the "CREATE" button. The bottom right of the interface features icons for help, dismiss, and start free, along with a user profile icon.

Project ID: pantherschools-420912. It cannot be changed later. [EDIT](#)

Location * [BROWSE](#)
No organization

CREATE CANCEL

DISMISS START FREE

A red arrow points from the text "Create Project: PantherSchools" in the notification to the "SELECT PROJECT" button.

Notifications

Create Project: PantherSchools Just now

SELECT PROJECT

This is how your Project Workspace will look like

The screenshot shows the Google Cloud Platform dashboard for the project "PantherSchools". On the left sidebar, under "PINNED PRODUCTS", the "APIs & Services" item is highlighted with a blue background. A tooltip over this item says "Now viewing project 'PantherSchools' in organization 'No organization'". The main content area displays "Project info" and "Resources" sections, along with monitoring and error reporting status boxes.

From the left-hand side under the **Pinned Project** Click on the arrow next to **APIs & Services**

And then Click on "**OAuth consent screen**"

The screenshot shows the same Google Cloud Platform dashboard as before, but with a red circle highlighting the "APIs & Services" item in the left sidebar. A red arrow points to the small arrow icon next to it, indicating to "Open submenu". A green oval highlights the "OAuth consent screen" option in the dropdown menu that appears. The URL at the bottom of the screen is <https://console.cloud.google.com/apis/authuser=4&organizationId=0&project=pantherschools-420912>.

Select **External** from the available Radio Buttons and then Click **Create**

API APIs & Services	OAuth consent screen
<input type="checkbox"/> Enabled APIs & services <input type="checkbox"/> Library <input type="checkbox"/> Credentials <input checked="" type="checkbox"/> OAuth consent screen <input type="checkbox"/> Page usage agreements	<p>Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.</p> <p>User Type</p> <p><input type="radio"/> Internal <small>?</small></p> <p>Only available to users within your organization. You will not need to submit your app for verification. Learn more about user type</p> <p><input checked="" type="radio"/> External <small>?</small></p> <p>Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. Learn more about user type</p> <p>CREATE</p> <p>Let us know what you think about our OAuth experience</p>

Provide the App Name, Select the Email from the available List and then select the Logo if you have one

API APIs & Services	Edit app registration
<input type="checkbox"/> Enabled APIs & services <input type="checkbox"/> Library <input type="checkbox"/> Credentials <input checked="" type="checkbox"/> OAuth consent screen <input type="checkbox"/> Page usage agreements	<p>1 OAuth consent screen — 2 Scopes — 3 Test users — 4 Summary</p> <p>App information</p> <p>This shows in the consent screen, and helps end users know who you are and contact you</p> <p>App name * <input type="text" value="PantherSchools"/></p> <p>The name of the app asking for consent</p> <p>For users to contact you with questions about their consent. Learn more</p> <p>App logo</p> <p>This is your logo. It helps people recognize your app and is displayed on the OAuth consent screen.</p> <p>After you upload a logo, you will need to submit your app for verification unless the app is configured for internal use only or has a publishing status of "Testing". Learn more</p> <p>App logo preview</p> <p>Logo file to upload <input type="text" value="Panther Schools_icon-01.png"/> BROWSE</p>

Scroll to the Bottom, Provide the Developer Email and then Click on “**Save And Continue**”

Authorized domains ?

When a domain is used on the consent screen or in an OAuth client's configuration, it must be pre-registered here. If your app needs to go through verification, please go to the [Google Search Console](#) to check if your domains are authorized. [Learn more](#) about the authorized domain limit.

[+ ADD DOMAIN](#)

Developer contact information

Email addresses *

These email addresses are for Google to notify you about any changes to your project.

[SAVE AND CONTINUE](#)

CANCEL

Again Click on “**Save And Continue**”



🔒 Your restricted scopes

Restricted scopes are scopes that request access to highly sensitive user data.

API ↑

Scope

User-facing description

No rows to display

[SAVE AND CONTINUE](#)

CANCEL

Again Click on “**Save And Continue**”



Test users

While publishing status is set to "Testing", only test users are able to access the app. Allowed user cap prior to app verification is 100, and is counted over the entire lifetime of the app. [Learn more](#)

[+ ADD USERS](#)

Filter Enter property name or value



User information

No rows to display

[SAVE AND CONTINUE](#)

CANCEL

Then from the final screen Click on **Back to Dashboard**

Test users

0 users (0 test, 0 other) / 100 user cap



Filter

Enter property name or value



User information

No rows to display

[BACK TO DASHBOARD](#)

Now, Publish the Application by clicking on "**Publish App**"

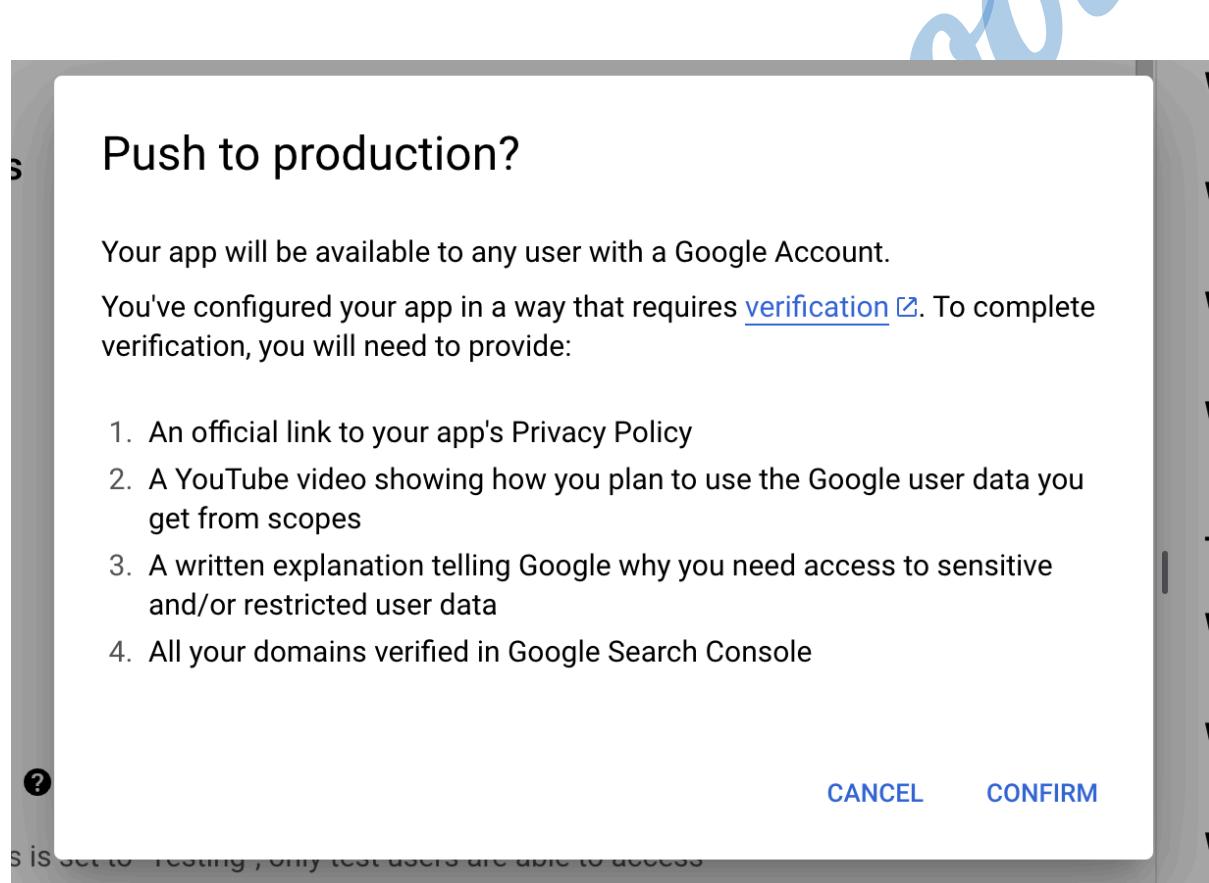
OAuth consent screen

PantherSchools EDIT APP

Publishing status

Testing

[PUBLISH APP](#)



Create Credentials in the Google Console

Once you have created the “**OAuth consent screen**”, it’s time for you to create the Credentials so that you can get the client ID and client secret

While you are on the **APIs & Services** Page, Click on **Credentials** from the

Left Side.

The screenshot shows the Google Cloud Platform API & Services Credentials page. The 'Credentials' tab is active. At the top right, there are buttons for '+ CREATE CREDENTIALS', 'DELETE', and 'RESTORE DELETED CREDENTIALS'. A green circle highlights the '+ CREATE CREDENTIALS' button. Below it, a warning message states: 'To protect you and your users, your consent screen and application need to be verified by Google.' A 'CONFIGURE CONSENT SCREEN' button is also present. The page is divided into sections: 'API Keys', 'OAuth 2.0 Client IDs', and 'Service Accounts'. Each section has a table with columns like Name, Creation date, Restrictions, and Actions. The 'OAuth 2.0 Client IDs' section has a note: 'No OAuth clients to display'.

Click on “+Create Credentials” and Select “**OAuth client ID**” from the dropdown

The screenshot shows the same Google Cloud Platform API & Services Credentials page, but the 'Help me choose' option in the dropdown is now highlighted with a red arrow pointing to it. This step asks a few questions to help decide which type of credential to use. The rest of the interface remains the same, with the '+ CREATE CREDENTIALS' button still highlighted.

- For the Application Type select “**Web application**”
- Provide the Name of the **Application**
- Skip the “**Authorized redirect URIs**” Step we will add it later
- Click on **Create** Button
- Copy and Store the **Client ID and Client Secret** at a place where you can access it easily.

[←](#) Create OAuth client ID

A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See [Setting up OAuth 2.0](#) for more information. [Learn more](#) about OAuth client types.

Application type *
Web application

Name *
PantherSchools

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

i The domains of the URIs you add below will be automatically added to your [OAuth consent screen](#) as [authorized domains](#).

Authorized JavaScript origins [?](#)

For use with requests from a browser

[+ ADD URI](#)

Authorized redirect URIs [?](#)

For use with requests from a web server

[+ ADD URI](#)

Note: It may take 5 minutes to a few hours for settings to take effect

[CREATE](#)

[CANCEL](#)



[Learn more ↗](#)

OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services

i OAuth is limited to 100 [sensitive scope logins](#) until the [OAuth consent screen](#) is verified. This may require a verification process that can take several days.

Client ID	
Client secret	
Creation date	
Status	

[!\[\]\(63f48b0b1f7a4cd171bcc6e44599cc42_img.jpg\) DOWNLOAD JSON](#)

[OK](#)

Test Google API using Postman

Below are the steps that we will perform to get the token for Google Calendar

- Get the Authorization Code
- Get the access token using Postman

- List All the events for the authenticated account
- Create a simple event using Postman

Create Event From Salesforce to Google Calendar

We are going to create the Google Calendar event from Salesforce using two different ways

- Static Event for testing purposes
- Send the Salesforce Event object data to Google Calendar

Assignments

#1

Authorise the Google API using Salesforce, here are the steps outlined for the same.

1. Add Remote Site Setting
2. Create a Custom Metadata to Store Google Credentials and Other Details
3. Create a Custom Label to store the MasterLabel of Custom Metadata Record
4. Make changes to the VF Page that we create in an earlier lecture
5. Prepare the URL and get the authorization code
6. Get the access token and store it in Metadata Object
7. Implement the Refresh Token method
8. Create a new Method in Apex Class to create the Event in Google Calendar

The **Custom metadata** can look like below

Custom Metadata Type
Google Config

Standard Fields [0] | Custom Fields [12] | Validation Rules [0] | Page Layouts [1]

[Edit](#) [Delete](#) [Manage Google Configs](#)

Singular Label	Google Config	Description	
Plural Label	Google Configs	Visibility	Public
Object Name	Google_Config	Protection Level	2.711
API Name	Google_Config_mdt	Record Size	2.711
Created By	Admin User, 10/24/2020, 8:20 AM	Modified By	Admin User, 10/24/2020, 8:20 AM

Standard Fields

Action	Field Label	Field Name	Data Type	Indexed
Edit Del	Created By	CreatedBy	Lookup(User)	
Edit Del	Custom Metadata Record Name	DeveloperName	Text(40)	
Edit Del	Label	MasterLabel	Text(40)	
Edit Del	Last Modified By	LastModifiedBy	Lookup(User)	
Edit Del	Namespace Prefix	NamespacePrefix	Text	
Edit Del	Protected Component	IsProtected	Checkbox	

Custom Fields

Action	Field Label	API Name	Data Type	Field Manageability	Indexed	Controlling Field	Modified By
Edit Del	access_token	access_token_c	Long Text Area(32768)	Upgradable			Admin User, 10/25/2020, 4:09 AM
Edit Del	auth_url	auth_url_c	URL(255)	Upgradable			Admin User, 10/25/2020, 4:46 AM
Edit Del	Base Url	Base_URL_c	URL(255)	Upgradable			Admin User, 10/25/2020, 4:13 AM
Edit Del	client_id	client_id_c	Text(255)	Upgradable			Admin User, 10/25/2020, 4:10 AM
Edit Del	client secret	client_secret_c	Text(255)	Upgradable			Admin User, 10/25/2020, 4:11 AM
Edit Del	Event Url	Event_URL_c	URL(255)	Upgradable			Admin User, 10/25/2020, 4:14 AM
Edit Del	expires in (seconds)	expires_in_c	Number(10, 0)	Upgradable			Admin User, 10/25/2020, 4:15 AM
Edit Del	expires in time	expires_in_time_c	Date/Time	Upgradable			Admin User, 10/25/2020, 4:16 AM
Edit Del	PageName	PageName_c	Text(255)	Upgradable			Admin User, 10/25/2020, 4:12 AM
Edit Del	refresh token	refresh_token_c	Long Text Area(32768)	Upgradable			Admin User, 10/25/2020, 4:10 AM
Edit Del	scope	scope_c	Long Text Area(32768)	Upgradable			Admin User, 10/25/2020, 4:17 AM
Edit Del	token url	token_url_c	URL(255)	Upgradable			Admin User, 10/25/2020, 4:47 AM

#2

Create a Lightning Web/Aura Component which will take the necessary input from the users of an Event When the user clicks on Create Event that event must be created in Salesforce and then into Google Calendar as well.

Hint:- You only need to create the event in Salesforce and the rest logic will be taken care of by the apex trigger. The component should look like the one below.

 Google Event

Subject

Start Date
Date Time

End Date
Date Time

Create Event

Introduction to LinkedIn

LinkedIn is a professional network where you can make your professional network for your career growth.

1. Create a Free LinkedIn developer account using your existing LinkedIn account. [Here](#) is the link.
2. Read the API Document and find out the authentication scheme. You can get the authentication document [Here](#).
3. Set up the connected application in the LinkedIn developer account.
4. Read more about LinkedIn API

<https://learn.microsoft.com/en-us/linkedin/shared/authentication/authentication>

`https://www.linkedin.com/oauth/v2/authorization?response_type=code&client_id=YOUR_CLIENT_ID&redirect_uri=https://login.salesforce.com/services/oauth2/success&scope=r_liteprofile%20r_emailaddress%20w_member_social`

In the above URL replace the Client ID and Redirect Uri with the original values that are related to you.

Implement LinkedIn Integration with Salesforce

Here are the steps that we are going to perform to test if the Integration is working or Not.

- **Get basic User details** - We have done this in our previous post.
 - Here is the Link to the Profile API -
<https://api.linkedin.com/v2/me>
- **Create a Post to your LinkedIn profile using API**
 - Here is the Link to the API that we will be referring to create a post linked under the authenticated account
 - <https://learn.microsoft.com/en-us/linkedin/marketing/integrations/community-management/shares/posts-api?view=li-lms-2023-04&tabs=http>

Assignments

#1

Authenticate LinkedIn using Salesforce, follow the below-outlined steps to get the access token

1. Add Remote Site Setting
2. Create a Custom Metadata to Store LinkedIn Credentials and Other Details
3. Create a Custom Label to store the MasterLabel of Custom Metadata Record
4. Make changes to the VF Page that we create in an earlier lecture
5. Prepare the URL and get the authorization code
6. Get the access token and store it in the Metadata Object

Note: - LinkedIn does not support the refresh token flow as of now for the free developer account, so you can skip that part.

Here are the custom metadata details

Custom Metadata Type		LinkedIn Config			
		Help for this Page 			
Custom Metadata Type Detail		Standard Fields (8) Custom Fields (1) Validation Rules (0) Page Layouts (1)			
Singular Label	LinkedIn Config		Description		
Plural Label	LinkedIn Configs		Visibility	Public	
Object Name	LinkedIn_Config		Protection Level		
API Name	LinkedIn_Config_mdt		Record Size	3,156	
Created By	Admin User, 10/24/2020, 10:17 AM		Modified By	Admin User, 10/24/2020, 10:17 AM	
Standard Fields					
Action	Field Label	API Name	Field Name	Data Type	Indexed
Edit Del	Created By		CreatedBy	Lookup(User)	
Edit	Custom Metadata Record Name		DeveloperName	Text(40)	
Edit Del	Label		MasterLabel	Text(40)	
Edit Del	Last Modified By		LastModifiedBy	Lookup(User)	
Edit Del	Namespace Prefix		NamespacePrefix	Text	
Edit Del	Protected Component		IsProtected	Checkbox	
Custom Fields					
New					
Action	Field Label	API Name	Data Type	Field Manageability	Indexed
Edit Del	access_token	access_token_c	Long Text Area(32768)	Upgradable	Controlling Field
Edit Del	auth_url	auth_url_c	URL(255)	Upgradable	Modified By
Edit Del	client_id	client_id_c	Text(255)	Upgradable	Admin User, 10/28/2020, 4:02 AM
Edit Del	client_secret	client_secret_c	Text(255)	Upgradable	Admin User, 10/28/2020, 3:58 AM
Edit Del	comment_url	commenturl_c	URL(255)	Upgradable	Admin User, 10/28/2020, 3:59 AM
Edit Del	Create Post	Create_Post_c	URL(255)	Upgradable	Admin User, 10/30/2020, 4:05 AM
Edit Del	expires_in (seconds)	expires_in_c	Number(10, 0)	Upgradable	Admin User, 10/28/2020, 4:09 AM
Edit Del	expires_in_time	expires_in_time_c	Date/Time	Upgradable	Admin User, 10/28/2020, 4:03 AM
Edit Del	like_url	likeurl_c	URL(255)	Upgradable	Admin User, 10/30/2020, 4:06 AM
Edit Del	Linked Person Id	LinkedPersonId_c	Text(100)	Upgradable	Admin User, 10/30/2020, 4:27 AM
Edit Del	PageName	PageName_c	Text(90)	Upgradable	Admin User, 10/28/2020, 3:59 AM
Edit Del	Profile Url	Profile_Url_c	URL(255)	Upgradable	Admin User, 10/28/2020, 4:07 AM
Edit Del	refresh_token	refresh_token_c	Long Text Area(32768)	Upgradable	Admin User, 10/28/2020, 4:01 AM
Edit Del	scope	scope_c	Long Text Area(32768)	Upgradable	Admin User, 10/28/2020, 4:01 AM
Edit Del	token_url	token_url_c	URL(255)	Upgradable	Admin User, 10/28/2020, 4:02 AM

#2

1. Create a Custom object “**LinkedIn Post**” with the following fields.
2. Develop a trigger when you create any **LinkedIn Post** record, it should create the post in LinkedIn as well.
3. Find the fields below

Below are the field details for the object.

Standard Fields						Standard Fields Help 
Action	Field Label	Field Name	Data Type	Controlling Field	Indexed	
Edit Del	Created By	CreatedBy	Lookup(User)			
Edit	Currency	CurrencyIsoCode	Picklist			
Edit Del	Last Modified By	LastModifiedBy	Lookup(User)			
Edit	Owner	Owner	Lookup(User,Group)		<input checked="" type="checkbox"/>	
Edit	Title	Name	Text(80)		<input checked="" type="checkbox"/>	

Custom Fields & Relationships							Custom Fields & Relationships Help 
Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By	
Edit Del	Content	Content_c	Rich Text Area(32768)			SFDC Panther, 10/30/2020, 3:58 AM	
Edit Del	Subject	Subject_c	Text(255)			SFDC Panther, 10/30/2020, 3:58 AM	
Edit Del	Supporting Url	Supporting_Url_c	URL(255)			SFDC Panther, 10/30/2020, 3:59 AM	
Edit Del	Thumbnail Image Url	Thumbnail_Image_Url_c	URL(255)			SFDC Panther, 10/30/2020, 4:00 AM	

#3

Modify the Apex class that is responsible for integrating Salesforce with LinkedIn and add the two additional methods to it.

- Add a method that will add the comment under the created post.

- Create a Custom Object “**LinkedIn Post Comment**” with just one field Comment and the data type can be text area.
- Once the comment is created under any **LinkedIn** post then it should send the same to LinkedIn under the same comment.
- Add another method that will be like the created post.
 - Whenever the comment is posted on LinkedIn, the same process should be like the same parent post.

More Apis to Work upon

1. JIRA
2. ZERO
3. <https://squareup.com>
4. Currency
5. Finance
6. IMDB API (Check if there is one)
7. Spotify
8. Twitter
9. Instagram
10. AWS
11. DropBox
12. SharePoint
13. Google Drive

Named Credentials in Salesforce

Named credentials are used to make the API callout and simplify the process of storing the credentials securely. When we used named credentials it gave a better way to handle -

- Need not create the Remote Site Setting
- No need to worry about Refresh Token if the target system does

return the Access Token

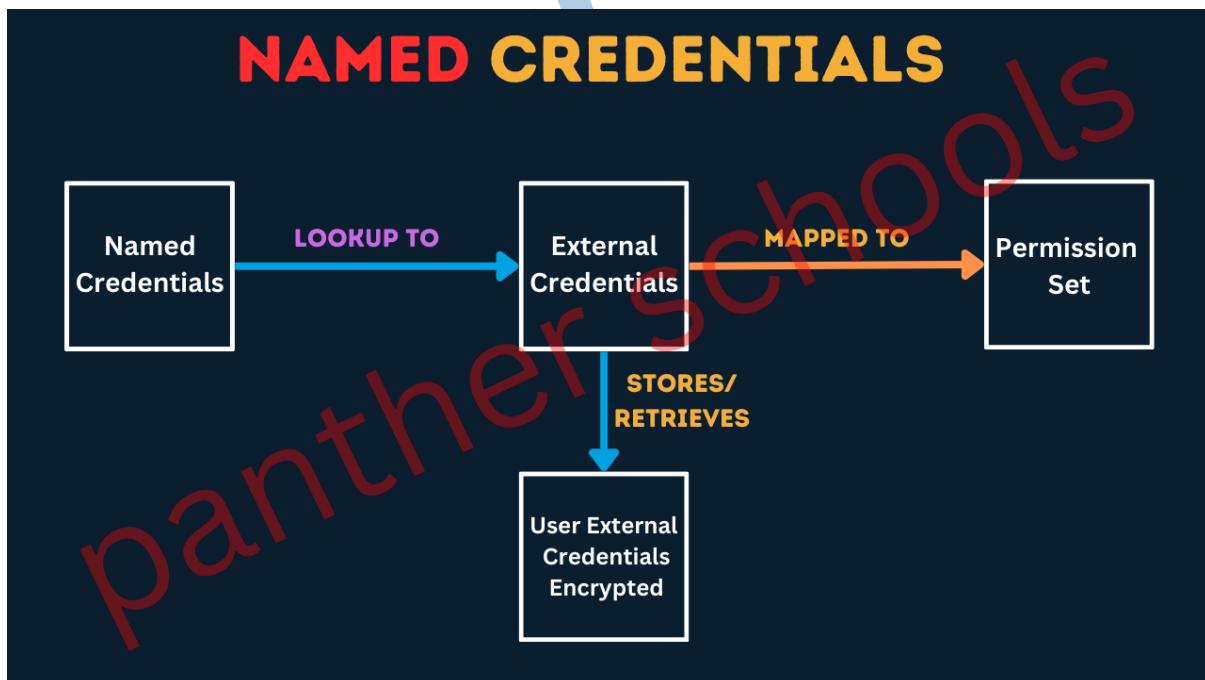
- A more secure way to store the credentials and access token than storing the details over the custom metadata

If we have named credentials created we can refer to the same in our code like:-

callout:My_Named_Credential/some_path

You can read more about named credentials [Here](#).

- https://help.salesforce.com/s/articleView?id=sf.named_credentials_about.htm&type=5
- https://help.salesforce.com/s/articleView?id=sf.nc_custom_headers_basic_auth.htm&type=5
- https://help.salesforce.com/s/articleView?id=sf.nc_named_creds_formula_functions.htm&type=5

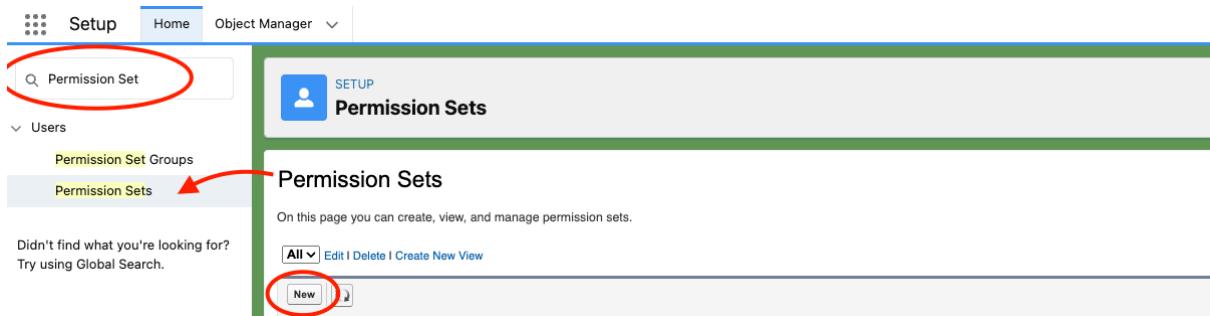


Create Permission Set

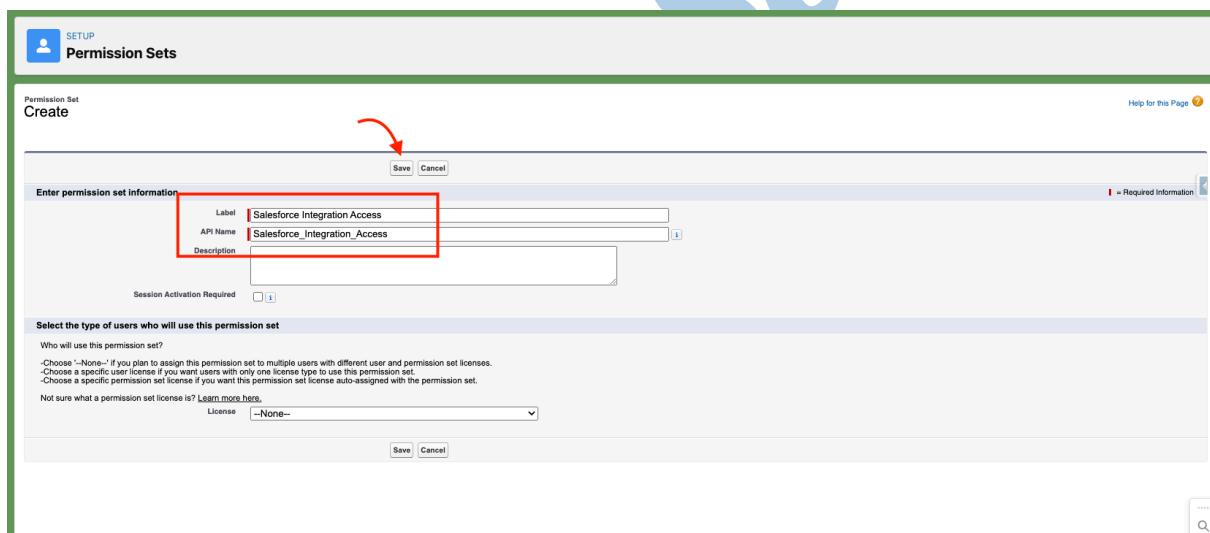
Before we create Named Credentials and External Credentials, we need to

create the Permission Set and this permission set will be used to provide access to External Credentials so that the User can easily use Named Credentials for making the API Callouts.

- Login to Salesforce
- Navigate to Setup → Permission Sets → New



Provide the Name of the Application and Click Save to Save the permission Set



Provide “User External Credentials” Object Access to Permission Set

User External Credentials objects are the key object which is used to provide access to External Credentials and those credentials are used to make the API Calls.

While you are on the Permission Set Detail Page Click on “**Object Settings**” to navigate to the list of Objects

Permission Set
User External Credentials Object Access

API Name UserExternalCredentialsObjectAccess
Namespace Prefix
Created By Amit Singh, 4/21/2024, 1:26 AM
Last Modified By Amit Singh, 4/21/2024, 9:30 AM

Permission Set Overview

Description	API Name
License	UserExternalCredentialsObjectAccess
Session Activation Required	<input type="checkbox"/>
Permission Set Groups Added To	0

Apps

- Assigned Apps
- Assigned Connected Apps
- Object Settings
- App Permissions
- Apex Class Access

Search for the “**User External Credentials**” Object and Click on it

Shipments	Shipment
Signature Task Line Items	SignatureTaskLineItem
Signature Tasks	SignatureTask
Solutions	Solution
SOS Sessions	SOSSession
Stores	WebStore
Streaming Channels	StreamingChannel
Subscriptions	ContentSubscriptions
Tasks	Task
Token Project	01rHu000001Zgva
Trainers	Trainer__c
Training Centers	TrainingCenter__c
User External Credentials	UserExternalCredential
User Provisioning Requests	UserProvisioningRequest
Users	User
Vehicle User Assignments	VehicleUserAssignment
Visited Parties	VisitedParty
Visitors	Visitor
Visits	Visit
Web Cart Documents	WebCartDocument
Work Order Line Items	WorkOrderLineitem
Work Orders	WorkOrder

Provide the following permissions and Click on Save

Permission Set
User External Credentials Object Access

Object Permissions

Permission Name	Enabled
Read	<input checked="" type="checkbox"/>
Create	<input checked="" type="checkbox"/>
Edit	<input checked="" type="checkbox"/>
Delete	<input checked="" type="checkbox"/>
View All	<input checked="" type="checkbox"/>
Modify All	<input checked="" type="checkbox"/>

Implement Named credentials in Salesforce

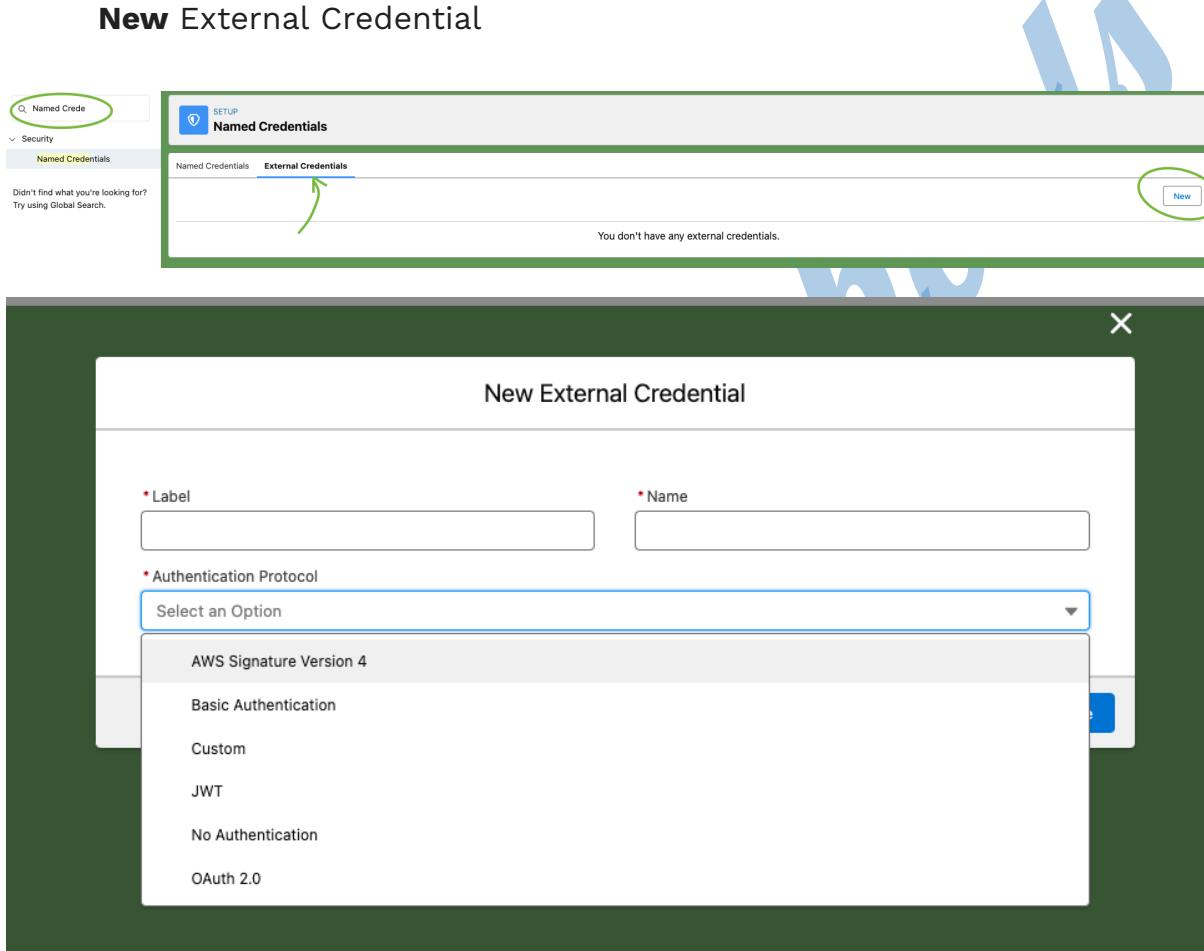
As we have talked about the named credentials in Salesforce, now let's use the Freshdesk implementation with Named Credentials and get rid of the

Custom Labels that we are using to authenticate for every request.

Create External Credentials

To create the External credentials follow the below steps

- Login to Salesforce
- Navigate to Setup → **Named Credentials** → **External Credentials** → **New External Credential**



Provide the label and Name (should not contain any space or special characters), For Authentication Protocol select Basic Authentication and Click on Save

The screenshot shows the 'NAMED CREDENTIALS' section in the Freshdesk setup. A single credential is listed with the following details:

- Label:** Freshdesk
- Name:** Freshdesk
- Authentication Protocol:** Basic Authentication
- Managed Package Access:** Not selected
- Created By Namespace:** [User]

Buttons at the top right include 'Edit' and 'Delete'.

Create Principals related to External Credentials

After you have created the External Credentials, Scroll down to the **Principals** section and click on **New** button

The screenshot shows the 'Principals' section. It displays a placeholder message: 'You don't have any principals configured.' There is a decorative graphic of mountains and clouds. A 'New' button is located in the top right corner, which is circled in red.

Provide the “**Parameter Name**” for “**Identity Type**” select “**Per User Principals**” and click on save

The screenshot shows the 'Create Principal' dialog box. It contains the following fields:

- * Parameter Name:** Support Users
- * Sequence Number:** 1
- * Identity Type:** Per User Principal

At the bottom right are 'Cancel' and 'Save' buttons.

Provide Access to Principals at the Permission Set Level

Navigate to the Permission Set that you have created in the previous step and Click on “**External Credential Principal Access**”

Permission Set
User External Credentials Object Access

Find Settings... | Clone | Delete | Edit Properties | Manage Assignments | View Summary (Beta) | Video Tutorial | Help for this Page

Permission Set Overview

Description	API Name	UserExternalCredentialsObjectAccess
License	Namespace Prefix	
Session Activation Required	Created By	Amit Singh, 4/21/2024, 1:26 AM
Permission Set Groups Added To	Last Modified By	Amit Singh, 4/21/2024, 9:45 AM

Apps

- Assigned Apps**
Settings that specify which apps are visible in the app menu
- Assigned Connected Apps**
Settings that specify which connected apps are visible in the app menu
- Object Settings**
Permissions to access objects and fields, and settings such as tab availability
- App Permissions**
Permissions to perform app-specific actions, such as "Manage Call Centers"
- Apex Class Access**
Permissions to execute Apex classes
- Visualforce Page Access**
Permissions to edit Visualforce pages
- External Data Source Access**
Permissions to authenticate against external data sources
- Flow Access**
Permissions to execute Flows
- Named Credential Access**
Permissions to authenticate against named credentials
- External Credential Principal Access**
Permissions to authenticate with external credential principal mappings
- Custom Permissions**
Permissions to access custom processes and apps
- Custom Metadata Types**
Permissions to access custom metadata types
- Custom Setting Definitions**
Permissions to access custom settings
- Organization-Wide Email Address Access**
Permissions to send email with organization-wide email address
- Standard Invocable Action Type Access**
Permissions to access invocable actions

Click on the “**Edit**” button add your Principal from **Available External Credential Principals** to **Enabled External Credential Principals** section and Click on Save.

Permission Set
User External Credentials Object Access

Find Settings... | Clone | Delete | **Edit Properties** | Manage Assignments | View Summary (Beta)

Permission Set Overview > External Credential Principal Access

External Credential Principal Access

Available External Credential Principals

Freshdesk - Support Users

Enabled External Credential Principals

—None—

Add  Remove 

Save | Close

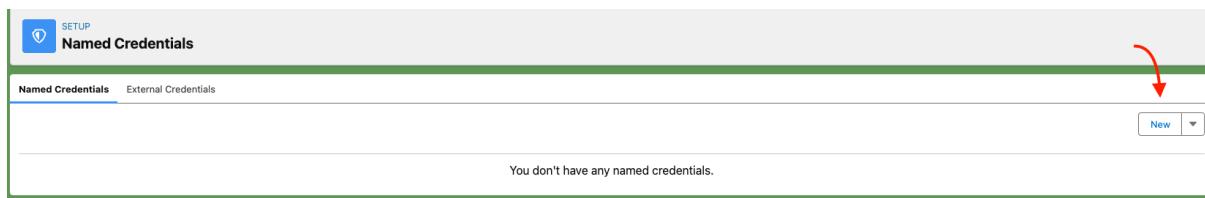
Assign this Permission Set to your User or to the user who is used for Making the External API Callouts.

Create Named Credentials

To create the Named credentials follow the below steps

- Login to Salesforce

- Navigate to Setup → **Named Credentials** → **Named Credentials** → **New**
Named Credential



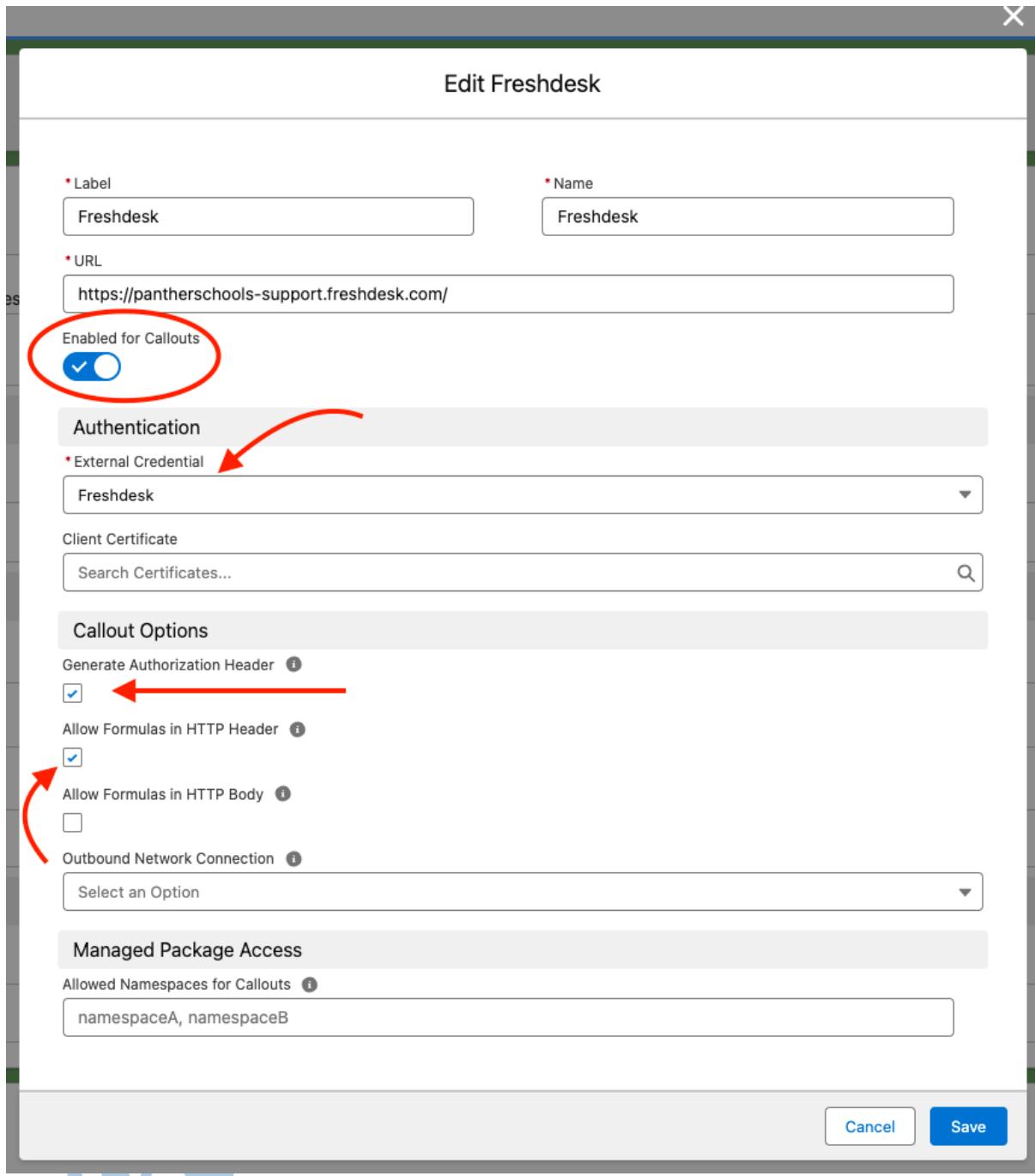
- Provide the Name & Label of the Named Credentials
- Provide the API base URL of your Freshdesk Org (If you are using any other API then use the correct Base URL)
- Make Sure “**Enabled for Callouts**” is enabled
- Select the appropriate External Credential
- Be sure that “**Generate Authorization Header**” is enabled under **Callout Option**
 - This will be disabled if you have selected “**Named Principals**” while creating the Principals in External Credentials

The screenshot shows a 'Create Principal' form. At the top center, it says 'Create Principal'. Below this, there are several input fields:

- * Parameter Name: An empty text input field.
- * Sequence Number: An input field containing the number '2'.
- * Identity Type: A dropdown menu set to 'Named Principal'. A red curved arrow points to this dropdown from the left side of the screen.
- * Username: An empty text input field.
- * Password: An empty text input field.

 At the bottom right of the form are two buttons: 'Cancel' and 'Save'.

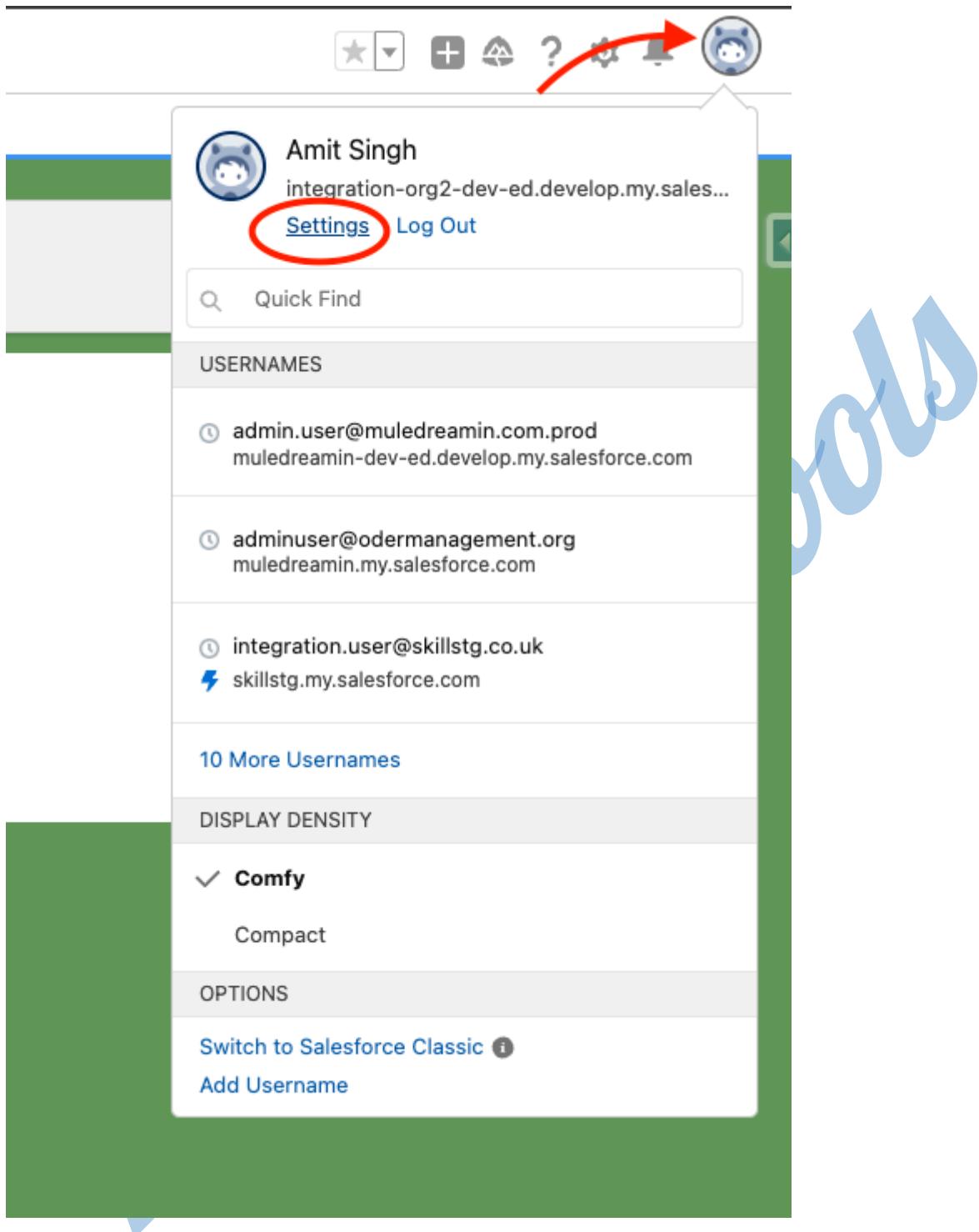
- In the named credential, be sure that **Allow Formulas in the HTTP Header** is enabled.
- Click on **Save**



Configure the External Credentials at the User Level (Optional)

You can skip this Step if you are using “**Named Principals**” while creating the Principals in External Credentials.

- Login to Salesforce
- Click on your User Icon and Select Settings



- Select “**External Credentials**” from the left-hand side
- Click on **Allow Access**
- Provide the username and Password

Quick Find

- My Personal Information
- Advanced User Details
- Approver Settings
- Authentication Settings for External Systems
- Change My Password
- Connections
- External Credentials**
- Grant Account Login Access
- Language & Time Zone
- Login History
- Personal Information

External Credentials

Click **Allow Access** to authenticate to an external organization.

Freshdesk

Freshdesk

Allow Access

Freshdesk

Enter your credentials to allow Salesforce to access this system on your behalf.

* Username

* Password 

Cancel **Allow Access**

You will see the success message there

External Credentials

Click **Allow Access** to authenticate to an external organization.

Freshdesk

Freshdesk

Revoke Access

Configured ✓

Use Named Credentials in Apex Class

To use the Named Credentials in Salesforce Apex Class you need to directly use the name of Named Credentials in the **setEndpoint** method.

Sample code below -

```
HttpRequest req = new HttpRequest();
// https://pantherschools-support.freshdesk.com
// ticket - Url/Endpoint - /api/v2/tickets
req.setEndpoint('callout:Freshworks/api/v2/tickets');
req.setHeader('Content-Type', 'application/json');
req.setHeader('Accept', 'application/json');
req.setMethod('POST');

String requestBody = '{ "description": "Named Credentials Demo",
"subject": "Named Credentials Demo", "email": "tom@outerspace.com", "priority": 1, "status": 2, "cc_emails": ["ram@freshdesk.com", "diana@freshdesk.com"] }';
req.setBody(requestBody);

HTTP HTTP = new HTTP();
HTTPResponse res = http.send(req);

System.debug(res.getBody());
System.debug(res.getStatusCode());
```

Authentication (Auth.) Provider in Salesforce

Auth. Providers are the **external service/identity providers** by which users can log in to Salesforce using external credentials like Google Email, Facebook, LinkedIn, Amazon, Salesforce, Okta &, etc. OR Users can log in to the External System using their Salesforce credentials.

Auth. Providers can also be used to integrate and make third-party callouts from Salesforce. You can also create your own auth. provider.

While using the Auth. Providers we do not need to develop the apex code for getting the access token and refresh token. Also, we do not need to create custom metadata to store the token information.

Benefits of using Auth. Providers

- Easy to maintain the users
- Required less workforce
- Can control users from one place
- Can control what users can do, see
- and many more

To Create the auth provider for any external system we need the following information

- Client Id
- Client Secret
- Access Token Url - **API Document**
- Authorization Url - **API Document**
- User Info Url - **Optional - API Document**
- List of Scope - **Optional - API Document**

The screenshot shows the Salesforce Setup interface with a search bar at the top left. Below it, a sidebar on the left lists 'Identity' and 'Auth. Providers'. The main content area is titled 'Auth. Providers' and contains a sub-section titled 'Auth. Provider'. A modal window titled 'Auth. Provider Edit' is open, showing fields for 'Provider Type' (set to 'Google'), 'Name' (empty), 'URL Suffix' (empty), 'Consumer Key' (empty), 'Consumer Secret' (empty), 'Authorize Endpoint URL' (set to 'https://accounts.google.com/o/oauth2/auth'), 'Token Endpoint URL' (set to 'https://accounts.google.com/o/oauth2/token'), and 'User Info Endpoint URL' (set to 'https://www.googleapis.com/oauth2/v3/userinfo'). Other fields like 'Default Scopes' and 'Custom Error URL' are also present. At the bottom of the modal, there are 'Save', 'Save & New', and 'Cancel' buttons.

Google Email as an Authentication Provider

1. Create an authentication provider from Google
2. Use Google to Login into Salesforce with Google Auth Provider.
3. Use Auth. Provider to Create Calendar Event in Google Calendar (Named Credentials)
4. Per User Principals

Create an authentication provider from Google

To Create the Auth. Provider, we need to follow the below steps

- Login to Salesforce
- Navigate to Setup → Auth. Providers → Click on the New Button
- Provide the necessary information like client ID & client secret
 - We already have this information as part of previous exercises.
 - https://accounts.google.com/o/oauth2/v2/auth?prompt=consent&access_type=offline
 - **Authorise Url -**
<https://accounts.google.com/o/oauth2/v2/auth?prompt=consent>

&access_type=offline

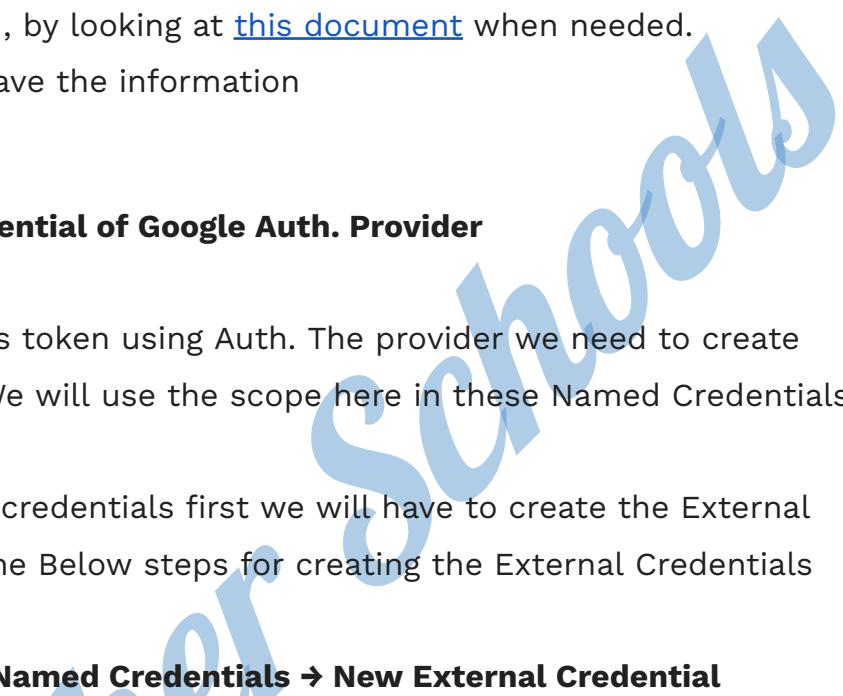
- **Scopes** - openid <https://www.googleapis.com/auth/calendar> <https://www.googleapis.com/auth/calendar.events>
 - We are integrating the Google Calendar we have just added the scope for Google Calendar Only
 - You can add more scopes like for Google Drive or YouTube API, by looking at [this document](#) when needed.
- Click Save to save the information

Create a Named credential of Google Auth. Provider

For getting the access token using Auth. The provider we need to create named credentials. We will use the scope here in these Named Credentials.

To create the named credentials first we will have to create the External Credentials. Follow the Below steps for creating the External Credentials

Navigate to Setup → Named Credentials → New External Credential



New External Credential

* Label	MyZoom	* Name	MyZoom
* Authentication Protocol	OAuth 2.0	Scope add additional scopes here	
* Authentication Flow Type	Browser Flow		
* Authentication Provider	MyZoom	Cancel Save	

A blue arrow points from the 'Label' field to the 'MyZoom' input. A blue circle highlights the 'OAuth 2.0' selection in the 'Authentication Protocol' dropdown. A blue arrow points from the 'Authentication Provider' section to the 'MyZoom' input.

After you have created the External Credentials and while you are on the detail page of the External Credentials you need to create the PrinciPal. (Please refer to the Previous demo for Freshdesk)

Create Principal

* Parameter Name GoogleCalendarUsers	* Sequence Number 1
* Identity Type Per User Principal	Scope
	
Cancel Save	

Once the Principal is created, you need to grant the access at permission set level.

Navigate to the Permission Set that you have created in the previous step and Click on “**External Credential Principal Access**”

Permission Set
User External Credentials Object Access

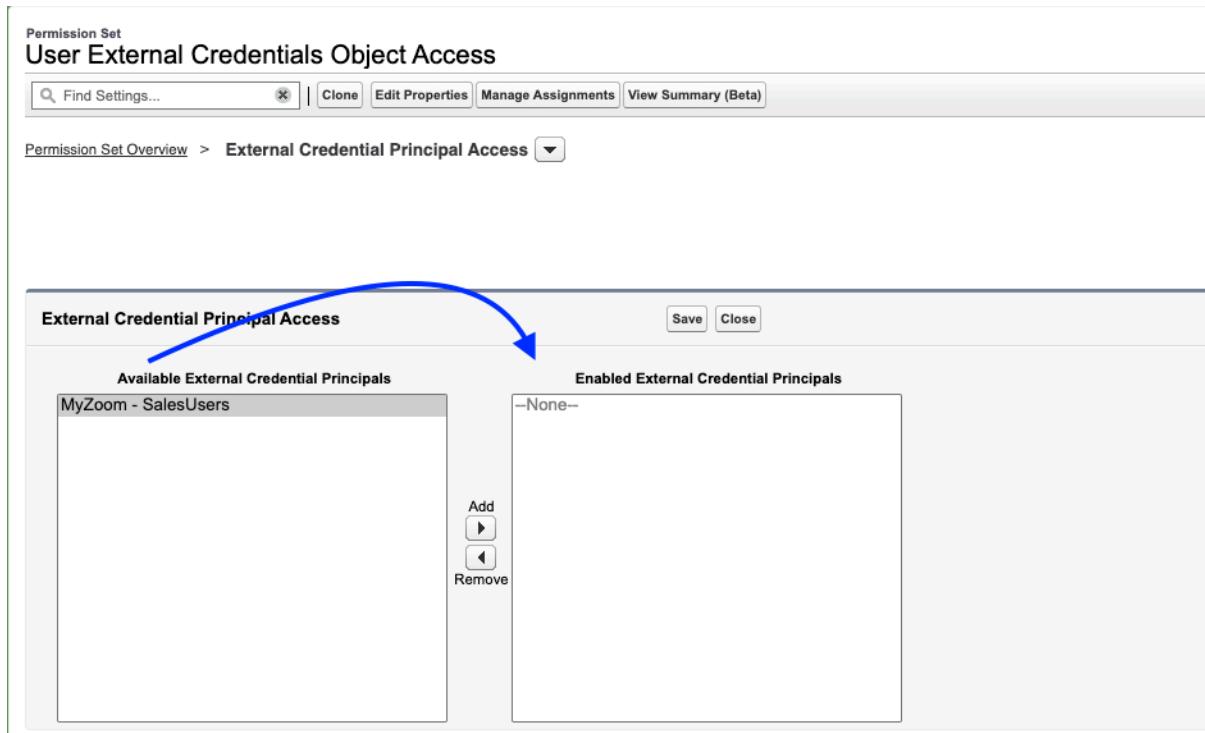
API Name: UserExternalCredentialsObjectAccess
Namespace Prefix:
Created By: Amit Singh, 4/21/2024, 1:26 AM
Last Modified By: Amit Singh, 4/21/2024, 9:45 AM

Apps

Assigned Apps
Assigned Connected Apps
Object Settings
App Permissions
Apex Class Access
Visualforce Page Access
External Data Source Access
Flow Access
Named Credential Access
External Credential Principal Access Permissions to authenticate with external credential principal mappings

Custom Permissions
Custom Metadata Types
Custom Setting Definitions
Organization-Wide Email Address Access
Standard Invocable Action Type Access

Click on the “**Edit**” button add your Principal from **Available External Credential Principals** to **Enabled External Credential Principals** section and Click on Save.



After you have completed the above two steps, now the time is to create the Named Credentials

Navigate to Setup → Named Credentials → New Named Credential

New Named Credential

* Label

* Name

* URL

Enabled for Callouts

Authentication

* External Credential

Client Certificate 🔍

Callout Options

Generate Authorization Header i

Allow Formulas in HTTP Header i

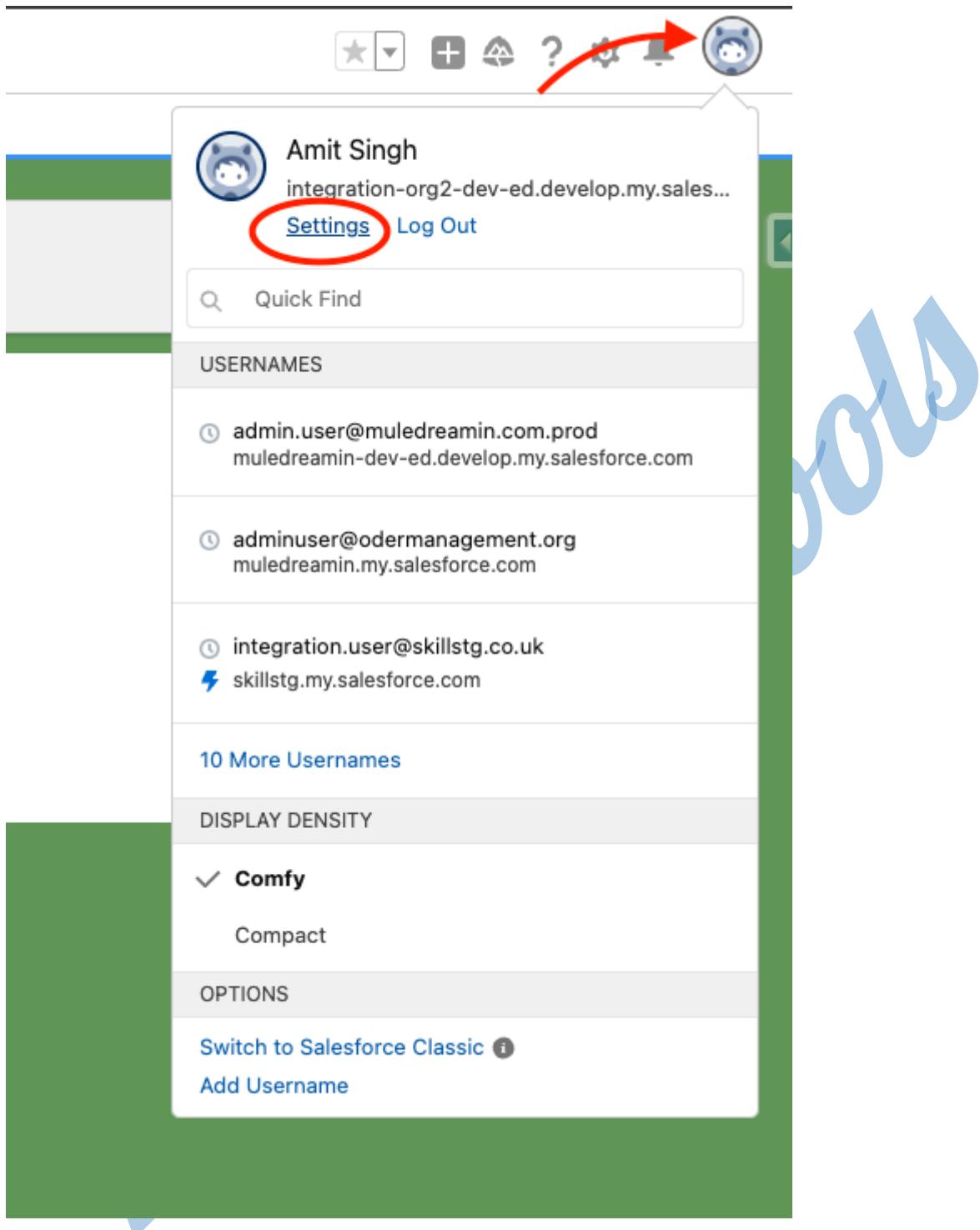
Allow Formulas in HTTP Body i

Cancel Save

Configure the External Credentials at the User Level (Optional)

You can skip this Step if you are using “**Named Principals**” while creating the Principals in External Credentials.

- Login to Salesforce
- Click on your User Icon and Select Settings



- Select “**External Credentials**” from the left-hand side
- Click on **Allow Access**
- Provide the username and Password

The screenshot shows the 'External Credentials' section of a user profile. On the left, a sidebar lists various account settings. The 'External Credentials' option is highlighted with a red oval and a blue underline. The main content area displays a 'Freshdesk' connection with a 'Allow Access' button. A red arrow points from the text 'Click Allow Access to authenticate to an external organization.' to the 'Allow Access' button.

The screenshot shows the 'External Credentials' section of a user profile. The sidebar includes the 'External Credentials' option, which is underlined in blue. The main content area displays a 'MyZoom' connection with a 'Google' entry and an 'Allow Access' button. A blue arrow points from the text 'Click Allow Access to authenticate to an external organization.' to the 'Allow Access' button.

If you get the below Error that means you have not assigned the permission set to your user or the Principal access is missing.



If everything is right then you will be redirected to the Login Url and will be asked to grant the access



PantherSchools would like permission to:

- ✓ Enable Zoom App within Zoom Meeting Client
- ✓ Create a meeting for a user
- ✓ View a user
- ✓ Verify a user's email
- ✓ Create polls for a meeting
- ✓ Add registrants to a meeting
- ✓ View a meeting's summary
- ✓ View a user's token
- ✓ Create invite links for a meeting

Types of data PantherSchools will access:



Content

Content generated in Zoom products, which may include audio, video, messages, transcriptions, feedback, responses to polls and Q&A, and files, and related context , such as invitation details, meeting or chat name, and meeting agenda.

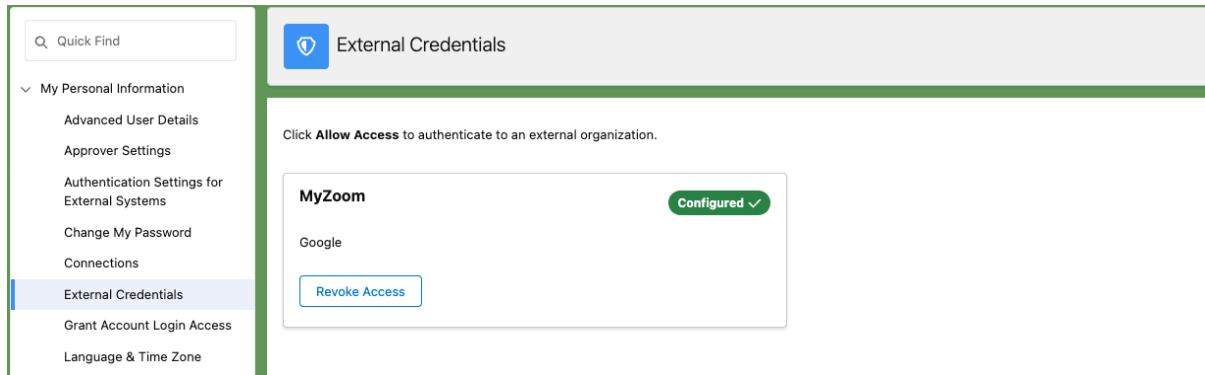
Allow this app to use my shared access permissions. [Learn more](#)

By clicking Allow, you give permission to this app to use your information in accordance . You can remove this app at any time in [My Apps](#).

[Allow](#)

[Decline](#)

After you have granted access then you will see the success message below



Use the Named credential in the Apex Class and test it.

```
String endPoint = 'callout:ZoomMeeting/users/me';

HttpRequest req = new HttpRequest();
req.setMethod('GET');
req.setHeader('Content-type', 'application/json');
req.setEndpoint(endPoint);

Http http = new Http();
HttpResponse res = http.send(req);
System.debug('response from Zoom User API' +res.getBody());

String endPoint =
'callout:Google/calendar/v3/calendars/cse.amitallenhouse@gmail.com
/events';

String params =
'?sendNotifications=true&sendUpdates=all&supportsAttachments=true&
alt=json&prettyPrint=true';

HttpRequest req = new HttpRequest();
req.setMethod('POST');
req.setHeader('Content-type', 'application/json');
req.setEndpoint(endPoint+params);

String requestBody = '{ '+
```

```
'      "end": {'+
'        "date": "2024-04-28"+
'}, '+
'      "start": {'+
'        "date": "2024-04-27"+
'}, '+
'      "attendees": ['+
'        {'+
'          "email": "engineeringkipathshala@gmail.com",'+
'          "displayName": "Engineering ki Pathshala"+
'}, '+
'        {'+
'          "email": "sfdcpanther@gmail.com",'+
'          "displayName": "Engineering ki Pathshala"+
'}']+
'], '+
'      "attachments": ['+
'        {'+
'          "fileUrl": "https://www.canva.com/design/DAF8IHx0BZ0/f-Mrh9VaKOTwa3WQI0h6Qw/edit",'+
'            "title": "Canva Files"+
'}']+
'], '+
'      "summary": "This is the Test Google Event",'+
'      "description": "This is the Test Google Event using Salesforce Named Credentials",'+
'      "organizer": {'+
'        "displayName": "Amit Singh",'+
'        "email": "cse.amitallenhouse@gmail.com"+
'}'+
'}';
```

```

req.setBody(requestBody);

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

System.debug('response from Google Calendar API' +res.getBody());

```

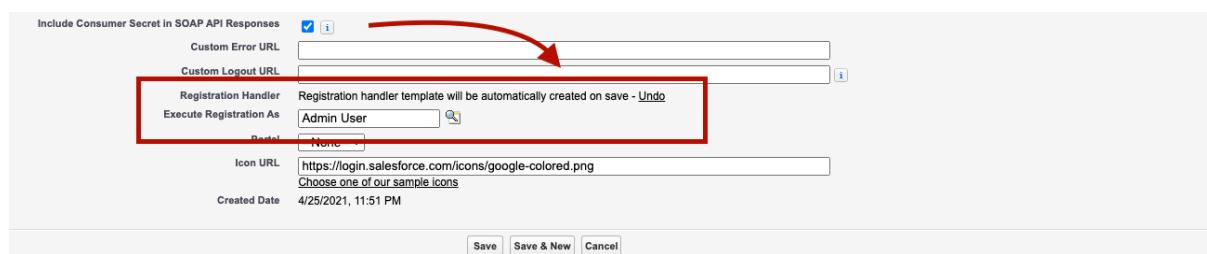
Use Auth. Provider to Login into Salesforce (Social Sign-On)

So far, we have seen that we can use Auth. Provider to get the access token and make the API calls to external applications like Google, QuickBooks, Dropbox, Sharepoint, LinkedIn, AWS &, etc.

Now, let's modify the same, Auth. Provider so login into Salesforce Org as Social Sign-On.

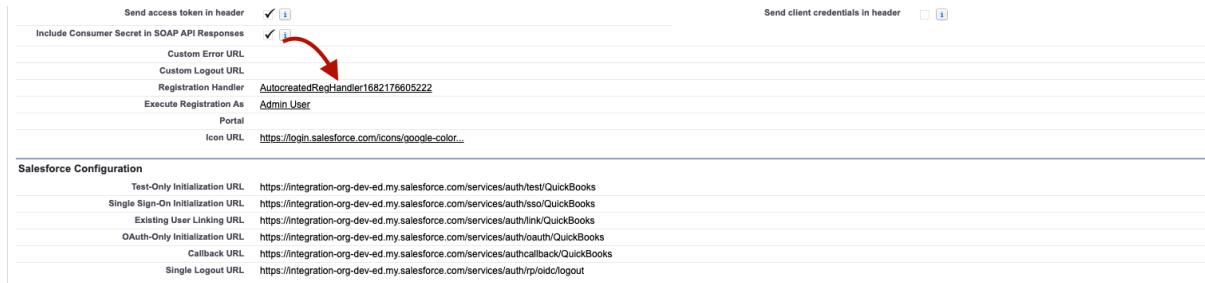
Modify the Auth. Provider

- Navigate to your **Auth. Provider** that you have created and Edit the same.
- Click on “**Automatically create a registration handler template**” to generate the Apex Class for the “**Registration Handler**” field
- Select your user for “**Execute Registration As**”
- Click Save to update the Auth. Provider



Modify the Apex Class

After you have saved the changes, you will see that the class has been automatically created for you. Click on that apex class to view the code. So we need to modify that apex class so that we can make it work as per our needs.



https://integration-org2-dev-ed.develop.my.salesforce.com/nc_external/entity/sso/ui/AuthorizationError?ErrorCode=REGISTRATION_HANDLER_ERROR&ErrorDescription=Attempt+to+de-reference+a+null+object&ProviderId=OSOHu0000013CJI&startURL=%2F#/

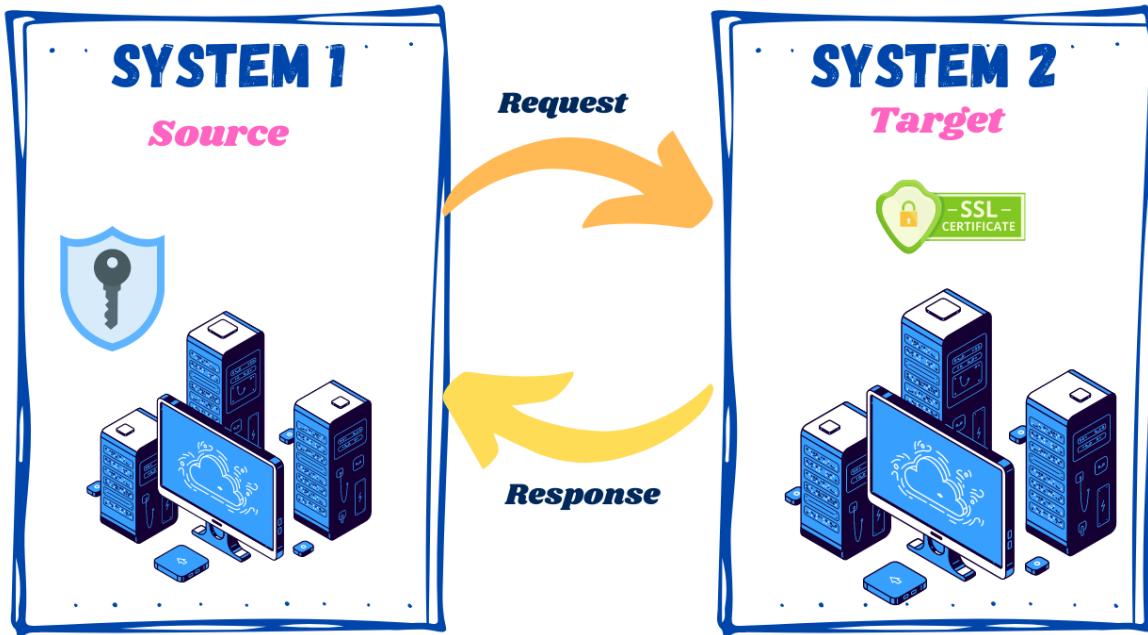
Assignment

- Create a GitHub Account if you do not have one already.
- Read the GitHub authentication API and note down the important information like auth URL, token URL, base URL, userinfo URL, scope
- Create an authentication provider in Salesforce and log in using that auth. Provider in salesforce. The logged user profile should be “**Chatter Free User**”
- Create a Named Credential in Salesforce and Use that Named Credential to Get all the repositories of your GitHub Account.

Salesforce JWT Authentication using Apex Class

In the context of the OAuth 2.0 JWT bearer token flow, the client initiates

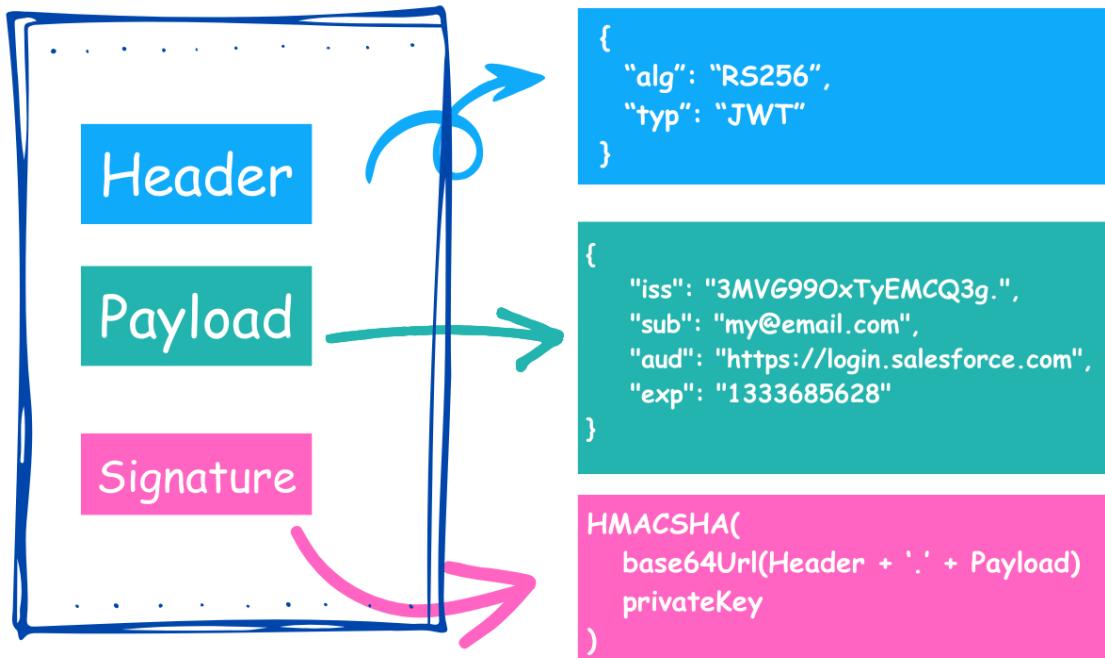
the process by sending a signed JWT to the Salesforce OAuth token endpoint. **Commonly used user authentication and authorization**



JWT Structure

1. **Headers** – These contain the algorithm that will be used to sign the request `{"alg": "RS256"}`
2. **Payload** – This contains claims information, an object containing information about the user and additional data. Claims are set using the parameters- `{"Iss, aud, sub, exp"}`
3. **Signature** – The signature consists of 3 parts and the structure is given below

```
<headerbase64encodedurl>.  
<claimsbase64encodedclaims>.  
<signature(uses an algorithm like RS 256)>
```



Salesforce receives the JWT and validates the digital signature to confirm the token's authenticity.

If the application is authorised, Salesforce then issues an access token to the client, allowing them to access protected resources on behalf of the user.

Prerequisites

- [Create a Private Key and Self-Signed Digital Certificate](#)
- [Create a Connected App in Your Org](#) - For JWT token purposes we will use a completely different Connected Application that will use a digital signature.

Steps involved for JWT

- Create a JWT
 - JWT header
 - JWT Claims
- Combine the JWT Header and JWT claims
- Get the private Key

- Sign & Base64UrlEncode the Private Key
- Get the access token
- Access Protected Resources

Create a JWT Header

- Create a JWT header - **{"alg":"RS256"}**
- Base64url encodes the JWT header and the result should be equal to
eyJhbGciOiJSUzI1NiJ9

Create the JWT Claims

to construct the JWT claims we need the following parameters and prepare the JSON

- **iss** - The issuer and it will contain the Client ID
- **aud** - The login URL of your salesforce org.
- **sub** - the Salesforce username
- **exp** - The current date time in UNIX format and add the minutes when you wanted to expire the token

Here is the example JSON file

```
{  
  "iss": "3MVG990xTyEMCQ3gNp2PjkqeZKxnmAiG1xV4oHh9AKL_rSK.",  
  "sub": "my@email.com",  
  "aud": "https://login.salesforce.com",  
  "exp": "1333685628"  
}
```

Now do the base64UrlEncode to the JWT claims data.

→ Combine the JWT header and Claims

Now, combine the JWT header & claims and store them in a variable. The concatenation should be with a dot (.) in between the header and claims like below

```
eyJhbGciOiJSUzI1NiJ9.eyJzdWlIiOiJzZmRjcGFudGhlcitoaW5kaUBnbWFpbC5jb2  
0iLCJpc3MiOilzTVZHOU5fSHZFVEdocjNDT0dTQloyUkd3MFVDdGhuZG1EY1drb  
GJkSGRuMDhMMElkbszz01wOFA3YzzpTnNUbmhNXzYxWnV2UDNyckFxT2Nw  
SGk1liwiZXhwIjoxNjgyMDgxOTA5NTIwLCJhdWQiOiJodHRwczovL2xvZ2luLnNh  
bGVzZm9yY2UuY29tIn0
```

→ Get the Private Key

we have already created the private key “**server.key**” as part of the prerequisite, upload the file in your salesforce org so that we can use it in Apex Class. To use in Postman we will open the key file in a notepad and can use the value.

→ Sign & Base64UrlEncode the Private Key

The private key must need to be signed using any of the supported algorithms however the recommendation is to use **rsa-sha256** for signing the key. You can use crypto methods of Salesforce Apex Class and some online tools like [jwt.io](#) for testing purposes.

→ Append the **Base64UrlEncoded** key with jwt header & claim and use dot(.) as a separator

Below is the complete JWT assertion that will look like. You will see the clear differentiation in the colours for

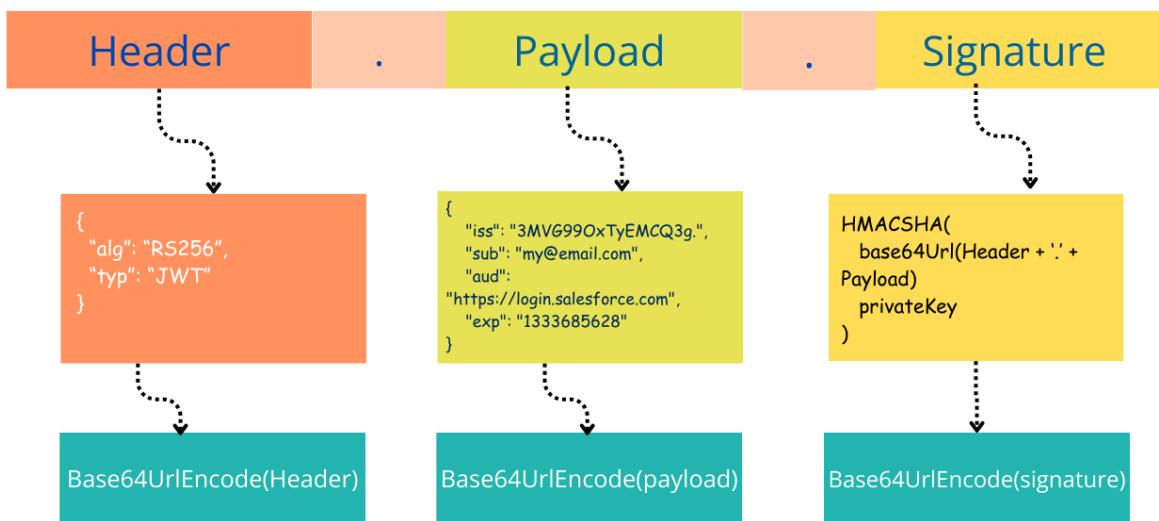
- header - **eyJhbGciOiJSUzI1NiI**sInR5cI6IkpxVCJ9
- claims -
eyJpc3MiOilzTVZHOU5fSHZFVEdocjNDT0dTQloyUkd3MFVDdGhuZG1EY1

drbGJkSGRuMDhMMElkbszZ01wOFA3YzZpTnNUbmhNXzYxWnV2UDNy
ckFxT2NwSGk1liwic3Viljoic2ZkY3BhbnRoZXIraGluZGlAZ21haWwuY29tliw
iYXVkljoiaHR0cHM6Ly9sb2dpbi5zYWxlC2ZvcmNlLmNvbSIsImV4cCI6MTY
4MjA4NTM3OTM1MH0

- signature -

ZWd7CnYg3—pZKjb44TrCtKc4UCTh1`GHb0kOTWOPmZsjl6Lya7_GXa7-g
1ULTb05ckLVYURuP7zURUpLkEjNqlLILSlYI6K20v6wgpFdziXbkAYznJRgCX
Lt6Vz_pwPeK6Z24I9xtuSefuR0LPDNkrKF80npTP6m_XFpYPfl5jf1yO7SEiv
rB8l1IJUstunEGjtQ5mIOMM_9oSiJbGUHFq9TlbdMLnDabZks7u-K3l5fxb-
dam28gz8w1N6K4rzVYSppZwbgNXaljz9sAMn-8P6dRYwTFy4TKlU62ePN
x18uB8Jq3gvjLSRqALcyk4evCRQpePn0IKVunVXBA36tg

JWT ASSERTION



eyJhbGciOiJSUzI1NiIsInR5cCI6IkpxVCJ9.eyJpc3MiOilzTVZHOU5fSHZFVEdocjN
DT0dTQloyUkd3MFVDdGhuZG1EY1drbGJkSGRuMDhMMElkbszZ01wOFA3YzZ
pTnNUbmhNXzYxWnV2UDNyckFxT2NwSGk1liwic3Viljoic2ZkY3BhbnRoZXIraGlu
ZGlAZ21haWwuY29tliw iYXVkljoiaHR0cHM6Ly9sb2dpbi5zYWxlC2ZvcmNlLmNvb
SIsImV4cCI6MTY4MjA4NTM3OTM1MH0.ZWd7CnYg3—pZKjb44TrCtKc4UCTh1`G
Hb0kOTWOPmZsjl6Lya7_g1ULTb05ckLVYURuP7zURUpLkEjNqlLILSlYI6K
20v6wgpFdziXbkAYznJRgCXL6Vz_pwPeK6Z24I9xtuSefuR0LPDNkrKF80npTP6
m_XFpYPfl5jf1yO7SEivrB8l1IJUstunEGjtQ5mIOMM_9oSiJbGUHFq9TlbdMLnDab

Zks7u-K3l5fxb-dam28gz8w1N6K4rzVYSppZwbgNXaljz9sAMn-8P6dRYwTFy4T
KLU62ePNx18uB8Jq3gvjLSRqALcyk4evCRQpePn0IKVunVXBA36tg

**JUNT** by Clicka

[Debugger](#) [Libraries](#) [Introduction](#) [Ask](#)

Crafted by  by Clicka

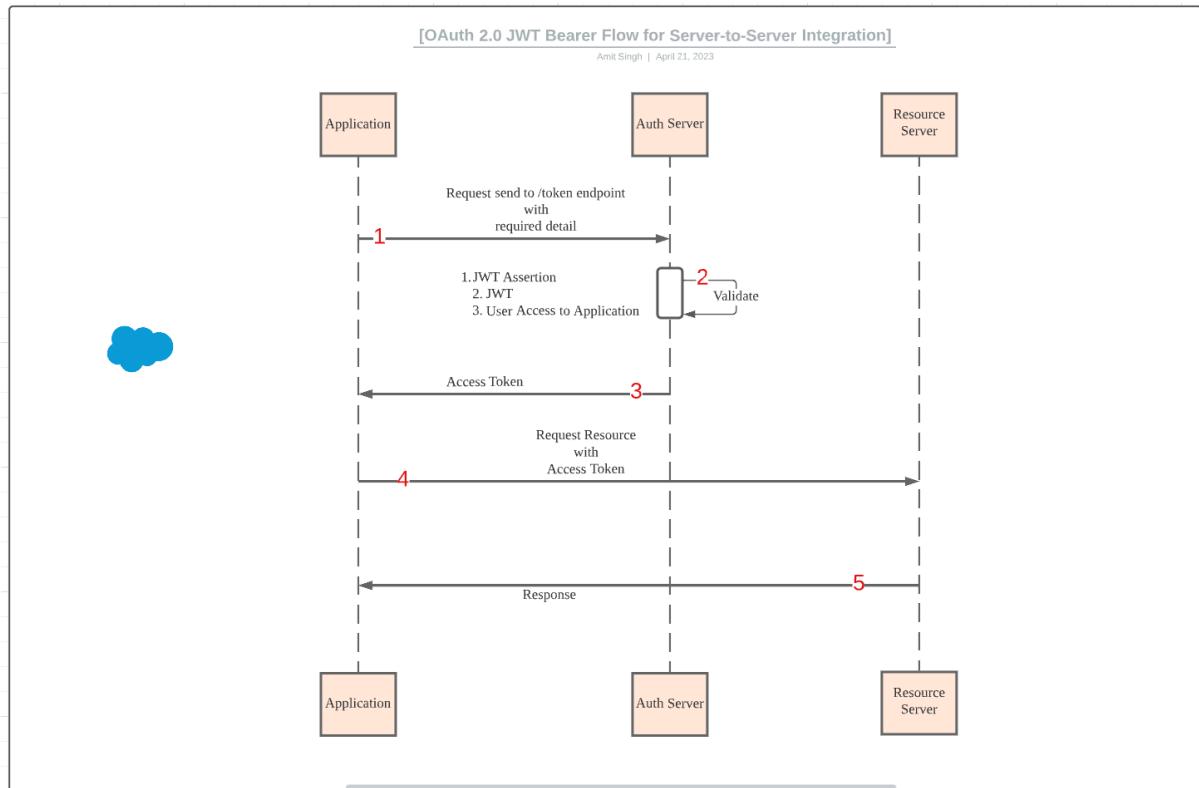
Encoded PASTE A TOKEN HERE

```
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiIzTVZH0W5fSHZFVEdocjNDT0dTQloyUkd3MFVDdGhuZG1EY1drbGJkSGRuMDhMME1kbWszZ01wOFA3YzzTnNUbmhNXzYxWhV2UDNyckFxT2NwSGk1Iiwig3ViIjoic2ZkY3BhbnRoZXIraGluZGLAZ21haWwuY29tIiwiYXVkIjoiaHR0cHM6Ly9sb2dpbi5zYWxlc2ZvcmNlLmNvbSIsImV4cCI6MTY4MjA4NTM3OTM1MH0.ZWd7CnYg3--pZKjb44TrCtKc4UCTh1`GHb0kOTWOPmZsjl6Lya_e7_Gxa7-g1ULTb05ckLVYURuP7zURUpLkEjNql1ILS1YI6K20v6wgpFdziXbkAYznJrgCXLt6Vz_pwPeK6Z24I9xtuSefuR0LPDNkrKF80npTP6m_XFpYPf15jf1y07SEivrB8l1IJUstunEGjtQ5mI0MM_9oSijbGUHFq9TlbdMLnDabZks7u-K3l5fxb-dam28gz8w1N6K4rzVYSppZwbgNXaljz9sAMn-8P6dRYwTFy4TK1U62ePNx18uB8Jq3gvjLSRqALcyk4evCRQpePn0IKVunVXBA36tg
```

Decoded EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE
{ "alg": "RS256", "typ": "JWT" }
PAYOUT: DATA
{ "iss": "3MVG9n_HvETGhr3COGSBZRGw0UCthndmDcWk1bdHdn08L0Idmk3gMp8P7c6inNsTnhM_61ZuvP3rrAqCpHi5", "sub": "sfdcpanther+hindi@gmail.com", "aud": "https://login.salesforce.com", "exp": 1682085379350 }
VERIFY SIGNATURE
RSASHA256(base64UrlEncode(header) + "." + base64UrlEncode(payload), Public Key in SPKI, PKCS #1, X.509 Certificate, or JWK stri- ng format. -----BEGIN RSA PRIVATE KEY----- --- MIIEpaIBAAKCAQEaoi7/+mBiouquiIML5w2+1cz8nq1xjZ8pf609yLBQHvSYnhG+)

panu



→ Get the access token

The Token endpoint will look like the below for the Salesforce Org.

- For Production or Developer Org -
<https://login.salesforce.com/services/oauth2/token>
- For Sandbox OR Scratch ORG -
<https://test.salesforce.com/services/oauth2/token>

The content type must be application/x-www-form-urlencoded

The body would be

```
grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion=COMPL
ETE_JWT_ASSERTION
```

Here is the sample request from Postman

The screenshot shows a Postman request for an OAuth 2.0 access token. The method is POST, and the URL is {{token_url}}. The Headers section shows 'Content-Type: application/x-www-form-urlencoded'. The Body section is set to 'x-www-form-urlencoded' and contains two parameters: 'grant_type' (with value 'urn:ietf:params:oauth:grant-type:jwt-bearer') and 'assertion' (with value 'eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiZTVZHOU5fSHZFVEdocJNDT...'). The response status is 200 OK, with a response body containing an access token and other metadata.

Get the Access Token using Custom Apex Class and a Private Key

Here is the complete code for the JWT using Apex

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 05-04-2024
```

```
* @last modified by : Amit Singh - PantherSchools
*/
public class JWT {

    private static final String JWT_HEADER = '{"alg":"RS256"}';

    public static String retrievePrivateKey(){
        ContentVersion base64Content = [SELECT VersionData FROM
ContentVersion WHERE Title = 'server.key' ORDER BY Title LIMIT 1];
        String keyContents = base64Content.VersionData.toString();
        keyContents = keyContents.replace('-----BEGIN PRIVATE
KEY-----', '');
        keyContents = keyContents.replace('-----END PRIVATE
KEY-----', '');
        keyContents = keyContents.replace('\n', '');
        return keyContents;
    }
    // JWT.run();
    public static void run(){

        DateTime rightNow = System.now(); // Current Date Time
        rightNow      = rightNow.addMinutes(3);
        Long expiry = rightNow.getTime(); // UnixTimeStamp

        String CLAIMS = '{'+
            '"iss":'
            "3MVG90Gq41FnYVsH1.2RQz_AEPeoPQYTczFa28uZUhyY_3", '+
            '"sub": "epic@epic.th", '+
            '"aud": "https://login.salesforce.com", '+
            '"exp": '+expiry+
        '}';
    }
}
```

```

        String base64UrlJWTHeader = SFDC_BASE64_URLENCODE(
Blob.valueOf(JWT_HEADER) );
        String base64UrlJWTClaims = SFDC_BASE64_URLENCODE(
Blob.valueOf(CLAIMS) );

        String combinedHeaderClaim =
base64UrlJWTHeader+'.'+base64UrlJWTClaims;
        System.debug('combinedHeaderClaim \n
'+combinedHeaderClaim);

        String jwtAssertion =
shaSignPrivateKey(combinedHeaderClaim);
        System.debug('jwtstr \n '+ jwtAssertion);

        getAccessToken(jwtAssertion);
    }

    public static String shaSignPrivateKey(String
combinedHeaderClaim){
        // base64Decode the Private key Content
        Blob privateKey = EncodingUtil.base64Decode(
retrievePrivateKey() );
        // Sign the Private Key using the rsa-sha256 Algorithm
        Blob signature = Crypto.sign('rsa-sha256',
Blob.valueOf(combinedHeaderClaim), privateKey);
        // Add the signature to the assertion
        combinedHeaderClaim += '.' +
SFDC_BASE64_URLENCODE(signature);
        return combinedHeaderClaim;
}

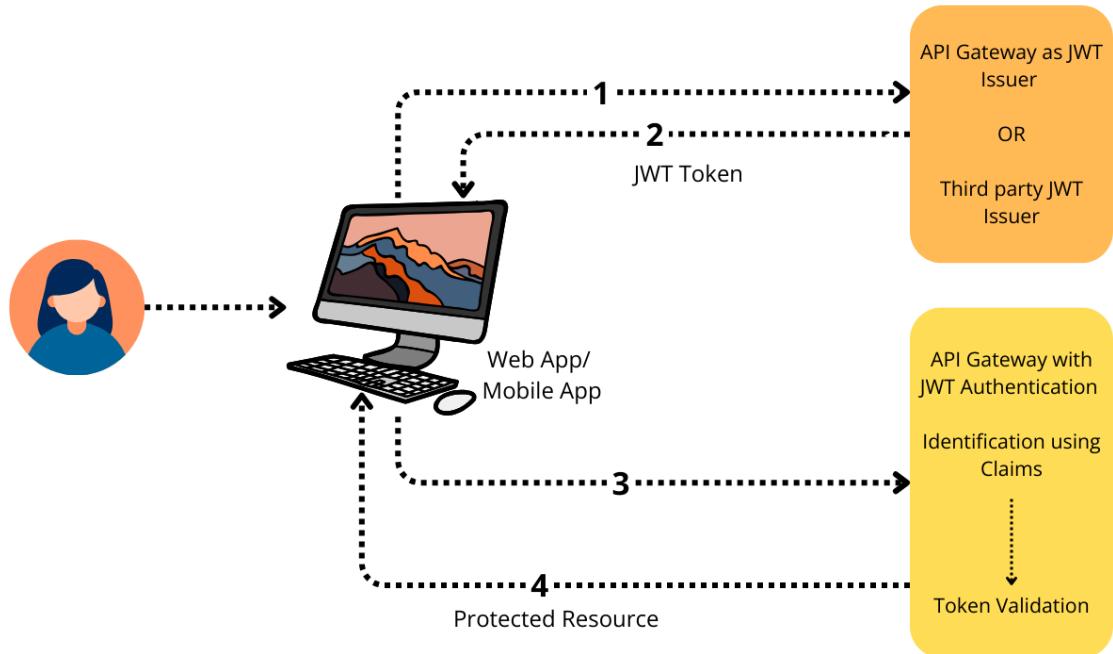
private static String SFDC_BASE64_URLENCODE(Blob input){

```

```
if(input == null) {
    return null;
}
return EncodingUtil.base64Encode(input)
    .replace('/', '_')
    .replace('+', '-')
    .replaceAll('=+$', '');
}

private static void getAccessToken(String assertionValue){
    HttpRequest req = new HttpRequest();

    req.setEndpoint('https://login.salesforce.com/services/oauth2/token');
    String requestBody =
'grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion='
'+assertionValue;
    req.setMethod('POST');
    req.setHeader('Content-Type',
'application/x-www-form-urlencoded');
    req.setBody(requestBody);
    Http http = new Http();
    HttpResponse res = http.send(req);
    System.debug(res.getBody());
    String response = res.getBody();
}
}
```



Convert .CRT to Salesforce Keystore JKS file

We already have a **server.crt** file by using our step1. Let's see how we can convert the same

1. Go to the same folder where you created the **server.key** file(By using step 1). Clone the **server.key** file and save it as **server.pem**.
2. Now execute this command : `openssl pkcs12 -export -in server.crt -inkey server.pem -out keystore.p12`
3. Now execute the **keytool command to create the jks file.** `keytool -importkeystore -srckeystore keystore.p12 -srcstoretype pkcs12`

```
-destkeystore servercert.jks -deststoretype JKS
```

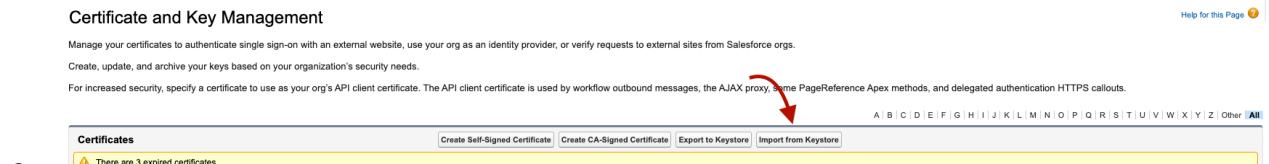
- a. It will ask you to create a password and remember it. We will use it soon.
4. Now Salesforce doesn't support default alias 1. So change the alias name with this command `keytool -keystore servercert.jks -changealias -alias 1 -destalias <name of your certificate>`

Upload the .jks file to Salesforce

Once you have generated the .jks file, let's upload the same to Salesforce under Certification & Key Management.

Follow the below steps for the same

- Login to Salesforce
- Navigate to **Setup → Security Control → Certification & Key Management → Select Import from the key store button**



- Choose your file, and make sure you are uploading the **.jks** file
- Provide the password
 - **Note:-** Use the same password you used at the time of generating the certificate



- Click Save

The Standard Salesforce Apex Class

- [Auth.JWT Class](#)
- [Auth.JWS Class](#)
- [Auth.JWTBearerTokenExchange](#)

```
/***
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-04-2024
 * @last modified by : Amit Singh - PantherSchools
 **/


public with sharing class PS_JWT {
    // PS_JWT.run();
    public static void run(){
        Auth.JWT jwt = new Auth.JWT();
        jwt.setSub('epic.510@epic.th');
        jwt.setAud('https://login.salesforce.com');
        jwt.setIss('3MVG90Gq41FnYVsH1.Fa28uZUhyY_3');
        System.debug(jwt.toJSONString());

        //Create the object that signs the JWT bearer token
        Auth.JWS jws = new Auth.JWS(jwt,'integrationconcert');

        String tokenEndpoint
        ='https://login.salesforce.com/services/oauth2/token';

        Auth.JWTBearerTokenExchange bearer = new
        Auth.JWTBearerTokenExchange(tokenEndpoint, jws);

        //Get the access token
        String accessToken = bearer.getAccessToken();
        system.debug('Access Token--> \n '+accessToken);
    }
}
```

```
    getUserInfo(accessToken);  
}  
  
public static void getUserInfo(String accessToken){  
    HttpRequest req = new HttpRequest();  
  
    req.setEndpoint('https://dhn00002vxjomao-dev-ed.develop.my.salesforce.com/services/oauth2/userinfo');  
    req.setMethod('GET');  
    req.setHeader('Authorization', 'Bearer '+accessToken);  
    req.setHeader('Content-Type', 'application/json');  
    Http http = new Http();  
    HttpResponse res = http.send(req);  
    System.debug(res.getBody());  
    String response = res.getBody();  
}  
}
```

You will have an outcome like **SESSION_ID_REMOVED** which means this method is working fine.

In the same way, you can do the authentication for any external application if their system supports the JWT authentication.

Salesforce JWT Authentication using Named Credentials

Previously, we have seen that we can use the Keystore and the Apex Class to get the access token. To enhance security and reduce the effort of getting the access token, we can use the named credentials for getting the access token.

Follow the below steps to create the named credentials

- Login to Salesforce org
- Navigate to Setup → Named Credentials → External Credentials → New External Credential
- For **Authentication Protocol** select **OAuth 2.0**
- For the **Authentication Flow Type** select “**JWT Bearer Flow**”
- For the **Identity Provider URL** provide the URL to the Access Token which is responsible for getting the access token
- Provide the details and click save
 - **Issuer (iss)** - The Client Id
 - **Subject (sub)** - The Salesforce User Name
 - **Audience (aud)** - The Salesforce Login Url
 - For Sandbox OR Scratch Org - <https://test.salesforce.com/>
 - For Production or Developer Org - <https://login.salesforce.com>
 - **JWT Expiration (Seconds)** - The no seconds after which tokens will be expired

Edit JWT

* Label * Name

* Authentication Protocol ▼

* Authentication Flow Type Scope

* Identity Provider URL

Common Claims ⓘ ▲

Issuer (iss)

Subject (sub)

Audience (aud)

JWT Expiration (Seconds) ⓘ

JWT Signing ▲

* Signing Certificate * Signing Algorithm ▼

Cancel Save



Once you have saved the named credentials, please use the below code for testing purposes if the Named Credentials are working or not.

Note: - Replace **MySalesforce** with the name of your Named Credentials

```
String loggedUserInfo = '/services/oauth2/userinfo';
```

```
HttpRequest req = new HttpRequest();
req.setEndpoint('callout:MySalesforce'+loggedInUserInfo);
req.setMethod('GET');
req.setHeader('Content-Type', 'application/json');
req.setHeader('Accept', 'application/json');

HTTPResponse res = (new Http()).send(req);
System.debug(res.getStatusCode());
System.debug(res.getBody());
```

Introduction to Streaming API

Streaming API enables the streaming of events using push technology and provides a subscription mechanism for receiving events in near real-time.

The streaming API uses a push technology mechanism that involves

- PushTopics Event (Legacy)
- Standard Events
- Platform Events
- Change Data Capture

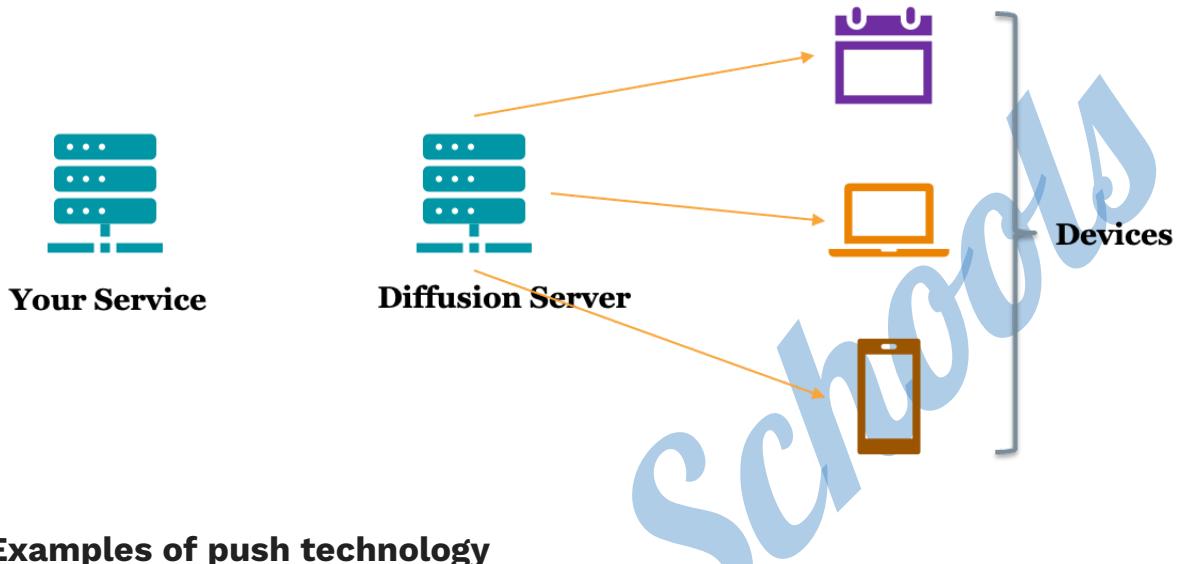
Introduction to Push Technology

Push technology, also called the publish/subscribe model, transfers information that is initiated **from a server to the client**.

This type of communication is the opposite of pull technology in which a request for information is made **from a client to the server**.

How does push technology work?

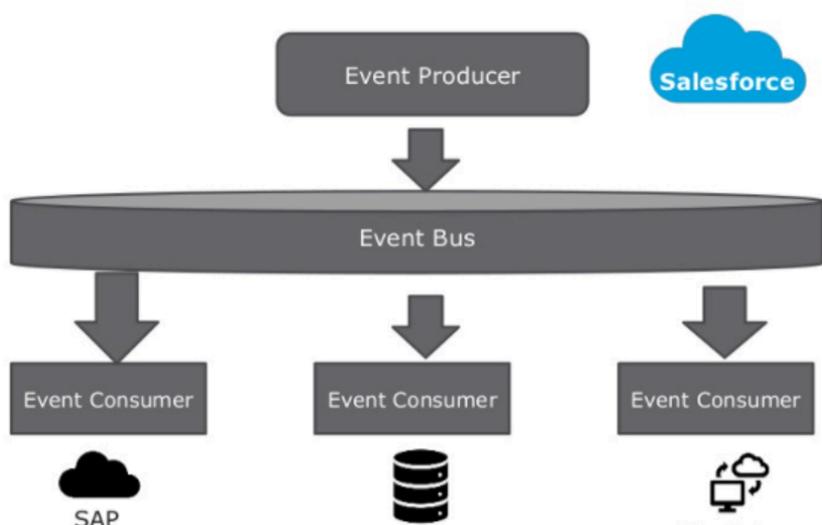
Push technology or server push, is a style of Internet-based communication where the request for a given transaction is initiated by the publisher or central server. Push Technology works on the publish-subscribe model.

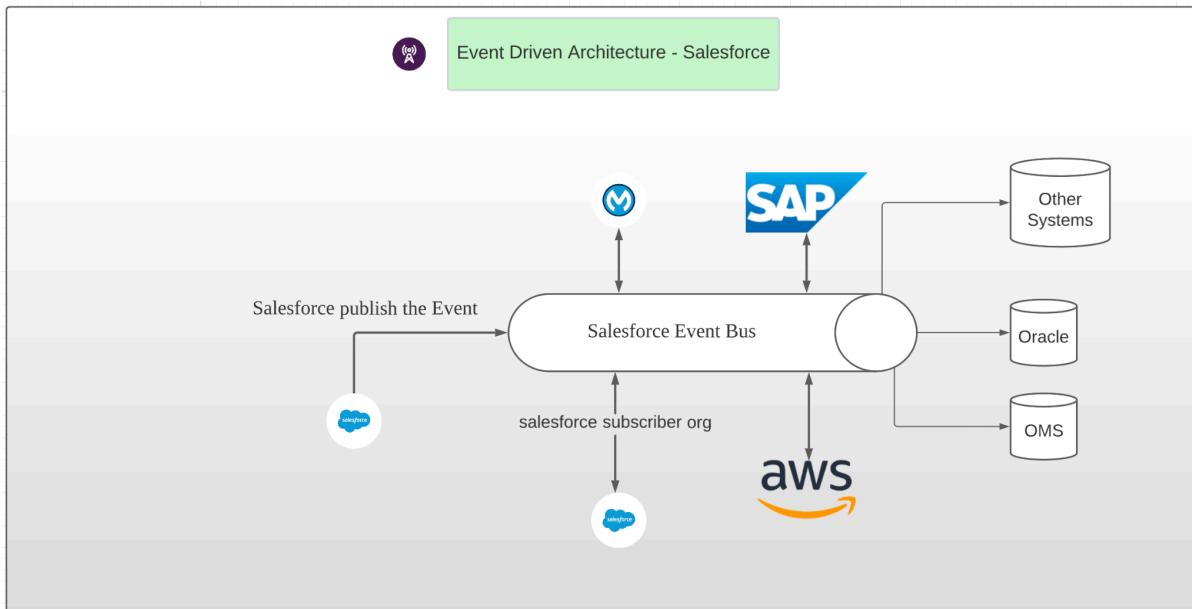


Examples of push technology

- Example 1 – App Notifications
- Example 2 – Email Notifications
- Example 3 - Messaging
- Example 4 – Integrations
- Example 5 – Entertainment

Publish-Subscribe Model





Important Terms

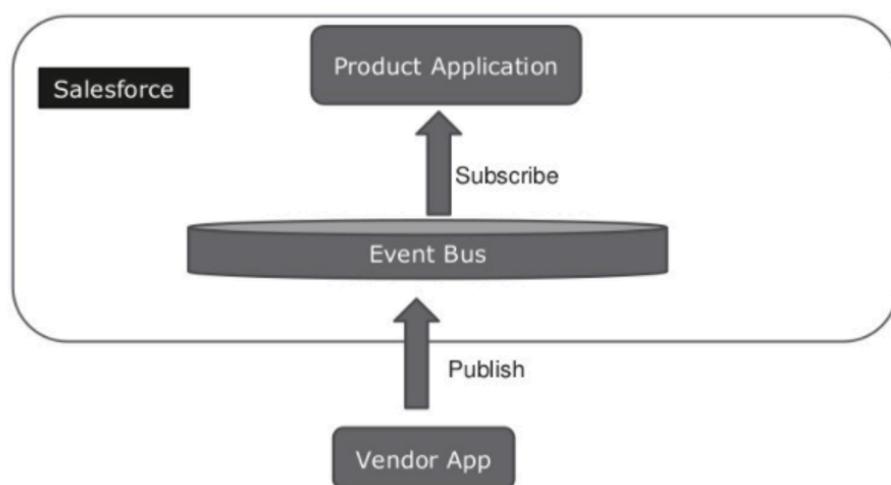
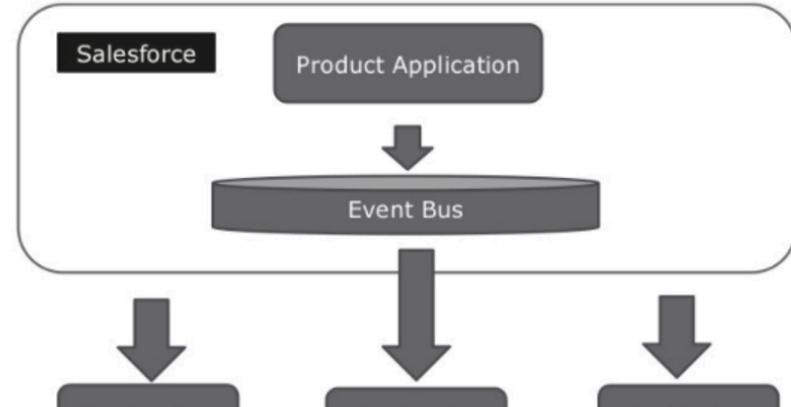
- **Event** – The creation, update, deletion, or undelete of a record. Each event might trigger a notification.
- **Notification** – A message in response to an event.
- **PushTopics** – A PushTopic triggers notifications for changes in Salesforce records resulting from a **create, update, delete, or undelete operation**.
- **Channel** – A stream of events to which a client can subscribe to receive event notifications
- **EventBus** – A conduit in which a publisher sends an event notification. Event subscribers subscribe to a channel in the event bus to receive event notifications.
- **Platform Event**
- **Change Data Capture**

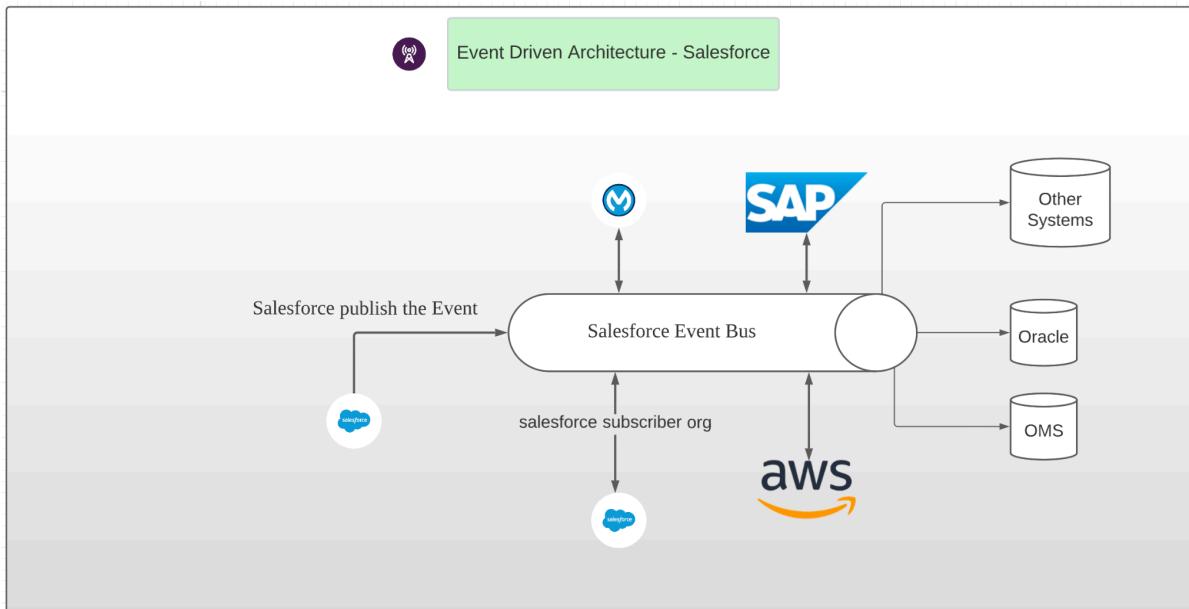
Introduction to Platform Events

- Use platform events to connect business processes in Salesforce and

external sources through the exchange of real-time event data.

- Platform events are secure and scalable.
- Define fields to customize your platform event data.
- Platform Events are first-class Members
- Platform Event works on Event-Driven Architecture





Types of Platform Events

- **Standard Platform Events** - Salesforce provides events with predefined fields, called standard platform events. An example of a standard platform event is
 - **AssetTokenEvent**, which monitors OAuth 2.0 authentication activity.
 - Another example is **BatchApexErrorEvent**, which reports errors encountered in batch Apex jobs.
 - **LogoutEventStream**
 - https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_objects_list.htm
- **High-Volume Platform Events** - Use high-volume platform events to publish and process millions of events efficiently and to scale your event-based apps.
 - After **45.0** API the custom events are **High Volume** by the default

What do we have in this course?

- Create Platform Event and Publish Using Apex Class

- Publish with Lightning Flow Builder
- Subscribe using Lighting Flow Builder
- Subscribe using Lightning Web Component
- Subscribe to Standard Platform Event using Lightning Web Component
- Subscribe to the **LogoutEventStream** event using the apex trigger and Component
- Subscribe to Platform Event using Node.js
- Assignment

Create Your 1st Platform Event & Publish it using Apex

- Platform Event always ends with **_e** instead of **_c**
- Platform Event can be migrated or retrieved under **Custom Object**
- To create a platform event you must need to have **Customize application** permission assigned to you.

To create the platform event follow the below steps

- Login to Salesforce
- Navigate to **Setup → Integrations → Platform Events → New Platform Event**
- Provide the details like the Label and other details, then click on Save

Platform Events

New Platform Event

Help for this Page

Platform Event Definition Edit

Save Save & New Cancel

Platform Event Information

Label

Plural Label

Starts with vowel sound

The object name is used when referencing the event via the API.

Object Name

Description

Event Type High Volume

Publish Behavior Publish After Commit

Deployment Status

In Development Deployed

Save Save & New Cancel

Once you have saved the platform event it will take you to the detail page of the Platform Event.

Now, create the below fields under the **Custom Fields & Relationships** section

- Account Id
- Account Name

Here is what your platform event will look like

The screenshot shows the Salesforce Platform Events setup page for the SAP Account Event. At the top, there's a header with the title "Platform Events" and a "SETUP" button. Below the header, it says "SAP Account Event". There are links for "Standard Fields [4]" and "Custom Fields & Relationships [2]".

Platform Event Definition Detail

Singular Label	SAP Account Event	Description	
Plural Label	SAP Account Events	Deployment Status	
Object Name	SAPAccountEvent	Deployed	
API Name	SAPAccountEvent_e		
Event Type	High Volume		
Publish Behavior	Publish Immediately		
Created By	Amit Singh, 4/15/2023, 6:48 AM	Modified By	Amit Singh, 4/15/2023, 6:53 AM

Standard Fields

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

Custom Fields & Relationships

Action	Field Label	API Name	Data Type	Indexed	Controlling Field	Modified By
Edit Del	Account Id	AccountId_c	Text(255)			Amit Singh, 4/15/2023, 6:49 AM
Edit Del	Account Name	AccountName_c	Text(255)			Amit Singh, 4/15/2023, 6:49 AM

Publishing behaviour of platform event

While creating the platform you noticed a field called publish behavior that has two values

- Publish Immediately
- Publish After Commit

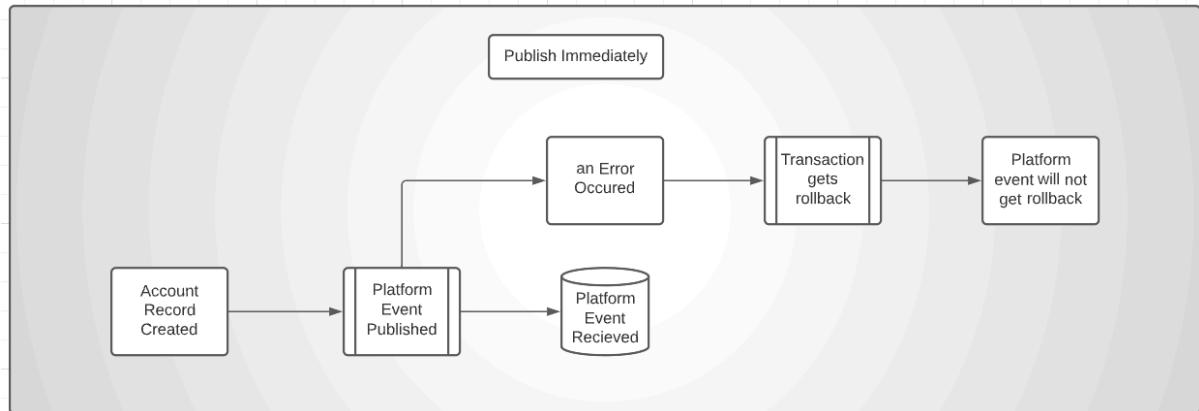
Publish Immediately - This is self-explanatory as it will publish the platform immediately as soon as you publish it using Apex, flow, or Process Builder.

We should be careful while using this behaviour for any platform Event.

Let's talk with an example, we have created the Platform Event for sending the SAP account and we are publishing the Platform Event from Apex Trigger.

After publishing the event there are some further processing happening for the same account record.

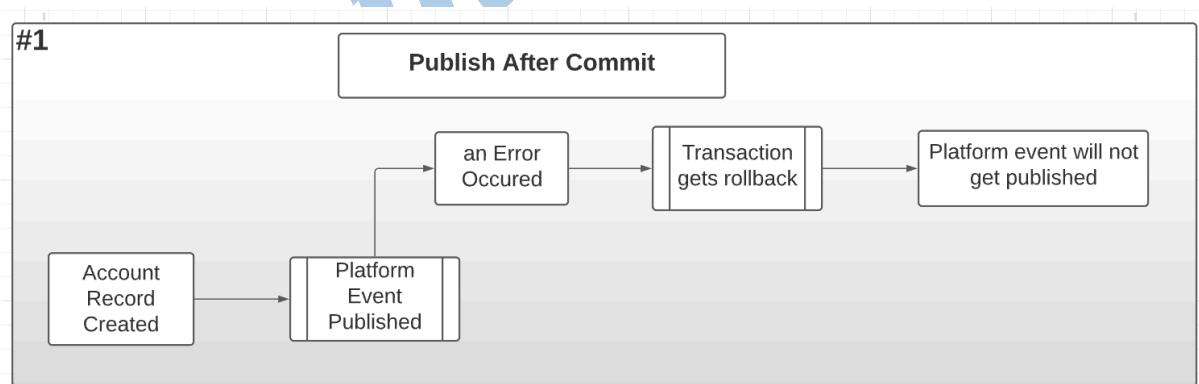
As soon as the event is published the subscribers will receive the data no matter if the transaction fails within Salesforce while processing the other part of logic.

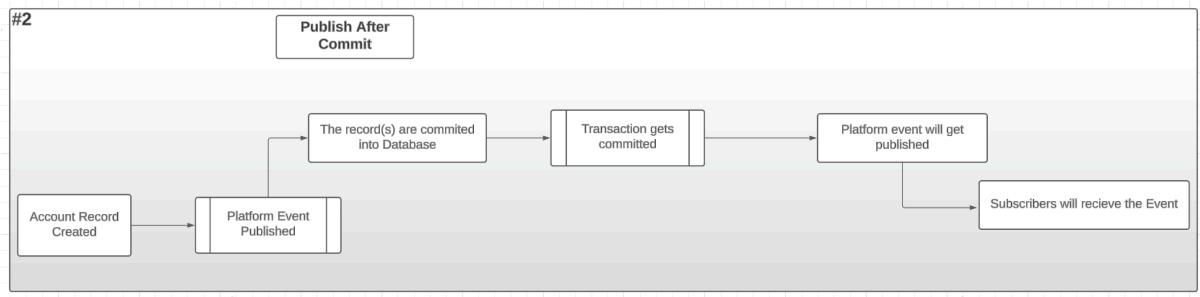


Publish After Commit

This type of Platform Event only gets published after all the processing has been completed and the record is committed into Salesforce.

Now, if we take the same example above, we have published the event and then later point in the code there is some error and the transaction gets rolled back in this case the Platform Event will not get published.





Publish the event using Apex Class

```

OrderStatus__e orderStatus = new OrderStatus__e();
orderStatus.Order_Number__c = 'ORD-24533';
orderStatus.Total_Amount_in_USD__c = 97734.3454;
EventBus.publish(orderStatus);

```

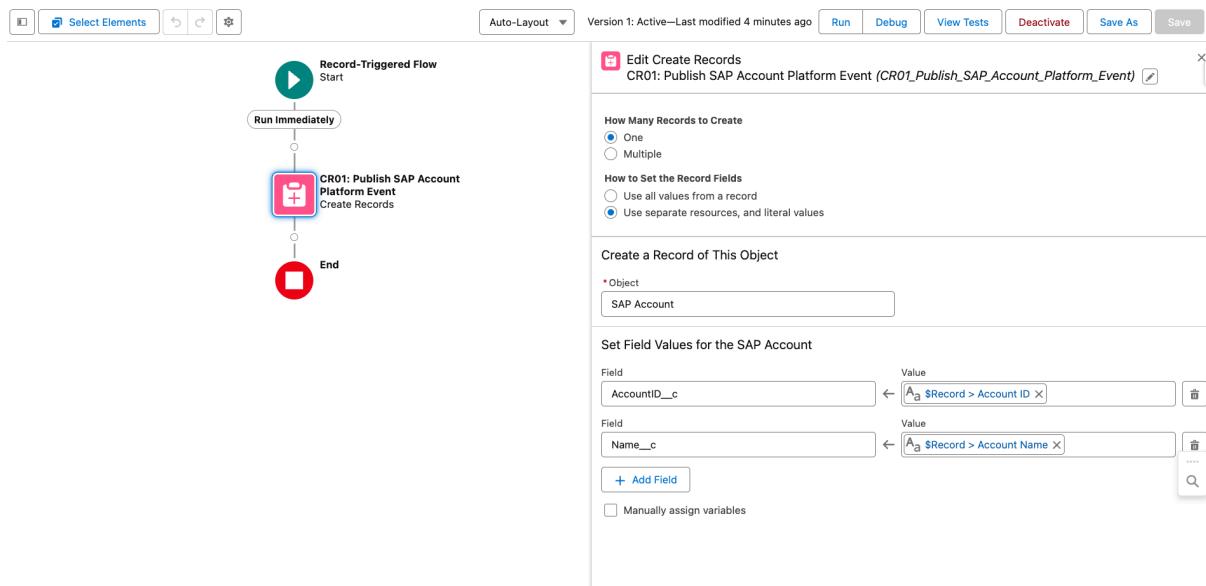
<https://workbench.developerforce.com/login.php>

<https://chromewebstore.google.com/detail/maven-tools-for-salesforce/kgookdijimmekebgdecakmblghjgiaoem>

Publish the Event Using Lightning Flow Builder

Use Case

- When an Account record is created then publish the platform using Process Builder.
- When a Platform Event is published then subscribe using Lightning Flow Builder.
- Update the account description with some static text and also do a chatter post.



Publish Event using Salesforce API

We can publish the platform event using Salesforce APIs as well. This is similar to creating the custom OR Standard Objects using the Salesforce Standard API.

Below are the steps that you need to follow

- [Login to workbench](#)
- Navigate to **Utilities → Rest Explorer**
- Create The record of the Platform Event using the Post Method

Endpoint -

<https://www.YOURDOMAIN.COM/services/data/v56.0/sobjects/<Your Platform>

Event API Name>

Method - POST

Content-Type: application/json

Body -

```
{
  "Name__c" : "United Oil & Gas, Singapore",
  "AccountId__c" : "0014x0000061xe3AAA"
}
```

```
}
```

Note:- Please remove all the fields that are marked as required while creating the fields and make sure you are using the correct API Name.

The screenshot shows the Salesforce REST Explorer interface. At the top, there are tabs for workbench, info, queries, data, migration, and utilities. A user is signed in as AMIT SINGH AT INTEGRATION ORG ON API 58.0. Below the tabs, it says "REST Explorer" and "Try the [Salesforce APIs for Postman](#)". A message says "Choose an HTTP method to perform on the REST API service URI below:". Below this are radio buttons for GET, POST, PUT, PATCH, DELETE, and HEAD, with POST selected. There are also "Headers" and "Reset" buttons. The URL input field contains "/services/data/v58.0/sobjects/SAP_Account__e". To the right of the URL is an "Execute" button. Below the URL input field, it says "Request Body" and shows a JSON object:

```
{  
    "AccountId_c": "AccountId924",  
    "Name_c": "PantherSchools Academy",  
    "AccountNumber_c" : "78834"  
}
```

[Expand All](#) | [Collapse All](#) | [Show Raw Response](#)

- ❖ id: e02xx0000000001AAA
- ❖ success: true
- 📁 errors

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 05-05-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
public with sharing class OrderTriggerHander {  
    public static void publishEvent(List<Order> newRecords){  
        List<Order> orderToPublish = [SELECT OrderNumber FROM Order  
WHERE Id IN:newRecords ];  
        List<OrderStatus__e> orderStatusList = new  
List<OrderStatus__e>();
```

```
    for(Order ord: orderToPublish ){
        OrderStatus__e orderStatus = new OrderStatus__e();
        orderStatus.Order_Number__c = ord.OrderNumber;
        orderStatus.Total_Amount_in_USD__c = 0.00;
        orderStatusList.add(orderStatus);
    }
    EventBus.publish(orderStatusList);
    // throw the Error
    Integer i = 10/0;
}
}
```

```
/***
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-05-2024
 * @last modified by : Amit Singh - PantherSchools
 */
trigger OrderTrigger on Order (after insert) {
    OrderTriggerHandler.publishEvent(Trigger.New);
}
```

```
/***
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-05-2024
 * @last modified by : Amit Singh - PantherSchools
 */
trigger PE_SAPAccountTrigger on SAP_Account__e (after insert) {
```

```
System.debug('SAP_Account__e Trigger Executed');
System.debug('SAP_Account__e Data \n '+
JSON.serializePretty(Trigger.new));
}
```

Use UUID to publish the Platform Event

It's very important to develop the solution in such a way that if the event fails or the business wants to track the separate events, the requirement can be achieved easily.

Use `sObjectType.newSObject(null, true)` concept to create the Object of the Platform Event that will generate the EventUuid `field`.

Example -

```
SAP_Account__e event =
(SAP_Account__e)SAP_Account__e.sObjectType.newSObject(null, true);
// The EventUuid value is returned after object creation
System.debug('EventUuid: ' + event.EventUuid);

event.AccountID__c = '001Hu00003C1wsVIAR';
event.AccountNumber__c = '78787234';
event.Name__c = 'Salesforce.com';

EventBus.publish(event);

SAP_Account__e event =
(SAP_Account__e)SAP_Account__e.sObjectType.newSObject(null, true);
```

```
// The EventUuid value is returned after object creation  
System.debug('EventUuid: ' + event.EventUuid);  
  
//event.AccountID__c = '001Hu00003C1wsVIAR';  
event.AccountNumber__c = '78787234';  
//event.Name__c = 'Salesforce.com';  
  
Database.SaveResult sr = EventBus.publish(event);  
System.debug(sr.isSuccess());  
System.debug(sr.getErrors());
```

Track the Success & Failure Platform Event

When we are working with event-driven architecture a developer/architect needs to keep in mind how the failures will be handled if there is any error while publishing the Platform Event using Salesforce Apex.

Salesforce provides multiple interfaces for handling the failure and success events of any Platform Event.

Here is the [interface list](#)

- EventPublishFailureCallback
- FailureResult
- EventPublishSuccessCallback
- SuccessResult

Reference Link -

https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/platform_events_publish_callback_publish.htm

You can implement the **EventPublishFailureCallback** & **EventPublishSuccessCallback** in the same apex class and implement the **onFailure** & **onSuccess** method of the interfaces respectively.

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on: 05-09-2024
 * @last modified by: Amit Singh - PantherSchools
 */

public with sharing class PE_SuccessAndErrorCallback implements
EventBus.EventPublishFailureCallback,
EventBus.EventPublishSuccessCallback {

    public void onFailure(EventBus.FailureResult result) {
        List<String> eventUuids = result.getEventUuids();
        System.debug(eventUuids.size() + ' events failed to
publish.');
        System.debug('Callback eventUuids to match with event
objects: ' + eventUuids);
    }

    public void onSuccess(EventBus.SuccessResult result) {
        List<String> eventUuids = result.getEventUuids();
        System.debug(eventUuids.size() + ' events were published
successfully.');
        System.debug('Callback eventUuids to match with event
objects: ' + eventUuids);
    }
}

```

You can publish the event with the callback example below

```

SAP_Account__e event =
(SAP_Account__e)SAP_Account__e.sObjectType.newSObject(null, true);
// The EventUuid value is returned after object creation
System.debug('EventUuid: ' + event.EventUuid);

event.AccountID__c = '001Hu00003C1wsVIAR';
event.AccountNumber__c = '78787234';
event.Name__c = 'Salesforce.com';

EventBus.publish(event, new PE_SuccessAndErrorCallback());

```

Apex Code used for the demo

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-12-2024
 * @last modified by : Amit Singh - PantherSchools
 **/ 

public with sharing class PE_SuccessHandler implements
EventBus.EventPublishSuccessCallback {
    public void onSuccess(EventBus.SuccessResult result){
        System.debug('Success');
        System.debug(result.getEventUuids());
    }
}

```

```

/***

```

```

* @description      :
* @author          : Amit Singh - PantherSchools
* @group           :
* @last modified on : 05-12-2024
* @last modified by : Amit Singh - PantherSchools
*/
public with sharing class PE_Handlers implements
EventBus.EventPublishSuccessCallback, EventBus.EventPublishFailureCallback {
    public void onSuccess(EventBus.SuccessResult result){
        System.debug('Success');
        System.debug(result.getEventUuids());
    }

    public void onFailure(EventBus.FailureResult result){
        System.debug('Failure');
        System.debug(result.getEventUuids());
    }
}

```

Subscribe to Standard Platform Event using Web Component (BatchApexErrorEvent)

Step1 – Import the required methods from the **lightning/empApi**

```

import { subscribe, unsubscribe, onError, setDebugFlag,
isEmpEnabled }
from 'lightning/empApi';

```

Subscribe to the Platform Event

```
handleSubscribe() {
```

```
const messageCallback = function (response) {
    console.log('New message received: ',
JSON.stringify(response));
};

subscribe(this.channelName, -1,
messageCallback).then((response) => {
    console.log(
        'Subscription request sent to: ',
        JSON.stringify(response.channel)
    );
    this.subscription = response;
});

}
```

Unsubscribe the Platform Event

```
handleUnsubscribe() {
    unsubscribe(this.subscription, (response) => {
        console.log('unsubscribe() response: ',
JSON.stringify(response));
    });
}
```

Handle the Error Message if there are any while subscribing to the event.

```
registerErrorListener() {
    onError((error) => {
        console.log('Received error from server: ',
JSON.stringify(error));
    });
}
```

Lightning Web Component Code

```
<template>
  <lightning-card title="Platform Event"
icon-name="standard:account">
  <div class="slds-m-around_large">
    <span>Subscribing the Batch Apex Error Event</span>
    <span>
      {eventMessage}
    </span>
  </div>
</lightning-card>
</template>
```

```
import { LightningElement } from 'lwc';
import { subscribe, unsubscribe, onError } from
'lightning/empApi';
export default class Ps_batchapexerrorevent extends
LightningElement {

  subscription;
  eventName = '/event/SAP_Account__e';
  eventMessage;

  connectedCallback(){
    this.registerErrorHandler();
    this.handleSubscribe();
  }

  handleSubscribe(){
    this.subscription = subscribe(this.eventName, -2,
```

```
this.handleSuccessErrorMessage.bind(this));
    console.log(this.subscription);
}

handleSuccessErrorMessage(message){
    console.log('Message Received!');
    this.eventMessage = JSON.stringify(message);
}

disconnectedCallback(){
    this.handleUnSubscribe();
}

handleUnSubscribe(){
    unsubscribe(this.subscription, (response) => {
        console.log(response);
    });
}

reisterErrorHandler(){
    onError((error) => {
        console.error(error);
    });
}
}
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata">
<apiVersion>59.0</apiVersion>
```

```
<isExposed>true</isExposed>
<masterLabel>Batch Apex Error Event</masterLabel>
<targets>
    <target>lightning__HomePage</target>
    <target>lightning__AppPage</target>
    <target>lightning__RecordPage</target>
</targets>
</LightningComponentBundle>
```

Assignment

- Create a Platform Event and Name it “**Opportunity Notification**”
 - This platform Event will have the following fields
 - Name
 - Stage
 - Close Date
 - Amount
 - Create a Flow Builder that will Publish the Event with the above details if the Opportunity is Closed Won.
 - Create an LWC Component that will subscribe to the Event and display the records in a table form.
 - Create a js file using Node.js and subscribe to the same event published using Flow Builder
- Create an Apex trigger on the - **LogoutEventStream** Event and store the details on the custom object.
 - The Fields can be found on the link -
https://developer.salesforce.com/docs/atlas.en-us.platform_events.meta/platform_events/sforce_api_objects_logouteventstream.htm

Subscribe to Platform Event using Node.js

To subscribe to the platform events, change data capture or Streaming API using node.js, we will use external libraries like jsforce. jsforce is a very popular and powerful library for setting up the integration with Salesforce using any platforms or external library like Node.js, VF Page, Browser, canvas or command line application.

<https://jsforce.github.io/>

<https://expressjs.com/en/starter/hello-world.html>

```
var jsforce = require('jsforce');
const express = require('express')
const app = express()
const port = 3000

var oauth2 = new jsforce.OAuth2({
  loginUrl : 'https://login.salesforce.com',
  clientId : '3MVG9VTfpJmxg1yjuseQiOTHifi8aUb0di',
  clientSecret : '477208E8D1A',
  redirectUri : 'http://localhost:3000/oauth2/callback'
});

app.get('/', (req, res, next) => {
  res.send('Hello World!')
});

app.get('/oauth2/auth', (req, res, next) => {
  res.redirect(oauth2.getAuthorizationUrl());
});

app.get('/oauth2/callback', function(req, res) {
  // http://localhost:3000/oauth2/callback
```

```
console.log(req.query);
var code = req.query.code;
console.log(code);

var conn = new jsforce.Connection({ oauth2 : oauth2 });

conn.authorize(code, function(err, userInfo) {
    if (err) {
        return console.error(err);
    }
    // Now you can get the access token, refresh token, and
instance URL information.

    // Save them to establish a connection next time.
    console.log(conn.accessToken);
    console.log(conn.refreshToken);
    console.log(conn.instanceUrl);
    console.log("User ID: " + userInfo.id);
    console.log("Org ID: " + userInfo.organizationId);
    const eventName = '/event/SAP_Account__e';
    const subscription =
conn.streaming.topic(eventName).subscribe(function(payload) {
        console.log('Received message: \n ',
JSON.stringify(payload, null, 2));
        // Handle the received platform event message
    });

    res.send('success'); // or your desired response
});
});

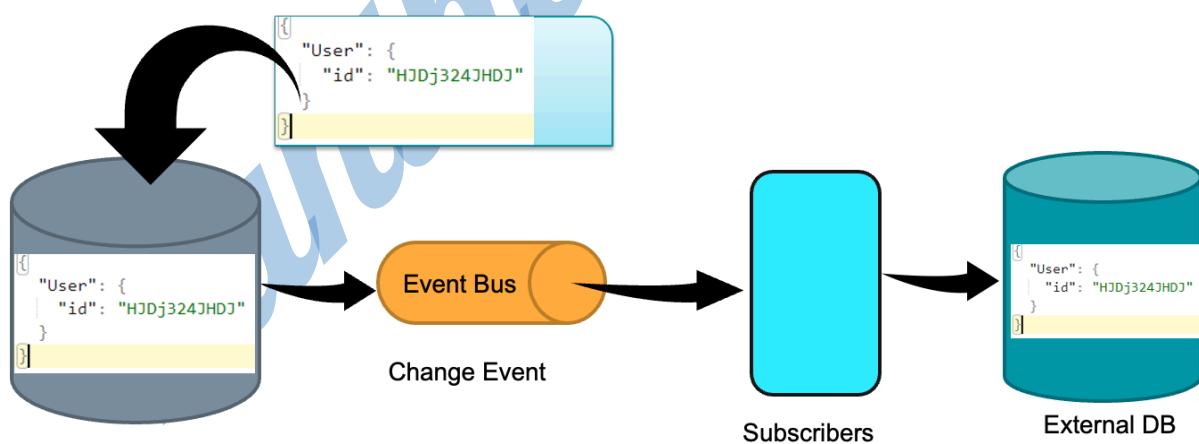
app.listen(port, ()=>{
    console.log(`Example app listening at
```

```
http://localhost:${port}`)  
})
```

Introduction to Change Data Capture

Change Data Capture is a streaming product on the Lightning Platform that enables you to integrate your Salesforce data with external systems efficiently.

- With Change Data Capture, you can receive changes in Salesforce records in near-real time and synchronise corresponding records in an external data store.
- CDC uses event-driven architecture
- CDC only used for record-level changes
- CDC respects the field-level security of your Salesforce Object



Enable Change Data Capture

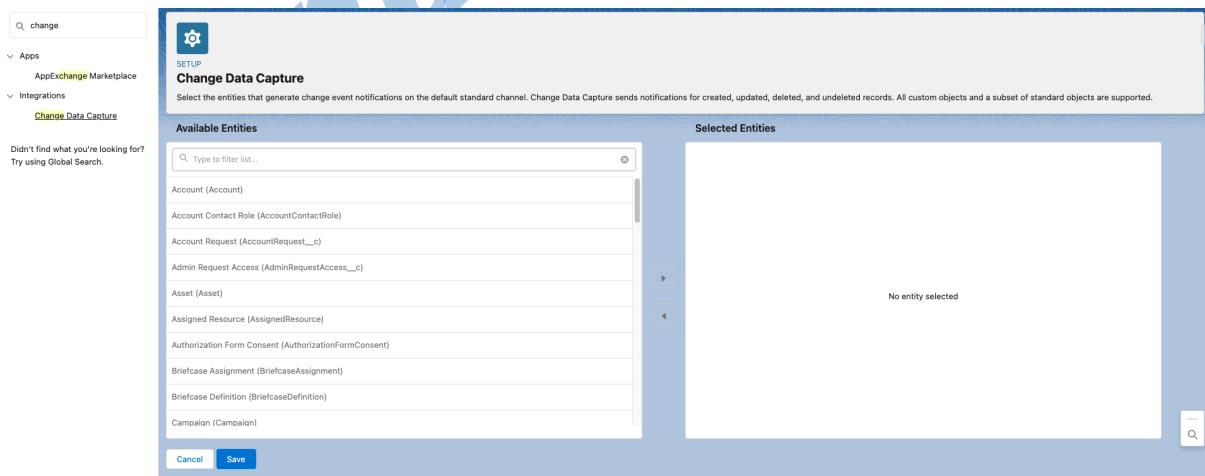
To use the CDC, we need to enable the setting from Salesforce and it has limitations to using some of the standard objects only however it does all

the important objects like

- Account
- Contact
- Lead
- Case
- Opportunity
- Order
- Campaign
- Event
- Product
- File
- Task
- User
- etc

Follow the below steps to enable the change data capture

- Login to Salesforce Org
- Navigate to Setup → Integrations → Change Data Capture



Prepare Subscription Channel

Like platform events, we also need to have a subscription channel that we

can subscribe to using Lighting Web Component, Workbench, Google Chrome Extensions like Mavens Tool for Salesforce, and any external application Node.js

Here is how we prepare the URL for the subscription

- For **Standard Object** /data/<ObjectAPIName>ChangeEvent
 - AccountChangeEvent
 - CaseChangeEvent
 - LeadChangeEvent
 - ContactChangeEvent
 - For All **Custom Object** /data/<ObjectName>_ChangeEvent
 - Course__c → Course__ChangeEvent
 - LinkedIn_Post_Comment__c → LinkedIn_Post_Comment__ChangeEvent
 - **Note:-** We need to exclude __c from the Custom object API Name.
 - For example, if the **object API Name is Invoice_Line__c** then the channel **will be Invoice_Line__ChangeEvent**

Event Channel

/data/ContactChangeEvent

Structure of Change Data Capture

```
{  
  "schema": "uniqueSchemaIdentifier",  
  "payload": {  
    "ChangeEventHeader": {  
      "entityName": "Account",  
      "recordIds": [  
        "001xx000003DHP0AA0"  
      ],  
      "changeType": "CREATE",  
      "changeOrigin": "com/salesforce/api/rest",  
      "changeTime": "2023-09-12T14:30:00Z",  
      "changeId": "001xx000003DHP0AA0",  
      "changeVersion": 1  
    }  
  }  
}
```

```
"transactionKey": "0000000000000000",
"commitTimestamp": 1622552400000,
"commitNumber": 1234567890123456800,
"sequenceNumber": 1,
"isTransactionEnd": true,
"changedFields": [
    "Name",
    "BillingCity"
],
"nulledFields": []
},
{
    "Name": "Acme Corporation",
    "BillingCity": "San Francisco",
    "LastModifiedById": "005xx000001Sv6VAA0",
    "LastModifiedDate": "2021-06-01T12:00:00Z"
},
{
    "event": {
        "replayId": 123456
    }
}
```

Apex Trigger Code

```
/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-18-2024
 * @last modified by : Amit Singh - PantherSchools
 */
```

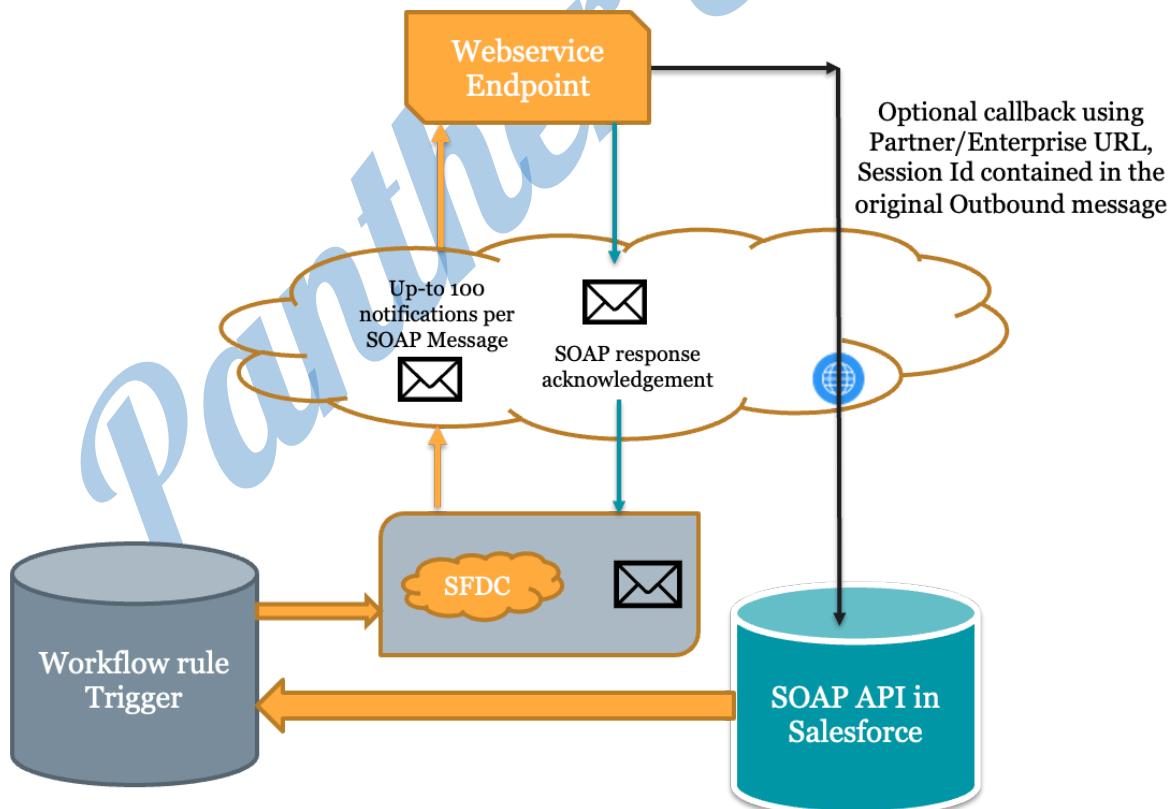
```
trigger PS_ContactChangeTrigger on ContactChangeEvent (after insert) {  
    for(ContactChangeEvent con: Trigger.new){  
        // check the change type  
        System.debug(System.JSON.serializePretty(con));  
        EventBus.ChangeEventHeader header = con.ChangeEventHeader;  
        String changeEntity = header.entityName;  
        String changeOperation = header.changeType;  
        if(changeOperation == 'CREATE'){  
            System.debug('Contact Created: ' + con.FirstName);  
        } else if(changeOperation == 'UPDATE'){  
            System.debug('Contact Updated: ' + con.FirstName);  
        } else if(changeOperation == 'DELETE'){  
            System.debug('Contact Deleted: ' + con.FirstName);  
        } else if(changeOperation == 'UNDELETE'){  
            System.debug('Contact Restored: ' + con.FirstName);  
        }  
    }  
}
```

Assignment

- Create a Custom Object called Employ with the following attribute
 - Name (standard field)
 - Email
 - Phone
- Enable the Change Data Capture
- Create a Lightning Web component and subscribe to the change data capture for the custom object.

Outbound Message

- The outbound message allows you to send the data to an external service without securely writing any web service.
- Outbound messaging uses the **notifications()** call to send SOAP messages over HTTP(S) to a designated endpoint when triggered by a workflow rule.
- Outbound Message uses SOAP messages i.e. XML format
- Outbound Message expects a valid acknowledgement from the 3rd party System
- Outbound Message has a retry mechanism if the message is not acknowledged or fails to send up to 24 Hours
- Outbound messages can only send record-level information to an external system.



Example

- There are 2 systems Salesforce and a Database to store the Leads. Whenever a lead comes into Salesforce it must go to an external Database.
- If there is any duplicate lead going to External Database then the Database will tell Salesforce that this is a duplicate lead and please mark this as a duplicate lead.

Structure of Notification Acknowledgement

Salesforce only considers an outbound message success only if Salesforce gets the expected response.

```
<soapenv:Envelope  
    xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"  
    xmlns:out="http://soap.sforce.com/2005/09/outbound">  
    <soapenv:Header/>  
    <soapenv:Body>  
        <out:notificationsResponse>  
            <out:Ack>true</out:Ack>  
        </out:notificationsResponse>  
    </soapenv:Body>  
</soapenv:Envelope>
```

Outbound in Action

Use Case – Send the Opportunity Details to a particular external system when an Opportunity is closed using Outbound Message.

- **Step 1 –** Create a Listener for outbound messages using <https://pipedream.com/>
- **Step 2 –** Create an outbound message for Opportunity Object
- **Step 3 –** Create a Workflow rule to execute that outbound message

New Outbound Message

Step 2 : Configure Outbound Message Step 2 of 2

Enter the details of your outbound message and select the fields you want included in this message. Note that the fields available depend on the type of record previously selected.

Edit Outbound Message: Account

Name:

Unique Name:

Description:

Endpoint URL:

User to send as: Admin User

Protected Component:

Send Session ID:

Account fields to send:

Available Fields	Selected Fields
AccountNumber AccountSource Active__c AnnualRevenue BillingCity BillingCountry BillingGeocodeAccuracy BillingLatitude BillingLongitude BillingPostalCode BillingState BillingStreet CleanStatus CreatedById	Id
Add Remove	

Help for this Page

Previous Save Cancel

I = Required Information

Apex Code to generate the Random Contact Records

```
public class GenerateContacts {

    public static void createContacts(Integer numberOfWorkers) {
        List<Worker> workersToCreate = new List<Worker>();
        String[] firstNames = new String[]{"John", "Jane", "Michael", "Sarah", "David", "Laura", "Robert", "Emily", "James", "Emma"};
        String[] lastNames = new String[]{"Smith", "Johnson", "Williams", "Jones", "Brown", "Davis", "Miller", "Wilson", "Moore", "Taylor"};
        String[] titles = new String[]{"Sales Manager", "Marketing Director", "Software Engineer", "Project Manager", "Consultant", "Product Manager", "Business Analyst", "HR Manager", "Accountant", "Support Specialist"};
        String[] departments = new String[]{"Sales", "Marketing", "Engineering", "HR", "Finance", "Customer Support", "IT", "Operations", "Legal", "Administration"};
    }
}
```

```
    for (Integer i = 0; i < numberOfContacts; i++) {
        String firstName =
firstNames[Math.mod(Math.abs(Crypto.getRandomInteger()),
firstNames.size())];
        String lastName =
lastNames[Math.mod(Math.abs(Crypto.getRandomInteger()),
lastNames.size())];
        String title =
titles[Math.mod(Math.abs(Crypto.getRandomInteger()),
titles.size())];
        String department =
departments[Math.mod(Math.abs(Crypto.getRandomInteger()),
departments.size())];
        String email = firstName.toLowerCase() + '.' +
lastName.toLowerCase() + '@example.com';
        String phone = generatePhoneNumber();
        String fax = generatePhoneNumber();

        Contact contact = new Contact(
            FirstName = firstName,
            LastName = lastName,
            Email = email,
            Phone = phone,
            Title = title,
            Department = department,
            Fax = fax
        );
        contactsToCreate.add(contact);
    }

    if (!contactsToCreate.isEmpty()) {
```

```

        insert contactsToCreate;
        System.debug('Successfully created ' + numberOfContacts
+ ' contacts.');
    }
}

private static String generatePhoneNumber() {
    String phoneNumber = '(';
    phoneNumber += String.valueOf(Math.mod(Math.abs(Crypto.getRandomInteger()), 900)
+ 100) + ')';
    phoneNumber += String.valueOf(Math.mod(Math.abs(Crypto.getRandomInteger()), 900)
+ 100) + '-';
    phoneNumber += String.valueOf(Math.mod(Math.abs(Crypto.getRandomInteger()),
10000) + 1000);
    return phoneNumber;
}
}

```

Apex Code to generate Random Opportunities

```

public class GenerateOpportunities {

    public static void createOpportunities(Integer
numberOfOpportunities) {
        List<Opportunity> opportunitiesToCreate = new
List<Opportunity>();
        String[] names = new String[]{ 'Acme Corp', 'Global

```

```
Industries', 'Tech Solutions', 'Innovative Systems', 'Dynamic  
Enterprises', 'Peak Performance', 'Superior Services', 'Bright  
Future', 'NextGen Technologies', 'Prime Ventures'};  
  
        String description = 'This is a sample opportunity  
description.';  
  
        Decimal[] amounts = new Decimal[]{5000, 10000, 15000,  
20000, 25000, 30000, 35000, 40000, 45000, 50000};  
  
        Date closeDate = Date.today().addMonths(1);  
  
  
        for (Integer i = 0; i < numberOfOpportunities; i++) {  
            String name =  
names[Math.mod(Math.abs(Crypto.getRandomInteger()), names.size())]  
+ ' - ' + (i + 1);  
  
            Decimal amount =  
amounts[Math.mod(Math.abs(Crypto.getRandomInteger()),  
amounts.size())];  
  
            String orderNumber = generateRandomNumber(8);  
            String trackingNumber = generateRandomNumber(12);  
  
  
            Opportunity opportunity = new Opportunity(  
                Name = name,  
                CloseDate = closeDate,  
                StageName = 'Prospecting',  
                Description = description,  
                Amount = amount,  
                Type = 'New Customer',  
                LeadSource = 'Web',  
                OrderNumber__c = orderNumber,  
                TrackingNumber__c = trackingNumber  
            );  
            opportunitiesToCreate.add(opportunity);  
        }  
    }
```

```
if (!opportunitiesToCreate.isEmpty()) {
    insert opportunitiesToCreate;
    System.debug('Successfully created ' +
numberOfOpportunities + ' opportunities.');
}

private static String generateRandomNumber(Integer length) {
    String number_x = '';
    for (Integer i = 0; i < length; i++) {
        number_x +=

String.valueOf(Math.mod(Math.abs(Crypto.getRandomInteger()), 10));
    }
    return number_x;
}
}
```

METADATA API IN SALESFORCE

Salesforce Metadata API is a powerful tool for developers and administrators to manage and deploy customizations across Salesforce orgs programmatically. It allows you to retrieve, deploy, create, update, or delete metadata (such as custom objects, fields, layouts, workflows, Apex classes, etc.) rather than manually making changes through the Salesforce user interface.

<https://github.com/certinia/apex-mdapi/blob/master/apex-mdapi/src/classes/MetadataService.cls>

```
<apex:page >  
Start_Of_Session_Id{!$Api.Session_ID}End_Of_Session_Id  
</apex:page>
```

```
public static String getSessionId(){  
    String sessionId = '';  
    PageReference reportPage = Page.SessionId;  
    String vfContent = reportPage.getContent().toString();  
  
    Integer startP = vfContent.indexOf('Start_Of_Session_Id') +  
    'Start_Of_Session_Id'.length(), endP =  
    vfContent.indexOf('End_Of_Session_Id');  
    sessionId = vfContent.substring(startP, endP);  
    return sessionId;  
}
```

List Metadata Using Metadata API

```
public static List<MetadataService.FileProperties> listMetadata(String typex){  
  
    MetadataService.MetadataPort service = createService();  
    List<MetadataService.ListMetadataQuery> queries = new  
    List<MetadataService.ListMetadataQuery>();  
    MetadataService.ListMetadataQuery genericMetadata = new  
    MetadataService.ListMetadataQuery();  
    genericMetadata.type_x = typex;  
    queries.add(genericMetadata);
```

```
MetadataService.FileProperties[] fileProperties =
service.listMetadata(queries, 50);
for(MetadataService.FileProperties prop : fileProperties){
    System.debug(prop.fullName);
}
return fileProperties;
}
```

```
public static void readMetadata(){

    MetadataService.MetadataPort service = createService();

    MetadataService.Profile profileDetails = (MetadataService.Profile)
service.readMetadata('Profile',
                     new String[] {
'Admin' }).getRecords()[0];

    MetadataService.ProfileLoginHours loginHours =
profileDetails.loginHours;
    MetadataService.ProfileLoginIpRange[] loginIpRanges =
profileDetails.loginIpRanges;
    System.debug(' loginHours \n '+loginHours);
    System.debug(' loginIpRanges \n '+loginIpRanges);

}
```

```
/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-25-2024
 * @last modified by : Amit Singh - PantherSchools
 */
```

```
public with sharing class MetadataUtils {

    public static MetadataService.MetadataPort createService() {
        MetadataService.MetadataPort service = new
        MetadataService.MetadataPort();
        service.SessionHeader = new MetadataService.SessionHeader_element();
        service.SessionHeader.sessionId = UserInfo.getSessionId();
        return service;
    }

    // MetadataUtils.listMetadata();
    public static void listMetadata(){
        MetadataService.MetadataPort service = createService();

        List<MetadataService.ListMetadataQuery> queries = new
        List<MetadataService.ListMetadataQuery>();

        MetadataService.ListMetadataQuery query = new
        MetadataService.ListMetadataQuery();
        query.type_x = 'CustomObject';

        MetadataService.ListMetadataQuery query_profile = new
        MetadataService.ListMetadataQuery();
        query_profile.type_x = 'Profile';

        queries.add(query);
        queries.add(query_profile);

        List<MetadataService.FileProperties> properties =
        service.listMetadata(queries, 60);
        System.debug(System.JSON.serializePretty(properties));
    }

    // MetadataUtils.readMetadata();
    public static void readMetadata(){
        MetadataService.MetadataPort service = createService();
```

```

List<String> itemList = new String[] { 'Admin' };
//itemList.add();

/*
    MetadataService.IReadResult result = service.readMetadata('Profile',
itemList);
    MetadataService.Profile profileDetails = (MetadataService.Profile)
result.getRecords().get(0);
*/
    MetadataService.Profile profileDetails = (MetadataService.Profile)
        service.readMetadata('Profile', itemList).getRecords()[0];
//System.debug(System.JSON.serializePretty(profileDetails));
    MetadataService.ProfileLoginHours loginHours = profileDetails.loginHours;
    MetadataService.ProfileLoginIpRange[] loginIpRanges =
profileDetails.loginIpRanges;
    System.debug(' loginHours \n '+loginHours);
    System.debug(' loginIpRanges \n '+loginIpRanges);
}

}

```

Create Object using API

```

public static void createObject(String objectLabel, String pluralLabel,
String description){

    MetadataService.MetadataPort service = createService();

    MetadataService.CustomObject customObject = new
MetadataService.CustomObject();
    customObject.fullName = objectLabel.replace(' ', '_')+'__c'; // Test
Object => Test_Object__c
    customObject.label = objectLabel;
}

```

```

        customObject.description = description;
        customObject.enableActivities = true;
        customObject.enableFeeds = true;
        customObject.enableSearch = true;
        customObject.enableReports = true;
        customObject.enableFeeds = true;

        customObject.pluralLabel = pluralLabel;
        customObject.nameField = new MetadataService.CustomField();
        customObject.nameField.type_x = 'Text';
        customObject.nameField.label = objectLabel+' Record';
        customObject.deploymentStatus = 'Deployed';
        customObject.sharingModel = 'ReadWrite';
        List<MetadataService.SaveResult> results = service.createMetadata( new
        MetadataService.Metadata[] { customObject } );
        handleSaveResults(results[0]);
    }
}

```

Create Fields using API



```

public static void createFields(String objectApiName, String fieldLabel,
Boolean required,
                                Boolean externalId, Boolean
unique, Boolean caseSensitive){

    MetadataService.MetadataPort service      = createService();
    MetadataService.CustomField customField = new
    MetadataService.CustomField();
    customField.fullName          = objectApiName+'. '+fieldLabel.replace('
', '_')+'__c'; // Account - Active => Account.Active__c
    customField.label            = fieldLabel;
    customField.type_x           = 'Text';
    customField.description       = 'This is the field created from Custom

```

```

        Metadata';
        customField.inlineHelpText      = 'This is the field created from Custom
Metadata';
        customField.length              = 60;
        customField.required            = required;
        customField.unique               = unique;
        customField.externalId          = extenalId;
        customField.defaultValue         = '"904242"';
        customField.caseSensitive        = caseSensitive;

        List<MetadataService.SaveResult> results = service.createMetadata( new
MetadataService.Metadata[] { customField } );
        handleSaveResults(results[0]);
    }
}

```

Create Lookup Fields using Metadata API

```

public static void createLookupField(String objectApiName, String
fieldLabel, String relatedToAPiName,
                                         String typex, String description,
String helpText, String relationshipLabel,
                                         String relationshipName){

    MetadataService.MetadataPort service      = createService();
    MetadataService.CustomField customField = new
MetadataService.CustomField();
    customField.fullName                  =
objectApiName+'.'+fieldLabel.replace(' ', '_')+'__c';
    customField.label                    = fieldLabel;
    customField.type_x                  = typex; // Lookup //
MasterDetail
    customField.description             = 'This is the '+typex+
field related to '+relatedToApiName +' Object';
}

```

```

        customField.inlineHelpText          = 'This is the '+typex+
field related to '+relatedToApiName +' Object';
        customField.relationshipLabel      = relationshipLabel;
        customField.relationshipName       = relationshipName;
        customField.referenceTo          = relatedToApiName;

List<MetadataService.SaveResult> results = service.createMetadata( new
MetadataService.Metadata[] { customField } );
handleSaveResults(results[0]);
}

```

Create a Picklist Field using Metadata API



```

public virtual class CustomValue extends Metadata {
    public String color;
    public Boolean default_x;
    public String fullName;
    private String[] fullName_type_info = new
String[]{"fullName",SOAP_M_URI,null,'0','1','false'};
    public String description;
    public Boolean isActive;
    public String label;

    private String[] color_type_info = new
String[]{"color",SOAP_M_URI,null,'0','1','false'};
    private String[] default_x_type_info = new
String[]{"default",SOAP_M_URI,null,'1','1','false'};
    private String[] description_type_info = new
String[]{"description",SOAP_M_URI,null,'0','1','false'};
    private String[] isActive_type_info = new
String[]{"isActive",SOAP_M_URI,null,'0','1','false'};
    private String[] label_type_info = new

```

```

String[]{'label',SOAP_M_URI,null,'0','1','false'};
    private String[] apex_schema_type_info = new
String[] {SOAP_M_URI,'true','false'};
    private String[] field_order_type_info = new
String[] {'fullName','color','default_x','description','isActive','label'}
;
}

```



```

public static void createPicklistField(String objectApiName, String
fieldLabel, List<String> options){

    MetadataService.MetadataPort service      = createService();
    MetadataService.CustomField customField = new
MetadataService.CustomField();
    customField.fullName          =
objectApiName+'.'+fieldLabel.replace(' ', '_')+'__c';
    customField.label            = fieldLabel;
    customField.type_x           = 'Picklist';
    customField.description     = 'This is the picklist field created
from Custom Metadata while recording UDEMY INTEGRATION';
    customField.inlineHelpText   = 'This is the picklist field
created from Custom Metadata while recording UDEMY INTEGRATION';

    MetadataService.ValueSet valueSetDef          = new
MetadataService.ValueSet();

    MetadataService.ValueSetValuesDefinition valueDefinition = new
MetadataService.ValueSetValuesDefinition();

    List<MetadataService.CustomValue> values          = new
List<MetadataService.CustomValue>();

```

```
        for(String value : options){

            MetadataService.CustomValue customValue = new
MetadataService.CustomValue();
            customValue.fullName   = value;
            customValue.default_x = false;
            customValue.isActive  = true;
            customValue.label     = value;
            values.add(customValue);
        }

        valueDefinition.value           = values;
        valueDefinition.sorted          = true;

        valueSetDef.valueSetDefinition = valueDefinition;
        valueSetDef.restricted        = true;

        customField.valueSet           = valueSetDef;
        customField.required          = true;
        List<MetadataService.SaveResult> results = service.createMetadata(
new MetadataService.Metadata[] { customField } );
        handleSaveResults(results[0]);
    }
}
```

Panu

Delete Custom Metadata Record using metadata API

```
public static void deleteMetadataRecord(String metadataType, List<String>
```

```

names){

    MetadataService.MetadataPort service = createService();
    List<String> recordsToDelete = new List<String>();
    for(String name : names){
        recordsToDelete.add(metadataType+'.'+name);
    }
    service.deleteMetadata('CustomMetadata', recordsToDelete); //250
}

```

For More Examples -

<https://gist.github.com/amitastreait/267ea91b57629eae6de4c63c0962e928>

Complete Apex Class

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 05-26-2024
 * @last modified by : Amit Singh - PantherSchools
 **/


public with sharing class MetadataUtils {

    public static String getSessionId(){
        String sessionId = '';
        PageReference sessionPage = Page.SessionId;
        String vfContent = sessionPage.getContent().toString();
        /*String tempSessionId = vfContent.substringAfter('Start_Of_Session_Id');
        sessionId = tempSessionId.substringBeforeLast('End_Of_Session_Id');*/
        Integer startP = vfContent.indexOf('Start_Of_Session_Id') +
        'Start_Of_Session_Id'.length();
        Integer endP = vfContent.indexOf('End_Of_Session_Id');
        sessionId = vfContent.substring(startP, endP);
        return sessionId;
    }
}

```

```
}

public static MetadataService.MetadataPort createService() {
    MetadataService.MetadataPort service = new
MetadataService.MetadataPort();
    service.SessionHeader = new MetadataService.SessionHeader_element();
    service.SessionHeader.sessionId = getSessionId();
    return service;
}

// MetadataUtils.listMetadata();
public static void listMetadata(){
    MetadataService.MetadataPort service = createService();

    List<MetadataService.ListMetadataQuery> queries = new
List<MetadataService.ListMetadataQuery>();

    MetadataService.ListMetadataQuery query = new
MetadataService.ListMetadataQuery();
    query.type_x = 'CustomObject';

    MetadataService.ListMetadataQuery query_profile = new
MetadataService.ListMetadataQuery();
    query_profile.type_x = 'Profile';

    queries.add(query);
    queries.add(query_profile);

    List<MetadataService.FileProperties> properties =
service.listMetadata(queries, 60);
    System.debug(System.JSON.serializePretty(properties));
}

// MetadataUtils.readMetadata();
public static void readMetadata(){
    MetadataService.MetadataPort service = createService();
```

```
List<String> itemList = new String[] { 'Admin' };
//itemList.add();

/*
    MetadataService.IReadResult result = service.readMetadata('Profile',
itemList);
    MetadataService.Profile profileDetails = (MetadataService.Profile)
result.getRecords().get(0);
*/
    MetadataService.Profile profileDetails = (MetadataService.Profile)
        service.readMetadata('Profile', itemList).getRecords()[0];
//System.debug(System.JSON.serializePretty(profileDetails));
    MetadataService.ProfileLoginHours loginHours = profileDetails.loginHours;
    MetadataService.ProfileLoginIpRange[] loginIpRanges =
profileDetails.loginIpRanges;
    System.debug(' loginHours \n '+loginHours);
    System.debug(' loginIpRanges \n '+loginIpRanges);
}

// MetadataUtils.createObject();
public static void createObject(String label, String pluralLabel){
    MetadataService.MetadataPort service = createService();
    MetadataService.CustomObject customObject = new
MetadataService.CustomObject();
    customObject.label = label;
    customObject.pluralLabel = pluralLabel;
    customObject.fullName = label.deleteWhitespace()+'__c';
    customObject.enableHistory = true;
    customObject.enableActivities = true;
    customObject.enableSearch = true;
    customObject.enableReports = true;

    MetadataService.CustomField nameField = new
MetadataService.CustomField();
    nameField.type_x = 'Text';
```

```

nameField.label = 'Stripe Invoice ID';

customObject.deploymentStatus = 'Deployed';
customObject.sharingModel = 'ReadWrite';

customObject.nameField = nameField;

List<MetadataService.Metadata> customObjectList = new
List<MetadataService.Metadata>();
customObjectList.add(customObject);

List<MetadataService.SaveResult> saveResult =
service.createMetadata(customObjectList);
System.debug(System.JSON.serializePretty(saveResult));
}

// MetadataUtils.createFields();
public static void createFields(){

    MetadataService.MetadataPort service      = createService();

    MetadataService.CustomField customField = new
MetadataService.CustomField();
    // ObjectApiName.FieldName
    customField.fullName      = 'StripeInvoice__c.StripeUUID__c'; // Account -
Active => Account.Active__c
    customField.label        = 'Stripe UUID';
    customField.type_x       = 'Text';
    customField.description   = 'This is the field created from Salesforce
Metadata API';
    customField.inlineHelpText = 'This is the field created from Salesforce
Metadata API';
    customField.length        = 120;
    customField.required      = false;
    customField.unique        = false;
}

```

```

customField.externalId      = true;
//customField.defaultValue    = '"904242"';
//customField.caseSensitive   = true;
List<MetadataService.Metadata> metadata = new
List<MetadataService.Metadata>();
metadata.add(customField);
List<MetadataService.SaveResult> results = service.createMetadata(
metadata );
System.debug(System.JSON.serializePretty(results));
}
// MetadataUtils.createLookupFields();
public static void createLookupFields(){

    MetadataService.MetadataPort service    = createService();
    MetadataService.CustomField customField = new
MetadataService.CustomField();
    customField.fullName                  = 'StripeInvoice__c.StripeCustomer__c';
    customField.label                    = 'Stripe Customer';
    customField.type_x                  = 'Lookup'; // Lookup // MasterDetail
    customField.description             = 'This is the Lookup field related to
Account Object';
    customField.inlineHelpText          = 'This is the Lookup field related to
Account Object';
    customField.relationshipLabel       = 'Stripe Invoices';
    customField.relationshipName        = 'StripeInvoices';
    customField.referenceTo            = 'Account';

List<MetadataService.SaveResult> results = service.createMetadata( new
MetadataService.Metadata[] { customField } );
System.debug(System.JSON.serializePretty(results));
}
/*
List<String> options = new List<String>{
    'Open',
    'Closed',

```

```
'Paid',
'Voided',
'Part-Paid'

};

MetadataUtils.createPicklistField(options);

*/
public static void createPicklistField(List<String> options){

    MetadataService.MetadataPort service = createService();

    MetadataService.CustomField customField = new
MetadataService.CustomField();

    customField.fullName      = 'StripeInvoice__c.InvoiceStatus__c';
    customField.label        = 'Invoice Status';
    customField.type_x       = 'Picklist';
    customField.description   = 'This is the picklist field created from
Salesforce Metadata API';
    customField.inlineHelpText = 'This is the picklist field created from
Salesforce Metadata API';

    MetadataService.ValueSet valueSetDef      = new
MetadataService.ValueSet();

    MetadataService.ValueSetValuesDefinition valueDefinition = new
MetadataService.ValueSetValuesDefinition();

    List<MetadataService.CustomValue> values      = new
List<MetadataService.CustomValue>();
    for(String value : options){
        MetadataService.CustomValue customValue = new
MetadataService.CustomValue();
        customValue.fullName = value;
        customValue.default_x = false;
        customValue.isActive = true;
```

```

        customValue.label    = value;
        values.add(customValue);
    }

    valueDefinition.value          = values;
    valueDefinition.sorted         = true;

    valueSetDef.valueSetDefinition = valueDefinition;
    valueSetDef.restricted        = true;

    customField.valueSet          = valueSetDef;
    customField.required          = true;

    List<MetadataService.SaveResult> results = service.createMetadata( new
    MetadataService.Metadata[] { customField } );
    System.debug(System.JSON.serializePretty(results));
    handleSaveResults(results.get(0));
}

/*
List<String> names = new List<String>{'PS_ZoomToken'};
MetadataUtils.deleteMetadataRecord('Google_Config', names);
*/
public static void deleteMetadataRecord(String metadataType, List<String>
names){

    MetadataService.MetadataPort service = createService();
    // MetadataObjectName.RecordName
    // Google_Config.PS_ZoomToken
    List<String> recordsToDelete = new List<String>();
    for(String name : names){
        recordsToDelete.add(metadataType+'.'+name);
    }
    MetadataService.DeleteResult[] deleteResults =
service.deleteMetadata('CustomMetadata', recordsToDelete); //250
    handleDeleteResults(deleteResults.get(0));
}

public class MetadataServiceExamplesException extends Exception { }

```

```

public static void handleSaveResults(MetadataService.SaveResult saveResult){
    // Nothing to see?
    if(saveResult==null || saveResult.success)
        return;
    // Construct error message and throw an exception
    System.debug(' saveResult.errors \n '+saveResult.errors);
    if(saveResult.errors!=null){
        List<String> messages = new List<String>();
        messages.add(
            (saveResult.errors.size()==1 ? 'Error ' : 'Errors ') +
            'occured processing component ' + saveResult.fullName + ':');
        for(MetadataService.Error error : saveResult.errors) {
            messages.add(
                error.message + ' (' + error.statusCode + ')' +
                ( error.fields!=null && error.fields.size()>0 ?
                    ' Fields ' + String.join(error.fields, ',') + ' : ' ) );
        }
        if(messages.size()>0)
            throw new MetadataServiceExamplesException(String.join(messages, ','));
    }

    System.debug(' Message ' +String.join(messages, ' '));
}

if(!saveResult.success)
    throw new MetadataServiceExamplesException('Request failed with no
specified error!');

}

public static void handleDeleteResults(MetadataService.DeleteResult
deleteResult){
    // Nothing to see?
    if(deleteResult==null || deleteResult.success)
        return;
    // Construct error message and throw an exception

```

```

if(deleteResult.errors!=null){
    List<String> messages = new List<String>();
    messages.add(
        (deleteResult.errors.size()==1 ? 'Error ' : 'Errors ') +
        'occured processing component ' + deleteResult.fullName + ':');
    for(MetadataService.Error error : deleteResult.errors)
        messages.add(
            error.message + ' (' + error.statusCode + ')' +
            ( error.fields!=null && error.fields.size()>0 ?
                ' Fields ' + String.join(error.fields, ',') + ' : ' ) );
    if(messages.size()>0)
        throw new MetadataServiceExamplesException(String.join(messages, ','));
}
if(!deleteResult.success)
    throw new MetadataServiceExamplesException('Request failed with no
specified error!');

}

```

```

public static void handleUpsertResults(MetadataService.UpsertResult
upsertResult){
    // Nothing to see?
    if(upsertResult==null || upsertResult.success)
        return;
    // Construct error message and throw an exception
    if(upsertResult.errors!=null){
        List<String> messages = new List<String>();
        messages.add(
            (upsertResult.errors.size()==1 ? 'Error ' : 'Errors ') +
            'occured processing component ' + upsertResult.fullName + ':');
        for(MetadataService.Error error : upsertResult.errors)
            messages.add(
                error.message + ' (' + error.statusCode + ')' +
                ( error.fields!=null && error.fields.size()>0 ?

```

```
        ' Fields ' + String.join(error.fields, ',') + ' : " ) );  
    if(messages.size()>0)  
        throw new MetadataServiceExamplesException(String.join(messages, ','));  
    }  
    if(!upsertResult.success)  
        throw new MetadataServiceExamplesException('Request failed with no  
specified error!');  
    }  
  
}
```

Bulk API

The Bulk API in Salesforce is designed for handling large volumes of data by allowing asynchronous processing of records in batches. This API is particularly useful for efficiently processing data loads that involve large datasets, minimising system resource usage and optimising performance.

Key Features of Bulk API:

1. **Asynchronous Processing:** Data is processed in batches, allowing operations to be queued and executed when system resources are available.
2. **High Performance:** Designed to handle large data volumes efficiently, reducing processing time.
3. **Batch Processing:** Data is sent in batches, allowing for parallel processing and improved throughput.
4. **Error Handling:** Detailed error reporting helps identify and resolve issues with individual records within batches.

Use Cases for Bulk API:

1. **Data Migration:** Ideal for migrating large datasets from legacy systems to Salesforce.
2. **Data Integration:** Efficient for integrating data from external systems,

especially when dealing with high volumes.

3. **Mass Data Operations:** Useful for performing bulk updates, inserts, deletes, or upserts on Salesforce records.
4. **Data Archival:** Helps in archiving old data by exporting large datasets for storage outside Salesforce.
5. **Data Cleansing:** Facilitates bulk updates or deletions during data cleansing processes to maintain data quality.

Example Scenarios:

- **Migrating Legacy Data:** A company moving from an old CRM system to Salesforce can use the Bulk API to import millions of customer records efficiently.
- **Periodic Data Sync:** A retail business synchronising inventory data from its ERP system to Salesforce can leverage the Bulk API to handle daily updates involving large datasets.
- **Mass Updates:** A marketing team needing to update the status of thousands of leads after a campaign can use the Bulk API to perform these updates quickly.

Resource -

https://developer.salesforce.com/docs/atlas.en-us.api_asynch.meta/api_asynch/bulk_api_2_0.htm

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 05-22-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
public with sharing class BulkAPIService {  
  
    public static void processCSVFromFiles(String fileTitle) {
```

```
ContentVersion file = [SELECT Id, Title, VersionData FROM ContentVersion WHERE Title = :fileTitle LIMIT 1];
Blob csvBlob = file.VersionData;
String csvContent = csvBlob.toString();
bulkAPIOperation(csvContent);

}

private static void bulkAPIOperation(String csvContent) {
    HttpRequest req = new HttpRequest();
    req.setEndpoint(URL.getOrgDomainUrl().toExternalForm() +
'/services/data/v60.0/jobs/ingest/');
    req.setMethod('POST');
    req.setHeader('Content-Type', 'application/json');
    req.setHeader('Authorization', 'Bearer ' +
UserInfo.getSessionId());

    String jobBody =
'{"object":"Account","operation":"insert","contentType":"CSV","lineEnding": "LF"}';
    req.setBody(jobBody);

    Http http = new Http();
    HttpResponse res = http.send(req);
    System.debug(res.getBody());
    if (res.getStatusCode() != 200) {
        System.debug('Failed to create job: ' + res.getBody());
        return;
    }

    Map<String, Object> jobInfo = (Map<String, Object>)
JSON.deserializeUntyped(res.getBody());
    String jobId = (String) jobInfo.get('id');

    HttpRequest batchReq = new HttpRequest();
    batchReq.setEndpoint(URL.getOrgDomainUrl().toExternalForm() +
```

```
'/services/data/v60.0/jobs/ingest/' + jobId + '/batches/');
batchReq.setMethod('PUT');
batchReq.setHeader('Content-Type', 'text/csv');
batchReq.setHeader('Authorization', 'Bearer ' +
UserInfo.getSessionId());
batchReq.setBody(csvContent);

HttpResponse batchRes = http.send(batchReq);
System.debug(batchRes.getBody());
if (batchRes.getStatusCode() != 201 && batchRes.getStatusCode() != 200) {
    System.debug('Failed to add batch: ' + batchRes.getBody());
    return;
}

HttpRequest closeReq = new HttpRequest();
closeReq.setEndpoint(URL.getOrgDomainUrl().toExternalForm() +
'/services/data/v60.0/jobs/ingest/' + jobId);
closeReq.setMethod('PATCH');
closeReq.setHeader('Content-Type', 'application/json');
closeReq.setHeader('Authorization', 'Bearer ' +
UserInfo.getSessionId());
closeReq.setBody('{"state": "UploadComplete"}');

HttpResponse closeRes = http.send(closeReq);
System.debug(closeRes.getBody());
if (closeRes.getStatusCode() != 200 && closeRes.getStatusCode() != 201) {
    System.debug('Failed to close job: ' + closeRes.getBody());
}
}
```

WRITE UNIT TESTS FOR INTEGRATION

Why Unit Test

Writing unit tests for integration in Salesforce is crucial for several reasons, especially considering the platform's complexity and the interdependence of its components. Here are some key reasons why integration testing is important in Salesforce:

1. Ensuring Correct Integration Between Components

Salesforce applications often involve multiple components like Apex classes, Visualforce pages, Lightning components, triggers, and workflows. Integration tests ensure that these components work together as expected. For example, if a trigger and an Apex class both update a record, integration tests verify that their combined effect is correct.

2. Preventing Regression Bugs

Integration tests help catch regression bugs when changes are made to one part of the system. They ensure that new changes do not break existing functionality. For instance, if a new feature is added to an Apex class, integration tests will help ensure that it doesn't negatively impact the existing integrations with other classes or components.

3. Maintaining Data Integrity

Salesforce applications heavily rely on data integrity. Integration tests ensure data remains consistent and valid across different operations and components. For example, if a business process involves multiple objects like Accounts, Contacts, and Opportunities, integration tests can verify that the data flow between these objects remains correct after any changes.

4. Validating Business Processes

Salesforce often automates complex business processes using Process Builder, Flow, and Apex. Integration tests validate these processes end-to-end, ensuring that the entire workflow functions as expected. For instance, creating an Account

should trigger related Contact creation and send notifications; integration tests can verify this flow.

5. Improving Code Quality and Reliability

By writing comprehensive integration tests, developers can ensure higher code quality and reliability. These tests help identify and fix issues early in the development cycle, reducing the likelihood of bugs in production. Reliable code leads to better user experiences and reduces maintenance costs.

6. Facilitating Safe Refactoring

Refactoring is a common practice to improve code quality and maintainability. With a robust suite of integration tests, developers can refactor code confidently, knowing that the tests will catch any unintended side effects.

7. Meeting Compliance and Governance Requirements

For industries with strict regulatory requirements, integration tests help ensure compliance by verifying that business processes adhere to required standards. They provide an additional layer of validation, ensuring that all parts of the system interact correctly according to predefined rules.

Creating a Mock Test for a REST API

Let's assume you have an Apex class that makes a callout to an external REST API to fetch some data.

```
public with sharing class AccountService {
    public static String getAccountData(String accountId) {
        Http http = new Http();
        HttpRequest request = new HttpRequest();
        request.setEndpoint('https://api.example.com/account/' +
                           accountId);
        request.setMethod('GET');

        HttpResponse response = http.send(request);
        if (response.getStatusCode() == 200) {
            return response.getBody();
        }
    }
}
```

```
    } else {
        throw new CalloutException('Failed to get account data. Status
code: ' + response.getStatusCode());
    }
}
```

Mock Class Interface

Create a mock class that implements the `HttpCalloutMock` interface.

https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apexref/apex_interface_http_calloutmock.htm

```
@isTest
global class AccountServiceMock implements HttpCalloutMock {
    global HttpResponse respond(HttpRequest req) {
        // Create a new instance of HttpResponse
        HttpResponse res = new HttpResponse();
        res.setHeader('Content-Type', 'application/json');
        res.setBody('{"id": "0011w00000FFkm7AAD", "name": "Test Account",
"phone": "1234567890"}');
        res.setStatusCode(200);
        return res;
    }
}
```

Example Test Class

Now, create a test class to use the mock response.

```
@isTest
public class AccountServiceTest {
    @isTest
    static void testGetAccountData() {
        // Set mock callout class
        Test.setMock(HttpCalloutMock.class, new
```

```

MockHttpResponseGenerator());

        // Perform the callout
        String result =
AccountService.getAccountData('0011w00000FFkm7AAD');

        // Verify the results
        System.assertEquals(null, result, 'Result should not be null');
        System.assert(result.contains('"id": "0011w00000FFkm7AAD"'),
'Result should contain the account ID');
        System.assert(result.contains('"name": "Test Account"'), 'Result
should contain the account name');
        System.assert(result.contains('"phone": "1234567890"'), 'Result
should contain the account phone');
    }
}

```

Test Class We developed in the Class

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-08-2024
 * @last modified by : Amit Singh - PantherSchools
 */
@IsTest
public with sharing class PS_AccountManagerTest {

    @IsTest
    public static void deleteRecordsTest(){

        Account acc = new Account(
            Name = 'Salesforce.com'
        );
    }
}

```

```
insert acc;

RestRequest req = new RestRequest();
req.requestURI = '/v1/PS_AccountManager/'+acc.Id;

RestContext.request = req;

Test.startTest();
PS_AccountManager.deleteRecords();
Test.stopTest();
}

@IsTest
private static void getAccountListTest(){
    Account acc = new Account(
        Name = 'Salesforce.com'
    );
    insert acc;

    RestRequest req = new RestRequest();
    req.requestURI = '/v1/PS_AccountManager/'+acc.Id;

    RestContext.request = req;

    Test.startTest();
    Account accRecord = PS_AccountManager.getAccountList();
    Test.stopTest();
    Assert.AreEqual('Salesforce.com', accRecord.Name, 'The account
name does not match!');
}

@IsTest
private static void createAccountTest(){
    RestRequest req = new RestRequest();
    req.requestURI = '/v1/PS_AccountManager/';
```

```
String requestBody = '{'+
    '"Name" : "Salesforce.com",'+
    '"Rating": "Hot"' +
'}' ;

req.requestBody = Blob.valueOf(requestBody);

RestContext.request = req;
Test.startTest();
PS_AccountManager.createAccount();
Test.stopTest();
}

@IsTest
private static void createAccountErrorTest(){
    RestRequest req = new RestRequest();
    req.requestURI = '/v1/PS_AccountManager/';

    RestContext.request = req;

    RestResponse res = new RestResponse();
    RestContext.response = res;

    Test.startTest();
    PS_AccountManager.createAccount();
    Test.stopTest();
}

@IsTest
private static void updateAccountTest(){
    Account acc = new Account(
        Name = 'Salesforce.com'
    );
    insert acc;
```

```

RestRequest req = new RestRequest();
req.requestURI = '/v1/PS_AccountManager/' + acc.Id;

req.addHeader('X-API-Key', 'api_key_uuyuy564');
req.addParameter('FirstName', 'Amit Singh');

String requestBody = '{' +
    '"accountName" : "PantherSchools.com", ' +
    '"active" : "Yes", ' +
    '"accountIndustry" : "Education", ' +
    '"AnnualRevenue" : 8983454.345' +
'}';
req.requestBody = Blob.valueOf(requestBody);

RestContext.request = req;
Test.startTest();
Account accRecord = PS_AccountManager.updateAccount();
Test.stopTest();

Assert.AreEqual('PantherSchools.com', accRecord.Name, 'The account
name does not match!');
Assert.AreEqual('Education', accRecord.Industry);
}

}

```

Unit test for Apex REST Service

Writing test classes for Apex REST custom web services in Salesforce involves several key steps. You need to create mock data, call the REST endpoints, and verify the responses. Salesforce provides built-in functionalities to facilitate the testing of REST endpoints. Below is a comprehensive guide on how to write test classes for an Apex REST custom web service.

Example Apex REST Service

Assume we have a simple REST service that handles GET and POST requests for an Account object.

```
@RestResource(urlMapping='/accounts/*')
global with sharing class AccountRestService {

    @HttpGet
    global static Account getAccount() {
        RestRequest req = RestContext.request;
        String accountId =
            req.requestURI.substring(req.requestURI.lastIndexOf('/') + 1);
        Account result = [SELECT Id, Name, Phone FROM Account WHERE Id =
            :accountId LIMIT 1];
        return result;
    }

    @HttpPost
    global static String createAccount(String name, String phone) {
        Account acc = new Account(Name = name, Phone = phone);
        insert acc;
        return acc.Id;
    }
}
```

Example Test Class

Now, let's write a test class for this REST service.

```
@isTest
private class AccountRestServiceTest {

    @testSetup
    static void setupTestData() {
        // Setup test data
    }
}
```

```
    Account testAcc = new Account(Name = 'Test Account', Phone =
'1234567890');
    insert testAcc;
}

@isTest
static void testGetAccount() {
    // Fetch the test account
    Account testAcc = [SELECT Id, Name FROM Account WHERE Name = 'Test
Account' LIMIT 1];

    // Set up the RestRequest
    RestRequest req = new RestRequest();
    RestResponse res = new RestResponse();
    req.requestURI = '/services/apexrest/accounts/' + testAcc.Id;
    req.httpMethod = 'GET';
    RestContext.request = req;
    RestContext.response = res;

    // Call the method
    Account result = AccountRestService.getAccount();

    // Verify the response
    System.assertNotEquals(null, result, 'Account should be
returned');
    System.assertEquals('Test Account', result.Name, 'Account name
should match');
    System.assertEquals('1234567890', result.Phone, 'Account phone
should match');
}

@isTest
static void testCreateAccount() {
    // Set up the RestRequest
    RestRequest req = new RestRequest();
```

```

RestResponse res = new RestResponse();
req.requestURI = '/services/apexrest/accounts/';
req.httpMethod = 'POST';
req.addParameter('name', 'New Test Account');
req.addParameter('phone', '0987654321');
RestContext.request = req;
RestContext.response = res;

// Call the method
String newAccountId = AccountRestService.createAccount('New Test
Account', '0987654321');

// Verify the account was created
Account newAcc = [SELECT Id, Name, Phone FROM Account WHERE Id =
:newAccountId LIMIT 1];
System.assertEquals(null, newAcc, 'New account should be
created');
System.assertEquals('New Test Account', newAcc.Name, 'New account
name should match');
System.assertEquals('0987654321', newAcc.Phone, 'New account phone
should match');
}
}

```

Test Classed developer in the test class

Error Mock Class

```

/**
 * @description      :
 * @author           : Amit Singh - PantherSchools
 * @group            :
 * @last modified on : 06-08-2024

```

```

* @last modified by : Amit Singh - PantherSchools
*/
@IsTest
public with sharing class Assignment_CaseTriggerErrorMock
implements HttpCalloutMock {
    public HttpResponse respond(HttpRequest req){

        String json = '{ "errorMessage": "This is an error
message!" }';

        HttpResponse response = new HttpResponse();
        response.setBody(json);
        response.setHeader('Content-Type',
'application/json');
        response.setStatusCode(403);
        response.setStatus('ERROR');
        return response;
    }
}

```

Success Mock Class

```

/**
* @description      :
* @author           : Amit Singh - PantherSchools
* @group            :
* @last modified on : 06-08-2024
* @last modified by : Amit Singh - PantherSchools
*/

```

```
@IsTest

public with sharing class Assignment_CaseTriggerMock implements
HttpCalloutMock {
    public HttpResponse respond(HttpRequest req){
        String json=      '{'+
            '"ticket": {'+
                '"url": "https://pantherschoolhelp.zendesk.com/api/v2/tickets/14.json",'+
                    '"id": 14,'+
                    '"external_id": null,'+
                    '+
                    '"created_at": "2024-03-15T08:39:37Z",'+
                    '"updated_at": "2024-03-15T08:39:37Z",'+
                    '"generated_timestamp": 0,'+
                    '"type": "incident",'+
                    '"subject": "My printer is on fire!",'+
                    '"raw_subject": "My printer is on fire!",'+
                    '"description": "The smoke is very colorful.",'+
                    '"priority": "urgent",'+
                    '"status": "open",'+
                    '"recipient": null,'+
                    '"requester_id": 17691919870738,'+
                    '"submitter_id": 17691919870738,'+
                    '"assignee_id": 17691919870738,'+
                    '"organization_id": 17691875790994,'+
                    '"group_id": 17691920139922,'+
                    '"forum_topic_id": null,'+
                    '"problem_id": null,'+
                    '"has_incidents": false,'+
                    '"is_public": true,'+
                    '"due_at": null,'+
                    '+
                    '"followup_ids": [],'+
                    '"ticket_form_id": 17691875688594,'+
                    '+
                    '"last_update": "2024-03-15T08:39:37Z",'+
                    '"last_status_change": "2024-03-15T08:39:37Z",'+
                    '"last_note": "Initial report from user.",'+
                    '"last_email": "user@example.com",'+
                    '"last_email_time": "2024-03-15T08:39:37Z",'+
                    '"last_email_type": "User",'+
                    '"last_email_subject": "Printer issue",'+
                    '"last_email_body": "My printer is on fire! The smoke is very colorful.",'+
                    '"last_email_recipient": "user@example.com",'+
                    '"last_email_from": "user@example.com",'+
                    '"last_email_to": "user@example.com",'+
                    '"last_email_cc": "user@example.com",'+
                    '"last_email_bcc": "user@example.com",'+
                    '"last_email_reply_to": "user@example.com",'+
                    '"last_email_in_reply_to": "user@example.com",'+
                    '"last_email_references": "user@example.com",'+
                    '"last_email_content_type": "text/plain",'+
                    '"last_email_charset": "UTF-8",'+
                    '"last_email_content": "My printer is on fire! The smoke is very colorful.",'+
                    '"last_email_headers": "Content-Type: text/plain; charset=UTF-8",'+
                    '"last_email_attachments": "[]",'+
                    '"last_email_is_html": false,'+
                    '"last_email_is_text": true,'+
                    '"last_email_is_rtf": false,'+
                    '"last_email_is_pdf": false,'+
                    '"last_email_is_image": false,'+
                    '"last_email_is_binary": false,'+
                    '"last_email_is_attachment": false,'+
                    '"last_email_is_email": true,'+
                    '"last_email_is_file": false,'+
                    '"last_email_is_link": false,'+
                    '"last_email_is_text_file": false,'+
                    '"last_email_is_rtf_file": false,'+
                    '"last_email_is_pdf_file": false,'+
                    '"last_email_is_image_file": false,'+
                    '"last_email_is_binary_file": false,'+
                    '"last_email_is_attachment_file": false,'+
                    '"last_email_is_email_file": false,'+
                    '"last_email_is_file_file": false,'+
                    '"last_email_is_link_file": false,'+
                    '"last_email_is_text_file_file": false,'+
                    '"last_email_is_rtf_file_file": false,'+
                    '"last_email_is_pdf_file_file": false,'+
                    '"last_email_is_image_file_file": false,'+
                    '"last_email_is_binary_file_file": false,'+
                    '"last_email_is_attachment_file_file": false,'+
                    '"last_email_is_email_file_file": false,…



                    '}'
    }
}
```

```

        "brand_id": 17691875769234,'+
        "allow_channelback": false,'+
        "allow_attachments": true,'+
        "from.messaging_channel": false'+
    }'+
};

HttpResponse response = new HttpResponse();
response.setBody(json);
response.setHeader('Content-Type', 'application/json');
response.setStatusCode(200);
response.setStatus('OK');
return response;
}
}

```

Test Class using both Success/Error Mock Class

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-08-2024
 * @last modified by : Amit Singh - PantherSchools
 */
@IsTest
public with sharing class Assignment_CaseTriggerTest {

    @IsTest
    private static void createCaseTest(){
        Account acc = new Account(
            Name = 'Salesforce.com'
        );
        insert acc;
        Contact con = new Contact(

```

```
        FirstName = 'Amit',
        LastName = 'Singh',
        Email = 'amit.singh@gmail.com',
        Phone = '9087654321',
        AccountId = acc.Id
    );
    insert con;
Case caseRecord = new Case(
    ContactId = con.Id,
    Subject = 'Test Class',
    Description = 'Test Class',
    Status = 'New',
    Priority = 'High',
    Origin = 'Web'
);
Test.startTest();
Test.setMock(HttpCalloutMock.class, new
Assignment_CaseTriggerMock());
insert caseRecord;
Test.stopTest();
}

@IsTest
private static void createCaseErrorTest(){
    Account acc = new Account(
        Name = 'Salesforce.com'
    );
    insert acc;
    Contact con = new Contact(
        FirstName = 'Amit',
        LastName = 'Singh',
        Email = 'amit.singh@gmail.com',
        Phone = '9087654321',
        AccountId = acc.Id
    );
}
```

```
insert con;
Case caseRecord = new Case(
    ContactId = con.Id,
    Subject = 'Test Class',
    Description = 'Test Class',
    Status = 'New',
    Priority = 'High',
    Origin = 'Web'
);
Test.startTest();
Test.setMock(HttpCalloutMock.class, new
Assignment_CaseTriggerErrorMock());
insert caseRecord;
Test.stopTest();
List<Case> caseList = [SELECT Id, CaseNumber, Subject,
ZendeskErrorMessage__c FROM Case WHERE Id =: caseRecord.Id LIMIT 1];
if(caseList?.size() > 0){
    Case updateCase = caseList.get(0);
    System.debug(updateCase.ZendeskErrorMessage__c);
    Assert.AreEqual('{ "errorMessage": "This is an error message!"',
    updateCase.ZendeskErrorMessage__c, 'Error message is not matching!');
}
}
```



Test Class for Metadata API

To develop the Apex Class for the SOAP API, we need to create the Mock class by implementing the WebServiceMock interface

Here is the link to interface -

https://developer.salesforce.com/docs/atlas.en-us.apexref.meta/apex_interface_webservicemock.htm#apex_System_WebServiceMock_doinvoke

Unit Test that we developed in the Apex Class

Mock Class

```
/**  
 * @description      :  
 * @author           : Amit Singh - PantherSchools  
 * @group            :  
 * @last modified on : 06-08-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
@IsTest  
public with sharing class MetadataUtilsMock implements WebServiceMock {  
  
    public void doInvoke(Object stub, Object soapRequest,  
Map<String, Object> responseMap, String endpoint,  
        String soapAction, String requestName,  
        String responseNamespace, String responseName, String  
responseType){  
        if(soapRequest instanceof MetadataService.listMetadata_element ){  
            MetadataService.listMetadataResponse_element response_x = new  
MetadataService.listMetadataResponse_element();  
            responseMap.put('response_x', response_x);  
        } else if(soapRequest instanceof  
MetadataService.readMetadata_element){  
            MetadataService.Profile profile = new
```

```

MetadataService.Profile();
    profile.fullName = 'Admin';

        MetadataService.ReadProfileResult result = new
MetadataService.ReadProfileResult();
    result.records = new MetadataService.Profile[] { profile };

        MetadataService.readProfileResponse_element responseElement =
new MetadataService.readProfileResponse_element();
    responseElement.result = result;

        responseMap.put('response_x', responseElement);
} else if(soapRequest instanceof
MetadataService.createMetadata_element){
    MetadataService.createMetadataResponse_element response_x =
new MetadataService.createMetadataResponse_element();
    responseMap.put('response_x', response_x);
}

}
}

```

Test Class

```

/**
* @description      :
* @author          : Amit Singh - PantherSchools
* @group           :
* @last modified on : 06-08-2024
* @last modified by : Amit Singh - PantherSchools
*/
@IsTest
public with sharing class MetadataUtilsTest {

```

```
@IsTest
public static void listMetadataTest(){
    Test.setMock(WebServiceMock.class, new MetadataUtilsMock());
    Test.startTest();
    MetadataUtils.listMetadata();
    Test.stopTest();
}

@IsTest
public static void readMetadataTest(){
    Test.setMock(WebServiceMock.class, new MetadataUtilsMock());
    Test.startTest();
    MetadataUtils.readMetadata();
    Test.stopTest();
}

@IsTest
public static void createObjectTest(){
    Test.setMock(WebServiceMock.class, new MetadataUtilsMock());
    Test.startTest();
    MetadataUtils.createObject('Invoice', 'Invoices');
    Test.stopTest();
}

@IsTest
public static void createFieldsTest(){
    Test.setMock(WebServiceMock.class, new MetadataUtilsMock());
    Test.startTest();
    MetadataUtils.createFields('Account', 'Stripe UUID');
    Test.stopTest();
}
```

External Objects & Cross Org Adopter

External Objects in Salesforce

Link - <https://orderdb.herokuapp.com/orders.svc/>

External Objects in Salesforce are a powerful feature designed to enable seamless integration with data stored outside of Salesforce. They allow users to access and interact with external data sources as if they were native Salesforce objects. This is especially useful for integrating large data sets that reside outside Salesforce, like in legacy systems, databases, or other cloud services.

Key Concepts of External Objects

1. Data Integration Without Duplication:

- External objects let you integrate external data without importing it into Salesforce, avoiding duplication and saving storage costs.

2. External Data Source:

- An external data source is a Salesforce configuration that specifies how to connect to the external system. It includes details like the URL endpoint, authentication settings, and other connection properties.

3. Read-Only or Read-Write Access:

- Depending on the external data source and configuration, external objects can support both read-only and read-write access.

4. Data in Real-Time:

- Since the data is accessed in real-time, any updates in the external system are immediately visible in Salesforce, ensuring data consistency.

How External Objects Work

1. Define External Data Source:

- In Salesforce Setup, you create an external data source that defines how Salesforce connects to the external system.

2. Create External Objects:

- Once the external data source is defined, you create external objects in Salesforce. These objects map to tables or entities in the external system.

3. External Object Fields:

- Define fields for the external objects, mapping them to the fields in the external system.

Supported Protocols and Adapters

Salesforce supports several protocols and adapters to connect to external data sources:

1. OData 2.0 and 4.0 and 4.01:

- Salesforce can connect to data sources that support the OData protocol, a standard for building and consuming RESTful APIs.

2. Salesforce Connect:

- Salesforce Connect is a tool specifically for integrating with external systems using OData or custom adapters.

Example Use Cases

1. Integrating ERP Data:

- Access and work with ERP system data directly from Salesforce without the need to duplicate the data.

2. Connecting to Legacy Databases:

- Use external objects to integrate with legacy SQL databases, allowing users to view and interact with legacy data within Salesforce.

3. Real-Time Data Access:

- Provide real-time access to data that is constantly changing, such as inventory levels or order statuses from an external system.

Benefits of Using External Objects

- **Real-Time Data Access:** Always access the most current data without the need for periodic synchronization.
- **Reduced Storage Costs:** No need to store large datasets within Salesforce.
- **Seamless Integration:** Users can interact with external data within the familiar Salesforce interface.

Limitations

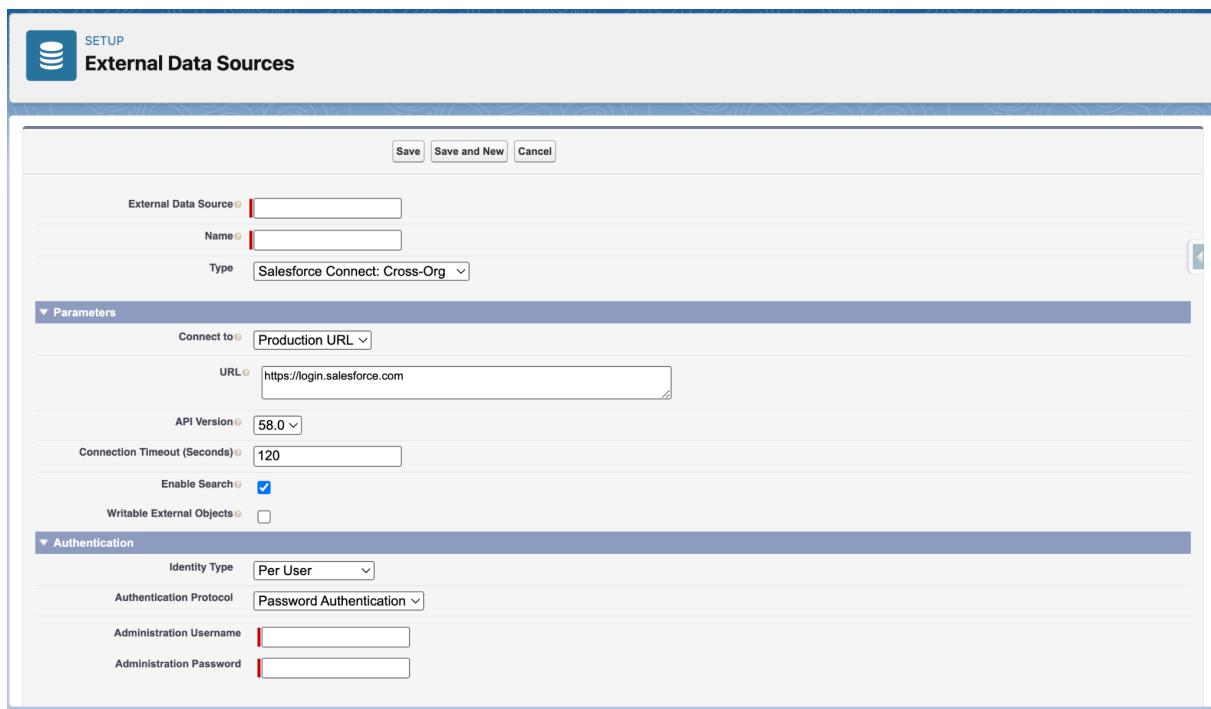
- **Performance:** Accessing data in real-time can sometimes be slower than accessing native Salesforce data.
- **Dependency on External Systems** If the external system is down or slow, it impacts data access in Salesforce.
- **Read-Only Constraints:** Some external data sources may only support read-only access.

Cross-Org Adopter in Salesforce - Salesforce Connect

Salesforce Connect is a feature that allows Salesforce users to integrate and access data stored in external systems directly within their Salesforce environment. One powerful application of Salesforce Connect is Cross-Org Adopter, which enables seamless data integration between different Salesforce organizations (orgs). This is particularly useful for companies with multiple

Salesforce instances that need to share data across those instances without duplicating it.

External Data Source Create/Edit Page



The screenshot shows the 'External Data Sources' page in Salesforce Setup. The page title is 'External Data Sources'. At the top right are three buttons: 'Save', 'Save and New', and 'Cancel'. Below the buttons are fields for 'External Data Source' (with a redacted value) and 'Name' (with a redacted value). A dropdown menu 'Type' is set to 'Salesforce Connect: Cross-Org'. The main configuration area is divided into sections: 'Parameters' and 'Authentication'. Under 'Parameters', the 'Connect to' dropdown is set to 'Production URL', the 'URL' field contains 'https://login.salesforce.com', the 'API Version' is '58.0', 'Connection Timeout (Seconds)' is '120', 'Enable Search' is checked, and 'Writable External Objects' is unchecked. Under 'Authentication', the 'Identity Type' is 'Per User', the 'Authentication Protocol' is 'Password Authentication', and the 'Administration Username' and 'Administration Password' fields both have redacted values.

External Data Source Detail Page

SETUP

External Data Sources

« Back to External Data Sources

External Data Source		PantherSchools Org		
Name	PantherSchools_Org			
Type	Salesforce Connect: Cross-Org			
Parameters				
Connect to	Production URL			
URL	https://login.salesforce.com			
API Version	60.0			
Connection Timeout (Seconds)	120			
Enable Search	✓			
Writable External Objects	✓			
Authentication				
Identity Type	Per User			
Authentication Protocol	Password Authentication			
Administration Username	sfdcpanther+hindi@gmail.com			
External Objects				
Action	Label	Namespace Prefix	Description	Table Name
Edit Erase	External Account		Account	
Edit Erase	Account Request		AccountRequest__c	
Edit Erase	Admin Request Access		AdminRequestAccess__c	

Edit Validate and Sync Delete

... 

Validate & Sync



Validate External Data Source: PantherSchools Org

Confirm that you can connect to the external system, and synchronize its schema with your Salesforce org.

[« Back to External Data Source: PantherSchools Org](#)

Name	PantherSchools_Org
External Data Source	PantherSchools Org
Status	Success

Sync

Sync in background

Select	Table Name	External Object Name	Label	Plural Label	Synced
<input checked="" type="checkbox"/>	AIApplication	AIApplication	AI Application	AI Applications	<input type="checkbox"/>
<input checked="" type="checkbox"/>	AIApplicationConfig	AIApplicationConfig	AI Application config	AI Application configs	<input type="checkbox"/>
<input checked="" type="checkbox"/>	AllInsightAction	AllInsightAction	AI Insight Action	AI Insight Actions	<input type="checkbox"/>
<input type="checkbox"/>	AllInsightFeedback	AllInsightFeedback	AI Insight Feedback	AI Insight Feedbacks	<input type="checkbox"/>
<input type="checkbox"/>	AllInsightReason	AllInsightReason	AI Insight Reason	AI Insight Reasons	<input type="checkbox"/>
<input type="checkbox"/>	AllInsightValue	AllInsightValue	AI Insight Value	AI Insight Values	<input type="checkbox"/>
<input type="checkbox"/>	AIPredictionEvent	AIPredictionEvent	AI Prediction Event	AI Prediction Events	<input type="checkbox"/>
<input type="checkbox"/>	AIRecordInsight	AIRecordInsight	AI Record Insight	AI Record Insights	<input type="checkbox"/>
<input type="checkbox"/>	AcceptedEventRelation	AcceptedEventRelation	Accepted Event Relatio	Accepted Event Relatio	<input type="checkbox"/>
<input type="checkbox"/>	Account	ExternalAccount	External Account	External Accounts	<input checked="" type="checkbox"/>
<input type="checkbox"/>	AccountBrand	AccountBrand	Account Brand	Account Brands	<input type="checkbox"/>
<input type="checkbox"/>	AccountBrandShare	AccountBrandShare	Account Brand Share	Account Brand Share	<input type="checkbox"/>

Key Features of Salesforce Connect for Cross-Org Integration

1. Real-Time Data Access:

- Access data from another Salesforce org in real-time, ensuring that users always have the most up-to-date information.

2. No Data Duplication:

- External objects allow you to work with data from other orgs without importing or storing it in your local Salesforce org, reducing storage costs and avoiding data duplication.

3. Seamless Integration:

- Users can interact with external data as if it were native to their org, using standard Salesforce functionality like reports, dashboards, and SOQL queries.

4. OData Protocol:

- Salesforce Connect uses the Open Data Protocol (OData) for communication between Salesforce orgs, ensuring a standardized and efficient data exchange.

Example Use Case

1. External Data Source in Target Org

- In Setup, navigate to "External Data Sources".
- Create a new data source with type "Salesforce Connect: OData 4.0".
- Configure the endpoint URL and authentication.

2. Validate and Sync

- Validate the external data source to ensure the connection.
- Sync the external objects (e.g., Accounts, Contacts) to make them available in the target org.

3. Using External Objects

```
// Example Apex code to query external data
List<ExternalAccount__x> externalAccounts = [SELECT Id, Name FROM
ExternalAccount__x LIMIT 10];

for (ExternalAccount__x account : externalAccounts) {
    System.debug('External Account Name: ' + account.Name);
}
```

Benefits of Cross-Org Adopter

- **Unified View:** Access and manage data from multiple Salesforce orgs in one place.
- **Cost Efficiency:** Avoid data duplication and save on storage costs.
- **Data Consistency:** Ensure real-time access to the most current data across orgs.

Limitations

- **Performance:** Real-time data access can be slower compared to accessing native Salesforce data.
- **Dependency:** Data access depends on the availability and performance of the source org.
- **Read-Only Access:** In many cases, external objects are read-only, although some write capabilities can be configured.

Tooling API

You can use the tooling API when you want to

- Manipulate the Org Metadata like activating or deactivating the apex trigger
- Find the dependency of the metadata
- Retrieve the information about custom metadata
- and Many more

List All the Objects

```
curl --location  
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/  
v60.0/tooling/sobjects/' \  
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;  
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d  
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljk02jkcZne3uQrVCMnqU1QnttYVCD03t  
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL  
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy
```

```
S89volHNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKRfxTG0xJ69104a8mG6E  
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;  
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1'
```

List All custom field

```
curl --location  
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/v46.0/tooling/query?q=SELECT%20Id%2C%20Description%2C%20Label%2CEntityDefinitionId%2C%20DurableId%20FROM%20FieldDefinition%20WHERE%20EntityDefinitionId%20%3D%20%27Case%27' \  
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;  
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d  
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljqMK02jkcZne3uQrVCMnqU1QnttYVCDo3t  
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL  
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy  
S89volHNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKRfxTG0xJ69104a8mG6E  
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;  
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1'
```

Dependency Query

```
curl --location  
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/v47.0/tooling/query?q=SELECT%20MetadataComponentId%2CMetadataComponentName%2CRefMetadataComponentName%2CRefMetadataComponentId%20FROM%20MetadataComponentDependency%20WHERE%20RefMetadataComponentId%3D%20%2700NHu00000jUA5kMAG%27' \  
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;  
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d  
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljqMK02jkcZne3uQrVCMnqU1QnttYVCDo3t  
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL  
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy'
```

```
S89volHNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKRFxTG0xJ69104a8mG6E  
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;  
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1'
```

Deactivate the Apex Trigger using tooling API

Step1 - Create Metadata Container

To deactivate the Apex Trigger from production, we first need to start with creating the Metadata Container.

```
curl --location  
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/  
v60.0/tooling/sobjects/MetadataContainer' \  
--header 'Content-Type: application/json' \  
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;  
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d  
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljkMQK02jkcZne3uQrVCMnqU1QnttYVCD03t  
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL  
6yAOYd4xv73iXcA4DvfsR1Wjz/nsFWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy  
S89volHNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKRFxTG0xJ69104a8mG6E  
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;  
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1' \  
--data '{  
    "Name": "OrderTrigger"  
}'
```

Step2 - Create Apex Trigger Member Object Record

To create the Apex Trigger Member record, it needs the metadata container id from the previous step and the Apex Trigger Id

```
curl --location  
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/  
v60.0/tooling/sobjects/ApexTriggerMember' \  
--header 'Content-Type: application/json'
```

```
--header 'Content-Type: application/json' \
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAAQAA7U1wAwrQ43d
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljk02jkcZne3uQrVCMnqU1QnttYVCDo3t
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy
S89volHNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKRfxTG0xJ69104a8mG6E
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1' \
--data-raw '{
    "MetadataContainerId": "1dcHu000001zJH6IAM",
    "ContentEntityId": "01qHu000001QIijIAG",
    "Body": "/** * @description      : * @author          : Amit
Singh - PantherSchools * @group           : * @last modified on :
05-05-2024 * @last modified by : Amit Singh - PantherSchools**/trigger
OrderTrigger on Order (after insert) {
    OrderTriggerHandler.publishEvent(Trigger.New);}",
    "Metadata": {
        "status": "Inactive",
        "apiVersion": 60,
        "packageVersions": null,
        "urls": null
    }
}'
```

Step3 - Create Container Async Request

This is the final step which will start the deployment of the Apex trigger with Inactive status. This will require the Container Id that you created in Step 1.

```
curl --location
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/v60.0/tooling/sobjects/containerAsyncRequest/' \
--header 'Content-Type: application/json' \
```

```
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljqMK02jkcZne3uQrVCMnqU1QnttYVCDo3t
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy
S89vo1HnbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKrfxTG0xJ69104a8mG6E
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1' \
--data '{
    "MetadataContainerId": "1dcHu000001zJH6IAM",
    "isCheckOnly": "false"
}'
```

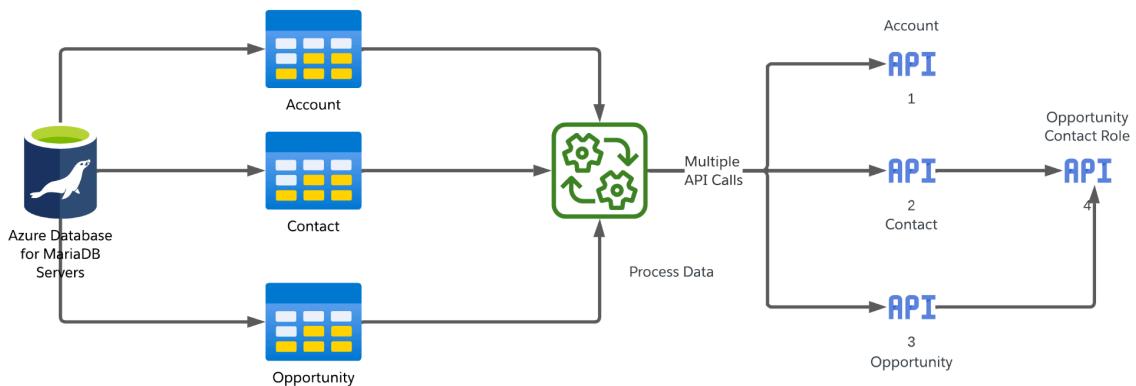
Step4 - Check the status of Container Async Request (Optional)

At the end, you can check the status of the Container Async request object that you created in the previous steps.

This will tell you the status of the deployment request created in step3.

```
curl --location
'https://integration-org2-dev-ed.develop.my.salesforce.com/services/data/v60.0/tooling/sobjects/containerAsyncRequest/1drHu000002WaBrIAK' \
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljqMK02jkcZne3uQrVCMnqU1QnttYVCDo3t
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy
S89vo1HnbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKrfxTG0xJ69104a8mG6E
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1'
```

Problem Before Composite API



Introduction Composite API

→ Introduction

Composite API is a very powerful concept in Salesforce that gives you the ability to execute a series of REST API requests in a single POST request or retrieve a list of other composite resources with a GET request.

Composite Request Parameters

- **allOrNone** - If the value for this flag is false, then the independent request will be processed even if one of the composite requests fails.
- **collateSubrequests** - Controls whether the API collates unrelated subrequests to bulky them (true) or not (false). If the value is set to true then the order of execution is not guaranteed
- **compositeRequest** - Accepts the array of the **Composite Request object**

Composite Request Object

- **body** - The input body for the subrequest.
 - { "Name": "Salesforce.com" }
- **httpHeaders** - Request headers and their values to include with the subrequest
- **method** - The HTTP method that you wanted to execute. GET, POST, PATCH, DELETE

- **referenceId**- A Unique Reference ID that can be used in the subsequent request to access the fields. **For example**, if the reference ID is **refAccount** then to access the ID you can use **@{refAccount.id}**
- **URL** - The resource that you wanted to execute, **For Example** - /services/data/v60.0/sobjects/Account

Limits

- **AllorNone** - If the value for this flag is set to **false** then the independent request will be processed even if one of the composite requests fails.
- At a time only one composite request can be sent and can send max 25 request
- The complete composite request can contain **Max 5 SOQL API calls**
- The depth limit of the Composite API call is 15

Structure of the Simple Request

Endpoint URL - </services/data/v60.0/composite>

Content-Type - application/json

Method - POST

Authentication - Bearer Token

Body - Below is the sample body

The below composite request body creates the Account Record and the related Contact Record.

```
{
  "compositeRequest": [
    {
      "method": "POST",
      "url": "/sobjects/Account"
    },
    {
      "method": "POST",
      "url": "/sobjects/Contact"
    }
  ]
}
```

```
"method": "POST",
"url": "/services/data/v60.0/sobjects/Account",
"referenceId": "refAccount_1",
"body": {
    "Name": "Sample Account",
    "Rating": "Hot",
    "Phone": "8997834534",
    "Description": "Sample Account",
    "Industry": "Education"
},
{
    "method": "POST",
    "url": "/services/data/v60.0/sobjects/Contact",
    "referenceId": "refContact_1",
    "body": {
        "LastName": "Sample Contact",
        "FirstName": "Sample Contact",
        "Email": "Sample Contact",
        "Phone": "Sample Contact",
        "Title": "Sample Contact",
        "AccountId": "@{refAccount_1.id}"
    }
},
{
    "method": "POST",
    "url": "/services/data/v60.0/sobjects/Opportunity",
    "referenceId": "refContact_1",
    "body": {
        "Name": "Sample Contact",
        "Amount": "",
        "CloseDate": "",
        "StageName": "",
        "Description": "",
        "AccountId": "@{refAccount_1.id}"
}
```

```
        }
    }
]
}
```

Here is the response from the Postman

The screenshot shows the Postman interface with a composite API request. The request URL is `POST {{instance_url}}/services/data/v57.0/composite`. The body is a JSON object containing two composite requests. The first request creates an account with reference ID `refAccount` and body `{"Name": "Sample Account"}`. The second request creates a contact with reference ID `refContact` and body `{"LastName": "Sample Contact", "AccountId": "@{refAccount.id}"}`. The response status is 200 OK, and the response body is a JSON object with two entries, each representing one of the composite requests.

AllOrNone Demo

```
{
  "allOrNone": true,
  "compositeRequest": [
    {
      "method": "POST",
      "url": "/services/data/v57.0/sobjects/Account",
      "referenceId": "refAccount",
      "body": {
        "Industry": "Education",
        "Phone": "9088973433"
      }
    }
  ]
}
```

```

        }
    },
    {
        "method": "POST",
        "url": "/services/data/v57.0/sobjects/Contact",
        "referenceId": "refContact",
        "body": {
            "LastName": "Sample Contact",
            "AccountId": "@{refAccount.id}"
        }
    }
]
}

```

The screenshot shows the Postman interface with a composite request setup:

- Method:** POST
- URL:** `((instance_url))/services/data/v57.0/composite`
- Body:** JSON (selected)
- Request Body (Raw JSON):**

```

1  {
2     "allOrNone": true,
3     "compositeRequest": [
4         {
5             "method": "POST",
6             "url": "/services/data/v57.0/sobjects/Account",
7             "referenceId": "refAccount",
8             "body": {
9                 "Industry": "Education",
10                "Phone": "9088973433"
11             }
12         },
13         {
14             "method": "POST",
15             "url": "/services/data/v57.0/sobjects/Contact",
16             "referenceId": "refContact",
17             "body": {

```
- Response Headers:** Status: 200 OK, Time: 423 ms, Size: 849 B
- Response Body (Pretty JSON):**

```

1  {
2     "compositeResponse": [
3         {
4             "body": [
5                 {
6                     "message": "Required fields are missing: [Name]",
7                     "errorCode": "REQUIRED_FIELD_MISSING",
8                     "fields": [
9                         "Name"
10                     ]
11                 }
12             ],
13             "httpHeaders": {},
14             "httpStatusCode": 400,
15             "referenceId": "refAccount"
16         },
17         {
18             "body": [
19                 {
20                     "errorCode": "PROCESSING_HALTED",
21                     "message": "The transaction was rolled back since another operation in the same transaction failed."
22                 }
23             ],
24         }
25     ]
26 }

```

Multiple API Calls Demo

Requirement

The business has legacy data in its Local Database and it wanted to migrate the

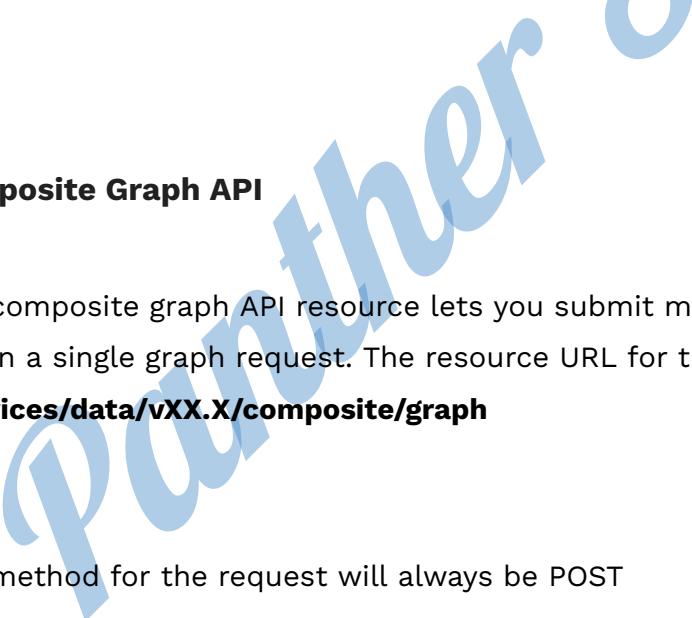
data to Salesforce. The data is in such a way that they need to create the Account, a related contact, and an opportunity and Add the contact as an Opportunity Contact Role.

They have in-house developers who will be sending the JSON file to Salesforce for creating the records.

Sample request body

```
{  
    "allOrNone": true,  
    "compositeRequest": [  
        {  
            "method": "POST",  
            "url": "/services/data/v57.0/sobjects/Account",  
            "referenceId": "refAccount",  
            "body": {  
                "Industry": "Education",  
                "Phone": "9088973433",  
                "Name": "Salesforce.com India Pvt. Ltd."  
            }  
        },  
        {  
            "method": "POST",  
            "url": "/services/data/v57.0/sobjects/Contact",  
            "referenceId": "refContact",  
            "body": {  
                "LastName": "Contact",  
                "FirstName": "Sample",  
                "Email": "abc@gmail.com",  
                "Phone": "9088973433",  
                "Fax": "9088973433",  
                "AccountId": "@{refAccount.id}"  
            }  
        }  
    ]  
}
```

```
        }
    },
    {
        "method": "POST",
        "url": "/services/data/v57.0/sobjects/Opportunity",
        "referenceId": "refOpportunity",
        "body": {
            "Name": "Salesforce.com Hyderabaad",
            "StageName": "Prospecting",
            "CloseDate": "2023-07-07",
            "AccountId": "@{refAccount.id}"
        }
    },
    {
        "method": "POST",
        "url": "/services/data/v57.0/sobjects/OpportunityContactRole",
        "referenceId": "refOpportunityContactRole",
        "body": {
            "ContactId": "@{refContact.id}",
            "OpportunityId": "@{refOpportunity.id}",
            "IsPrimary": true,
            "Role": "Business User"
        }
    }
]
```



The screenshot shows a POST request in the cURL interface of Postman. The URL is `/{{instance_url}}/services/data/v57.0/composite`. The request body is a JSON object containing three composite requests:

```

1 {
2   "allOrNone" : true,
3   "compositeRequest": [
4     {
5       "method": "POST",
6       "url": "/services/data/v57.0/sobjects/Account",
7       "referenceId": "refAccount",
8       "body": {
9         "Industry": "Education",
10        "Phone" : "9888973433",
11        "Name" : "Salesforce.com India Pvt. Ltd."
12      }
13    },
14    {
15      "method": "POST",
16      "url": "/services/data/v57.0/sobjects/Contact",
17      "referenceId": "refContact",
18      "body": {
19        "LastName": "Contact",
20        "FirstName": "Sample",
21        "Email" : "abc@gmail.com",
22        "Phone" : "9888973433",
23        "Fax" : "9888973433",
24        "AccountId": "@{refAccount.id}"
25      }
26    },
27    {
28      "method": "POST",
29      "url": "/services/data/v57.0/sobjects/Opportunity",
30      "referenceId": "refOpportunity",
31      "body": {
32        "Name": "Salesforce.com Hyderabad",
33        "StageName": "Prospecting",
34        "CloseDate": "2023-07-07"
35      }
36    }
37  ]
38 }

```

The response status is 200 OK, time 1189 ms, size 1.26 KB. The response body is:

```

1 {
2   "compositeResponse": [
3     {
4       "body": {
5         "id": "0012w00001wsqJmAAJ",
6         "success": true,
7         "errors": []
8       },
9       "httpHeaders": {
10         "Location": "/services/data/v57.0/sobjects/Account/0012w00001wsqJmAAJ"
11       }
12     }
13   ]
14 }

```

Composite Graph API

The composite graph API resource lets you submit multiple composite requests within a single graph request. The resource URL for the composite graph API is - **/services/data/vXX.X/composite/graph**

The method for the request will always be POST

The format for the request is JSON

Below is the sample API request body for the POST method

```
{  
    "graphId" : "graphId",  
    "compositeRequest" : [  
        compositeSubrequest,  
        compositeSubrequest,  
        ...  
    ]  
}
```

Response format

```
{  
    "graphs" : [  
        {  
            "graphId" : "graphId",  
            "graphResponse" : {  
                "compositeResponse" : [  
                    compositeSubrequestResult  
                ]  
            },  
            "isSuccessful" :True  
        },  
    ]  
}
```

You can have multiple graph requests in a Single API Call. The maximum number of graph requests that can be sent is 15 and within one graph request, we can have 500 Requests.

The depth limit of the Graph request is 15.

Limits of Graph API

→ **AllorNone** - There is no allOrNone concept for Composite Graph Request

- At a time We can send 15 Graph request
- One graph request can have up to 500 Composite requests.
- The depth limit of the Composite API call is 15
- If one graph fails then all the requests will fail within that Graph Only.

Structure of the Simple Request

Endpoint URL - </services/data/v57.0/composite/graph>

Content-Type - application/json

Method - POST

Authentication - Bearer Token

Body - Below is the sample body

```
{  
    "graphs": [  
        {  
            "graphId": "1",  
            "compositeRequest": [  
                {  
                    "method": "GET",  
                    "url": "/services/data/v60.0/query/?q=SELECT Id, Name FROM LEAD LIMIT 10",  
                    "referenceId": "refGetLead_1"  
                },  
                {  
                    "method": "POST",  
                    "url": "/services/data/v60.0/sobjects/Account",  
                    "referenceId": "refAccount_1",  
                    "body": "

Sample Body Content

"  
                }  
            ]  
        }  
    ]  
}
```

```
"body": {
    "Name": "{$randomCompanyName}",
    "Rating": "Hot",
    "Phone": "{$randomPhoneNumber}",
    "Description": "{$randomJobDescriptor}",
    "Industry": "Education"
},
{
    "method": "POST",
    "url": "/services/data/v60.0/sobjects/Contact",
    "referenceId": "refContact_1",
    "body": {
        "LastName": "{$randomLastName}",
        "FirstName": "{$randomFirstName}",
        "Email": "{$randomEmail}",
        "Phone": "{$randomPhoneNumber}",
        "Title": "{$randomJobTitle}",
        "AccountId": "@{refAccount_1.id}"
    }
},
{
    "method": "POST",
    "url": "/services/data/v60.0/sobjects/Opportunity",
    "referenceId": "refOpportunity_1",
    "body": {
        "Name": "{$randomCompanyName}",
        "Amount": 34534.35,
        "CloseDate": "2024-10-10",
        "StageName": "Prospecting",
        "Description": "{$randomJobDescriptor}",
        "AccountId": "@{refAccount_1.id}"
    }
},
{
```

```
        "method": "POST",
        "url": "/services/data/v60.0/sobjects/OpportunityContactRole",
        "referenceId": "refOpportunityContactRole_1",
        "body": {
            "OpportunityId": "@{refOpportunity_1.id}",
            "ContactId": "@{refContact_1.id}",
            "Role": "Decision Maker",
            "IsPrimary": true
        }
    }
],
},
{
    "graphId": "2",
    "compositeRequest": [
        {
            "method": "GET",
            "url": "/services/data/v60.0/query/?q=SELECT Id, Name FROM LEAD LIMIT 10",
            "referenceId": "refGetLead_1"
        },
        {
            "method": "POST",
            "url": "/services/data/v60.0/sobjects/Account",
            "referenceId": "refAccount_1",
            "body": {
                "Name": "{$randomCompanyName}",
                "Rating": "Hot",
                "Phone": "{$randomPhoneNumber}",
                "Description": "{$randomJobDescriptor}",
                "Industry": "Education"
            }
        },
        {
            "method": "POST",

```

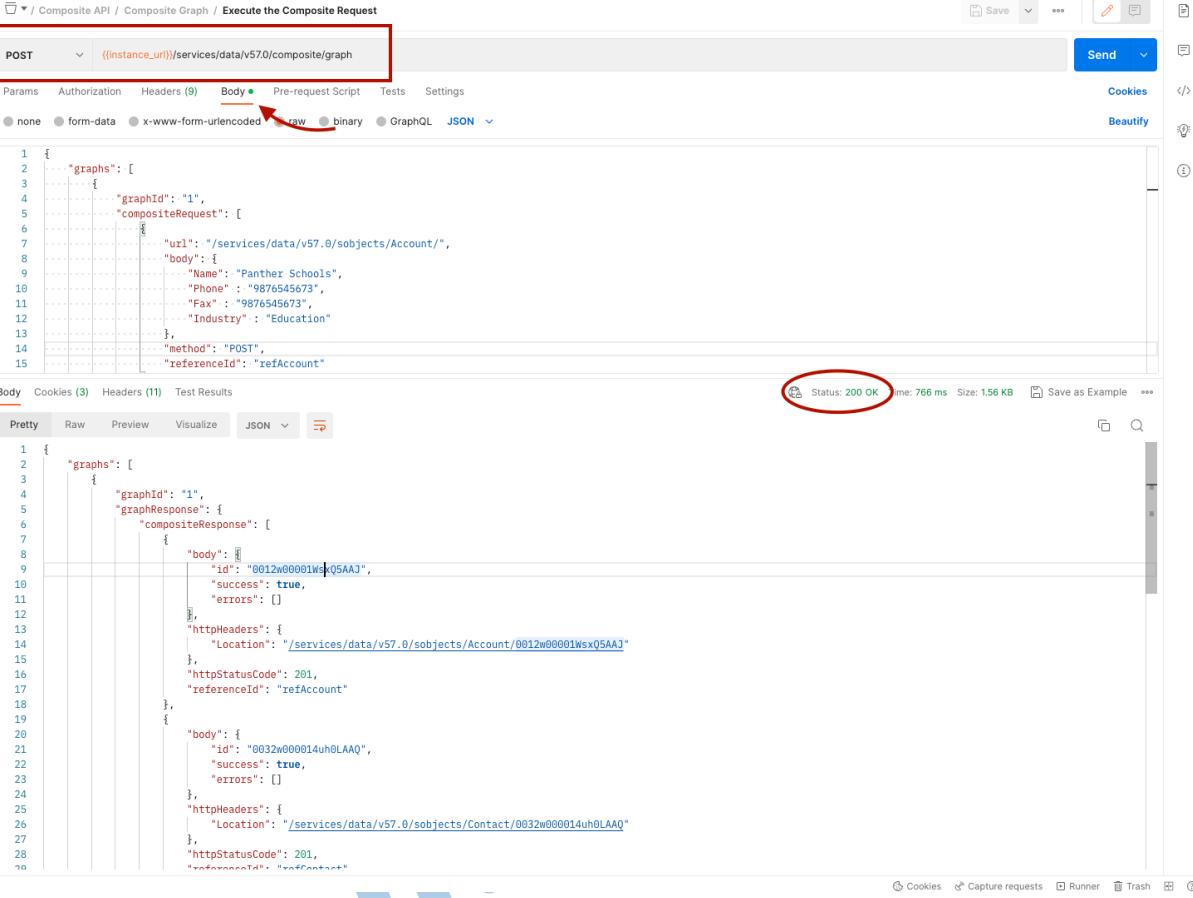
```
"url": "/services/data/v60.0/sobjects/Contact",
"referenceId": "refContact_1",
"body": {
    "LastName": "{$randomLastName}",
    "FirstName": "{$randomFirstName}",
    "Email": "{$randomEmail}",
    "Phone": "{$randomPhoneNumber}",
    "Title": "{$randomJobTitle}",
    "AccountId": "@{refAccount_1.id}"
}
},
{
    "method": "POST",
    "url": "/services/data/v60.0/sobjects/Opportunity",
    "referenceId": "refOpportunity_1",
    "body": {
        "Name": "{$randomCompanyName}",
        "Amount": 34534.35,
        "CloseDate": "2024-10-10",
        "StageName": "Prospecting",
        "Description": "{$randomJobDescriptor}",
        "AccountId": "@{refAccount_1.id}"
}
},
{
    "method": "POST",
    "url": "/services/data/v60.0/sobjects/OpportunityContactRole",
    "referenceId": "refOpportunityContactRole_1",
    "body": {
        "OpportunityId": "@{refOpportunity_1.id}",
        "ContactId": "@{refContact_1.id}",
        "Role": "Decision Maker",
        "IsPrimary": true
}
}
```

]

}

]

}



Here is the sample request

```
{
  "graphs": [
    {
      "graphId": "1",
      "graphResponse": {
        "compositeResponse": [
          {
            "body": {
              "id": "0012w00001WsxQ5AAJ",
              "success": true,
              "errors": []
            },
            "httpHeaders": {
              "Location": "/services/data/v57.0/sobjects/Account/0012w00001WsxQ5AAJ"
            },
            "httpStatusCode": 201,
            "referenceId": "refAccount"
          },
          {
            "body": {
              "id": "0032w000014uh0LAAQ",
              "success": true,
              "errors": []
            },
            "httpHeaders": {
              "Location": "/services/data/v57.0/sobjects/Contact/0032w000014uh0LAAQ"
            },
            "httpStatusCode": 201,
            "referenceId": "refContact"
          }
        ]
      }
    }
  ]
}
```

```
        "errors": []
    },
    "httpHeaders": {
        "Location":
"/services/data/v57.0/sobjects/Account/0012w00001WsxQ5AAJ"
    },
    "httpStatusCode": 201,
    "referenceId": "refAccount"
},
{
    "body": {
        "id": "0032w000014uh0LAAQ",
        "success": true,
        "errors": []
    },
    "httpHeaders": {
        "Location":
"/services/data/v57.0/sobjects/Contact/0032w000014uh0LAAQ"
    },
    "httpStatusCode": 201,
    "referenceId": "refContact"
},
{
    "body": {
        "id": "0062w00000LDrLsAAL",
        "success": true,
        "errors": []
    },
    "httpHeaders": {
        "Location":
"/services/data/v57.0/sobjects/Opportunity/0062w00000LDrLsAAL"
    },
    "httpStatusCode": 201,
    "referenceId": "refOpportunity"
}
```

```
        ],
    },
    "isSuccessful": true
},
{
    "graphId": "2",
    "graphResponse": {
        "compositeResponse": [
            {
                "body": {
                    "id": "0012w00001WsxQ6AAJ",
                    "success": true,
                    "errors": []
                },
                "httpHeaders": {
                    "Location":
                    "/services/data/v57.0/sobjects/Account/0012w00001WsxQ6AAJ"
                },
                "httpStatusCode": 201,
                "referenceId": "refAccount2"
            },
            {
                "body": {
                    "id": "0032w000014uh0MAAQ",
                    "success": true,
                    "errors": []
                },
                "httpHeaders": {
                    "Location":
                    "/services/data/v57.0/sobjects/Contact/0032w000014uh0MAAQ"
                },
                "httpStatusCode": 201,
                "referenceId": "refContact"
            }
        ]
    }
}
```

```

        },
        "isSuccessful": true
    }
]
}

```

The screenshot shows the Salesforce interface for an account named 'Panther Schools'. The top navigation bar includes Sales, Home, Opportunities, Leads, Tasks, Files, Accounts (selected), Contacts, Campaigns, Dashboards, Reports, Chatter, Groups, Calendar, and More. The account details section shows Type: Account, Phone: 9876545673, Website, Account Owner: Admin User, Account Site, and Industry: Education. Below this, the 'Related' tab is selected, displaying 'Contacts (1)' and 'Opportunities (1)'. The 'Contacts' section lists Jason Frank with fields for Title, Email, and Phone. The 'Opportunities' section lists an opportunity for Amazon.com with Stage: Prospecting, Amount, and Close Date: 22/5/2024. On the right side, there is an 'Activity' section showing an upcoming task: 'Created From Apex Trigg...' on 09-May, and a 'Chatter' feed with a message from Admin User.

sObject Tree

sObject tree is mainly used for creating multiple records from a single request. sObject Tree takes the parent object details and their child object details in the same request.

The sObject Tree can be used to create the record of the same type or its related objects.

Limits

- A total of 200 records across all trees
- Up to five records of different types
- sObject trees up to five levels deep

Below is the sample request

Url - /services/data/v57.0/composite/tree/<sObjectName>

Method - POST

Format - JSON

Body -

```
{  
    "records": [  
        {  
            "attributes": {  
                "type": "Account",  
                "referenceId": "refAccount"  
            },  
            "Name": "Panther Schools",  
            "Phone": "1234567890",  
            "website": "https://www.pantherschools.com",  
            "numberOfEmployees": "100",  
            "Industry": "Education",  
            "Contacts": {  
                "records": [  
                    {  
                        "attributes": {  
                            "type": "Contact",  
                            "referenceId": "refContact"  
                        },  
                        "FirstName": "Amit",  
                        "lastname": "Singh",  
                        "title": "Founder",  
                        "email": "sfdcpanther@gmail.com"  
                    },  
                    {  
                        "attributes": {  
                            "type": "Contact",  
                            "referenceId": "refContact2"  
                        },  
                        "FirstName": "Sonal",  
                        "lastname": "Singh",  
                        "title": "Co-Founder",  
                        "email": "sonal.singh@gmail.com"  
                    }  
                ]  
            }  
        }  
    ]  
}
```

```
        "referenceId": "refContact2"
    },
    "FirstName": "Amit",
    "lastname": "Singh",
    "title": "Vice President",
    "email": "sfdcpanther@gmail.com"
}
]
}
},
{
"attributes": {
    "type": "Account",
    "referenceId": "ref4"
},
"name": "SFDCPanther",
"phone": "1234567890",
"website": "https://www.pantherschools.com",
"numberOfEmployees": "100",
"industry": "Education",
"Contacts": {
    "records": [
        {
            "attributes": {
                "type": "Contact",
                "referenceId": "refContact4"
            },
            "FirstName": "Amit",
            "lastname": "Singh",
            "title": "Founder",
            "email": "sfdcpanther@gmail.com"
        },
        {
            "attributes": {
                "type": "Contact",

```

```

        "referenceId": "refContact3"
    },
    "FirstName": "Amit",
    "lastname": "Singh",
    "title": "Vice President",
    "email": "sfdcpanther@gmail.com"
}
]
}
}
]
}

```

POST [\(\(instance_url\)\)/services/data/v57.0/composite/tree/Account](((instance_url))/services/data/v57.0/composite/tree/Account)

Body

```

33     "email": "sfdcpanther@gmail.com"
34   ]
35   }
36   }
37   {
38     "attributes": [
39       {
40         "type": "Account",
41         "referenceId": "ref4"
42       }
43     ],
44     "name": "SFCPanther",
45     "phone": "1234567890",
46     "website": "https://www.pantherschools.com",
47     "numberOfEmployees": "100",
48     "industry": "Education",
49     "Contacts": [
50       {
51         "records": [
52           {
53             "attributes": {
54               ...
55             }
56           }
57         ]
58       }
59     ]
60   }
61 }
```

Status: 201 Created Time: 659 ms Size: 798 B Save as Example

Pretty Raw Preview Visualize JSON

```

1   {
2     "hasErrors": false,
3     "results": [
4       {
5         "referenceId": "refAccount",
6         "id": "0012w00001XR1C6AAL"
7       },
8       {
9         "referenceId": "ref4",
10        "id": "0012w00001XR1C7AAL"
11      },
12      {
13        "referenceId": "refContact",
14        "id": "0032w000014uhTbAAI"
15      },
16      {
17        "referenceId": "refContact2",
18        "id": "0032w000014uhTcAAI"
19      },
20      {
21        "referenceId": "refContact4",
22        "id": "0032w000014uhTdAAI"
23      },
24    ]
25  }

```

Apex SOAP API

SOAP (Simple Object Access Protocol) services are important for Salesforce Apex

for several reasons:

1. Interoperability

- **Standardized Protocol:** SOAP is a standardized protocol that is widely used and supported across different platforms and languages. This allows Salesforce to communicate with various external systems, regardless of their underlying technology.
- **WS-Standards Compliance:*** SOAP supports a range of WS-* standards (such as WS-Security, WS-ReliableMessaging), making it suitable for complex and secure enterprise integrations.

2. Strong Typing and Contract Definition

- **WSDL (Web Services Description Language):** SOAP services in Salesforce use WSDL, which provides a formal contract between the service provider and the consumer. This contract includes details about the service methods, request and response structures, and data types.
- **Type Safety:** The strong typing ensures that the data being exchanged adheres to the defined contract, reducing the likelihood of errors due to data inconsistencies.

3. Enterprise-Level Integration

- **Complex Operations:** SOAP is well-suited for complex, stateful operations that require a higher level of reliability and security, often needed in enterprise environments.
- **Transactional Support:** SOAP can handle transactions across multiple operations, which is crucial for enterprise applications that require atomicity, consistency, isolation, and durability (ACID) properties.

4. Security

- **WS-Security:** SOAP supports WS-Security, which provides a robust security framework for message integrity, confidentiality, and authentication.
- **HTTPS:** SOAP messages can be encrypted and sent over HTTPS, ensuring secure communication between Salesforce and external systems.

5. Legacy System Integration

- **Compatibility with Legacy Systems:** Many legacy systems and enterprise applications still use SOAP for web services. Salesforce's ability to consume and expose SOAP services facilitates integration with these systems.
- **Mainframe and Middleware Integration:** SOAP is commonly used for integrating with mainframes, middleware, and other enterprise-grade solutions that may not support REST.

6. Salesforce-Specific Use Cases

- **Apex Callouts:** Salesforce allows Apex code to make callouts to external SOAP services, enabling real-time data integration and retrieval.
- **Web Service Classes:** Apex can define web service classes that external systems can call, facilitating bidirectional data exchange and process integration.

7. Tooling and Support

- **Development Tools:** Salesforce provides tools and support for developing, testing, and deploying SOAP services, including the Apex Web Service classes and the ability to generate and consume WSDLs.
- **Documentation and Community Support:** There is extensive documentation and community support for implementing SOAP services in Salesforce, making it easier for developers to find solutions and best practices.

Example Scenario

Imagine a scenario where a Salesforce instance needs to integrate with an external financial system that provides SOAP-based web services for processing transactions. Here's how SOAP services would be beneficial:

- **WSDL Contract:** The financial system provides a WSDL file, allowing Salesforce developers to understand the exact structure of requests and responses.
- **Security:** WS-Security ensures that sensitive financial data is securely transmitted.
- **Complex Operations:** The financial system's SOAP services handle complex

operations, such as multi-step transaction processing, which requires a reliable and stateful protocol.

Code Used in Apex Class

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 06-15-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
global with sharing class AccountPlanner {  
  
    webservice static String helloWorld(String name){  
        return 'Hello ' + name + '!';  
    }  
  
    webservice static String createAccount(String name, String Industry){  
        Account acc = new Account();  
        acc.Name = name;  
        acc.Industry = Industry;  
        insert acc;  
        return acc.Id;  
    }  
  
    webservice static InputWrapper createContact(InputWrapper input){  
        /**  
         * Create the Contact Record and return the response  
         */  
        return input;  
    }  
}
```

```

webservice static Lead createLead(Lead leadRecord){
    /**
     * Create the Contact Record and return the response
     */
    return leadRecord;
}

global class InputWrapper{
    webservice String firstName;
    webservice String lastName;
    webservice String email;
    webservice String accountId;
}

}

```

cURL Request - Create Lead

```

curl --location
'https://course-pantherschools-dev-ed.develop.my.salesforce.com/services/
Soap/class/AccountPlanner' \
--header 'SOAPAction: helloWorld' \
--header 'Content-Type: text/xml' \
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d
QPucsUxVEMP6W8q1mMs4MCJ9sbYgb1u/u4aAa8ljk02jkcZne3uQrVCMnqU1QnttYVCD03t
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy
S89vo1HNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKrfxTG0xJ69104a8mG6E
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1' \

```

```
--data '<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ps="http://soap.sforce.com/schemas/class/AccountPlanner">
    <soapenv:Header>
        <ps:SessionHeader>
            <ps:sessionId>000MHycmemJ</ps:sessionId>
        </ps:SessionHeader>
    </soapenv:Header>
    <soapenv:Body>
        <ps:createLead> <!-- Method Call -->
            <ps:leadRecord> <!-- Pass method parameters value -->
                <ps:FirstName>Salesforce.com</ps:FirstName>
                <ps:LastName>Education</ps:LastName>
                <ps:Company>Education</ps:Company>
                <ps:Email>Education</ps:Email>
            </ps:leadRecord>
        </ps:createLead>
    </soapenv:Body>
</soapenv:Envelope>'
```

cURL Request for Create Account

```
curl --location
'https://course-pantherschools-dev-ed.develop.my.salesforce.com/services/
Soap/class/AccountPlanner' \
--header 'SOAPAction: createAccount' \
--header 'Content-Type: text/xml' \
--header 'Cookie: BrowserId=J42C6gnEEe-Uf6W-nXzufA;
_abck=58018F777074ADE9F8D27086141D9918~-1~YAAQpW4/F8xETPyJAQAA7U1wAwrQ43d
QPucsUxVEMP6W8q1mMs4MCJ9sbYgbIu/u4aAa8ljkczne3uQrVCMnqU1QnttYVCDo3t
gmnXV5mX30kaQpgvCTnIzaTP1AH03YdEcIG9A5Ed7SDLQc8pmznNfUmsMuAnSkj1mez0Eu3sL
6yAOYd4xv73iXcA4DvfsR1Wjz/nsfWP/6s/hcitDggwqodjt3FhwIOD8iFB0hwrhBt60gdeoy'
```

```

S89volHNbWJwc0XNkFp9L59Go/Qv8/0Jq3ztq3JS5HVviVzfQum1tKRfxTG0xJ69104a8mG6E
wBW6QVJuCo4ZChPqR2Ujiho1lQy9w0xr5MUBwM/1BgtRzAAX5gJz+0DCF~-1~-1~-1;
CookieConsentPolicy=0:1; LSKey-c$CookieConsentPolicy=0:1' \
--data '<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ps="http://soap.sforce.com/schemas/class/AccountPlanner">
<soapenv:Header>
<ps:SessionHeader>
<ps:sessionId>00R0MHycmemJ</ps:sessionId>
</ps:SessionHeader>
</soapenv:Header>
<soapenv:Body>
<ps:createAccount> <!-- Method Call -->
<ps:name>Salesforce.com</ps:name> <!-- Pass method parameters
value -->
<ps:Industry>Education</ps:Industry> <!-- Pass method
parameters value -->
</ps:createAccount>
</soapenv:Body>
</soapenv:Envelope>

```

Code Used in apex class

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-16-2024
 * @last modified by : Amit Singh - PantherSchools
*/
global with sharing class AccountPlanner {

    webservice static String helloWorld(String name){

```

```

        return 'Hello ' + name + '!';
    }

webservice static InputWrapper createContact(InputWrapper input){
    /**
     * Create the Contact Record and return the response
     */
    return input;
}

global class InputWrapper{
    webservice String firstName;
    webservice String lastName;
    webservice String email;
    webservice String accountId;
}
}

```

Introduction to MuleSoft

MuleSoft is an Integration and API Platform which helps us to :

- Build and consume Web-services (RESTful & Soap)
- Helps to connect different systems (Integration)
- Perform Scheduler / Batch Jobs
- Ability to deploy applications on- Cloudbus/ Onprem

API Lead Connectivity

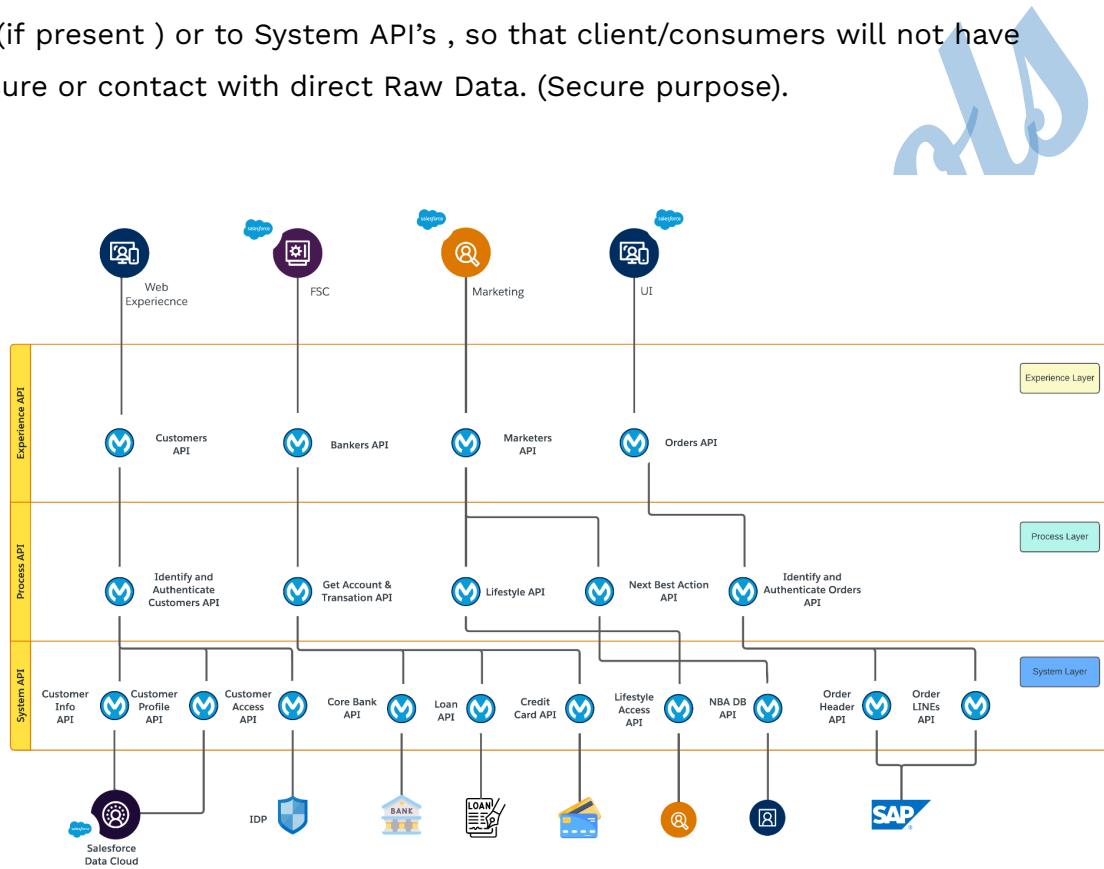
API -Led Connectivity Approach: API Led connectivity approach is widely successful approach in building API's . The concept of **System - Process - Experience API's** has been achieved to build reusable assets and highly secured.

In a simple way →

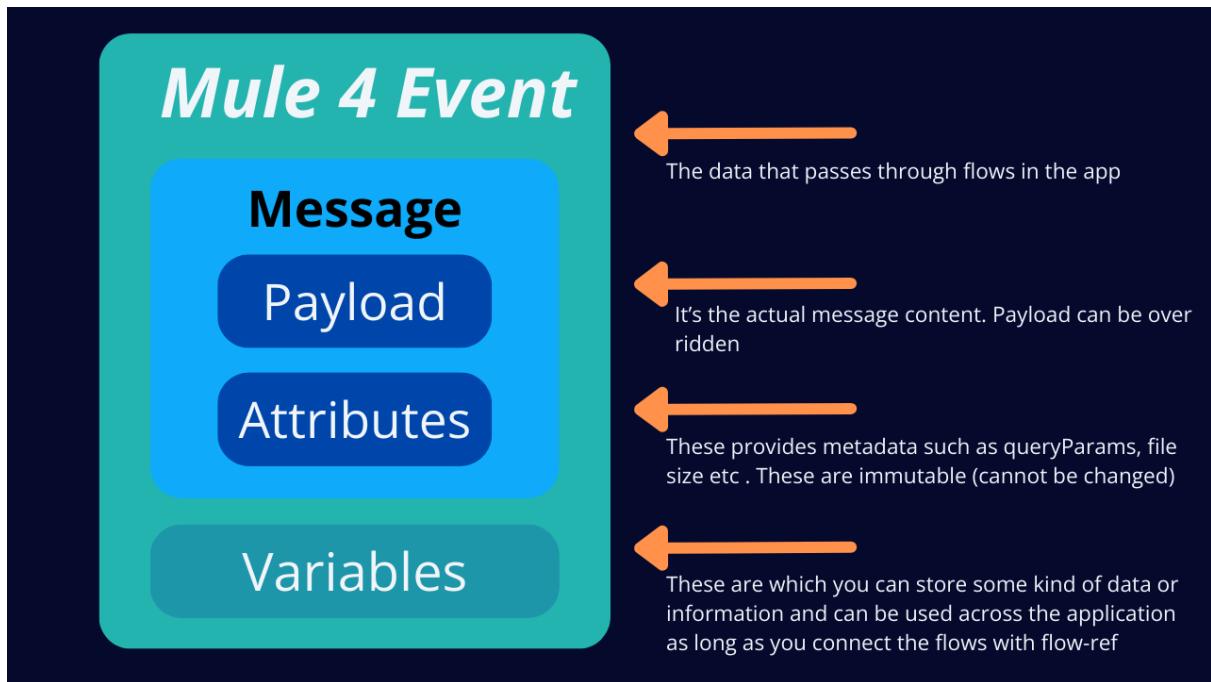
System API's : Are API's built to connect through underlying systems (Any system which contains the raw data) .

Process API's: Process API's are built to compose or combine two or more System API's to do any kind of transformation or to implement any kind of business logics.

Experience API's : These are API's which are a kind of a wrapper either to Process API's (if present) or to System API's , so that client/consumers will not have exposure or contact with direct Raw Data. (Secure purpose).



Mule Event



Signup Link - <https://anypoint.mulesoft.com/login/signup?apintent=generic>

Download Anypoint Studio - <https://www.mulesoft.com/lp/dl/anypoint-mule-studio>



Products Solutions Services Resources

Developers Partners

Contact Us
1-800-596-4880



Login

Free trial

Code Builder, Studio, or Mule

New

Anypoint Code Builder and Mule 4

- Design APIs across OAS and RAML specifications
- Test and validate API functionality with built-in mocking service
- Automatically implement API logic from existing specification
- Build integrations to connect any app, system, or data together
- Jumpstart integration development with the power of generative AI

Anypoint Studio 7 and Mule 4

Mule 4 standalone

Previous versions

Get started for free today

Anypoint Studio and Mule

Latest

Mac OS

First Name

Last Name

Email*

*Please, use a valid email. We'll send the download link to the provided address.

Company

Job Title

Phone Number

Industry

Employees

Country/Region

By registering, you agree to MuleSoft's [License Agreement](#) and the processing of your personal data by Salesforce as described in the [Privacy Statement](#).

Download

Panther

Integration Design Patterns

When you are working with Salesforce and using multiple systems it is quite obvious that you will be integrating multiple systems with Salesforce.

Integration patterns help you to define the strategy and best practices to develop the integration in the best way so that the result (solution) poses the below qualities

- Simple
- Scalable
- Maintainable
- Good Performance

Salesforce has defined the 6 integration patterns that are listed below →

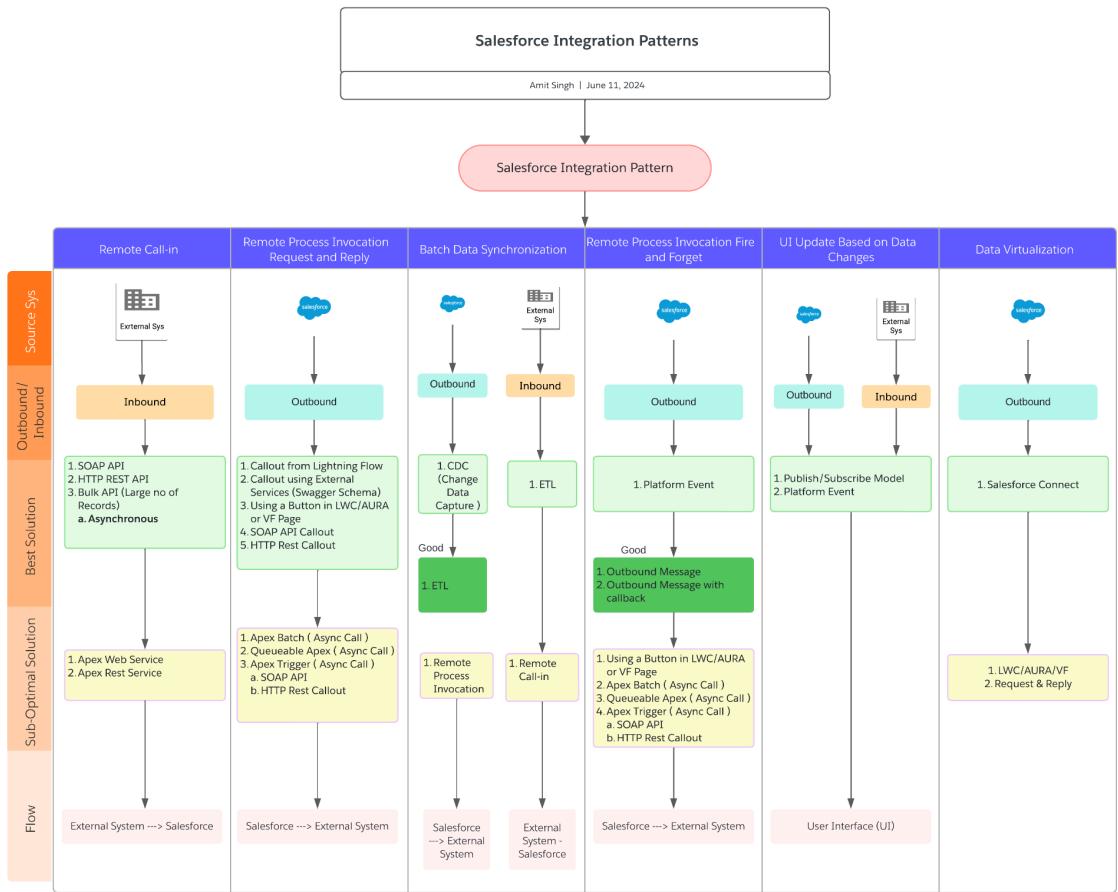
- Remote Process Invocation—Request and Reply
- Remote Process Invocation—Fire and Forget
- Batch Data Synchronization
- Remote Call-In
- UI Update Based on Data Changes
- Data Virtualization

Pattern Template

While selecting the integration pattern to solve any problem, the patterns are going to be helpful for you. Each integration pattern is going to follow a consistent structure to make it easy for you to pick the right solution for you.

Salesforce does provide the template for all the Integration Patterns and you can access those using the below link

https://developer.salesforce.com/docs/atlas.en-us.integration_patterns_and_practices.meta/integration_patterns_and_practices/integ_pat_remote_process_invocation_state.htm



Setup AWS Account

Follow the given link to create the AWS Account

<https://k21academy.com/amazon-web-services/aws-solutions-architect/create-aws-free-tier-account/>

Step by Step YouTube Link

<https://youtu.be/FsUsw0M1YmM?si=xLfbbHql1RtEvZm5>

The below content has been copied from the <https://k21academy.com/> website for the safer side if they remove the content from the web.

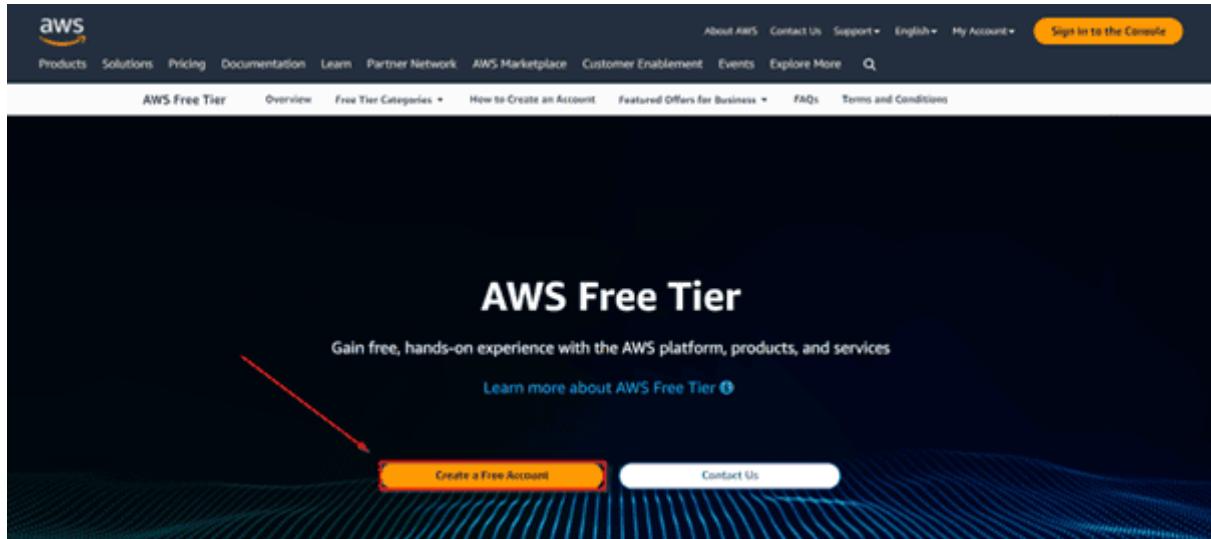
Credit goes to - <https://k21academy.com/>

This post covers the **AWS Free Tier Account** Overview. Amazon Web Services (AWS) is providing 12 months of Free Tier accounts to new subscribers to get hands-on experience with all the AWS cloud services.

In this AWS Free Tier account, Amazon is giving no. of different services used with some of the limitations to get hands-on practice and more knowledge on [AWS Cloud services](#) as well as regular business use. The AWS Free Tier is mainly designed to give hands-on experience with [AWS Cloud Services](#) for customers free of cost for a year. With the AWS Free Tier account, all the services offered have a limit on what we can use without being charged.

REGISTER FOR AWS FREE-TIER ACCOUNT

- Step 1:** First Open your web browser and navigate to the [AWS Free Tier Page](#)
Step 2: On the middle click of **Create a Free Account**



- Step 3: Verify your email address.**



Sign up for AWS

Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.



Root user email address

Used for account recovery and some administrative functions

@k21academy.com

AWS account name

Choose a name for your account. You can change this name in your account settings after you sign up.

Verify email address

OR

Sign in to an existing AWS account



- **Provide password:** Provide the details that you want to use to log in to your **AWS** account and click on **Continue**

Panther!

Explore Free Tier products with a new AWS account.

To learn more, visit aws.amazon.com/free.



Sign up for AWS

Create your password

It's you! Your email address has been successfully verified.

Your password provides you with sign in access to AWS, so it's important we get it right.

Root user password

Confirm root user password

OR



- **Email address:** Enter the email ID that hasn't registered yet with Amazon AWS
- **Password:** Type your Password
- **Confirm password:** Confirm the Password
- **Captcha:** enter the given security check

Step 4: Contact Information

Select your AWS type (Professional/ Personal) Fill in the correct information to validate your account if you're going to create personal use then click on **“Personal Account”** else use **“Company Account”**, Accept the Terms and condition and then click on **Create Account and Continue**

Free Tier offers

All AWS accounts can explore 3 different types of free offers, depending on the product used.



Always free

Never expires



12 months free

Start from initial sign-up date



Trials

Start from service activation date

Sign up for AWS

Contact Information

How do you plan to use AWS?

- Business - for your work, school, or organization
 Personal - for your own projects

Who should we contact about this account?

Full Name

Shivam Kumar

Phone Number

+91 9876543210

Country or Region

India

Address

Flat No. 101, Sector 5, Noida

Apartment, suite, unit, building, floor, etc.

City

Noida

State, Province, or Region

Uttar Pradesh

Postal Code

201301

Customers with an Indian contact address are served by Amazon Web Services India Private Limited, the local seller for AWS services in India.

- I have read and agree to the terms of the AWS Customer Agreement [\[link\]](#).

Continue (step 2 of 5)

Note: Make sure to provide proper contact details and mobile number to get the Verification code from AWS.

Step 5: Payment and PAN information: In this step, you must fill in your credit card /Debit Card info and billing address and **click on Secure Submit**.

Secure verification

We will not charge you for usage below AWS Free Tier limits. We may temporarily hold up to \$1 USD (or an equivalent amount in local currency) as a pending transaction for 3-5 days to verify your identity.



Sign up for AWS

Billing Information

Credit or Debit card number

VISA MasterCard AMEX RuPay

AWS accepts most major credit and debit cards. To learn more about payment options, review our [FAQ](#).

Expiration date

March
▼
2027
▼

Security code ⓘ

Cardholder's name

Save card information for faster future payments
Securely save card information payments as per RBI guidelines. [Learn more](#).

Billing address

Use my contact address
(Red arrow points here)

Use a new address

Do you have a PAN?
Permanent Account Number (PAN) is a ten-digit alphanumeric number issued by the Indian Income Tax Department. This 10-digit number is printed on the front of your PAN card.

Yes
 No
You can go on the Tax Settings Page on Billing and Cost Management Console to update your PAN information.

Verify and Continue (step 3 of 5)

You might be redirected to your bank's website to authorize the verification charge.

[Read AWS CLI Secrets](#)

Manager.

Step 6: In this step, it will take you to the payment gateway to validate your payment information and for your credit card verification, Amazon will charge the minimum price based on Country. Here I have provided India, so Amazon charged **2 INR**.

Card Number
XXXX XXXX XXXX 8208

Merchant
AMAZON

Amount
Rs 2.00

Mobile
X6XXXX7XX7 

An OTP (One Time Password) has been sent to your registered mobile number. Please authenticate the transaction using this OTP.

Enter OTP

[Resend OTP](#)

OTPs are SECRET. DO NOT disclose it to anyone. Bank NEVER asks for OTP.

[Submit](#) [Cancel](#)

This page will automatically timeout after 180 seconds.

Step 7: Phone verification: Here you will be taken to an identity verification page that will already have your phone number, so you just have to select either “Text message or Voice call” Provide a valid phone number, Solve the captcha, and then click on Send SMS or Call Me Now(depending upon your selection).

Sign up for AWS

Confirm your identity

Before you can use your AWS account, you must verify your phone number. When you continue, the AWS automated system will contact you with a verification code.

How should we send you the verification code?

Text message (SMS)
 Voice call

Country or region code
India (+91)

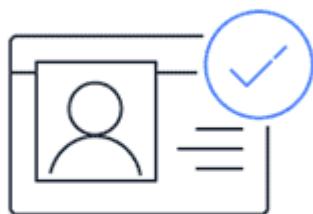
Mobile phone number
7017083421

Security check


Type the characters as shown above

[Send SMS \(step 4 of 5\)](#)

Step 8: After clicking on Send SMS or Call me Now, you will immediately receive a call or SMS from Amazon, for verification code, Enter your code then click on **Verify Code**.



Sign up for AWS

Confirm your identity

Verify code

7674

[Continue \(step 4 of 5\)](#)

Having trouble? Sometimes it takes up to 10 minutes to retrieve a verification code. If it's been longer than that, [return to the previous page](#) and try again.

Step 9: Support plan: AWS support offers a selection of plans to meet your business needs.

Select your suitable plan then click continue.

pauv



Sign up for AWS

Select a support plan

Choose a support plan for your business or personal account. Compare plans and pricing examples
 You can change your plan anytime in the AWS Management Console.

Basic support - Free

- Recommended for new users just getting started with AWS
- 24x7 self-service access to AWS resources
- For account and billing issues only
- Access to Personal Health Dashboard & Trusted Advisor



Developer support - From \$29/month

- Recommended for developers experimenting with AWS
- Email access to AWS Support during business hours
- 12 (business)-hour response times



Business support - From \$100/month

- Recommended for running production workloads on AWS
- 24x7 tech support via email, phone, and chat
- 1-hour response times
- Full set of Trusted Advisor best-practice recommendations



Need Enterprise level support?
From \$15,000 a month you will receive 15-minute response times and concierge-style experience with an assigned Technical Account Manager. [Learn more](#)

[Complete sign up](#)

Note: All customers receive free basic support.

Step 10: Registration Confirmation page.

Once you complete all the above steps and processes. You'll get the confirmation page below. Now your account will be processed for activation. It may take somewhere between 30 minutes to 1 hour for you to receive an email confirmation that your Amazon Cloud Services account has been activated.

[Sign in to the Console](#)

[Contact Sales](#) [Support](#) [English](#) [My Account](#) [Sign in to the Console](#)

[Products](#) [Solutions](#) [Pricing](#) [Documentation](#) [Learn](#) [Partner Network](#) [AWS Marketplace](#) [Customer Enablement](#) [Events](#) [Explore More](#) [Contact Sales](#)

Welcome to Amazon Web Services

Thank you for creating an Amazon Web Services Account. We are activating your account, which should only take a few minutes. You will receive an email when this is complete.

[Sign in to the Console](#)

[Check your tax details for accurate invoicing](#)

[Contact Sales](#)

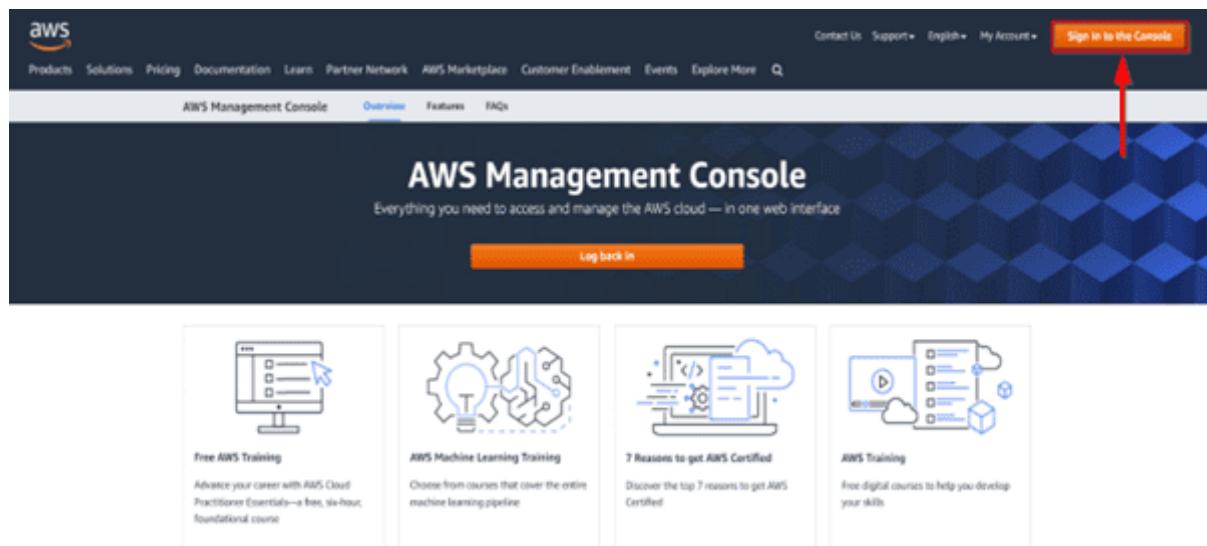
In this Blog, we have successfully created the **AWS Free Tier Account**.

Log in to the AWS Console

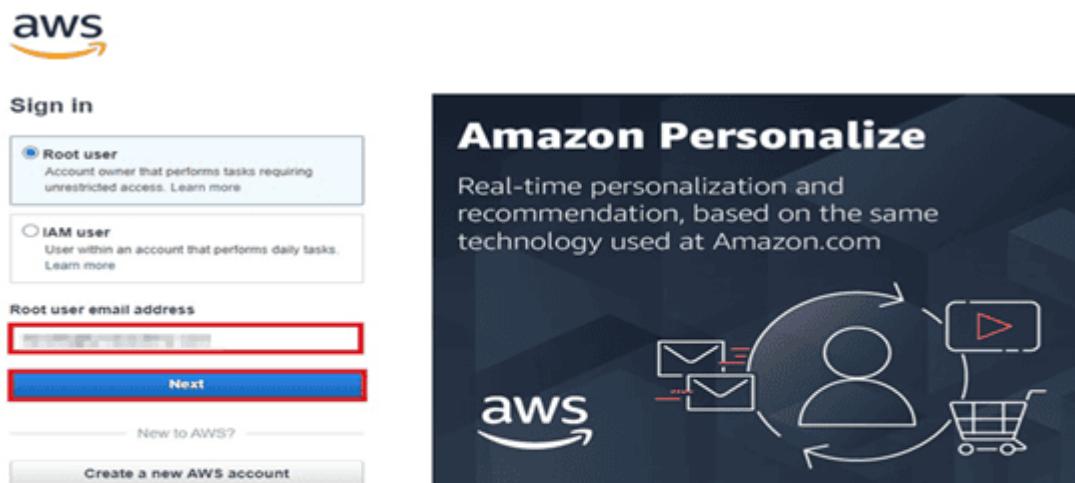
Note: Make Sure you log in as Root-user in your AWS Account.

Step 1: Open your web browser, navigate to

<https://aws.amazon.com/console/>, and **Click on sign-in to the console.**



Step 2: Enter the **username** which you have chosen while creating the account and **click on Next.**



About Amazon.com Sign In
Amazon Web Services uses information from your Amazon.com account to identify you and allow access to Amazon Web Services. Your use of this site is governed by our Terms of Use and Privacy Policy linked below. Your use of Amazon Web Services products and services is governed by the AWS Customer Agreement linked below unless you have entered into a separate agreement with Amazon Web Services or an AWS Value Added Reseller to purchase these products and services. The AWS Customer Agreement was updated on March 31, 2017. For more information about these updates, see Recent Changes.

Step 3: Enter the **password**, associated with the user and then click on **Sign in**.



Root user sign in

Email:

Password [Forgot password?](#)

[Sign in to a different account](#)

[Create a new AWS account](#)

Amazon Personalize

Real-time personalization and recommendation, based on the same technology used at Amazon.com

The diagram shows a central user profile icon. Arrows point from this profile to three separate icons: a video player (play button), an envelope (email), and a shopping cart. This visualizes how personalization technology integrates multiple user interactions (video viewing, messaging, and purchasing) to provide tailored recommendations.

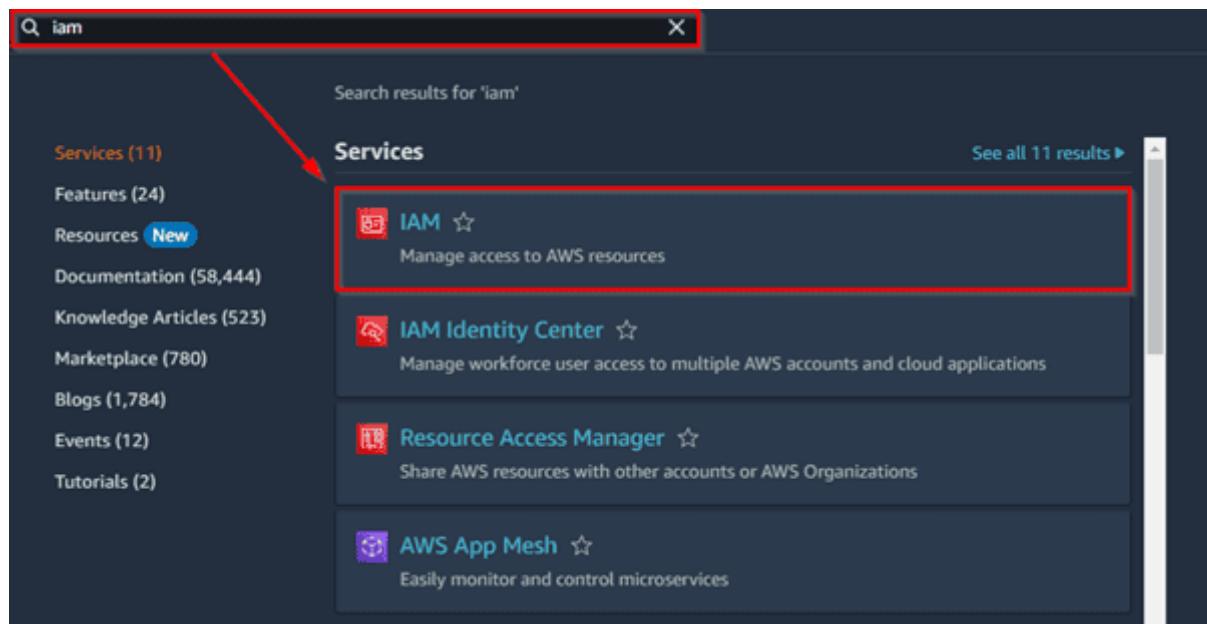
Step 4: You can see the home screen below after you log in.

The screenshot shows the AWS Console Home page. At the top, there's a navigation bar with the AWS logo, a 'Services' dropdown, a search bar, and account information for 'N. Virginia'. Below the navigation is a 'Console Home' header with a 'Recently visited' section. This section lists several services with their icons: S3, AWS Cost Explorer; Billing, AWS Health Dashboard; EC2, API Gateway; Support, CloudWatch; Resource Groups & Tag Editor, Elastic Beanstalk; and Directory Service. There are also 'Reset to default layout' and 'Add widgets' buttons.

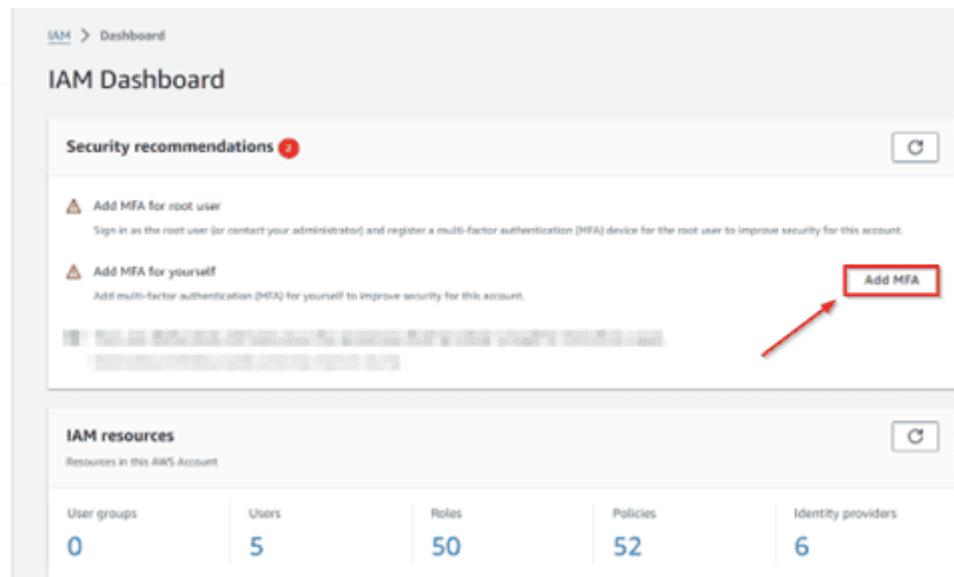
Add MFA

Note: For security reasons, you need to add MFA which provides an extra layer of security

Step 1: Navigate to IAM



Step 2: In the IAM dashboard you will see the add MFA button click on it



Step 3: Provide the **MFA device name** and then **Select the MFA device** as the **Authenticator app**. Click on Next.

Select MFA device Info

MFA device name

Device name
Enter a meaningful name to identify this device.
 Maximum 128 characters. Use alphanumeric and '-' characters.

MFA device

Select an MFA device to use, in addition to your username and password, whenever you need to authenticate.

 Authenticator app
Authenticate using a code generated by an app installed on your mobile device or computer.

 Security Key
Authenticate using a code generated by touching a YubiKey or other supported FIDO security key.

 Hardware TOTP token
Authenticate using a code displayed on a hardware Time-based one-time password (TOTP) token.

Cancel Next

Step 4: Now Install Google Authenticator on your phone.

Android: [Click here](#)

iOS: [Click here](#)

Step 5: Now open the Google Authenticator App Click on Get Started and Scan the QR code.





Step 6: Now **Click on Show QR Code in AWS Console** and **open the Google Authenticator app on your phone.** Scan the code in the phone and then Enter the code from your Phone into MFA code 1 and MFA code 2.

Then Click on the **Add MFA button.**

Set up device Info

Authenticator app

A virtual MFA device is an application running on your device that you can configure by scanning a QR code.

1

Install a compatible application such as Google Authenticator, Duo Mobile, or Authy app on your mobile device or computer.

[See a list of compatible applications](#)

2



Open your authenticator app, choose **Show QR code** on this page, then use the app to scan the code. Alternatively, you can type a secret key. [Show secret key](#)

3

Fill in two consecutive codes from your MFA device.

MFA code 1

339360

MFA code 2

249794

[Cancel](#)

[Previous](#)

Add MFA

Note: Take a screenshot of the code so that in the future if you lose your phone you can use it to re-enable MFA

Step 7: Now you will see that the device has been added for MFA

MFA device assigned
You can register up to 8 MFA devices of any combination of the currently supported MFA types with your AWS account root and IAM user. With multiple MFA devices, you only need one MFA device to sign in to the AWS console or create a session through the AWS CLI with that user.

Account details		Edit account name, email, and password	
Account name	DeepthiK21	Email address	deepthi@k21academy.com
AWS account ID	225043157131	Canonical user ID	df2935233ef6cca32082f5673d9666e98fa0064fddbc52662c441d6e610fca96

Multi-factor authentication (MFA) [1]

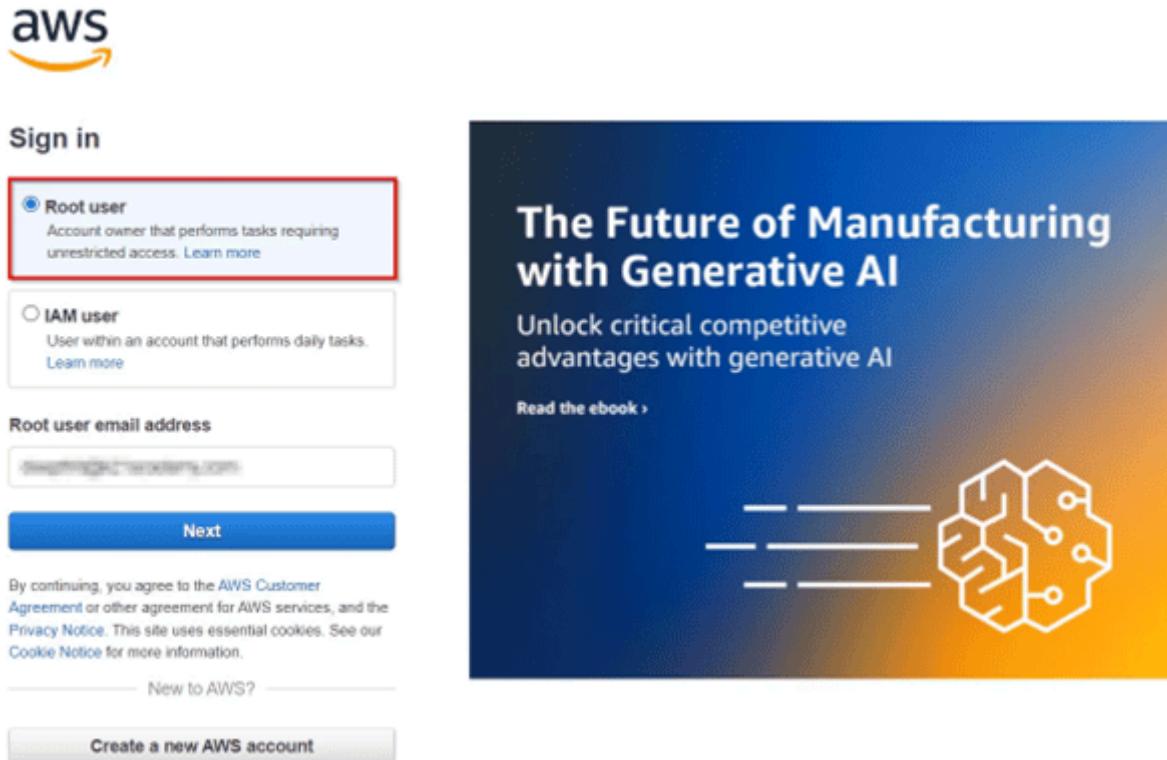
Use MFA to increase the security of your AWS environment. Signing in with MFA requires an authentication code from an MFA device. Each user can have a maximum of 8 MFA devices assigned. [Learn more](#)

Device type	Identifier	Certifications	Created on
Virtual	arn:aws:iam::225043157131:mfa/K21Academy	Not Applicable	2 minutes ago

Now you have successfully **Activated MFA on your root account** setting.

Accessing AWS Console Using MFA

Step 1: Open your AWS console login page and click on **Root User** then **enter your email ID and Click on Next.**



Step 2: Enter your **Password** corresponding to the **Email address**.





Root user sign in

Email: root@localhost.localdomain

Password [Forgot password?](#)

Sign in

[Sign in to a different account](#)

[Create a new AWS account](#)

The Future of Manufacturing with Generative AI

Unlock critical competitive advantages with generative AI

[Read the ebook >](#)



Step 3: Use your **Google Authenticator** Application on mobile and **enter the MFA code** in the **AWS Console**.



Multi-factor authentication

Your account is secured using multi-factor authentication (MFA). To finish signing in, turn on or view your MFA device and type the authentication code below.

Email address: root@localhost.localdomain

MFA code

Submit

[Troubleshoot MFA](#)

[Cancel](#)

The Future of Manufacturing with Generative AI

Unlock critical competitive advantages with generative AI

[Read the ebook >](#)



Verify Your Account

AWS Support offers four support plans: Basic, Developer, Business, and Enterprise.

By default, we have a basic plan and if you want to change your plan you can change it accordingly

To verify your account plan, click on this link:

<https://console.aws.amazon.com/support/plans/home#/>

AWS Support Plans				
At AWS, we want you to be successful.				
Features	Basic Your current plan	Developer	Business	Enterprise
Price	See pricing detail and sampler included with all AWS accounts	Review upgrade	Review upgrade	Review upgrade
Customer service and communities	24/7 access to customer service, documentation, whitepapers and AWS re:Post	24/7 access to customer service, documentation, whitepapers and AWS re:Post	24/7 access to customer service, documentation, whitepapers and AWS re:Post	24/7 access to customer service, documentation, whitepapers and AWS re:Post
Recommendations	AWS Trusted Advisor recommendations: Only service quotas and core security checks	AWS Trusted Advisor recommendations: Only service quotas and core security checks	AWS Trusted Advisor recommendations: Full set of checks	<ul style="list-style-type: none">▪ AWS Trusted Advisor recommendations: Full set of checks▪ AWS Trusted Advisor Priority: Prioritized recommendations with Trusted Advisor Priority
Health status and notifications	AWS Health Dashboard: Access to the AWS Health Dashboard	AWS Health Dashboard: Access to the AWS Health Dashboard <ul style="list-style-type: none">▪ AWS Health API: Access to the AWS Health API	<ul style="list-style-type: none">▪ AWS Health Dashboard: Access to the AWS Health Dashboard▪ AWS Health API: Access to the AWS Health API	<ul style="list-style-type: none">▪ AWS Health Dashboard: Access to the AWS Health Dashboard▪ AWS Health API: Access to the AWS Health API
Technical support	<ul style="list-style-type: none">▪ Hours of operation: Business hours access▪ Support person: AWS Cloud support engineers▪ Support channel: Email	<ul style="list-style-type: none">▪ Hours of operation: 24/7 access▪ Support person: AWS Cloud support engineers▪ Support channel: Email, chat, phone	<ul style="list-style-type: none">▪ Hours of operation: 24/7 access▪ Support person: AWS Cloud support engineers▪ Support channel: Email, chat, telephone	

Free Tier Limits



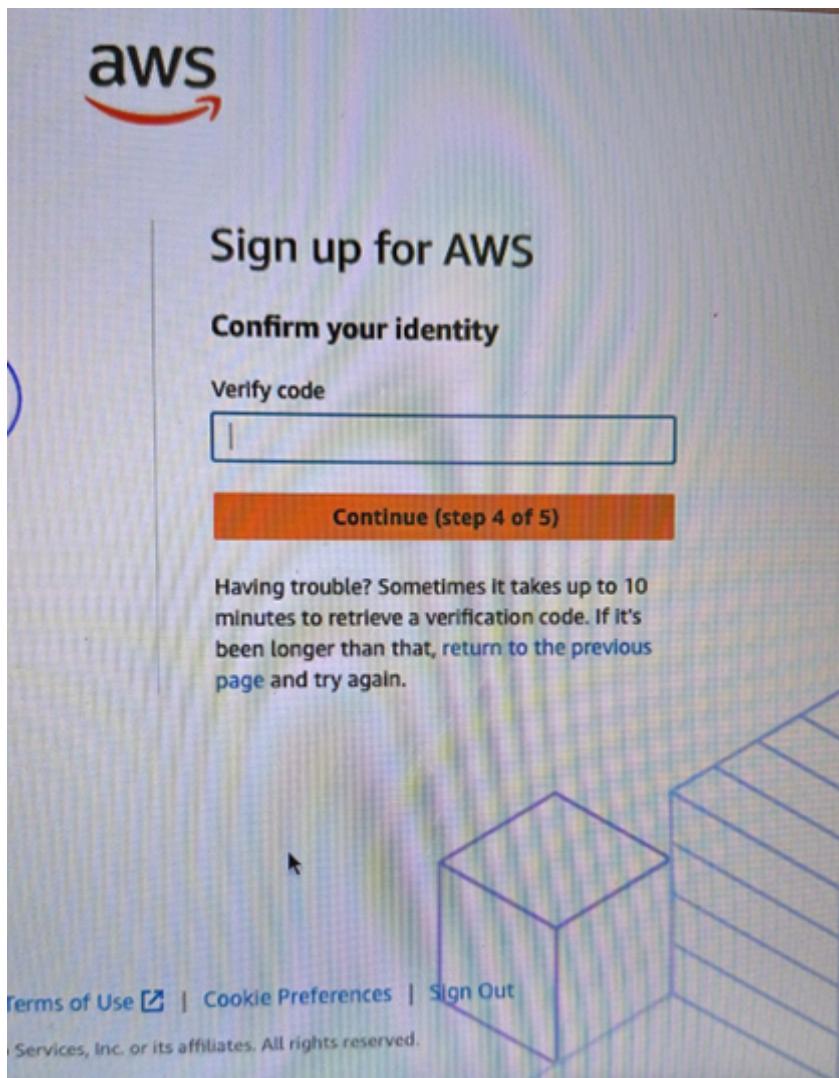
- Monthly **750 hours of Amazon EC2** Cloud computing capability that is scalable.
- **5 GB of basic storage on Amazon S3**, Infrastructure for scalable, robust, and secure object storage.
- Monthly database usage allotted to **Amazon RDS 750 hours** (for relevant database engines) SQL Server, MariaDB, PostgreSQL, and MySQL managed relational database services.

- **5 GB of Amazon EFS storage.** A shared file storage solution that is easy to use and scales for Amazon EC2 instances.
- **30 GB** of General Purpose (SSD) or Magnetic Elastic Block Storage from **Amazon Elastic Store** are long-lasting, dependable, low-latency block-level storage volumes for EC2 instances.

Troubleshooting

1. OTP not Received on your Mobile

Error: While creating an AWS Account I didn't receive an SMS or call to complete the identity verification process.

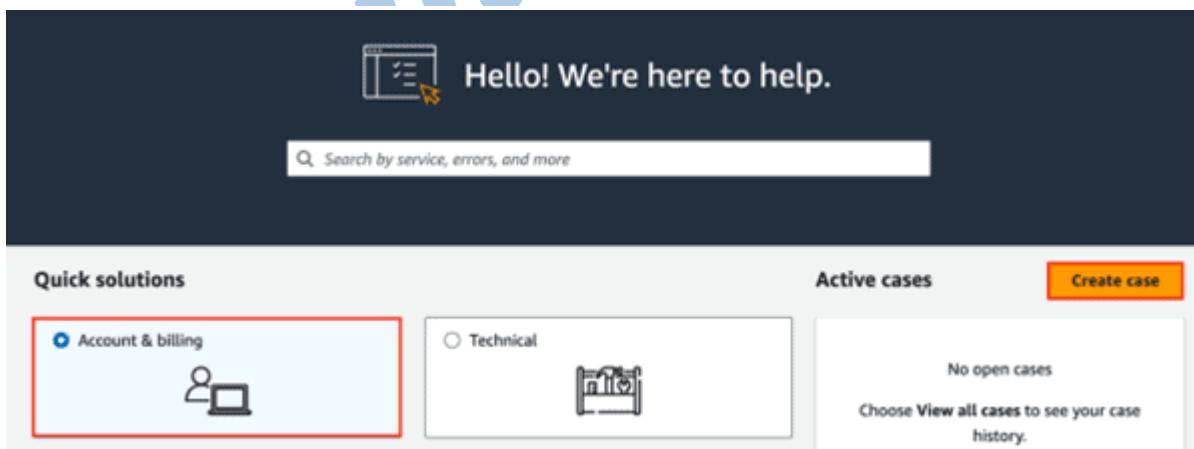


Cause: There might be some error from the backend of the AWS, The Server might be busy at that time or you might have entered the wrong number. Also, there might be a possibility that the country is not supported by AWS.

Fix: There are several fixes to this issue:

1. You've entered the correct telephone number and selected your country code correctly during the sign-up process.
2. If you have entered the Correct Telephone number & Country Code and are still not able to receive the SMS or Voice call then you need to Create a Case with the AWS Support Team in this case, they will help you to verify the telephone number.
Please follow the below steps: Create the Case with AWS Support.

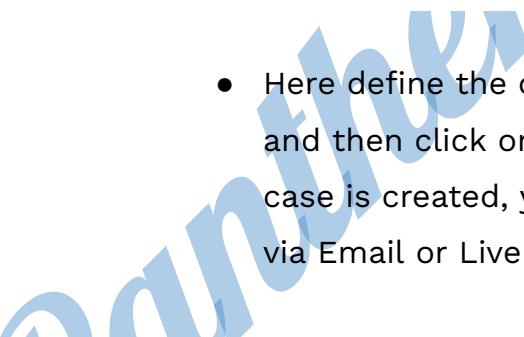
- Visit AWS Support from the below-mentioned link
<https://support.console.aws.amazon.com/support/home?region=us-east-1#/>
- Click on Create Case.



- Select the **Account and Billing** Service: **Account Activation** Category: **Account Verification**

Severity: General Question

Click on **Next Steps: Additional Information**



Account and billing
Assistance for your account, such as billing, pricing, and reserved instances.

Technical
Support for service-related technical issues, such as Amazon EC2, Amazon S3 and more.

Service

Category

Severity [Info](#)

Recommendations to common "Account Activation, Account Verification" questions

[My AWS Account is Compromised - 1000X Monthly Charge](#)

[Service charge free tier account](#)

[Amazon won't take my money](#)

▶ See more recommendations

Cancel [Next step: Additional information](#)

- Here define the complete issues that you are facing and then click on Solve Now or Contact Us. Once the case is created, you have the option to connect them via Email or Live Chat to resolve your issues

Additional information

Describe your question or issue.

Subject

Briefly summarize your question or issue

Maximum 250 characters (250 remaining)

Description

Don't share any sensitive information in case correspondences, such as credentials, credit cards, signed URLs, or personally identifiable information.

Learn more 

Describe your question or issue in detail

Maximum 8000 characters (8000 remaining)

 Attach files

You can attach up to 3 files. Each file can be up to 5 MB.



Description Guidance

To get you up and running as quickly as possible, please ensure to follow and complete all steps send via email. Please include any error messages you see.

Cancel

Previous

Next step: Solve now or contact us

2. The account is on Hold

Error: AWS Account is on Hold

Pantuv

AWS Account on Hold: Response Required

Inbox

no-reply@amazonaws.com 7:27 AM to me



Greetings from Amazon Web Services,

We were unable to validate details about your Amazon Web Services (AWS) account, so we have suspended your account. While your account is suspended, you will not be able to log in to the AWS console or access AWS services.

If you do not respond by 03/16/2023, your AWS account, and any content on the account, will be deleted. AWS reserves the right to expedite the deletion of your content in certain situations.

As soon as possible but before the date and time above, please send us a copy of a current bill (utility bill, phone bill or similar) showing your name and address. If the owner of the payment method is different from the AWS account holder, please provide current bills for both.

Please email this information using the email link below:

Cause: There can be several reasons why an AWS account is put on hold.

Some common causes include:

- Payment Issues:** If there are problems with the payment method linked to the AWS account, such as an expired credit card or insufficient funds, AWS may place the account on hold.
- Verification Required:** AWS might require additional verification of account ownership or billing information, and until the verification is completed, the account remains on hold.
- Security Concerns:** In some cases, AWS might detect suspicious activities or security issues associated with the account, prompting them to put it on hold for investigation

Fix: To resolve the “**AWS account on hold**” issue, follow these steps:

Check Payment Method, Contact AWS Support, Provide Documentation & Check for Email Notifications

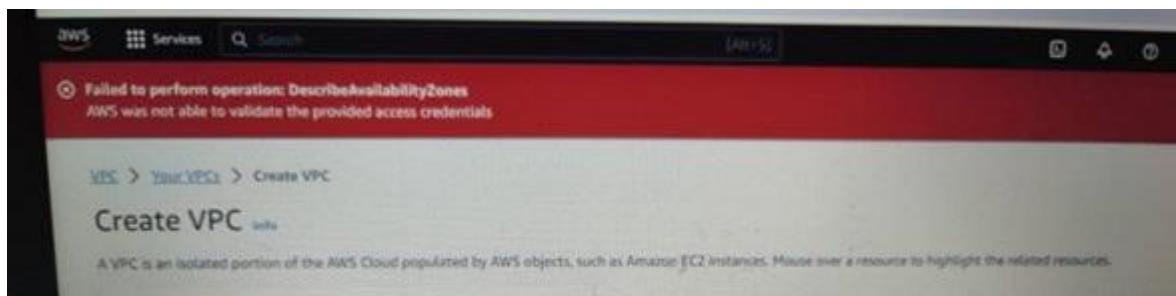
Keep in mind that the specific steps to resolve the issue may vary depending on the cause of the account hold. It is essential to address the root cause promptly and follow AWS instructions to lift the hold and regain access to your AWS resources.

Note: You can find all the details in your email.

To download the complete Certified AWS Solutions Architect Exam Questions Guide, [click here](#)

3. AWS was not able to validate the provided access credentials

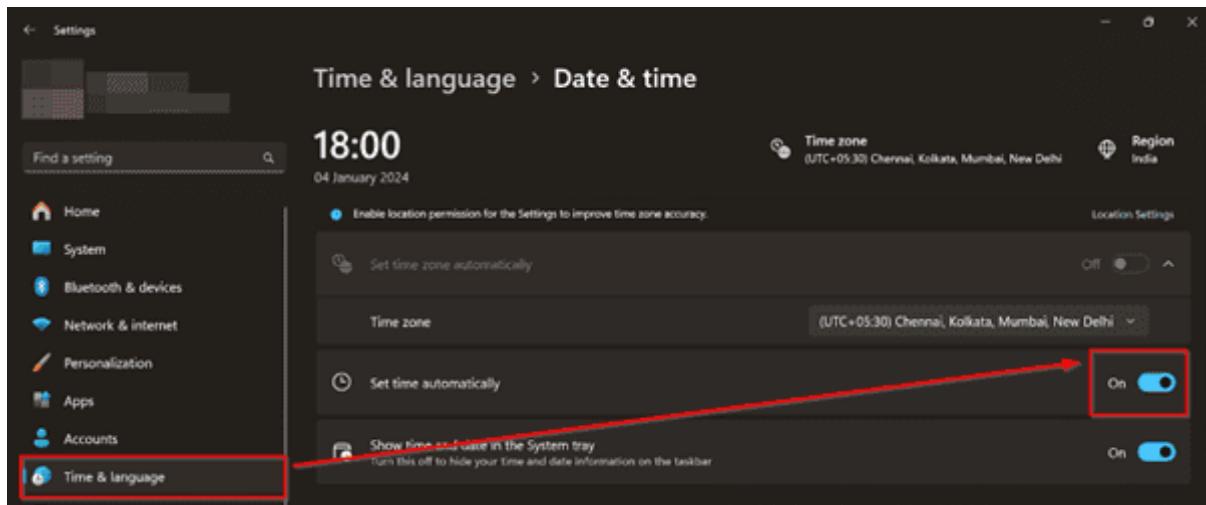
Error: Failed to operate: DescribeAvailabilityZones



Cause: The error “Failed to operate: DescribeAvailabilityZones” indicates that AWS was unable to verify your access credentials. This means it couldn’t confirm your identity and permissions to access the requested information.

This error typically occurs due to: Incorrect Time or Network connectivity issues

Fix: To Fix this issue, check if the timings of your PC (Laptop/Desktop) are correct or not. If it’s not correct you can update the timings manually and select the option Enable Set Time Automatic.



Setup DropBox Account

Use below Link to create the dropbox account and create the application

<https://www.dropbox.com/developers/>

Additional Assignments

Complete Use Case

The PantherSchools have two different Salesforce Org and sometimes the users of the first org are required to access/create the data from the second org and vice versa. Currently, they are manually switching the screens which is taking so much time and their productivity is getting degraded day by day.

The business wants to develop a connection between both the Org so that the Source Org(You can consider any org as Source) can access the data of the Target Org when needed and also create the data into the Target Org when needed.

The Architect has recommended developing a custom solution using oAuth 2.0 Web Server Flow by keeping the scalability and maintainability of the Solution.

Below is the initial requirement that the business wants you to develop

1. **Create Account** - There is a checkbox in the Source Org account object “**Sent to Target**” and if this checkbox is checked you need to create the account in the Target Org.
 - a. Following are the fields that the target organisation needs
 - i. Name
 - ii. Industry
 - iii. Rating
 - iv. Phone
 - v. Fax
 - vi. Type
 - vii. Billing Address
 - viii. Shipping Address
 - ix. Owner Name
 - x. Owner Email
 - xi. Owner Id
 - b. Once the account is created in the Target Org then update the Account ID with the record ID returned by the Target Org.
2. **Create Contact** - There is a checkbox in the Source Org Contact object “**Sent to Target**” and if this checkbox is checked you need to create the Contact in the Target Org.
 - a. Following are the fields that the target organisation needs
 - i. First Name
 - ii. Last Name
 - iii. Title
 - iv. Phone
 - v. Fax
 - vi. Email
 - vii. Mailing Address
 - viii. Other Address
 - ix. Owner Name
 - x. Owner Email
 - xi. Owner Id
 - xii. Account Id
 - b. Once the Contact is created in the Target Org then update the Contact ID with the record ID returned by the Target Org.

Prerequisites

Before we start implementing the requirement there are a few prerequisites that we need to complete so that we do not get into any issues.

1. Get Another Free Salesforce Developer Org. You can sign up from -
<https://developer.salesforce.com/signup>
2. Create a “**Sent to Target**” Checkbox field on Account & Contact within Source Org
3. Create an “**Account ID**” field on the Account Object within the Source Org
4. Create a “**Contact ID**” field on the Contact Object within the Source Org
5. Create the following fields under Account & Contact in Target org. You can create the text fields for all three
 - a. Source Owner Name
 - b. Source Owner Email
 - c. Source Owner Id

Target Org Development

#2 - Develop the Apex Rest Service in the target Org to Create an Account Record

As the requirement is to send the account information from the Source Org to the Target Org you need to develop a web service that will accept the information from the source org and will create the account record. Once the record is created it should return a success message. If there is any error the proper message should get returned to the source org.

Here are the fields that would be sent from the target org.

- Name
- Industry
- Rating
- Phone
- Fax
- Type
- Billing Address

- Shipping Address
- Owner Name
- Owner Email
- Owner Id

#3 - Develop the Apex Rest Service in the target Org to Create a Contact Record

As the requirement is to send the contact information from the Source Org to the Target Org you need to develop a web service that will accept the information from the source org and will create the contact record under the same account. Once the record is created it should return a success message. If there is any error the proper message should get returned to the source org.

Here are the fields that would be sent from the target org.

- First Name
- Last Name
- Title
- Phone
- Fax
- Email
- Mailing Address
- Other Address
- Owner Name
- Owner Email
- Owner Id
- Account Id

#4 - Develop the Apex Rest Service in the target Org to Create a Lead Record

When the Lead is created in your source org, then the same lead should be created in the target org with the following fields

- First Name
- Last Name

- Company
- Title
- Phone
- Email
- Status
- Address
- Description

#5 - Develop the Apex Rest Service in the target Org to Return the Case Record(s)

Develop an Apex Rest in target org which will accept a String parameter and as a result, it will return all the cases which are containing that text in the subject. If the parameter is blank/null then return the top 10 cases based on the created date.

Source Org Development

#1 - Authenticate Target Salesforce Org from Source Salesforce using OAuth 2.0

1. Create a connected application in the target org (second org)
2. Create a VF page in integration org to use as a redirect URI
3. Read the OAuth 2.0 document.
4. Create a custom metadata and store salesforce credential details of the custom metadata
5. Prepare the URL and get the auth code using the VF page.
6. Get the Access Token and deploy custom metadata
7. Implement the refresh token method - Consider that the Salesforce Access Token will be expired every 60 minutes

Apex Development →

1. Develop the Apex Trigger on Account that will check if the “**Send to Target**” checkbox is checked either at the time of creation or updating the account.

It should send the information to the target and update the target org account.

- a. Name
 - b. Industry
 - c. Rating
 - d. Phone
 - e. Fax
 - f. Type
 - g. Billing Address
 - h. Shipping Address
 - i. Owner Name
 - j. Owner Email
 - k. Owner Id
2. Develop the Apex Trigger on Contact that will check if the "**Send to Target**" checkbox is checked either at the time of creation or updating the Contact. It should do the following things
 - a. If the related account is not created in the target org then first call the API to create the Account Record and then create the Contact Record under the newly created account.
 - b. If the related account record is present in the target org then create the Contact record under that account
 - c. Below are the fields that you need to send for the Contact Record
 - i. First Name
 - ii. Last Name
 - iii. Title
 - iv. Phone
 - v. Fax
 - vi. Email
 - vii. Mailing Address
 - viii. Other Address
 - ix. Owner Name
 - x. Owner Email
 - xi. Owner Id
 - xii. Account Id
 3. Develop an Apex Trigger so that whenever the Lead is created in the Source Org then it should send the lead to the Target org and update the lead

record with the returned Lead Id. send the below information via the request

- a. First Name
 - b. Last Name
 - c. Company
 - d. Title
 - e. Phone
 - f. Email
 - g. Status
 - h. Address
 - i. Description
4. Create a Lightning Web Component that will have only one input that will accept the case ID or Case Number, if the user provides the Case Number or Care ID then it should return the information about the Case and display that information in the Lighting Web Component.

AWS S3 Integration with Salesforce

Project Overview

The objective of this project is to integrate AWS S3 with Salesforce to automate the creation and management of S3 buckets and folders based on Salesforce account, opportunity, and order data. The integration aims to streamline the storage of files associated with specific accounts, opportunities, and orders, ensuring that all relevant data is stored efficiently and accessible both in AWS S3 and within Salesforce.

Key Features

1. Account-Based Bucket Creation:

- a. When a new account is created in Salesforce, and the "**Sync with S3**" checkbox is checked, an AWS S3 bucket will automatically be created for that account.
 - i. The bucket name should be Account Name + Account Id all in lowercase

- ii. Salesforce.com
 - iii. salesforce-com-00178734bnhjhjsd
 - b. Two folder under the Same bucket should be created with the static name opportunities & orders
2. **Opportunity and Order Folder Management:**
- a. For each opportunity or order created under a synced account, a corresponding folder will be created within the respective account's S3 bucket.
 - i. The bucket folder name should be Opportunity Name/Order Number + Record Id all in lower case
3. **File Storage and Management:**
- a. Files associated with opportunities and orders will be uploaded to the respective folders in AWS S3.
 - b. Details of these files, including their metadata, will be stored and accessible within Salesforce.
4. **Bucket and File Display in LWC:**
- a. Fetch and display AWS S3 buckets and their contents in a Lightning Web Component (LWC).
 - b. Display folders and files differently. Folders should be navigable, and files should have a download button.
 - c. Clicking the download button should download the file to the local system.

Technical Requirements

1. **AWS S3 Setup:**
- a. Configure AWS S3 to allow programmatic creation of buckets and folders.
 - b. Setup IAM User
 - c. Ensure proper IAM roles and permissions are set up to enable Salesforce to interact with S3
2. **Salesforce Setup:**
- a. Create a custom checkbox field "Sync with S3" on the Account object.
 - b. Create a custom Text field " S3 Bucket Name" on the Account object.

- c. Create a custom Text field " S3 Folder Name" on the Opportunity & Order object.
 - d. Develop triggers to detect the creation of accounts, opportunities, and orders.
 - e. Implement Apex classes and methods to handle the interaction with AWS S3.
3. **Integration:**
- a. Utilize REST API to enable communication between Salesforce and AWS S3.
 - b. Implement error handling and logging to monitor integration processes.
4. Create an LWC to fetch and display AWS S3 buckets and their contents.
- a. Implement logic to distinguish between folders and files.
 - b. Provide download functionality for files.

Workflow

1. **Account Creation and Sync:**
 - a. A new account is created in Salesforce.
 - b. If the "Sync with S3" checkbox is checked, an S3 bucket is created with the account's name or a unique identifier.
 - c. The bucket's details are stored in a custom object or field within Salesforce for reference.
2. **Opportunity/Order Creation:**
 - a. When a new opportunity or order is created under a synced account, a corresponding folder is created in the account's S3 bucket.
 - b. The folder's name includes the opportunity or order ID to maintain uniqueness.
3. **File Upload and Management:**
 - a. Files related to opportunities or orders are uploaded to their respective folders in S3.
 - b. Metadata and details of these files, such as file name, upload date, and URL, are recorded in Salesforce.
 - c. Implement file upload functionality within Salesforce using Visualforce, Lightning Components, or standard file upload mechanisms.

4. Bucket and File Display in LWC:

- a. Create an LWC to fetch and display the list of AWS S3 buckets.
- b. When a bucket is selected, display its contents (folders and files).
- c. For folders, allow navigation into the folder to view its contents.
- d. For files, provide a download button. When clicked, the file should be downloaded to the local system.

Benefits

1. **Streamlined File Management:** Automates the creation and organization of files related to accounts, opportunities, and orders.
2. **Centralized Storage:** Ensures all files are stored in a centralized, scalable, and secure manner using AWS S3.
3. **Enhanced Data Accessibility:** Provides seamless access to file details and metadata within Salesforce.
4. **Efficiency:** Reduces manual effort and errors in managing account-related files.

Code used in the Class

AWS S3 Manager Apex Class

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 06-23-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
public with sharing class AWSS3Manager {  
  
    public static void createBucket(String bucketName, String recordId,  
        String region){  
  
        bucketName = bucketName.toLowerCase().replaceAll(' ', '-') +  
            '-' +recordId.toLowerCase();
```

```

// Salesforce India Pvt Ltd --> salesforce-india-pvt-ltd-recordId
System.debug(bucketName);

HttpRequest httpReq = new HttpRequest();
httpReq.setEndpoint('callout:AWSS3Cred'+'/'+
EncodingUtil.urlEncode(bucketName, 'UTF-8') );
httpReq.setMethod('PUT');

String requestBody = '<?xml version="1.0" encoding="UTF-8"?>' +
'<CreateBucketConfiguration
xmlns="http://s3.amazonaws.com/doc/2006-03-01/">' +
'<LocationConstraint>' +region+ '</LocationConstraint>' +
'</CreateBucketConfiguration>';

if(String.isNotBlank(region) &&
!region.equalsIgnoreCase('us-east-1') ){
    httpReq.setBody(requestBody);
}

try {
    HttpResponse response = (new Http()).send(httpReq);
    if(response.getStatusCode() == 200 || response.getStatusCode()
== 201){

        createFolderInsideBucket(bucketName, 'opportunities');
        createFolderInsideBucket(bucketName, 'orders');

        // Update the account with the bucket Name
        Account acc = new Account(Id = recordId, S3BucketName__c =
bucketName);
        update acc;
    } else {
        // handle Error
        System.debug( response.getBody() );
    }
}

```

```

    } catch (System.Exception ex) {
        System.debug(' '+ex.getMessage() );
    }
}

public static void createFolderInsideBucket(String bucketName, String
folderName){
    HttpRequest httpReq = new HttpRequest();
    httpReq.setEndpoint('callout:AWSS3Cred'+'/'+
EncodingUtil.urlEncode(bucketName, 'UTF-8') + '/' +
EncodingUtil.urlEncode(folderName, 'UTF-8') + '/');
    httpReq.setMethod('PUT');
    httpReq.setHeader('Content-Length', '0');
    try {
        HttpResponse response = (new Http()).send(httpReq);
        if(response.getStatusCode() == 200 || response.getStatusCode()
== 201){

    } else {
        System.debug( response.getBody() );
    }
} catch (System.Exception ex) {
    System.debug(' '+ex.getMessage() );
}
}

public static void createFolderInsideBucketFolder(String bucketName,
String folderPath, String folderName){
    HttpRequest httpReq = new HttpRequest();
    httpReq.setEndpoint('callout:AWSS3Cred'+'/'+
EncodingUtil.urlEncode(bucketName, 'UTF-8') + '/' +
EncodingUtil.urlEncode(folderPath, 'UTF-8') + '/' +
EncodingUtil.urlEncode(folderName, 'UTF-8') + '/');
    httpReq.setMethod('PUT');
    httpReq.setHeader('Content-Length', '0');
}

```

```

try {
    HttpResponse response = (new Http()).send(httpReq);
    if(response.getStatusCode() == 200 || response.getStatusCode()
== 201){

} else {
    System.debug( response.getBody() );
}
} catch (System.Exception ex) {
    System.debug(' '+ex.getMessage());
}
}

public static void createFolderInsideBucket(String endpoint){
    // EncodingUtil.urlEncode(bucketName, 'UTF-8') + '/' +
EncodingUtil.urlEncode(folderPath, 'UTF-8') + '/' +
EncodingUtil.urlEncode(folderName, 'UTF-8') + '/'

HttpRequest httpReq = new HttpRequest();
httpReq.setEndpoint('callout:AWSS3Cred'+'/'+ endpoint);
httpReq.setMethod('PUT');
httpReq.setHeader('Content-Length', '0');

try {
    HttpResponse response = (new Http()).send(httpReq);
    if(response.getStatusCode() == 200 || response.getStatusCode()
== 201){

} else {
    System.debug( response.getBody() );
}
} catch (System.Exception ex) {
    System.debug(' '+ex.getMessage());
}
}

/*

```

```

        String endPoint =
EncodingUtil.urlEncode('panther-schools-001hu00003cvddciaj', 'UTF-8') +
'/' + EncodingUtil.urlEncode('orders', 'UTF-8') + '/' +
EncodingUtil.urlEncode('00000107', 'UTF-8');
AWSS3Manager.uploadFile('068Hu00000bBYf4IAG', endPoint);

Trigger - Content Version
ContentDocumentId
    LinkedEntityId FROM ContentDocumentLink WHERE
ContentDocumentId =:
    if(link.LinkedEntityId.getObjectType() ==
Order.getObjectType()
        || link.LinkedEntityId.getObjectType() ==
Opportunity.getObjectType())
/*
public static void uploadFile(String recordId, String endPoint){
    ContentVersion version = [SELECT Id, VersionNumber, Title,
FileType, FileExtension, VersionData
        FROM ContentVersion WHERE Id =:
recordId];

    String fileName = version.Title+'.'+version.FileExtension;
    HttpRequest httpReq = new HttpRequest();
    httpReq.setEndpoint('callout:AWSS3Cred'+'/'+ endPoint
+ '/' +fileName );
    httpReq.setMethod('PUT');
    httpReq.setBodyAsBlob(version.VersionData);

try {
    HttpResponse response = (new Http()).send(httpReq);
    if(response.getStatusCode() == 200 || response.getStatusCode()
== 201){

} else {
    // handle Error
}
}

```

```

        System.debug( response.getBody() );
    }
} catch (System.Exception ex) {
    System.debug(' '+ex.getMessage());
}
}

/*
String endPoint =
EncodingUtil.urlEncode('panther-schools-001hu00003cvddciaj', 'UTF-8');
AWSS3Manager.listBucketItems(endPoint);
*/
public static void listBucketItems(String endPoint){
    HttpRequest httpReq = new HttpRequest();
    httpReq.setEndpoint('callout:AWSS3Cred'+'/'+endPoint+'/');
    httpReq.setMethod('GET');

    try {
        HttpResponse response = (new Http()).send(httpReq);
        String ns = 'http://s3.amazonaws.com/doc/2006-03-01/';
        if(response.getStatusCode() == 200 || response.getStatusCode()
== 201){
            Dom.Document doc = response.getBodyDocument();
            Dom.XmlNode rootNote = doc.getRootElement();
            Dom.XmlNode nameNode = rootNote.getChildElement('Name',
ns);
            String bucketName = '';
            if(nameNode != null){
                bucketName = nameNode.getText();
            }

            ListBucketResult itemWrapper = new ListBucketResult();
            itemWrapper.name = bucketName;
            itemWrapper.items = new List<ListBucketResult.Contents>();

```

```
for(Dom.XmlNode root : rootNote.getChildElements() ){
    if(root.getName() == 'Contents'){ // multiple items
        ListBucketResult.Contents item = new
ListBucketResult.Contents();
        for(Dom.XmlNode contentNode :
root.getChildElements() ){
            String nodeName = contentNode.getName();
            String nodeValue = contentNode.getText();
            switch on nodeName {
                when 'Key' {
                    item.key = nodeValue;
                }
                when 'LastModified' {
                    }

                }
                when 'ETag' {
                    }

                }
                when 'Size' {
                    item.size = nodeValue;
                }
                when 'StorageClass' {
                    }

                }
                when 'Owner' {
                    for(Dom.XmlNode ownerNode:
contentNode.getChildElements() ) {
                        String nodeOwnerName =
contentNode.getName();
                        switch on nodeOwnerName {
                            when 'ID' {
                                }

                            }
                            when 'DisplayName' {
                                
```

```
        }
    }
}
}
itemWrapper.items.add(item);
}
}

System.debug(itemWrapper);

} else {
    // handle Error
    System.debug( response.getBody() );
}

} catch (System.Exception ex) {
    System.debug(' '+ex.getMessage());
}

}
}
```

ListBucketResult Apex Class

```
/***
 * @description      :
 * @author           : Amit Singh - PantherSchools
 * @group            :
 * @last modified on : 06-23-2024
 * @last modified by : Amit Singh - PantherSchools
 */
```

```
/**/

public with sharing class ListBucketResult {
    public String name;
    public List<Contents> items;

    public class Contents {
        public String key;
        public String size;
    }
}
```

AccountTriggerHandler Apex Class

```
/***
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-23-2024
 * @last modified by : Amit Singh - PantherSchools
 ***/

public with sharing class AccountTriggerHandler {

    public static void createBucket(List<Account> newRecords){
        for (Account acc : newRecords) {
            createBucket(acc.Name, acc.Id, '');
        }
    }

    @future(callout = true)
```

```

    public static void createBucket(String bucketName, String recordId,
String region){
        AWSS3Manager.createBucket(bucketName, recordId, region);
    }
}

```

OrderTriggerHandler Apex Class



```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-23-2024
 * @last modified by : Amit Singh - PantherSchools
*/
public with sharing class OrderTriggerHandler {

    public static void createFolderInsideBucket(Order order){
        List<Account> accList = [SELECT Id, SyncWithS3__c,
S3BucketName__c FROM Account WHERE Id =: order.AccountId LIMIT 1];
        if(accList?.size() > 0 && accList.get(0).SyncWithS3__c &&
String.isNotBlank( accList.get(0).S3BucketName__c ) ){
            // create a folder
            createFolderInsideBucket(accList.get(0).S3BucketName__c,
'orders',
order.OrderNumber.toLowerCase()+' - '+String.valueOf(order.Id).toLowerCase());
        }
    }

    @future(callout = true)
    public static void createFolderInsideBucket(String bucketPath, String

```

```

FolderPath, String folderName ) {
        String endPoint = EncodingUtil.urlEncode(bucketPath, 'UTF-8') +
        '/' + EncodingUtil.urlEncode(folderPath, 'UTF-8') + '/' +
        EncodingUtil.urlEncode(folderName, 'UTF-8') + '/';
        AWSS3Manager.createFolderInsideBucket(endPoint);
    }
}

```

AccountTrigger Apex Class

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-23-2024
 * @last modified by : Amit Singh - PantherSchools
 */
trigger AccountTrigger on Account (after insert, after update) {

    if(Trigger.isAfter){
        if(Trigger.isInsert && !System.isBatch() && !System.isFuture() &&
        !System.isQueueable() ){
            Account acc = Trigger.new.get(0);
            if(acc.SyncWithS3__c == True &&
            String.isBlank(acc.S3BucketName__c)){
                AccountTriggerHandler.createBucket(new
List<Account>{acc});
            }
        }
        if(Trigger.isUpdate && !System.isBatch() && !System.isFuture() &&
        !System.isQueueable() ){
            // check if sync with s3 is changed and marked as true
        }
    }
}

```

```

        Account acc = Trigger.new.get(0);
        Account oldRecord = Trigger.oldMap.get(acc.Id);
        if(acc.SyncWithS3__c <> oldRecord.SyncWithS3__c &&
acc.SyncWithS3__c == True && String.isBlank(acc.S3BucketName__c) ){
            AccountTriggerHandler.createBucket(new
List<Account>{acc});
        }
    }
}

```

OrderTrigger Trigger

```

/**
 * @description      :
 * @author          : Amit Singh - PantherSchools
 * @group           :
 * @last modified on : 06-23-2024
 * @last modified by : Amit Singh - PantherSchools
*/
trigger OrderTrigger on Order (after insert) {
    if(Trigger.isAfter){
        if(Trigger.isInsert && !System.isBatch() && !System.isFuture() &&
!System.isQueueable() ){
            OrderTriggerHandler.createFolderInsideBucket(Trigger.new.get(0));
        }
    }
}

```

AWSErrorWrapper Apex Class

```
public class AWSErrorWrapper {  
  
    public String code;  
    public String message;  
    public String requestId;  
    public String hostId;  
  
}
```

AWSErrorXmlParser Apex Class

```
/**  
 * @description      :  
 * @author          : Amit Singh - PantherSchools  
 * @group           :  
 * @last modified on : 06-23-2024  
 * @last modified by : Amit Singh - PantherSchools  
 */  
  
public class AWSErrorXmlParser {  
    public static AWSErrorWrapper parseXmlResponse(String xmlResponse) {  
        try {  
            Dom.Document doc = new Dom.Document();  
            doc.load(xmlResponse);  
  
            Dom.XMLNode rootNode = doc.getRootElement();  
  
            String code      = rootNode.getChildElement('Code',  
null).getText();  
            String message   = rootNode.getChildElement('Message',  
null).getText();  
        }  
    }  
}
```

```

        String requestId      = rootNode.getChildElement('RequestId',
null).getText();
        String hostId         = rootNode.getChildElement('HostId',
null).getText();

        AWSErrorWrapper wrapper = new AWSErrorWrapper();
        wrapper.code          = code;
        wrapper.message       = message;
        wrapper.requestId    = requestId;
        wrapper.hostId        = hostId;

        return wrapper;
    } catch (Exception e) {
        System.debug('Exception: ' + e.getMessage());
        throw new XmlException( e.getMessage() );
    }
}
}
}

```

Integration Interview Questions

General Integration Questions

1. What is Salesforce integration and why is it important?
2. What are the different methods available for integrating Salesforce with other systems?
3. What is an API and how is it used in Salesforce integration?
4. What are the different types of APIs available in Salesforce (REST, SOAP, Bulk, Streaming)?
5. What are the best practices for integrating Salesforce with external systems?
6. What security considerations should be taken into account when integrating with Salesforce?
7. How can you handle data synchronisation between Salesforce and external systems?

Specific Integration Scenarios

8. How do you integrate Salesforce with an ERP system (e.g., SAP, Oracle)?
9. How can you connect Salesforce with a marketing automation tool (e.g., Marketo, HubSpot)?
10. What are the steps to integrate Salesforce with a payment gateway (e.g., PayPal, Stripe)?
11. How do you set up a real-time data integration between Salesforce and a database (e.g., SQL Server, MySQL)?
12. What are the considerations for integrating Salesforce with a custom-built application?
13. How can you integrate Salesforce with a telephony system (e.g., Twilio, Avaya)?



Tools and Middleware

14. What middleware solutions can be used for Salesforce integration (e.g., MuleSoft, Dell Boomi, Informatica)?
15. How do you configure and use MuleSoft for Salesforce integration?
16. What is Salesforce Connect and how can it be used for integration?
17. What are the advantages of using middleware for Salesforce integration?
18. How do you set up and use an ETL (Extract, Transform, Load) tool for Salesforce integration?



Salesforce-Specific Tools

19. What is the Salesforce Data Loader and how is it used for integration?
20. How can you use Salesforce External Services for integration?
21. What is Salesforce's Platform Events and how can it be used for integration?
22. How do you configure Salesforce to use Named Credentials for external callouts?
23. What are Salesforce Connect and External Objects, and how do they facilitate integration?

Authentication and Security

24. How do you set up OAuth 2.0 for secure Salesforce integration?
25. What is the difference between user-level and application-level

authentication in Salesforce?

26. How do you manage API limits and quotas in Salesforce integrations?
27. What is the importance of IP whitelisting in Salesforce integrations?

Troubleshooting and Monitoring

28. How do you monitor and debug Salesforce integration issues?
29. What are some common errors encountered during Salesforce integration and how can they be resolved?
30. How do you ensure data integrity during Salesforce integrations?
31. What tools can be used for monitoring Salesforce API usage and performance?



Advanced Topics

32. How do you handle complex data transformations during Salesforce integration?
33. What is the role of asynchronous processing in Salesforce integration?
34. How can you leverage Salesforce's asynchronous Apex (e.g., Future Methods, Batch Apex) for integration tasks?
35. What are the considerations for integrating Salesforce with IoT (Internet of Things) devices?
36. How do you implement a microservices architecture for Salesforce integration?



Integration Use Cases

37. How do you integrate Salesforce with a content management system (e.g., WordPress, Drupal)?
38. How can Salesforce be integrated with social media platforms (e.g., Facebook, Twitter)?
39. What are the steps to integrate Salesforce with an e-commerce platform (e.g., Shopify, Magento)?
40. How can Salesforce be integrated with a Customer Relationship Management (CRM) system?
41. How do you integrate Salesforce with cloud storage solutions (e.g., Google Drive, Dropbox)?

42. How can Salesforce be integrated with a project management tool (e.g., Jira, Asana)?

Apex REST

- 43. What is Apex REST in Salesforce?
- 44. How do you create a custom Apex REST API in Salesforce?
- 45. What are the best practices for designing and implementing Apex REST services?
- 46. How do you handle authentication and authorization in Apex REST?
- 47. How do you test and debug Apex REST services?
- 48. How do you handle JSON and XML payloads in Apex REST?
- 49. What are the considerations for versioning Apex REST APIs?
- 50. How do you implement error handling in Apex REST services?
- 51. How do you consume external REST APIs using Apex in Salesforce?
- 52. What are some common use cases for Apex REST in Salesforce?



Platform Events

- 53. What are Platform Events in Salesforce and how do they work?
- 54. How do you create and publish Platform Events in Salesforce?
- 55. How do you subscribe to Platform Events in Salesforce?
- 56. What are the differences between high-volume and standard-volume Platform Events?
- 57. How do you use Platform Events to integrate Salesforce with external systems?
- 58. How do you ensure reliability and fault tolerance when using Platform Events?
- 59. What are the considerations for designing event-driven architectures with Platform Events?
- 60. How do you monitor and debug Platform Events in Salesforce?
- 61. How do Platform Events differ from other Salesforce integration patterns, such as Change Data Capture and Streaming API?
- 62. What are some best practices for managing Platform Events?

Unit Testing for Integration

63. What are the best practices for writing unit test cases for Salesforce integrations?
64. How do you mock external service callouts in Salesforce unit tests?
65. What is the HttpCalloutMock interface and how is it used in Salesforce tests?
66. How can you test REST API integrations in Salesforce?
67. How do you write unit tests for SOAP API integrations in Salesforce?
68. What strategies can you use to test integrations with Platform Events?
69. How do you use Test.setMock and Test.startTest/Test.stopTest methods in integration tests?
70. How do you ensure code coverage for external callouts in Salesforce?
71. What are some common challenges in testing Salesforce integrations and how can they be addressed?
72. How do you validate the data returned from external services in unit tests?
73. How do you test error handling and retry logic for failed integrations?
74. How do you write test cases for Apex REST services in Salesforce?
75. How can you test complex data transformations during integration?
76. How do you ensure data integrity and consistency in integration test cases?
77. How do you test asynchronous integrations, such as those using Future Methods, Batch Apex, or Queueable Apex?
78. What tools and frameworks can assist in writing and running integration tests in Salesforce?
79. How do you use the Test.setMock method to simulate different responses from external services?
80. What are the considerations for testing integrations in a multi-org environment?
81. How do you handle API limits and quotas in integration test cases?
82. How do you verify the correctness of integration logic in unit tests?
83. How can you test integrations involving Salesforce Connect and External Objects?
84. How do you write test cases for integrations using Change Data Capture?
85. How can you test the end-to-end flow of an integration in Salesforce?
86. What are the best practices for organising and maintaining integration test cases?

Auth Providers and Named Credentials

Concepts:

- Explain the difference between Auth Providers and Named Credentials in Salesforce.
- Describe scenarios where you would use Auth Providers vs. Named Credentials.
- Discuss the security benefits of using Named Credentials.



Configuration:



- How do you create a Named Credential in Salesforce?
- What information is required to configure a Named Credential?
- How can you assign permissions to users to access specific Named Credentials?
- Explain the role of External Credentials in conjunction with Named Credentials.



Integration:



- Describe how Named Credentials can be used to simplify callouts to external systems during integrations.
- How do Named Credentials eliminate the need for Remote Site Settings in some cases?
- Can you provide an example of how you would use Named Credentials and Auth Providers together for an integration project?



Security:



- How does Salesforce secure the credentials stored within Named Credentials?
- What are some best practices for managing the security of Named Credentials?

Troubleshooting:

- Describe some common issues that can arise when using Named Credentials and how would you troubleshoot them?

- How can Named Credentials help with debugging integration issues?

Bonus:

- Discuss the limitations of Named Credentials and when alternative approaches might be preferable.
- How do Named Credentials integrate with Salesforce Open API (SOAP or REST)?



Bulk API



- What is the Bulk API, and what are its advantages for data manipulation in Salesforce?
- When would you choose Bulk API over the standard Data Loader for data import/export?
- Explain the concept of batch jobs and governor limits in the context of the Bulk API.
- How do you handle errors and exceptions when using the Bulk API?



Metadata API



- What is the Metadata API, and what functionalities does it offer?
- Describe some common use cases for the Metadata API in Salesforce development.
- How can the Metadata API be used for automating deployments and managing configurations?
- What are some security considerations when using the Metadata API?
- How does the Metadata API compare to tools like Salesforce DX for managing deployments?

Integration with Other APIs

- Can Named Credentials be used for authentication with the Bulk API or Metadata API?
- How do Auth Providers interact with other Salesforce APIs like REST or SOAP API?
- Discuss potential scenarios where Bulk API, Metadata API, and Named Credentials might be used together for an integration project.

Salesforce Integration Fundamentals

General Concepts

- Explain the benefits of integrating Salesforce with other applications.
- What are some common challenges faced during Salesforce integrations?
- Describe different types of Salesforce integration patterns (e.g., point-to-point, middleware-based).
- How do you ensure data consistency and accuracy during Salesforce integrations?



Integration Techniques



- Explain the role of APIs (REST, SOAP) in Salesforce integrations.
- How can Salesforce workflows and processes be leveraged for integration purposes?
- Describe the concept of Change Data Capture (CDC) in the context of Salesforce integrations.
- Discuss the role of tools like MuleSoft or Informatica Cloud in complex integrations.



Security Considerations



- How do you secure data flow between Salesforce and external systems during integrations?
- What are some best practices for managing user access and authorization for integrations?

Troubleshooting



- Describe common troubleshooting techniques for Salesforce integration issues.
- How can logs and debug logs be used to identify and resolve integration errors?

Additional Topics

- Discuss the role of Apex code in developing custom integrations within Salesforce.

- How can pre-built connectors and packages from Salesforce AppExchange simplify integrations?
- Explain the concept of Governor Limits and their impact on integrations.

Scenario-Based Questions

- How would you approach integrating Salesforce with a legacy system that only supports file-based data exchange?
- You need to sync customer data between Salesforce and an external CRM system in real-time. What integration strategy would you use and why?
- Your Salesforce org needs to integrate with an external service that has rate limits on API calls. How would you design your integration to handle this?
- Describe how you would implement an integration that allows Salesforce to receive updates from an external system via webhooks.
- You are tasked with migrating data from an on-premises database to Salesforce. What tools and processes would you use?
- How would you handle data transformations when integrating Salesforce with a system that has a different data model?
- An external system requires data from Salesforce, but the data should only be sent once per day. How would you implement this requirement?
- You need to integrate Salesforce with an ERP system to automate order processing. Describe your approach.
- How would you ensure secure data transmission between Salesforce and an external system?
- A third-party service provides data to Salesforce via a REST API. How would you consume this data and ensure it is correctly processed?
- You have to integrate Salesforce with a system that supports only SOAP-based web services. What steps would you take?
- How would you use Platform Events to notify external systems about changes in Salesforce records?
- You need to build an integration that supports bidirectional data flow between Salesforce and an external system. What challenges might you face and how would you address them?

- Describe a scenario where you would use Salesforce Connect for integration and explain why it's suitable.
- How would you handle error logging and monitoring for a Salesforce integration that uses Apex callouts?
- An integration you developed is failing due to network issues. How would you troubleshoot and mitigate these failures?
- How would you design an integration to handle high data volumes between Salesforce and an external system?
- You are asked to integrate Salesforce with a third-party payment gateway. Describe your approach to handle transactions securely.
- A client needs to integrate Salesforce with multiple external systems. How would you ensure the integrations are maintainable and scalable?