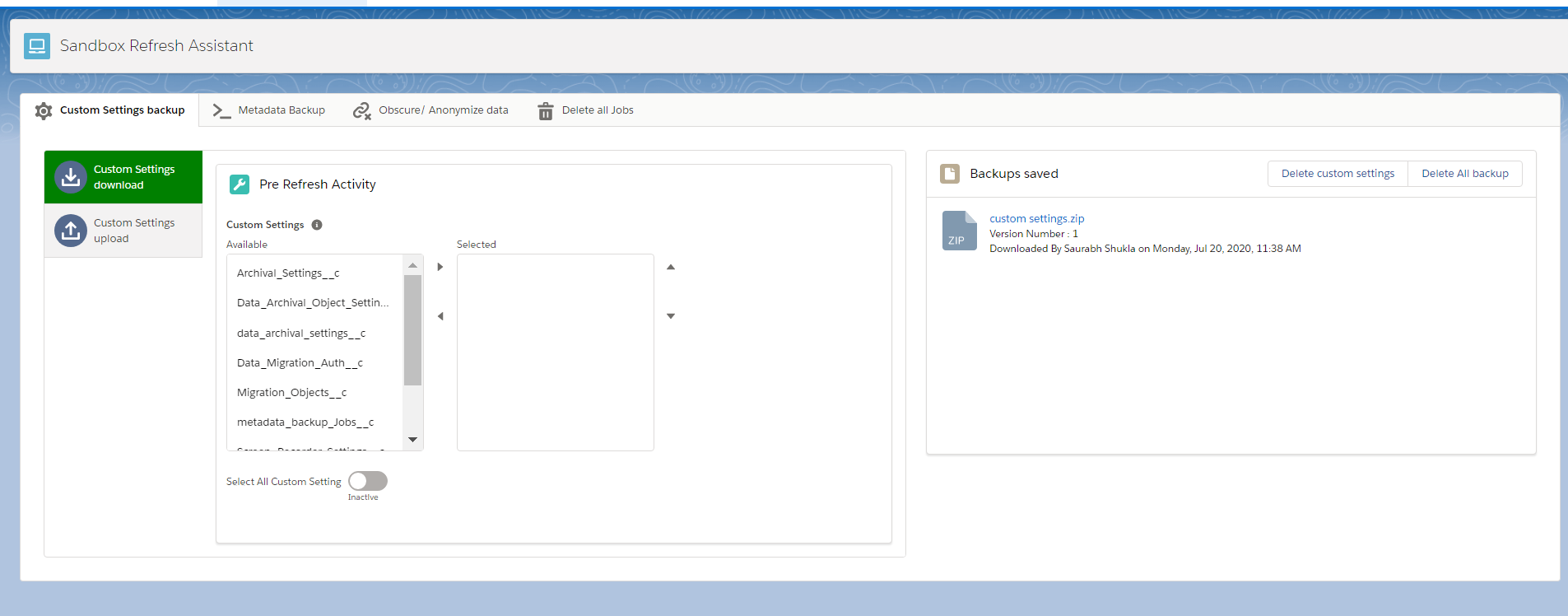
**Application Overview:**

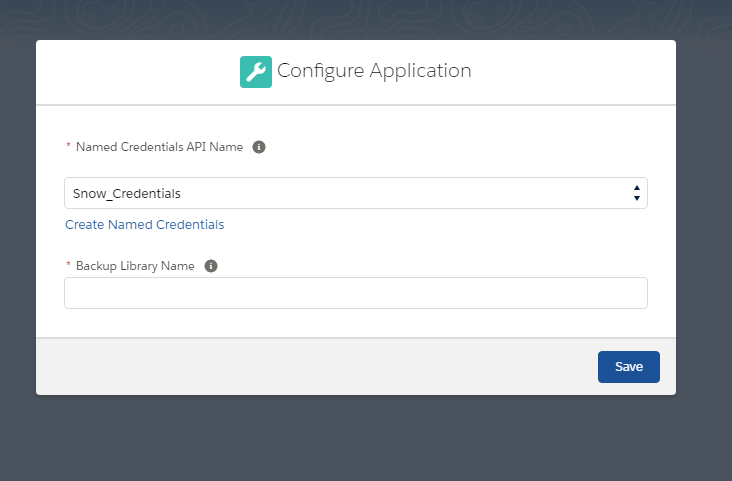
* When a sandbox refresh happens in salesforce, we will have to do some pre refresh and post refresh activities.
* These activities include taking backup of metadata, custom setting records, anonymizing customer data after refresh, data loading custom setting records and aborting all scheduled jobs.
* All these activities are time consuming and have a high risk of data loss if done manually

Ex: custom setting records need to be queried individually and stored in a local disk and data loaded back again from the local disk. This can take a significant amount of time for a support resource, also can have issues while comparing the existing data with the backup.



* We Built an application that allows an admin to take backup of all the custom setting records inside a zip file and push it into production where it can be safe until the refresh happens where it will come back to the sandbox.
* The admin can use this zip file to update all the custom setting records directly or can also perform a conflict with the existing settings and merge/ edit the backup before loading the data.
* This also includes a tab to take backup for all metadata and deploy it to the sandbox post refresh.
* A data anonymizing tool is included which will allow users to select the object and fields that contains personal information and fill those with junk. Upto 5 selections are saved as user preferences for future use.
* The application has a tab to abort all the running jobs in sandbox to avoid any automated field updates and emails to be triggered.

**Initial Configuration Components:**

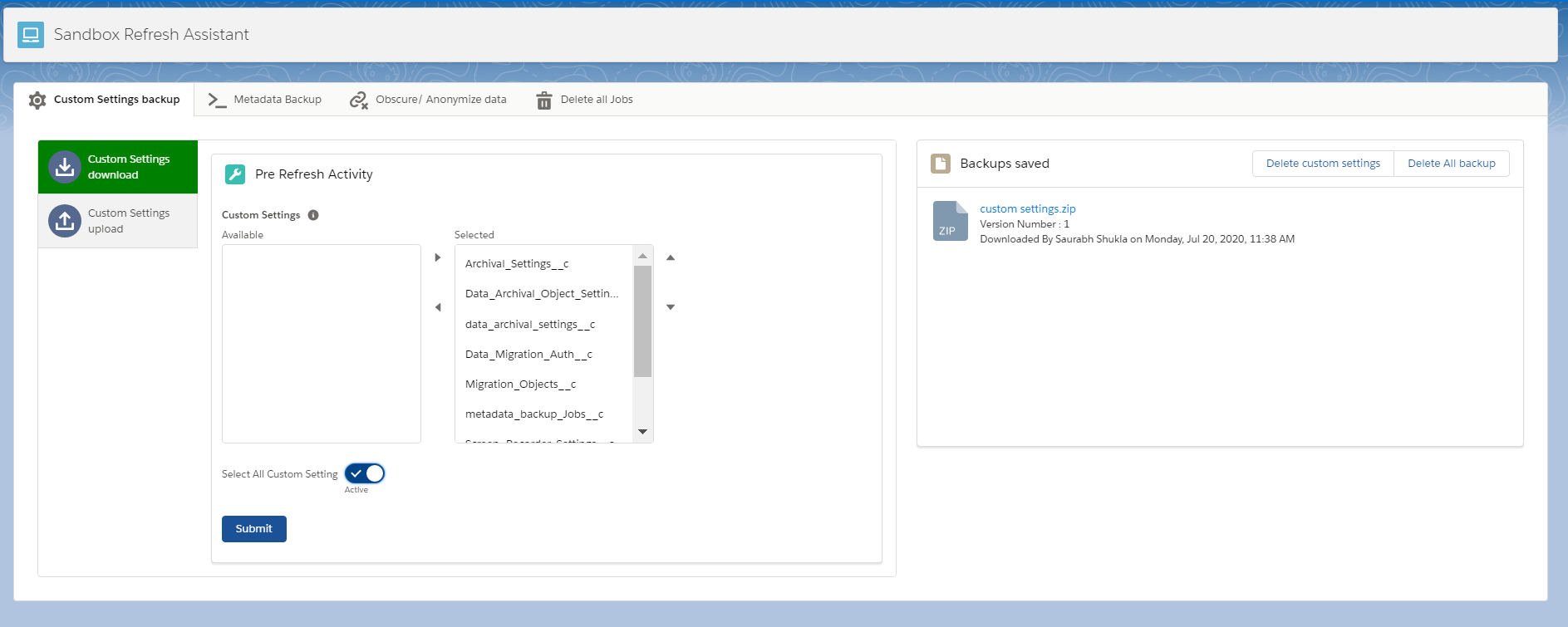


This is shown when the app is opened for the first time to configure the named credentials that will be used for connecting to target org and to specify the backup library API name from the target org.

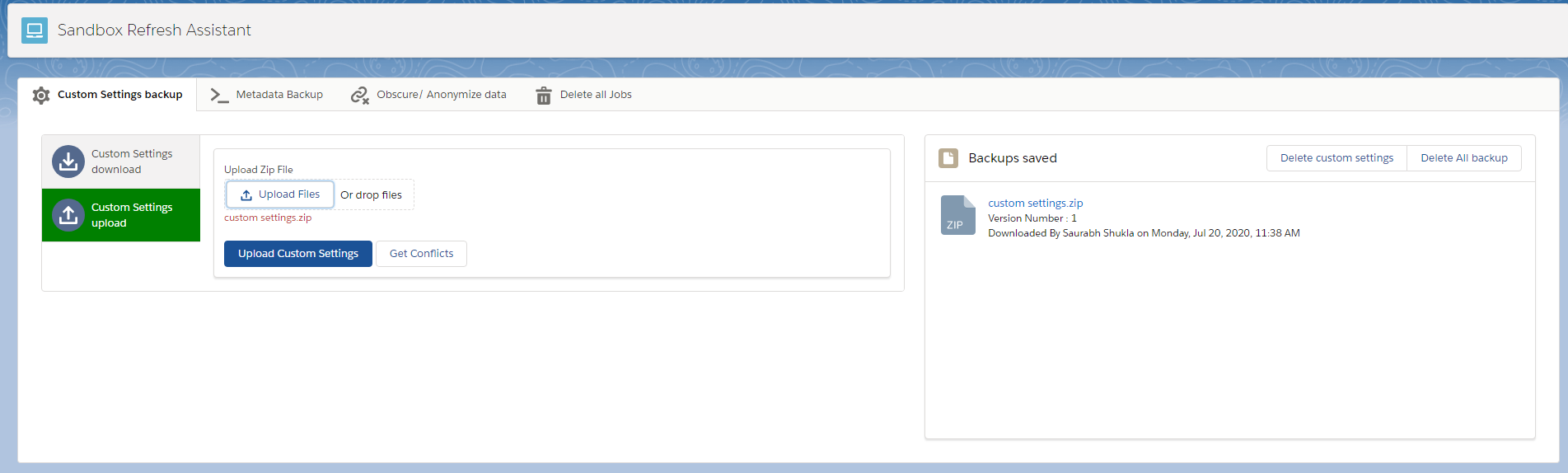
**Tab 1 – Custom Setting Backup**

In this **pre refresh activity**, users are allowed to take a backup of all custom settings records into a zip file containing the records of each custom setting object in a CSV. When users submit the form after selecting the custom setting objects the screen will show a message indicating that a backup job is in progress. At this point users can’t do anything in the ‘Custom settings download’ tab. Below is the example of zip file we get after taking backup in the ‘backup saved’ section which users can download.

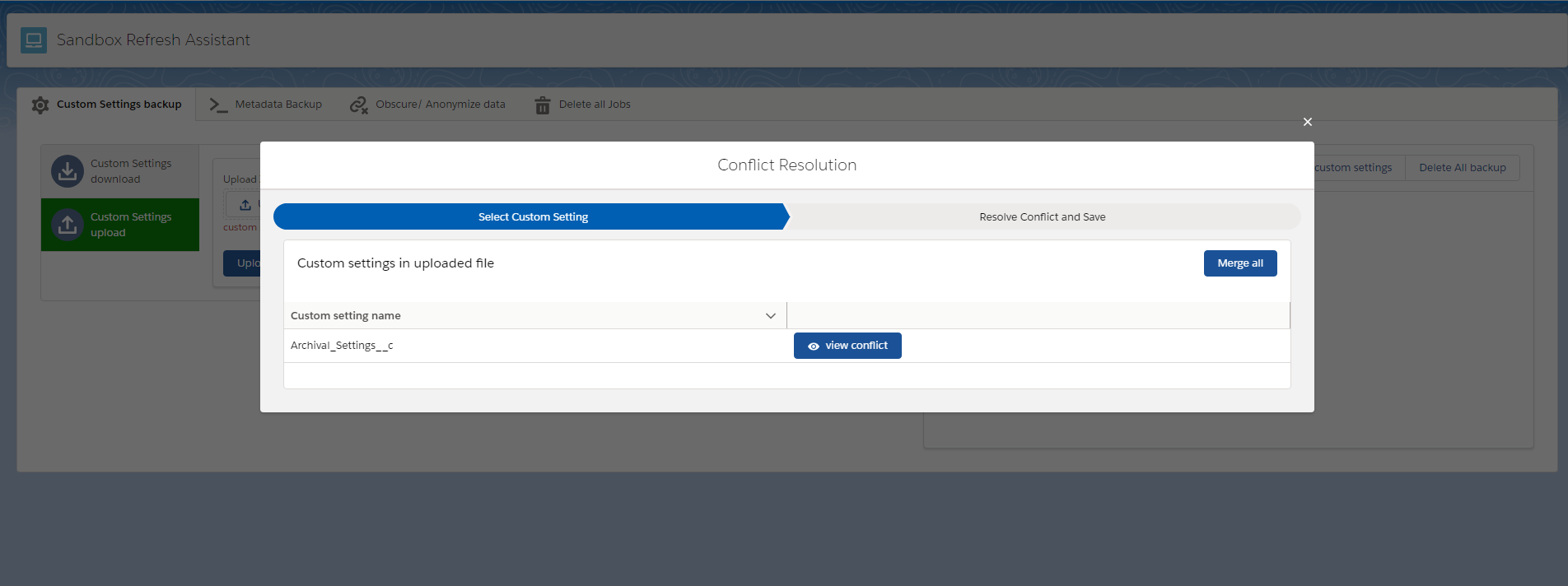




In this **post refresh activity**, tab name is ‘custom settings upload’ where users can upload the zip file and can perform a bulk update of all records in the custom settings inside the zip file.



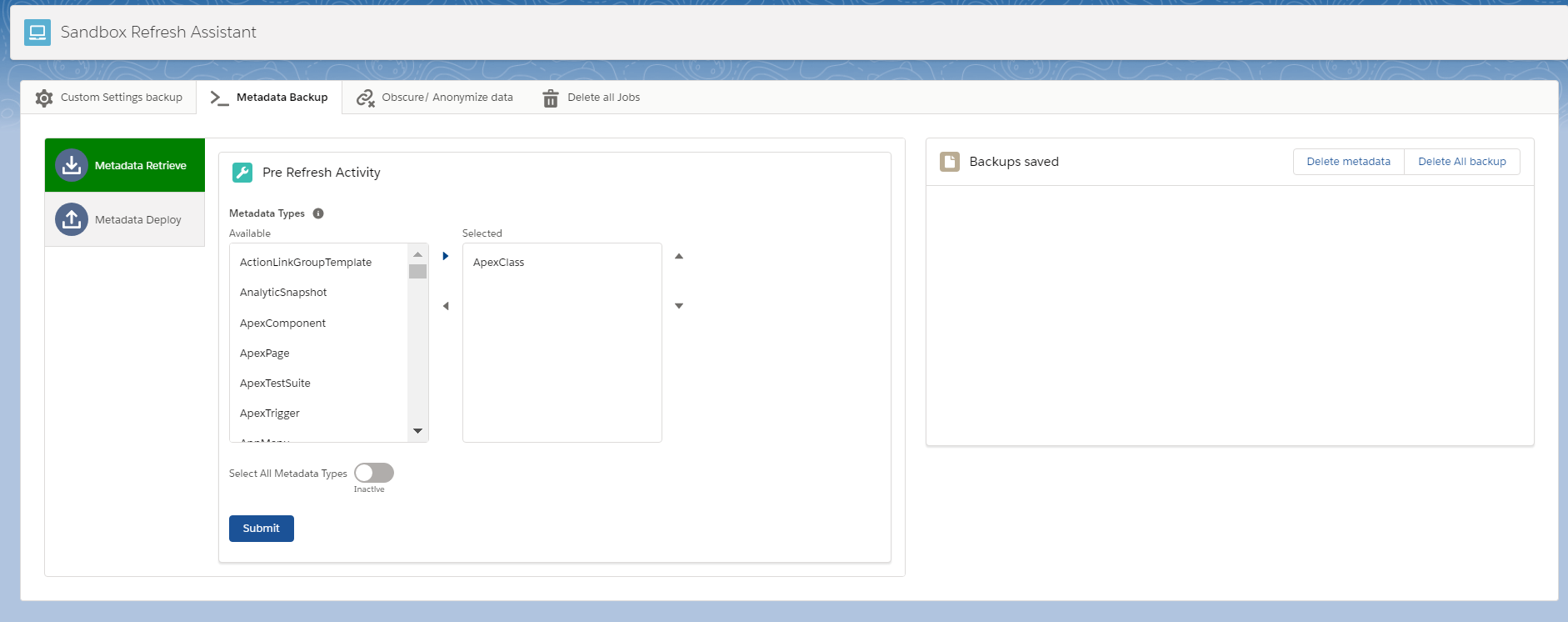
There is another option that allows users to perform a conflict resolution where users can open each csv file inside the zip and save the required records individually instead of updating everything.



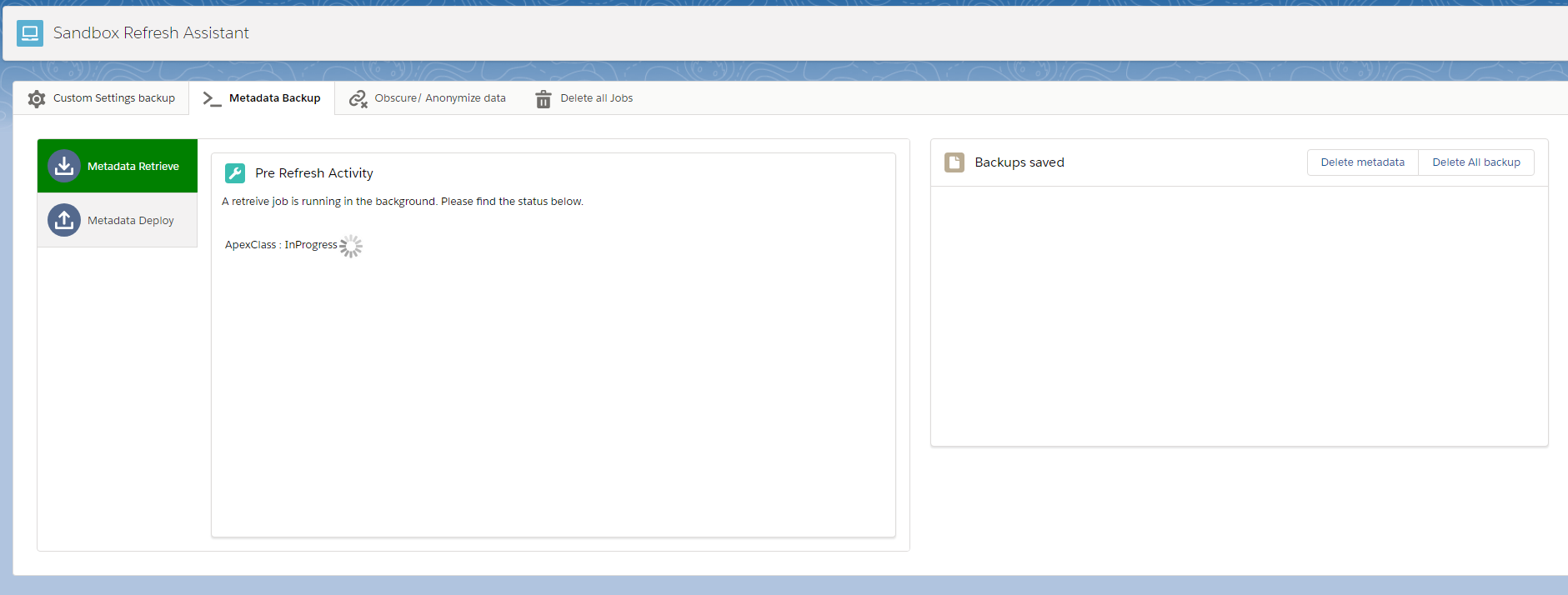
**Tab 2 – Metadata Backup**

In this **pre refresh activity**, users are allowed to take a backup of all the metadata from the org and store the backup in the target org in the form of a zip file in a content library. The backup files stored in target are shown in the backups saved section. This section has 2 vertical tabs which are for download and for deployment.

In the download tab users can select the metadata object to download and submit the job.

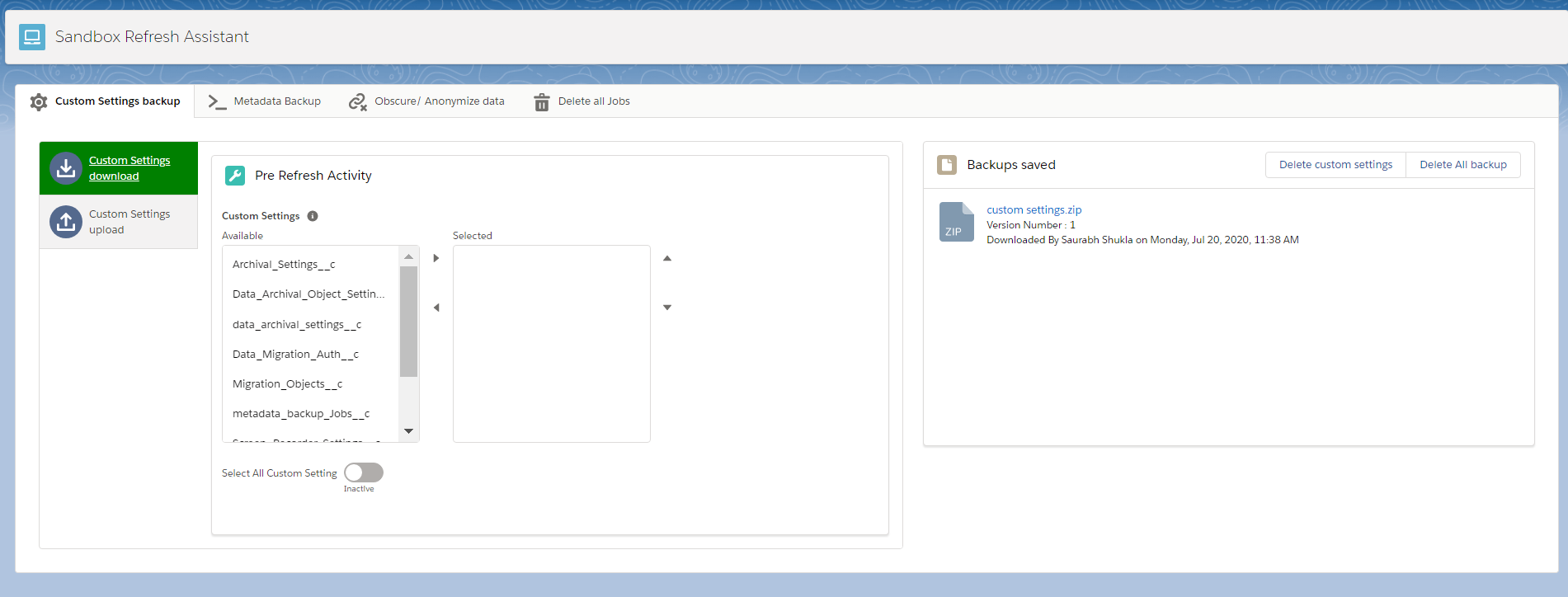


Once submitted we get the live status of retrieval as shown below.

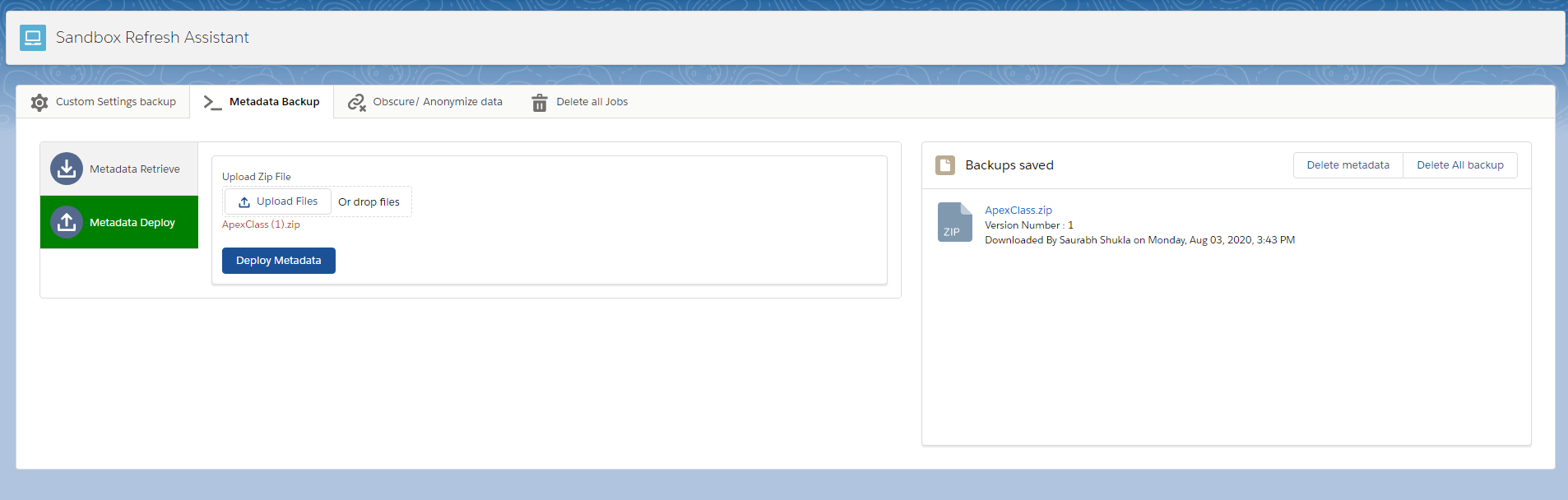


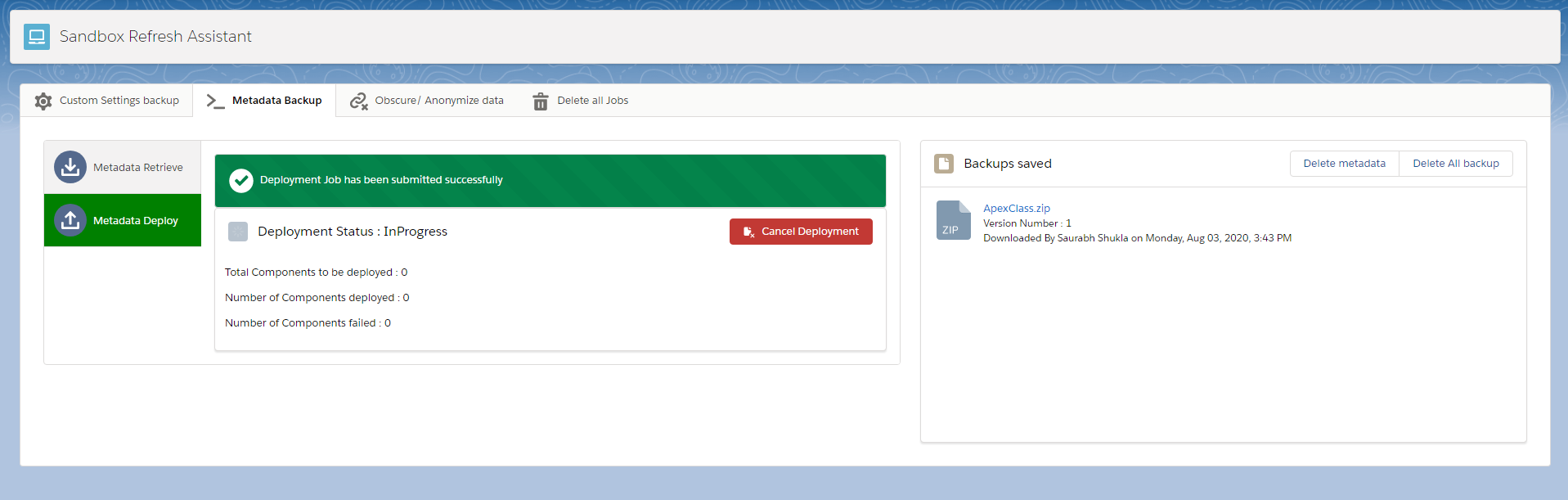
After the job is successful, we can see the zip file for ApexClass in the backups saved section. Unlike custom setting backup, the files are stored seperated for each metadata object.

Through this we can take backup for all metadata objects including custom labels and custom metadata records.



In this **post refresh activity**, we can deploy the components by uploading the zip file in the deploy metadata section. Which shows the live status of deployment and allows for deployment cancellation as well.

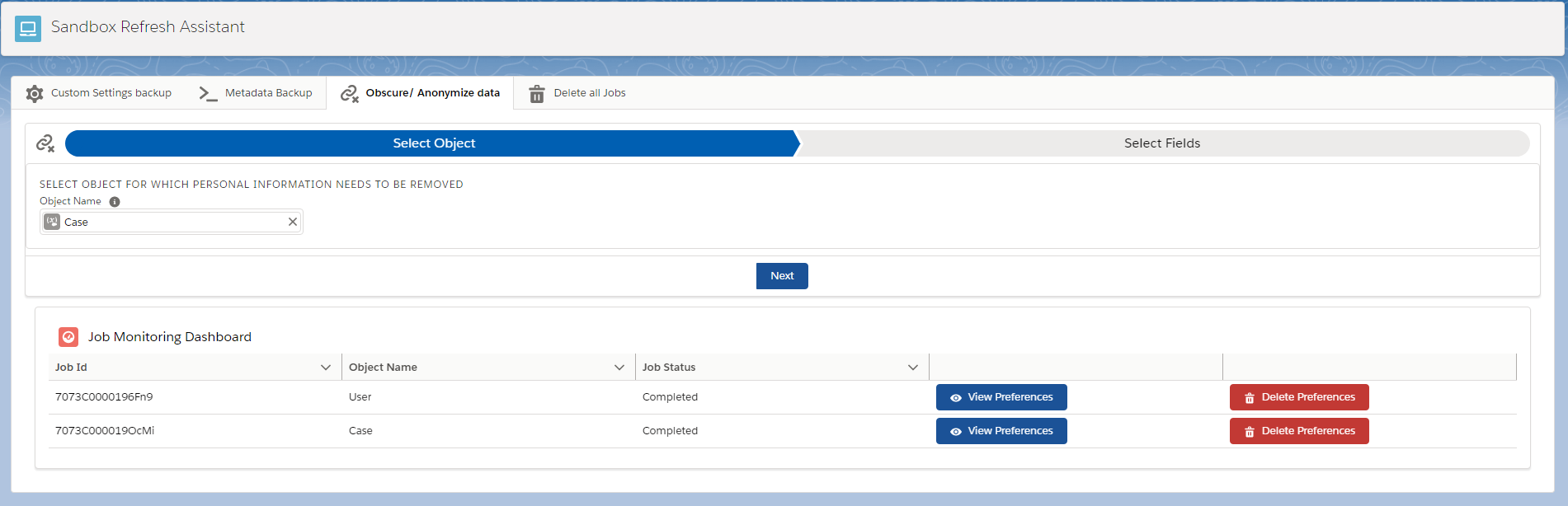




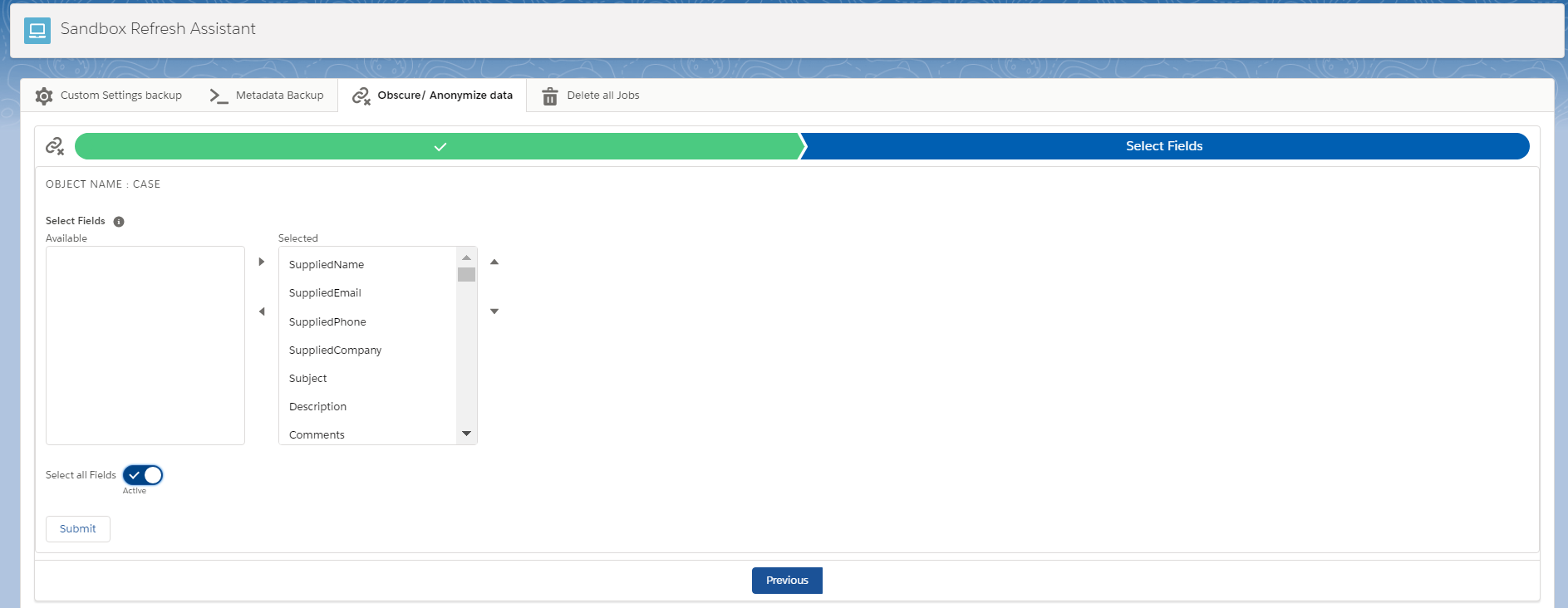
**Tab 3 – Obscure/ Anonymize data**

This is a **post refresh activity**, and allows users to obscure the customer related data from all sobject in the org.

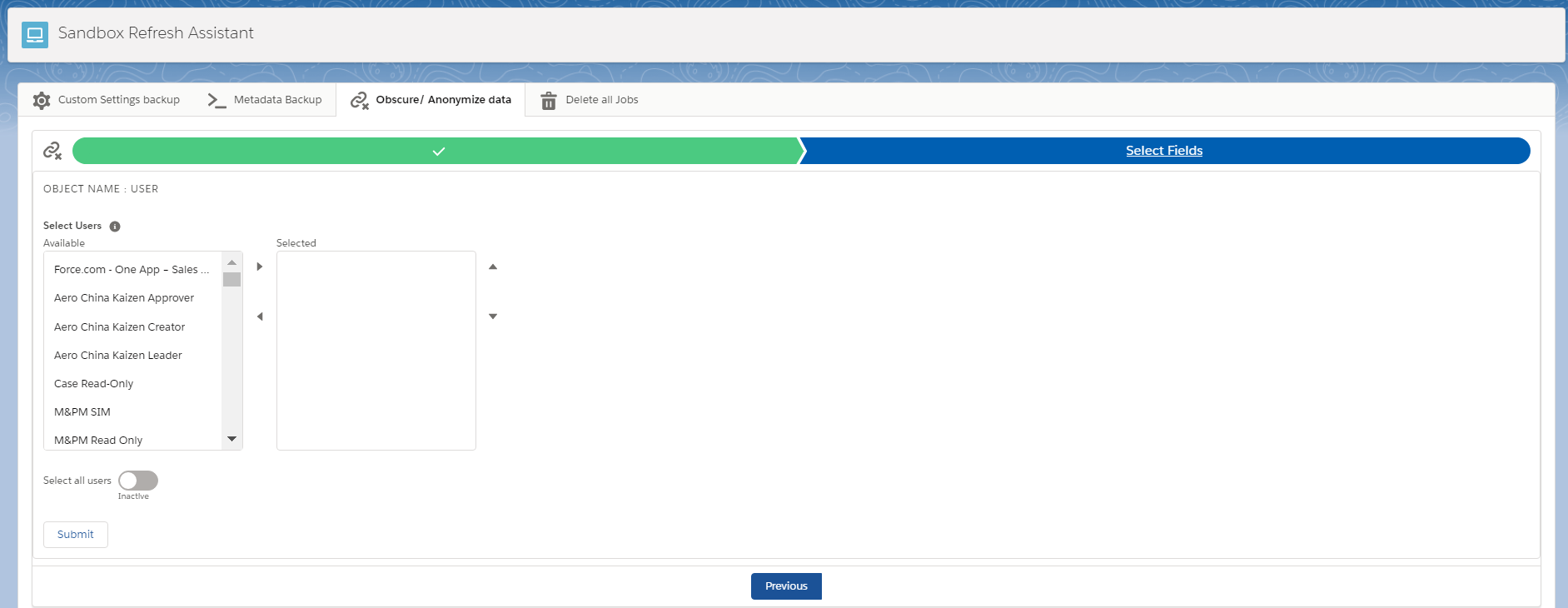
We first ask the users to select the object from which the records need to be obscured.



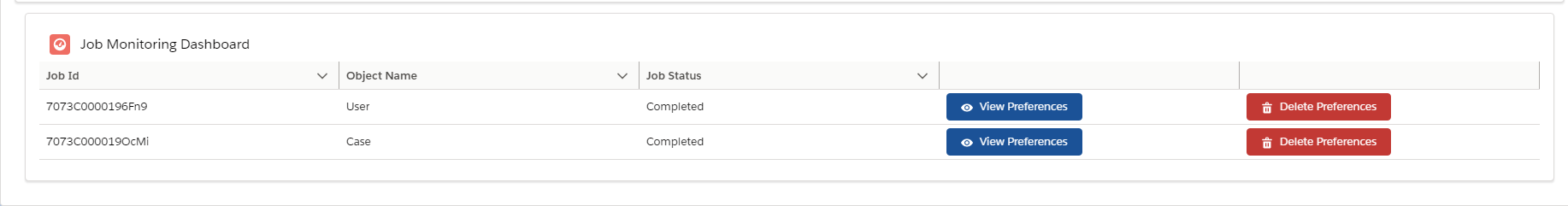
Then the user needs to select the fields where the data has to be obscured. Only text, email, number fields are allowed to be obscured.



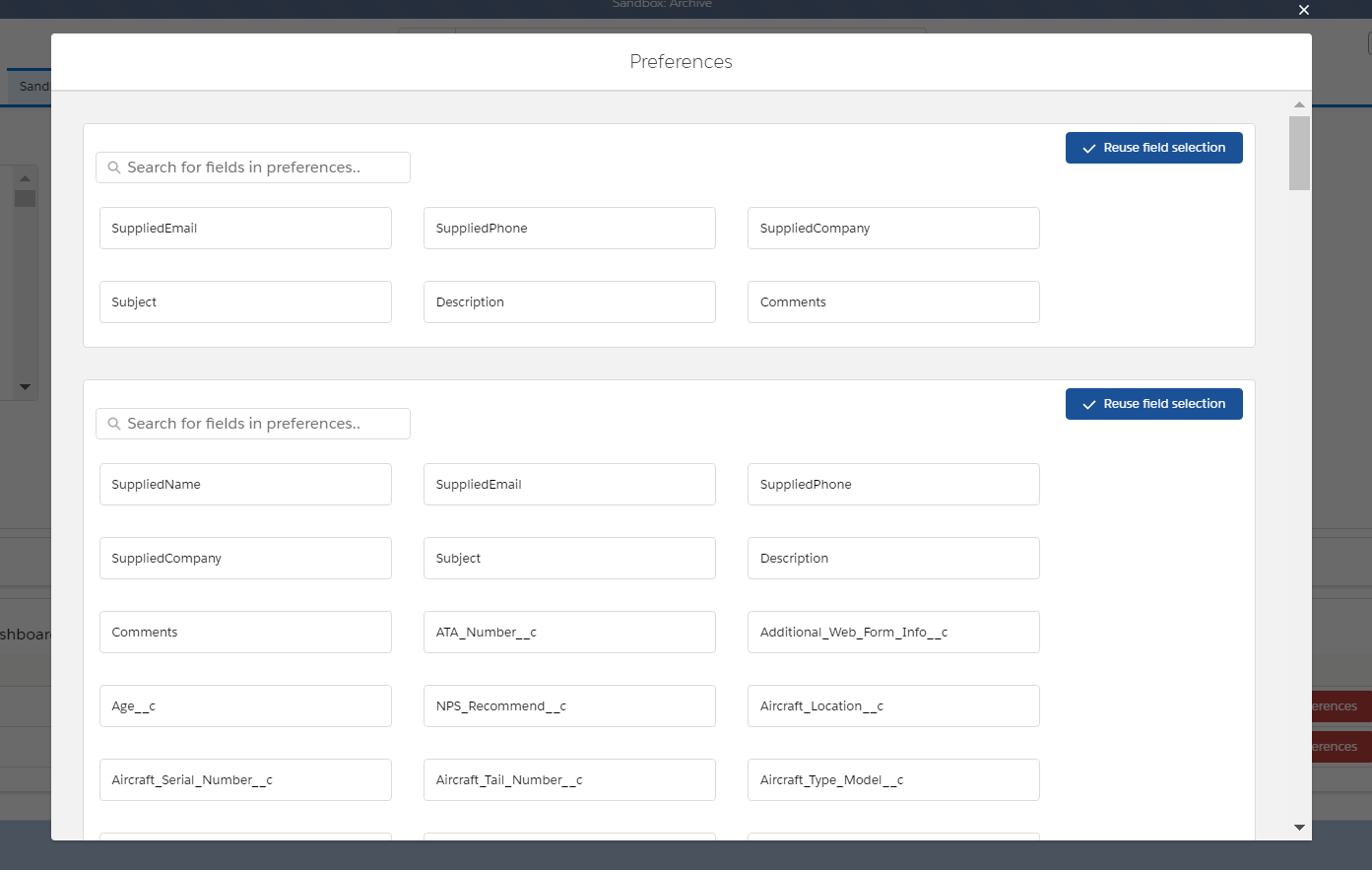
If the object selected is ‘**User**’ then the user has to select the profiles of the user records which need to be obscured.

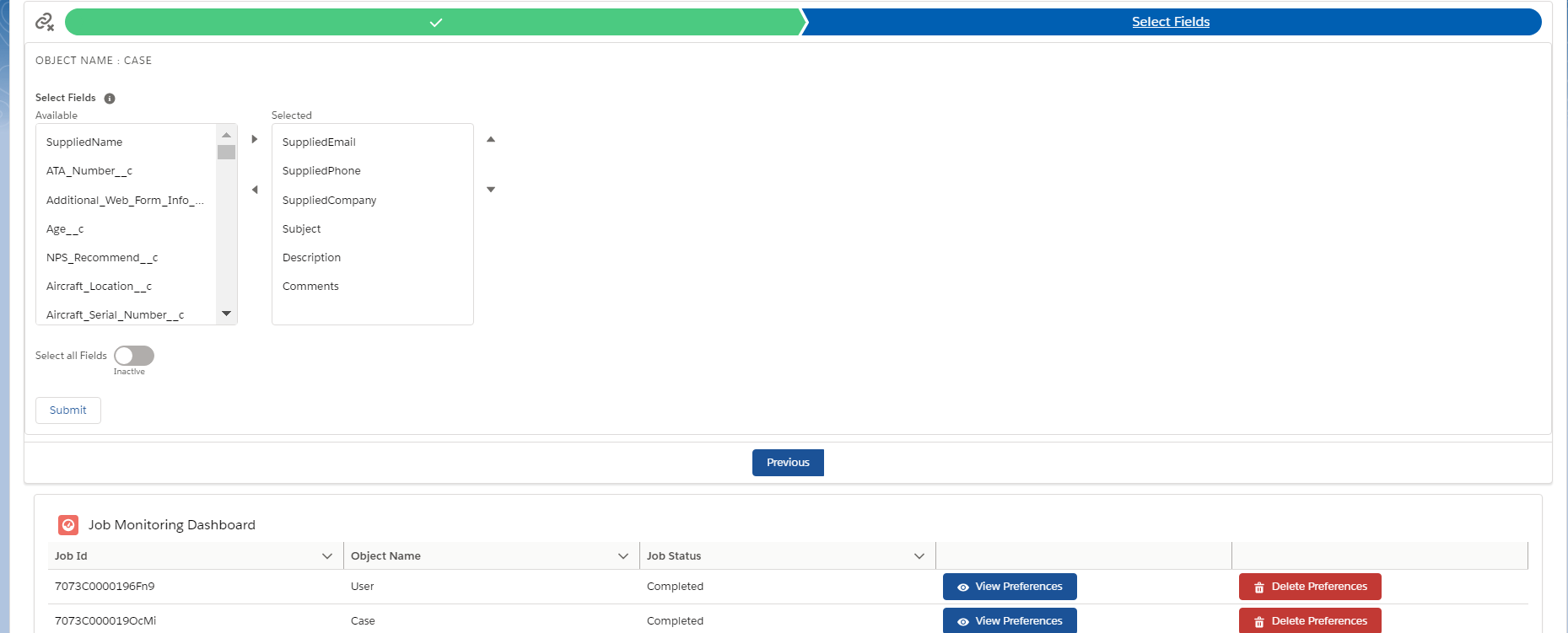


Once submitted, all the selected fields of the records of the selected object will be replaced with junk randomized values. Along with this there will be user preferences automatically created using the selection which will have the object and fields selected so that it can be reused in future. The job monitoring dashboard will show the job status along with the object names and job Id.



When users click on ‘View preferences’, there will be a modal box opened which will show all the field preferences for that object. Users can search for fields in the preferences and click on ‘Reuse Field selection’ button to auto fill the object and fields. Maximum of 5 preferences can be created for each object after which the tool asks the user to delete the preferences for that selected object using ‘Delete Preferences’ button.

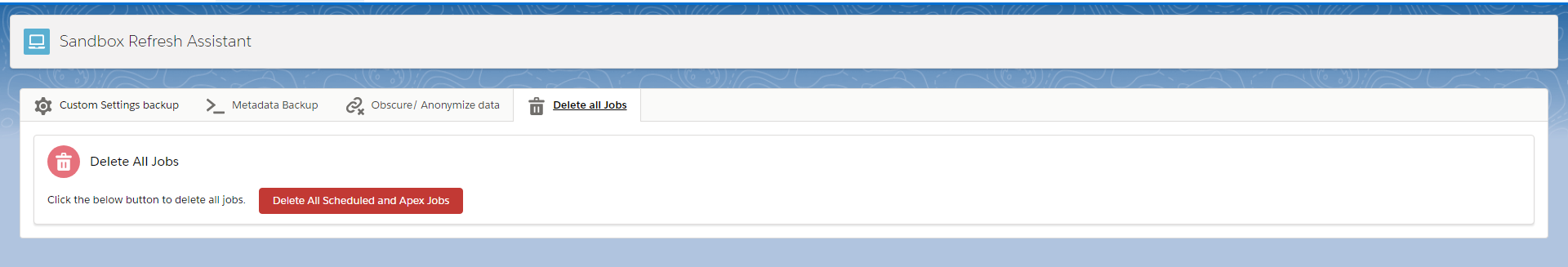




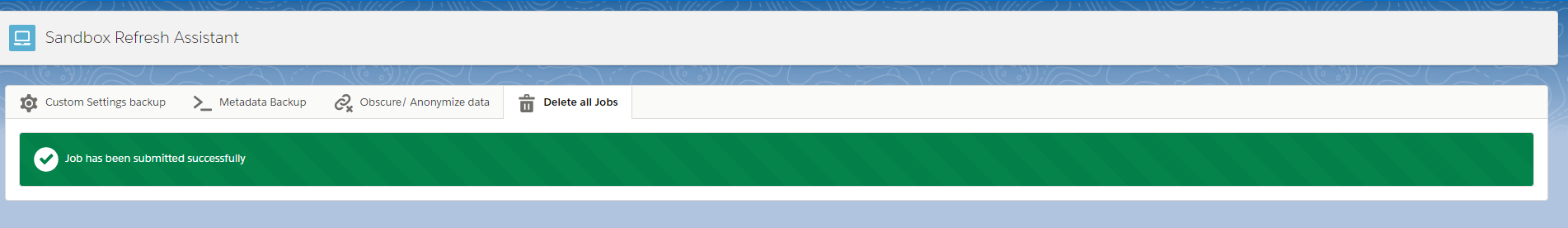
**Tab 4 – Delete all jobs**

This is a **post refresh activity** and this tab consists of a button that allows the user to delete all the scheduled jobs and apex jobs that are still running in the newly refreshed sandbox to avoid manual activity.

The button is hidden once clicked and a job will be submitted to delete all scheduled jobs.

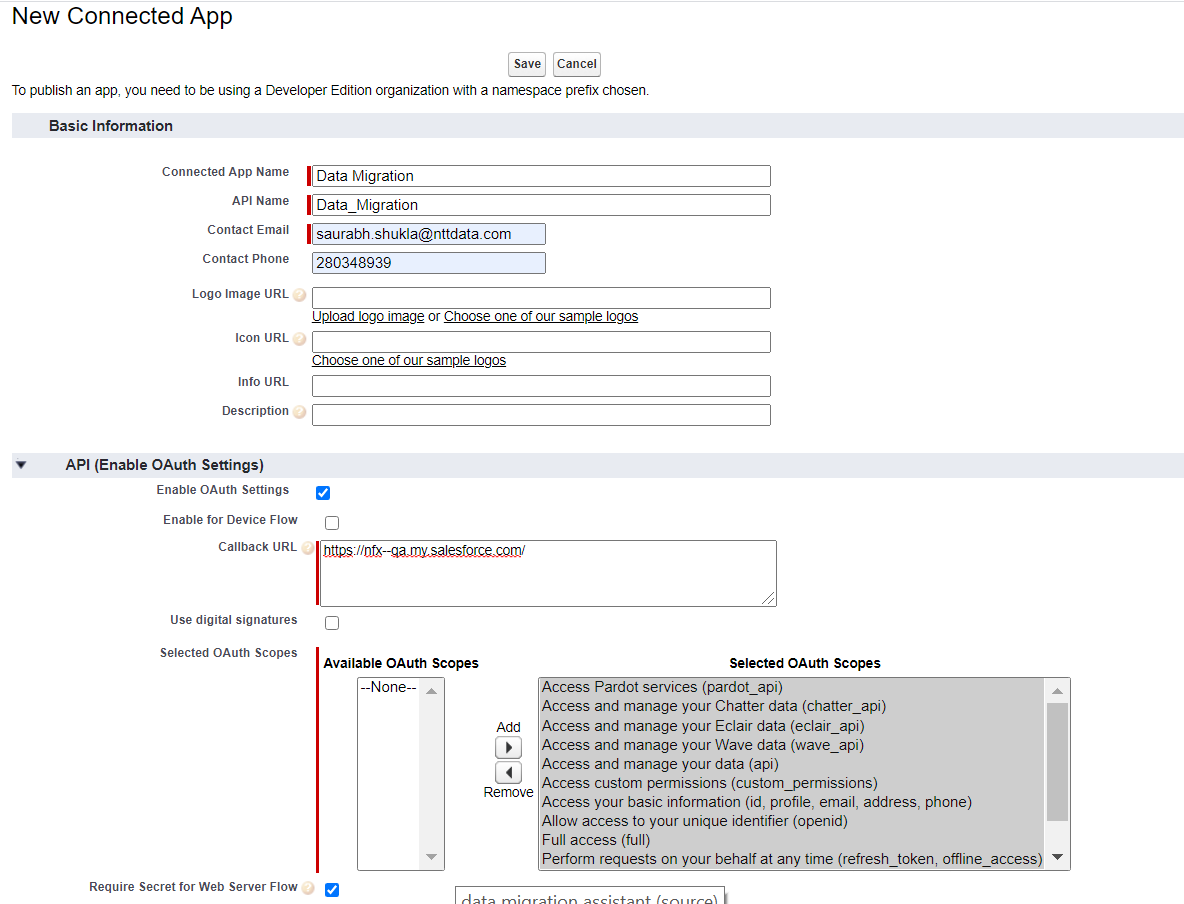


After submitting the job, we get the below message:

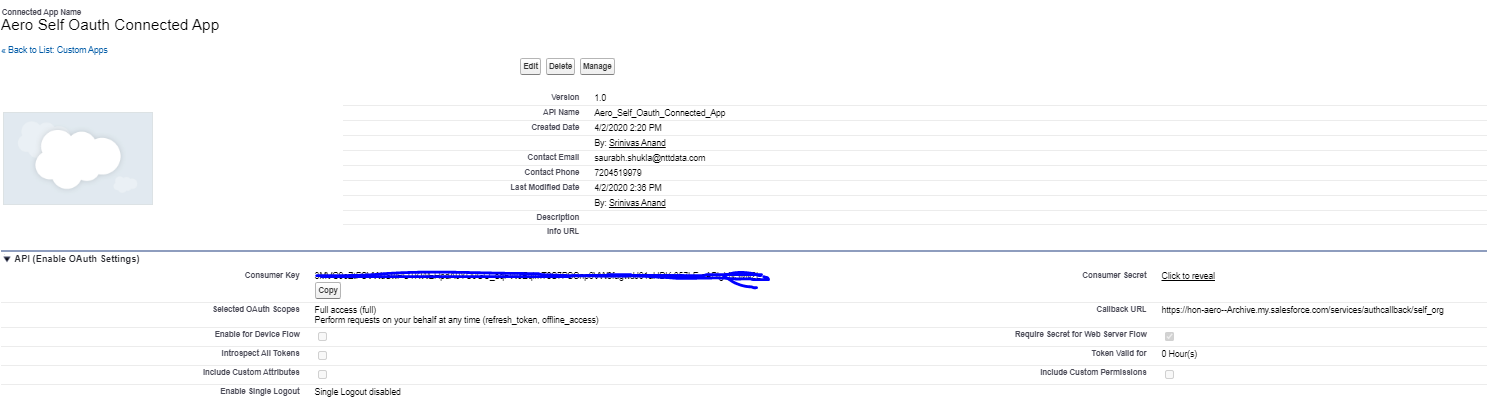


**Connected App setup:**

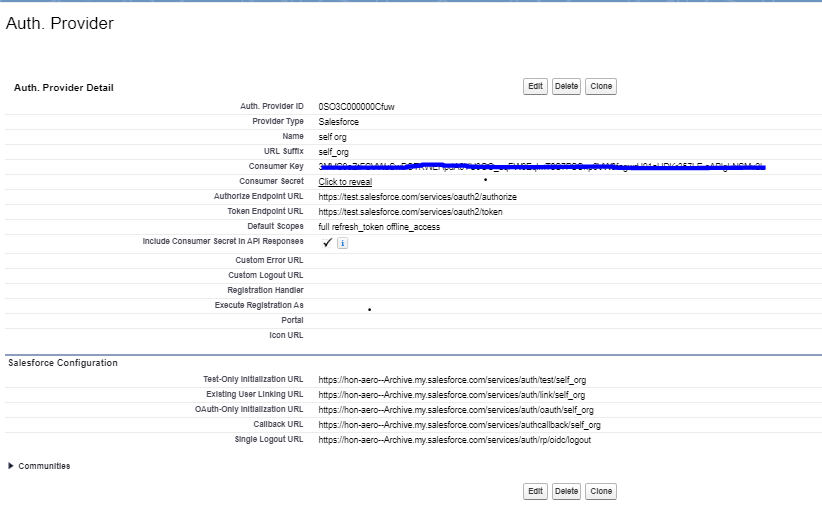
1. Create a connected app in the **target org** with oauth enabled and below settings. Callback url can be the source org for now and select all oauth scopes.



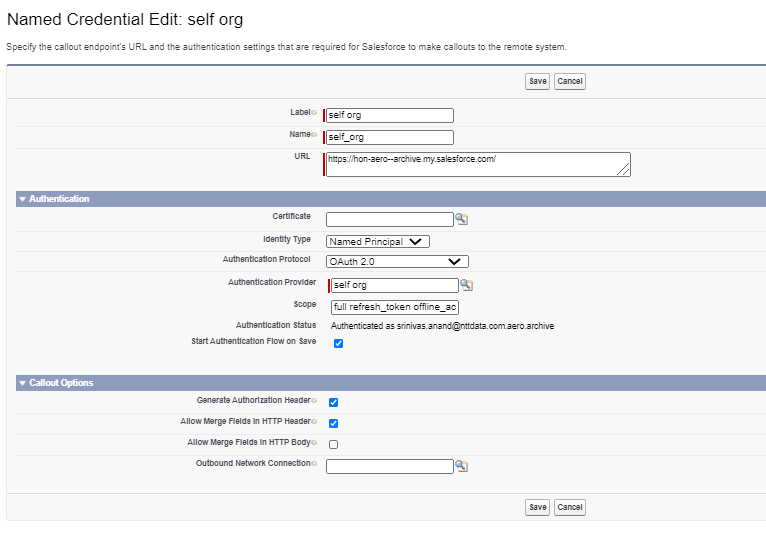
1. Save the app.



1. Create a auth. Provider in the **source org** with salesforce settings. Copy the client Id and client secret from the above created connected app and paste it in the auth provider.



1. Save the auth. Provider and you will find a callback url in the related list. Copy and paste the callback url in the connected app callback url.
2. Create a named credential in the **Source Org** use the Named principal authentication and oauth 2.0 option.
3. Lookup the auth provider that was created and use the same in the named credential.
4. Allow merge fields in the Http header, generate authorization header, start authentication flow on save should be checked.



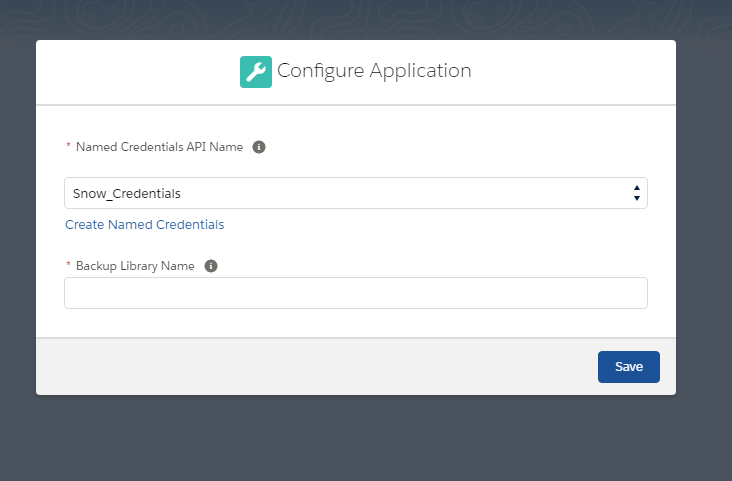
1. Authorize the named credential on Save.

**Post Installation Steps Target Org:**

1. Go to the custom label **backup\_library\_size\_limit** and provide the maximum size in Bytes that should be dedicated for storing backup files from all orgs.
2. Create a library to be used as storage for backup files and give it a unique name.

**Post Installation Steps Source Org:**

1. Open the application ‘Sandbox Refresh Assistant and you will see the popup below:



1. It will show the list of named credentials in your org. Select the one that was created previously.
2. Enter the backup library API name from the target org and Save.