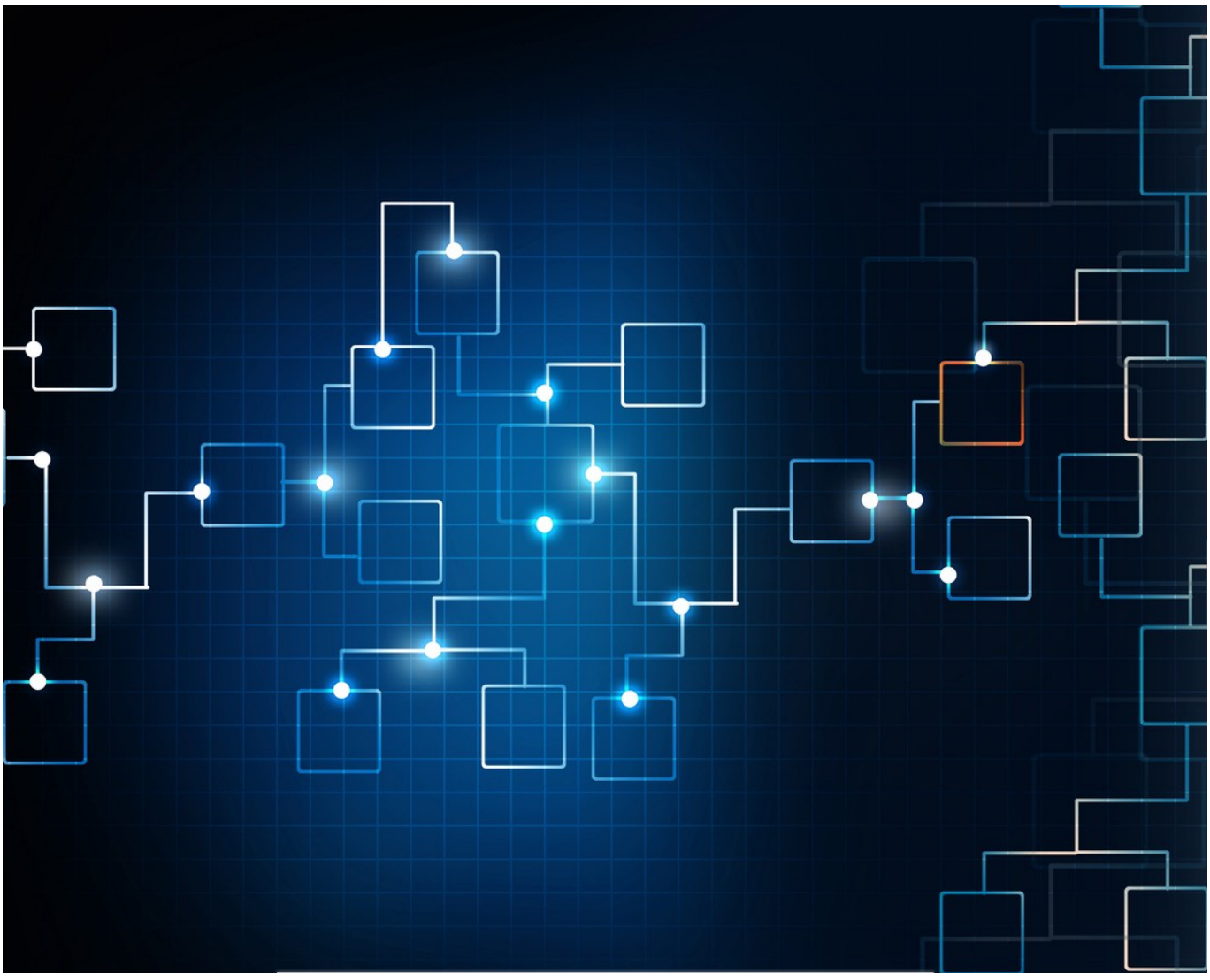
salesforce architects

Following 2.1K Followers About

Multi-Org Visibility for Work.com

Dan Mehlman Nov 11, 2020 · 11 min read



Disclaimer: This application in this blog post may enable access to potentially sensitive or regulated data, including in the form of self-reported employee wellness data. Customers and users are responsible for ensuring that their use of this application to access or process that information complies with all applicable laws and regulations, including any applicable employee data protection laws and regulations.

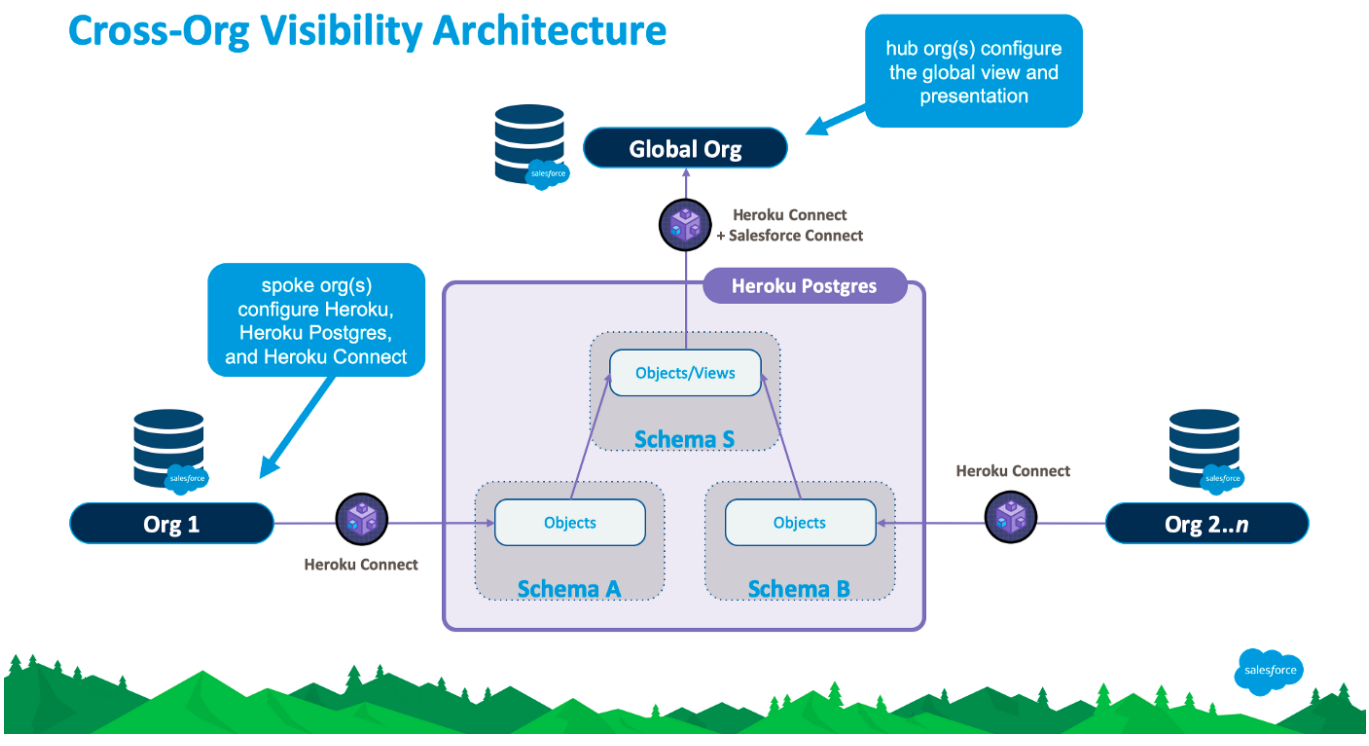
If I were to describe Heroku in a single sentence, I'd say "Heroku is a set of capabilities that makes it easy to build solutions." And while is this true, it's hardly comprehensive. Elsewhere in the Heroku blog, my colleagues have discussed features such as Heroku Connect, which, simply put, is the "easy button" for exposing Salesforce data across your enterprise while providing relief for API management and Salesforce governor limits. Heroku Connect is a prime example of what Heroku is beyond a set of capabilities that makes it easy to *build* solutions: Heroku is also a platform that makes it easy to *extend* solutions.

In this post, we'll address an important need of our customers to provide data visibility across their multiple Salesforce organizations using Heroku capabilities. Why is this relevant? Any number of Salesforce customers have multiple Salesforce organizations for reasons such as acquisitions — where one Salesforce customer acquired another Salesforce customer — or data locality concerns as countries have addressed growing privacy concerns by requiring the storage of certain data with their country's borders. And where multiple environments do exist, somewhere within the business there will be the need for an enterprise-level view of the company's data.

Ironically, while our customers have been looking for simple solutions that enable the data visibility across organizations, Heroku's Salesforce organization itself was an example. Heroku was a Salesforce customer at the time of the acquisition and we maintained our Salesforce environment, which was customized for how we ran our business. After the acquisition, there was the need to roll up sales and customer data into the master Salesforce environment. Over the years, a number of customizations were put in place to enable visibility between Salesforce's Salesforce organization and Heroku's Salesforce organization, which enabled the sales and business development teams to effectively do their jobs. The Salesforce multi-org solution described in this post could have simplified this integration, had the solution existed at the time.

Where we are today

Perhaps no recent challenge has emphasized the need for cross-org visibility and reporting as readily as Work.com. Maintaining security and privacy around employee wellness data is of paramount importance, and many countries have standards requiring data locality. At the same time, organizations across the globe are faced with the challenge of monitoring the health of a globally distributed workforce.



In the remainder of [illegible] oku Connect solution that will agg[illegible] geographies and enable reporting up[illegible], please ensure access to this data remains restricted only to the individuals who require access to it.

One final word before we get started

While the example use case is specific to Work.com, this multi-org solution is generic and applicable to all Salesforce organizations. We chose the Employee and EmployeeCrisisAssessment objects for this demo because they're less likely to be customized than objects like Account or Opportunity would be, and therefore are easier to work with from the perspective of a generic demo. Using this solution, you can create aggregated datasets from any Salesforce organizations based on any Salesforce objects, and easily virtualize your data into any Salesforce instances to take advantage of features such as the drag-and-drop report builder, list views, and other Salesforce functionality.

Please contact your Salesforce Account Executive if you have any questions, or wish to be put in contact with one of our implementation partners.

Getting started

To successfully complete this solution, you will require:

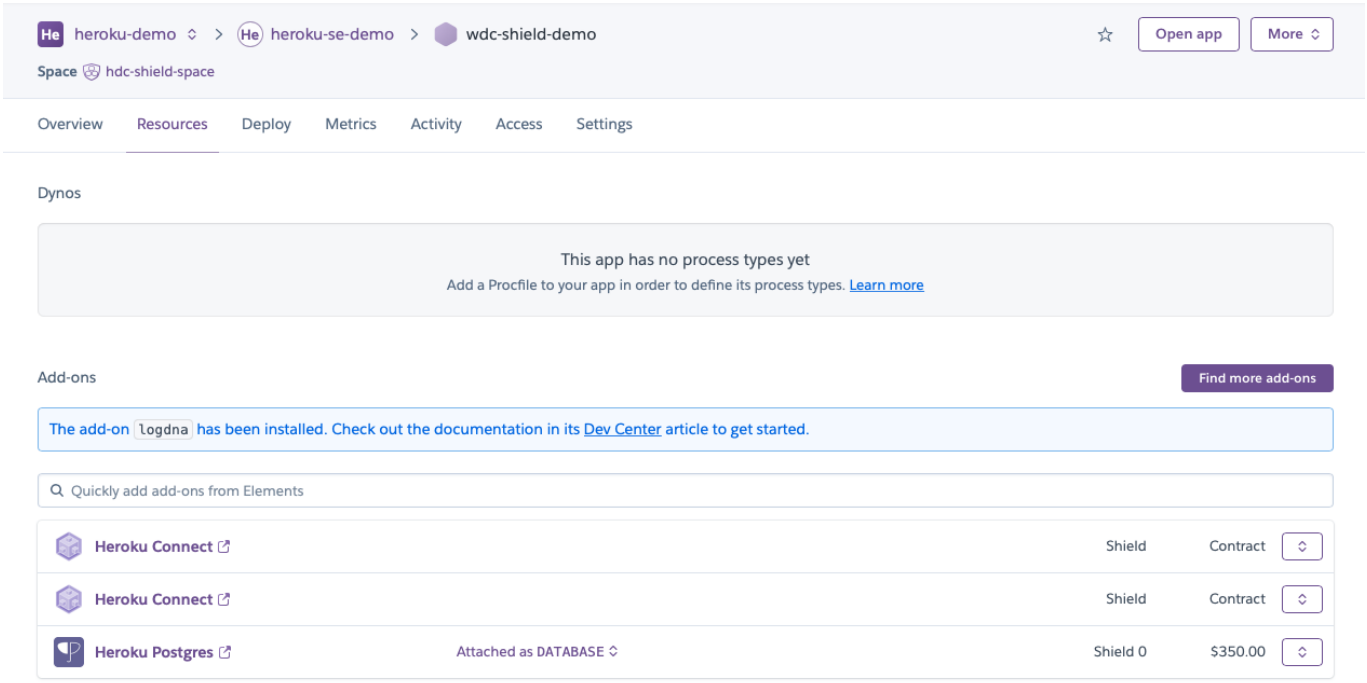
1. Access to a Heroku account. If you do not have an account and wish to sign up for a free developer environment, you can do so here.
2. Administrative-level access within the Salesforce organizations containing the employee wellness data.
3. Unfortunately Work.com trial organizations are not yet available. When they are, we will update this post appropriately. In the meantime, as an alternative, you can complete this exercise using the Account and Contact objects. **UPDATE: Work.com trial orgs are available here.**
4. You can create two or more scratch organizations to simulate your data producing organizations. See the App Development with Salesforce DX Trailhead module for details on how to create a scratch organization.

5. Administrative- ; consuming the aggregated em| rce Connect will also be necessa. y within this organization. You can also use a scratch organization for this purpose.

Step 1: Provision the Heroku app

The first step is to provision a Heroku app from the Heroku Dashboard, and include the Heroku Connect and Heroku Postgres add-ons.

Heroku Connect creates a connection between one Salesforce instance and a Heroku Postgres database. We require multiple Heroku Connect instances for this demo because we're pulling data from multiple Salesforce organizations, but we'll sync our data to a single database.



In this screenshot, you can see we've created a Heroku app and provisioned a Heroku Postgres database and two instances of Heroku Connect. You can create your own Heroku app using one of the links in step 1a below, or alternatively, create the Heroku app and provision the add-ons via the Heroku dashboard or CLI. Regardless of how you create your app, it should be similar to the one shown here, although the number of Heroku Connect instances can vary, depending on the number of Salesforce orgs you're drawing data from. The demo pulls data from two Salesforce organizations, so it has two instances of Heroku Connect. The links in step 1a also provision two Heroku

Connect instances, ing step is complete.

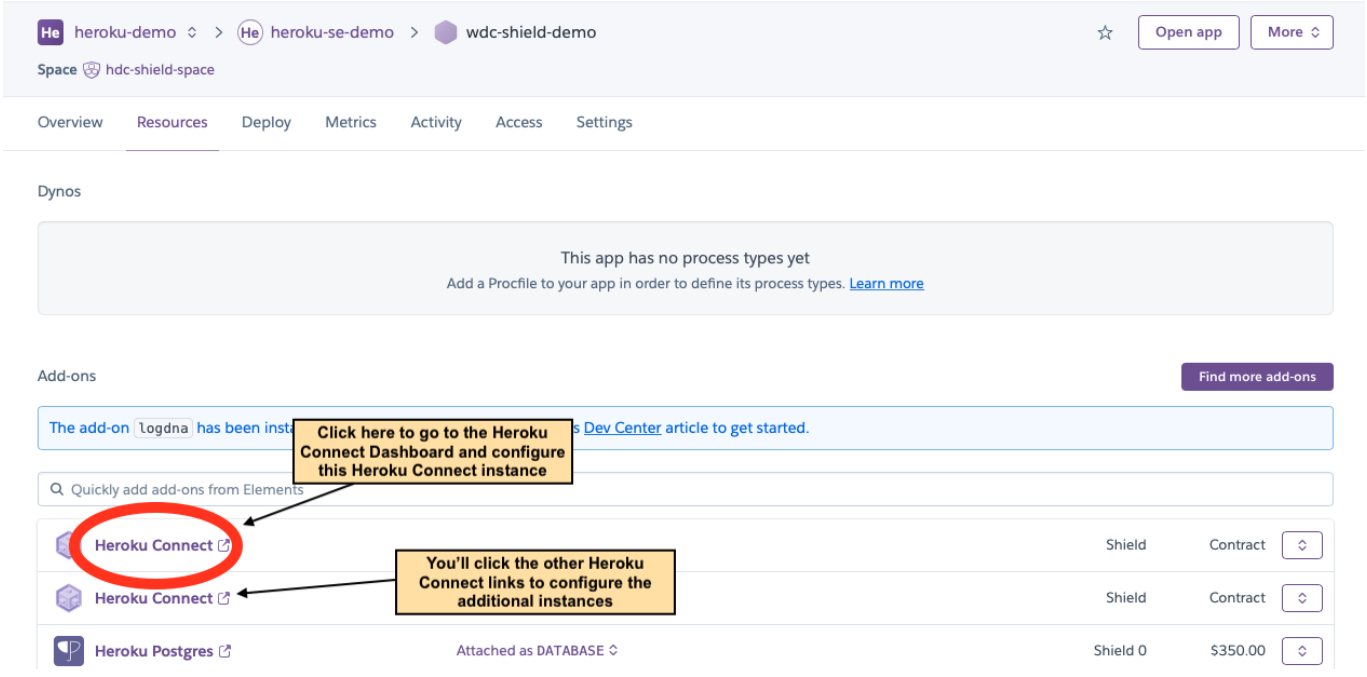
In the example above, the app is deployed within a Shield Private Space using Shield tiers for Heroku Postgres and Heroku Connect. While not required, these high-compliance resources provide the highest level of security controls and provide auditing capabilities. As you go through this exercise, the use of a Heroku Private Space, or a Heroku Shield Private Space, is strongly recommended but not required.

Step 1a: An optional shortcut

To automatically deploy a Heroku app matching the above configuration within the Common Runtime, use this template. Alternatively, if you wish to deploy the same app within a Shield Private Space you can use this template. Please note that if you are deploying into a Heroku Enterprise Team or other shared environment, appropriate access controls should be used to ensure that access is limited to the Postgres database when production data is involved.

The above deployment setups will also provision a single Heroku dyno when the app is deployed to an Enterprise Team or a Shield Private Space. You can reduce the dyno count to zero after the deployment has completed because dynos are not needed for this exercise.

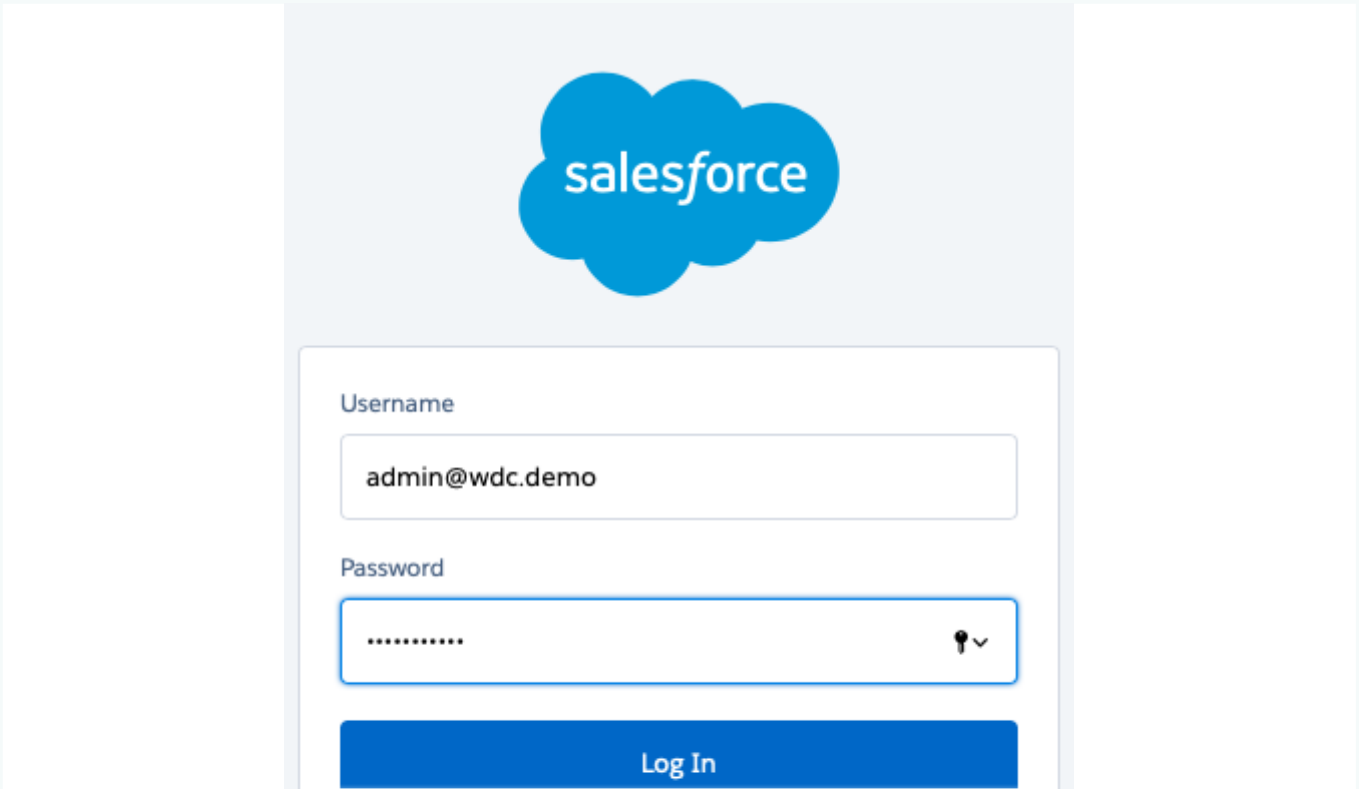
Step 2: Configure the Heroku Connect instances

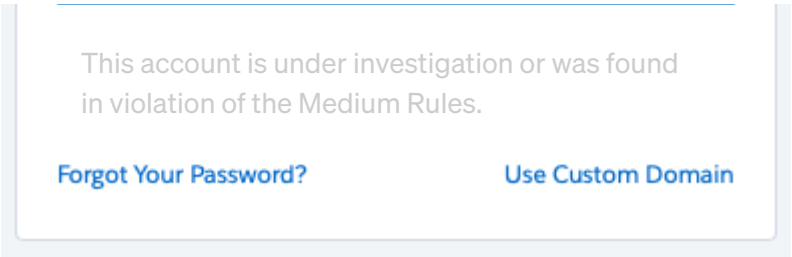


In this step, we're ab ne connections between our Heroku Postgres database and our Salesforce instances. We'll do this one instance at a time and repeat the process for each of the Heroku Connect instances, specifying a different Salesforce organization each time. It will be easier to complete steps 2 and 3 of this guide for one of the Heroku Connect instances before repeating both steps for each subsequent instance. To stress this point: *complete steps 2 and 3 for one geography, and then repeat as necessary for each additional geography.*

Because we're syncing all of our Salesforce data to a single Heroku Postgres database, we will need to use unique schemas for each of our Heroku Connect instances. In the example below, the schema name reflects the geography where the Salesforce data resides.

As a best practice, this example also adheres to the principle of least privilege. The user specified as the administrative user has been created with "Read All" access to the objects that we're syncing as part of this exercise (either Employee and EmployeeCrisisAssessment or Account and Contact). Access to other data is restricted.

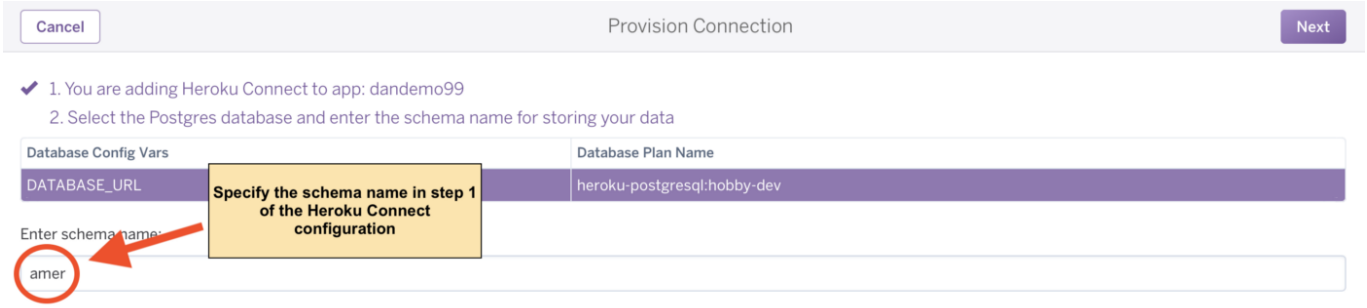




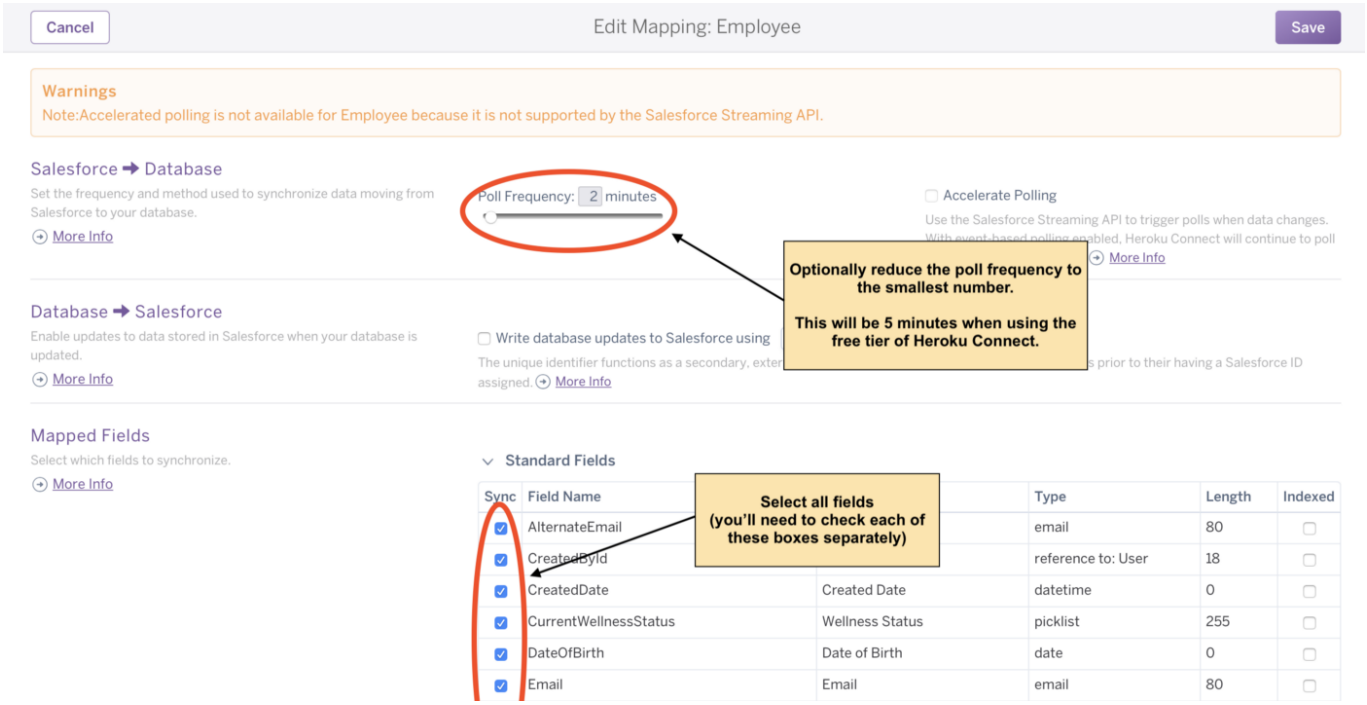
You will need to provide access credentials to your data-producing organizations — either the Work.com instances if you have them, or the scratch organizations you created earlier.

You can obtain the password for your scratch organizations via the CLI:

```
sfdx force:user:display -u <username provided by the Scratch Org>
```

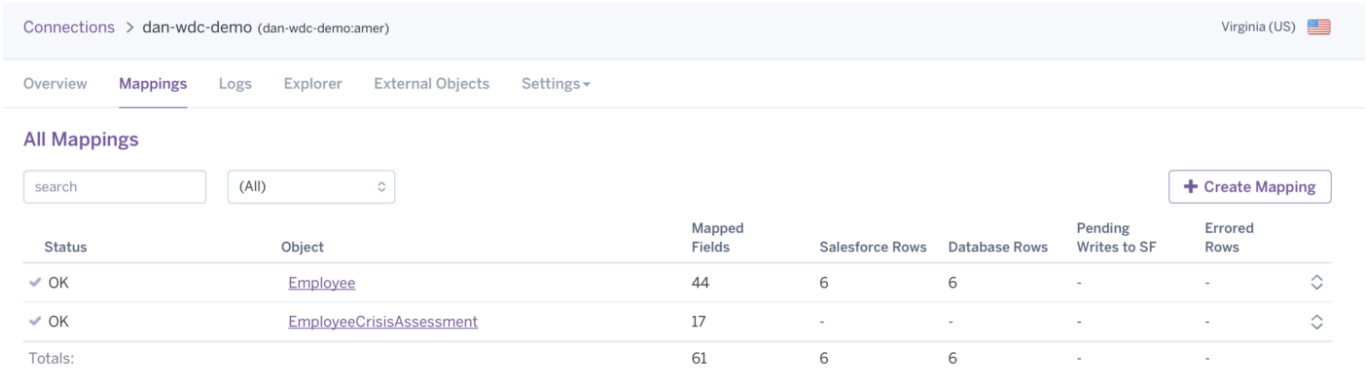


Step 3: Map objects



From the Heroku Cc ate Mapping. You'll
want to select the E k your settings
against this screens.not and then click "Save."

Next you'll want to do the same thing for the EmployeeCrisisAssessment object.



When you're done, your screen will again look similar to this screenshot, although the numbers in the Salesforce Rows and Database Rows will vary since we're looking at the record count of data with your Salesforce organization and row count within the Heroku Postgres table.

Reminder: Complete steps 2 and 3 for each of your Salesforce instances that will be providing data to the aggregated view. In this example, we have Salesforce organizations representing the AMER and EMEA geographies, so these steps need to be completed twice.

Step 4: Create the aggregated data view

In this step we're accessing Heroku Postgres directly. We have pgweb running on a dyno within the Shield Private Space, which provides access to the data tier of the Heroku app directly (of note, you may want to dial the dynos down to zero when you're not using this app to avoid unnecessary expenses). If you're building this demo in the multitenant environment (which can be done, but is not recommended due to the privacy concerns surrounding the data), you can use the command `heroku pg:psql --app <app name>` from the Heroku CLI and access the database from the command line if you have psql installed locally on your machine.

Now, we need to cre s simple!). There are a couple of conside se this data at your OData endpoint (or more specifically, as a Heroku External Object).

- Heroku Connect requires External Objects have a column named 'id'
- The contents of the 'id' column must be unique
- We're using a 'geo' column to keep track of where the data originated

You can easily meet these requirements as you create the view. The syntax is as follows:

```
CREATE OR REPLACE VIEW global_assessment as
SELECT employeecrisisassessment.sfid AS id,
employeecrisisassessment.assessment,
employeecrisisassessment.assessmentdate,
employee.lastname, employee.firstname,
'geo designation' AS geo
FROM <schema name>.employeecrisisassessment, <schema name>.employee
WHERE employeecrisisassessment.employeeid = employee.sfid
UNION [repeated select for each of our other schemas];
```

heroku_aggregated_data_view.sql hosted with ❤ by GitHub view raw

Specific to this example, which has two Salesforce regions (and consequently two schemas in the Heroku Postgres database — AMER and EMEA), the command to create the view looks like this:

```
CREATE OR REPLACE VIEW global_assessment as
SELECT employeecrisisassessment.sfid AS id,
employeecrisisassessment.assessment,
employeecrisisassessment.assessmentdate,
employee.lastname, employee.firstname,
'AMER' AS geo
FROM amer.employeecrisisassessment, amer.employee
WHERE employeecrisisassessment.employeeid = employee.sfid
UNION
SELECT employeecrisisassessment.sfid AS id,
employeecrisisassessment.assessment,
employeecrisisassessment.assessmentdate,
```

```
13  employee.lastna
14  'EMEA' AS geo
15  FROM emea.emplo
16  WHERE employeec
```

This account is under investigation or was found in violation of the Medium Rules.

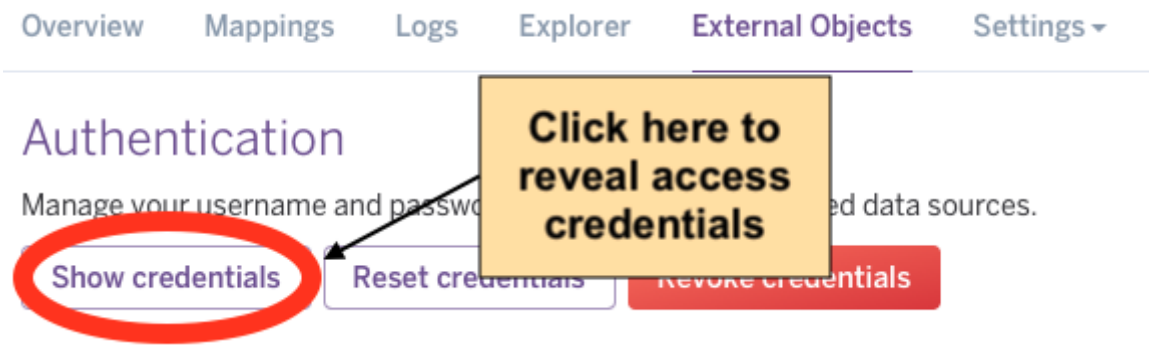
heroku_data_view_amer_emea.sql hosted with ❤ by GitHub view raw

Alternatively, if you're using Accounts and Contacts:

```
1  CREATE OR REPLACE VIEW global_assessment as
2  SELECT contact.sfid as id, industry, account.name, phone, fax,
3  lastname, firstname, email, mobilephone,
4  'AMER' AS geo
5  FROM amer.account, amer.contact
6  WHERE amer.account.sfid = amer.contact.accountid
7  UNION
8  SELECT contact.sfid as id, industry, account.name, phone, fax,
9  lastname, firstname, email, mobilephone,
10 'EMEA' AS geo
11 FROM emea.account, emea.contact
12 WHERE emea.account.sfid = emea.contact.accountid;
```

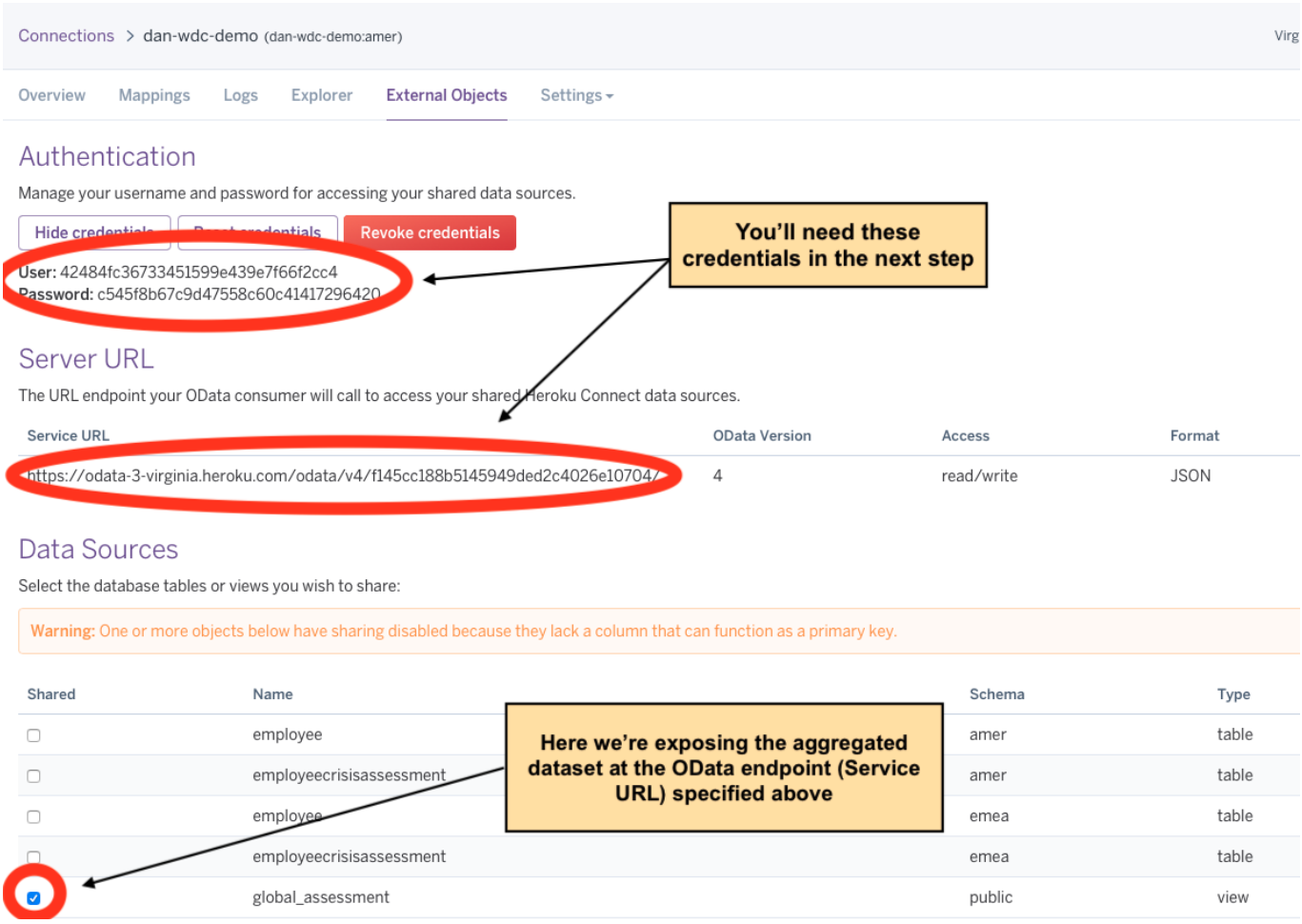
heroku_data_view_for_accounts_contacts.sql hosted with ❤ by GitHub view raw

Step 5a: Expose the aggregated view via Heroku External Objects



1. From the External Objects tab within the Heroku Connect Dashboard, click Show Credentials.
2. Make note of the User, Password, and Service URL parameters, which will be required to consume the aggregated data set within the Salesforce organizations that need access to the global data.

3. Click the checkl /iew you created in the previous ste



Note: Unlike steps 2 and 3, this step only needs to be completed once. The External Objects feature of Heroku Connect can potentially expose any of the tables within the Salesforce instance. The global_assessment view will be visible from any of them.

Step 5b: Consume the aggregated view from your Salesforce instance

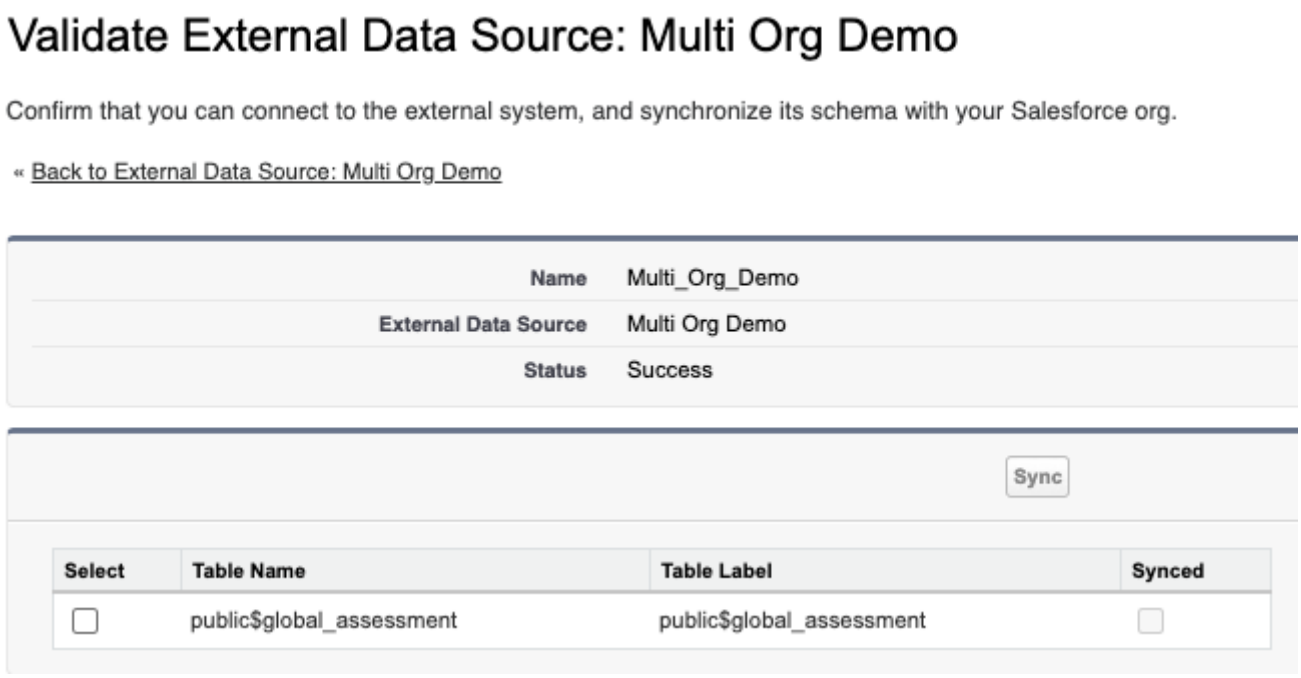
To virtualize the aggregated view within a Salesforce instance, you'll need an instance of Salesforce Connect within your Salesforce instance. As noted earlier, you can use a scratch organization for this purpose.

To configure the External Data Source:

```
1. Click the gear icon in the upper-right corner.
2. Click Setup.
3. Enter "External" in the Quick Find box and click External Data Sources.
4. Click New External Data Source. For the New External Data Source screen, fill in the
```

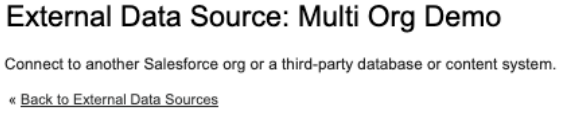
```
5       a. External
6       b. Type: Sa
7       c. URL: Thi
8       d. Identity Type: Named Principal
9       e. Authentication Protocol: Password Authentication
10      f. Username: The value from the User field, as highlighted in step 5a.
11      g. Password: The value from the Password field, as highlighted step 5a.
12  5. Click Save.
13  6. Click Validate and Sync.
```

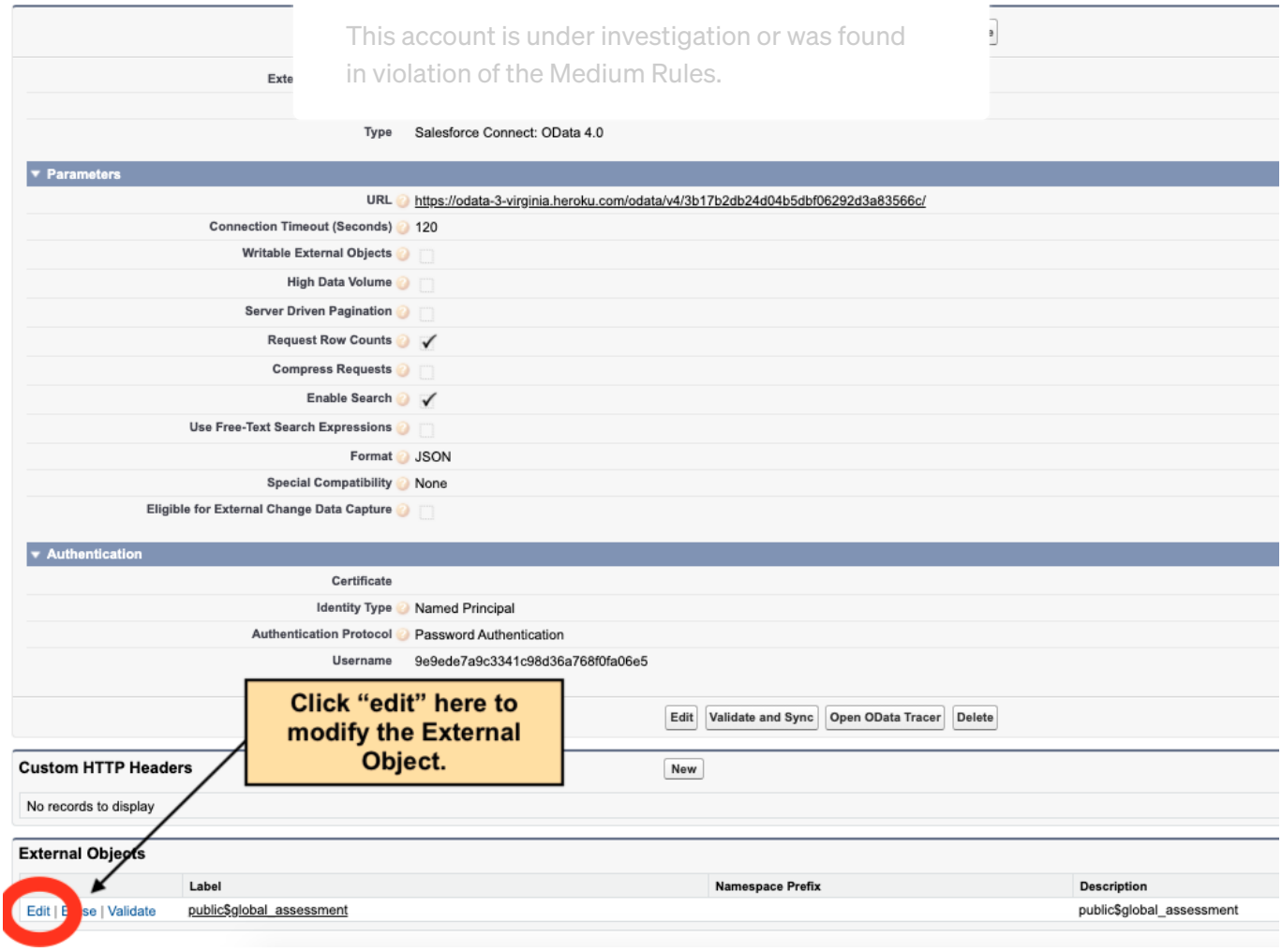

Steps to configure the External Data Source in Salesforce



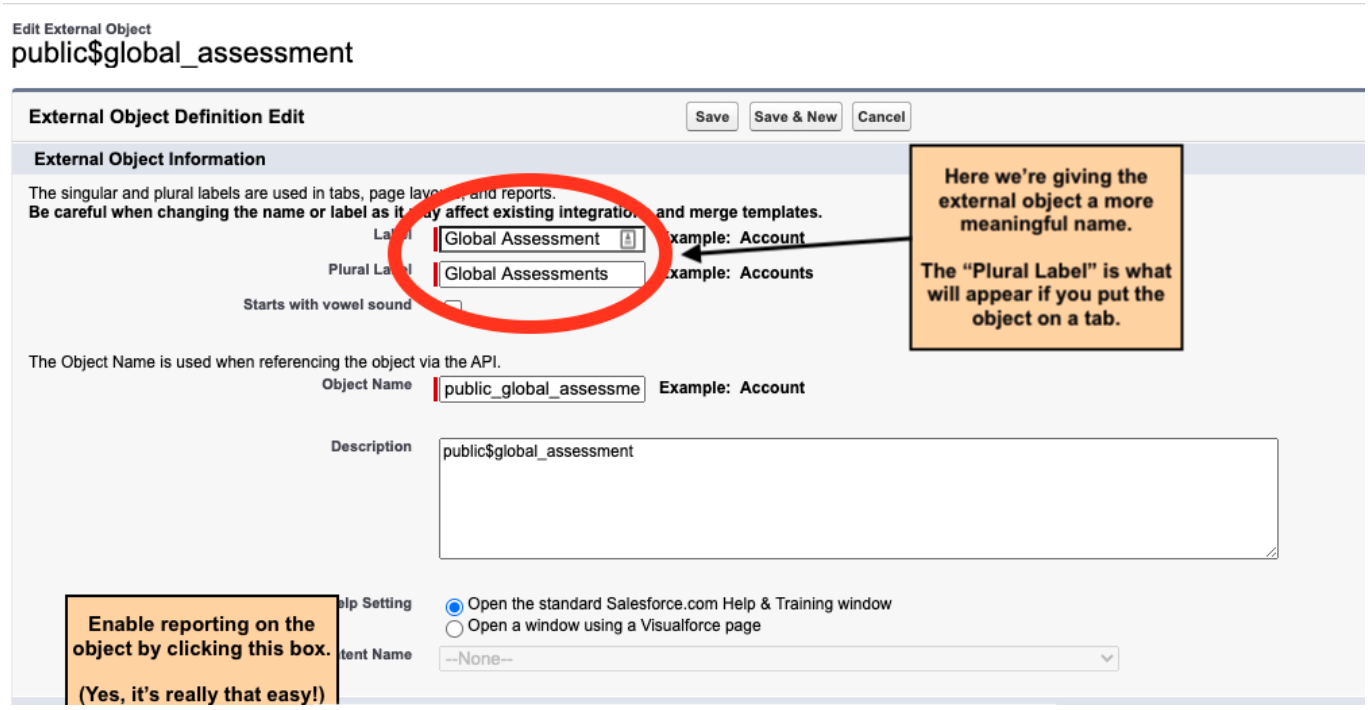
- When the sync completes, you'll see a screen similar to this. Click the Select checkbox next to "public$global_assessment." At this point, the object name is reflected as <schema name>$<table name>. We'll give this a more meaningful name in the next step.
- Click "Sync." (The Sync button will be greyed out until a table is selected.)

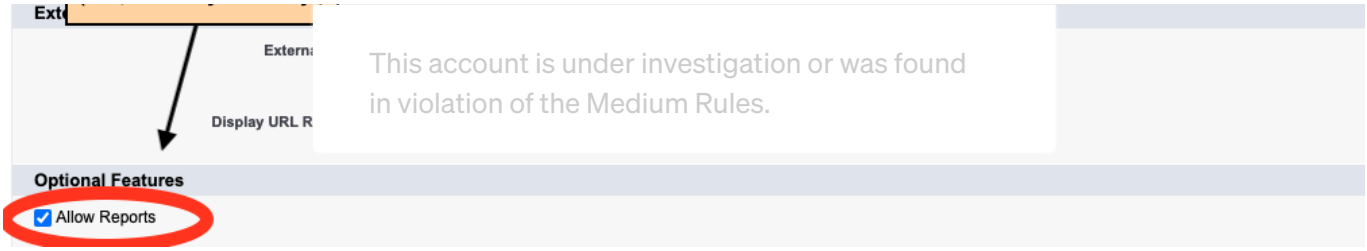
Step 6: Enable reporting (and other fine tuning)





When the sync process is complete, the External Object will appear at the bottom of the screen. Click Edit to give the object a more meaningful name and enable reporting.





Step 7: Enjoy!

With reporting enabled, you have the opportunity to use the drag-and-drop report builder and the drag-and-drop dashboard builder. Additionally you can put the External Object on a tab and take advantage of Salesforce features such as list views to view and filter records at the console.

Note: In this example, the geo field is initially created as a long text field. You can change the data type to a short text field of length 15. This allows the field to be used as a filter for list views, which is not possible with the long text field.

But wait! There's more!

We've kept the above example simple, and while we didn't account for global challenges such as multiple currencies, languages, timezone differences, or evolving data definitions there wasn't a need to. Real world challenges are likely to be more complicated, but the ease with which similar frameworks can be implemented enables to you focus on these details — the parts that are most useful to your customers or employees — while not having to worry about the infrastructure.

We're building this solution in a managed package that will soon be available on AppExchange. This Labs package will do everything we've walked through in the post, from creating and deploying a Heroku application to configuring multiple Salesforce instances to creating the unified view of the Work.com employee wellness data. Please keep an eye out for updates!

Conclusion

The app described in this post shows the value of the Salesforce Platform. As you develop solutions, you should be focused on engaging and delighting your customers, whoever they happen to be, and spending your time where it matters most. In this post, we've created reports containing data that spans multiple Salesforce environments. Heroku Connect will continue to sync data to Heroku Postgres and our

join will update autc contain up-to-date
data. Best of all, no (iired.

About the Author

Once upon a time, Dan Mehlman was a developer and experienced sleepless nights plagued by the very challenges Heroku readily solves. He's much happier these days (in general) and well-rested (specifically) and is known to spend his time talking about the many wonderful capabilities of Heroku (specifically) and the Salesforce Platform (in general). Today he is the Heroku Director of Technical Architecture.

Heroku Inside Salesforce Data Visualization Salesforce Architect Careers