Platform Academy

Durgesh Dhoot – Heroku/Salesforce India – Feb 2019

Introduction

Thank you for attending the Platform Academy. In this session, you will be creating a simple web application based on modern web frameworks called Angular JS, Ionic and Node.js. You will deploy it to Heroku and extend the application using Heroku Postgres from the Heroku add-on marketplace. In the process you will learn about Heroku Enterprise, architecture best practices, and gain an understanding of how Heroku works.

You will also configure Heroku Connect and use a simple hands-on exercise to provision a connection between Heroku Postgres Database and Salesforce to map objects and fields from Lightning Platform Custom objects to Heroku.

You will end this training with a recap of the key product items we have covered and help answer any questions you may have for your own projects.

Prerequisites

Heroku Account & Toolbelt

> *On Windows make sure to first install git before you install the Heroku toolbelt!*

If you don't have a Heroku developer account then please please sign up using the **https://signup.heroku.com/** page. If you are Salesforce employee then be sure to register with your Salesforce.com email address so you benefit from a host of free add-ons. Sign-up takes minutes and you will have a developer account from which you can manage your applications in one place.

As a developer your main interaction with Heroku will be via your command line. Heroku Toolbelt provides the commands required to interact with your Heroku service. Please go ahead and install the Heroku Toolbelt for your favourite operating system using **https://toolbelt.heroku.com/**.

Salesforce Account

Please sign up for a fresh Developer Org at:
https://developer.salesforce.com/signup.
As you will be using an unmanaged package to prepare the Org for the app, you might run into problems if you use another org that already contain objects with the same name than the package. If you are a Salesforce employee and are planning to use an Org created from **DemoForce**, the unmanaged packages required for setup and configuration might **not** work. **Please use a fresh Developer Org** instead.

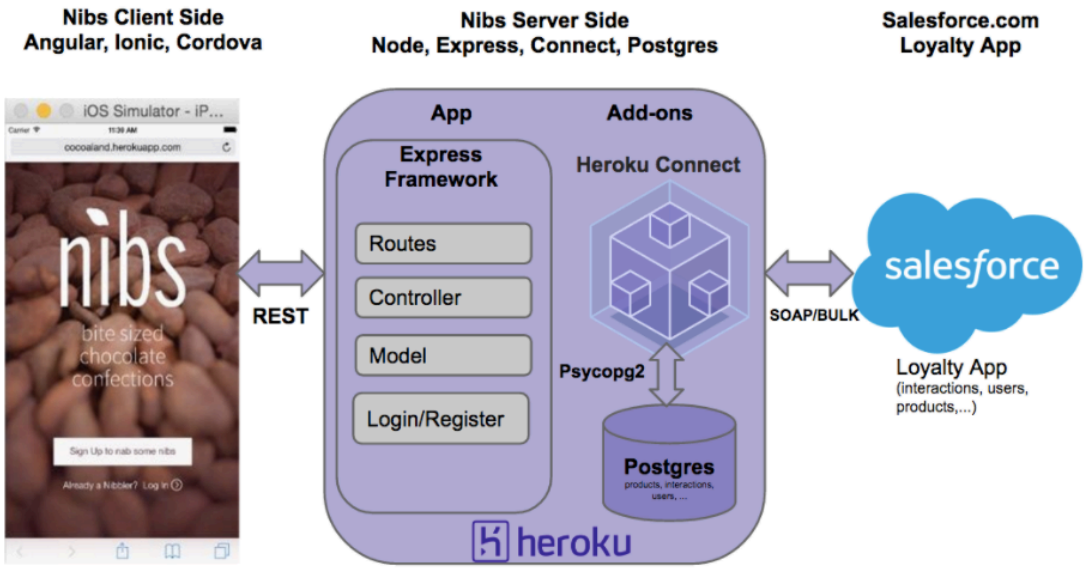
Heroku and Github Setup Verification

- Validate Heroku Installation
    - Launch a command-line and type:
    - `heroku --help`

- Git -- If you have not installed the git command-line tools (try typing **git** in a terminal) you can do so
    - On Mac download the installer (**https://git-scm.com/download/mac**)
    - On Windows download the installer (**http://git-scm.com/downloads**)
    - On linux with the following command:
        - sudo apt-get install git / sudo yum install git
    - Validate your Git Installation by typing `git --help`

Nibs App Intro and Architecture

You will be using the **Nibs** app as a hands-on experience during this session. Nibs is a sample mobile application built with AngularJS, Ionic and Node.js that you will deploy in Heroku. Nibs uses a backend Postgres database to store information and Heroku connect to synchronize that data with the Salesforce Lightning Platform.



Nibs uses **Lightning Platform** for the employee-facing side of the application, including products, offers and user management. Employees can manage Nibs data in the browser or using the Salesforce Mobile app.

Heroku provides the tools and services you need to build and deploy customer-facing applications without having to worry about infrastructure, including continuous delivery.

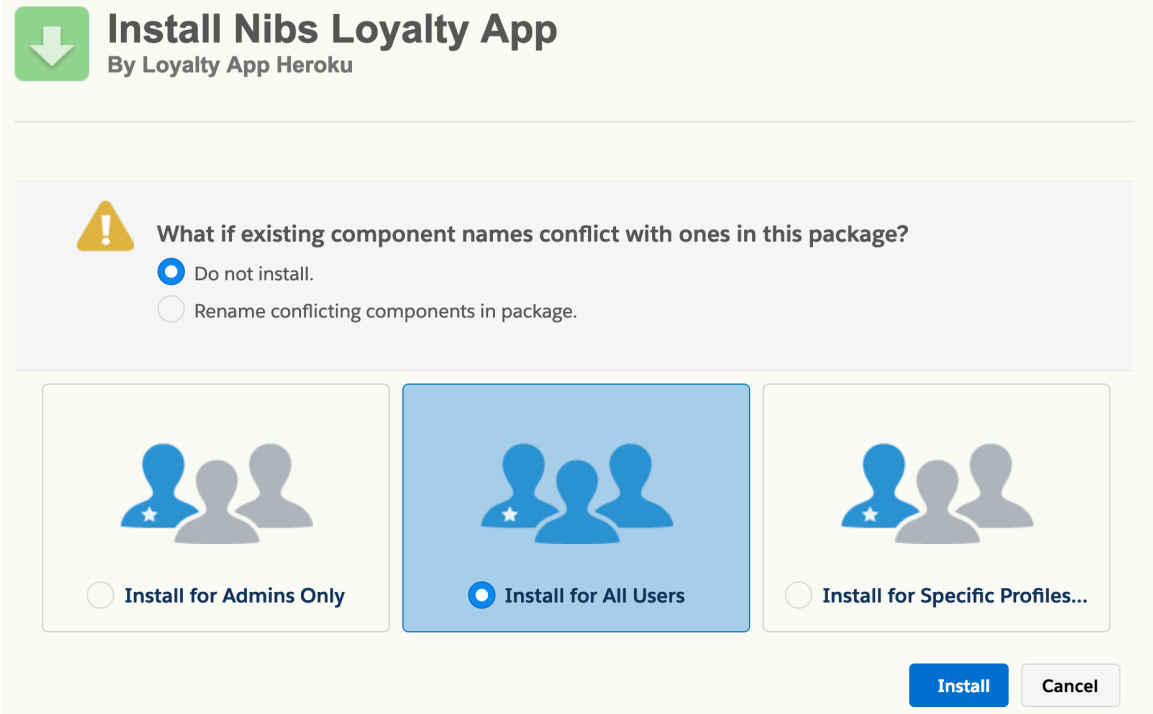
Step 1 - Preparing Salesforce

1.1 Log into your Developer Org at http://login.salesforce.com

1.2 You will use an unmanaged AppExchange package to install the custom objects and sample data needed to start using the Nibs app right away.

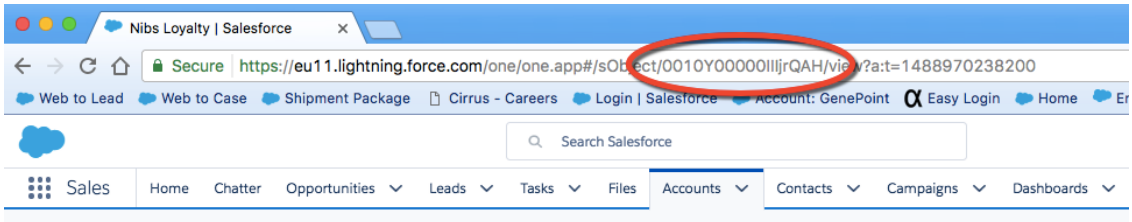
Open another tab in your browser and point it to
https://sfdc.co/nibs-sfdc-package

Click **Install for All Users - Install**



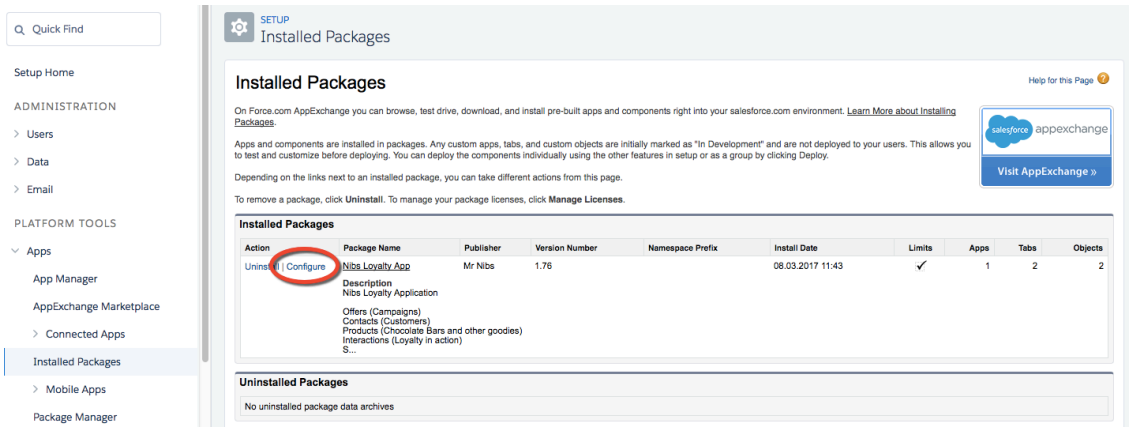
1.3 Post Managed Package Steps

Create the Nibs Loyalty Account so you can track customers who signup to use the Nibs app. Go to the tab **Accounts**, click "**New Account**" and fill in the Account Name as **Nibs Loyalty**. That is the only required field. Click "**Save**" and take note of the AccountID (like 0010Y00000IIIjrQAH from the screenshot below), you will need it during the app configuration in Heroku at step 3.3.

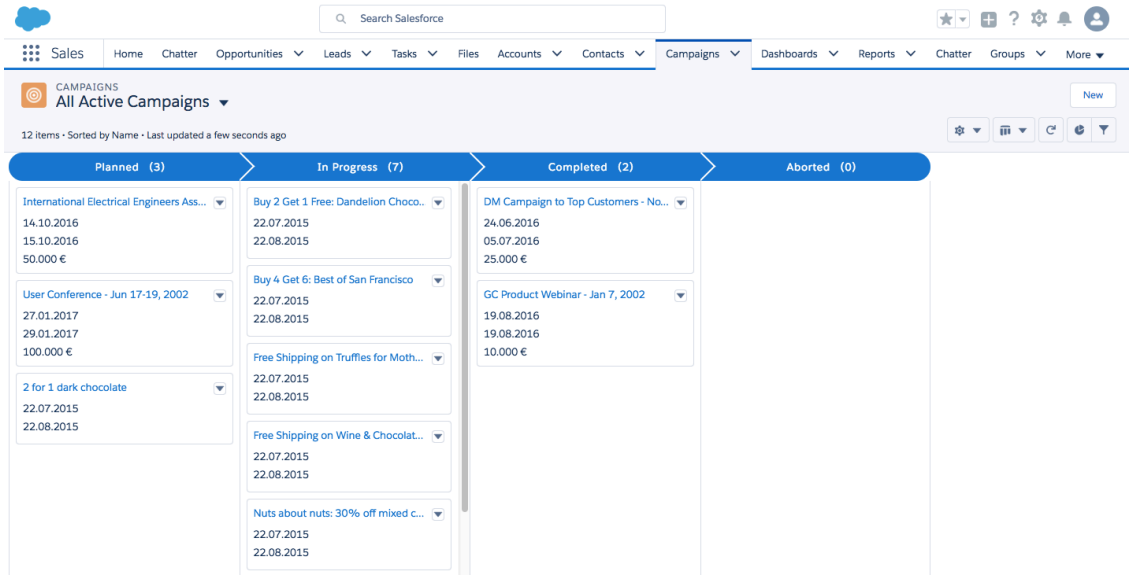


1.4 Post Managed Package Steps

You are now ready to populate the Org with some mock data. In your Developer Org go to **Setup → Installed Packages** and click on **"Configure"** on the left side of the "Nibs Loyalty App" package. On the next screen click on the "**Install Data**" button.



After you click the "**Install Data**" button you will setup data within various standard and custom objects. Click on the Campaigns and open the All Active Campaigns list view tab and should see all your newly created campaigns (optionally switch between Grid and Kanban view).



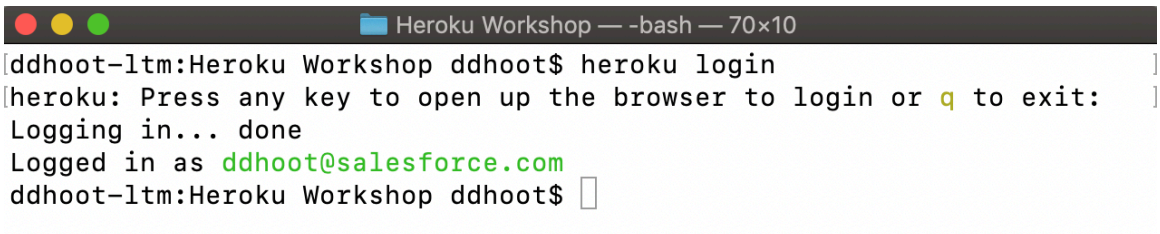
Step 2 - Deploying the Nibs app in Heroku

In Heroku most of the actions can be done using the web dashboard with point and click or using the command line interface. For this demo you will use the second way. Remember that you will need the Heroku Toolbelt previously installed and your Heroku developer account. The link is in the Prerequisites section of this manual. Change to your working directory e.g Desktop (or wherever you want your code to be created on your laptop).

2.1 Open a terminal in your laptop and type

```
heroku login
```

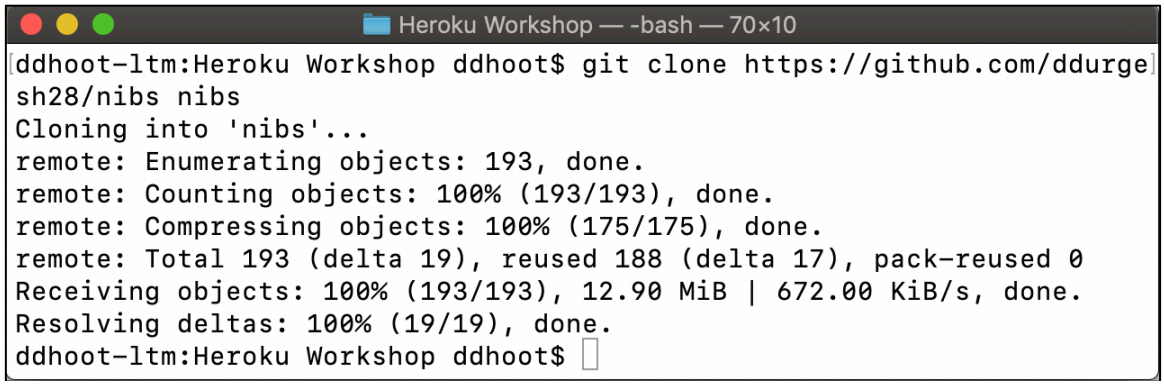
You will be prompted for your heroku credentials, which are the ones you used when you signed up for a Heroku developer account.



2.2 At this point you would have to write the application. Don't worry, we will save you a couple of hours, we have done that for you. The source code is ready for download from Github.

You only need to clone it to your laptop typing

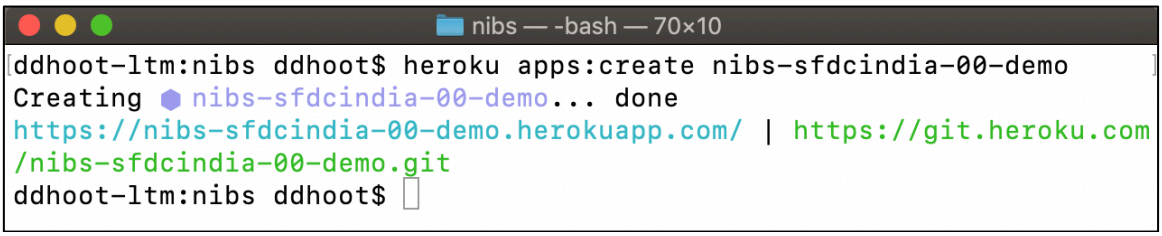
```
git clone https://github.com/ddurgesh28/nibs-buggy nibs
```



2.3 It's time to tell Heroku that we want to create a new app. Jump into the directory where the source code is and type:

```
cd nibs
heroku apps:create nibs-sfdcindia-$$-demo
```

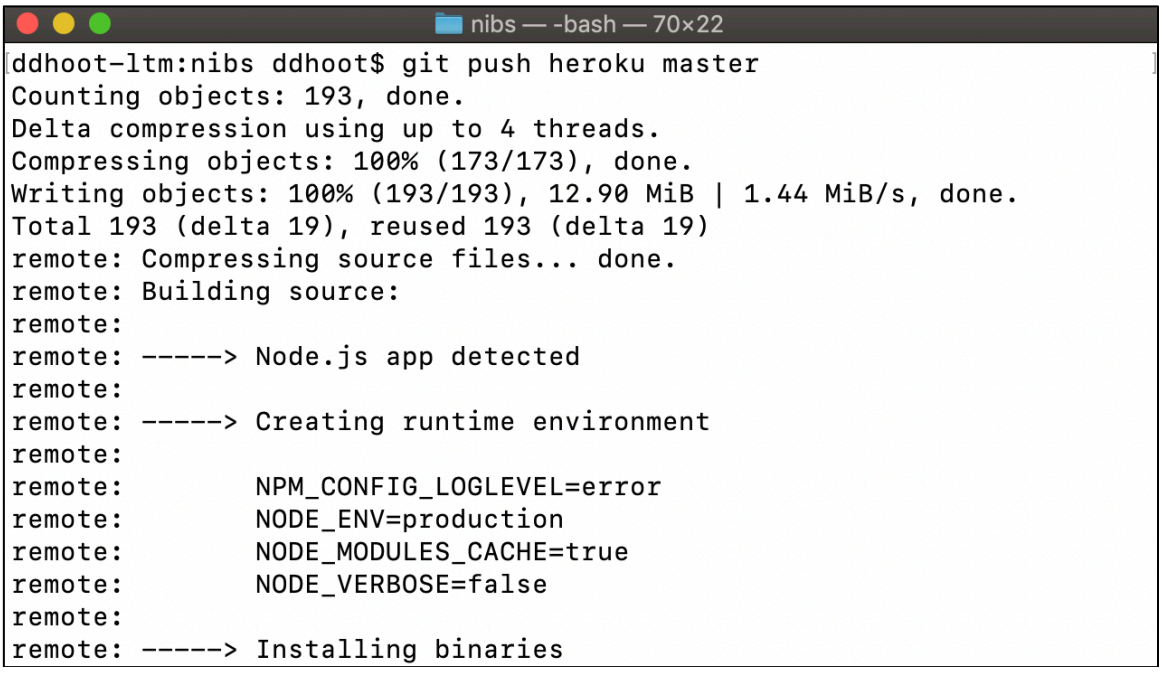
($$ is the same random number you used to create the integration user in Salesforce. The app name needs to be globally unique, that is why we are asking you to be creative and pick a random number).



It might seem simple (and it is!) but in the background Heroku has prepared the app and is ready to receive your source code. A git remote (called heroku) is also created and associated with your local git repository.

2.4 Let's push the source code of your app in heroku. Type

```
git push heroku master
```



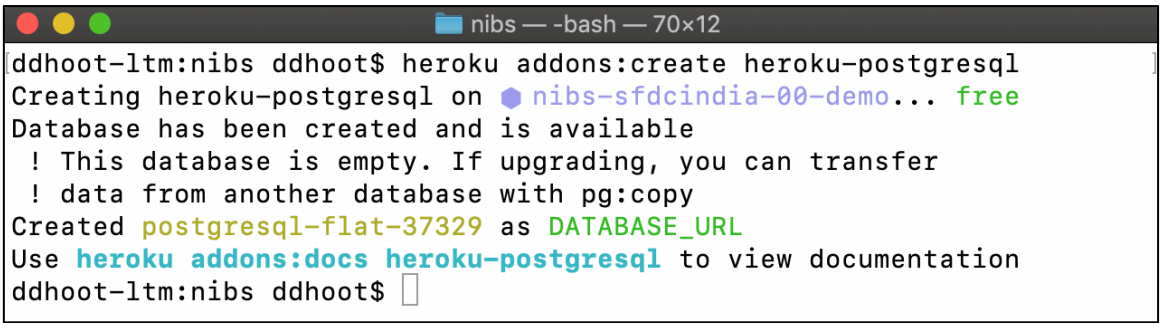
As you can see, the app has not only been uploaded to Heroku. It has also been compiled (creating the "slug" of the app) and is now ready to run!

Step 3 - Preparing the connection with Salesforce

The app is ready to run, but before that, you need to bring the data that you have in your Developer Org to Heroku, and enable your app in Heroku to make changes in your Org (that's truly bidirectional!!).

3.1 Install, prepare and configure a Postgres database in Heroku. Sounds like a work that will take more than an hour, right? Try this in your terminal and don't blink or you will miss it:

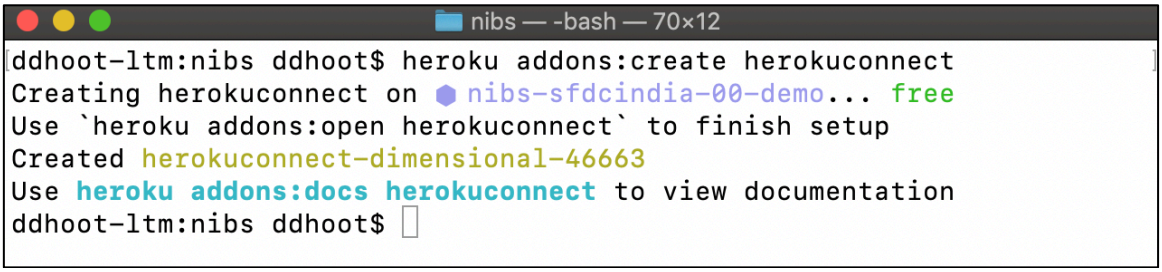
```
heroku addons:create heroku-postgresql
```



That's the backend database for the app installed, configured and connected. We have installed Postgres as an add-on, which are fully managed services, integrated for use with Heroku. As you saw, they can be provisioned and scaled in one command, there are over 170 available and provide services for logging, caching, monitoring, persistence and more… (check https://elements.heroku.com/)

3.2 Now you will have to connect the Postgres backend database with your Developer Org. Fear not! There is an add-on for that called Heroku Connect that enables bidirectional synchronization between both, and since it's an add-on, you will have it deployed right away!
Type in your terminal

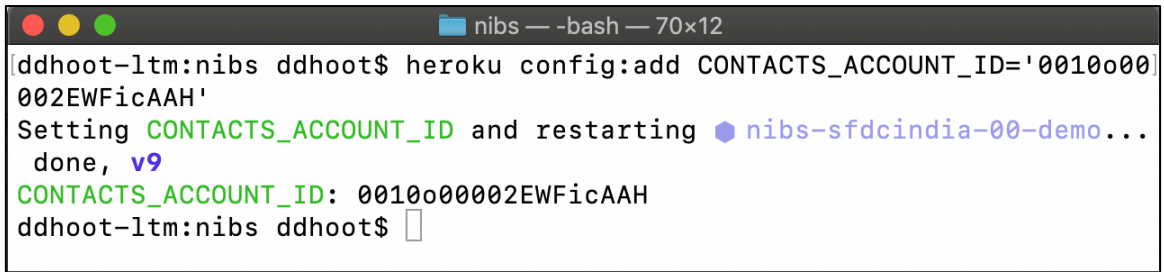
```
heroku addons:create herokuconnect
```



3.3 There is some configuration that your app will need to run. In Heroku the configuration of your app is shared between dynos using Configuration Variables (check **https://devcenter.heroku.com/articles/config-vars**).
Type on your terminal:

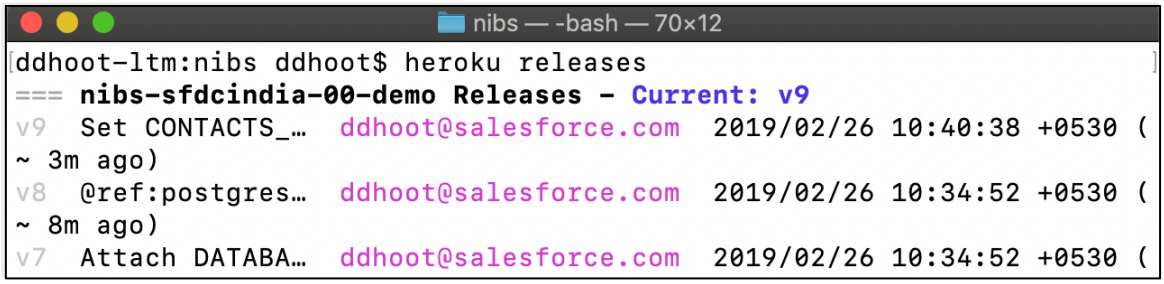
```
heroku config:add
CONTACTS_ACCOUNT_ID='0010o00002EWFicAAH'
```

(**you will have to change this ID with your own "Nibs Loyalty" accountID**, that you created a few minutes ago **in step 1.3**)



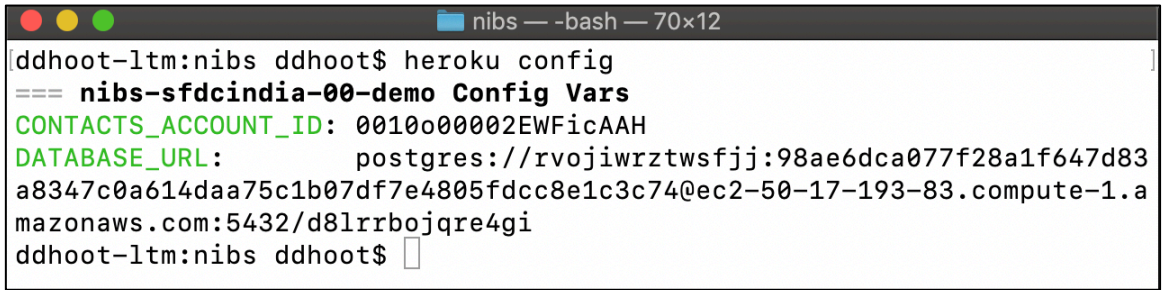
> ***Did you know:***
> *Every time a configuration change is done in the app, or when new code is pushed, or when you add or remove an add-on, Heroku creates a new release of your app and restarts it, so the app is always running the latest version of code and configuration.*

```
heroku releases
```



> *Also, if there is any issues with the new code or configuration it is easy to roll back to a previous release with just one command so your app will be back in a healthy state quickly (check **https://devcenter.heroku.com/articles/releases**)*

```
heroku config
```



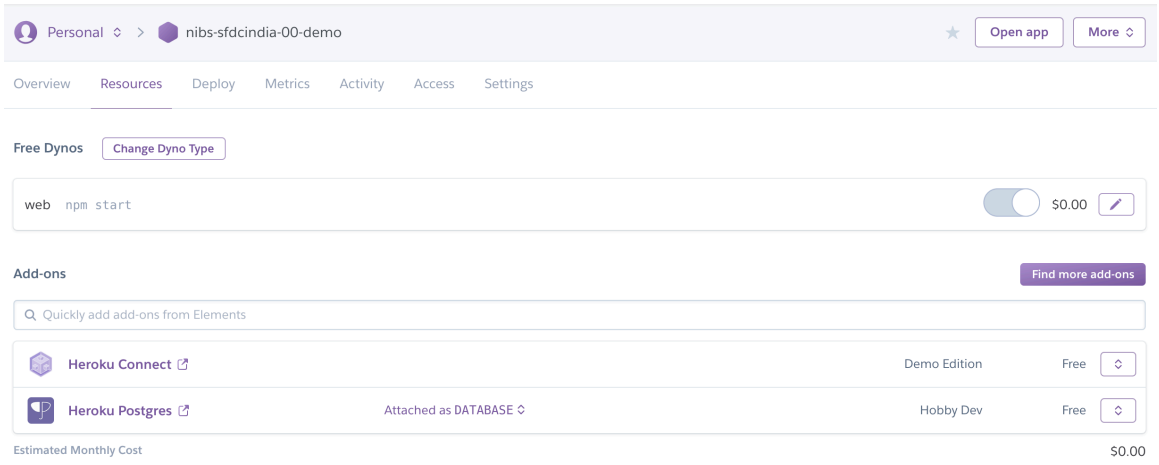
Step 4 - Configuring Heroku Connect

Let's switch gears and use the graphic interface for this step. Keep in mind that, as always, there is a command line interface way to do it.

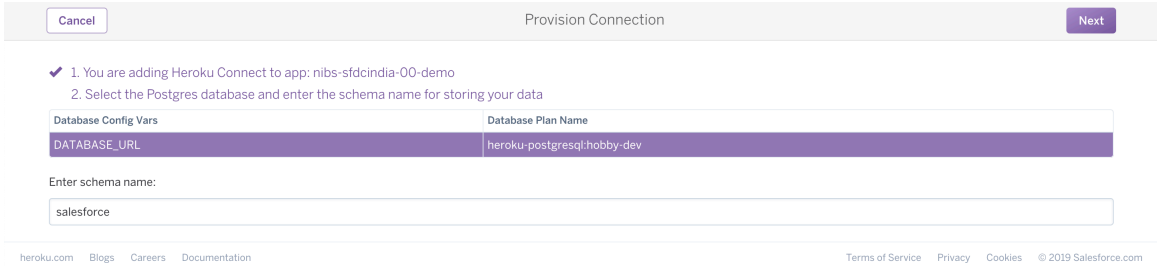
4.1 Login to your heroku dashboard (https://id.heroku.com/login) and click on your Nibs app (it will have the name **nibs-sfdcindia-$$-demo**).

That is the dashboard of your app. The "Resources" section shows your dyno formation (the number and type of dynos that make up your app) and the add-ons configured.

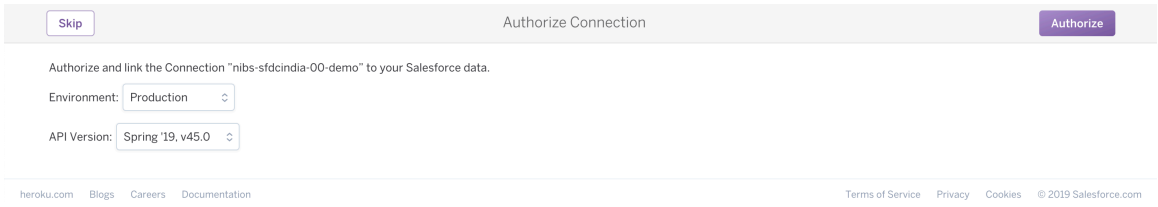
You should have a 'web' dyno, which is the front end of your app, and two add-ons: the Postgres database for the backend and Heroku Connect for the synchronization with your Salesforce org.



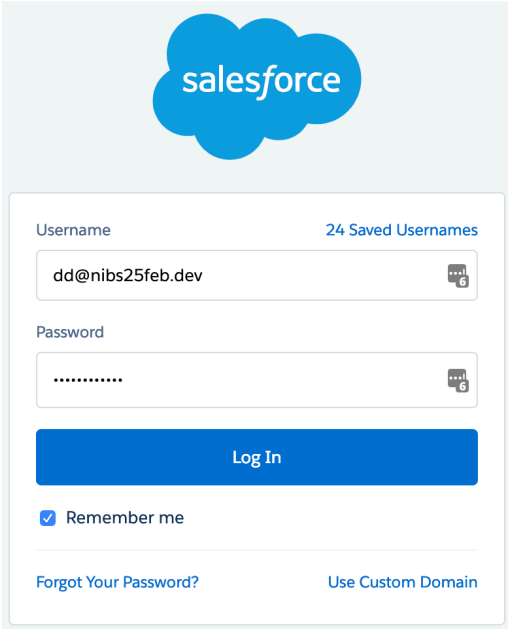
4.2 Click on the **Heroku Connect** from the add-ons section of your app, authorize **HerokuConnect**, click **Setup Connection**, accept the defaults on the 'Provision Connection' screen (make sure you don't change the schema name - it should be 'salesforce' for this app to run), and click **Next**.



4.3 In the 'Authorize Connection' screen, select **Production** environment and change the API version to the most current one: **Spring '19, v45.0** and click **Authorize**



Authenticate using the salesforce user account you created in Step 1.1 and click **Allow** on the next screen.



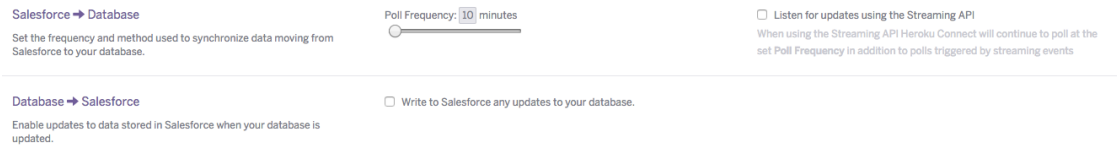
Now Heroku Connect can log in to your Salesforce org and synchronise changes.

4.4 The next step is to create the mappings. You will have to tell Heroku which objects (and which fields of those objects) from your org you want to be synchronized.

Click the '**Create Mapping**' button and a list of all the objects from your Salesforce org will be listed. Scroll down (or search) and select '**Store__c**'.
On the next screen you will see the fields of that object. Select **LastModifiedDate**, **Location_Latitude__s** and **Location_Longitude__s**.

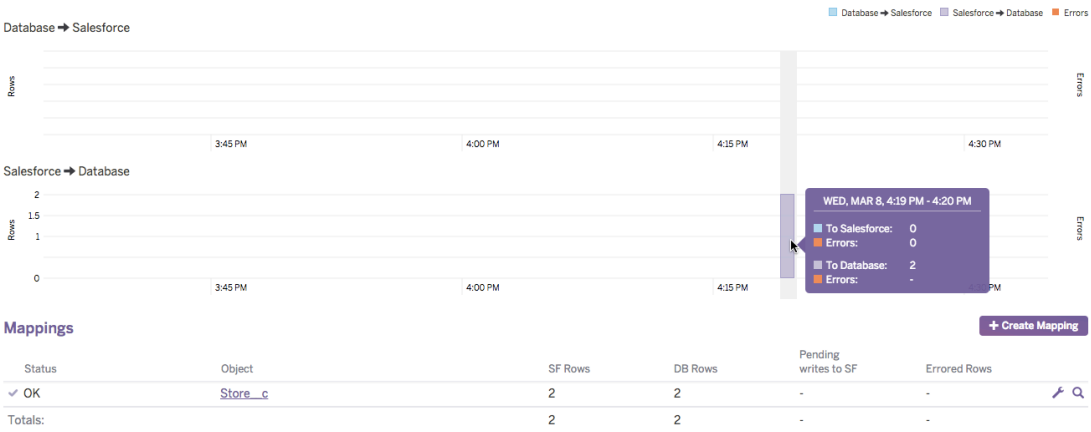
Sync	Field	Label	Type	Length	Indexed
☐	CreatedById	Created By ID	reference to: User	18	☐
☑	CreatedDate	Created Date	datetime	0	☐
☑	Id	Record ID	id	18	☑
☑	IsDeleted	Deleted	boolean	0	☐
☐	LastModifiedById	Last Modified By ID	reference to: User	18	☐
☑	LastModifiedDate	Last Modified Date	datetime	0	☑
☐	LastReferencedDate	Last Referenced Date	datetime	0	☐
☐	LastViewedDate	Last Viewed Date	datetime	0	☐
☑	Location__Latitude__s	Location (Latitude)	double	0	☐
☑	Location__Longitude__s	Location (Longitude)	double	0	☐
☑	Name	Store Name	string	80	☐
☐	OwnerId	Owner ID	reference to: Group	18	☐
☑	SystemModstamp	System Modstamp	datetime	0	☑

> ***Did you know:***
> *On the mappings page you have also the configuration for the polling frequency from Salesforce to Heroku, that can be adjusted from 2 to 60 minutes (10 to 60 in the demo plan).*
> *Custom objects and many standard objects also support streaming mode, in which Salesforce will notify Heroku when there has been changes in the object and it triggers a poll.*
> *By default, Heroku Connect is configured in 'read only', meaning that no data will be synchronized from Heroku to your Salesforce org. If you need bidirectional synchronization then just click the 'Database → Salesforce' box.*



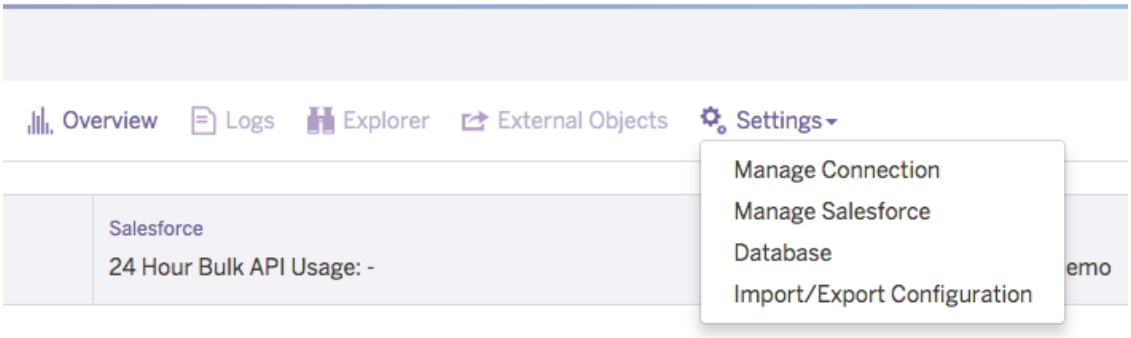
> *You can find more information here:*
> *https://devcenter.heroku.com/articles/herokuconnect#synchronization-mode*

Click '**Save**' and you will see at the bottom of the screen that a new table for the Store__c object is created and it starts the synchronization automatically from Salesforce.

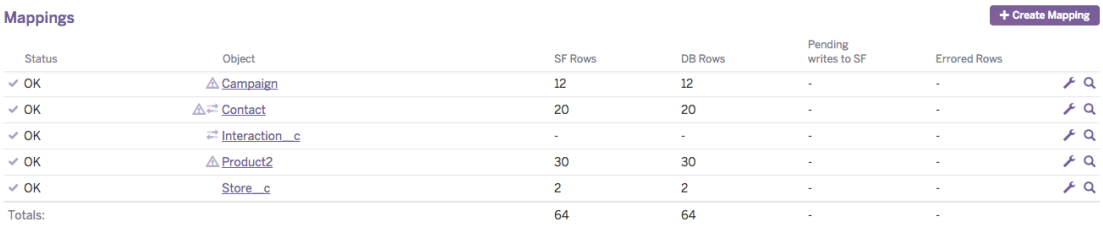


4.5 You would have to map the rest of the objects for the app now. To save some time we have created a json file with the rest of the mappings. You can download it from here: https://sfdc.co/nibs-mapping

At the top of the page, click **Settings → Import/Export Configuration**



Then click **Import** and choose the mapping file you downloaded a moment ago. Then click **Upload**. You will see at the bottom of the screen how the new tables are created and the synchronization starts.



And that's it! Your Salesforce Org and your Heroku app are in sync and you are ready to go…

Step 5 - Testing the Nibs application

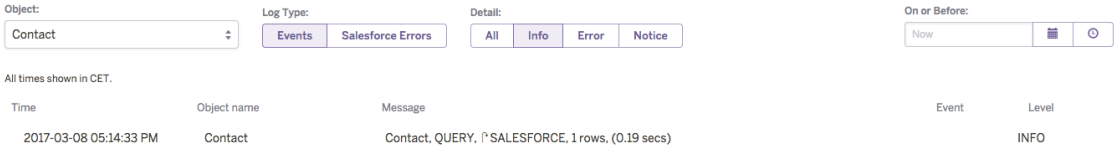
Your application is ready! Let's give it a try.

5.1 Launch the Nibs app by typing on your terminal `heroku open` (you guessed it, you can also open your app using the Heroku dashboard).
You can even access the app from your mobile device if you point the browser to https://nibs-sfdcindia-$$-demo.herokuapp.com/ (remember to change $$ for the number you chose).

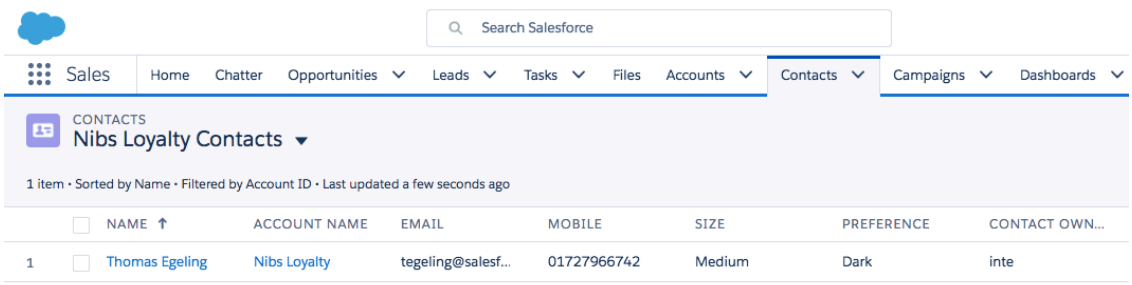
5.2 Create a new user. Click on the **Sign Up** button and fill in the form. Then, login to the app with the credentials of the new user. (Facebook login will not work at this point, you will configure it later in this session)

Check if everything is working smoothly. When you create a new user in the app, it should be created as a new contact in your Salesforce org.

5.3 Go to the **Heroku dashboard** of your app, click on the **Heroku Connect** add-on and then click on the **Logs** section. You will see that there has been an insert operation to Salesforce:



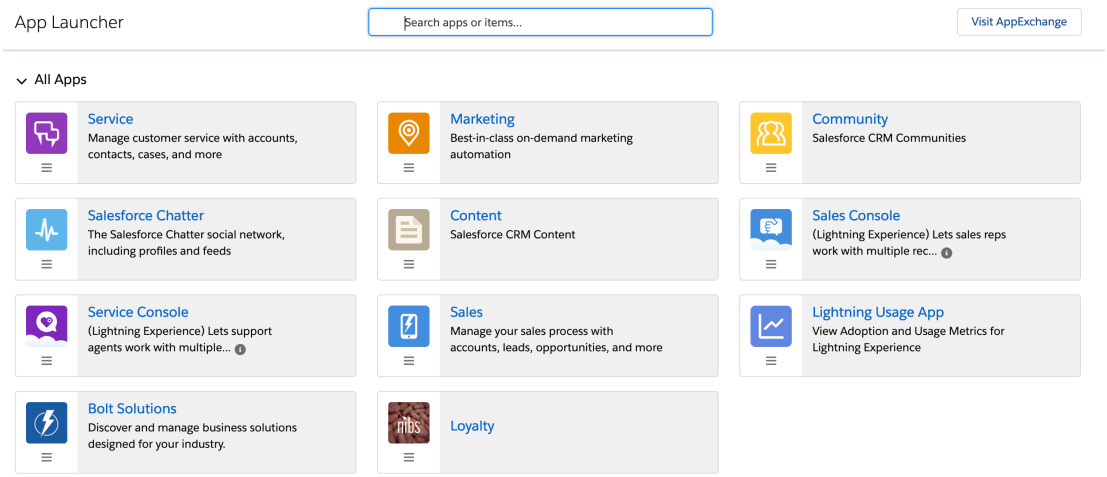
If this is correct, you could also check in your Salesforce org that the contact has been created:



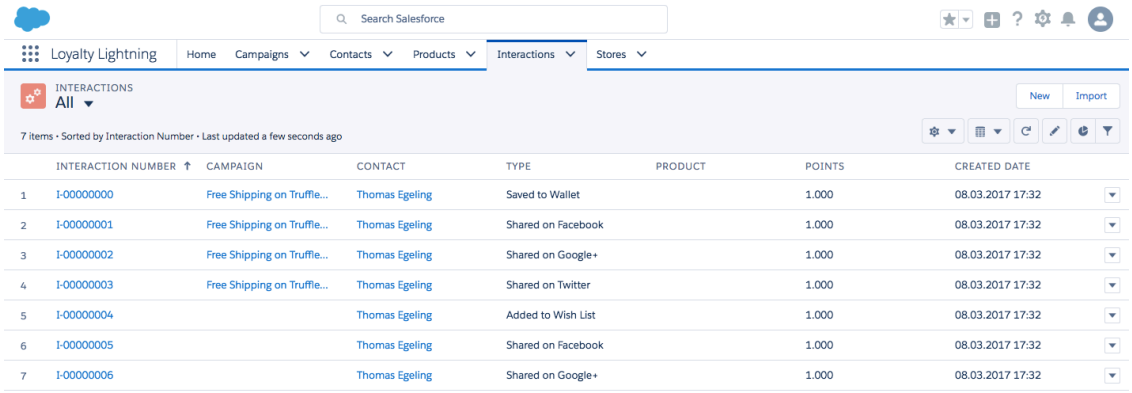
5.4 So far so good! Let's interact a bit with the application. Click on the top left corner of the app to deploy the menu. Check the **Selections** and add a few products to your **Wish List**.

You can also check the **Specials** (on the menu again) and **Save to Wallet** a few of them, or **Share** on facebook or twitter (don't worry, it's a mock and the app will not post anything annoying on your behalf).

5.5 Open App Loyalty

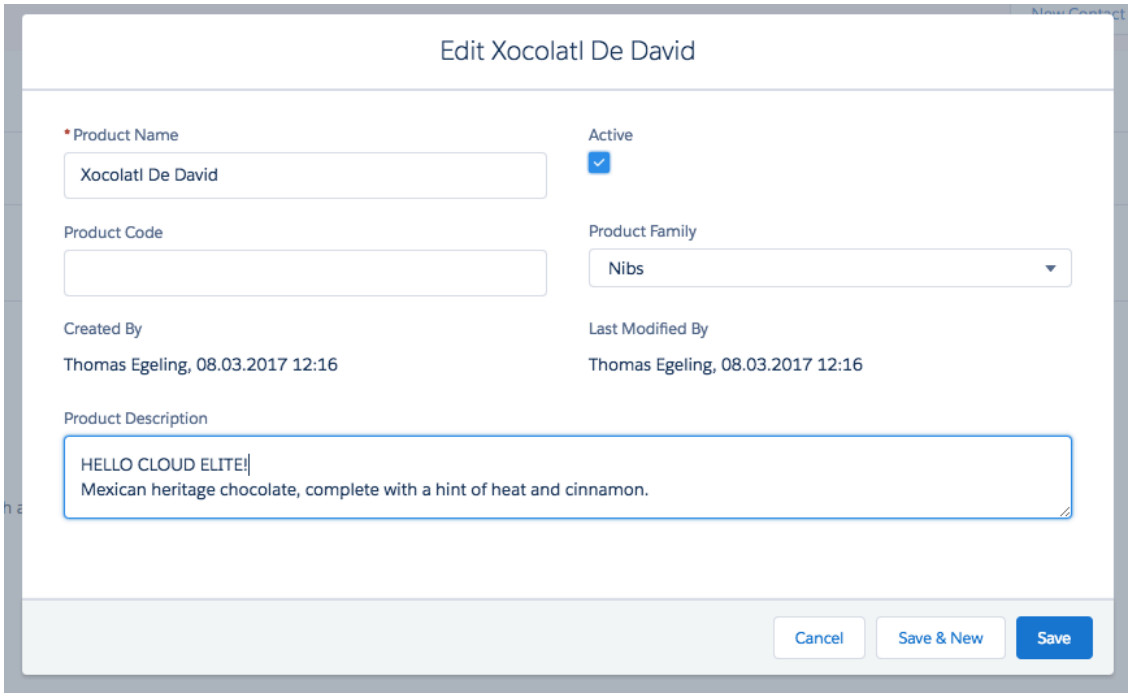


5.6 Let's check if the interactions are being synced with your Salesforce org. Go to your Developer org and check the **Interactions** tab (you might have to select the application '**Loyalty**' on the top right dropdown menu to see the tab).

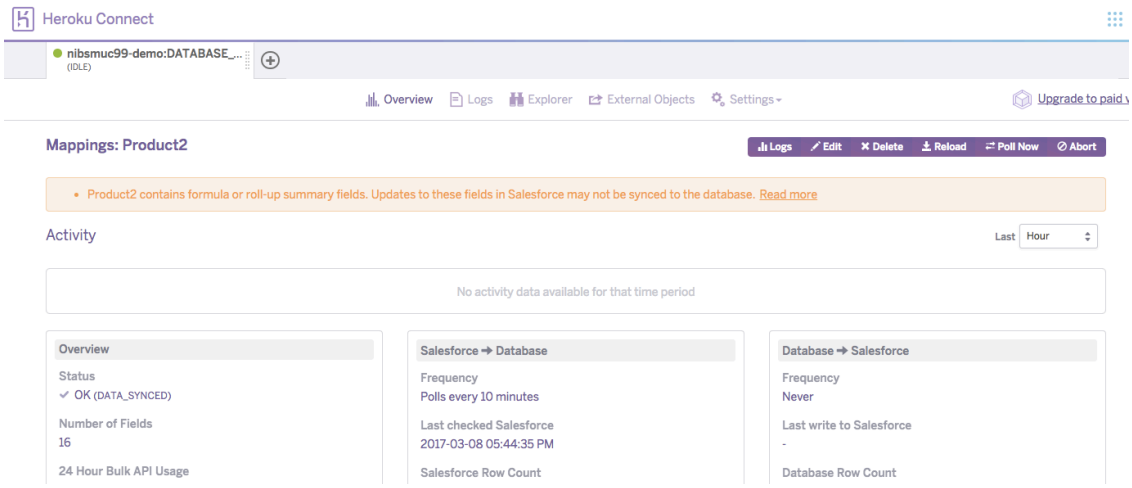


As you can see, all the interactions in the app in Heroku are being replicated to your Salesforce org, so your agents can keep track on what your customers are doing, understand patterns, give better customer service or plan new marketing strategies!

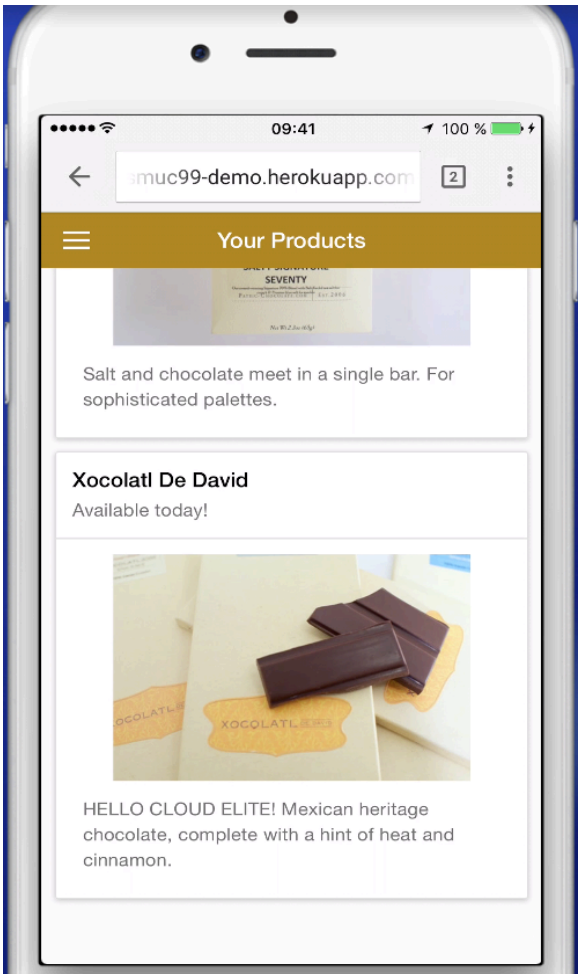
5.7 Sync also works the other way around. You maintain your products catalogue in Salesforce, and any changes on that catalogue will be synced with Heroku.
In your Developer org, go to the **Products** tab, select one of the products and change the description (and click **Save**):



If you remember, the poll time for synchronization you configured was 10 minutes. You can wait that time and go for a coffee or force the sync to happen. Go to your **Heroku dashboard**, click on your app **nibs$$-demo**, click in the **Heroku Connect** add-on and then at the bottom of the screen click on the **Product2** mapping. On the top right of the screen, click the **Poll Now** button.



Now go to your Nibs app on your mobile or your browser, click on the top left corner to open the **Menu** and go to **Selections**, scroll down until you see the product you changed…



Congratulations!

In a few minutes you have successfully connected and app running on Heroku with your Salesforce org, gathering data about your users and updating the contents of your app from your org, so you have a single point of truth.

Choose your own adventure!!!

If you want to learn how Heroku can help development teams seamlessly integrating **continuous delivery** and staging, carry on with **Step 6**.

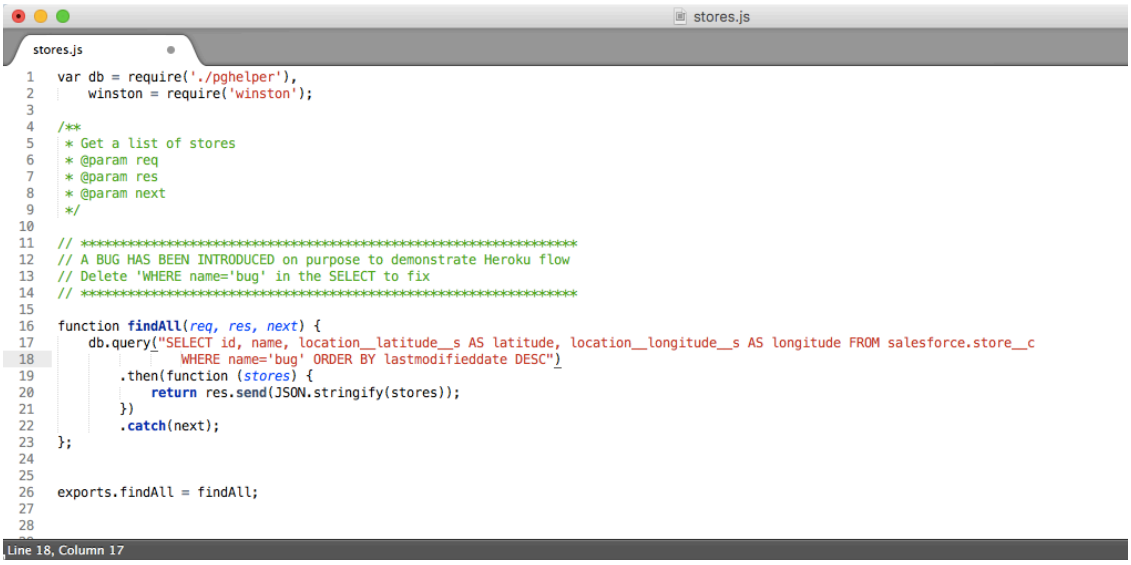
If you want to learn how to **integrate OAuth** and an external Sign-On platform (Facebook), jump to **Step 7**. You can always hop back to Step 6 later!

Step 6 - Fix the Nibs app!

If you have played a bit with the app you might have noticed that the Stores page does not work properly. It shows a map, but no stores in there… And you have two stores in your Salesforce org (go check!).

What is happening? It might be a bug on the code. Oh, well… These things happen. Let's try to fix it!

6.1 The problem seems to be only in the Store Locator page. From the source code of the app in your laptop, open the file **server/store.js** with your favorite text editor and take a look at the code… Someone left there a good clue of what is happening…



It was easy this time. Just delete **WHERE name='bug'** from the SELECT statement and it will work.

Are you sure it will work? What if you break the app by pushing this change straight to production? Of course, you have **heroku rollback** as you saw previously in this session to quickly bring back the last working version of your app, but best practices on development tell you should deploy the change first to a pull request, then merged into the master branch, which is tested in the staging environment before promoting it to production.

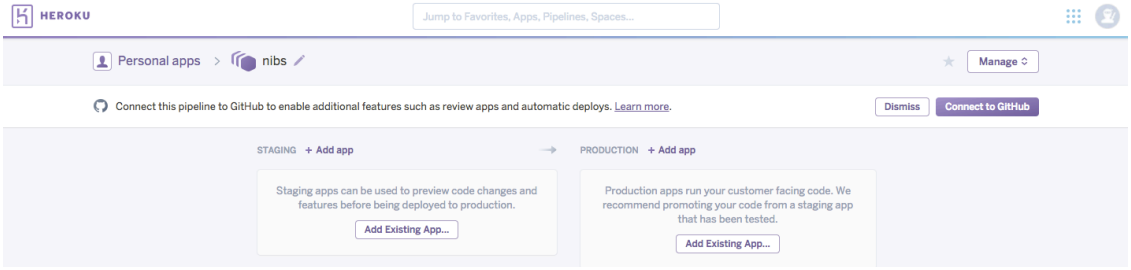
Heroku helps to streamline this process in a very easy way, with a friendly command line or graphic interface, eliminating common problems of continuous integration like pushing the wrong commit, releasing untested code or pushing the incorrect environment.

6.2 Heroku Pipelines allow you to define how your code should flow from one environment to the next. We will use a simple staging to production pipeline in

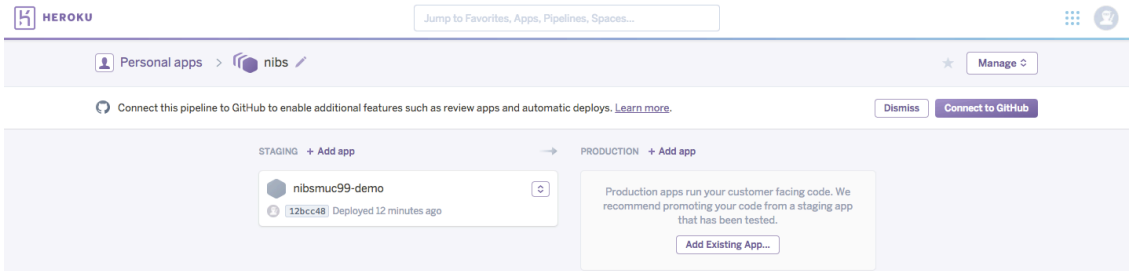
this demo, but more complex pipelines can be built on Heroku to adapt to the development team.

On your Heroku dashboard, **click on the New button** at the top right corner of the screen, and select **Create New Pipeline**.

Fill in the name for the pipeline (**nibs** sounds appropriate) and click **Create Pipeline**.



6.3 You will add the existing nibs$$-demo app as staging. On the staging section of the screen, click **Add Existing App** and type **nibs-sfdcindia-$$-demo**. The app will be added to the staging environment.



6.4 We need an exact copy of the Nibs app that we will consider production. But to do this we need to add a Heroku CLI. Plugins allow developers to extend the functionality of the Heroku command interface, adding commands or features. (check https://devcenter.heroku.com/articles/using-cli-plugins for more details)

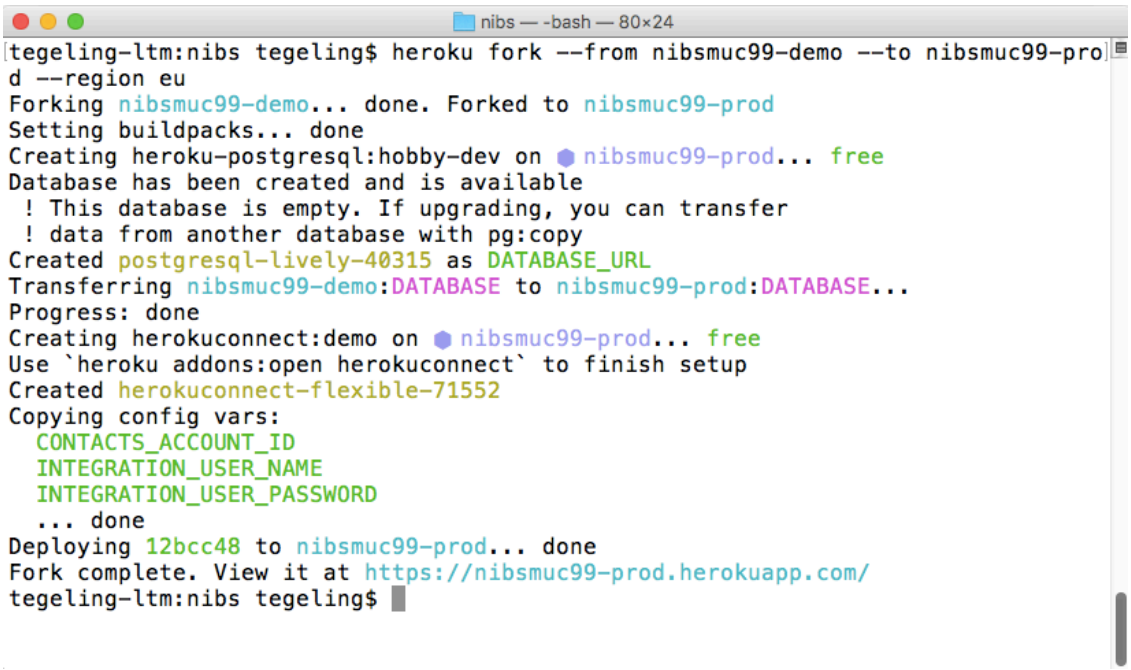
Go to your terminal and type

```
heroku plugins:install heroku-fork
```

This will install the cli plugin to fork an existing app into a new app. Now type:

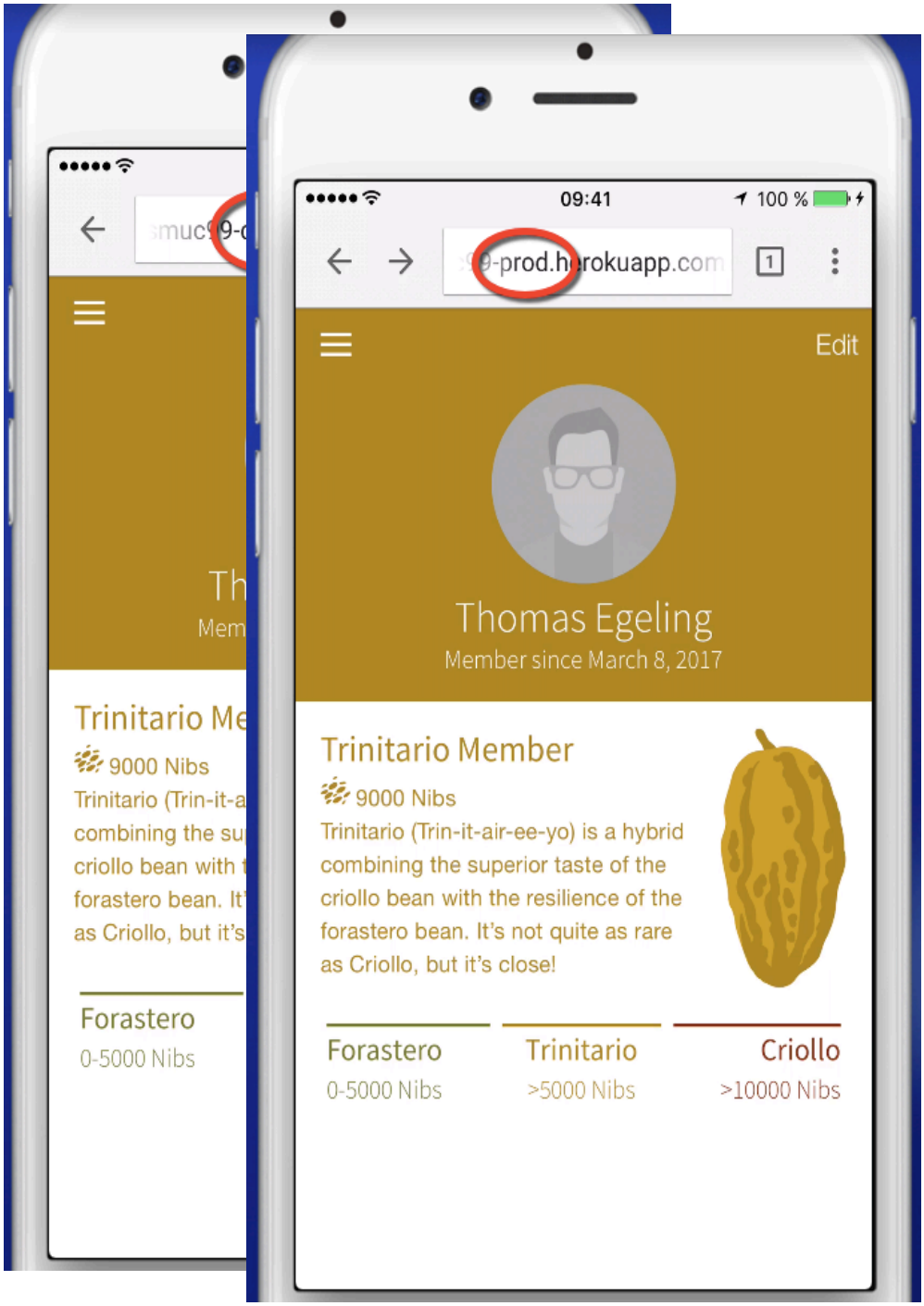
```
heroku fork --from nibs-sfdcindia-$$-demo --to nibs-sfdcindia-$$-prod
```

(as usual, change $$ for your random number). Heroku Fork copies not only the source code and the compiled code, but also the config variables and the add-ons.



We will use the new **nibs-sfdcindia-$$-prod** app for the production environment. Go back to the Heroku dashboard and add the new app in the nibs pipeline clicking **Add Existing App** in the production environment.

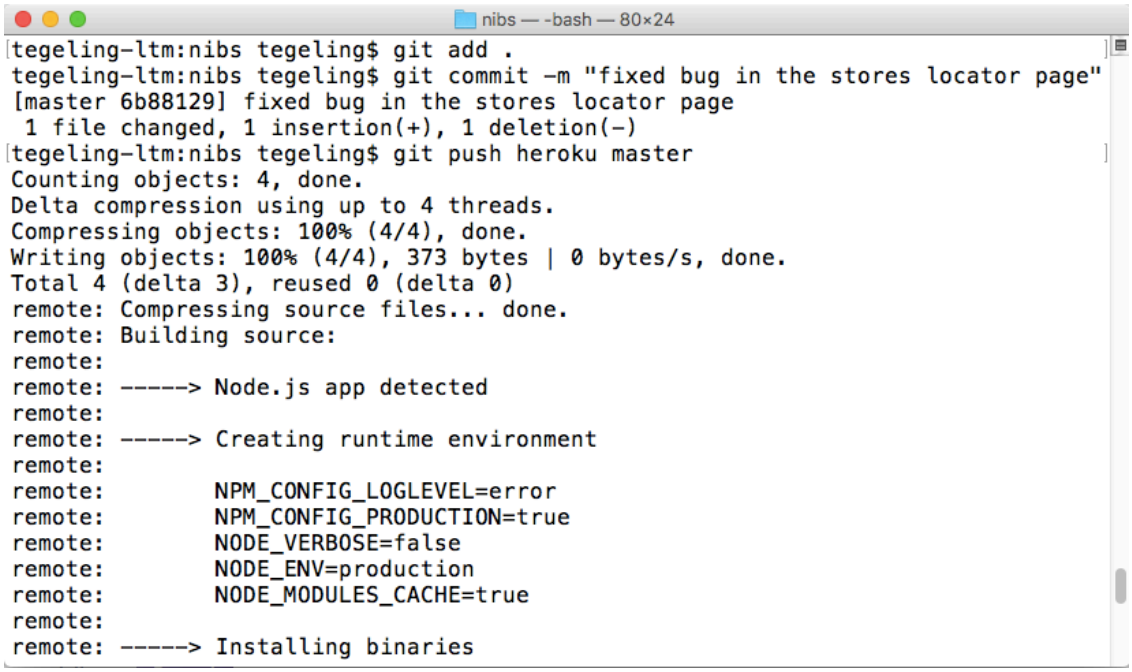
6.5 At this point, we have twin applications running (and both with the same bug!). Point your browser to **nibs-sfdcindia-$$-prod.herokuapp.com** to check that the fork has worked.



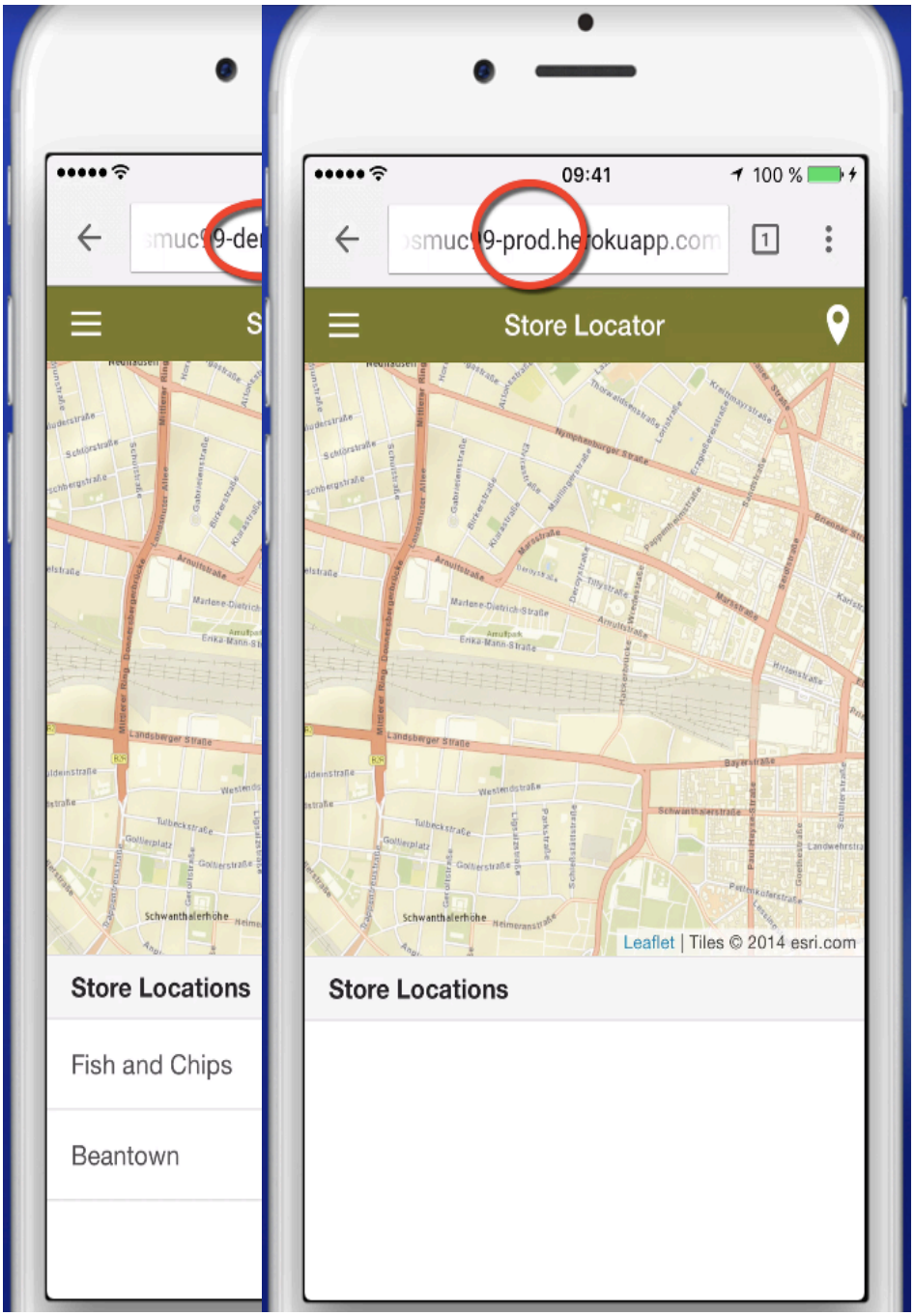
6.6 If you remember, in step 6.1 you fixed the source code, but you did not push it to Heroku, since we only had one app and it was in production. Now **nibs-sfdcindia-$$-demo** is our staging environment and the users would be using **nibs-sfdcindia-$$-prod**, so it's safe to push now to Heroku without affecting production.

Go back to the terminal, (make sure you are in the root directory of your source code) commit the changes on git and push them to Heroku:

```
git add .
git commit -m "fixed bug in the stores locator page"
git push heroku master
```



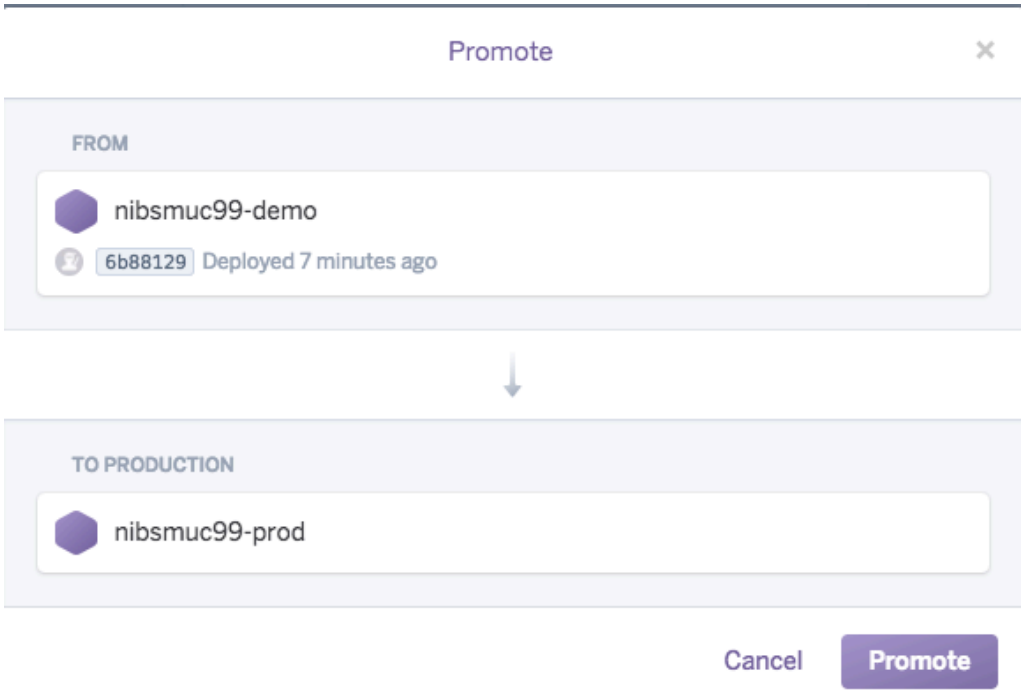
6.7 You can check now if your new code is working. Point your browser to **nibsmuc$$-demo.herokuapp.com**, click on the top left corner to show the menu and click on **Store Locator**. Can you see the list of the stores?



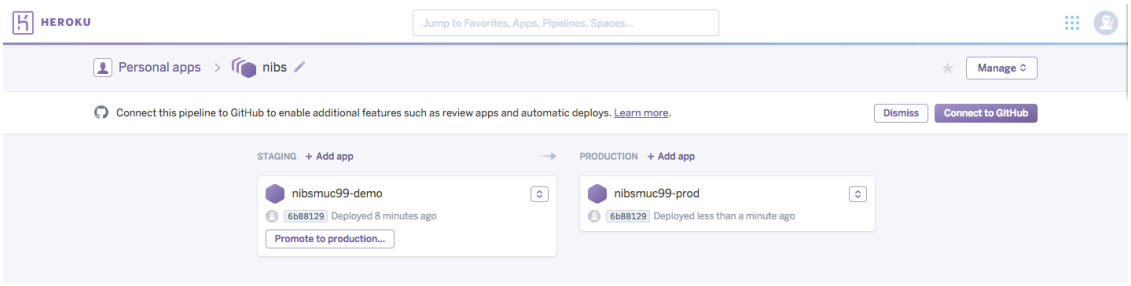
You did a great job! The test in staging is passed, so you can deploy to production for your users.

6.8 You can choose to promote your app to production using the CLI or the graphic interface.

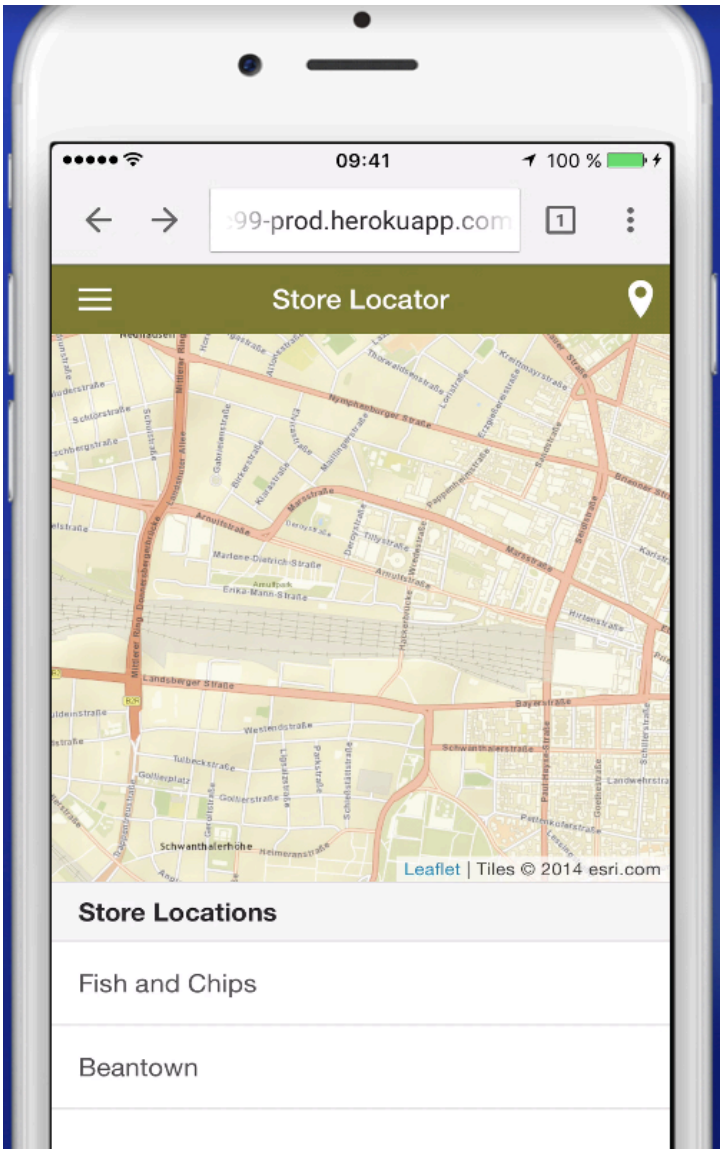
Go back to your **Heroku dashboard**, and in the nibs pipeline screen, click on **Promote to production...**



Confirm in the dialog that you want to promote from **nibs-sfdcindia-$$-demo** to **nibs-sfdcindia-$$-prod** clicking **Promote**.



And if you point your browser to **nibs-sfdcindia-$$-prod.herokuapp.com** you will see your application fixed, up and running for your users.



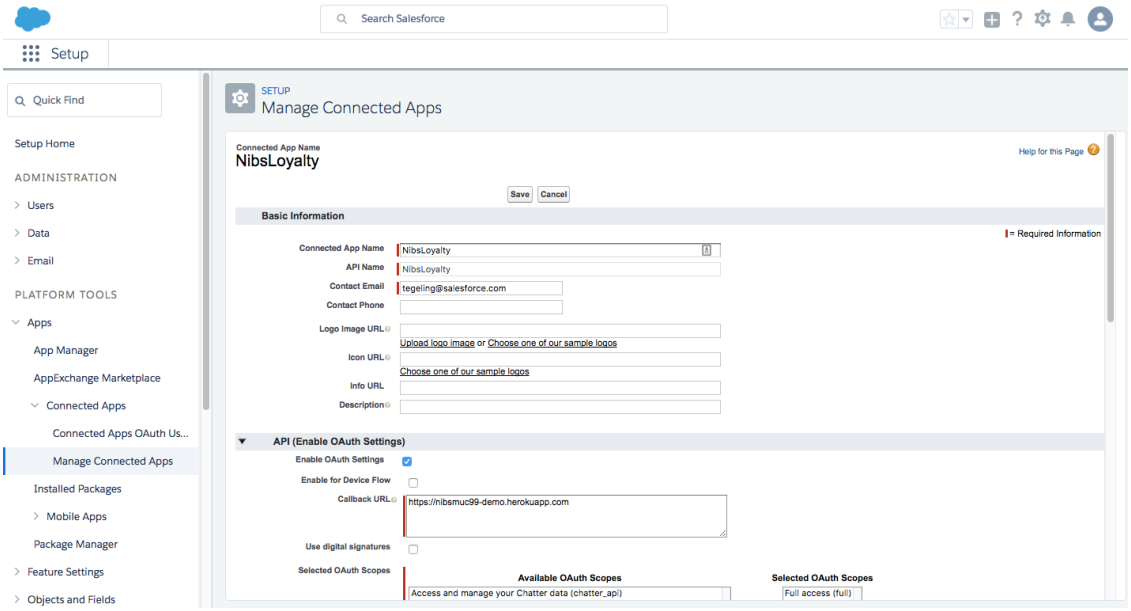
> ***Did you know:***
> *Heroku Pipelines is just a part of Heroku Flow, an easy to use structured workflow for continuous delivery.*
> *You can connect your Pipelines with your GitHub repo so you can deploy (either manually or automatically) a particular branch on every GitHub push.*
> *Heroku Review apps will spin up (manually or automatically) a temporary test app on an unique URL for every pull request if you are using GitHub integration.*
> ***Heroku Pipelines + Heroku Reviews + GitHub Integration = Heroku Flow**, the ultimate continuous integration tool.*
> *More information can be found here: **https://www.heroku.com/continuous-delivery***

Step 7 - Integration with Facebook

Do you want to earn bonus Elite points? If you remember from earlier in this session, the login with Facebook was not working. You can use OAuth to enable Facebook login with your application.

7.1 First, you need to create a connected app in your Salesforce org. **Go to Setup → Platform Tools → Apps → App Manager**, and click **New Connected Apps** in the upper right corner.

Fill in the **Connected App Name** (NibsLoyalty) and **Contact Email** (your email). Click on **Enable OAuth Settings** and set **https://nibsmuc$$-demo.herokuapp.com** as the **Callback URL**.
In the OAuth Scopes select **Full access** and the right arrow to add it and click **Save**.



Leave this screen open as you will need some data from it in a minute.

7.2 You have to add some configuration to your Heroku app so it has the OAuth credentials.

In a terminal type

```
heroku config:add OAUTH_CLIENT_ID='xxxxx'
```

(copy and paste the Consumer Key from your connected app in your Developer org)

```
heroku config:add OAUTH_CLIENT_SECRET='xxxxx'
```

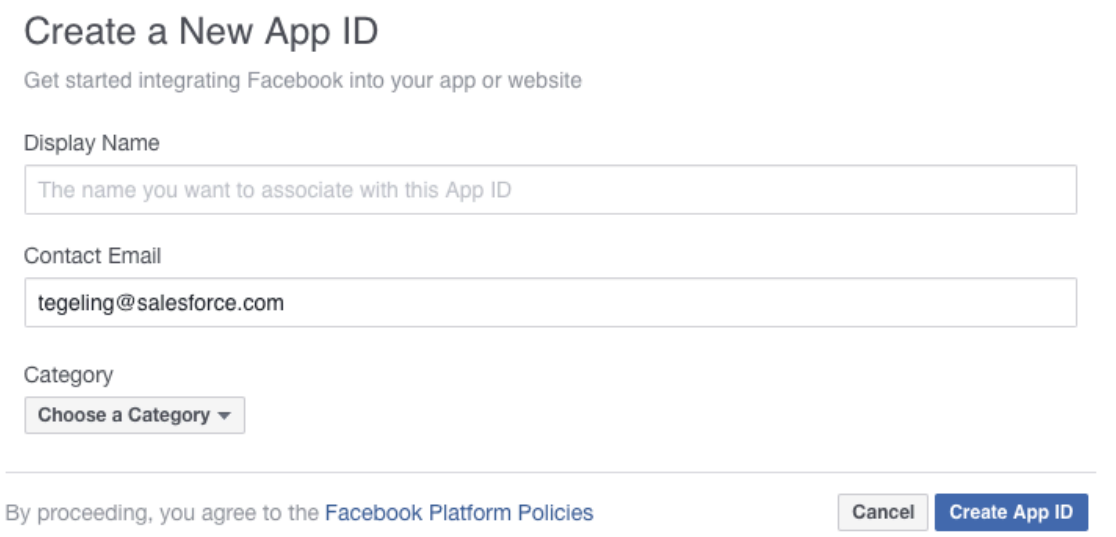
(copy and paste the Consumer Secret from your connected app in your Developer org)

```
heroku config:add OAUTH_REDIRECT_URL='https://nibsmuc$$-demo.herokuapp.com'
```

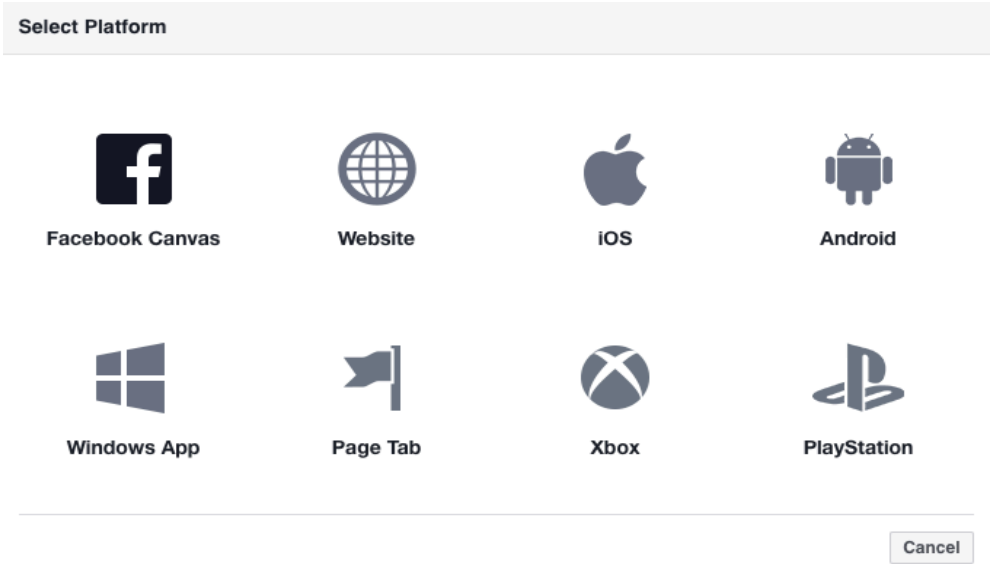
(change $$ for the random number you have been using during this session)

7.3 You will now create a Facebook app. Go to **https://developers.facebook.com** and login with your regular Facebook user. If you have never used the developer site you will be asked to agree with the terms and conditions.

Click on the **My apps** drop down menu and select **Add a New App**.



Enter a name for your app (**nibs-sfdcindia-$$-** where $$ is your random number sounds right!), specify a category such as **Food & Drink**, and click

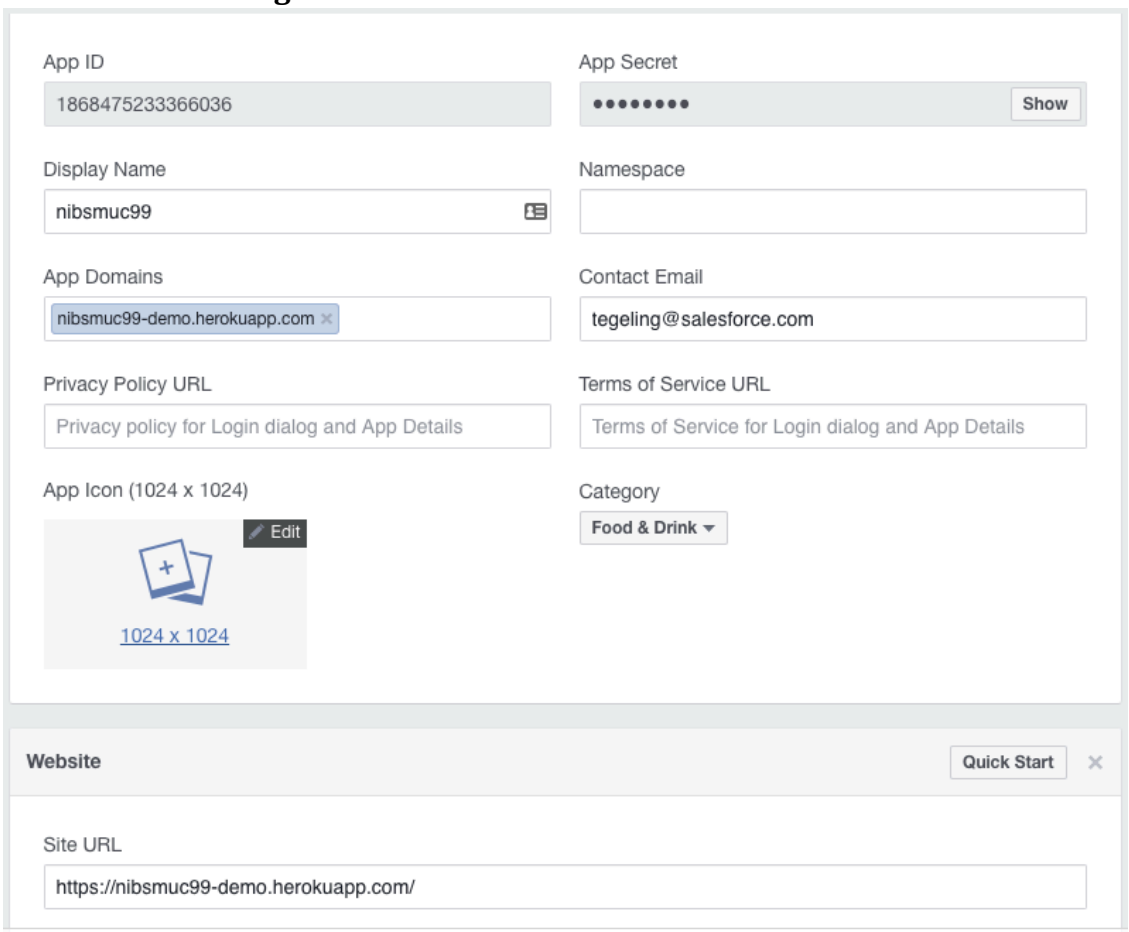


Create App ID.

On the next screen open the **Settings/Basic** tab and add a platform. Select **Website**.

Fill in the Site URL as **https://nibs-sfdcindia-$$-demo.herokuapp.com**

Fill in the App Domains field with **nibs-sfdcindia-$$-demo.herokuapp.com**. Make sure that you have an **AppID**, a **Secret** and that the **SiteURL** is correct. Click on **Save Changes**.



You are ready to make the app public. On the menu on the left click **App Review**, toggle the **Make Public** slide to Yes and **Confirm**.

7.4 Do you mind getting your hands dirty a bit for the next step?

Open the **client/js/config.js** on the directory where the source code of the app is in your laptop.

Change the following line (line 5):

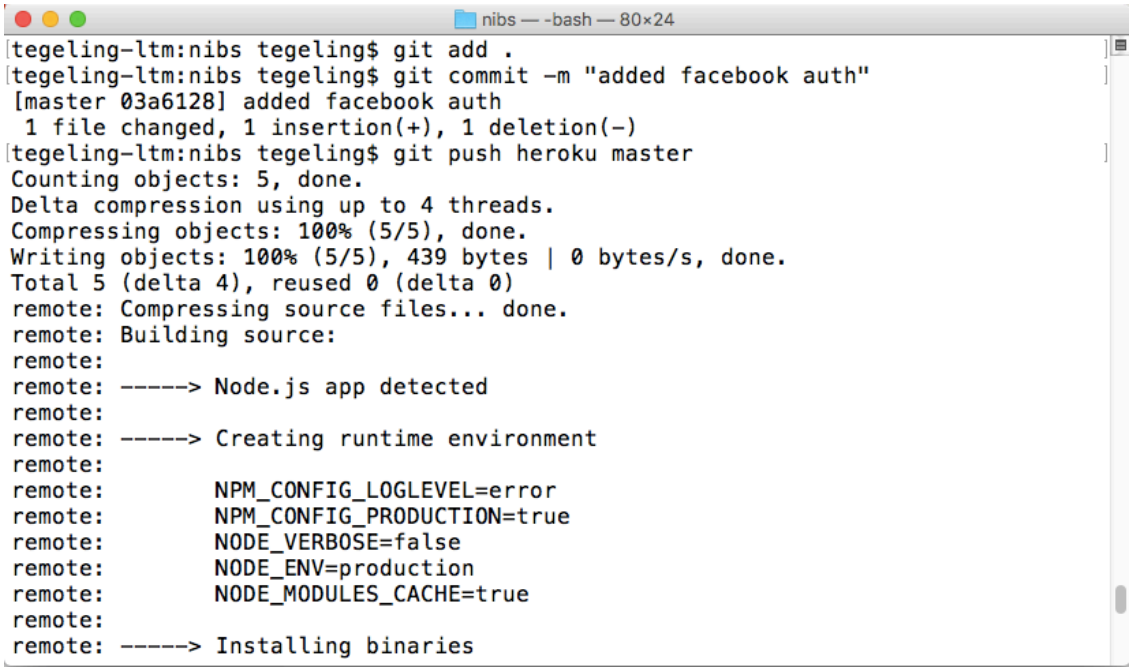
```
    .constant('FB_APP_ID','539135216271806')
```

(you'll find your APP ID in the basic settings page on Facebook)

You have made changes in the app source code, so you'll have to check them in and push them to Heroku. Your app will be recompiled and a new version will be deployed automatically!!

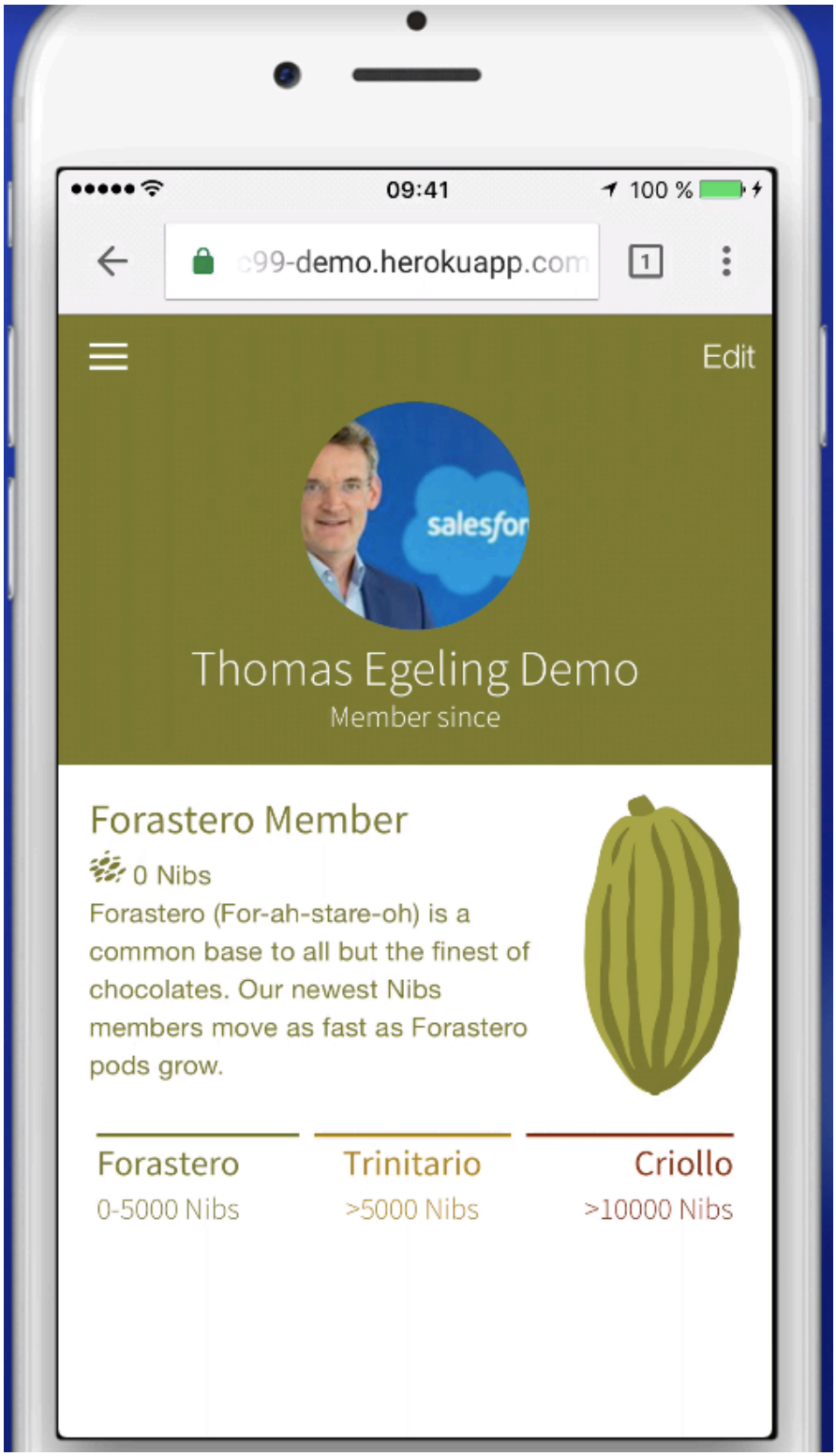
On a terminal window in the same directory where your source code is, type:

```
git add .
git commit -m "added facebook auth"
git push heroku master
```



7.5 Are you ready to try the Facebook integration?

Logout from your Nibs app and try logging in again, but this time use the **Login with Facebook** button. Type in your Facebook credentials and allow access to your profile (don't worry, the app will not publish anything on your profile).



Now you are logged in with your facebook user!!! You can even check in your Salesforce org that a new **Contact** has been created with your Facebook user!

Recap

In this brief session you…

- Set up a Salesforce org and prepared it to work in sync with a Heroku app
- Deployed a heroku app (with just one command line!)
- Deployed and configured a Postgres database (with just one command line!)
- Deployed and configured Heroku Connect to sync data with Salesforce
- Used Heroku Flow to stage an app that had a bug and fixed it
- Configured OAuth on Salesforce and Heroku to login with your Facebook