Cloud Developer Certification Preparation

Exercise 4.2:

Building your first app external authentication by using the IBM Single Sign on service in Bluemix

Exercise 4.2: Overview

When you develop an application that supports multiple users, you need to decide how to authenticate users. It might seem like a simple approach to create a database table and populate it with credentials, but this quickly creates further challenges. For one, how will your application be able to integrate with existing applications and their authentication services?

This exercise introduces the IBM Single Sign On (SSO) service for IBM Bluemix that you can use to configure your web application to authenticate users from social media sites, enterprise directories through SAML federation, and also a LDAP-based cloud identity source.

This exercise will show you how to add and configure the IBM Single Sign On service for Bluemix to the Node.js starter application. It provides instructions to create the application, add the service to the Bluemix workspace, configure a cloud identity source, bind and integrate the service into the application, and modify the application code to use external authentication.

Prerequisites

Sign up for a 30-day free trial IBM Bluemix account if you don't already have one.

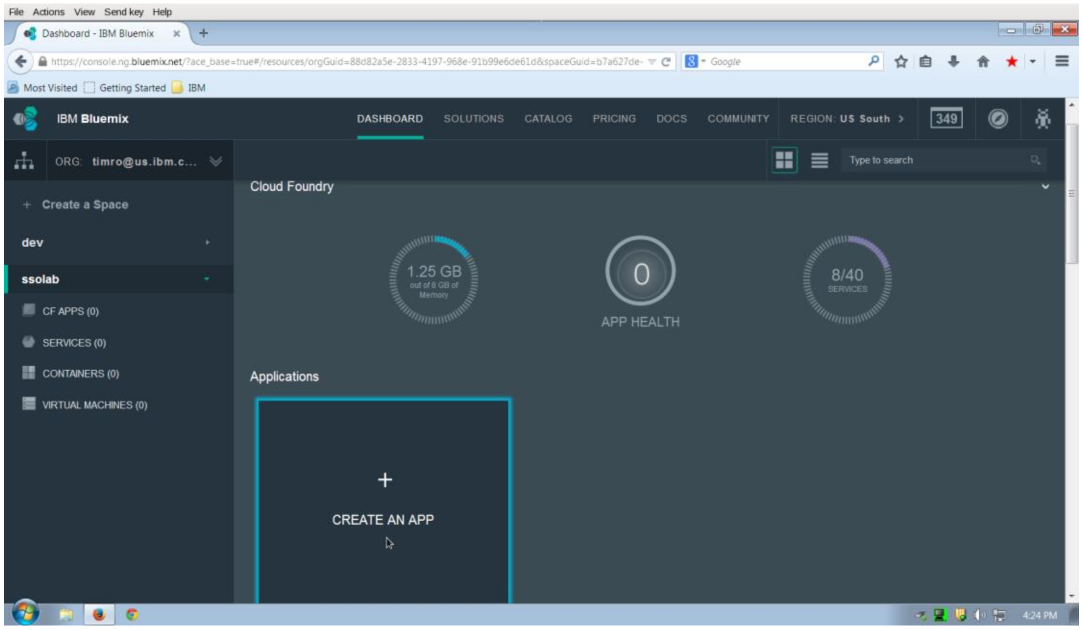
You also need the following software:

- A web browser supported by Bluemix. You should use one browser for working with the Bluemix and Single Sign On control panel in one browser and a separate browser for testing your application.
    - Chrome: the latest version for your operating system
    - Firefox: the latest version for your operating system and ESR 31 or ESR 38
    - Internet Explorer: version 10 or 11
    - Safari: the latest version for the Mac
- Node.js version 0.12.x: download from: https://nodejs.org/download/
- Cloud Foundry CLI tool version 6.9 or later

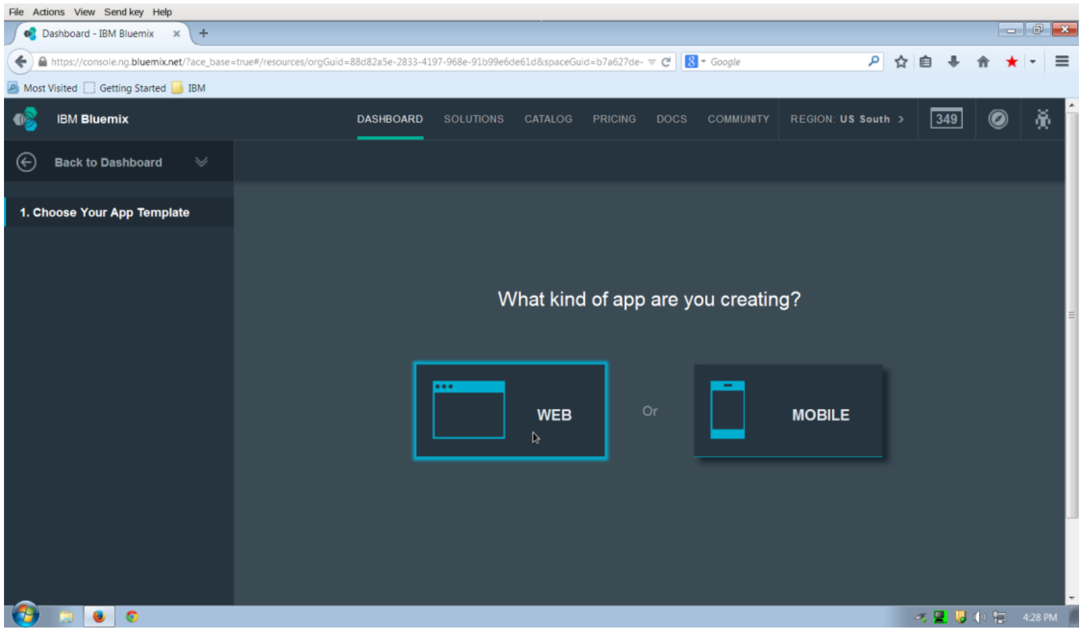
The app.js source code shown in the screen captures is provided at the end of this exercise if you don't want to enter the code for a particular step.

Exercise 4.2.1: Creating a Node.js starter application and adding the Single Sign On service

1. Sign in to Bluemix and go to the Dashboard. Under Cloud Foundry, click on the “CREATE AN APP” button to get started:

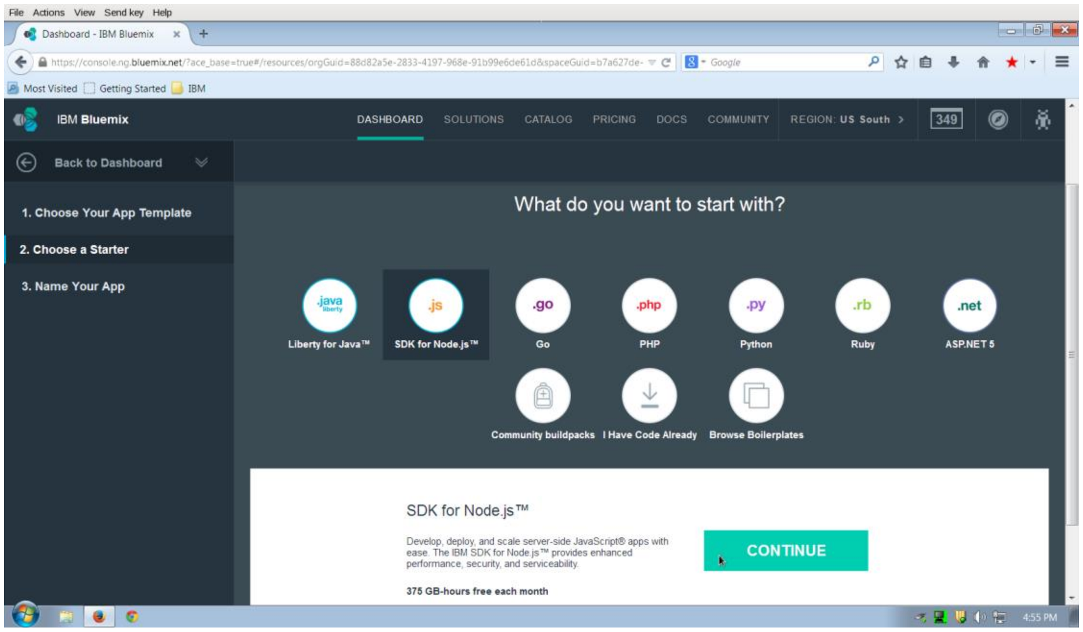


2. Click **WEB**.


IBM Bluemix
Back to Dashboard
1. Choose Your App Template
What kind of app are you creating?
WEB
Or
MOBILE

Next, you will need to select the runtime for the application.

3. For this exercise, select the **SDK for Node.js** and confirm by clicking on **CONTINUE**.

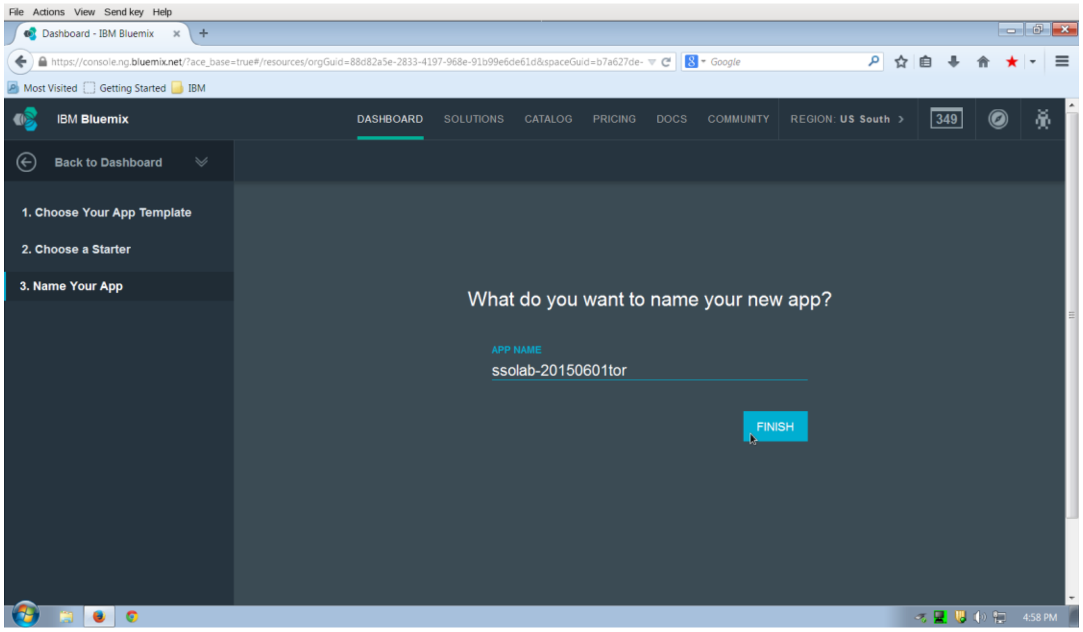


SDK for Node.js™

Develop, deploy, and scale server-side JavaScript® apps with ease. The IBM SDK for Node.js™ provides enhanced performance, security, and serviceability.

375 GB-hours free each month

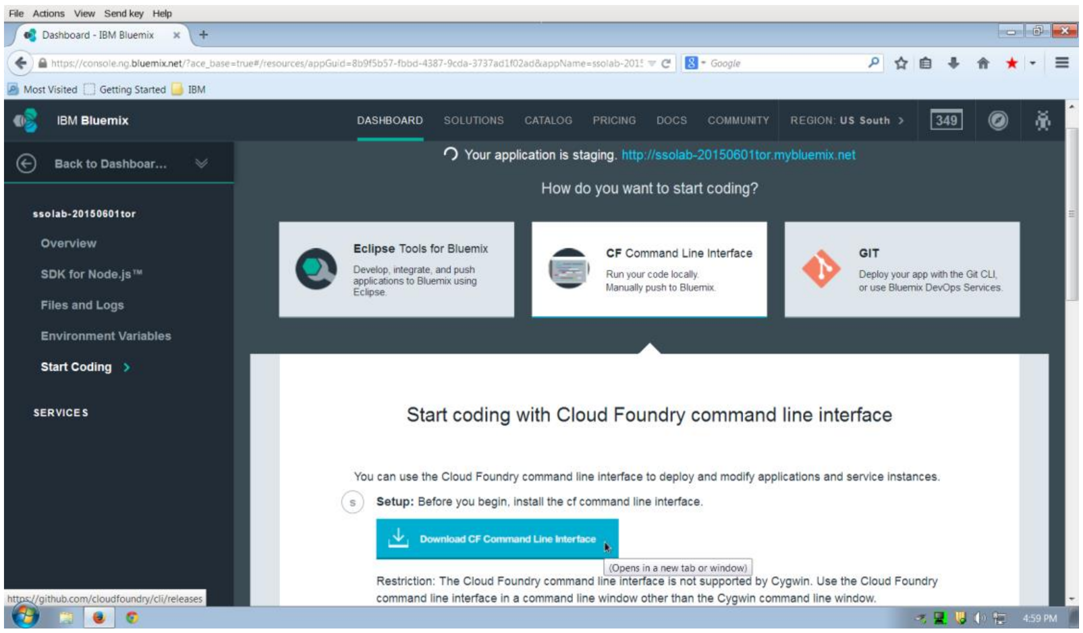
4. To complete adding the new application, provide a name. Each application in Bluemix public requires a unique name. To ensure that your app name is unique, add the date and your initials to the name. Then, click **FINISH.**



APP NAME
ssolab-20150601tor

Bluemix will show the application's Start Coding panel with a message at the top about the application starting the staging process.

5. If the Cloud Foundry CLI tool is not installed on your workstation, click the **Download** button to download the current version for your workstation. It will be later than 6.9 but that is ok. After the download is complete, install the cf CLI tool.


File Actions View Send key Help
Dashboard - IBM Bluemix
https://console.ng.bluemix.net/?ace_base=true#/resources/appGuid=8b9f5b57-fbbd-4387-9cda-3737ad1f02ad&appName=ssolab-201!
Google
Most Visited
Getting Started
IBM
IBM Bluemix
DASHBOARD
SOLUTIONS
CATALOG
PRICING
DOCS
COMMUNITY
REGION: US South
349
Back to Dashboar...
ssolab-20150601tor
Overview
SDK for Node.js™
Files and Logs
Environment Variables
Start Coding
SERVICES
Download CF Command Line Interface
(Opens in a new tab or window)
https://github.com/cloudfoundry/cli/releases
4:59 PM

Your application is staging. http://ssolab-20150601tor.mybluemix.net

How do you want to start coding?

Eclipse Tools for Bluemix
Develop, integrate, and push applications to Bluemix using Eclipse.

CF Command Line Interface
Run your code locally.
Manually push to Bluemix.

GIT
Deploy your app with the Git CLI, or use Bluemix DevOps Services.

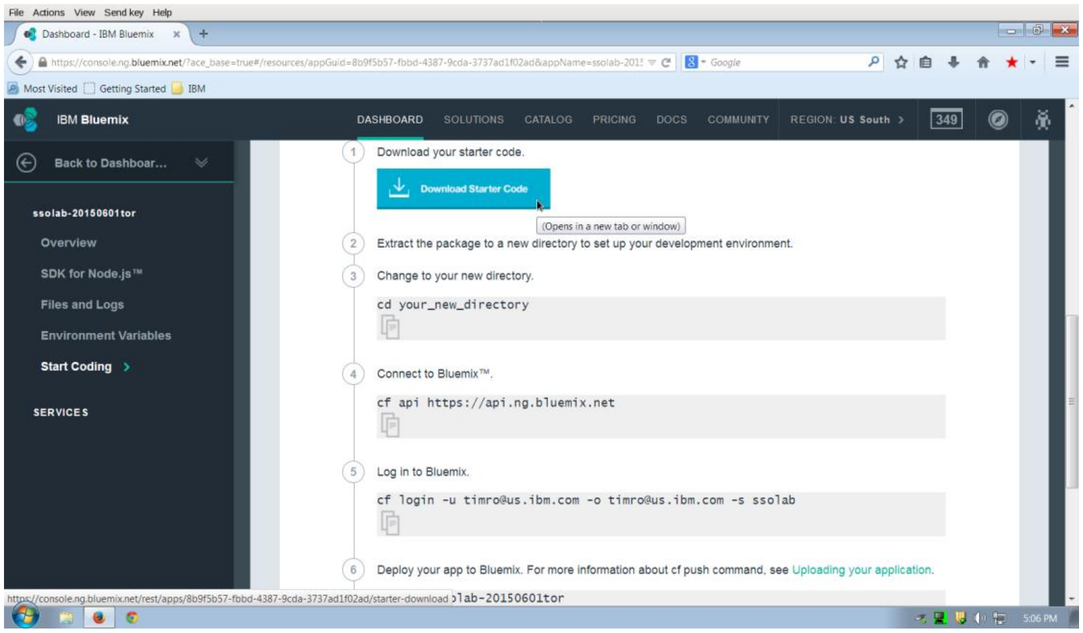
Start coding with Cloud Foundry command line interface

You can use the Cloud Foundry command line interface to deploy and modify applications and service instances.

s **Setup:** Before you begin, install the cf command line interface.

Restriction: The Cloud Foundry command line interface is not supported by Cygwin. Use the Cloud Foundry command line interface in a command line window other than the Cygwin command line window.

6. Scroll down on the same panel to the series of numbered steps starting with the Download Starter Code button. Make a note of these steps (they will not match the picture and instead will be customized for your Bluemix id, current Bluemix space and application name). Complete steps 1, 2 and 3 as shown to get a command prompt with a copy of your code. You will use this command prompt to add the node **passport** module to the starter application later.



1 Download your starter code.

2 Extract the package to a new directory to set up your development environment.

3 Change to your new directory.

```
cd your_new_directory
```

4 Connect to Bluemix™.

```
cf api https://api.ng.bluemix.net
```

5 Log in to Bluemix.

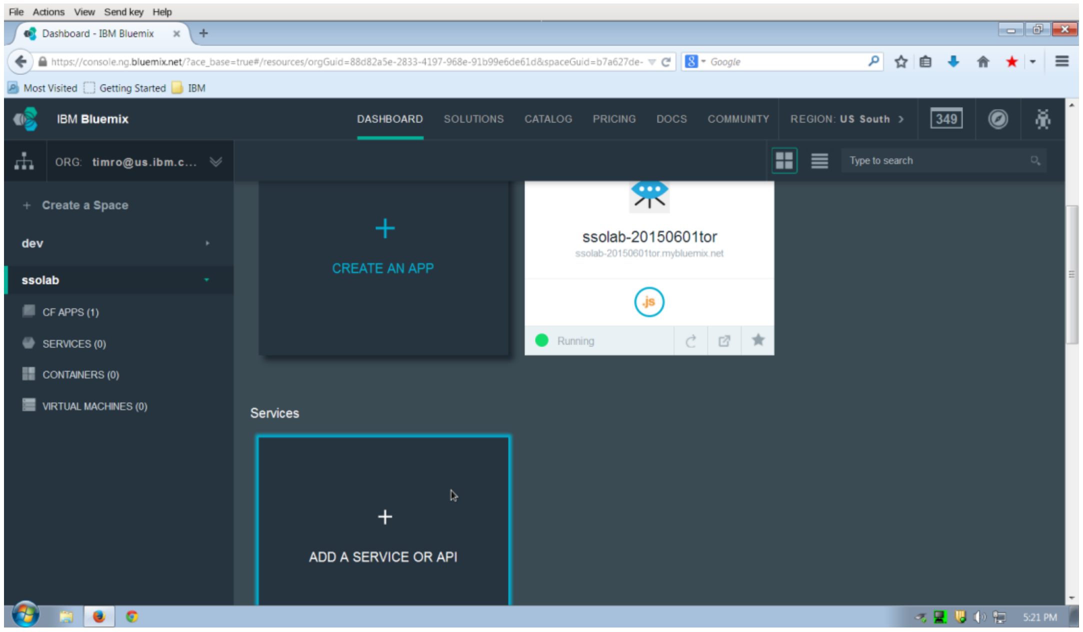
```
cf login -u timro@us.ibm.com -o timro@us.ibm.com -s ssolab
```

6 Deploy your app to Bluemix. For more information about cf push command, see Uploading your application.

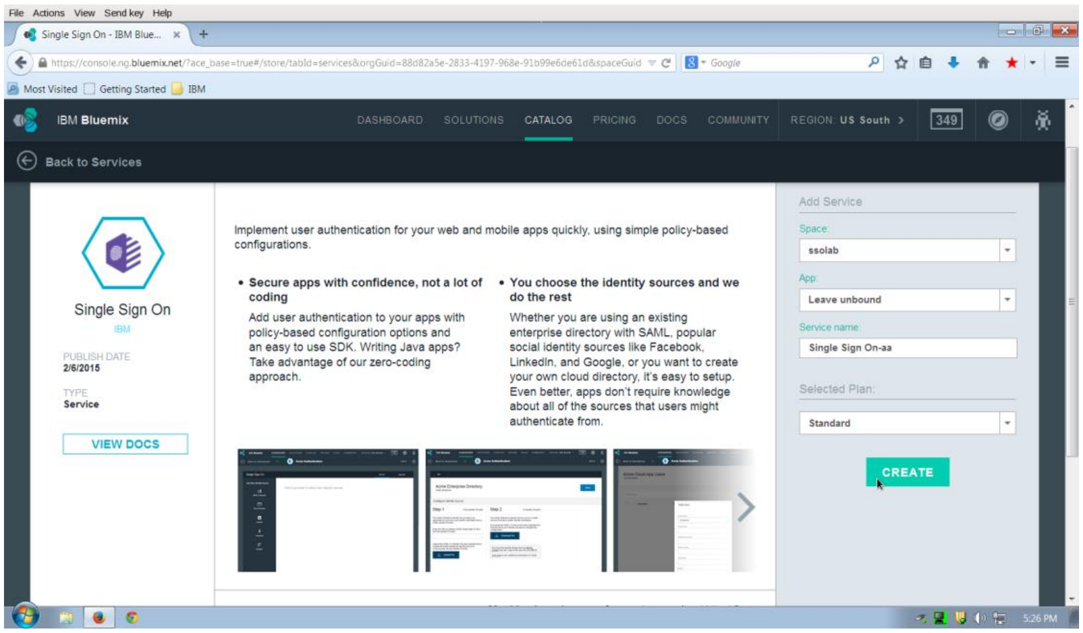
Next, we will add the Single Sign On service to the Dashboard.

7. Select an ID that will become the prefix of the URL for the service when used when applications redirect users to the service for selecting an identity provider, and also for handling callbacks from identity providers. This id can be up to 32 characters, and must start with an alphanumeric character. Make a note of it here: ____

8. In the Bluemix Cloud Foundry Dashboard, select **Add a Service**.



9. Scroll down in the services list to the Security section and then click the **Single Sign On** icon. This will show a configuration window for adding the service. On the right side, select **Leave unbound** under the **App:** section. You can change the service name that will be shown in the dashboard or leave the default. Then, click **CREATE**.



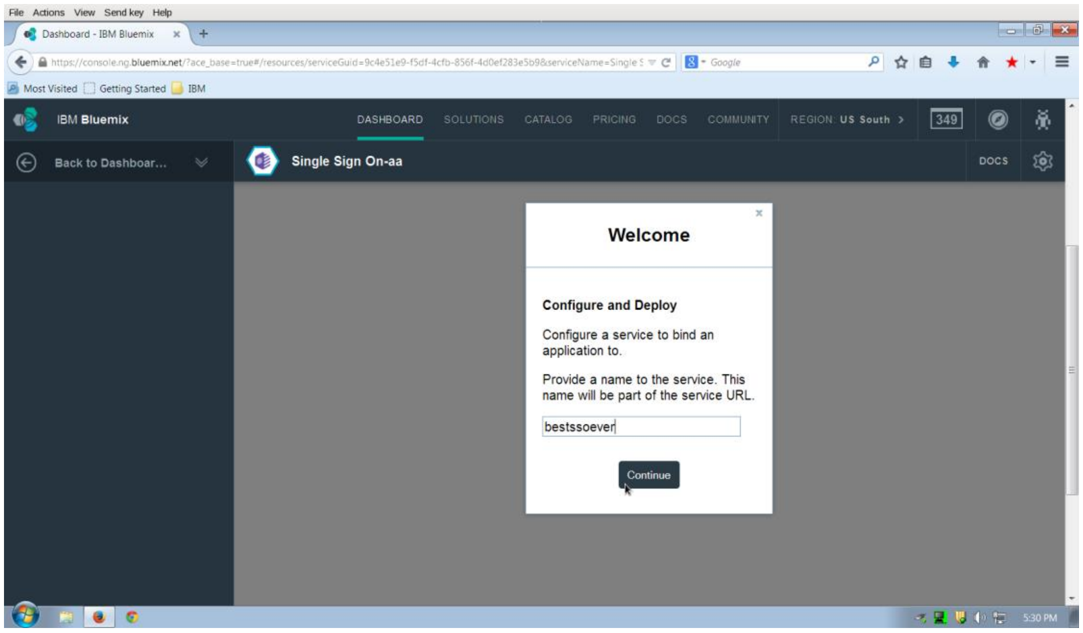
Single Sign On
IBM
PUBLISH DATE
2/6/2015
TYPE
Service

Implement user authentication for your web and mobile apps quickly, using simple policy-based configurations.

- Secure apps with confidence, not a lot of coding
  Add user authentication to your apps with policy-based configuration options and an easy to use SDK. Writing Java apps? Take advantage of our zero-coding approach.
- You choose the identity sources and we do the rest
  Whether you are using an existing enterprise directory with SAML, popular social identity sources like Facebook, LinkedIn, and Google, or you want to create your own cloud directory, it's easy to setup. Even better, apps don't require knowledge about all of the sources that users might authenticate from.

Add Service
Space: ssolab
App: Leave unbound
Service name: Single Sign On-aa
Selected Plan: Standard

10. Enter a name for the service. This is the ID that you chose earlier. It is not the name that was shown in the previous window as the Service name. Click **Continue**.



Welcome

Configure and Deploy

Configure a service to bind an application to.

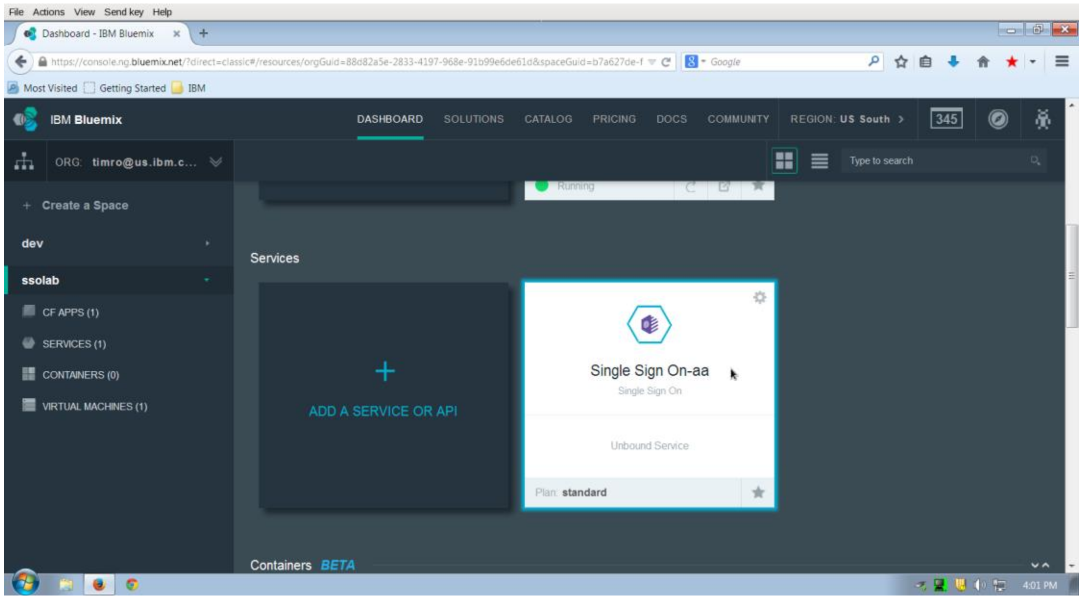
Provide a name to the service. This name will be part of the service URL.

bestssoever

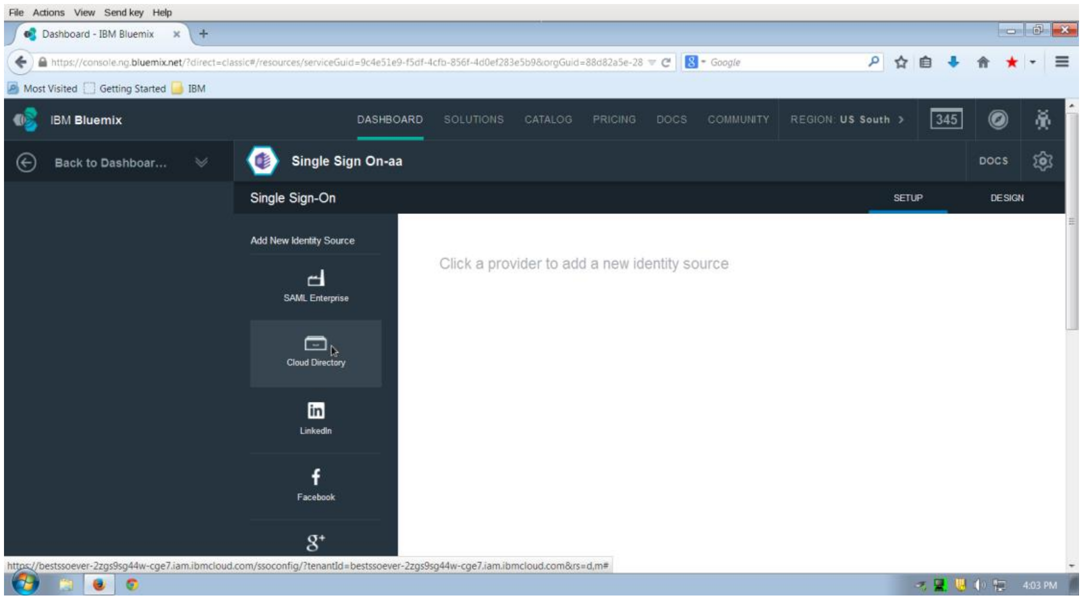
This will add the service to the dashboard. Now, you can create a simple cloud directory and add a test user for external authentication in the next exercise.

Exercise 4.2.2: Configuring a cloud directory identity source

1. In the IBM Bluemix Cloud Foundry dashboard, scroll down to find the SSO service that you just created. Click the service to open the configuration window.

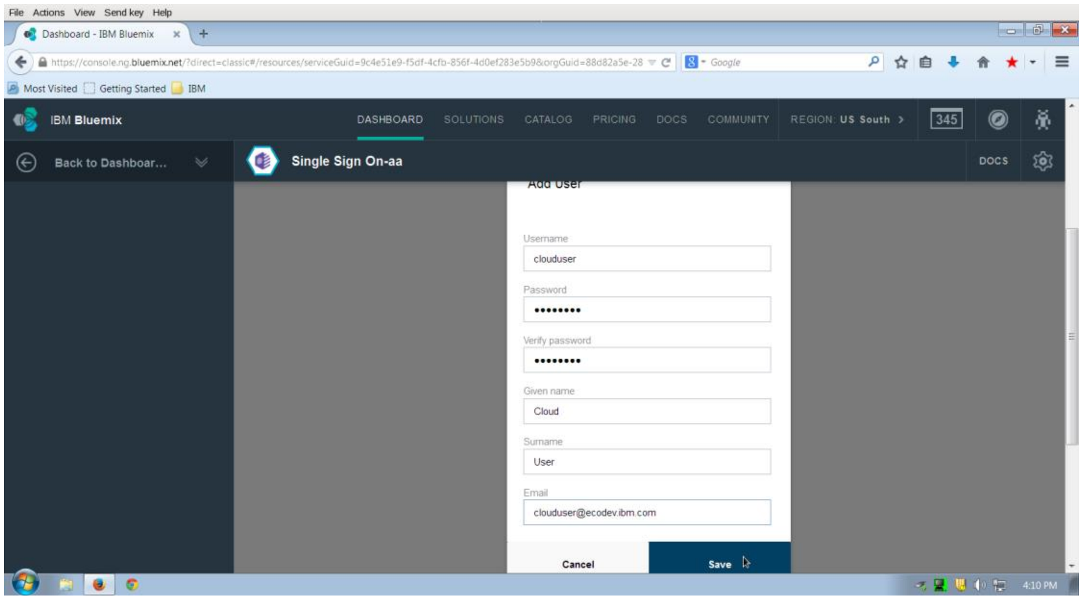


2. In the configuration window, click the Cloud Directory icon to add and configure a directory in the cloud for this exercise.

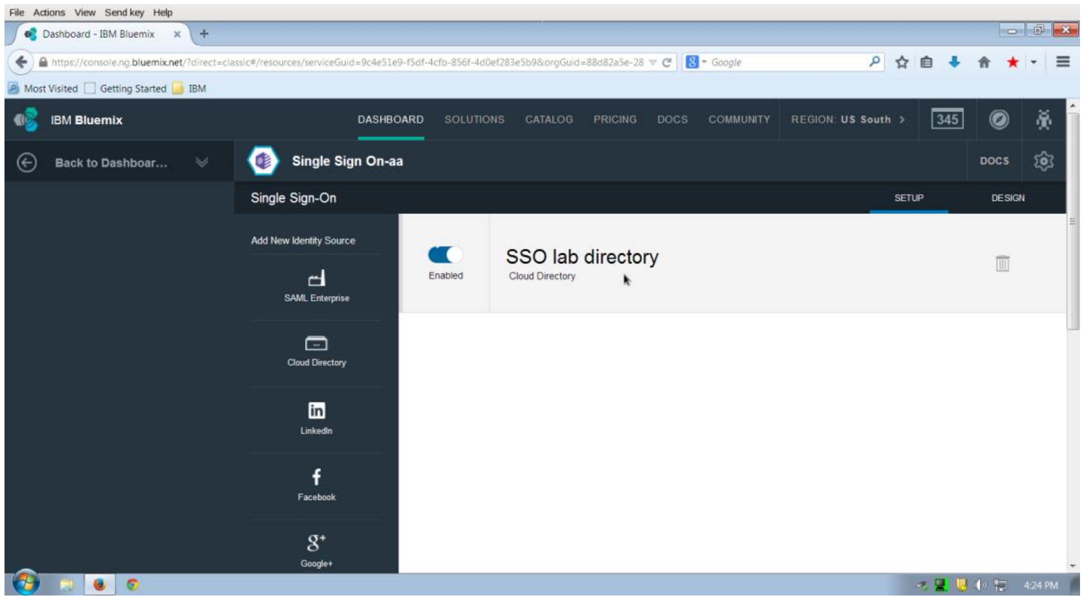


3. Optional: In the next screen, enter a new name for the identity source other than the default and see the current list of users. It's empty at first, so click the **Add** icon.

4. Add some information for a test user. This can be any user information that you prefer. However, make a note of the user ID and password because those credentials will be used when you test the external authentication service.



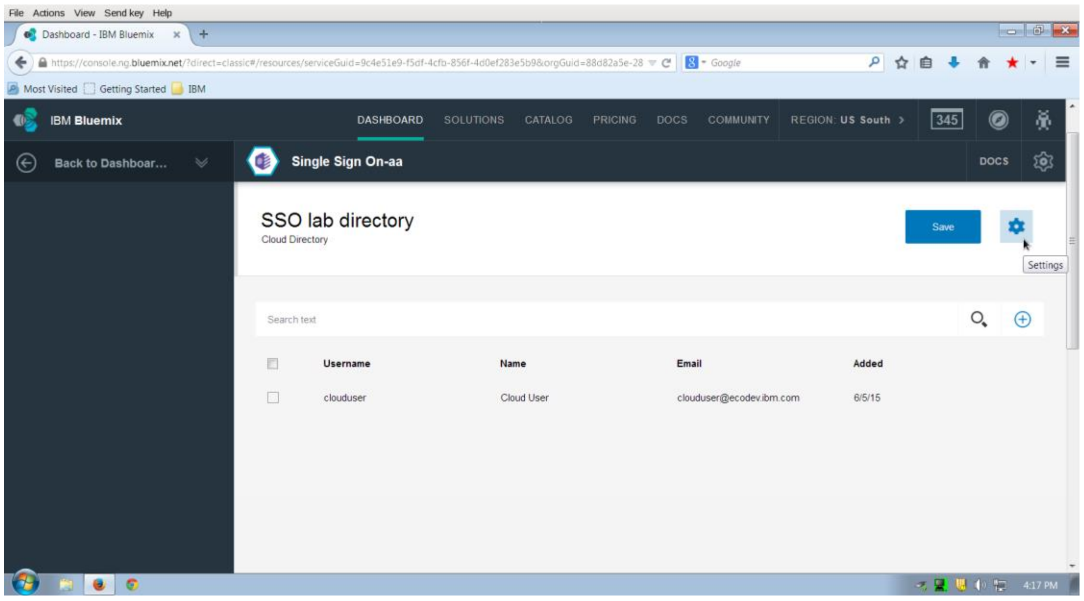
Username: clouduser
Password: ••••••••
Verify password: ••••••••
Given name: Cloud
Surname: User
Email: clouduser@ecodev.ibm.com



5. Save the user and the identity source configuration. Then, re-open the service by selecting it.

After re-opening the identity source configuration panel, you'll see a **Settings** icon in the upper right section of the window.

6. To manage the password policy that is applied to users in this identity source, click the **Settings** icon for the identity source.



SSO lab directory
Cloud Directory

Username	Name	Email	Added
clouduser	Cloud User	clouduser@ecodev.ibm.com	6/5/15

7. Leave the **Auto Consent** setting to **On** for this exercise to observe this prompt in action. The settings control has two option windows. The **Auto Consent** option tells the SSO service whether it should prompt the user after an authentication to allow the SSO service to retrieve personal information, such as name and email address, from the identity source.

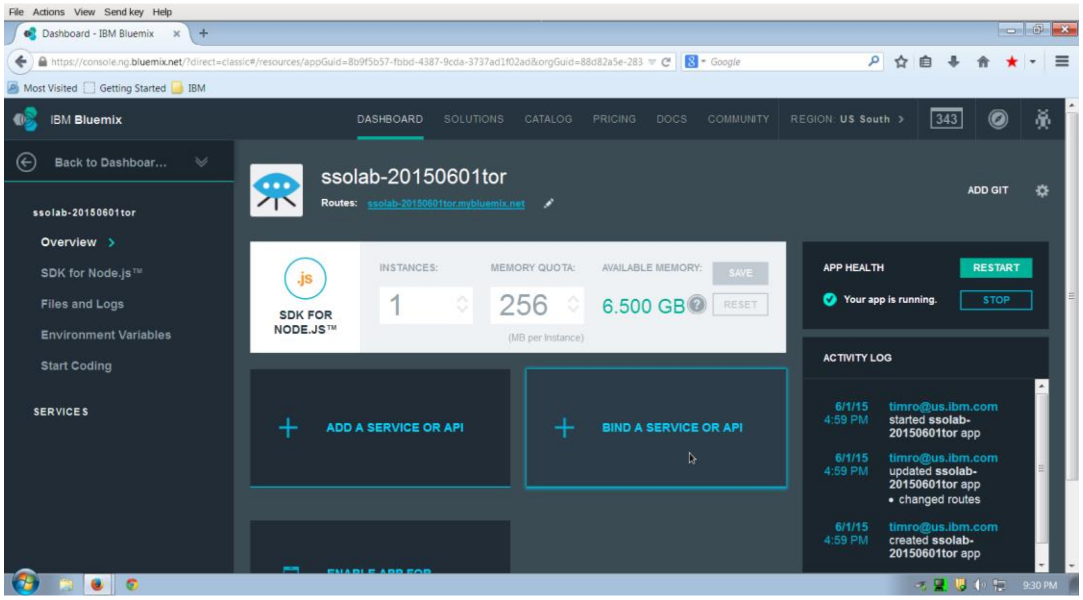
8. Select **Medium** for the Password Policy. The **Password Policy** option allows you to select various levels of password quality and lifetime. Currently, these policies are fixed, and you must select from one of the three. Then, click **Save**.

9. Click **Save** in the Identity Source Configuration window to return to the main SSO configuration window.

   The name shown for the identity source will be the name that you selected or the default name of cloud directory. The toggle (shown as **Enabled**) next to the name allows the service administrator to enable or disable a particular identity source for use by applications. Leave the toggle at **Enabled** and click **Back to Dashboard** in the upper left to return to the IBM Bluemix Cloud Foundry dashboard.

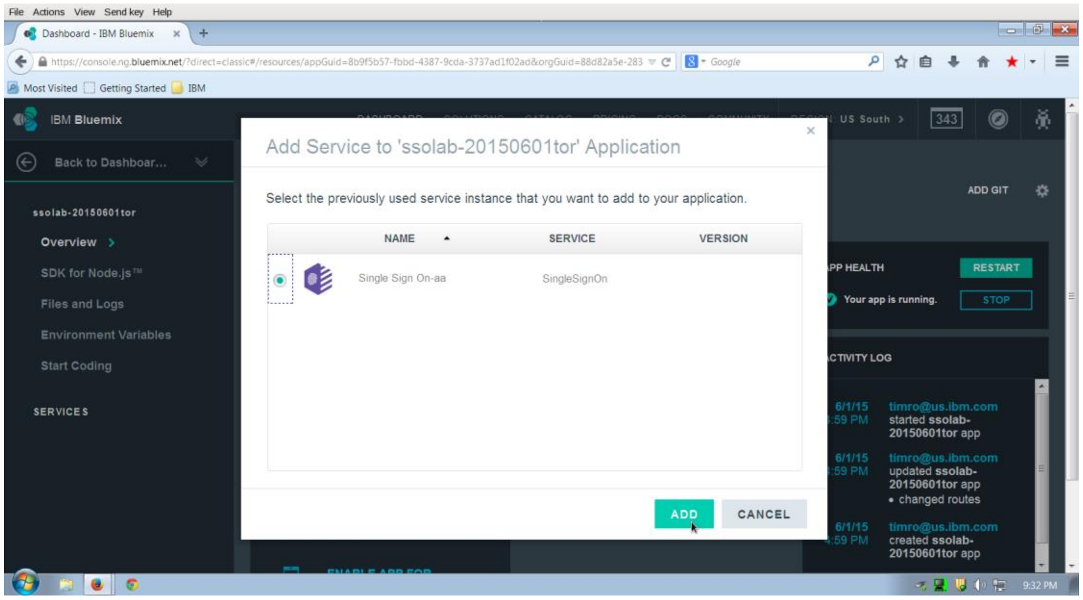
Exercise 4.2.3: Binding the Single Sign On service to the application

1. To bind the Single Sign On service with the configured identity source, click the application icon in the dashboard to see the application overview page. Then, click **Bind a Service or API.**



6/1/15 4:59 PM timro@us.ibm.com started ssolab-20150601tor app
6/1/15 4:59 PM timro@us.ibm.com updated ssolab-20150601tor app • changed routes
6/1/15 4:59 PM timro@us.ibm.com created ssolab-20150601tor app

2. Select the **Single Sign On service** radio button and then click **ADD**.

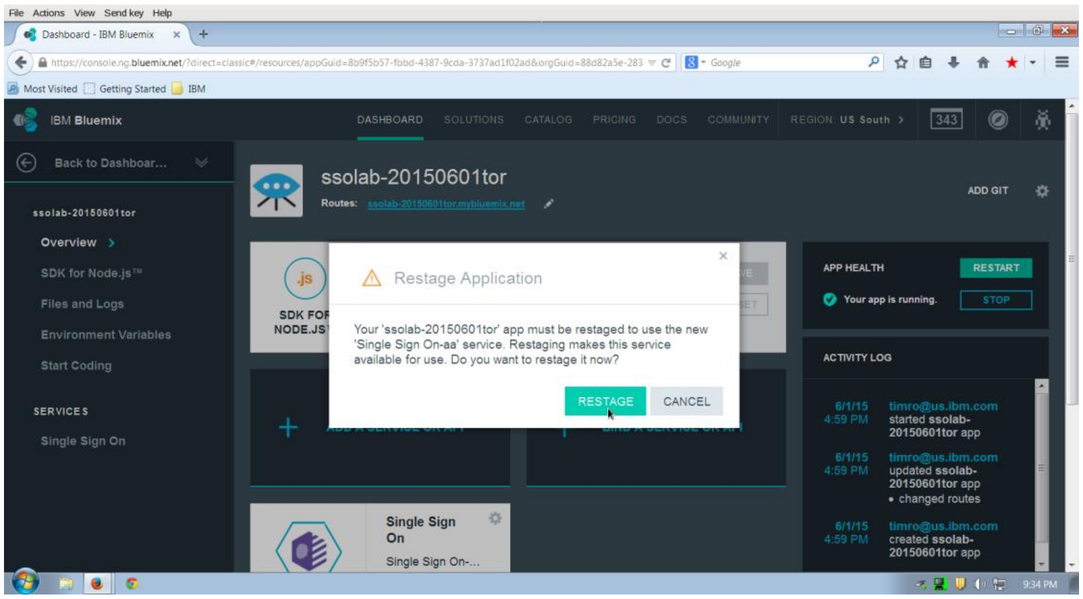

Add Service to 'ssolab-20150601tor' Application

Select the previously used service instance that you want to add to your application.

	NAME	SERVICE	VERSION
●	Single Sign On-aa	SingleSignOn	

You will be prompted to restage the application. A restage is used to add the VCAP_SERVICES environment variables to the application so that it will be able to access the Single Sign On service bound to the application.

3. Click **RESTAGE**.



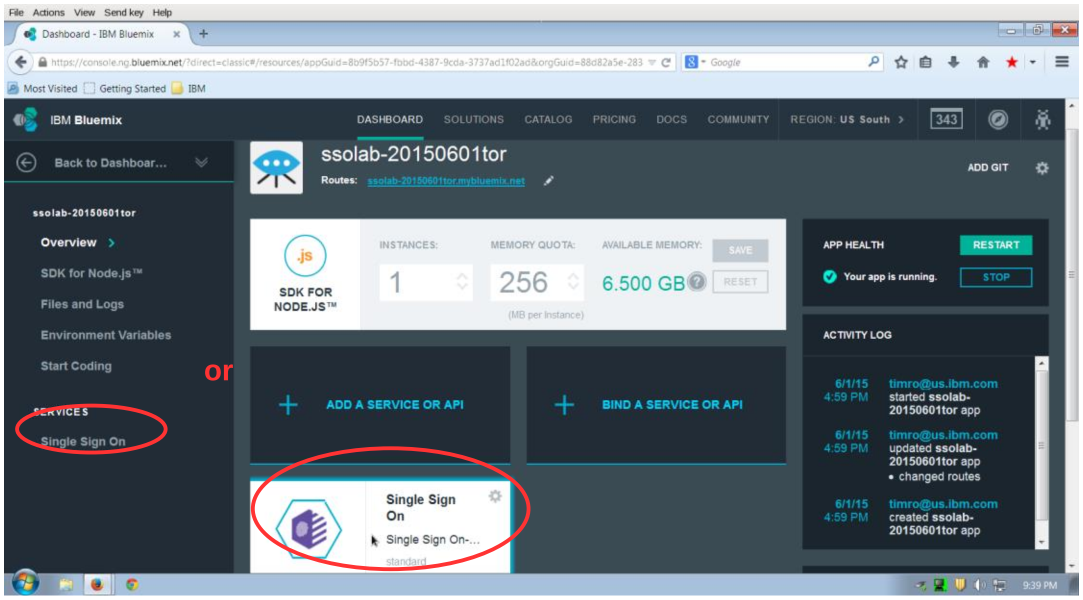
Your 'ssolab-20150601tor' app must be restaged to use the new 'Single Sign On-aa' service. Restaging makes this service available for use. Do you want to restage it now?

Your app is running.

```
6/1/15 4:59 PM  timro@us.ibm.com started ssolab-20150601tor app
6/1/15 4:59 PM  timro@us.ibm.com updated ssolab-20150601tor app
                • changed routes
6/1/15 4:59 PM  timro@us.ibm.com created ssolab-20150601tor app
```

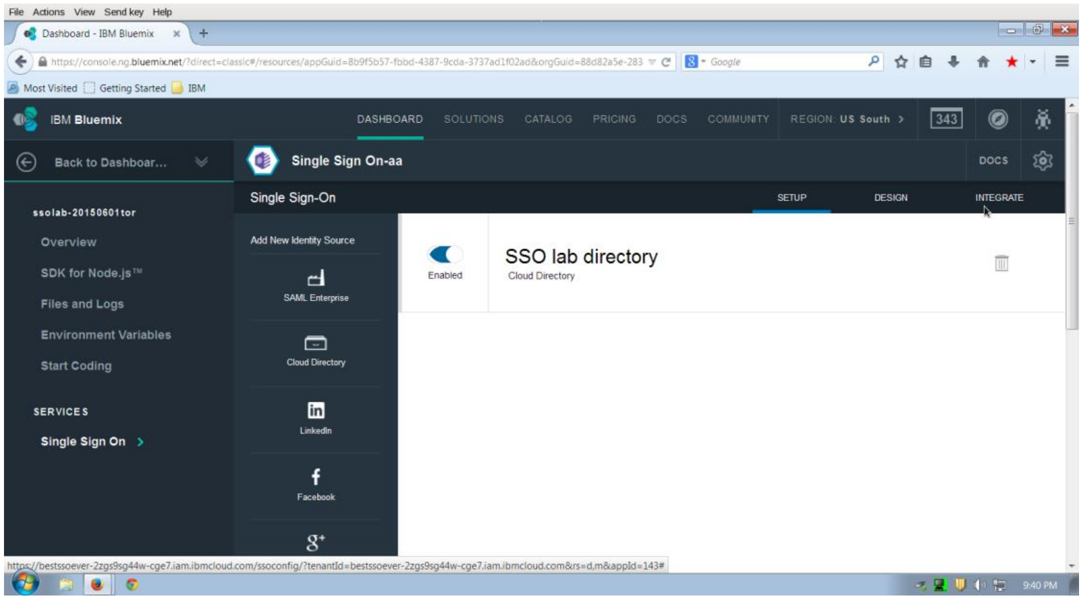
The restaging and restarting of the application will complete quickly. However, you do not need to wait before proceeding.

4. On the left side of the application's overview window, click **Single Sign On** or click on the icon for the service. This will open the configuration window for the service in the context of this application.

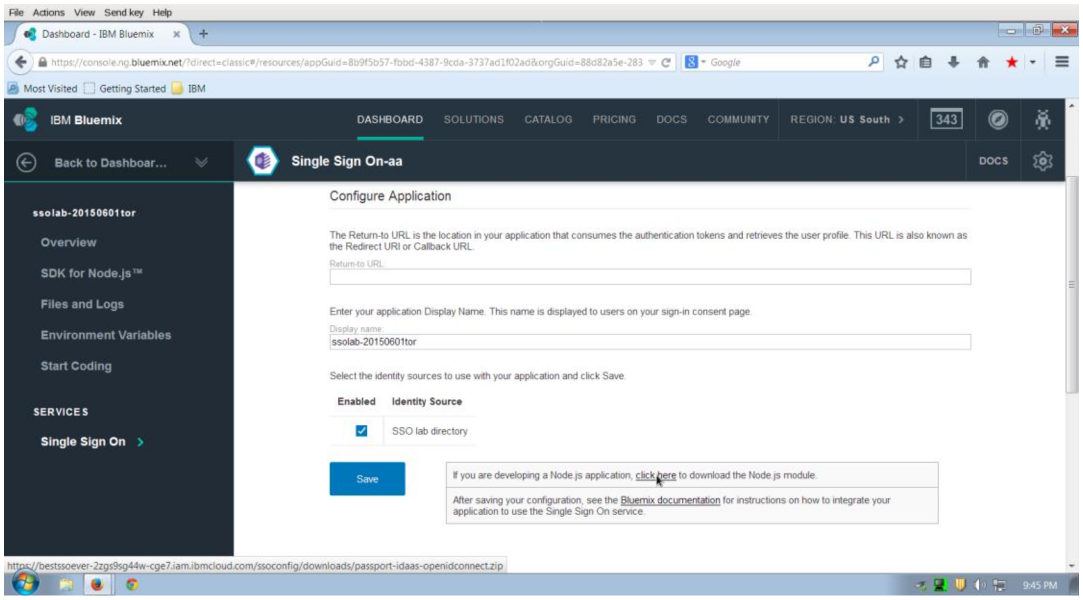


When you access the configuration window from within an application, an **INTEGRATE** tab will also be shown. For the Node.js application that you are configuring, a module needs to be downloaded from the Single Sign On service. The link to this code is available from the **INTEGRATE** tab.

5. Although you will return later to the **INTEGRATE** tab to finish configuring the application, switch to the **INTEGRATE** tab now:



6. Download the code for an additional Node.js module for your application that will provide support for OpenID Connect protocol for the passport service. To download this code, find the **click here** link in the light gray text box. Then, click the link to download the module.


File Actions View Send key Help
Dashboard - IBM Bluemix
https://console.ng.bluemix.net/?direct=classic#/resources/appGuid=8b9f5b57-fbbd-4387-9cda-3737ad1f02ad&orgGuid=88d82a5e-283
Most Visited Getting Started IBM
IBM Bluemix
DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY REGION: US South
Back to Dashboar...
Single Sign On-aa
DOCS
ssolab-20150601tor
Overview
SDK for Node.js™
Files and Logs
Environment Variables
Start Coding
SERVICES
Single Sign On

Configure Application

The Return-to URL is the location in your application that consumes the authentication tokens and retrieves the user profile. This URL is also known as the Redirect URI or Callback URL.

Return-to URL

Enter your application Display Name. This name is displayed to users on your sign-in consent page.

Display name: ssolab-20150601tor

Select the identity sources to use with your application and click Save.

Enabled	Identity Source
✓	SSO lab directory

Save

If you are developing a Node.js application, click here to download the Node.js module.

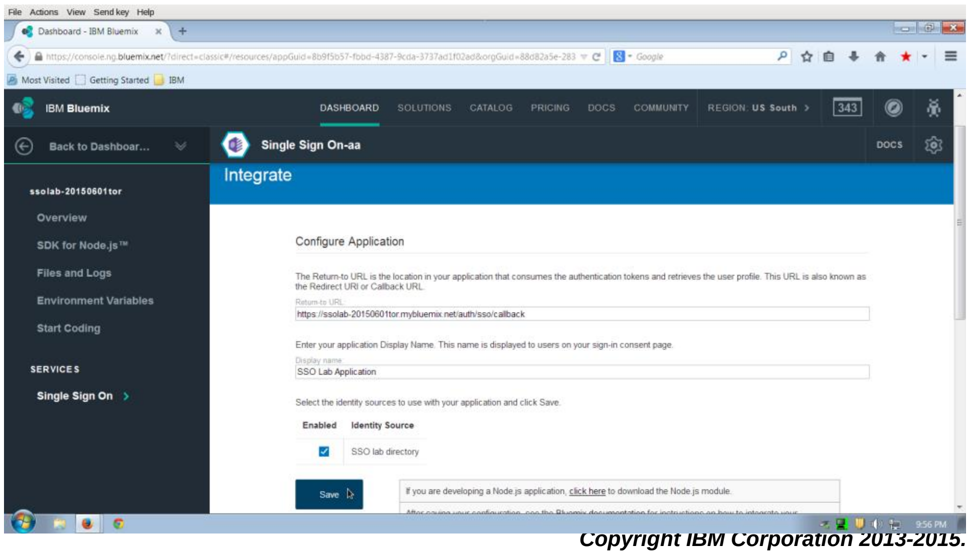
After saving your configuration, see the Bluemix documentation for instructions on how to integrate your application to use the Single Sign On service.

https://bestssoever-2zgs9sg44w-cge7.iam.ibmcloud.com/ssoconfig/downloads/passport-idaas-openidconnect.zip

7. Enter in the Callback URL for the SSO service to invoke after the authentication is completed in the Return-to URL box. The host part of this Callback URL must match the host name that you specified when you created the application. The rest of the host name should be the default Bluemix domain. Finish the URL with the route "`/auth/sso/callback`". Also, update the code for this route in the Node.js starter application. In the examples shown in this exercise, the correct URL is:

   `https://ssolab-20150601tor.mybluemix.net/auth/sso/callback`

8. Optional: Change the Display name for the application from the default. This is the name used by the SSO service when prompting a user for consent to allow the retrieval of personal information from the identity source. After you enter the callback URL and update the Display name (optional), click **Save**.


File Actions View Send key Help
Dashboard - IBM Bluemix
https://console.ng.bluemix.net/?direct=classic#/resources/appGuid=8b9f5b57-fbbd-4387-9cda-3737ad1f02ad&orgGuid=88d82a5e-283
Most Visited Getting Started IBM
IBM Bluemix
DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY REGION: US South
Back to Dashboar...
Single Sign On-aa
DOCS
ssolab-20150601tor
Overview
SDK for Node.js™
Files and Logs
Environment Variables
Start Coding
SERVICES
Single Sign On

Integrate

Configure Application

The Return-to URL is the location in your application that consumes the authentication tokens and retrieves the user profile. This URL is also known as the Redirect URI or Callback URL.

Return-to URL: https://ssolab-20150601tor.mybluemix.net/auth/sso/callback

Enter your application Display Name. This name is displayed to users on your sign-in consent page.

Display name: SSO Lab Application

Select the identity sources to use with your application and click Save.

Enabled	Identity Source
✓	SSO lab directory

Save

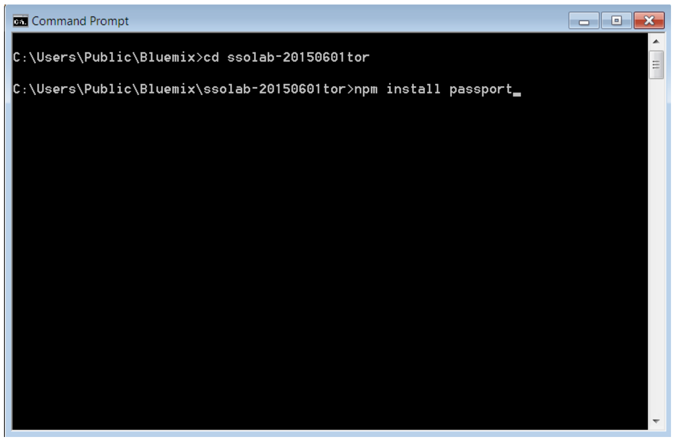
If you are developing a Node.js application, click here to download the Node.js module.

9. Click **Back to Dashboard** to return to the IBM Bluemix Cloud Foundry dashboard. Now, the SSO service is ready to be used with the application.

Next, you'll update the application source code to use the service.

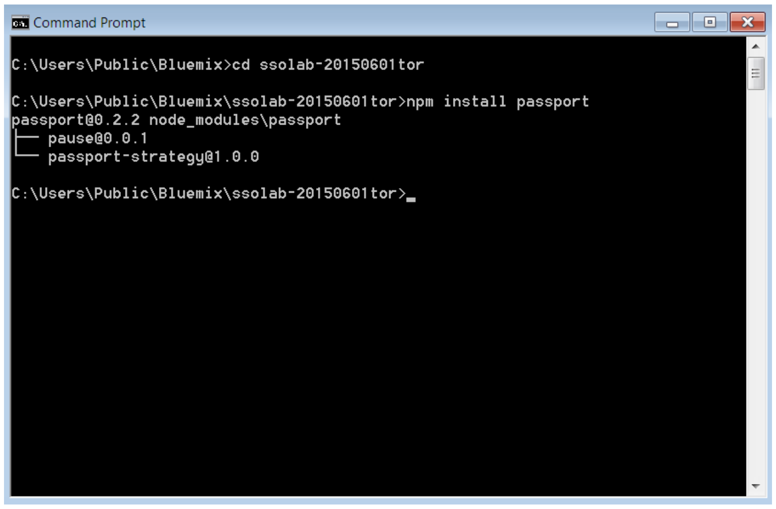
Exercise 4.2.4: Updating the Node.js starter application to use the Single Sign On service for external authentication

1. Click your Node.js application from the dashboard view and then select **Start Coding** to bring back the listing of steps that we saw in the first section. If you haven't done so already, open a terminal on your workstation and perform steps 1, 2 and 3. Within the top source code directory for your application on your workstation, use the node npm tool to install the passport module.



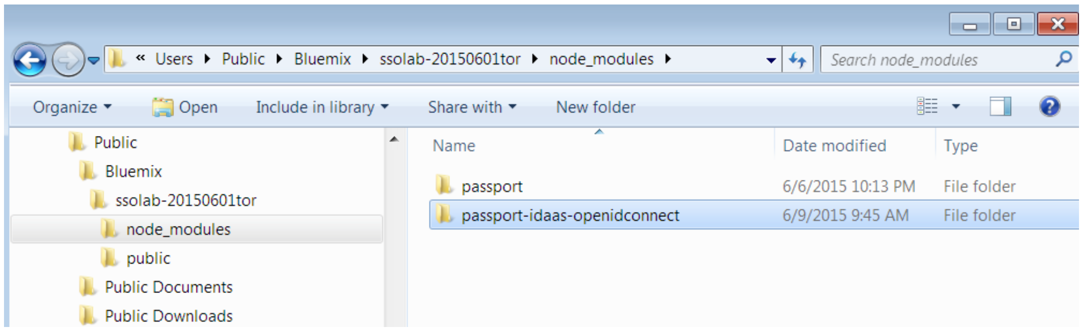
```
C:\Users\Public\Bluemix>cd ssolab-20150601tor

C:\Users\Public\Bluemix\ssolab-20150601tor>npm install passport_
```

This should complete without errors and place the passport module and any other dependencies inside the node_modules directory for the application.



```
C:\Users\Public\Bluemix>cd ssolab-20150601tor

C:\Users\Public\Bluemix\ssolab-20150601tor>npm install passport
passport@0.2.2 node_modules\passport
├── pause@0.0.1
└── passport-strategy@1.0.0

C:\Users\Public\Bluemix\ssolab-20150601tor>_
```

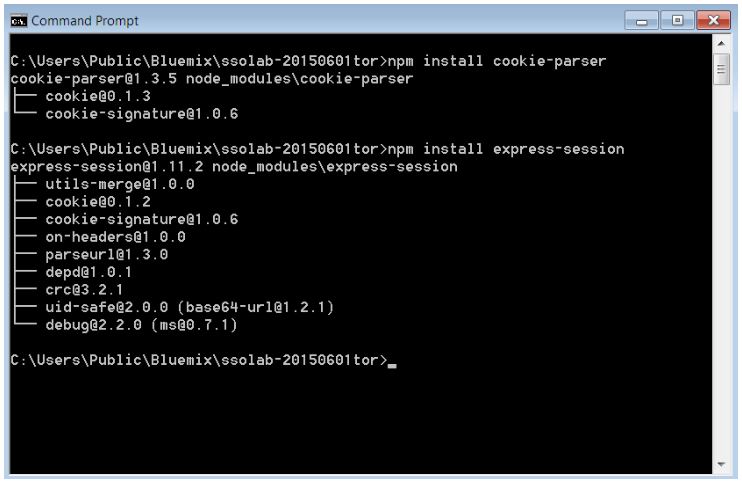
2. From your default download location, find the node module on the **INTEGRATE** tab. The ZIP file name is **passport-idaas-openidconnect.zip** if you need to search for it. Extract the contents of the ZIP file into the node_modules folder of your application. Once complete, the node_modules folder contents should look like this:



Name	Date modified	Type
passport	6/6/2015 10:13 PM	File folder
passport-idaas-openidconnect	6/9/2015 9:45 AM	File folder

3. From the same command window, run two more npm commands:

```
npm install cookie-parser

npm install express-session
```

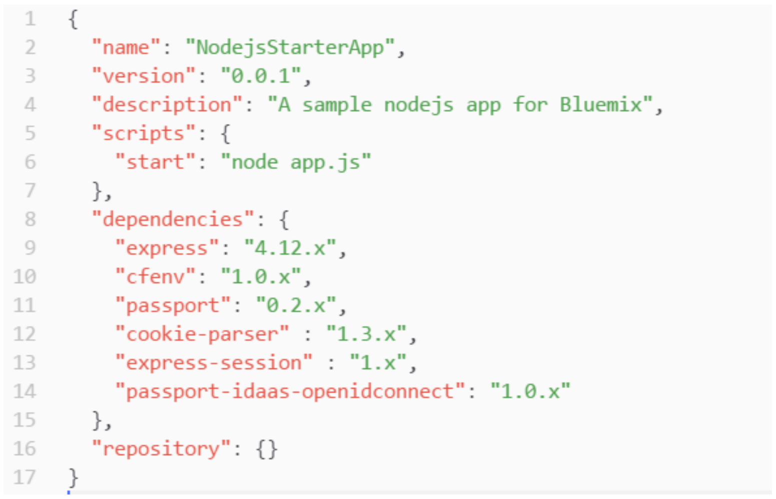


```
C:\Users\Public\Bluemix\ssolab-20150601tor>npm install cookie-parser
cookie-parser@1.3.5 node_modules\cookie-parser
├── cookie@0.1.3
└── cookie-signature@1.0.6

C:\Users\Public\Bluemix\ssolab-20150601tor>npm install express-session
express-session@1.11.2 node_modules\express-session
├── utils-merge@1.0.0
├── cookie@0.1.2
├── cookie-signature@1.0.6
├── on-headers@1.0.0
├── parseurl@1.3.0
├── depd@1.0.1
├── crc@3.2.1
├── uid-safe@2.0.0 (base64-url@1.2.1)
└── debug@2.2.0 (ms@0.7.1)

C:\Users\Public\Bluemix\ssolab-20150601tor>_
```

4. To ensure that module dependencies are retained when you push the application back into Bluemix, use a text editor to open the package.json file from the application code top directory and add the following lines into the dependencies section:

```
"passport": "0.2.x",
"cookie-parser" : "1.3.x",
"express-session" : "1.x"
"passport-idaas-openidconnect": "1.0.x"
```

The JSON file should look something like this when complete:

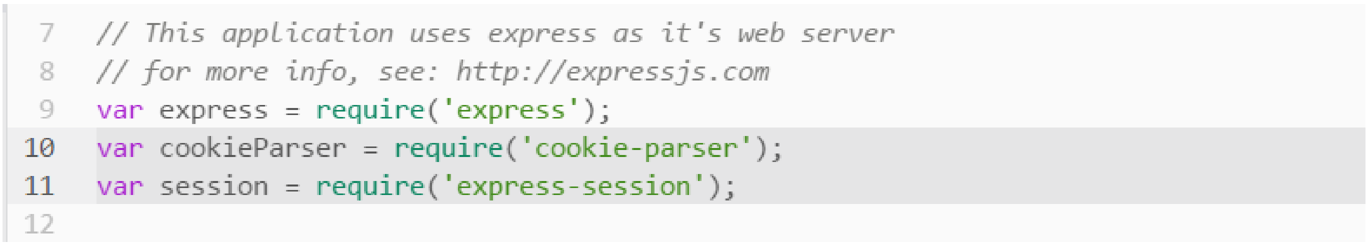


```
{
  "name": "NodejsStarterApp",
  "version": "0.0.1",
  "description": "A sample nodejs app for Bluemix",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "4.12.x",
    "cfenv": "1.0.x",
    "passport": "0.2.x",
    "cookie-parser" : "1.3.x",
    "express-session" : "1.x",
    "passport-idaas-openidconnect": "1.0.x"
  },
  "repository": {}
}
```

5. If there is a .cfignore file in the application top-level directory, remove any line that specifies `node_modules` unless it contains a subdirectory. For example, `node_modules/cookie-parser` is OK, but `node_modules` is not.

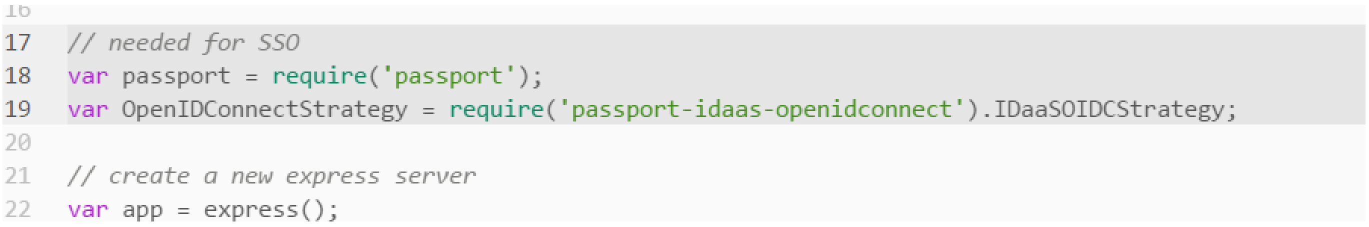
6. Add some code to the application to use the Single Sign On service to manage authentication of access to specified routes within the application. At the end of these exercises, you can see the complete source content if you prefer to copy and paste the updates.

7. Open the app.js file in the application top level directory and add the highlighted code just below the `var express = require('express');` statement.



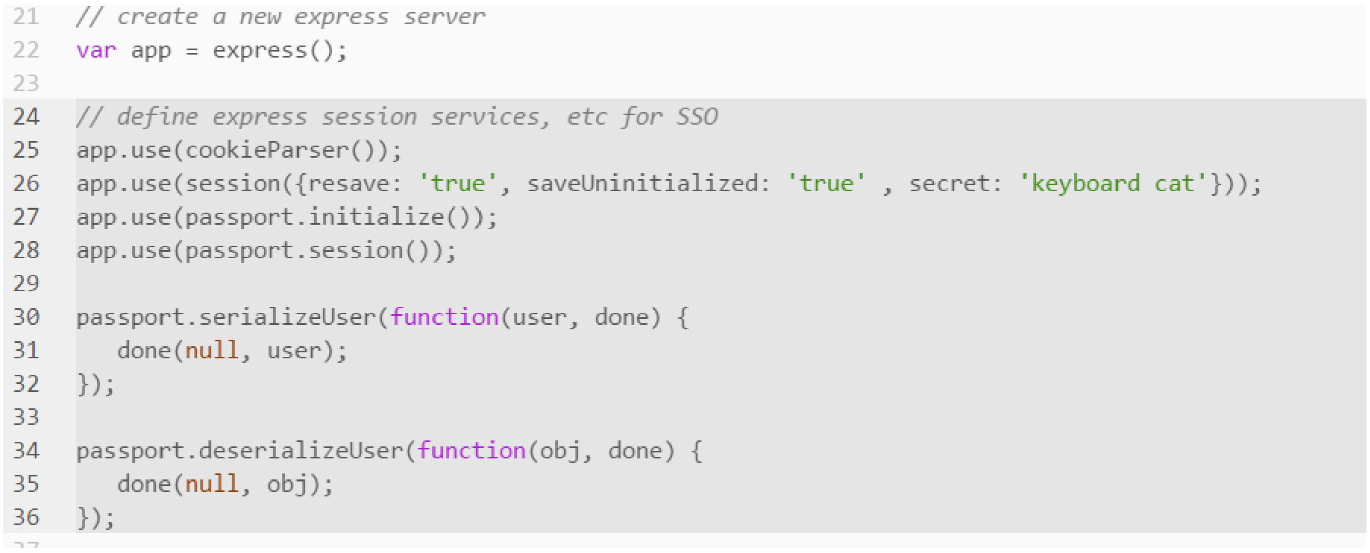
```
// This application uses express as it's web server
// for more info, see: http://expressjs.com
var express = require('express');
var cookieParser = require('cookie-parser');
var session = require('express-session');

```

8. Before the `var app = express();` statement, add this highlighted code:



```

// needed for SSO
var passport = require('passport');
var OpenIDConnectStrategy = require('passport-idaas-openidconnect').IDaaSOIDCStrategy;

// create a new express server
var app = express();
```

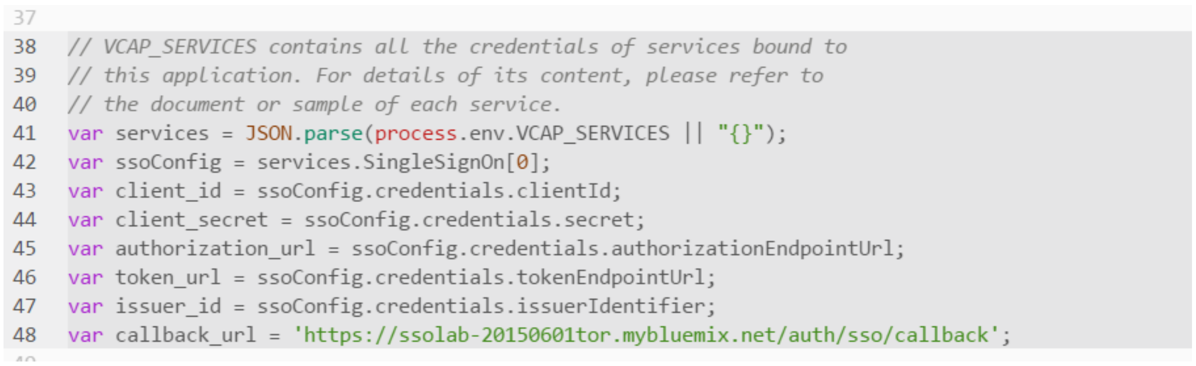
9. After the `var app = express();` statement, add code to create session handling services in express and configure the passport module to use it with this highlighted code.



```
// create a new express server
var app = express();

// define express session services, etc for SSO
app.use(cookieParser());
app.use(session({resave: 'true', saveUninitialized: 'true' , secret: 'keyboard cat'}));
app.use(passport.initialize());
app.use(passport.session());

passport.serializeUser(function(user, done) {
   done(null, user);
});

passport.deserializeUser(function(obj, done) {
   done(null, obj);
});
```

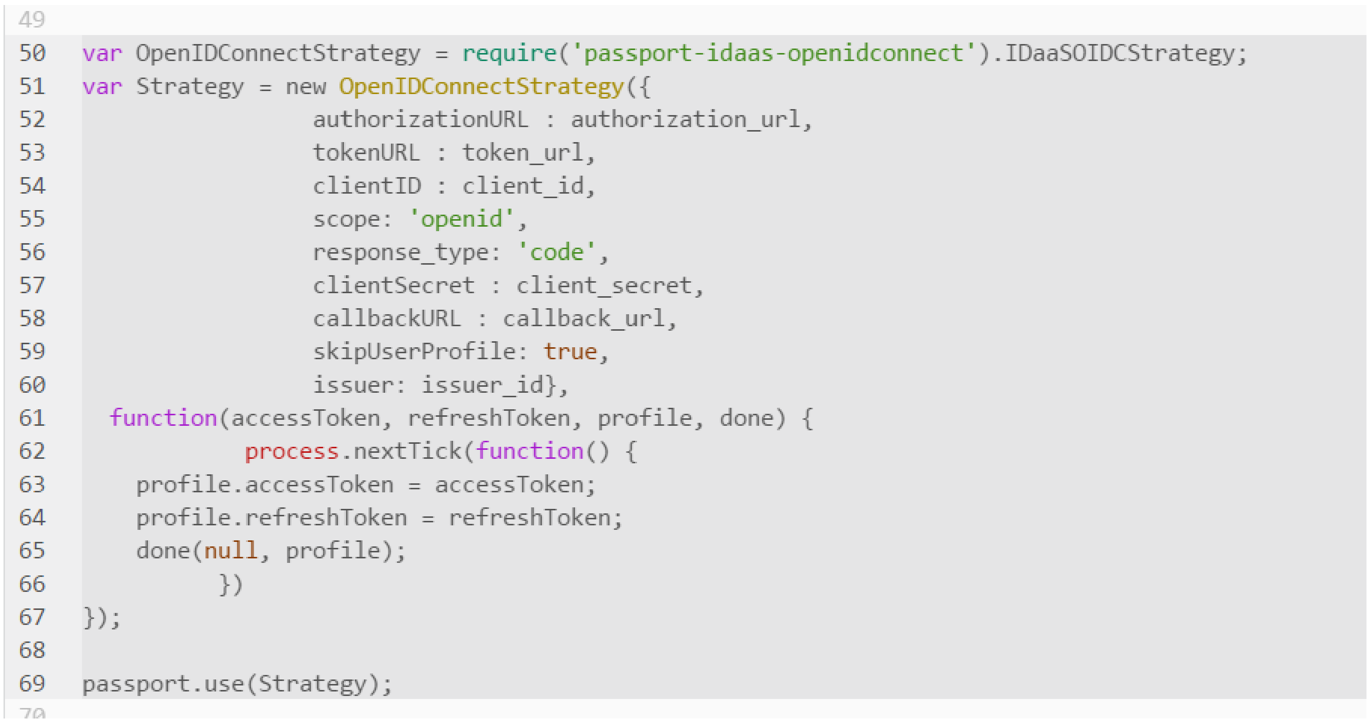
Following this code, you will continue by adding some logic to process the entries in the VCAP_SERVICES environment variable. When the SSO service is bound to the application, the service instance details including the client ID, client secret, authorization URL, token URL, and issuer ID used in the OpenID Connect strategy for passport are set in the VCAP_SERVICES variable.

10. Along with these entries, add the string used for the callback URL in the **INTEGRATE** tab.



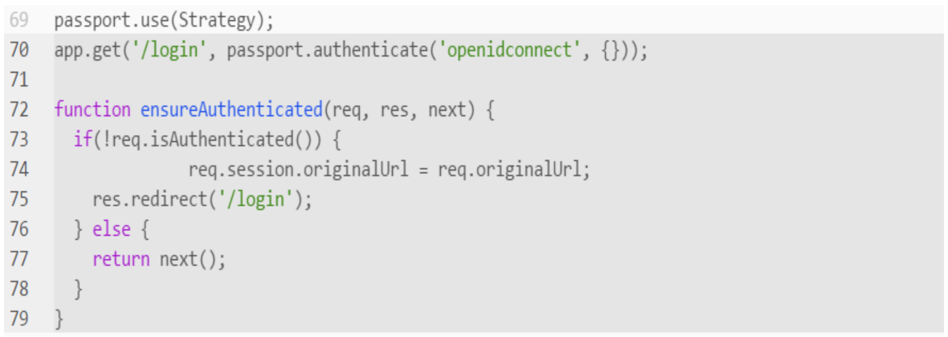
```

// VCAP_SERVICES contains all the credentials of services bound to
// this application. For details of its content, please refer to
// the document or sample of each service.
var services = JSON.parse(process.env.VCAP_SERVICES || "{}");
var ssoConfig = services.SingleSignOn[0];
var client_id = ssoConfig.credentials.clientId;
var client_secret = ssoConfig.credentials.secret;
var authorization_url = ssoConfig.credentials.authorizationEndpointUrl;
var token_url = ssoConfig.credentials.tokenEndpointUrl;
var issuer_id = ssoConfig.credentials.issuerIdentifier;
var callback_url = 'https://ssolab-20150601tor.mybluemix.net/auth/sso/callback';
```

These entries will then be used in a constructor for a new OpenID Connect strategy that will then be provided to passport. The code to be added is highlighted and immediately follows the previous code:



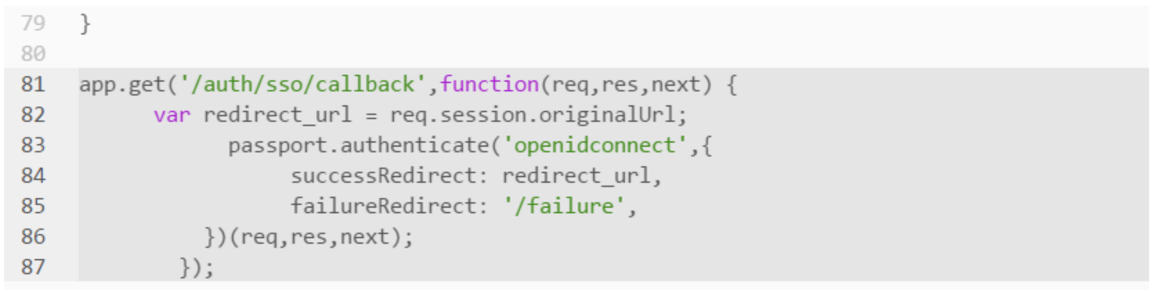
```
var OpenIDConnectStrategy = require('passport-idaas-openidconnect').IDaaSOIDCStrategy;
var Strategy = new OpenIDConnectStrategy({
                 authorizationURL : authorization_url,
                 tokenURL : token_url,
                 clientID : client_id,
                 scope: 'openid',
                 response_type: 'code',
                 clientSecret : client_secret,
                 callbackURL : callback_url,
                 skipUserProfile: true,
                 issuer: issuer_id},
  function(accessToken, refreshToken, profile, done) {
            process.nextTick(function() {
    profile.accessToken = accessToken;
    profile.refreshToken = refreshToken;
    done(null, profile);
          })
});

passport.use(Strategy);
```

11. For this sample application, define a route in express that will be used to initiate authentication for a session. Then, define a function that can be added to any express route to ensure that a session web session accessing the route is authenticated.



```
passport.use(Strategy);
app.get('/login', passport.authenticate('openidconnect', {}));

function ensureAuthenticated(req, res, next) {
  if(!req.isAuthenticated()) {
              req.session.originalUrl = req.originalUrl;
    res.redirect('/login');
  } else {
    return next();
  }
}
```

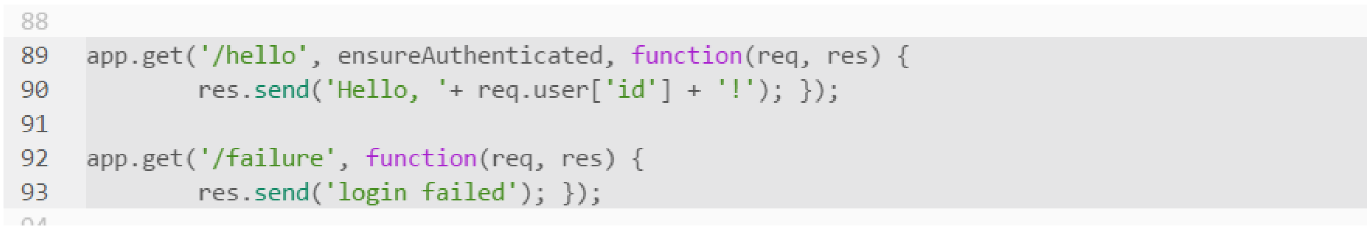
12. Define the authentication callback URL. This authentication callback function implementation will read the original request URL from the session and if the authentication has been successful, continue processing the original URL request. If the authentication has been unsuccessful, the



```
}

app.get('/auth/sso/callback',function(req,res,next) {
      var redirect_url = req.session.originalUrl;
            passport.authenticate('openidconnect',{
                 successRedirect: redirect_url,
                 failureRedirect: '/failure',
          })(req,res,next);
        });
```

user will be routed to a simple failure route.

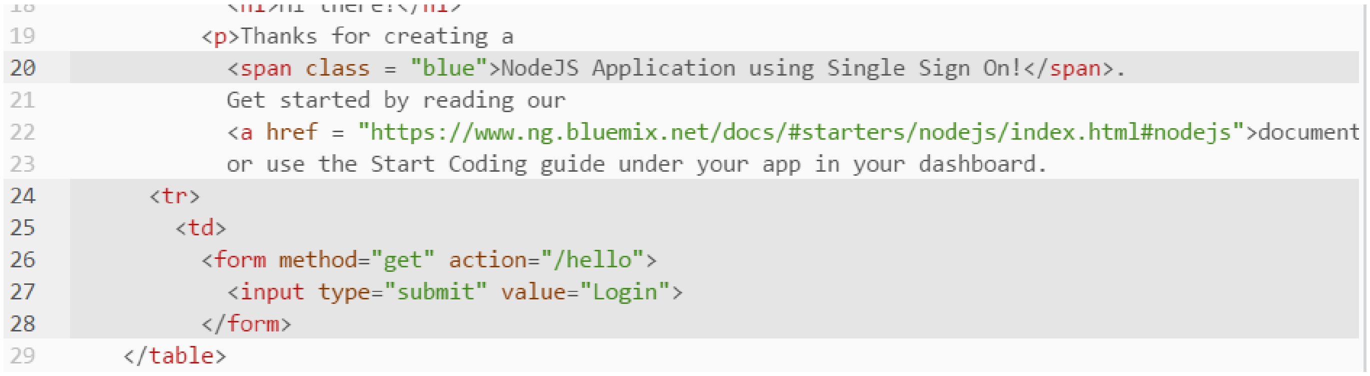
13. Add two routes to the application for user interaction. The first invokes the ensureAuthenticated() function to check to see whether the session is authenticated and if not, this function will authenticate the user and return. When authenticated, the first route will return a simple message based on the unique ID that is provided to the Single Sign On service by the identity provider. The second route provides the message when an authentication fails.



```

app.get('/hello', ensureAuthenticated, function(req, res) {
        res.send('Hello, '+ req.user['id'] + '!'); });

app.get('/failure', function(req, res) {
        res.send('login failed'); });
```

This completes the changes to the app.js code.

14. Save the changes in the editor and open the public/index.html file where you will make minor changes to the starter static html. Change the content in the `<span class = “blue”>` element to something more inspiring, and then add a simple form button as an additional table row.

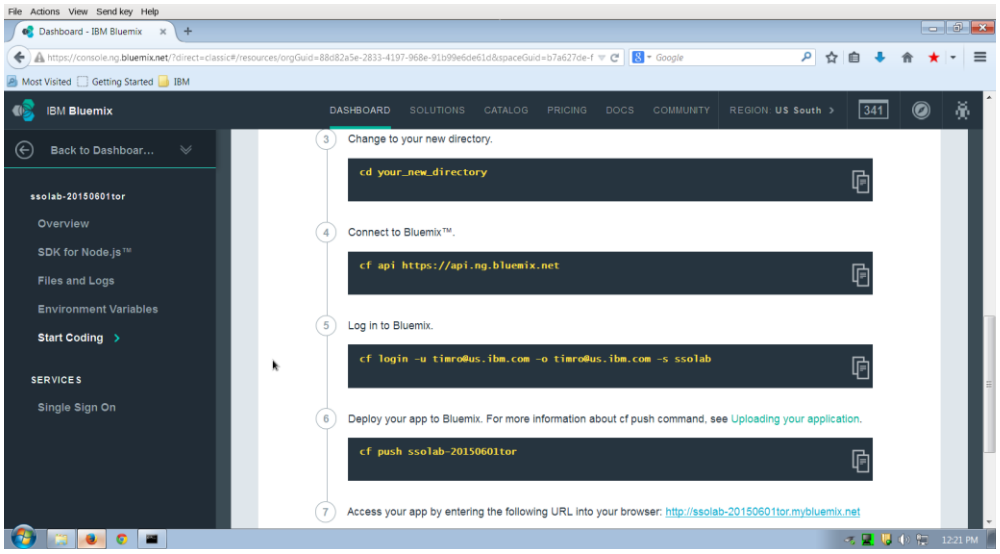


```
        <h1>Hi there!</h1>
        <p>Thanks for creating a
          <span class = "blue">NodeJS Application using Single Sign On!</span>.
          Get started by reading our
          <a href = "https://www.ng.bluemix.net/docs/#starters/nodejs/index.html#nodejs">document
          or use the Start Coding guide under your app in your dashboard.
      <tr>
        <td>
          <form method="get" action="/hello">
            <input type="submit" value="Login">
          </form>
    </table>
```

15. Save this file and then return to the command prompt window for the next step.

Exercise 4.2.5: Publishing your updated application to Bluemix

1. Return to the Bluemix Cloud Foundry dashboard and click on your application icon. Then select the Start Coding link. Scroll down in the window to see steps 4, 5, and 6.


File Actions View Send key Help
Dashboard - IBM Bluemix
https://console.ng.bluemix.net/?direct=classic#/resources/orgGuid=88d82a5e-2833-4197-968e-91b99e6de61d&spaceGuid=b7a627de-f
Most Visited Getting Started IBM
IBM Bluemix
DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY REGION: US South
Back to Dashboar...
ssolab-20150601tor
Overview
SDK for Node.js™
Files and Logs
Environment Variables
Start Coding
SERVICES
Single Sign On

3 Change to your new directory.

```
cd your_new_directory
```

4 Connect to Bluemix™.

```
cf api https://api.ng.bluemix.net
```

5 Log in to Bluemix.

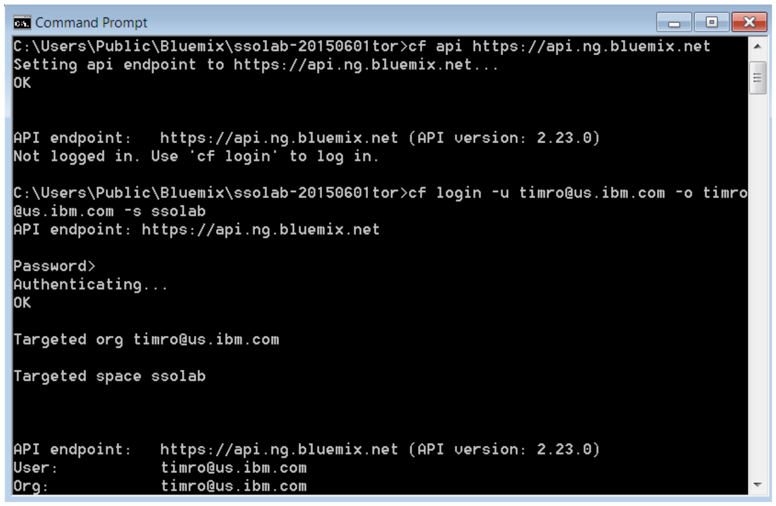
```
cf login -u timro@us.ibm.com -o timro@us.ibm.com -s ssolab
```

6 Deploy your app to Bluemix. For more information about cf push command, see Uploading your application.

```
cf push ssolab-20150601tor
```

7 Access your app by entering the following URL into your browser: http://ssolab-20150601tor.mybluemix.net

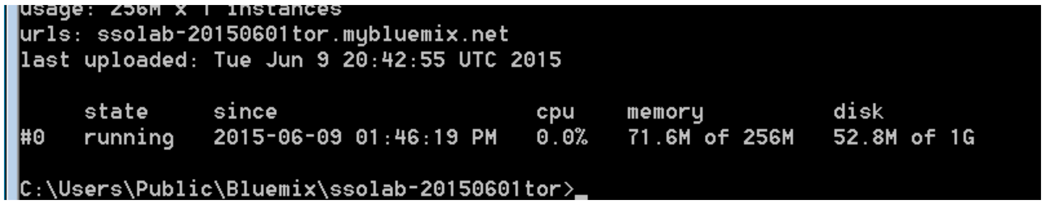
2. Return to or open a command prompt on your workstation. Change to the directory with your updated application and perform steps 4 and 5 as shown. Your Bluemix user, space, and application name will be different than what is shown in this screen capture.) You will see output like this:



Command Prompt

```
C:\Users\Public\Bluemix\ssolab-20150601tor>cf api https://api.ng.bluemix.net
Setting api endpoint to https://api.ng.bluemix.net...
OK


API endpoint:   https://api.ng.bluemix.net (API version: 2.23.0)
Not logged in. Use 'cf login' to log in.

C:\Users\Public\Bluemix\ssolab-20150601tor>cf login -u timro@us.ibm.com -o timro
@us.ibm.com -s ssolab
API endpoint: https://api.ng.bluemix.net

Password>
Authenticating...
OK

Targeted org timro@us.ibm.com

Targeted space ssolab



API endpoint:   https://api.ng.bluemix.net (API version: 2.23.0)
User:           timro@us.ibm.com
Org:            timro@us.ibm.com
```

3. Push your application by using the command `cf push appname`. There will be a lot of output as the application is uploaded back to Bluemix and then started. If all goes well, you will see something like this at the end:

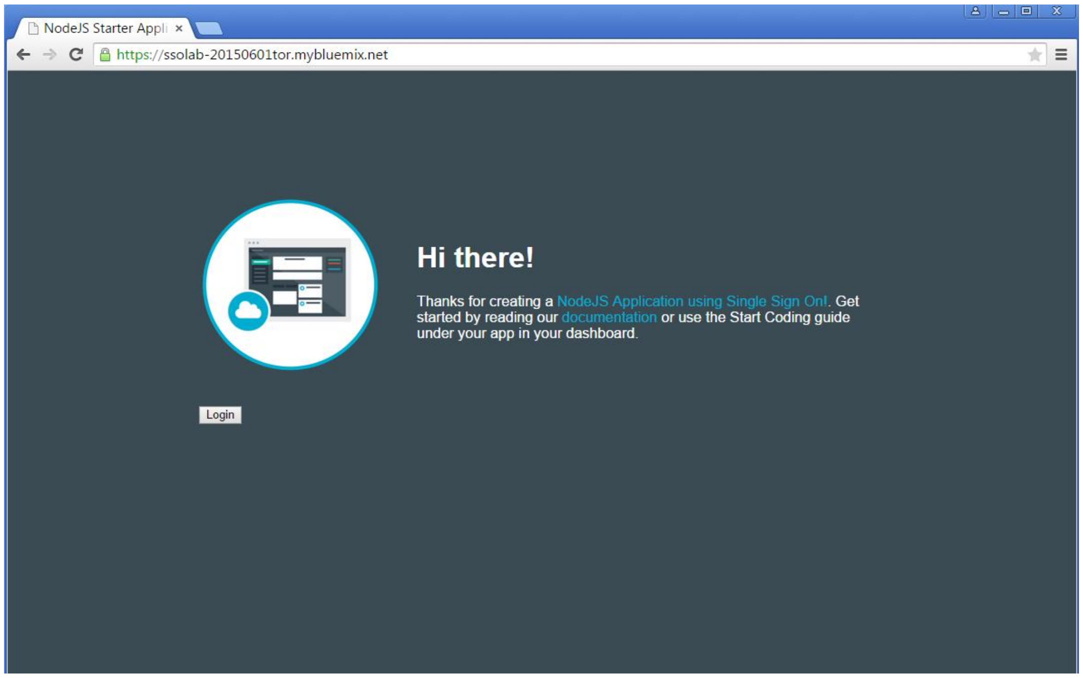


```
usage: 256M x 1 instances
urls: ssolab-20150601tor.mybluemix.net
last uploaded: Tue Jun 9 20:42:55 UTC 2015

     state     since                    cpu    memory          disk
#0   running   2015-06-09 01:46:19 PM   0.0%   71.6M of 256M   52.8M of 1G

C:\Users\Public\Bluemix\ssolab-20150601tor>_
```

Exercise 4.2.6: Testing the application

When testing with the Single Sign On service with the cloud directory, it can be easier to use a different browser security and cookie context (for example in a new “private” window in Firefox) to simulate the experience of a new user to your application. Or you can also just use an entirely different browser. Here we'll use a Chrome browser in the screen captures.

4. Open your application's URL: `https://<appname>.mybluemix.net`. (Be sure to use HTTPS.)



Hi there!

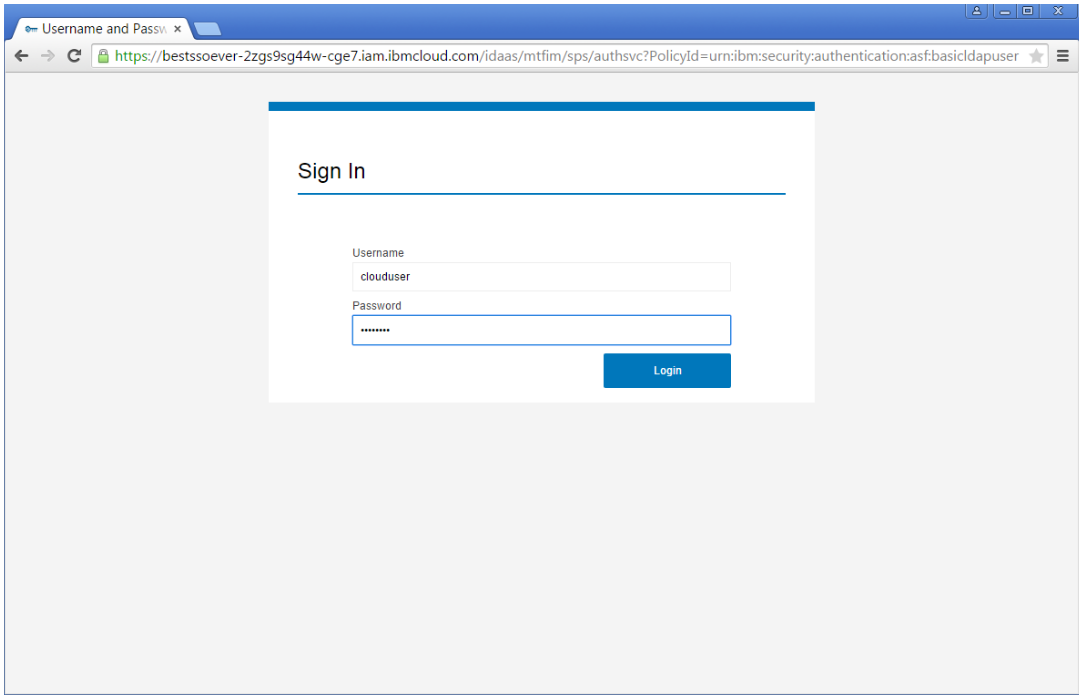
Thanks for creating a NodeJS Application using Single Sign On!. Get started by reading our documentation or use the Start Coding guide under your app in your dashboard.

Note that your customized message is appearing in the blue text and that the **Login** button you added is present.

5. Click the **Login** button to see the Single Sign On identity provider chooser list.

Because the Single Sign On service is configured with only one identity source, only one option is shown.

6. Click **Cloud Directory** and sign in as the user that you added to the directory previously.


Username and Passw

https://bestssoever-2zgs9sg44w-cge7.iam.ibmcloud.com/idaas/mtfim/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:basicldapuser

Sign In

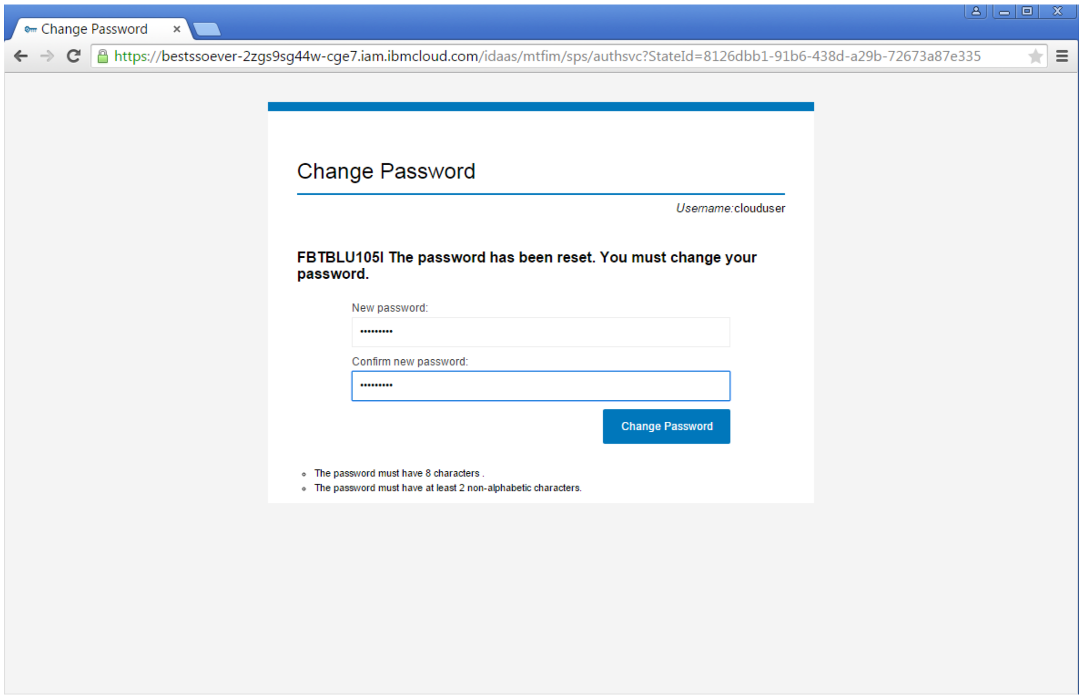
Username: clouduser

Password: ••••••••

Login

After you click the Login button, you will be prompted to change the user password. (This happens by default when using the Cloud Directory.) Note that requirements for the password are displayed in the form.

7. Enter in a new password.


Change Password

https://bestssoever-2zgs9sg44w-cge7.iam.ibmcloud.com/idaas/mtfim/sps/authsvc?StateId=8126dbb1-91b6-438d-a29b-72673a87e335

Change Password

*Username:*clouduser

FBTBLU105I The password has been reset. You must change your password.

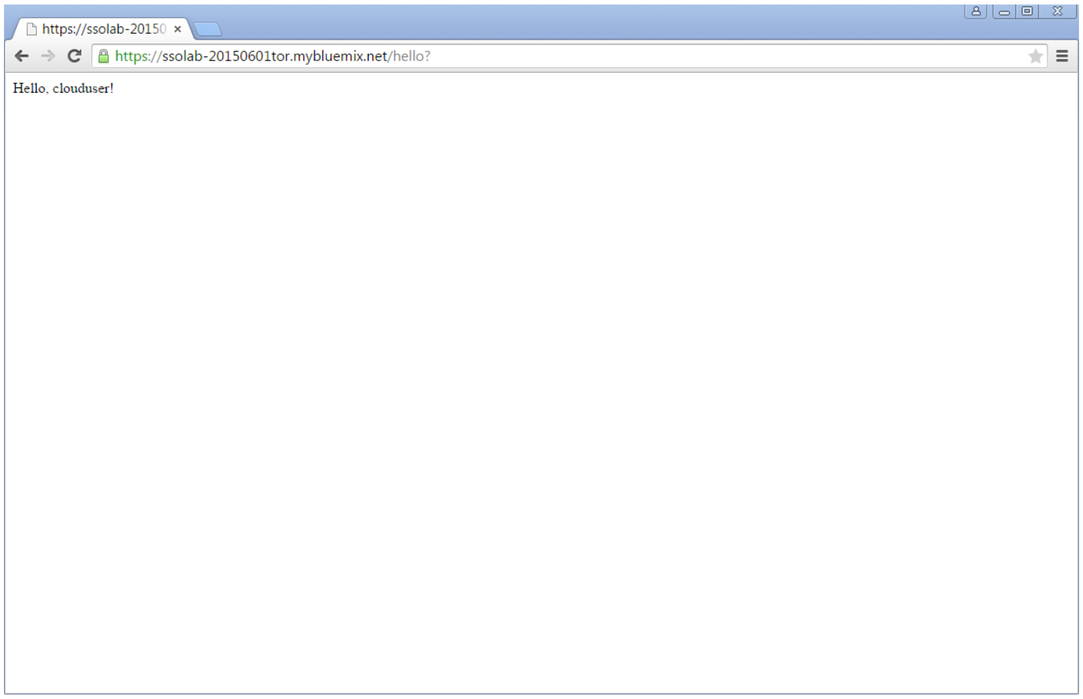
New password:

Confirm new password:

- The password must have 8 characters .
- The password must have at least 2 non-alphabetic characters.

8. Select **Allow** or **All and Remember** to retrieve personal information from the external authentication provider. If you select **Allow**, each time that you authenticate with this identity provider through the Single Sign On service, the prompt will be displayed.

   Then the application displays the successful authentication message:


https://ssolab-20150601tor.mybluemix.net/hello?

Hello, clouduser!

Summary and next steps

Now, you have an application that can use the Single Sign On service with a simple identity provider. The next step should be to read the Bluemix SSO documentation to add a Social Media identity provider. There are step-by-step instructions for this and for configuring SAML Enterprise identity sources. To access the documentation, from the configuration panel for the Single Sign On. service, click **DOCS**.

Code sample for app.js

This app.js file includes the additions from all of the exercises. If you want to copy and paste code to move through the exercises faster and avoid possible typos, use this sample code.

Be sure to update the `callback_url` variable to match your Bluemix application host name.

```
/*jshint node:true*/

//------------------------------------------------------------------------------

// node.js starter application for Bluemix

//------------------------------------------------------------------------------

// This application uses express as it's web server

// for more info, see: http://expressjs.com

var express = require('express');

var cookieParser = require('cookie-parser');

var session = require('express-session');

// cfenv provides access to your Cloud Foundry environment

// for more info, see: https://www.npmjs.com/package/cfenv

var cfenv = require('cfenv');

// needed for SSO

var passport = require('passport');

var OpenIDConnectStrategy = require('passport-idaas-
openidconnect').IDaaSOIDCStrategy;

// create a new express server

var app = express();


// define express session services, etc for SSO

app.use(cookieParser());
app.use(session({resave: 'true', saveUninitialized: 'true' , secret: 'keyboard
cat'}));

app.use(passport.initialize());

app.use(passport.session());

passport.serializeUser(function(user, done) {

   done(null, user);

});


passport.deserializeUser(function(obj, done) {

   done(null, obj);

});


// VCAP_SERVICES contains all the credentials of services bound to

// this application. For details of its content, please refer to

// the document or sample of each service.

var services = JSON.parse(process.env.VCAP_SERVICES || "{}");

var ssoConfig = services.SingleSignOn[0];

var client_id = ssoConfig.credentials.clientId;

var client_secret = ssoConfig.credentials.secret;

var authorization_url = ssoConfig.credentials.authorizationEndpointUrl;

var token_url = ssoConfig.credentials.tokenEndpointUrl;

var issuer_id = ssoConfig.credentials.issuerIdentifier;

var callback_url = 'https://ssolab-20150601tor.mybluemix.net/auth/sso/callback';

var OpenIDConnectStrategy = require('passport-idaas-
openidconnect').IDaaSOIDCStrategy;

var Strategy = new OpenIDConnectStrategy({

                 authorizationURL : authorization_url,

                 tokenURL : token_url,

                 clientID : client_id,

                 scope: 'openid',
                 response_type: 'code',
                 clientSecret : client_secret,
                 callbackURL : callback_url,
                 skipUserProfile: true,
                 issuer: issuer_id},
      function(accessToken, refreshToken, profile, done) {
                  process.nextTick(function() {
            profile.accessToken = accessToken;
            profile.refreshToken = refreshToken;
            done(null, profile);
            })
});

passport.use(Strategy);
app.get('/login', passport.authenticate('openidconnect', {}));


function ensureAuthenticated(req, res, next) {
      if(!req.isAuthenticated()) {
                  req.session.originalUrl = req.originalUrl;
            res.redirect('/login');
      } else {
            return next();
      }
}
app.get('/auth/sso/callback',function(req,res,next) {
          var redirect_url = req.session.originalUrl;
            passport.authenticate('openidconnect',{
                 successRedirect: redirect_url,
                 failureRedirect: '/failure',
          })(req,res,next);
        });

app.get('/hello', ensureAuthenticated, function(req, res) {
        res.send('Hello, '+ req.user['id'] + '!'); });

app.get('/failure', function(req, res) {
        res.send('login failed'); });

// serve the files out of ./public as our main files
app.use(express.static(__dirname + '/public'));

// get the app environment from Cloud Foundry
var appEnv = cfenv.getAppEnv();

// start server on the specified port and binding host
app.listen(appEnv.port, appEnv.bind, function() {

      // print a message when the server starts listening
  console.log("server starting on " + appEnv.url);
});
```