



IBM Single Sign On service for Bluemix – Build your first Java application using external authentication.

12/16/2015

IBM EcoDev

Contents

CONTENTS.....	2
INTRODUCTION	3
CREATE JAVA STARTER APPLICATION AND ADD SINGLE SIGN ON SERVICE TO DASHBOARD:.....	4
CONFIGURE A CLOUD DIRECTORY IDENTITY SOURCE	8
BIND THE SINGLE SIGN ON SERVICE TO THE APPLICATION	12
RUN THE APP.....	17
ADD SECURITY CONSTRAINTS TO JAVA APP	18
DOWNLOAD STARTER CODE	18
IMPORT INTO ECLIPSE	19
ADD SERVLET TO PROTECT.....	21
ADD HTTP CONSTRAINTS	23
MAP ROLES	23
PREPARE AND PUSH YOUR PROJECT.....	24
TESTING THE APPLICATION	25
SUMMARY AND NEXT STEPS.....	28

Introduction

When developing an application supporting multiple users, one of the first decisions you will need to face is how to authenticate each 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 lab will introduce the IBM Single Sign On (SSO) service for Bluemix which you can use to configure your web application to authenticate users from social media sites, enterprise directories via SAML federation, and also a ldap-based cloud identity source.

This lab will walk through the steps to add and configure the IBM Single Sign On service for Bluemix to the java starter application. It will provide step-by-step 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.

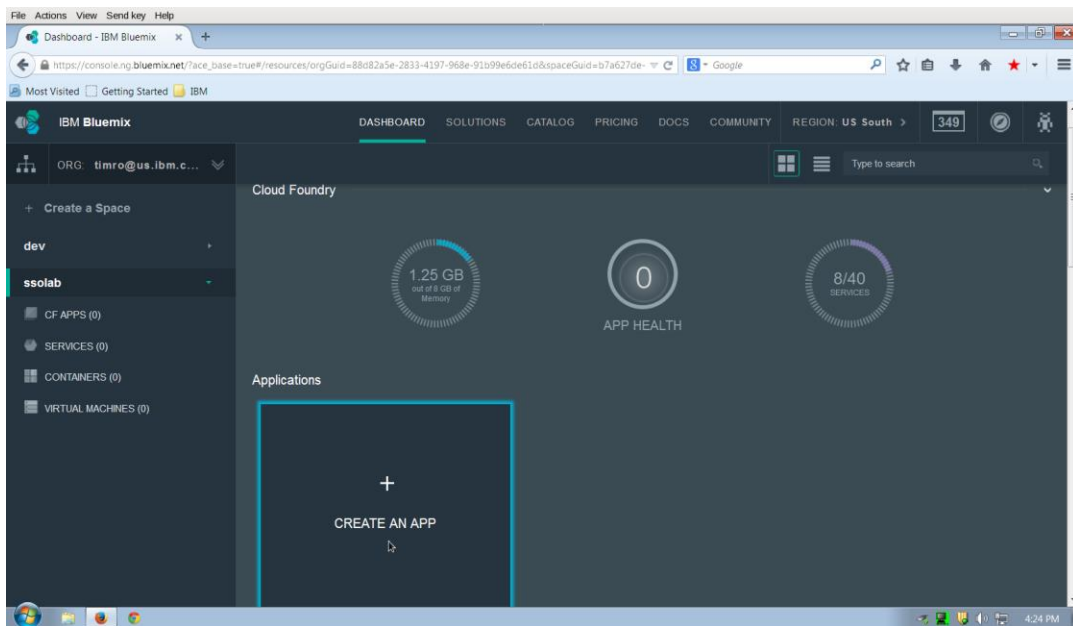
The lab requires a workstation (Windows, Mac OS X or Linux are all fine) with the following items installed:

- bluemix.net account (sign up for a 30-day free trial)
- Java JRE 7.1 or higher (www.java.com/download)
- Multiple web browsers (it's most convenient to use one browser for working with the Bluemix and Single Sign On control panel in one and a separate browser for testing your application, recent versions of Firefox, Chrome, Opera, Safari, Internet Explorer)
- Eclipse mars or later (www.eclipse.org/downloads)
- Cloud Foundry CLI tool version 6.9 or later – there is a step in the application setup where a link is provided for installation if it is not already on your workstation.

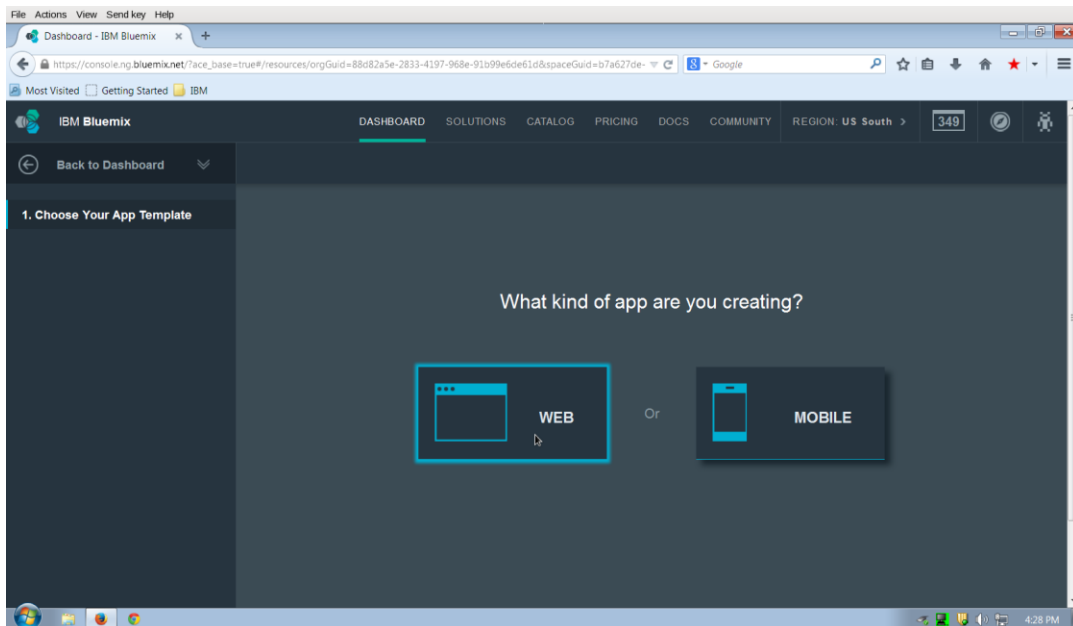
Now let's get started.

Create Java starter application and add Single Sign On service to dashboard:

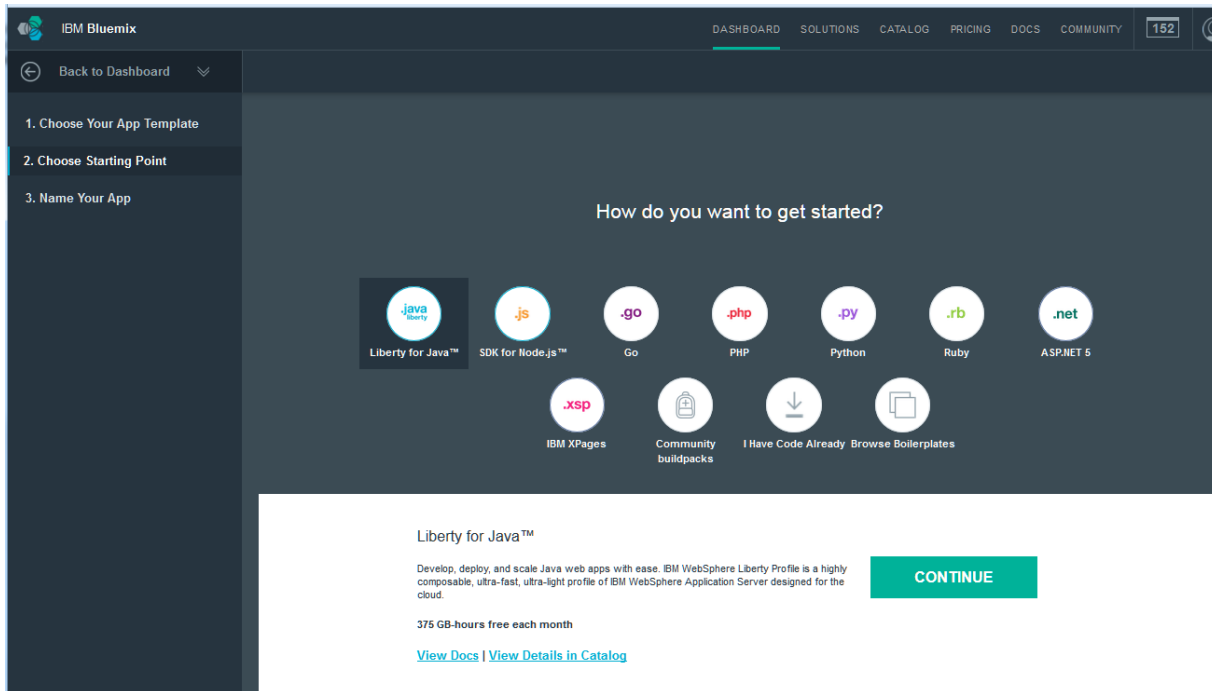
Sign in to Bluemix and go to the Dashboard. Under Cloud Foundry, click on the “CREATE AN APP” button to get started:



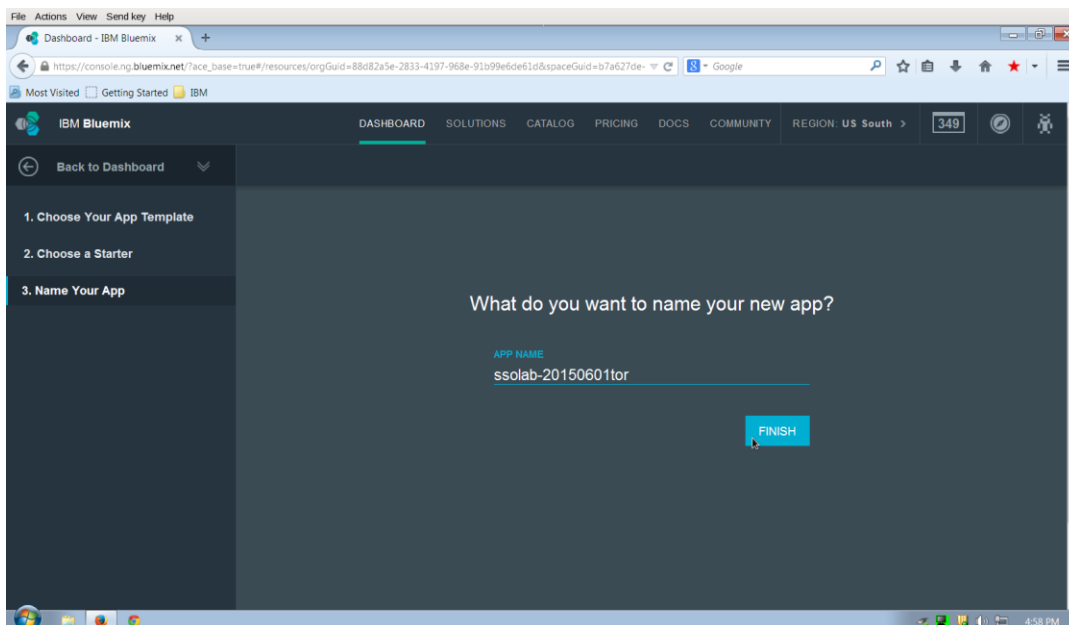
Then select Web app:



Next, you will need to select the runtime for the application. For this lab select the Liberty for Java SDK, and confirm by clicking on CONTINUE:

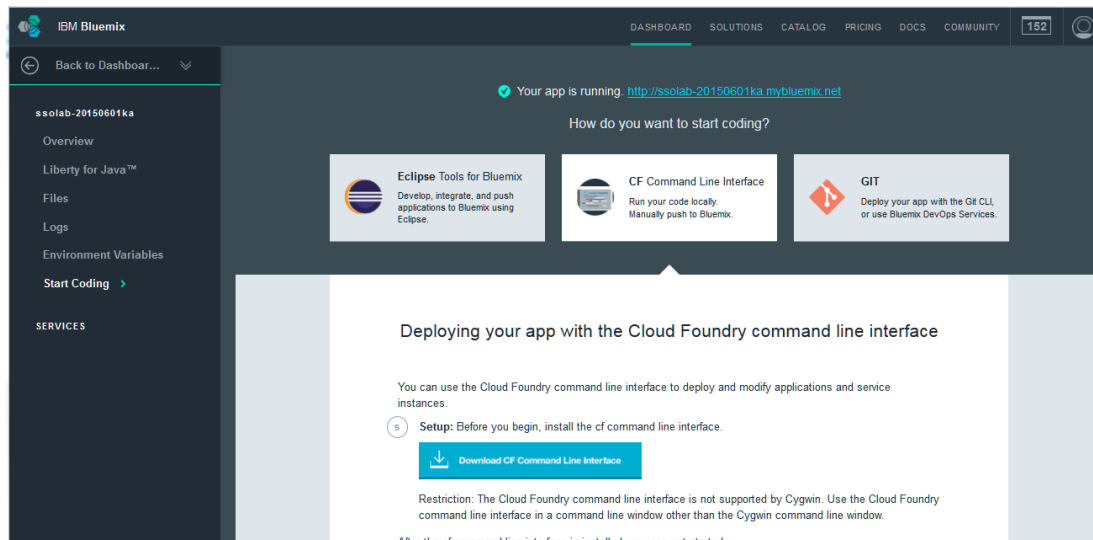


To complete adding the new application, provide a name. Each application in Bluemix public requires a unique name. One easy way to do this for a test app is to append the date and your initials to the name, and then select FINISH:



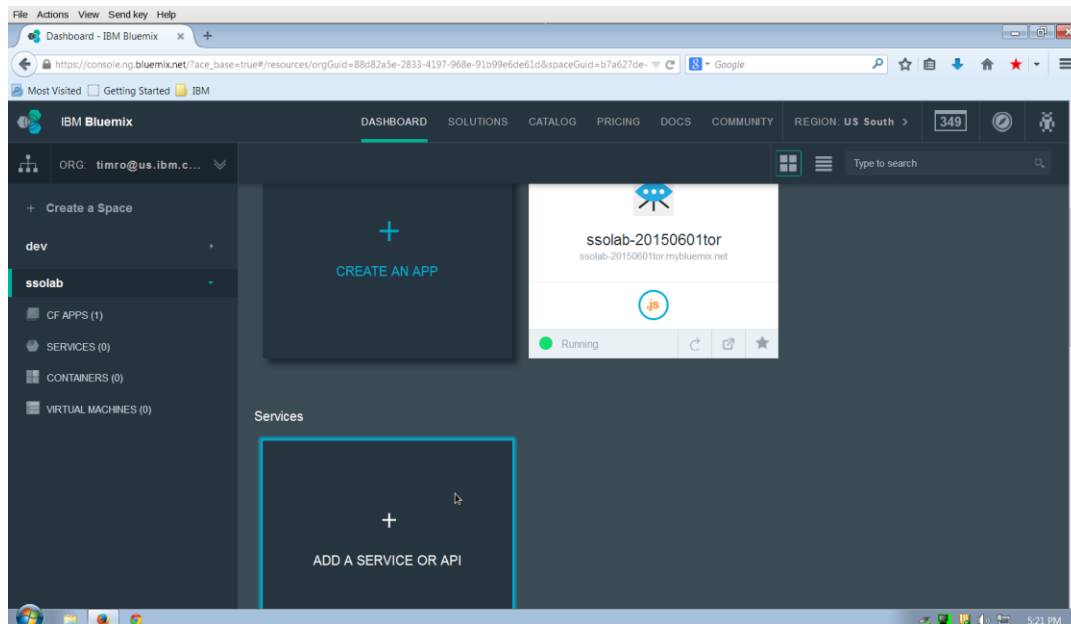
Bluemix will bring up the application's Start Coding panel with a message at the top about the application starting the staging process. If you do not already have the Cloud Foundry (cf) CLI tool installed on your workstation, click

on the link shown to go to GitHub and download the current version for your workstation (it will be later than 6.9 but that is ok). After the download is completed, install the cf tool and then proceed to the next step.

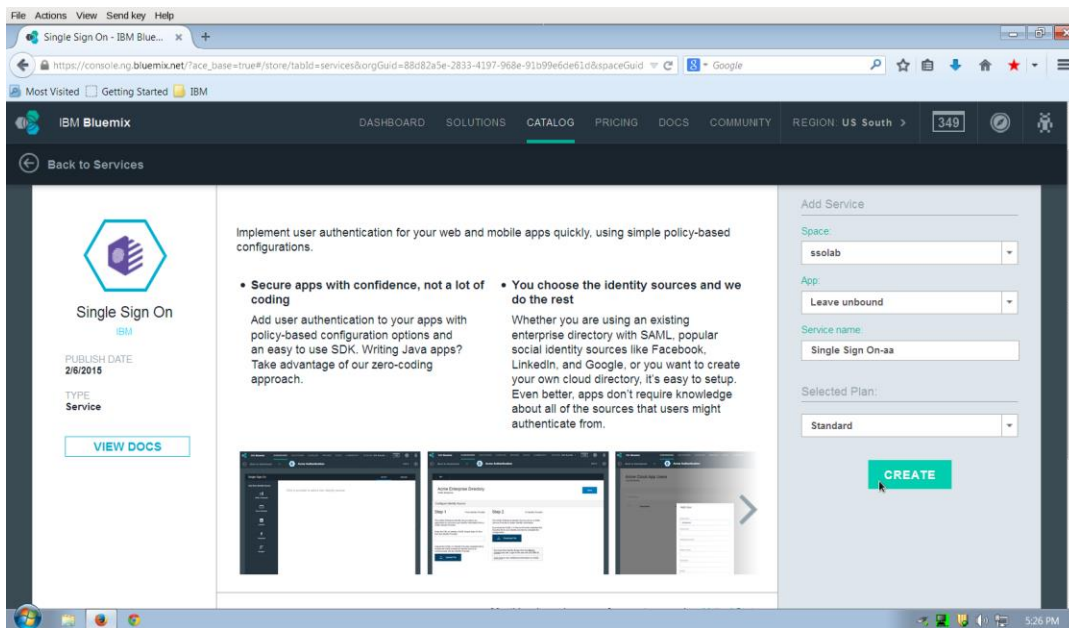


Next, add the Single Sign On service to the Dashboard. Before proceeding, you will need to select an id that will become the prefix of the URL for the service when used when applications redirect users to the service for selecting an identity provider, and also for handling callbacks from identity providers. This id can be up to 32 characters, and must start with an alphanumeric character. Make a note of it here if you like: _____

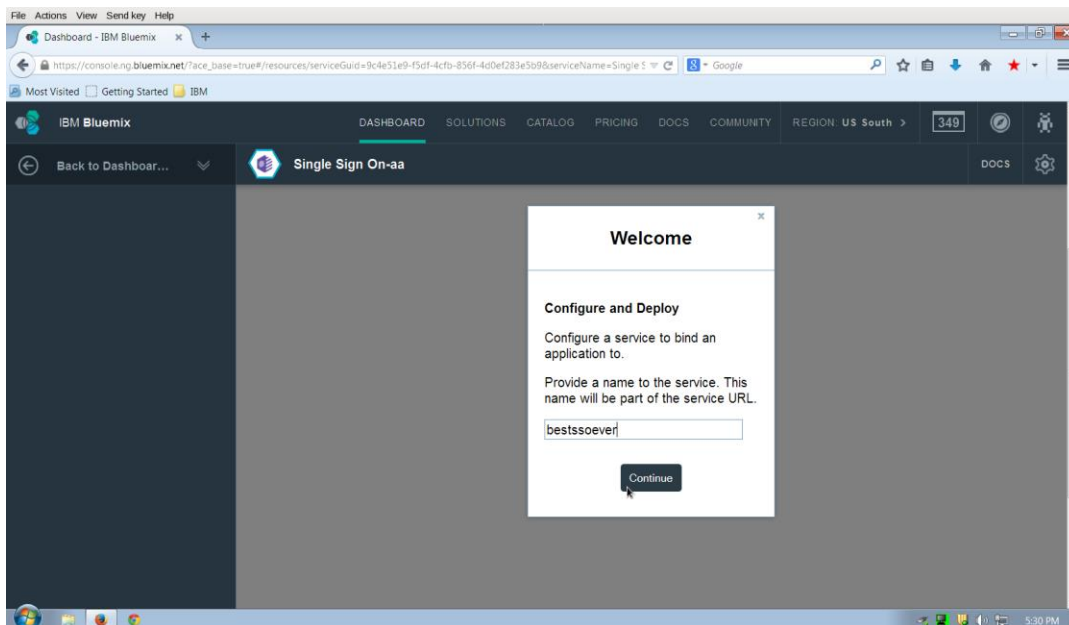
Then, go back to the Bluemix Cloud Foundry Dashboard and select Add a Service or API:



Scroll down in the services list to the Security section and then click on the Single Sign On icon, this will bring up a configuration panel for adding the service. On the right hand side, select Leave unbound under the App: section. You can change the Service name: that will be shown in the dashboard or leave it at the default. Then click on the CREATE button:



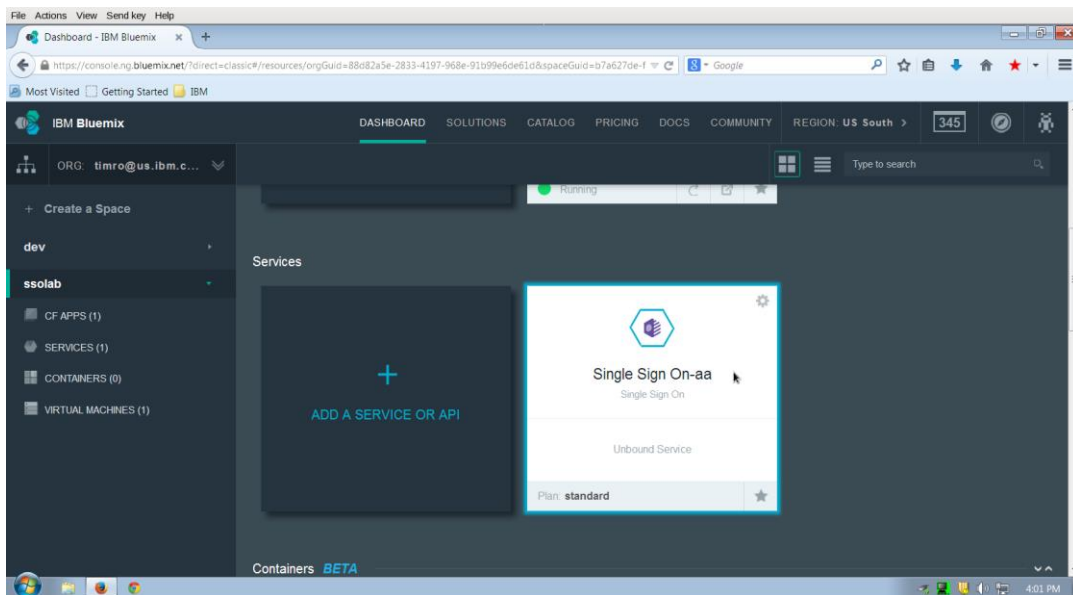
A panel will display to prompt for a "name" for the service. This is the id that you chose earlier, it is not the name that was shown in the previous panel as the Service name. Once provided, click on the Continue button:



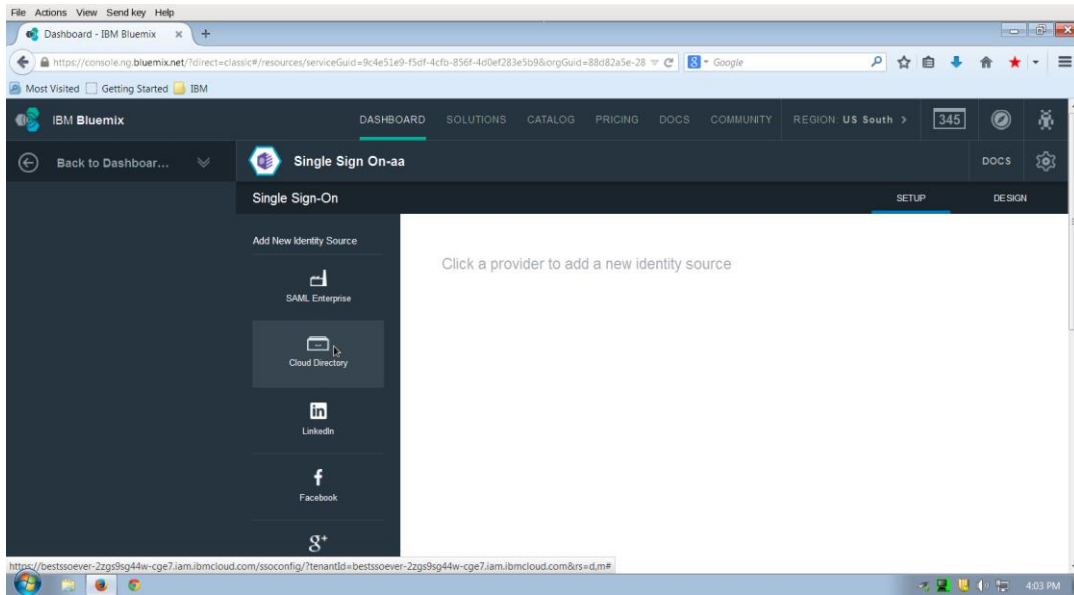
This will add the service to the dashboard and now you will be able to create a simple Cloud directory and add a testing user for external authentication in the next section of the lab.

Configure a Cloud directory identity source

Back in the IBM Bluemix Cloud Foundry dashboard, scroll down to find the SSO service that you just created. Click on the service to open up the configuration panel:



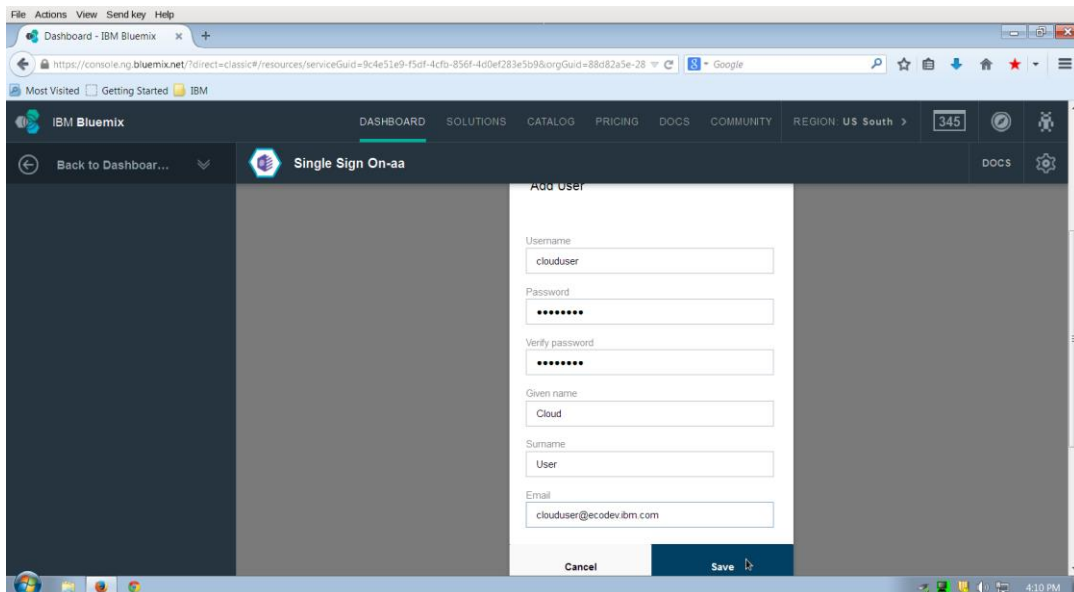
In the configuration panel, click on the Cloud Directory icon to add and configure a directory in the cloud for our lab exercises:



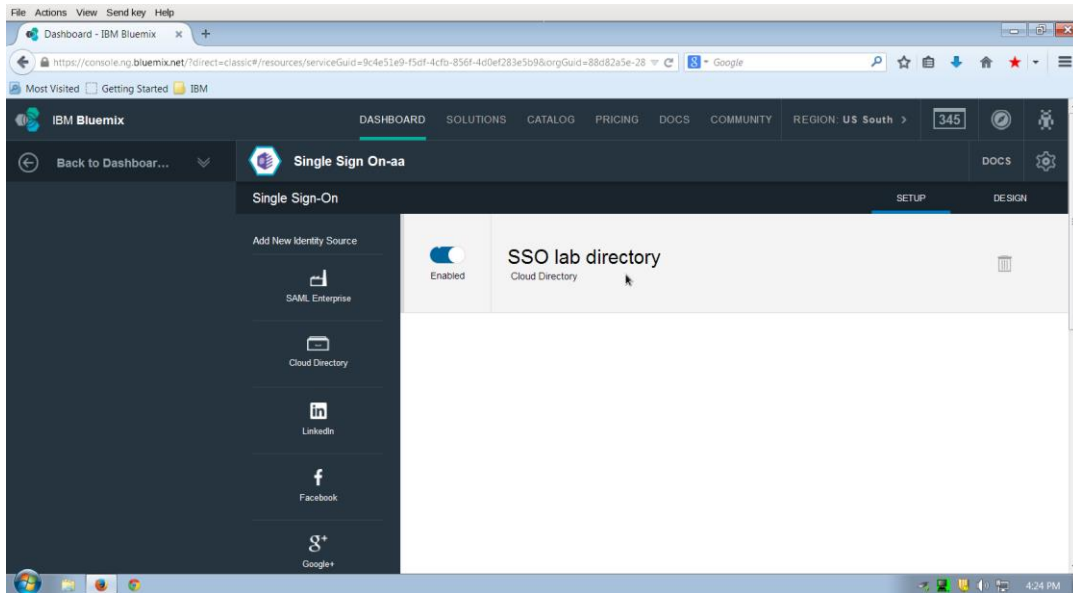
In the next screen, you may give the identity source a new name other than the default and see the current list of users. It's empty at first so click on the add icon:



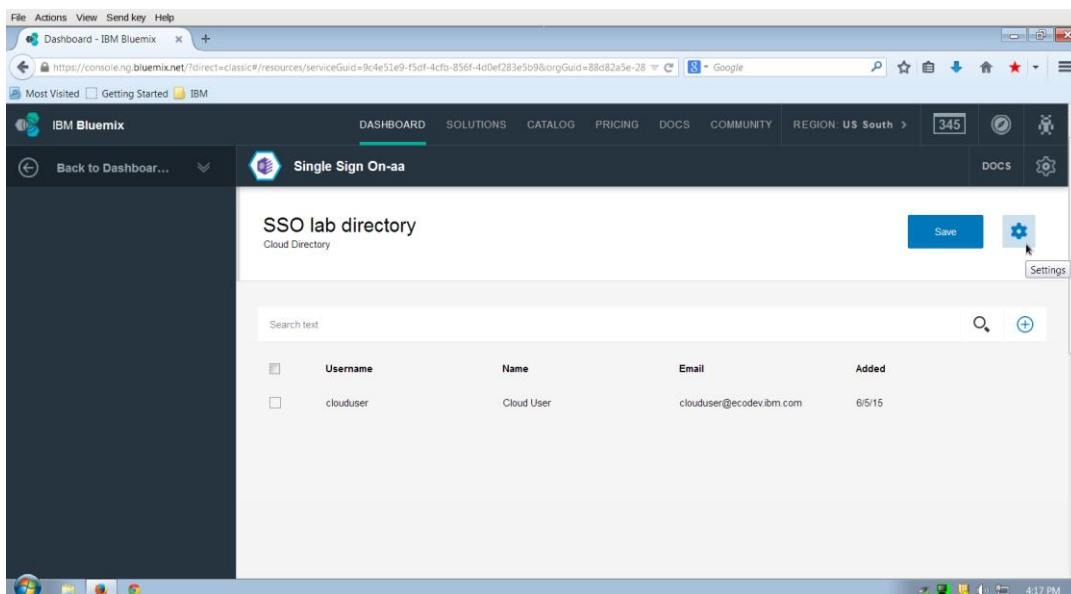
and put in some information for a test user. This can be any user information that you prefer, however make a note of the username and password set since that will be used when testing the external authentication service:



Save the user by clicking on the Save button, click on the Save button for the identity source configuration panel, and then re-open the service by selecting it:

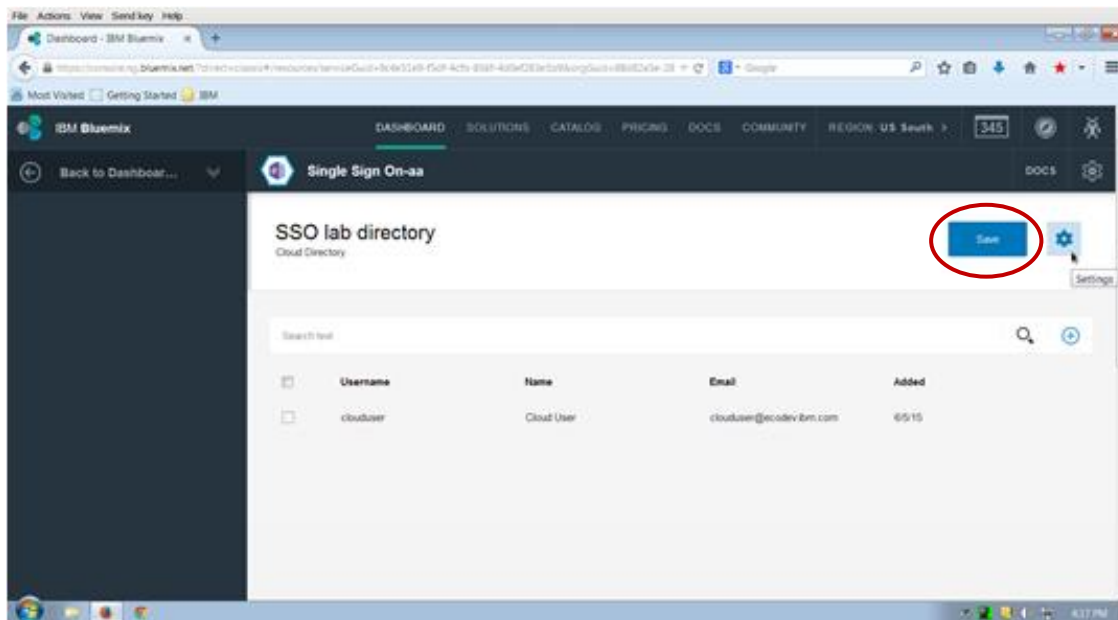


After re-opening the identity source configuration panel, a settings icon button will be shown in the upper right section of the panel. To manage the password policy applied to users in this identity source, you can click on the settings icon button for the identity source:

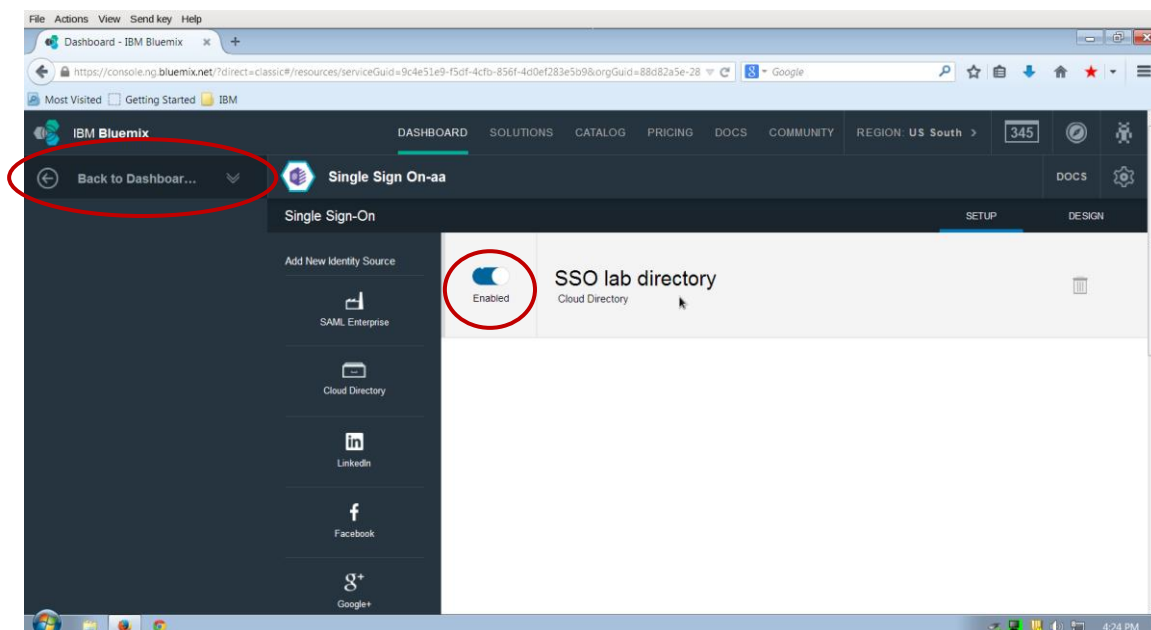


The settings control has two option panels. The Auto Consent option tells the SSO service if it should prompt the user after an authentication to allow the SSO service to retrieve personal (e.g. name and e-mail address) information from the identity source. Leave this setting to On for this lab to observe this prompt in action. The Password Policy option allows you to select various levels of password quality and lifetime. At the present time, these policies are fixed and you may select from one of the three. Choose Medium, and then click on Save.

Lastly, click on save in the identity source configuration panel to return to the main SSO configuration panel



The name shown for the identity source will be the name 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 on the Back to Dashboard link in the upper left to return to the IBM Bluemix Cloud Foundry dashboard.

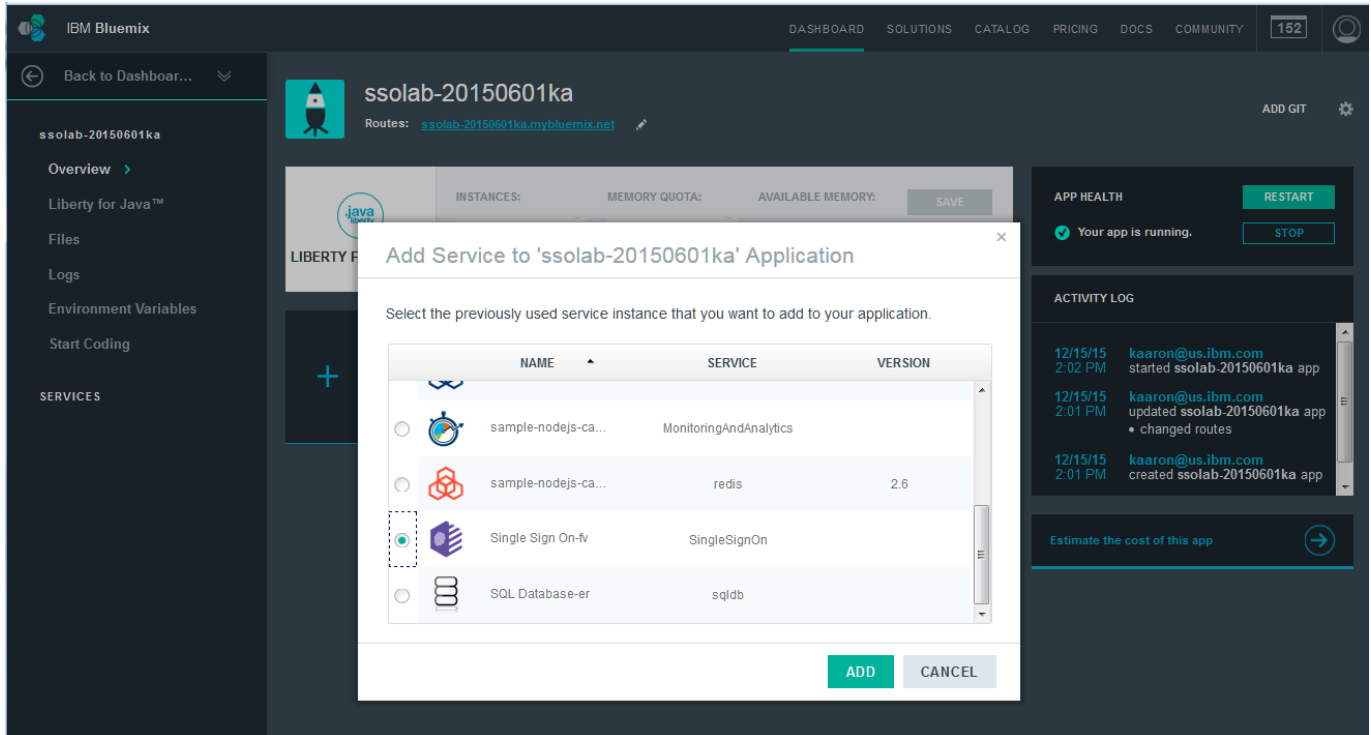


Bind the Single Sign On service to the application

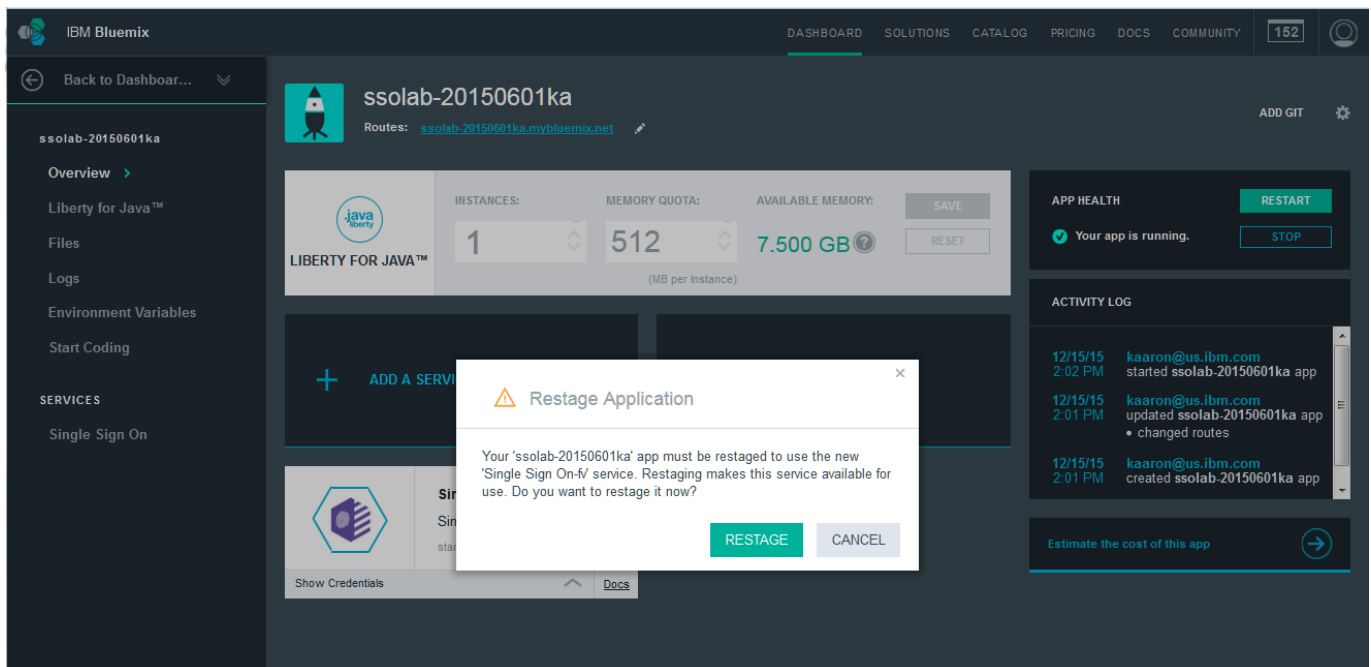
To bind the Single Sign On service with the configured identity source, first click on the application icon in the dashboard to bring up the application overview page. Then click on the Bind a Service or API button:

The screenshot shows the IBM Bluemix application overview page for an application named 'ssolab-20150601ka'. The page has a dark blue header with the IBM Bluemix logo and navigation links: DASHBOARD, SOLUTIONS, CATALOG, PRICING, DOCS, and COMMUNITY. A user profile icon is in the top right corner. On the left, a sidebar contains a 'Back to Dashboard...' link and a list of application management options: Overview (selected), Liberty for Java™, Files, Logs, Environment Variables, Start Coding, and SERVICES. The main content area displays the application's details: a rocket icon, the name 'ssolab-20150601ka', and its routes. Below this, there are three sections: 'INSTANCES' (showing 1 instance), 'MEMORY QUOTA' (512 MB per instance), and 'AVAILABLE MEMORY' (7.500 GB). There are 'SAVE' and 'RESET' buttons for these settings. Below these sections are two large buttons: 'ADD A SERVICE OR API' and 'BIND A SERVICE OR API'. On the right side, there is an 'APP HEALTH' section showing 'Your app is running.' with 'RESTART' and 'STOP' buttons. Below that is an 'ACTIVITY LOG' section showing a list of events: 'started ssolab-20150601ka app', 'updated ssolab-20150601ka app • changed routes', and 'created ssolab-20150601ka app'. At the bottom right, there is a link to 'Estimate the cost of this app'.

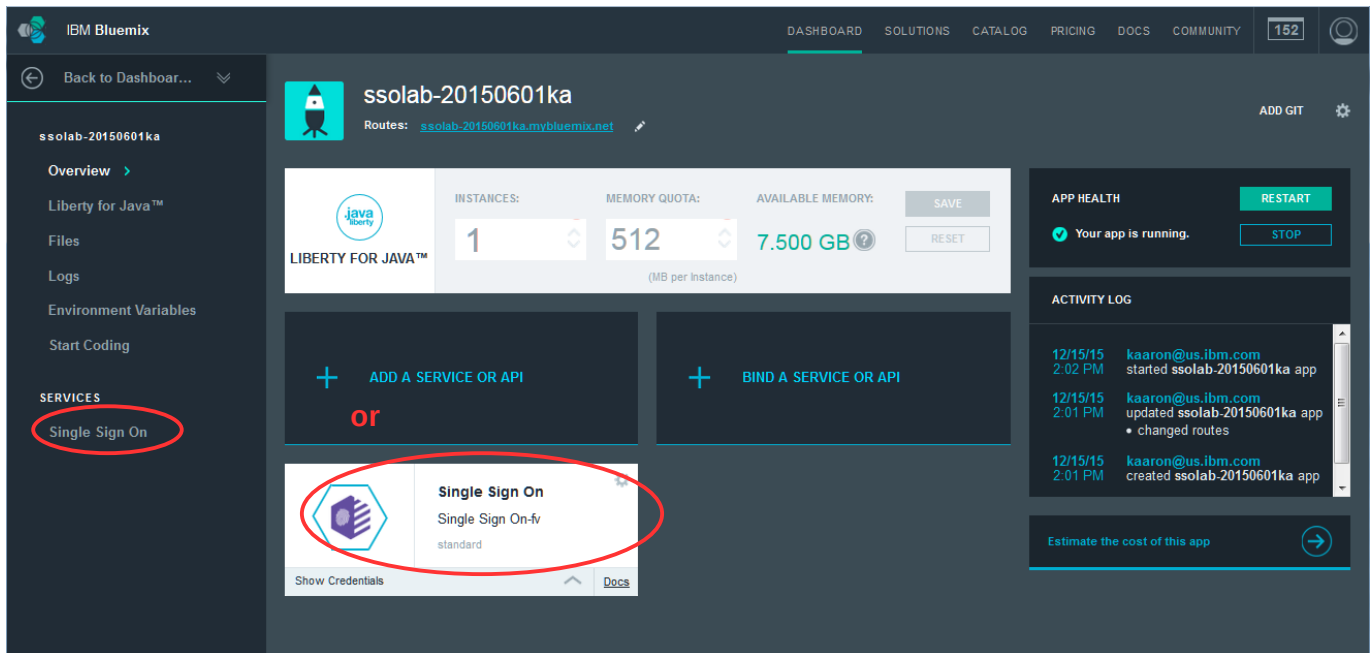
A panel will be displayed, select the radio button next to the Single Sign On service and then click on the ADD button:



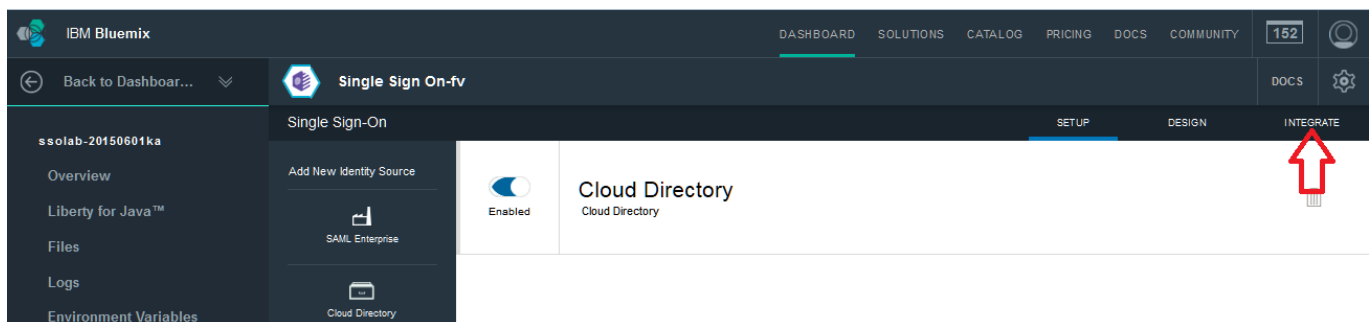
You will be prompted to restage the application. A restage is used to add the VCAP_SERVICES environment variables to the application so that it will be able to access the Single Sign On service bound to the application. Click on the RESTAGE button:



The restage and restart of the application will complete quickly, however, you do not need to wait before proceeding. On the left side of the application's overview panel, click on the Single Sign On label or click on the icon for the service. This will open the configuration panel for the service in the context of this application:



When accessing the configuration panel from within an application, an INTEGRATE tab will also be shown. For the Java application 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. Although we will return later to the INTEGRATE tab to finish configuring the application, click on the label to switch to this tab now:



Since this is binding to a Java app, the Return-to URL will already be set. The SSO Service will invoke the Return-to URL after authentication is completed.

For Liberty for Java Applications, the Single Sign On service leverages the OpenID Connect (OIDC) client feature from Liberty and the Bluemix Liberty build pack. As a result, Java applications running on Bluemix do not need to include any code to support the OpenID Connect protocol or Single Sign On.

(see

http://www.ng.bluemix.net/docs/services/SingleSignOn/configure_apps.html#tsk_configuringlibforjavaapp

for more info.) You will need to add some security constraints in the Java app to see the Single Sign On work.

IBM Bluemix

DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY 152

Back to Dashboar... Single Sign On-fv

ssolab-20150601ka

Overview

Liberty for Java™

Files

Logs

Environment Variables

Start Coding

SERVICES

Single Sign On >

Single Sign-On

SETUP DESIGN INTEGRATE

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:

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

Display name:

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

Enabled	Identity Source
<input checked="" type="checkbox"/>	Cloud Directory

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

You may 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. Once the Display name updated (if desired), click on the Save button:

IBM Bluemix

DASHBOARD SOLUTIONS CATALOG PRICING DOCS COMMUNITY 152

Back to Dashboar... Single Sign On-fv

ssolab-20150601ka

Overview

Liberty for Java™

Files

Logs

Environment Variables

Start Coding

SERVICES

Single Sign On >

Single Sign-On

SETUP DESIGN INTEGRATE

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:

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

Display name:

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

Enabled	Identity Source
<input checked="" type="checkbox"/>	Cloud Directory

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

Configure Application

Application configuration saved.

OK

After saving, click on Back to Dashboard in the upper left to return to the IBM Bluemix Cloud Foundry dashboard. At this point, the SSO service is ready to be used with the application.

Run the app

Now that it is all configured in Bluemix, go ahead and test that the app is working. From the Bluemix Dashboard, click the SSO application you created above. Click on the app url to open the application's home page.

The screenshot shows the IBM Bluemix dashboard for an application named 'ssolab-20150601ka'. The interface includes a left sidebar with navigation options like 'Overview', 'Liberty for Java™', 'Files', 'Logs', 'Environment Variables', 'Start Coding', and 'SERVICES'. The main content area displays the application's configuration, including a 'Routes' section with a URL 'ssolab-20150601ka.mybluemix.net' highlighted in a red box. Below this, there are sections for 'INSTANCES' (1 instance), 'MEMORY QUOTA' (512 MB), and 'AVAILABLE MEMORY' (7.500 GB). There are also buttons for 'ADD A SERVICE OR API' and 'BIND A SERVICE OR API'. A 'Single Sign On' service is listed with a 'Show Credentials' button. On the right, the 'APP HEALTH' section shows 'Your app is running.' with 'RESTART' and 'STOP' buttons. Below that is an 'ACTIVITY LOG' showing recent events. At the bottom right, there is a button to 'Estimate the cost of this app'.

You should see the home screen for the java starter app.

The screenshot shows the 'Hello World!' home screen for the Liberty for Java Starter Application. It features a circular icon on the left containing a screenshot of the application's interface. To the right of the icon, the text reads 'Hello World!' followed by a message: 'Thanks for creating a Liberty for Java Starter Application. Get started by reading our documentation or use the Start Coding guide under your app in your dashboard.' The background is a dark blue gradient.

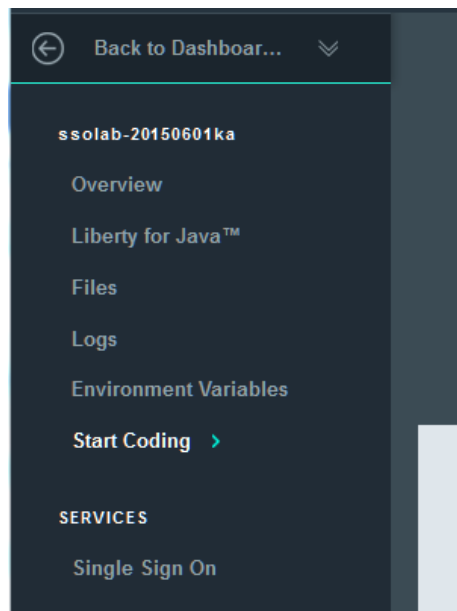
Since you have not put any security constraints in place, no pages in the app or protected.

Add security constraints to Java app

Now that Single Sign On is bound and configured to our java app, add security constraints so only the user you added to the Cloud Directory will be able to see the secure pages.

Download Starter Code

Now that your app and SSO is bound together, you can download the starter code. From the Bluemix dashboard, click your application to enter your application Overview. In the left menu, click Start Coding.



Scroll down 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 copy of your code.

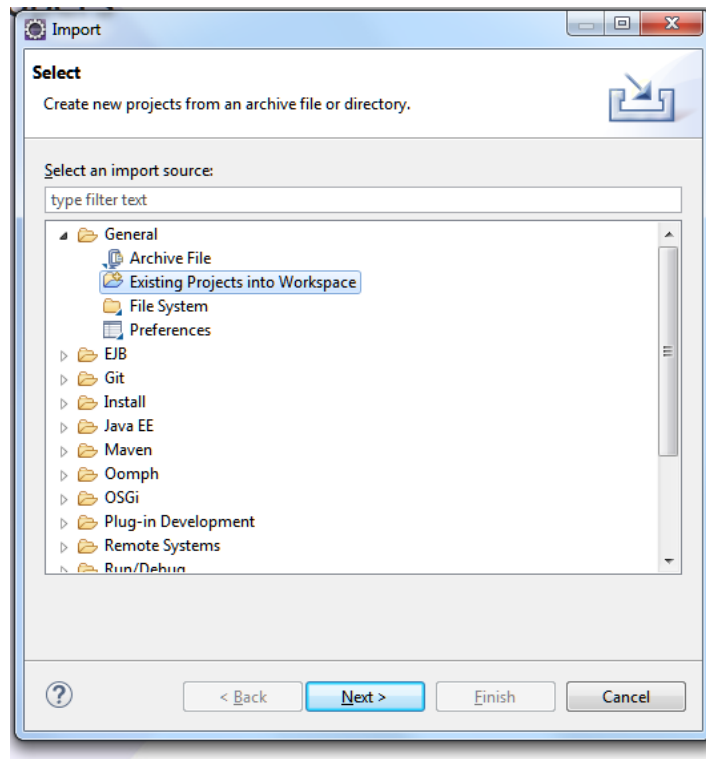
The screenshot displays the IBM Bluemix web interface. On the left is a dark sidebar with navigation links: 'Back to Dashboard...', 'ssolab-20150601ka', 'Overview', 'Liberty for Java™', 'Files', 'Logs', 'Environment Variables', 'Start Coding >', and 'SERVICES'. The main content area shows a 'Setup' guide with 7 numbered steps:

- Setup:** Before you begin, install the cf command line interface.
A button 'Download CF Command Line Interface' is shown.
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.
After the cf command line interface is installed, you can get started:
- 1** Download your starter code.
A button 'Download Starter Code' is shown.
- 2** Extract the package to a new directory to set up your development environment.
- 3** Change to your new directory.
A terminal snippet shows: `cd your_new_directory`
- 4** Connect to Bluemix™.
A terminal snippet shows: `cf api https://api.ng.bluemix.net`
- 5** Log in to Bluemix.
A terminal snippet shows: `cf login -u kaaron@us.ibm.com -o kaaron@us.ibm.com -s dev`
- 6** Deploy your app to Bluemix. For more information about cf push command, see [Uploading your application](#).
A terminal snippet shows: `cf push ssolab-20150601ka`
- 7** Access your app by entering the following URL into your browser: <http://ssolab-20150601ka.mybluemix.net>

Now that you have the code, it is time to update the application to have some security constraints.

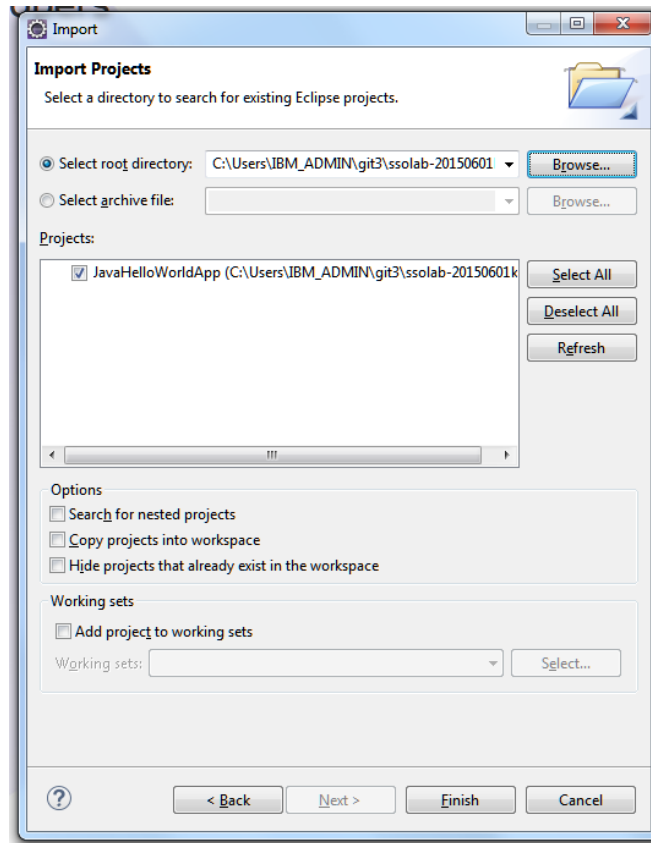
Import into Eclipse

Next, open Eclipse. Select File -> Import.. . Select General -> Existing Projects into Workspace in the Import Select dialog.



Select Next.

In the Import Projects dialog, click the Browse button and browse to the unzipped code you downloaded from Bluemix.



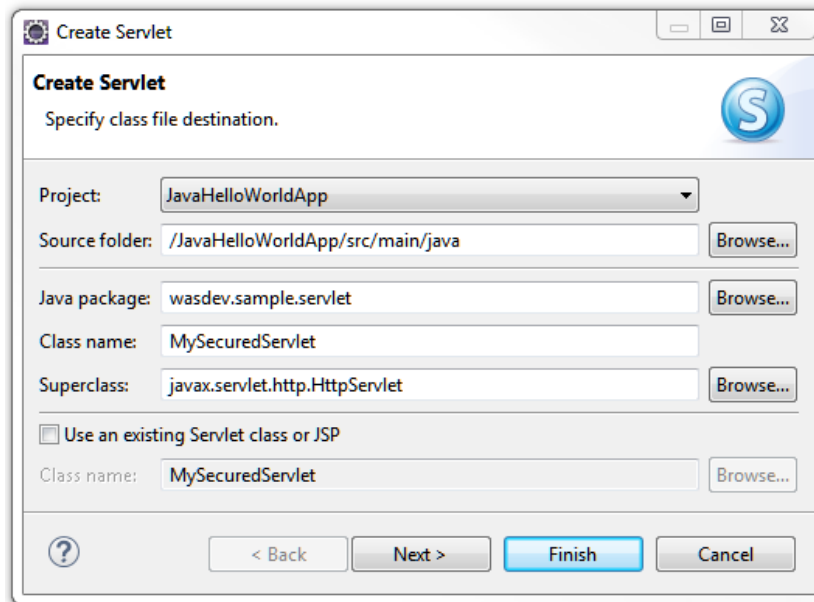
Click Finish.

Add servlet to protect

Now add a servlet to protect from all users except those authenticated by SSO.

In Eclipse, select File -> New -> Servlet.

Create a servlet named MySecuredServlet.



Click Finish.

The servlet you just created should already have a@WebServlet annotation set.

```

10 /**
11  * Servlet implementation class MySecuredServlet
12  */
13 @WebServlet("/MySecuredServlet")
14 public class MySecuredServlet extends HttpServlet {
15     private static final long serialVersionUID = 1L;
16
17     /**
18      * @see HttpServlet#HttpServlet()
19      */
20     public MySecuredServlet() {
21         super();
22         // TODO Auto-generated constructor stub

```

Update the response text in the doGet method to be a custom message you want to see.

```

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    // TODO Auto-generated method stub
    response.getWriter().append("You made it to the secured area. <b>Congrats!</b> ");
}

```

Next, open index.html.

Add an html link to the new servlet you created.

```

<p class='description'></p> Thanks for creating a <span class="blue">Liberty for Java Starter
href="https://www.ng.bluemix.net/docs/#starters/liberty/index.html#liberty">documentation</a>
or use the Start Coding guide under your app in your dashboard.
<p><a href="/MySecuredServlet">My Secured Servlet</a></p>
</td>

```

This will add a link from the home page to the secured servlet.

Add http constraints

Now you can protect the servlet. You are going to use javax annotations to secure the MySecuredServlet page.

First you need to include the libraries for the annotations.

Open pom.xml

There are several ways to add a dependency in the pom file. This example shows adding raw xml. So, click the pom.xml tab to see the raw xml for the pom file.

In the dependencies, add a dependency for jsr250-api.

```
<dependencies>
  <dependency>
    <groupId>org.apache.geronimo.specs</groupId>
    <artifactId>geronimo-servlet_3.0_spec</artifactId>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>javax.annotation</groupId>
    <artifactId>jsr250-api</artifactId>
    <version>1.0</version>
  </dependency>
</dependencies>
```

Next, open MySecuredServlet.java file.

Add a the servlet security and HTTP constraint annotations as shown in the next image.

```
@WebServlet("/MySecuredServlet")
@WebServletSecurity([@HttpConstraint(rolesAllowed={"any-authenticated"})])
public class MySecuredServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public MySecuredServlet() {
        super();
        // TODO Auto-generated constructor stub
    }
}
```

This will protect all access to this servlet except by those authenticated users who have the role of 'any-authenticated'.

Map roles

Next you are going to map the role 'any-authenticated' to a valid subject.

Add a folder named 'resources' under the src/main folder.

Add a folder named 'META-INF' under the newly created resources folder.

Create a new file named ibm-application-bnd.xml

Add the xml you see in the next image

```
<?xml version="1.0" encoding="UTF-8"?>
<application-bnd xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://websphere.ibm.com/xml/ns/javaee"
  xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee
    http://websphere.ibm.com/xml/ns/javaee/ibm-application-bnd_1_0.xsd"
  version="1.0">

  <security-role name="any-authenticated">
    <special-subject type="ALL_AUTHENTICATED_USERS" />
  </security-role>

</application-bnd>
```

This says that All Authenticated Users get the role of 'any-authenticated'.

Lastly, open the pom.xml again. You will need to inform maven to use the src/main/resources files when it builds the war file. In the build -> pluginManagement -> plugins -> maven war plugin, add the following

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-war-plugin</artifactId>
  <configuration>
    <failOnMissingWebXml>false</failOnMissingWebXml>
    <webResources>
      <resource>
        <directory>${project.basedir}/src/main/resources</directory>
      </resource>
    </webResources>
    <warName>JavaHelloWorldApp</warName>
  </configuration>
</plugin>
```

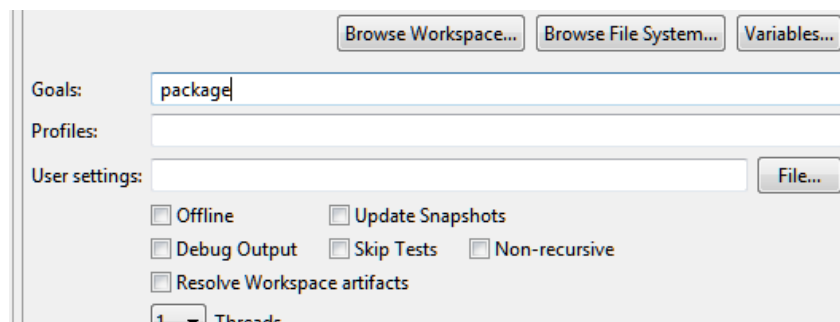
Prepare and push your project

You are almost ready to push the project to Bluemix.

Compile the war by right clicking on pom.xml from Enterprise Explorer view.

Select Run As -> Maven build... from the context menu.

In the goal field, type package.



Then click the Run button.

Check the Console view and make sure your build was successful.

Open a command prompt.

Go to the directory that contains the project you have been working on.

Using the cloud foundry command, login to Bluemix.

Once logged in, push your app. Note, you must be in the directory that has the manifest.yml file.

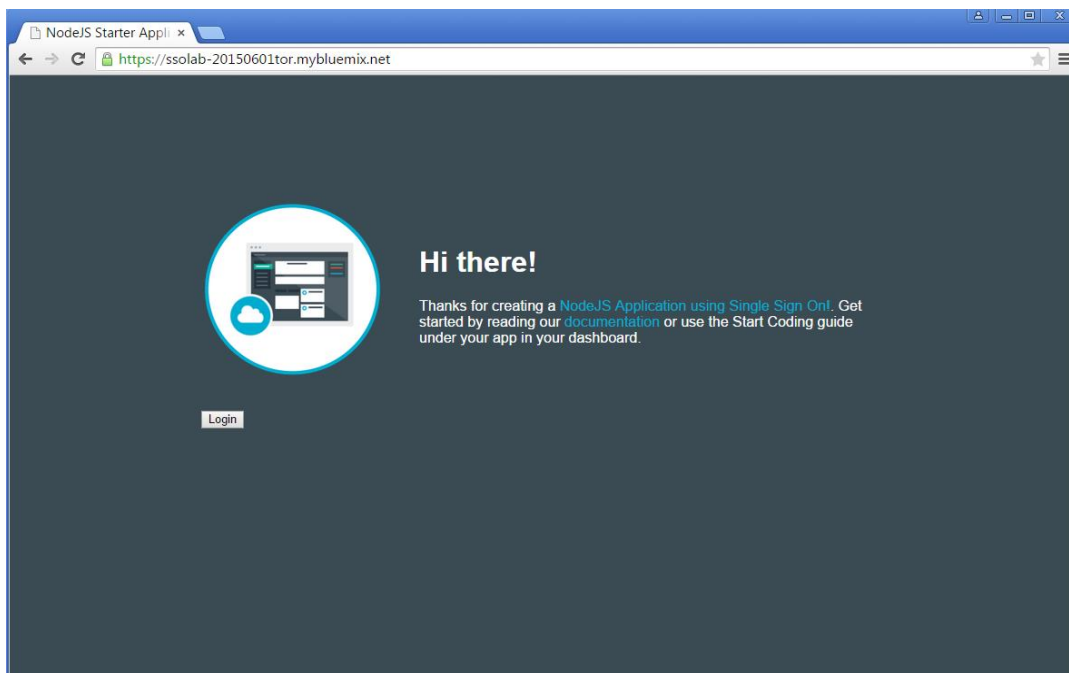
Run the command `cf push yourappname`, where yourappname is the name of the app on Bluemix.

Once your app is pushed and restarted, you are ready to test.

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 screenshots.

Begin by opening your application's url: `https://<appname>.mybluemix.net` (be sure to use https!):



Note that the My Secured Servlet link is at the end of the paragraph. Click on this link to bring up the Single Sign On identity provider chooser list:

Since the Single Sign On service is configured with only one identity source, only one option is shown. Click on Cloud Directory and sign in as the user you added to the directory previously:

Username and Passw. x

← → ↻ <https://bestssoever-2zgs9sg44w-cge7.iam.ibmcloud.com/idaas/mtfim/sps/authsvc?PolicyId=urn:ibm:security:authentication:asf:basicdapuser> ★ ☰

Sign In

Username
clouduser

Password

Login

After you click on the Login button – a prompt will be displayed (this happens by policy when using the Cloud Directory) stating that the user password has been reset and must be changed. Note that requirements for the password are displayed in the form. Enter in an updated password:

Change Password x

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

Change Password

Username: clouduser

FBTBLU105! The password has been reset. You must change your password.

New password:

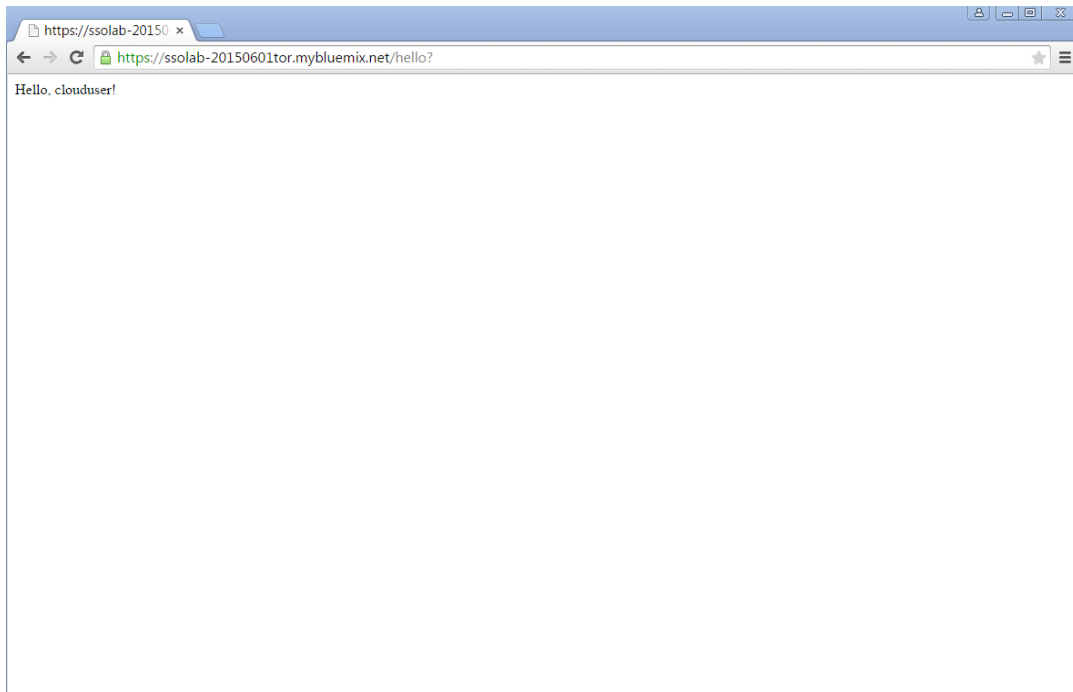
Confirm new password:

Change Password

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

Next, there will be a panel asking for permission (from the application name you set in the INTEGRATE panel) to retrieve your personal information from the external authentication identity provider. You may select either Allow or Allow and Remember. If you select Allow, then each time 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:



Summary and next steps

Now you have an application that can use the Single Sign On service with a simple identity provider. A very good next step would be to continue on with the Bluemix SSO documentation to add a Social Media identity provider. There are step-by-step instructions for this as well as configuration of SAML Enterprise identity sources. To access the documentation, from the configuration panel for the Single Sign On service, click on the DOCS button:

