**ICompare - Price comparison**

A Project Report

Presented to

The Faculty of the Computer Engineering Department

San Jose State University

In Partial Fulfillment

Of the Requirements for the Degree Master of Science in Software Engineering

By

Amit Kumar Rajoriya, Anushri Srinath Aithal, Mojdeh Keykhanzadeh, Ashwini Shankar Narayan

May 04, 2018

**Table of Contents**

**Chapter 1. Introduction................................................................................................. 3**

1.1. Purpose.................................................................................................................... 3

1.2. Product Scope ......................................................................................................... 3

1.3. Problem and Motivation ........................................................................................... 4

1.4. Market Research ..................................................................................................... 4

1.5. References……………............................................................................................. 5

**Chapter 2. System Design............................................................................................ 6**

2.1. System Architecture Design .................................................................................. 7

2.1.1. Single sign on architecture………........................................................................7

2.1.1. Complete system architecture..............................................................................8

**Chapter 3. System Implementation............................................................................ 9**

3.1. System Implementation Summary ......................................................................... 19

3.2. System implementation issues and resolutions ..................................................... 20

3.3. Used Technologies and Tools ............................................................................... 26

3.4. User Interface......................................................................................................... 26

**Chapter 4. System Testing and Experiment.............................................................. 28**

4.1. Testing Scope…..................................................................................................... 29

4.2. Testbed Set............................................................................................................ 30

4.3. Testing Scenarios & Cases ................................................................................... 33

**Chapter 5. Conclusion and Future Work................................................................... 34**

5.1. Project Summary.................................................................................................... 34

5.2. Future Work ........................................................................................................... 34

# **1.** **Introduction**

## **1.1** **Purpose**

When a user compare price for any product, their intention to buy that product is higher and user is close to final sale. A good price comparison site helps user to find good price and influence their decision for final sale. The project purpose is to have a price comparison engine that helps buyers make informed decisions. Price comparison engines not only saves money by providing the users the best deals but also helpers buyers save time of searching across various retailer website. The application also allows buyers to navigate to website that sells the actual website and let them complete the buying transaction. The users can also save the product of their interest on Wishlist. The application is a one-stop portal to compare prices and purchase the best deals.

The application “Administrator” is provided with a portal that can be accessed via Cloud Single Sign-On implemented using Okta. The admin can manage updating the API keys and secret at frequent intervals by accessing SSO based cloud app. The admin can also access cloud-based Jenkins which is integrated with SSO.

**1.2** **Product scope**

**ICompare Portal:**

ICompare portal allows users to make informed decision before placing order for a product. It lets users make quick decision by providing a price comparison the buyers can search for product without having to login. The result displayed contains product name, least price, price from various vendors, product image, brand, product short description and links to websites. This Portal allow users to access price comparison web application without login. The users can save the search result by adding it to a Wishlist. When user tries to add a product to Wishlist, they are redirected to login. Login is handled by Amazon Cognito integrated with Google and Facebook. Under Wishlist they can buy the product or remove from Wishlist. The user can search for desired product without logging in to the portal. The result will be shown according to search criteria. Also, the result will be sorted by lowest price available.

**Admin Portal:**

The admin portal is developed with the idea of providing a hassle-free login experience for administrators and developers to update api keys and access the SaaS apps within the organization. When the API keys expire, admin can update it via this portal. The CICD apps like Jenkins are accessible via this portal, all these apps are integrated with Okta for a SAML based SSO. Admin can configure and trigger builds to ICompare portal using Jenkins.

## **1.3** **Problem and motivation**

The drastic change in shopping trends from offline to online has led to many online retailers. Companies like Amazon, eBay have made shopping easy within the comforts of our own homes. However, often we always wonder if we made the right choice. Especially, if we paid the right amount, if it was the best deal we could have got?

Price comparison shopping is a general trend observed in online shoppers. Buyers are interested in getting the best price without investing much time in searching across various websites. Merchants are benefitted by observing the price for their products from other vendors and set a competitive price to attract more consumers.

Identity and Access management in most companies are achieved using Microsoft Active Directory(AD) or Lightweight Directory Access Protocol(LDAP). With the advance in cloud applications one can see that each cloud app maintains its own user store leading to redundant code and security maintenance. This problem only gets bigger with companies adopting more applications and managing user password lifecycles, activation, deactivation. In the proposed solution, a cloud based Active Directory is integrated with Okta to provide identity management.

Password fatigue is a common feeling expressed by most users. It is quite challenging and exhausting to remember excessive number of passwords for all the daily apps that one uses. It is not only to remember them but also to change them frequently. Most people end up reusing passwords or use insecure and weak passwords. Also, IT department in most companies must manage multiple vendors and cross platform cloud applications. The situation worsens when employees are locked out and takes hours to unlock accounts. The proposed solution in this project is to implements Single Sign-On using SAML 2.0 along with Okta AD integration.

## **1.4** **Marketing research**

There are plenty of website available in the market for consumer product price comparison but most of them are biased and run for their benefits. There are few sites only which provide best prices without any bias. So, we decided to build our own price comparison portal to fill the gap.

## **1.5** **References**

1. Semantics3 API reference (<https://docs.semantics3.com/>)
2. Wider report (https://www.gov.uk/government/news/cma-steps-in-to-give-people-a-better-deal-on-comparison-sites)
3. Mocha Test Automation (<https://mochajs.org/#getting-started>)
4. Node JS & Express Frame (<http://expressjs.com/en/starter/installing.html> )
5. AWS Directory Services Connectivity(<https://www.youtube.com/watch?v=SIBCi76wspQ>,<https://docs.aws.amazon.com/directoryservice/latest/admin-guide/microsoftadbasestep3.html>)
6. Jenkins-SSO(<https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-16-04>

<https://jansipke.nl/enable-https-jenkins/>

<https://www.digitalocean.com/community/tutorials/openssl-essentials-working-with-ssl-certificates-private-keys-and-csrs>

<https://support.cloudbees.com/hc/en-us/articles/115000105752-How-do-I-setup-OKTA-as-Identity-Provider-in-Jenkins-> )

7. Admin-Portal SSO

([https://developer.okta.com/blog/2017/03/16/spring-boot-saml#create-a-spring-boot-ap plication-with-saml-support](https://developer.okta.com/blog/2017/03/16/spring-boot-saml#create-a-spring-boot-application-with-saml-support)

<http://www.baeldung.com/spring-boot-app-as-a-service>)

8. Mocha-Chai testing

(<https://scotch.io/tutorials/test-a-node-restful-api-with-mocha-and-chai>

)

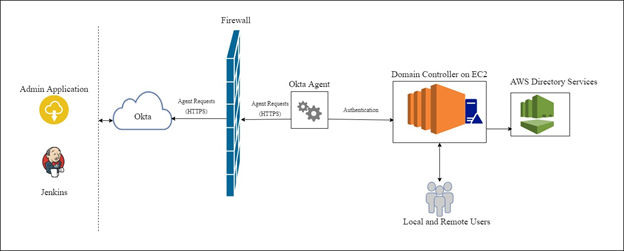
**2. System Design**

## **2.1 System Architecture Design**

**2.1.1 Single Sign-On Architecture:**  
  
 Okta’s cloud-based identity and access management service can be integrated with AWS Directory service to provide seamless and highly available solution to identity management problems. In this solution, a cloud based Active Directory is set up where in AWS Directory Services is used to provide Identity Management in connection with Microsoft Windows 2016 R2 Server deployed on EC2 instance. Okta Agent is installed on the domain controller that allows communication between domain controller and Okta to perform authentication and provisioning. Active Directory users are then imported on Okta sub-domain to perform authentication. Once set up, Okta provides an infrastructure that allows to manage both existing users and add new user identities.

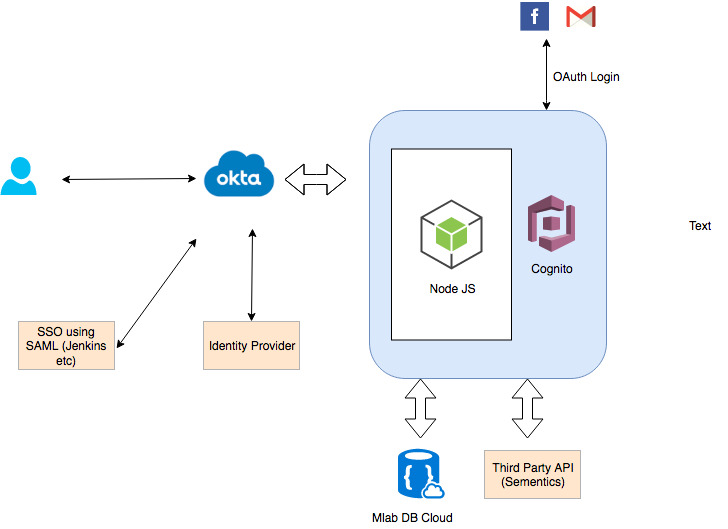
Once, Okta and AD integration is completed we can achieve Single Sign-On using Okta’s SAML 2.0 based plugins. Security Assertion Markup Language (SAML) is an XML based protocol that provides a secure communication between two vendors. When two vendors, an Identity Provider (IDP) which is AD integrated with Okta in our solution and a Service Provider (SP) - Jenkins and the custom Admin application, both implement SAML, they can authenticate accredited users associated with AD to allow them to use the services offered by the SP.

When an admin user attempt to access Jenkins or the custom application via invoking the URL on a browser the federated identity software at Okta is activated and confirm the user’s identity. The IDP notifies the SP about this authentication. A tokenized message which includes user details and permissions is exchanged and SP determines that the message is from a trusted IDP and creates a user session.



**Fig. Okta AWS AD Directory Service Integration**

**2.1.2 Complete system Architecture**

****

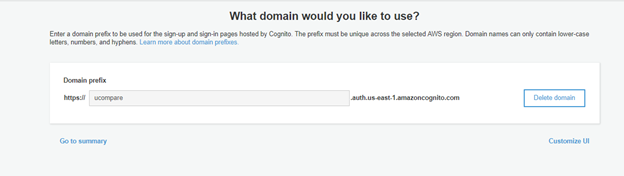
# **3. System Implementation**

## **3.1 System Implementation Summary**

**Cognito Setup**

Create a user pool and customize with the following configurations.

1. Add domain to user pool in App Integration

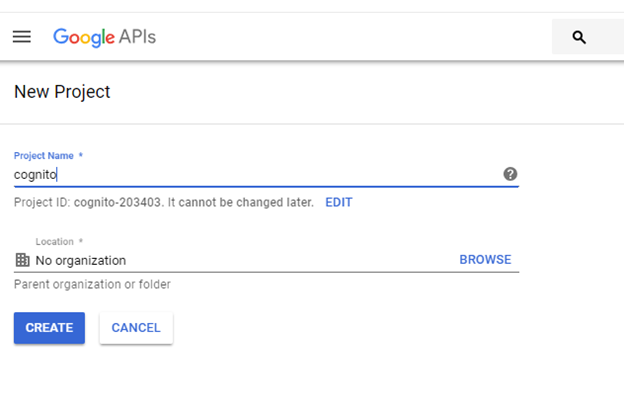


1. To enable Google and Facebook login, create Google and Facebook app by using developer account to get credential for user pool. Following is the steps to follow

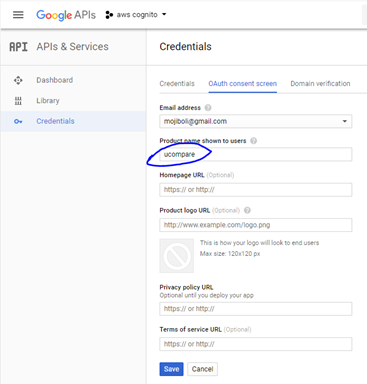
**Google**

Sign in to <https://console.developers.google.com/> and create an app with following setup

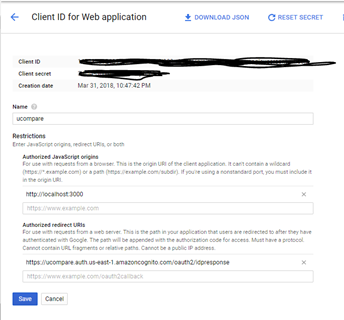
* Create project as first step



* Go to credentials to create credentials, specify product name to be able to proceed

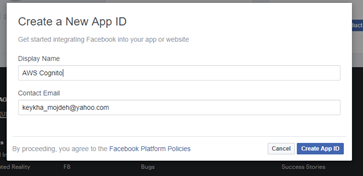


* In 0Auth consent screen, choose web application and specify the Authorized JavaScript origins and Authorized redirect URIs and after creating, the window popup with credentials (Client ID and Client Secret)

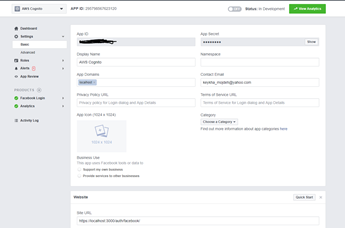


**Facebook**

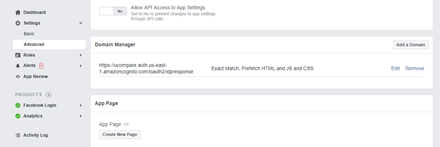
* Login to <https://developers.facebook.com/apps> and add a New App



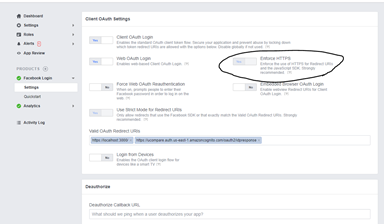
* In app Settings/Basic specify App Domain and Site URL



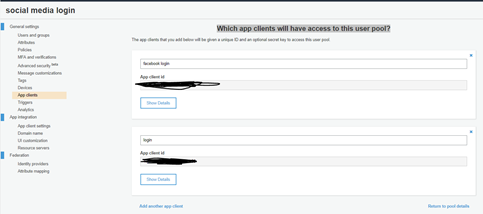
* In Setting/Advanced, specify the Domain Manager which is the domain name of user pool



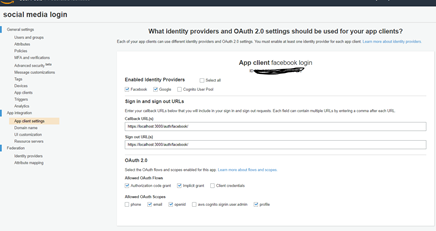
* In Facebook login /Settings specify Valid OAuth Redirect URIs. Note that Facebook Enforce HTTPS.

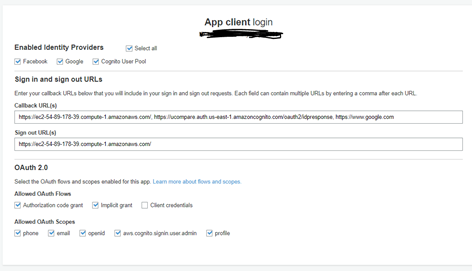


1. Back in User Pool, specify App clients (Facebook login is for Facebook and login for google in this case)

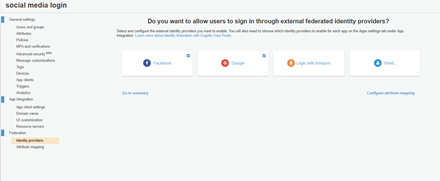


1. In App client settings choose Facebook and Google and specify the Callback URL(s) and Sign out URL(s) as well as Allowed OAuth Flows for both Facebook and Google

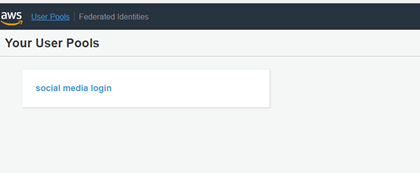




1. Under Federation / Identity providers, need to choose Facebook and Google



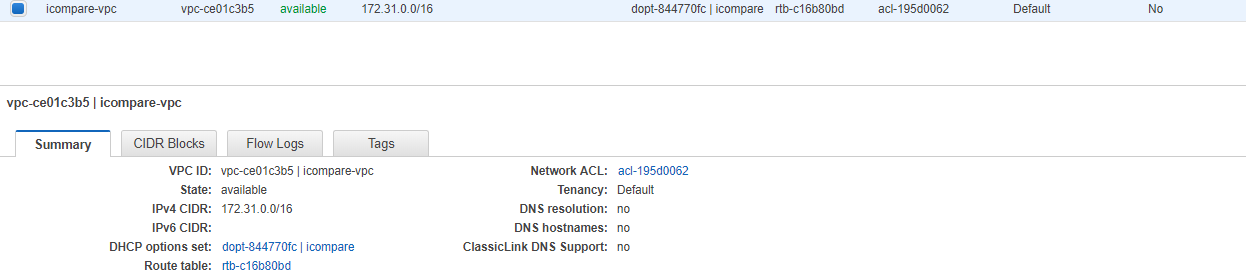
1. Following is the user pool that created following the steps above



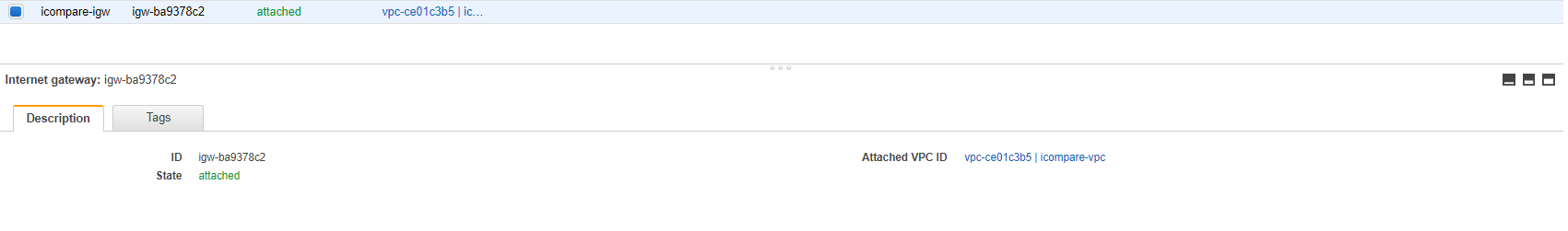
**Directory Services Set-up**

AWS Directory Service is used to set-up Microsoft Active Directory as the users store for Admin Applications. Below are the steps to configure Directory Services

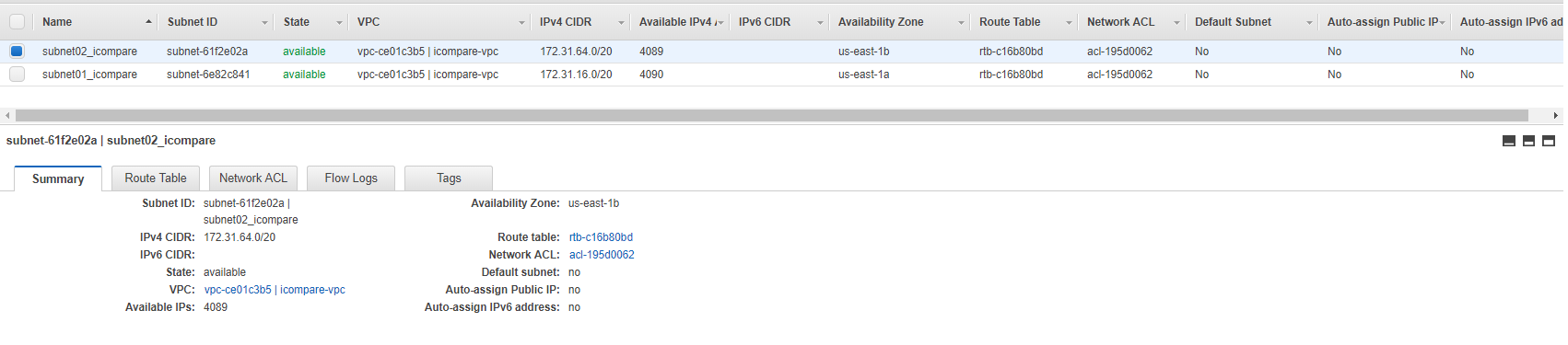
1. Create a VPC in which you want to set-up the Directory Service



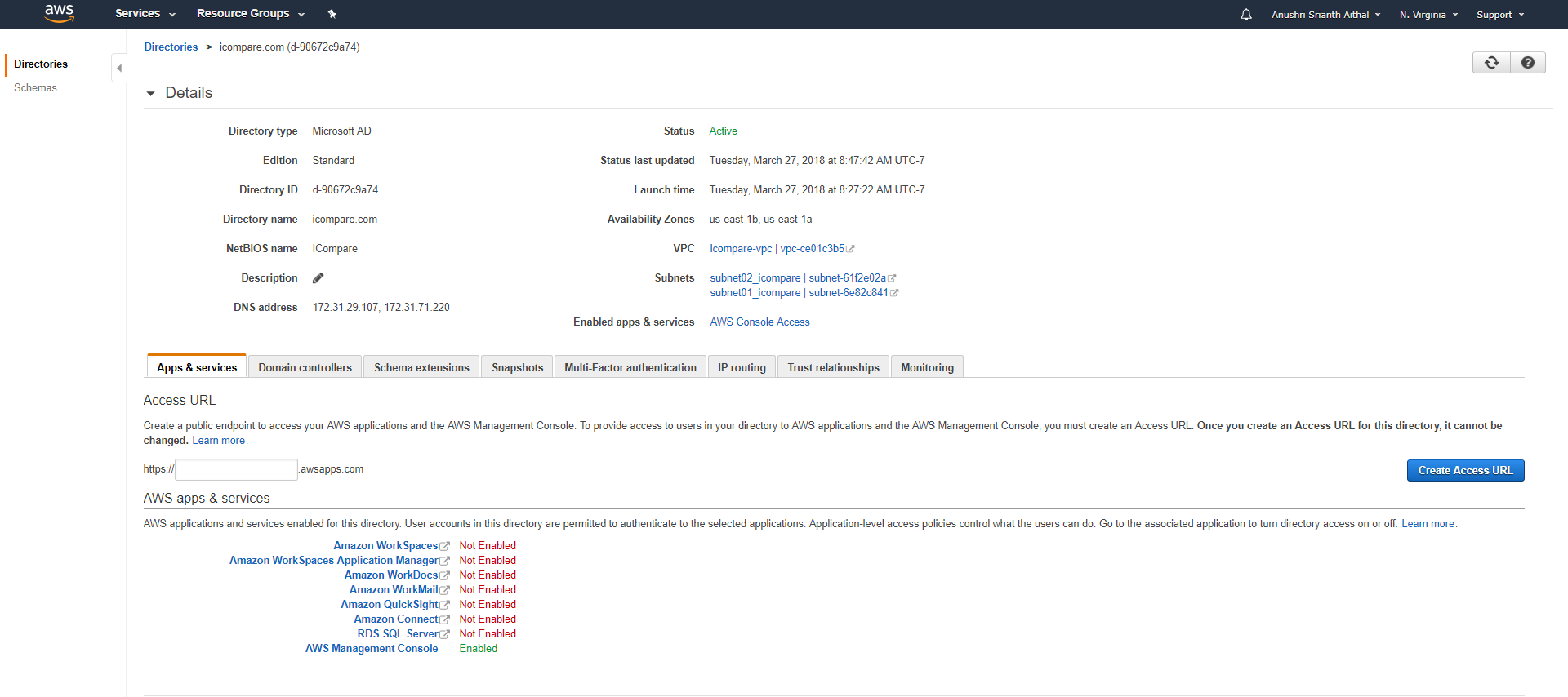
1. Set-up an Internet Gateway Configuration and associate it with the VPC created in step 1.



1. Create 2 Subnets in 2 different Availability Zones. On the Route Table tab, verify that there is a route with 0.0.0.0/0 as the destination and the internet gateway for your VPC as the target.



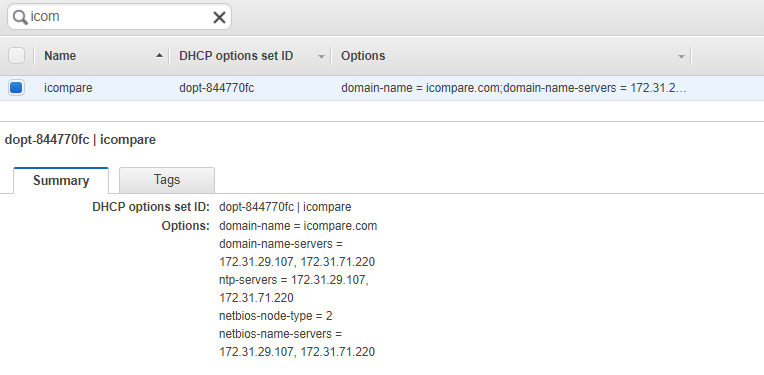
1. Navigate to AWS Directory Services, choose Microsoft AD and configure the AD Service as mentioned in the below image. Associate the previously created VPC and Subnets with this instance.



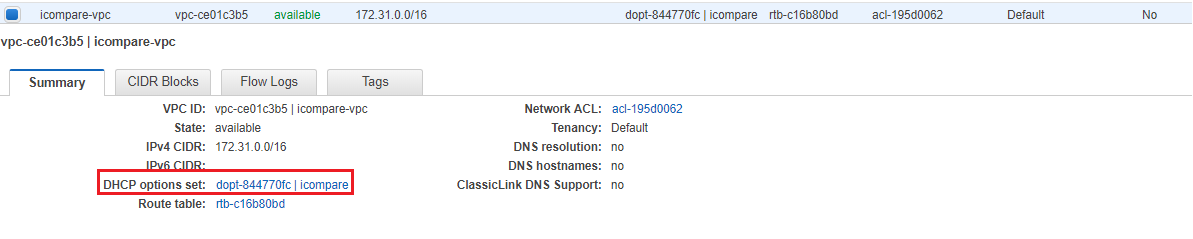
**EC2 and Directory Service Integration**

Once the AWS Directory Service is configured, we must instantiate EC2 instance with Microsoft Server 2016 as the flavor and promote that server as our AD’s domain controller. To achieve this the below steps, must be performed

1. Create a DHCP option set for the AWS Directory Service Created in above step. Associate this DHCP option set with the VPC created for AWS Directory Services.

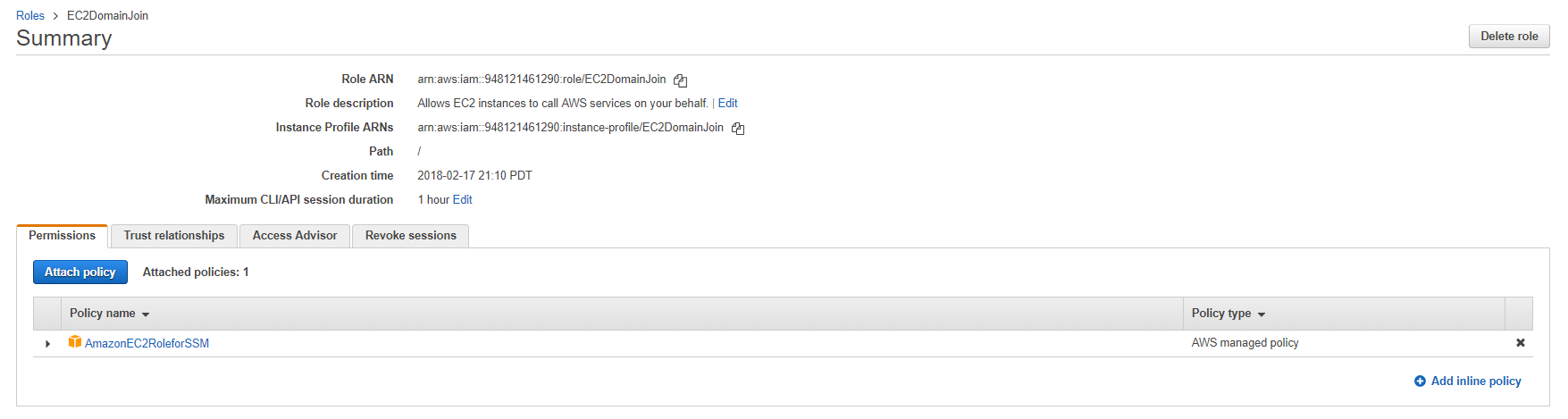


**Fig. DHCP Option Set**

****

**Fig. Associate DHCP with the VPC**

1. Create an IAM Role for the EC2 instance to be associated with the directory service. Assign AmazonEC2RoleforSSM to this IAM Role.



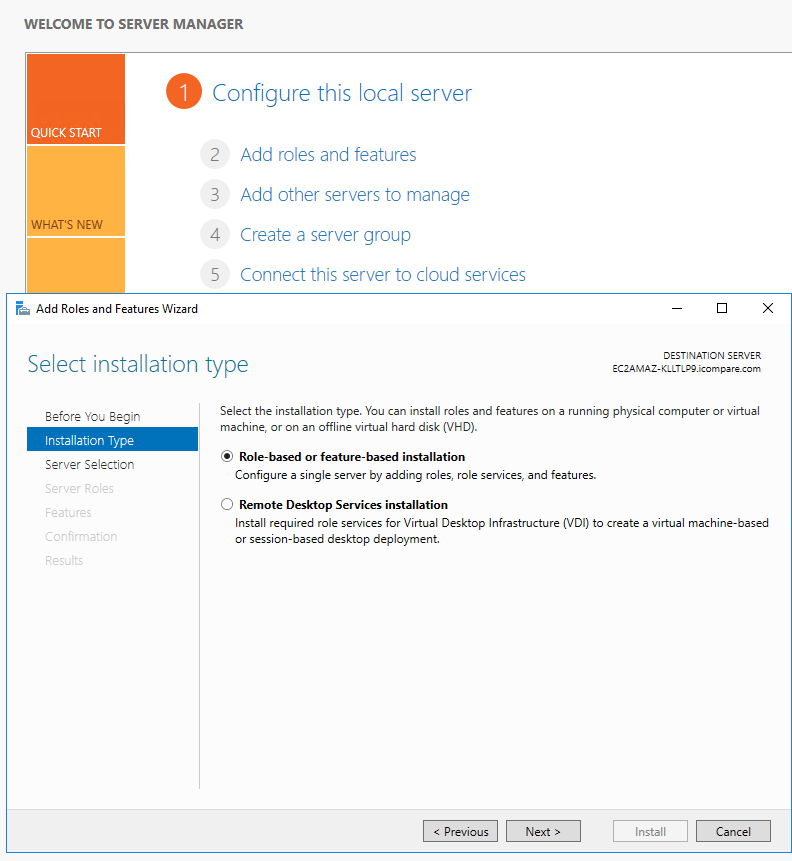
1. Launch EC2 instance from the console. For the network configurations, choose the VPC and subnet that was created during AWS Directory set-up. Also, associate the newly created IAM role in step 2.



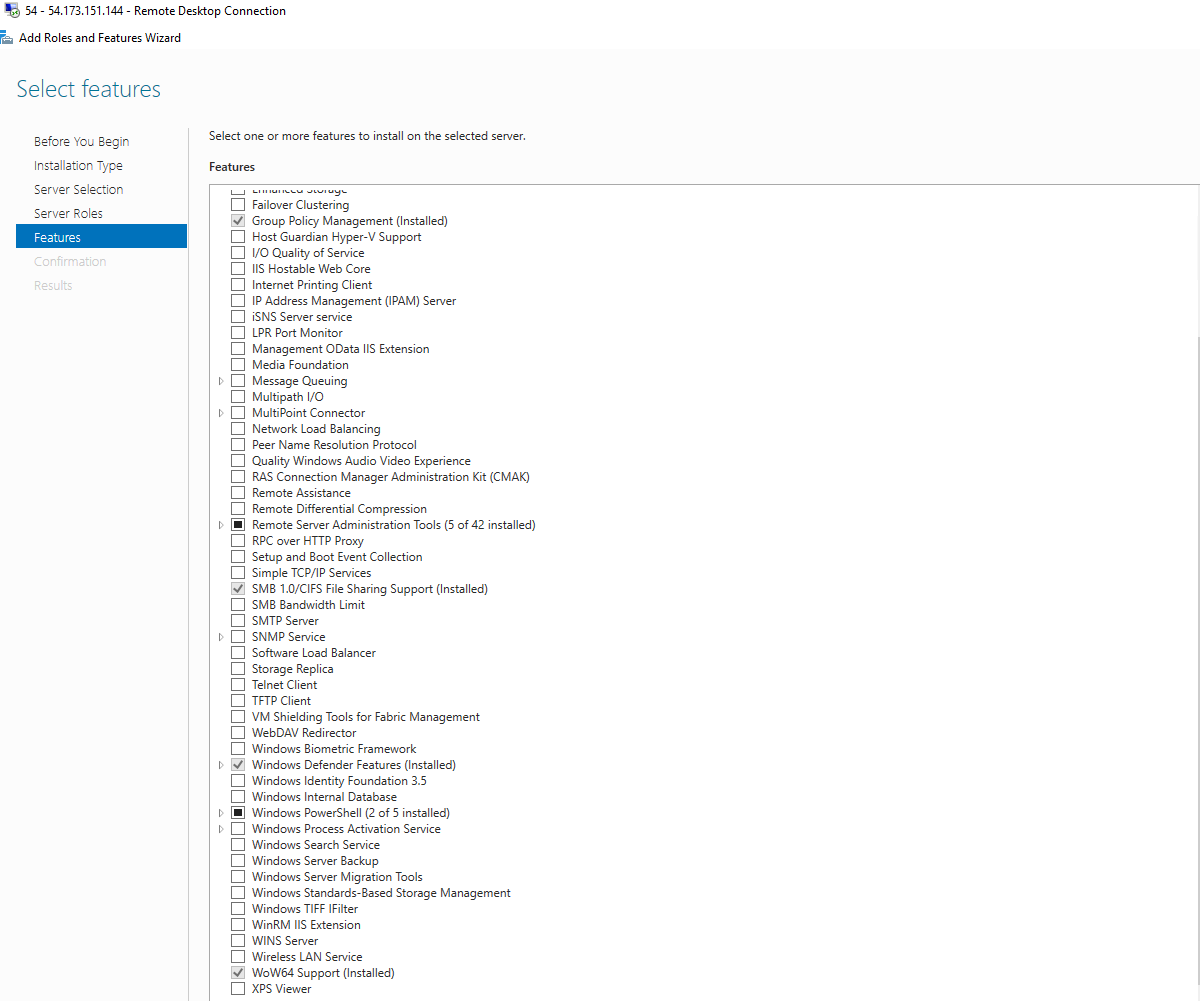
**AD and Okta Integration**

Once the AWS Directory Service is configured and EC2 is connected to this directory service we can proceed to set up AD components on EC2. The below steps are followed to promote EC2 as the domain controller.

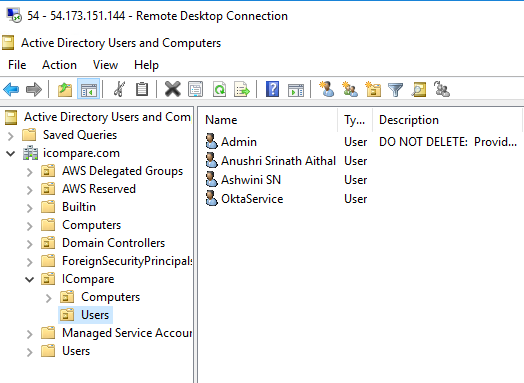
1. Use Remote Desktop(RDP) to connect to your windows EC2 instance. To promote this server as domain controller we need to set up appropriate AD components using the server manager. Once connected via RDP, open server manager installation type chooses role-based or feature based installation.



1. On features page, select Group Policy Management, Role Administration Tools, AD DS and AD LDS Tools, DNS Server Tools.

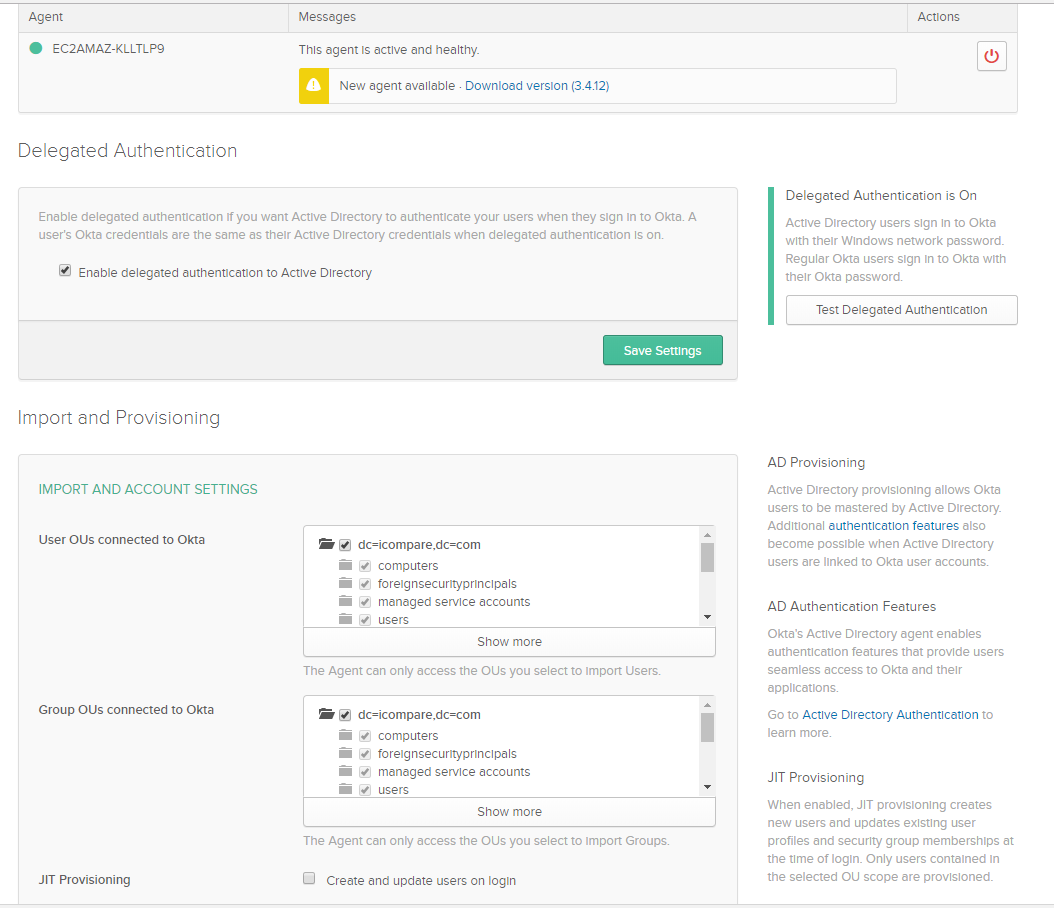


1. Confirm all the installation. Once installation is complete the server is promoted as the AD domain controller.
2. Navigate to Active Directory Users and Computers. Click on the domain icompare.com. Add new users under users tab.

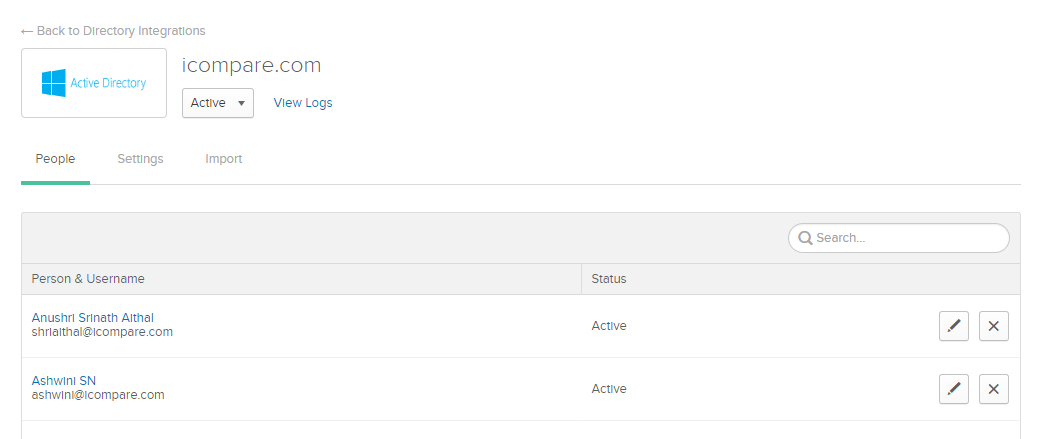


We now integrate our AD instance with Okta. The below steps illustrate Okta-AD integration

1. Create a developer account in account and create a subdomain. In this project we have created a sub-domain sjsuicompare.okta.com
2. Navigate to Directory -> Directory Integrations and click on integrate Active Directory.
3. Download the Okta AD agent setup on the domain controller. Use the domain <https://sjsuicompare.okta.com> to set up the agent.
4. Create an AD Admin user for Okta in AD users and groups. Use this as the administrator for Okta AD Integration.
5. Once the agent is set-up, import the OUs and Groups.



1. Import all the users to Okta.



**Jenkins Set-up**

Below steps illustrate setting up Jenkins on EC2 Ubuntu machine and configuring it to run on HTTPS

1. Instantiate an EC2 Ubuntu instance.
2. Execute the below commands to install Jenkins on Ubuntu

wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -

sudo apt-get install Jenkins

1. To setup Jenkins on HTTPs we need the SSL certificates. Run the below commands to obtain the certificate

openssl pkcs12 -inkey icompare.com.key -in icompare.com.crt -export -out keys.pkcs12

keytool -importkeystore -srckeystore keys.pkcs12 -srcstoretype pkcs12 -destkeystore /var/lib/jenkins/jenkins.jks

1. Use the password entered when prompted in step 2 to configure jenkins property in /etc/sysconfig/jenkins

JENKINS\_ARGS="--webroot=/var/cache/$NAME/war --httpPort=$HTTP\_PORT --httpsPort=443--httpsKeyStore=/var/lib/jenkins/jenkins.jks--httpsKeyStorePassword=ICompare1!"

1. Open firewall on ubuntu for port 8443 using below command

sudo ufw allow 8443

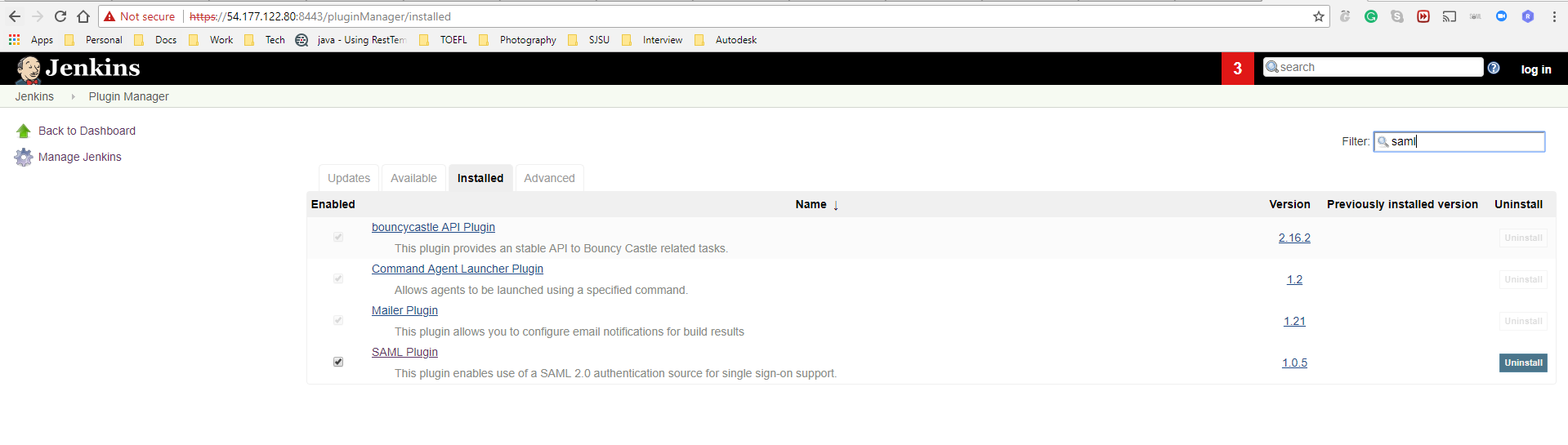
1. Run Jenkins using the following command

sudo systemctl start jenkins

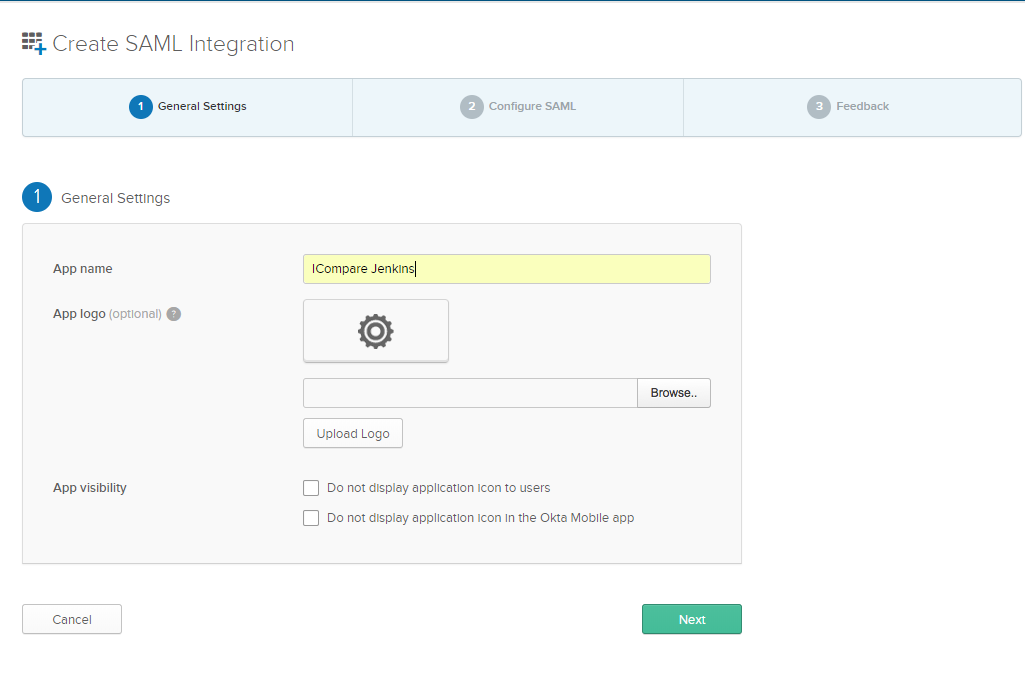
**Jenkins SSO using Okta**

Below steps are performed to set-up SSO for Jenkins using SAML 2.0 plugin and Okta.

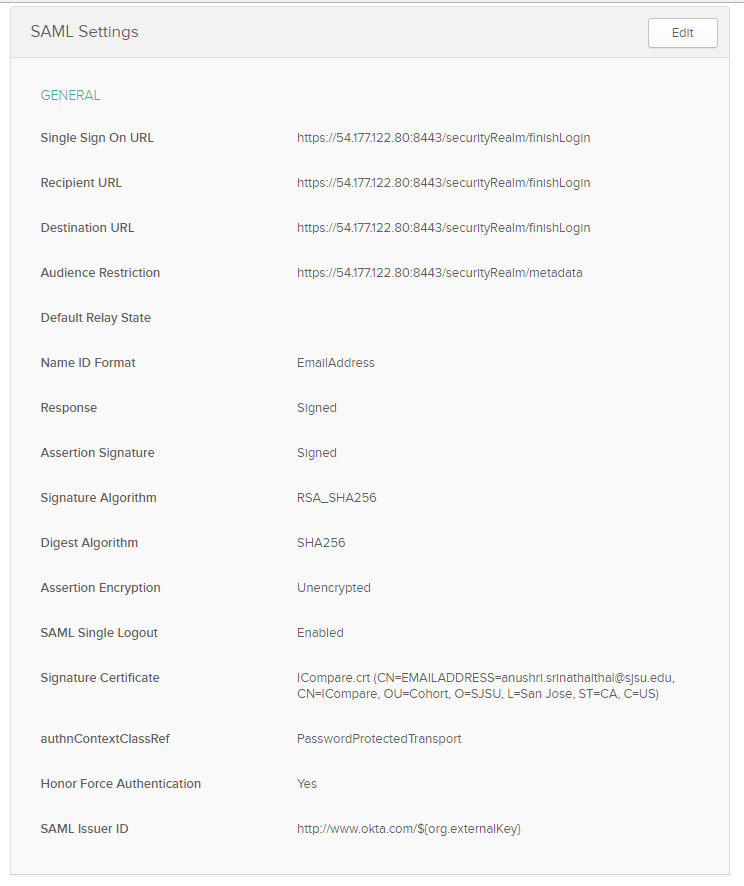
1. Install SAML 2.0 plugin on Jenkins by navigating to plugin management.



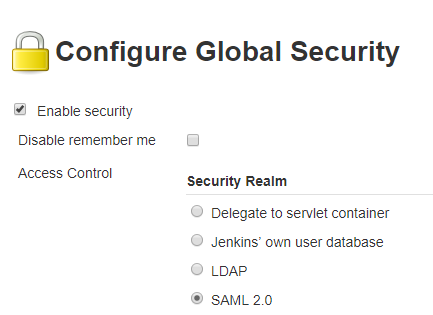
1. Create an SSO SAML app in Okta for jenkins.



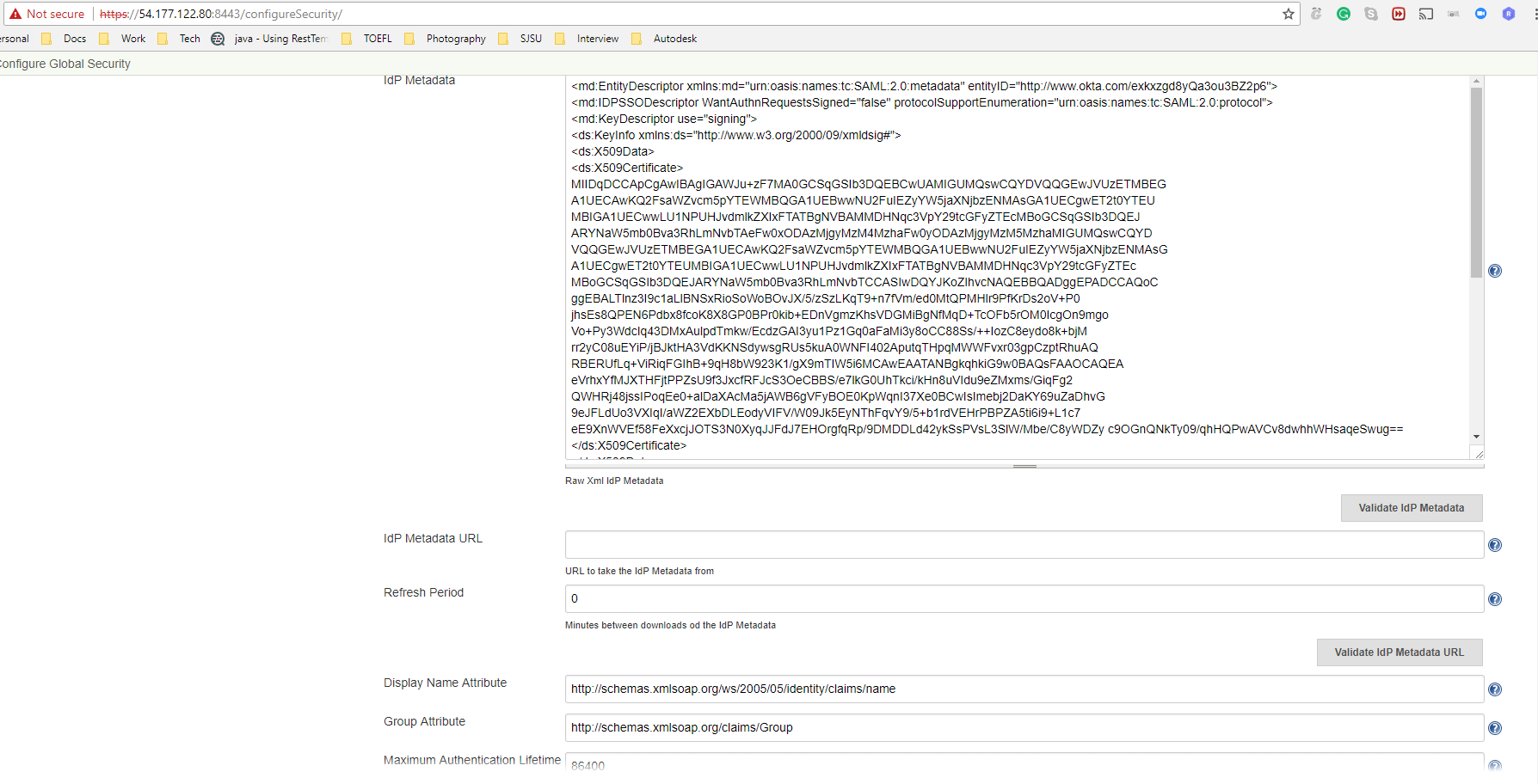
1. Configure the Sign On URL, Audience URI, Logout URL as below



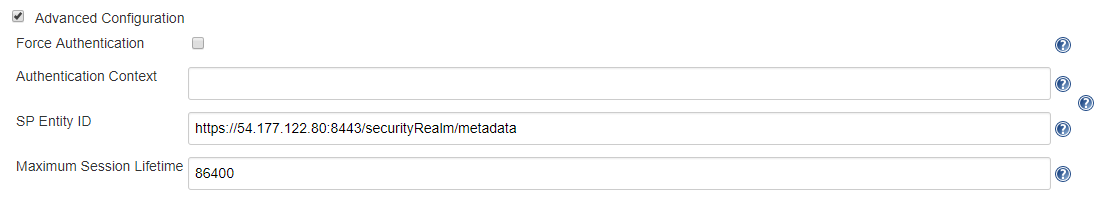
1. Download the SAML IDP certificate from okta and configure SAML security on Jenkins.



1. Navigate to global security, enable security config and choose SAML. Configure the IDP metadata with the certificate downloaded from Okta.



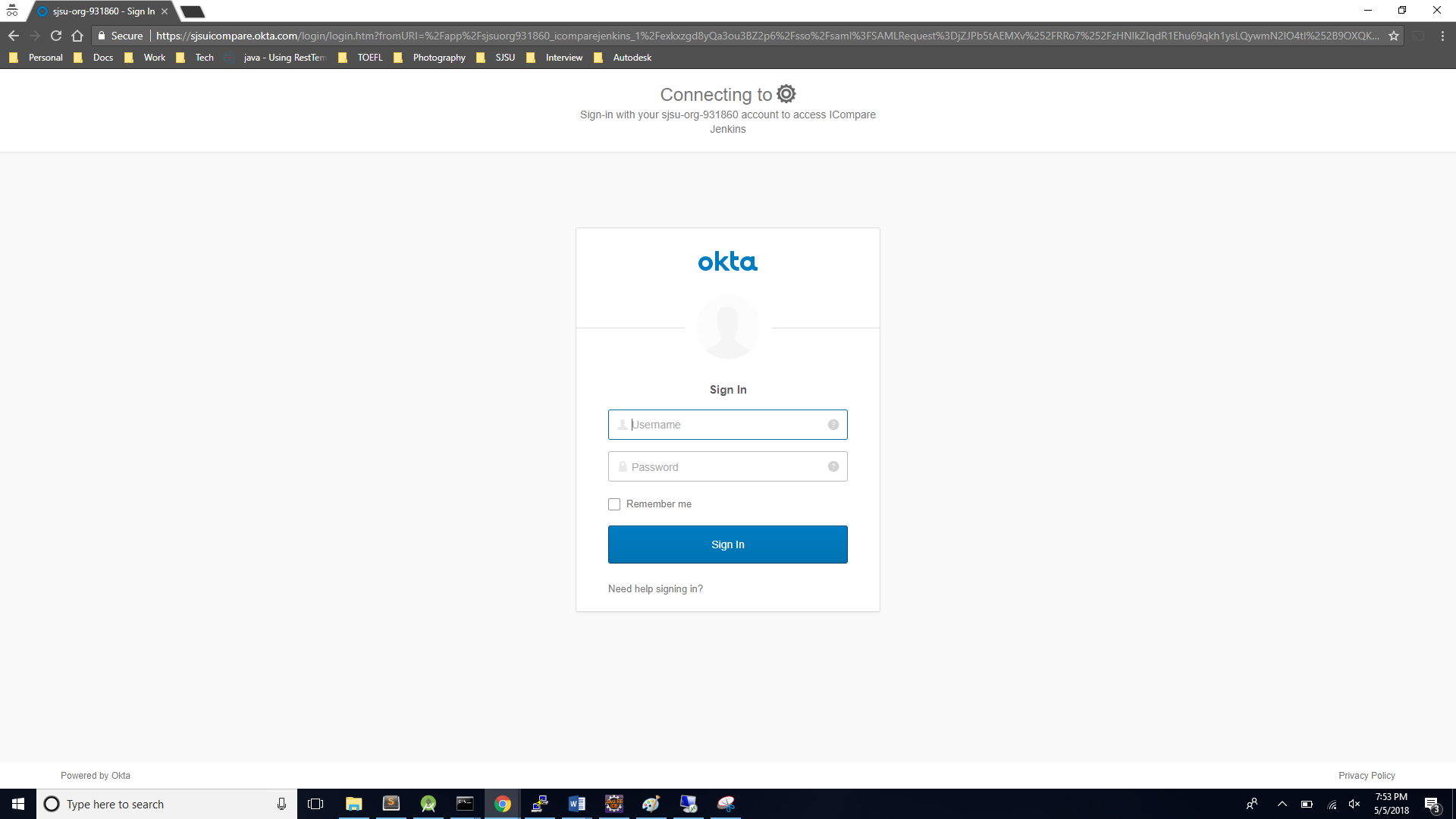
1. Configure advanced settings and SP Entity ID with the Audience URI.



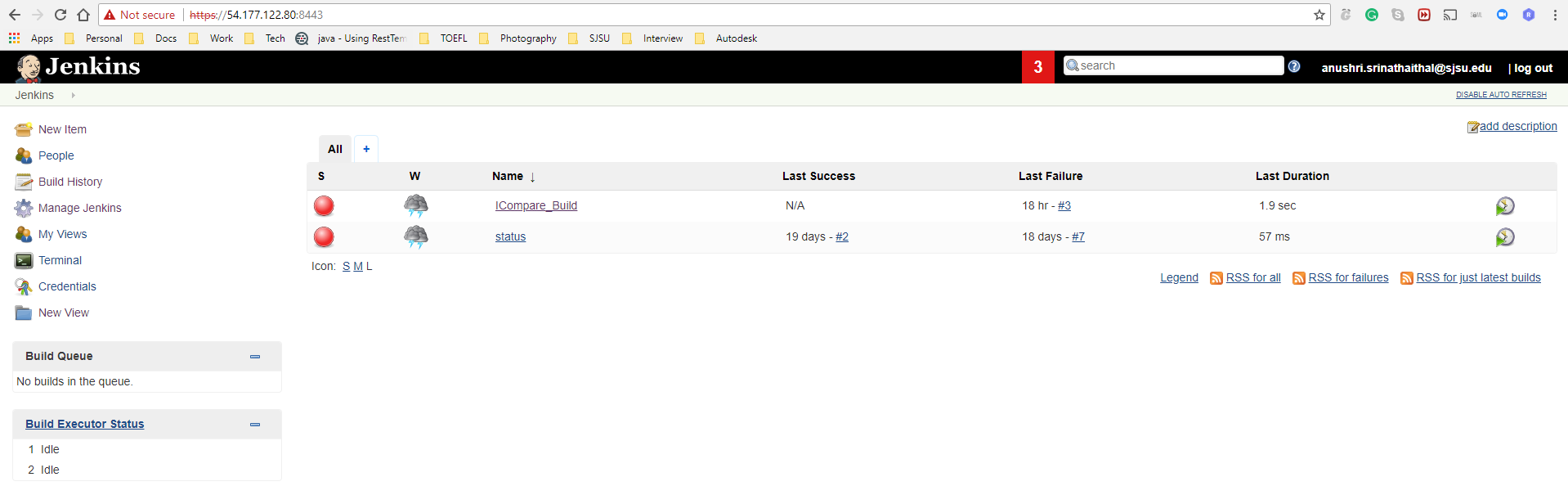
1. Configure logout URL as below



1. Assign users to the app in Okta. Check if SSO is working. Login page should navigate to Okta’s login page.



**Fig. Okta Login Page**

****

**Fig. Jenkins home page after SSO login**

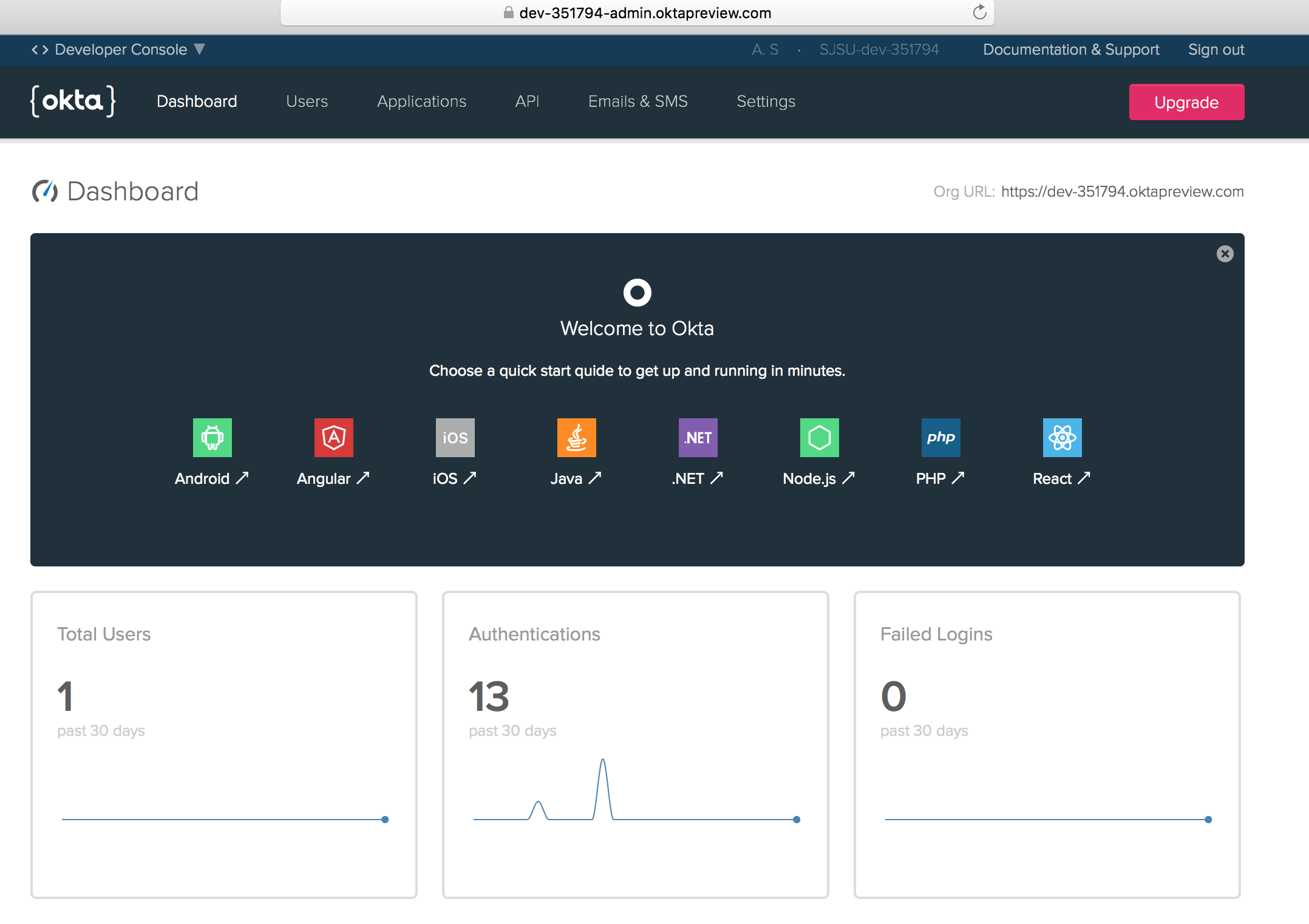
**ICompare Admin portal SSO using okta**

ICompare admin portal provides a platform for the application ‘Administrator’ to update the Semantics3 API keys. This provides an efficient way for the admin to manage the API keys with not have to deal with source code to update the new keys. The Admin portal comes with an added advantage of single-sign-on. The application is built using Spring Boot and the SSO functionality is leveraged using okta via SAML 2.0. Below are the steps followed to achieve this functionality:

SAML application creation on Okta

1. Create an Okta developer account.

2. Once the account is successfully created and activated login to the account to see the okta home page.



3. Go to on developer console in this page that will take you to the classic UI. In this page select ‘Add Applications’ to create a new application.

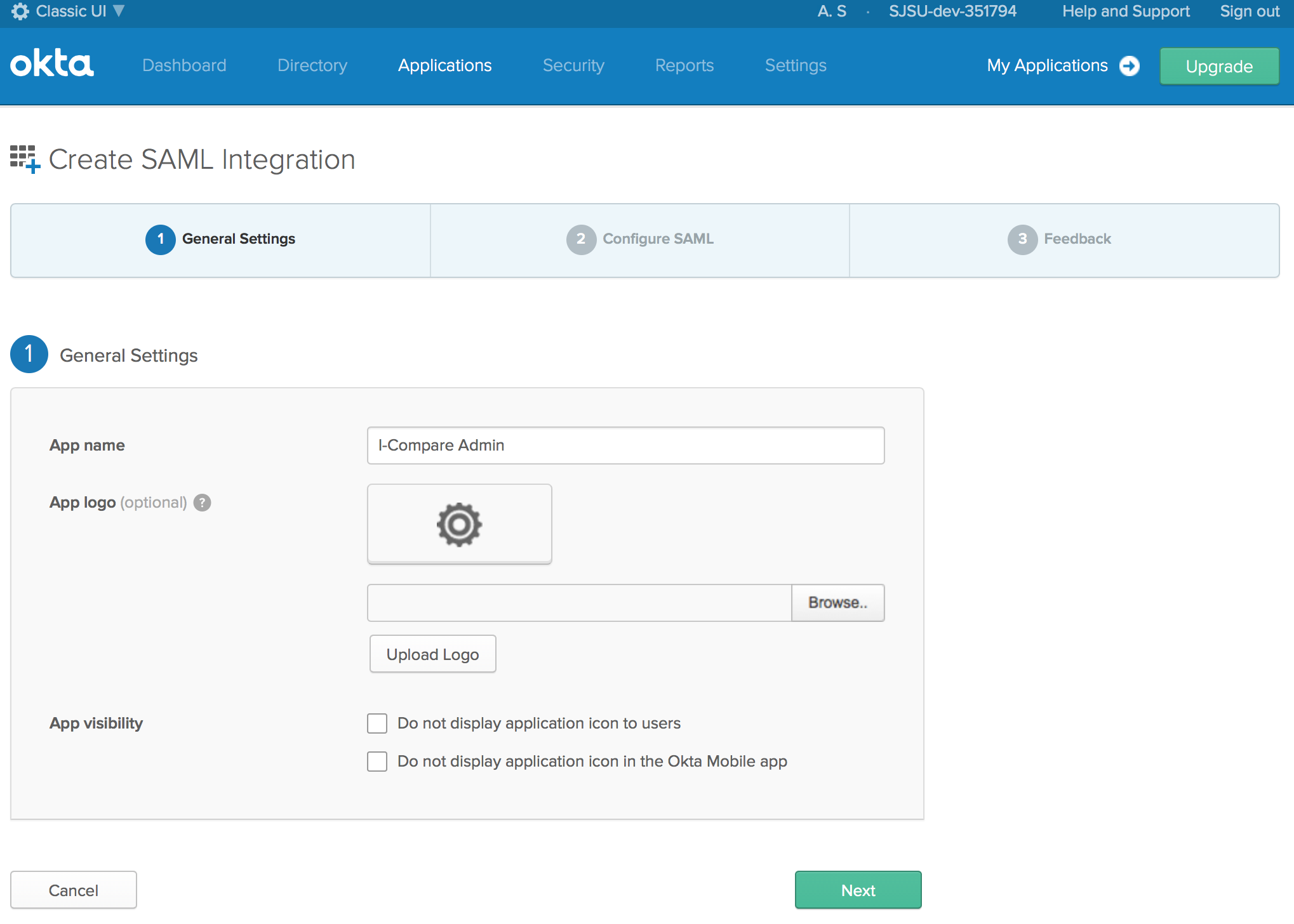


4. In the next page select ‘Create new App’ and choose web and SAML 2.0.

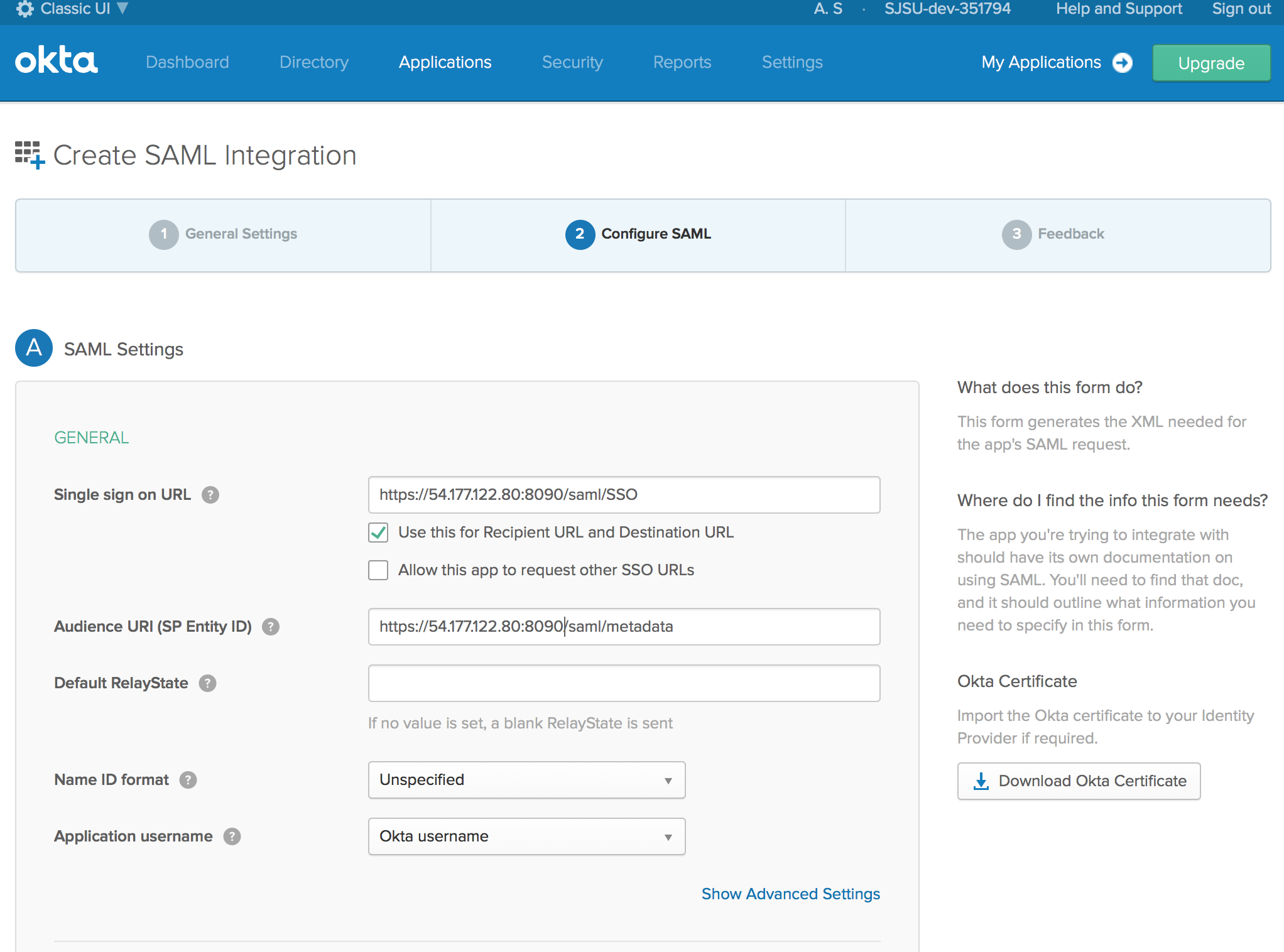
Select ‘create’.



5. In the next screen provide an app name ‘I-Compare Admin’ and click ‘Next’.



6. Configure the SSO URL and Audience URI as shown below and click ‘Next’.

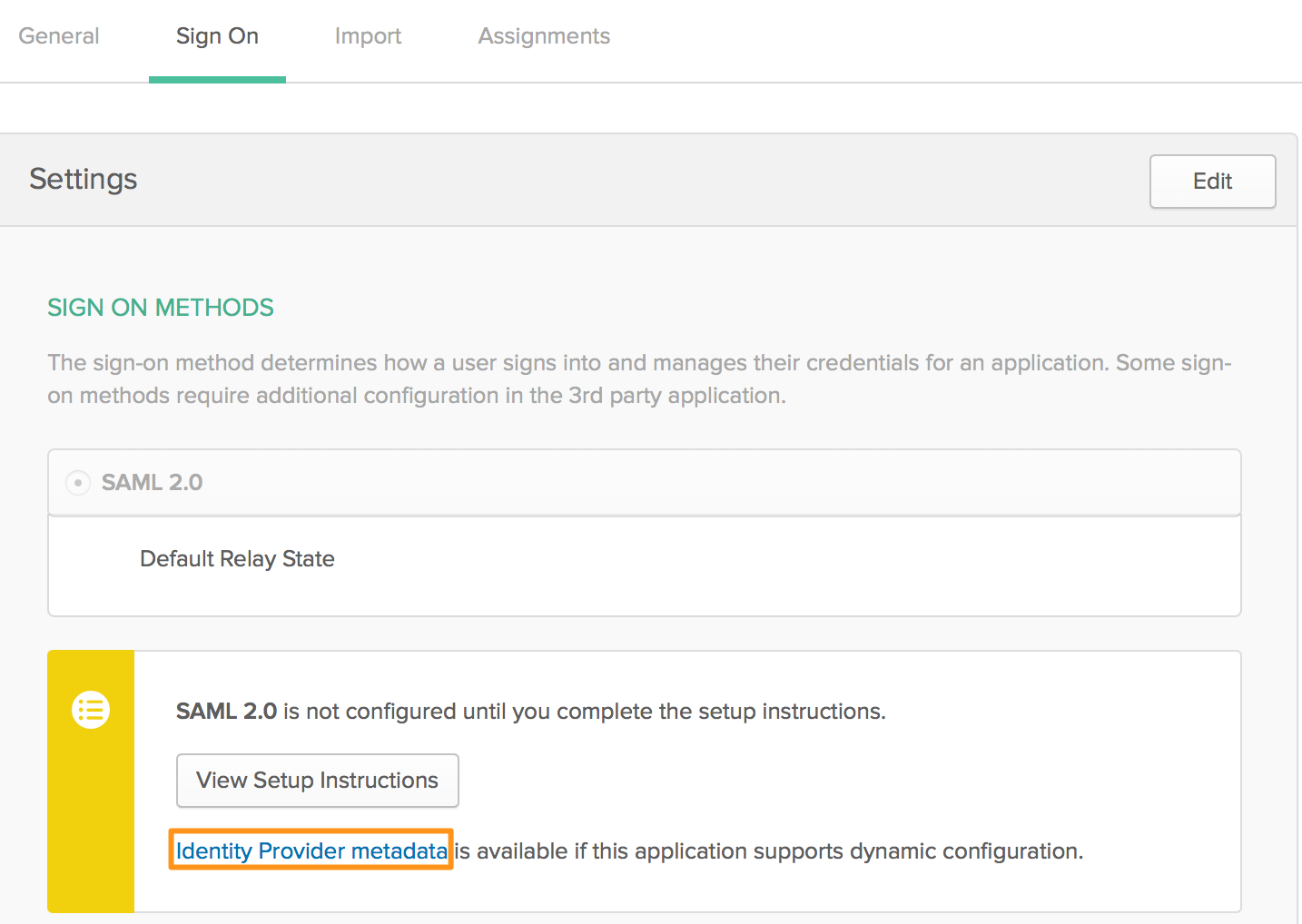


7. In the next page configure as shown below and click on ‘Finish’.

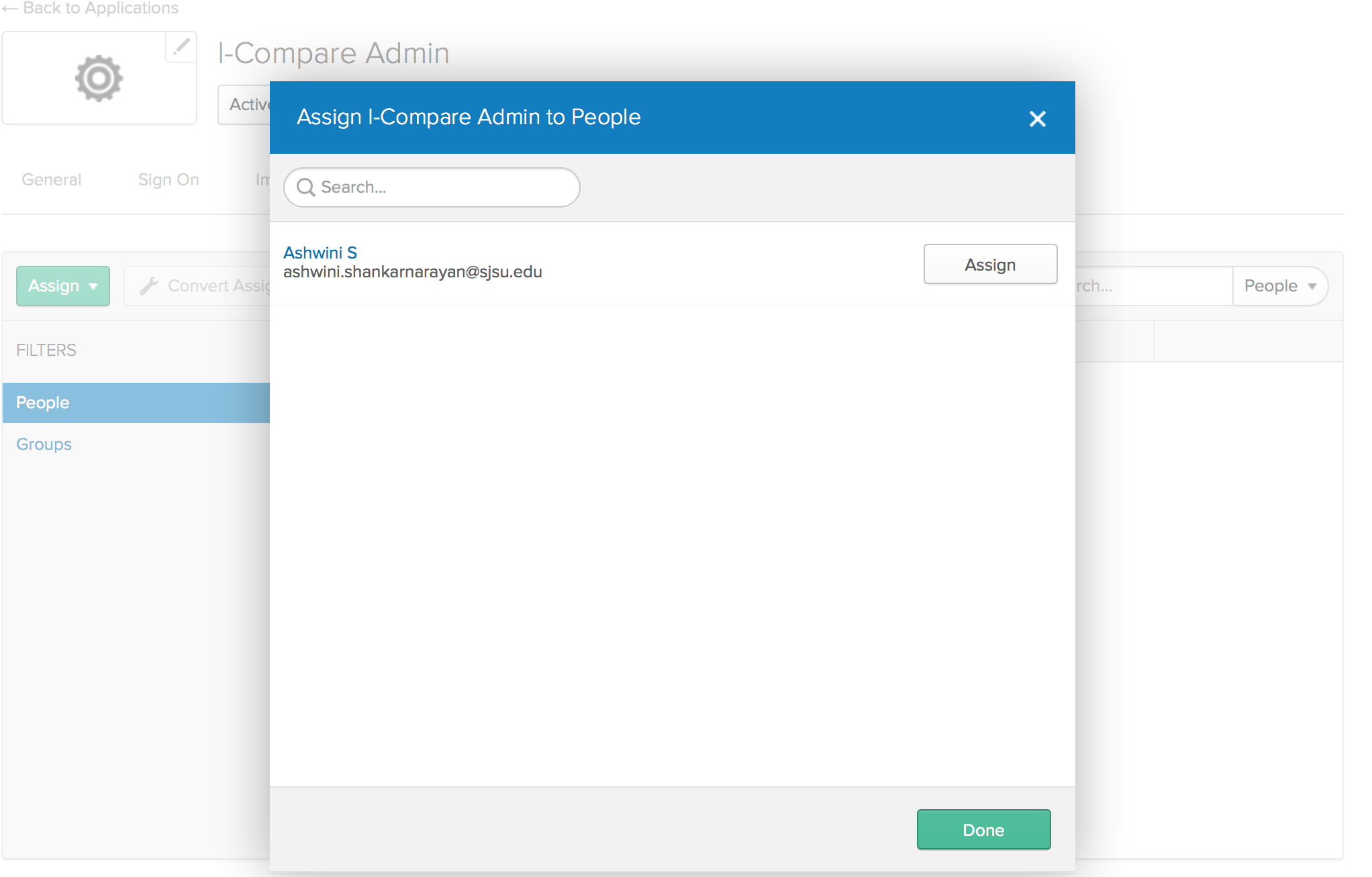


8. On the ‘Sign On’ tab copy the link for Identity Provider Metadata to configure it for

Spring boot application.

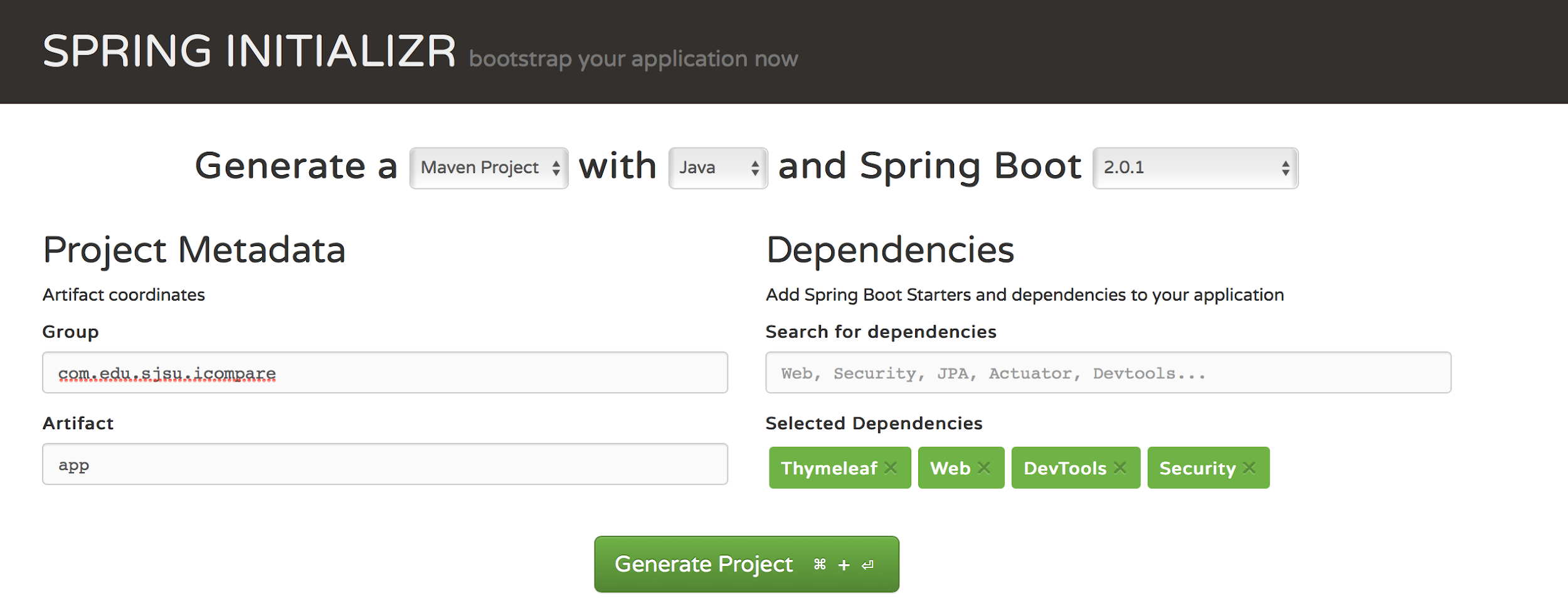


9. In the Assignments tab, click on Assign and then Assign to people and add the users who can access this application.

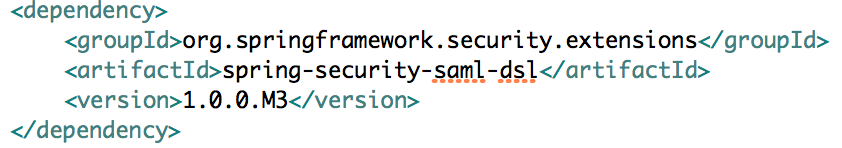


Spring Boot application using SAML

1. Create a Spring boot project using the link [https://start.spring.io](https://start.spring.io/) and include the dependencies for Thymeleaf, DevTools, Security and Web.



2. Once the project is generated, import the file to eclipse and add the ‘spring-security-saml-dsl’ dependency to pom.xml.



3. In the application properties add the below details including the Identity provider metadata URL that we copied previously.

server.port = 8090

server.ssl.enabled = true

server.ssl.key-alias = spring

server.ssl.key-store = classpath:saml/keystore.jks

server.ssl.key-store-password = secret

security.saml2.metadata-url = < Identity provider metadata URL>

4. On terminal navigate to src/main/resources and generate the keystore.jks file using the keytool command.

keytool -genkey -v -keystore keystore.jks -alias spring -keyalg RSA -keysize 2048 -validity 10000

5. Create a class with name ‘ICompareAdminSecurityConfiguration’ to enable the SAML based SSO using http.authorizeRequests and by providing port, passwords, key store, key store alias, hostname and metadata URL (The code is available on github link).

6. Include the HTML code to update API keys in the index.html file located in template folder.

7. Add the logic to update the Mongo DB table when user updates the API keys on front end. This is achieved using the Rest API call to POST.

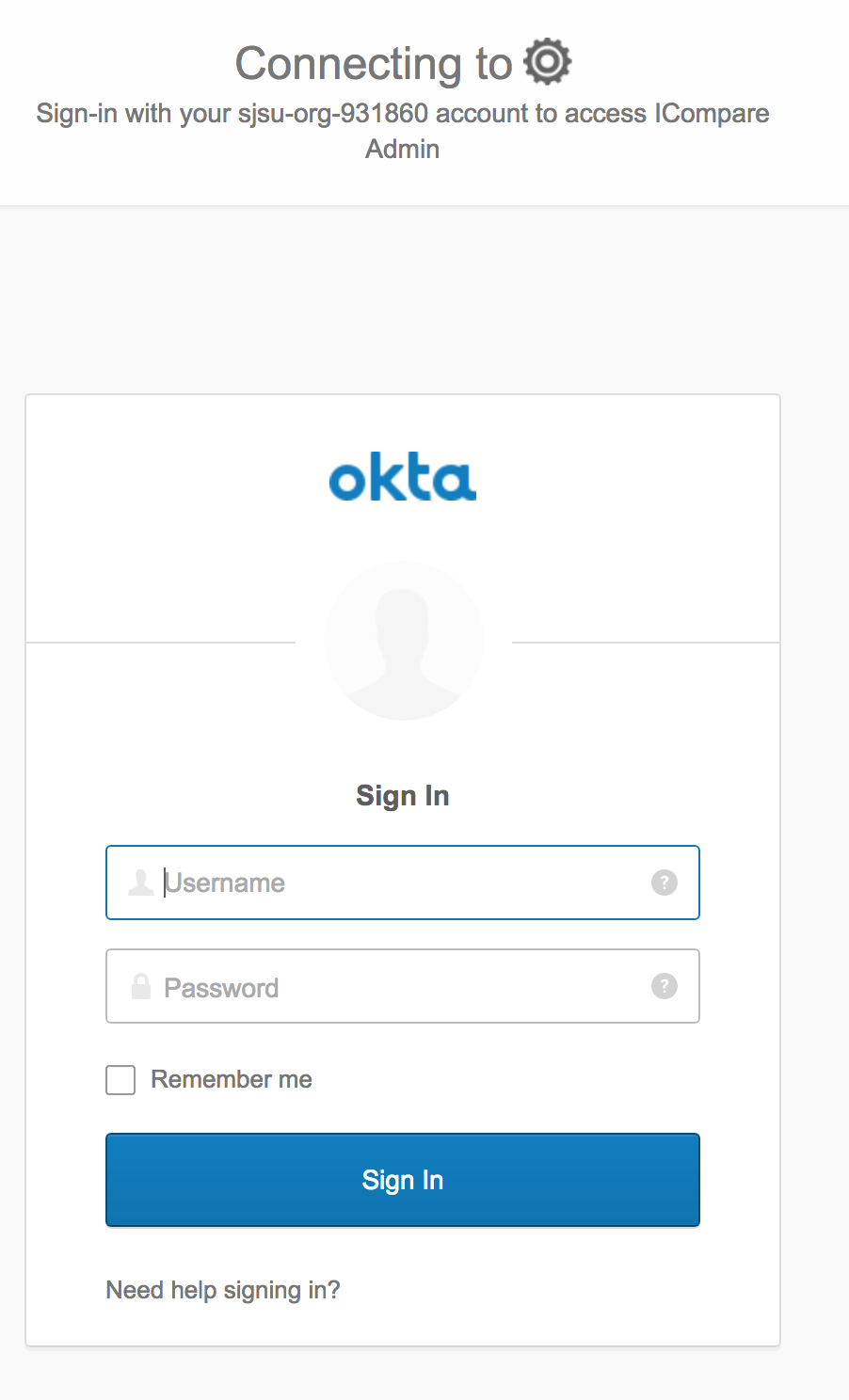


8. Build the admin app to navigate to the Jenkins from the same page.

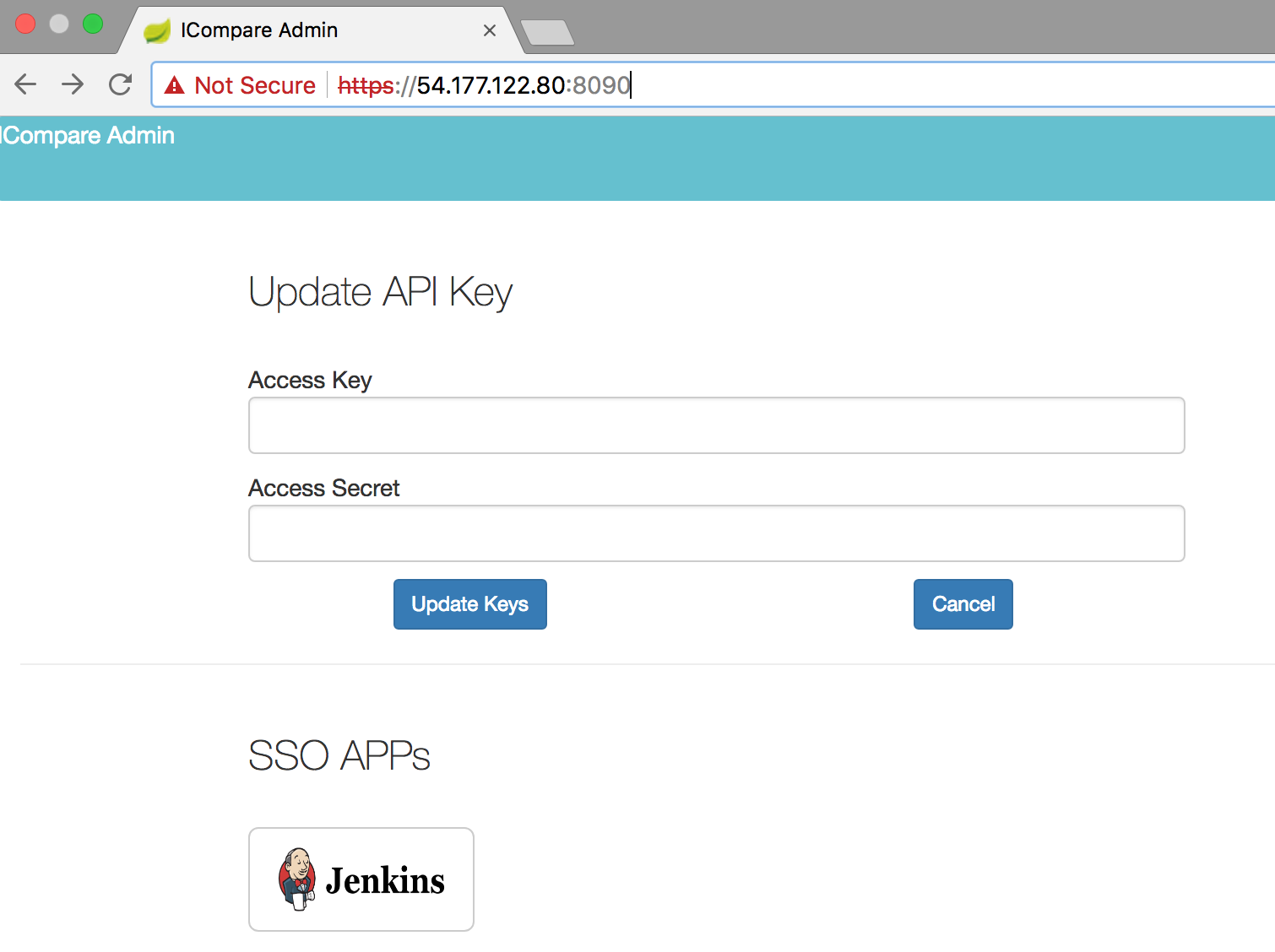
9. Build the source code (Maven build) and copy the jar on to the EC2 instance to make the admin portal application to be available on cloud.

10. In order to start the spring boot application as a service, add it as a service in init.d script.

11. Now on navigating to the URL<https://54.177.122.80:8090/> will redirect you to the okta sign in page.



12. Once the sign in is successful, you are directed to the ‘Admin portal’ of the ICompare application.

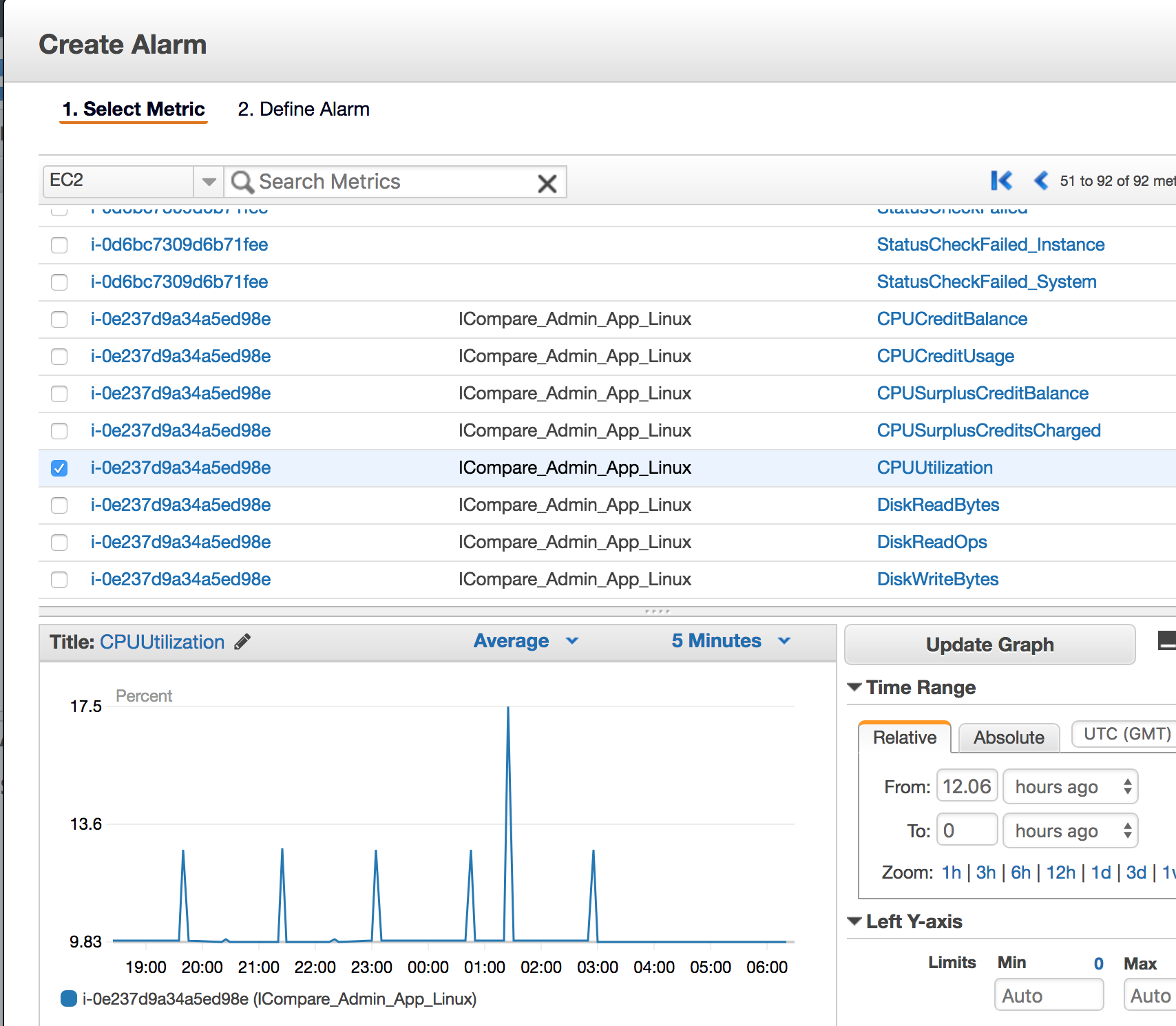


**Fig**. **ICompare Admin Portal login via SSO**

**CPU Usage Alarm for EC2 Instance using CloudWatch and SNS**

CPU usage alarms is set to monitor the health of the EC2 instance where the SSO is configured. If the CPU usage of the instance goes beyond 65 percent and SNS alert is triggered to send an email message to the admin of ICompare application. This will help the admin to take any action if the health of the application goes from OK to Alarm.

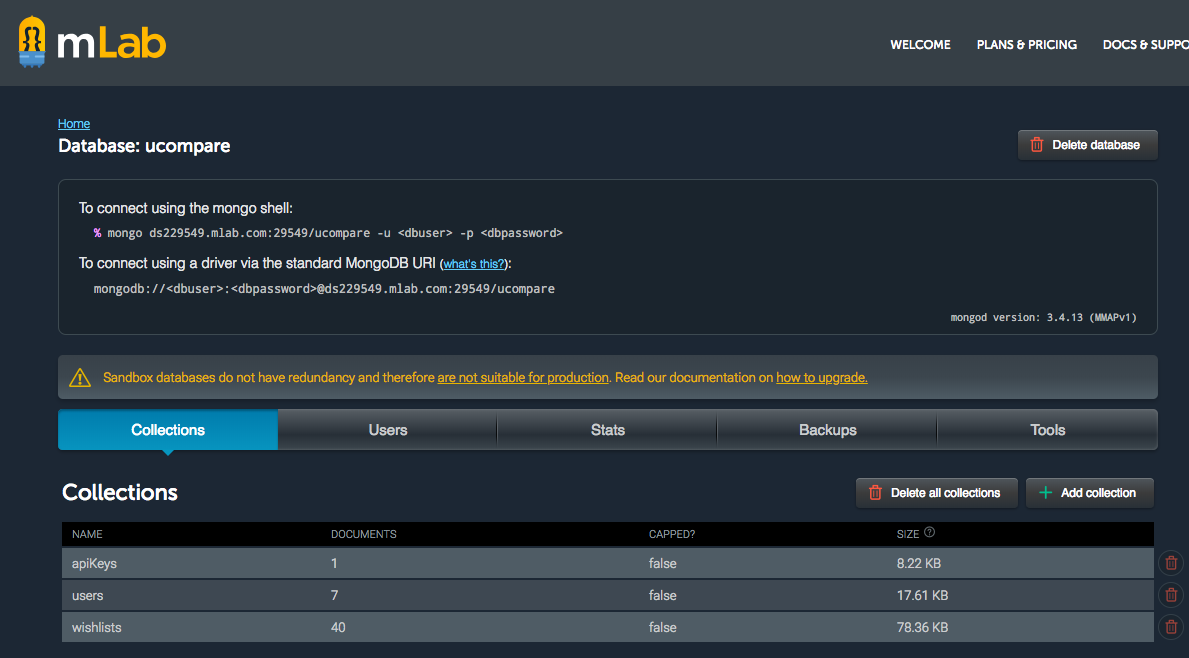
To achieve this, log in to AWS console and choose the EC2 instance and metrics that needs to be monitored. Metrics in our scenario will be ‘CPU utilization’. Configure the alarm threshold to be >65%. Set up the SNS topic with ‘ashwini.icompare@gmail.com’ as the receiver email ID.



**Fig. Configure the CloudWatch alarms and SNS**

**ICompare user portal:**

We have developer user portal in node js, express js and angular. We are using ‘Forever’ npm package to run it on server. We are using cloud based mongoDB server which is provided by Mlabs:



We have developed following APIs:

* GET /search?q=<product> (for search products)
* GET /wishlist <list of wishlist items)
* POST /wishlist/add <add an item in wishlist>
* GET /signin (For signin)
* POST /signup (For signup)
* GET /logout (For logout)

We have developed UI using angularjs and bootstrap template.

## **3.2 Used Technologies and Tools**

**Single Sign-On and Admin Portal**

1. Spring Boot – Custom Application to integrate with Okta and achieve SAML based SSO is developed using Spring boot.
2. AWS Directory Services - To create Microsoft Active Directory.
3. EC2 – Microsoft Server 2016 R2 base flavored EC2 for setting up domain controller. Okta SAML 2.0 – Okta agents and SAML 2.0 support to achieve SSO
4. Spring Security – To achieve
5. SSO.

**ICompare Portal**

**Frontend**

1. AngularJS
2. Bootstrap

**Backend**

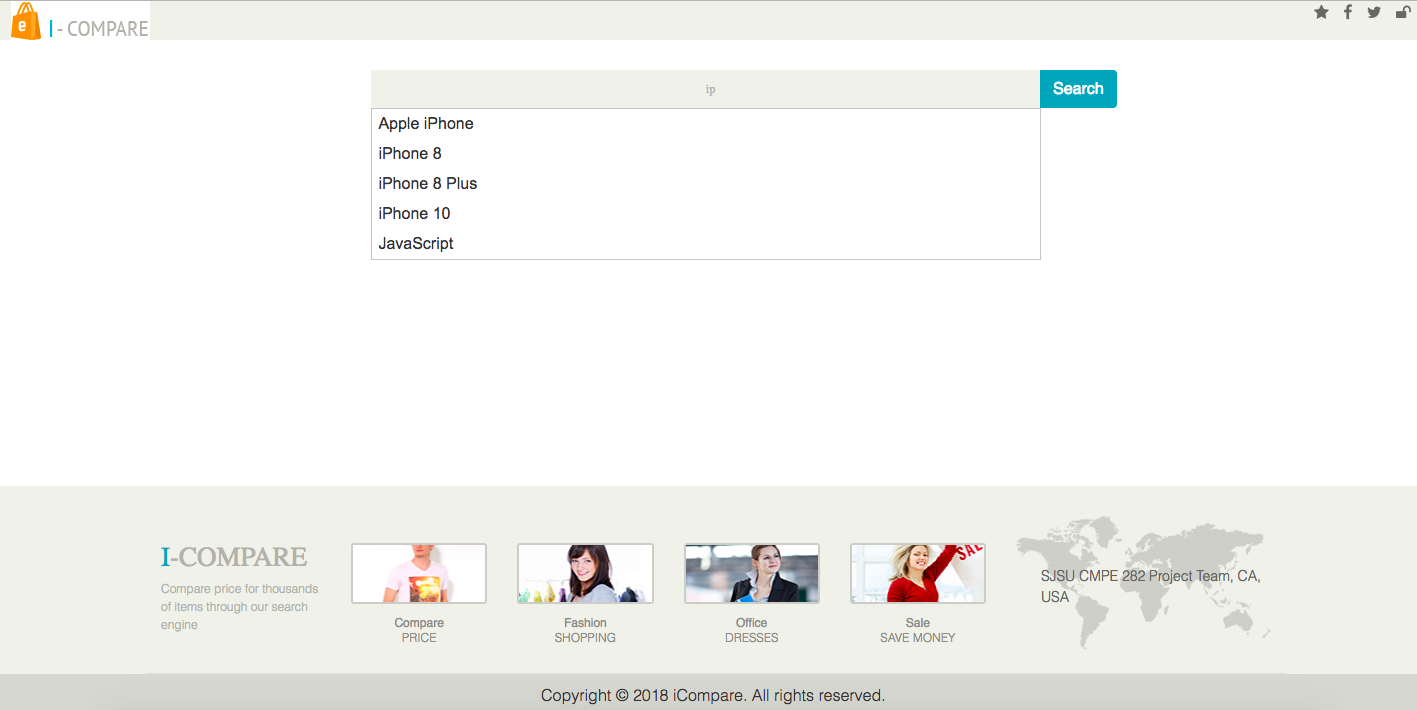
1. Express.js
2. Node.js
3. MongoDB

**Single Sign-On and Admin Portal**

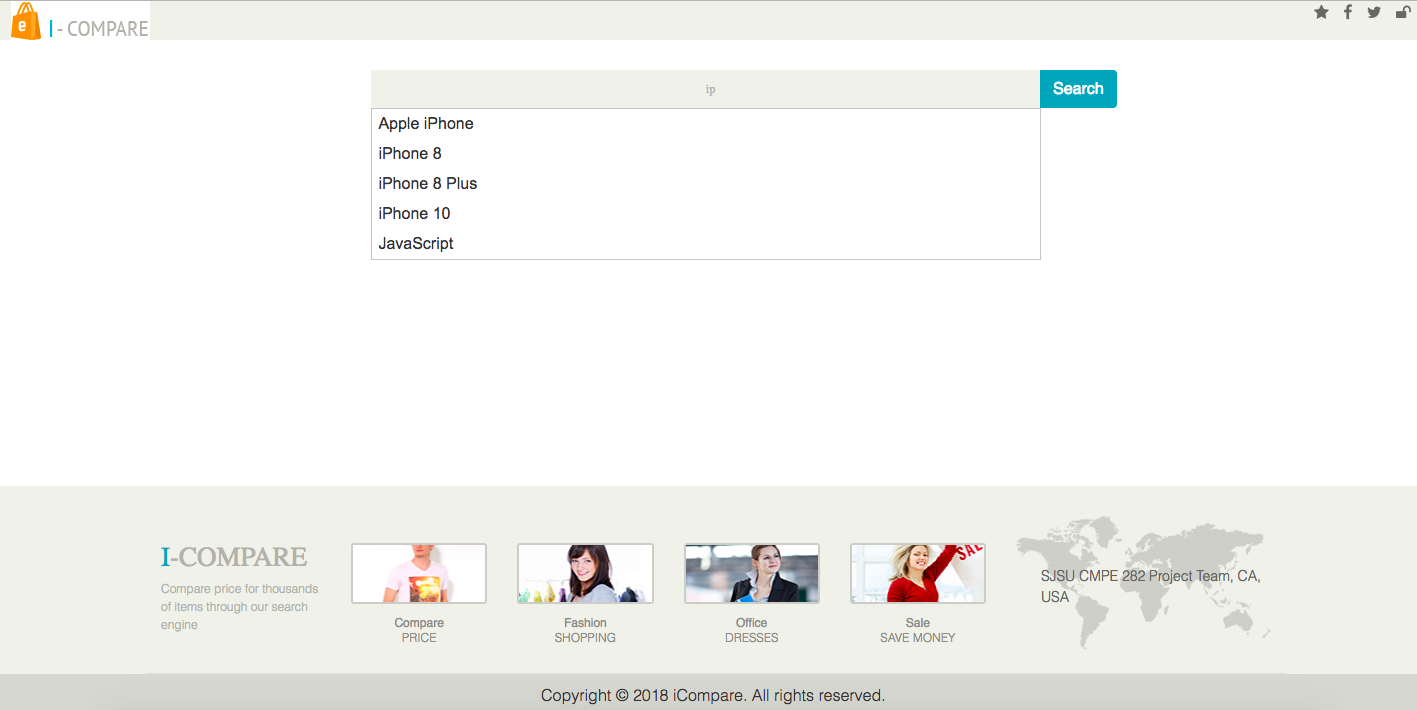
1. jQuery
2. Bootstrap

**3.4** **User Interface**

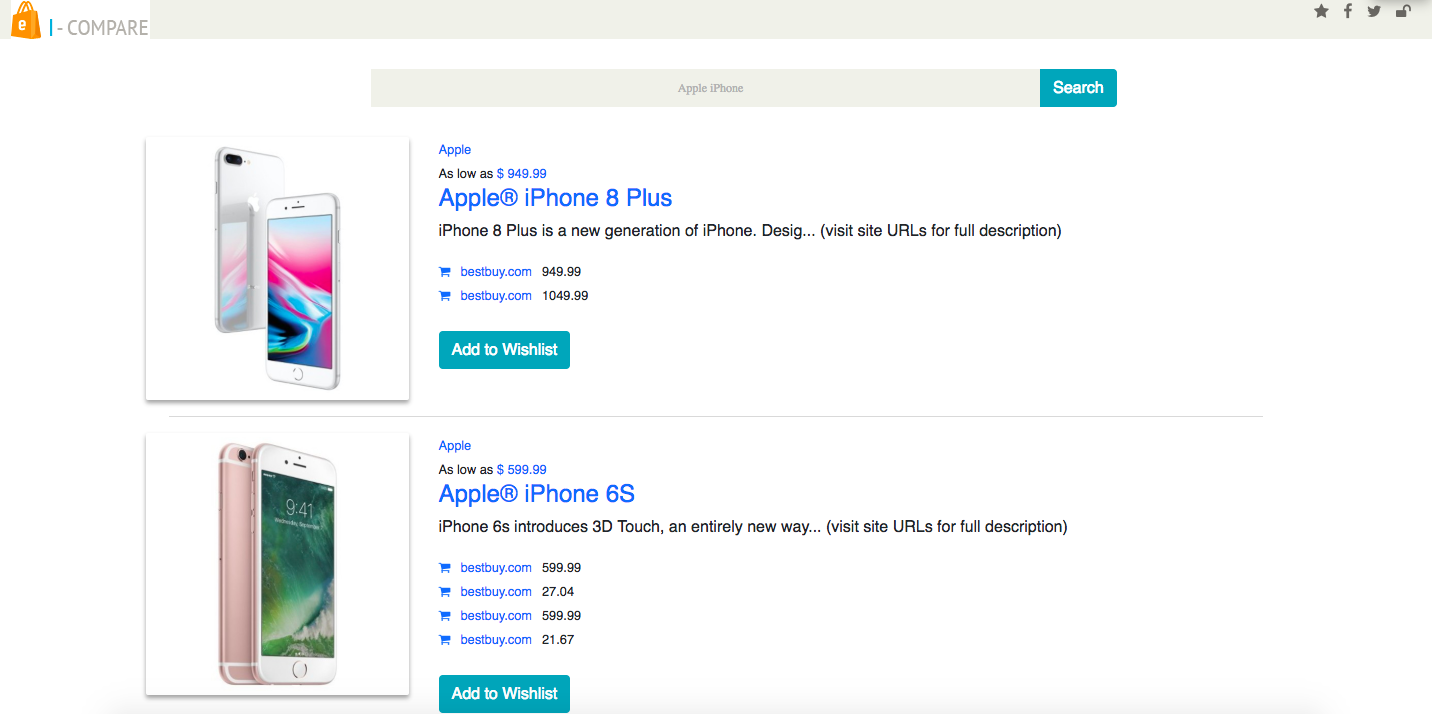
### 3.4.1 Customer Portal (Home page)



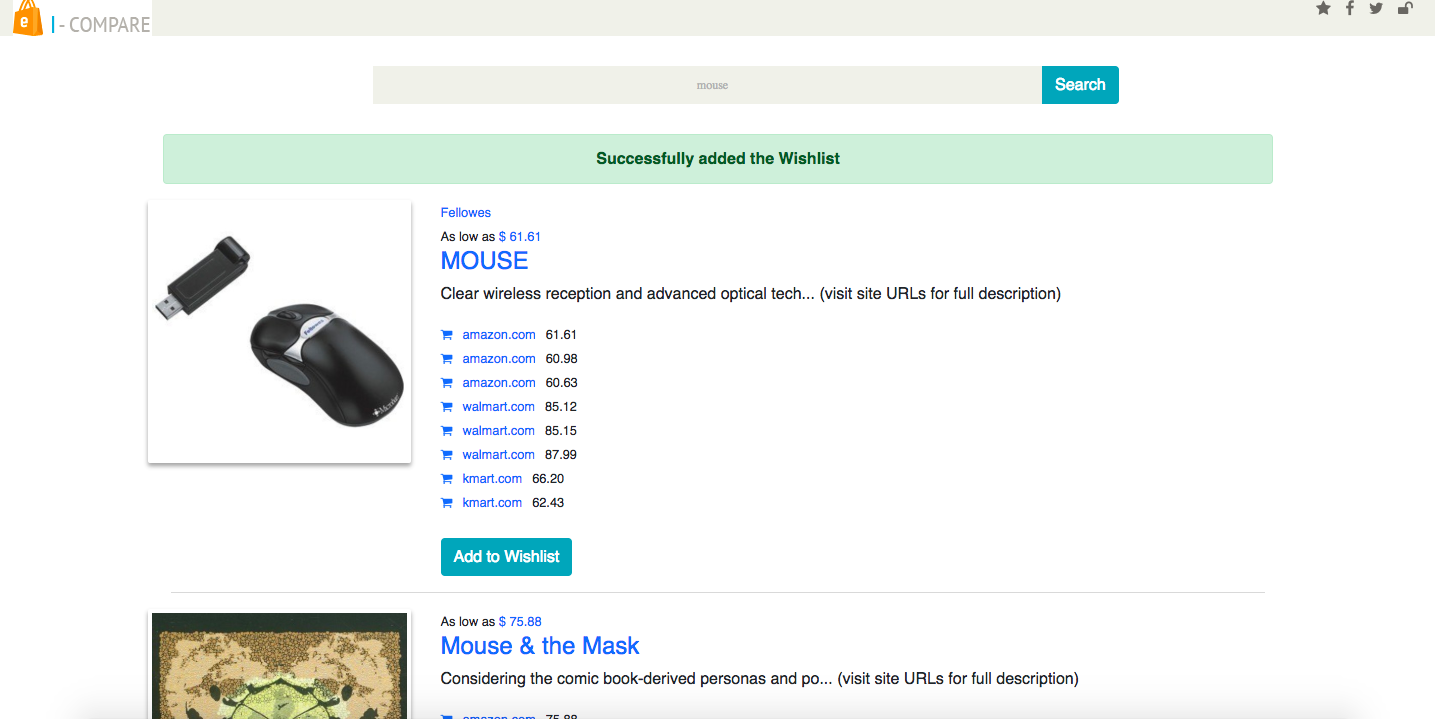
### 3.4.2 Search product for comparison (Auto populate suggestion)

****

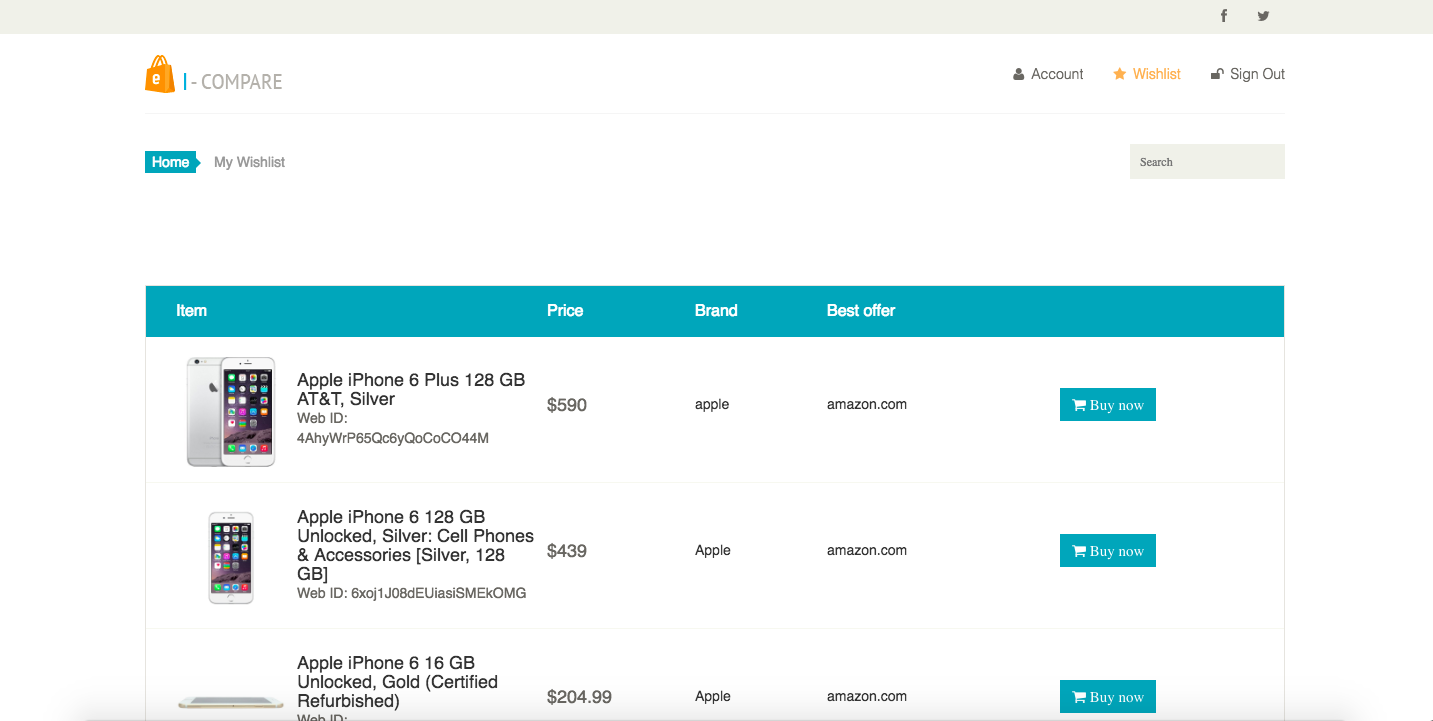
### 3.4.3 Search result page

****

### 4.4.4 Add to Wishlist



### 3.4.5 Wishlist page



**4** **System Testing and Experiment**

4.1. Testing Automation using Mocha and Chai

Mocha and Chai are used for automated testing of the ICompare application. The aim here is to test the outputs and behaviors of API routes in Node.js project. The application uses REST API routes to navigate between various user screens, GET/POST data to Mongo database and to invoke Semantics3 API. We have used Mocha and Chai to write tests against these APIs.

Mocha allows its users to perform asynchronous testing in JavaScript framework for Node.js. It provides a platform to use any assertion library of your choice to test the code. In our scenario, we have used Chai as our assertion library.

To setting up test platform, follow the below steps:

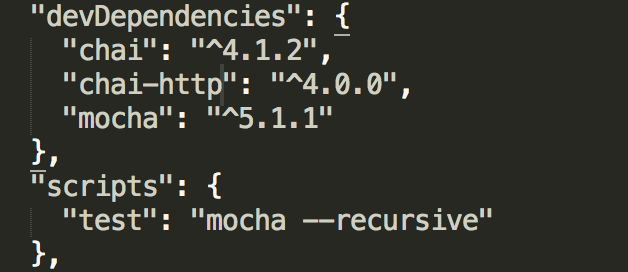
1. Install mocha and chai.

* Mocha: The main test runner
* Chai: The assertion library

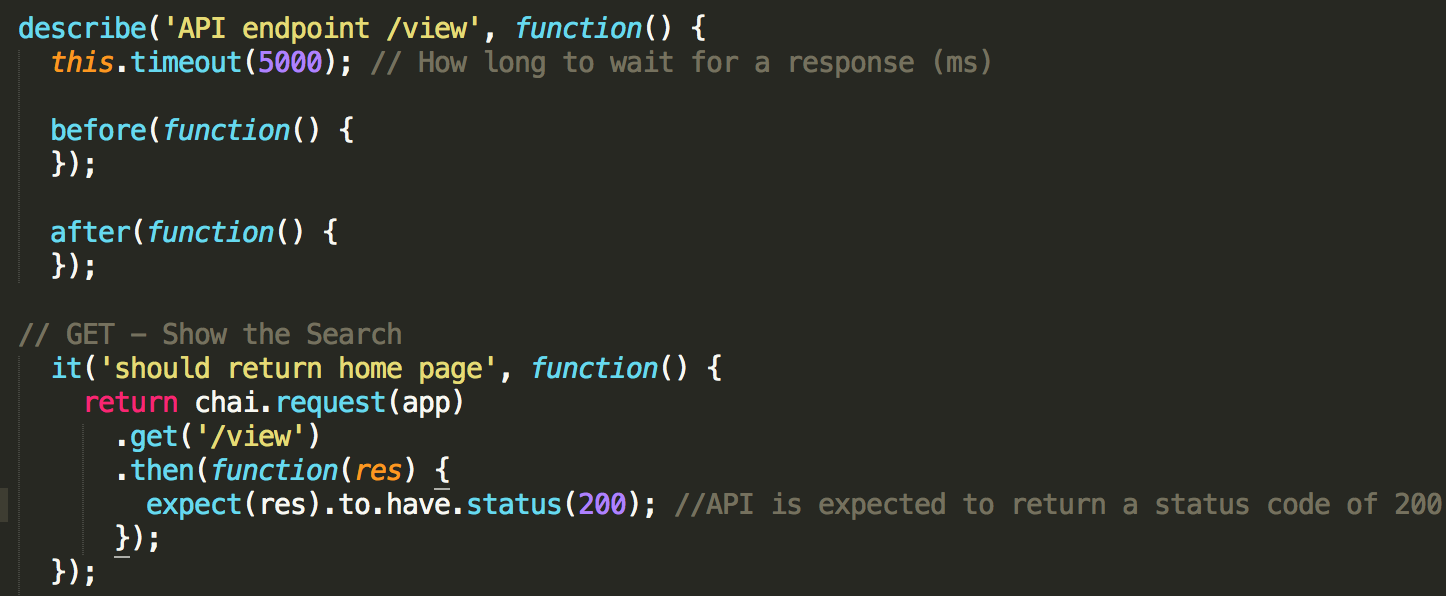
Execute the below command to install mocha and chai:

npm install mocha chai supertest--save – dev

2. The following dependencies are added to the package.json.



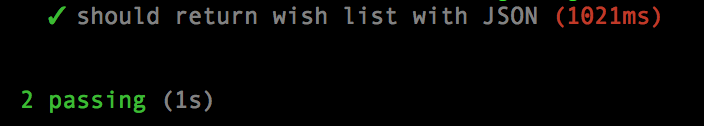
3. Create a test directory in the project and create a file ‘test.js’ to write some tests for our API. We include the ‘describe’ method to describe our test case inside which we write each of our tests using the key word ‘it’ (as shown below).



4. We can now run all the tests using the below command:

npm test

5. We can now see the output of our test cases.



**4** **Conclusion and Future Work**

Demo videos

ICompare Customer Portal: <https://youtu.be/41FNnWSZSik>

Admin Portal: <https://youtu.be/Vz9yebzo_js>

GitHub Repo URL

<https://github.com/amitr1983/ucompare>

Summary

We have developed ICompare which is a Cloud based price comparison web application. User can search for product and compare the price from different online merchant. If it matches their preference, then we can redirect them to merchant’s site, so they can buy the product directly.

To provide seamless experience, we have deployed our application on Cloud (AWS EC2 & beanstalk). We have added functionalities like single sign on, social login using AWS Cognito and have created admin interface to rotate the API keys.

We have followed agile methodology and we were meeting on weekly basis and were following the scrum strictly. We were using ‘Trello’ for project management.

For deployment, we are following continuous integration and deployment process. We are using Jenkins as CI/CD tool.

Future work

ICompare has very good future development prospects. As more and more users are shopping online instead of going to physicals store, price comparison tools, application can have huge influence over customers traffic. Our application can be extended further, and we can add following features:

1. We can implement filters, so customers can filter the results very on their needs. Few examples of filters are:
   1. Price range
   2. Preferred online merchant
   3. Product location
   4. Shipping preference
   5. Clearance/Sale
   6. New/Used products
2. We can upgrade to more robust and responsive UI to support both web and mobile platforms.