AWS and Okta Integration Guide

Okta Inc.
301 Brannan Street
San Francisco, CA 94107

Eric Karlinsky
Raphael Londner

Introduction

About AWS IAM Roles

Amazon Web Services (AWS) governs access to resources in a different way than most web applications, so Okta's integration with AWS is unique. First, let's take a look at a typical web application integration, based on SAML. In such an application, a user account exists in the application and a corresponding user account exists in Okta. Okta generates a SAML assertion which maps the Okta user identifier to the user identifier in the application, and that assertion is used to log into the application. From that point forward, authorization is dependent on the permissions assigned to the user account in the application.

This is a classic approach and it works, but there are downsides to it. For example, it's common for end users to have multiple roles, or sets of permissions, which allow them to perform different duties. This is often the case for administrators. Changing permissions on-the-fly, whenever a user logs in is untenable with the classic model, so this usually results in either a) a single end user having multiple accounts in the application or b) a single end user having one super account with more permissions than are strictly necessary. This becomes difficult to manage, and you lose accountability of which person is performing which actions in the application.

Now let's explore AWS's access management approach, which addresses this conundrum. In AWS, an identity administrator can set up Identity and Access Management (IAM) Roles, which contain a set of permissions on the AWS platform. An end user who logs into AWS via an Identity Provider (or "IdP") with SAML can then assume one of these IAM Roles at a time for the duration of an authenticated session. The IdP populates the SAML assertion with a list of IAM Roles that the end user can assume, and AWS will prompt the end user at login to choose which IAM Role should be used for the current session. The net result is that administrators regain accountability for actions in AWS because each session is bound to both an IAM Role and a user identifier. Also, least privilege principles are followed, since a user only logs in with the permissions that are required to perform the current task.

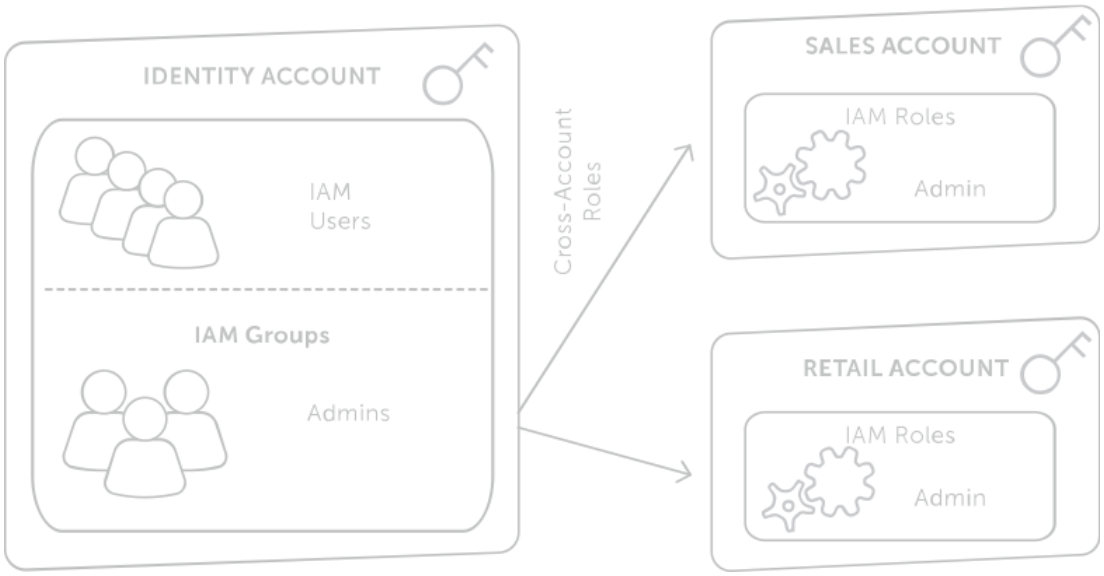
About AWS cross-account access

AWS also provides the ability to create cross-account IAM Roles, which are extremely useful when a user must access multiple AWS accounts with the same identity and fine-grained permissions. Amazon recently documented prescriptive guidance on AWS Multiple Account Security Strategy, which we highly recommend as pre-requisite reading.

Okta is compliant with Amazon's cross-account security guidelines and an important goal of this document is to provide a practical tutorial of the configuration actions required to implement the "Identity Account Structure" highlighted in Amazon's Multiple Account Security Strategy white paper. As a reminder, the Identity Account Structure, *"is ideal for organizations who want to create and manage all their users in a single account and enable user and group access to resources in other accounts. This model uses IAM cross-account roles to grant access control from one account to another. IAM roles grant temporary access to an AWS account based on the role's IAM policy and trust relationships. IAM cross-account roles establish a trust relationship between AWS accounts, granting predetermined users, groups, or roles in one AWS account permission to perform specific API actions in another AWS account. For example, to establish an identity account structure between IAM users in a parent identity account and other BU*

accounts, grant cross-account roles to users or groups in the parent account that allow them to manage AWS resources in the associated accounts." [1]

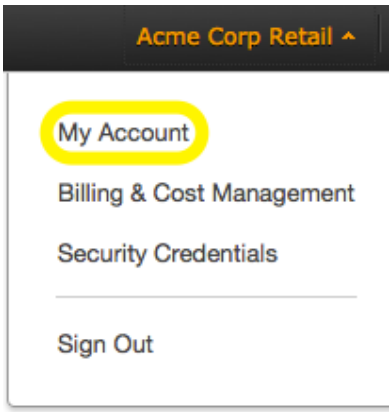
The graphic below (inspired by Amazon's whitepaper) best exemplifies the Identity Account Structure and forms the baseline we will use for this walkthrough.



Amazon Web Services configuration

Initial AWS configuration

1. In order to replicate the Identity Account Structure above, please navigate to https://aws.amazon.com/ and click on "Sign In to the Console".
2. Create 3 distinct AWS accounts (by signing out after each account creation) and make note of their account IDs by selecting **My Account** for each account:



[1] Source: AWS Multiple Account Security Strategy white paper

We've chosen to map their names and email addresses to the names in the Identity Account Structure screenshot:

Account Name	Email Address	Account Id
Acme Corp Identity	identity@acmecorp.me	671250594556
Acme Corp Sales	sales@acmecorp.me	881102293469
Acme Corp Retail	retail@acmecorp.me	253541269580

Moving forward, we'll refer to the accounts above for readability purposes.

Furthermore, we will illustrate the scenario above with 2 Okta users, John (john@acmecorp.me) and Jane (jane@acmecorp.me). John will be assigned to cross-account roles that make him EC2 administrator on both the Sales and Retail accounts, while Jane will be assigned to one cross-account role that makes her S3 administrator on the Retail account only.

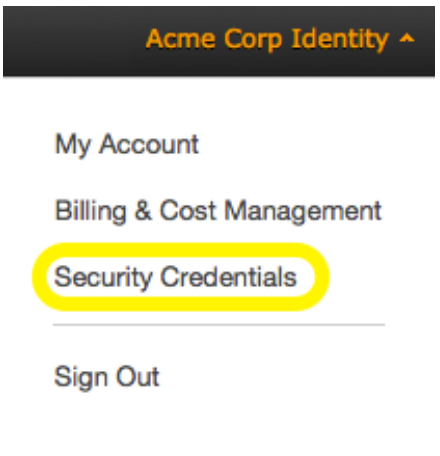
Generating the API Access Key for Okta

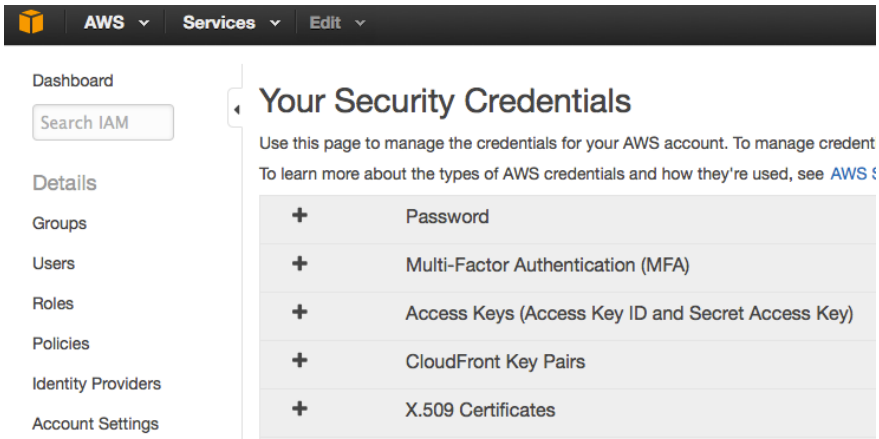
This section will walk through the integration of AWS with Okta. Okta's integration with AWS allows end users to authenticate to AWS using single sign-on with SAML. In addition, when logging in to AWS, end users can select a role to assume, which defines their permissions for the duration of that authenticated session. Administrators can then assign the application instance to individual users or groups of users depending on business requirements.

Okta requires an SSO User Account in AWS IAM, which grants Okta the access it needs to authenticate end users. Okta does not directly use this IAM account, but it does use the API access key and secret associated with this user (so the name of this IAM user is not critical). To create this user, follow the steps below:

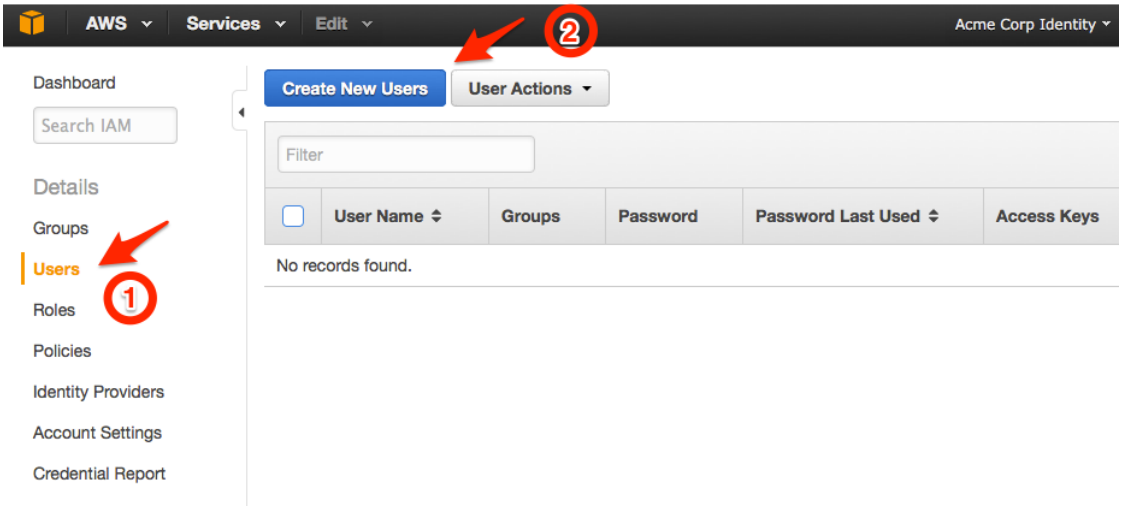
1. Sign in to the **Acme Corp Identity** account at https://console.aws.amazon.com. Pull up the Name menu at the top and select **Security Credentials**. This will redirect you to the AWS IAM Management Console home page.

1.

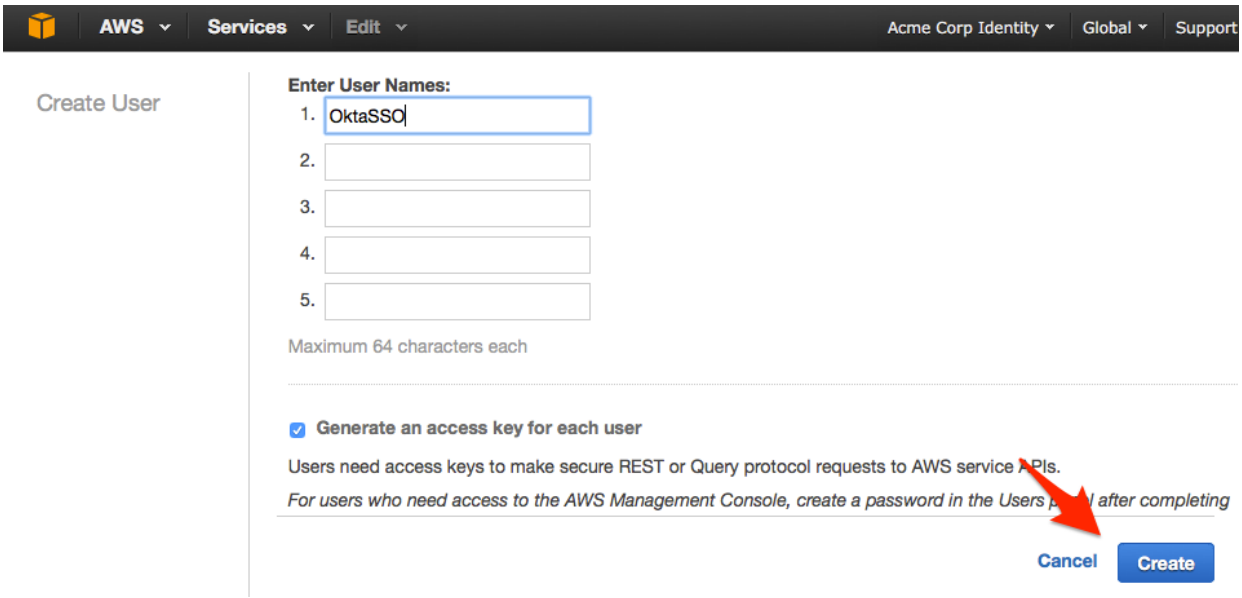




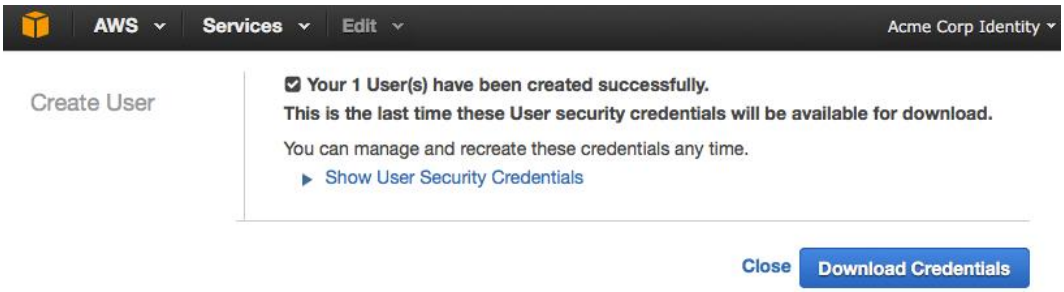
2. Click on Users, then Create New Users:



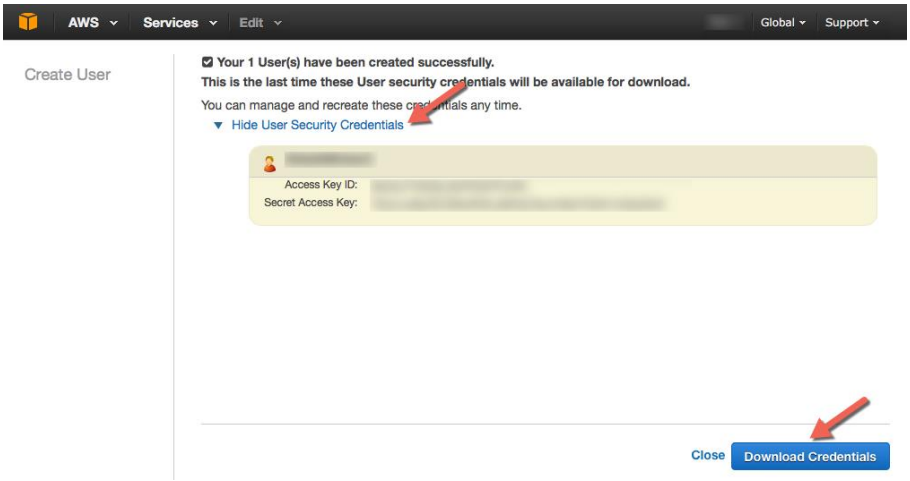
3. Enter a name for this SSO User (such as **OktaSSO**), make sure the "Generate an access key for each user" box is checked and click **Create**.



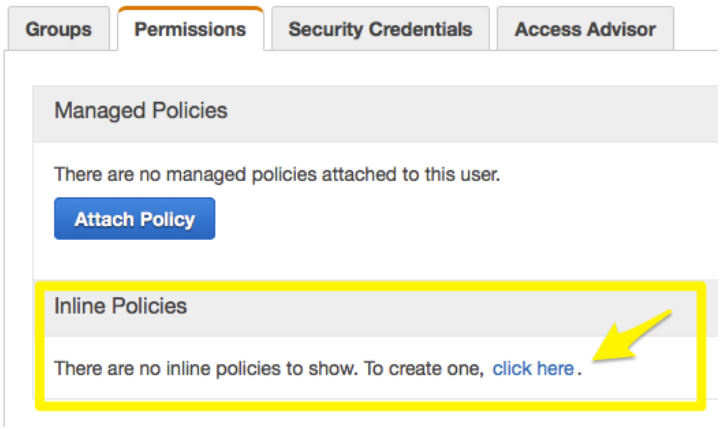
4. The following screen appears.



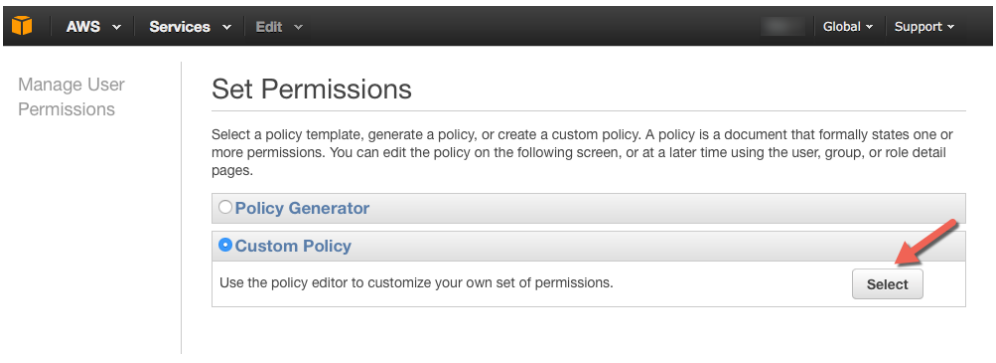
5. Either select **Show User Security Credentials** or click **Download Credentials** in order to save the SSO User's credentials. You will need them to configure the Okta integration in the next section.



6. Now assign the SSO user account the proper permissions. The Okta SSO user has to be able to 1) list all the AWS roles (standard or cross-account) that Okta users may be assigned to and 2) assume any role in the account, so that end users leveraging this integration can log in with the permissions they need.
7. Back in the Users page, select the Okta SSO User you just created. Select the **Permissions** tab. Under Inline **Policies**, select the "**click here**" link, as shows in the screenshot below.



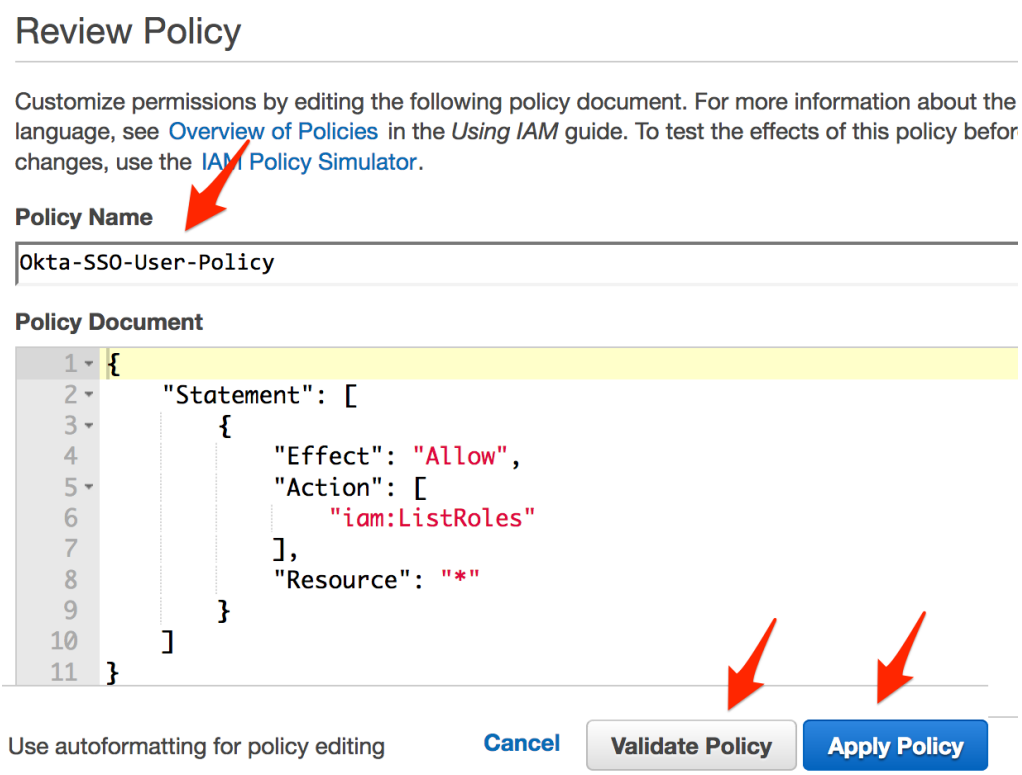
8. In the opening screen, select **Custom Policy**. Click **Select**.



9. Set the **Policy Name** field to a value of your choice (such as Okta-SSO-User-Policy) and paste the following policy definition into the Policy Document editor:

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

10. Click **Validate Policy**, then **Apply Policy**.



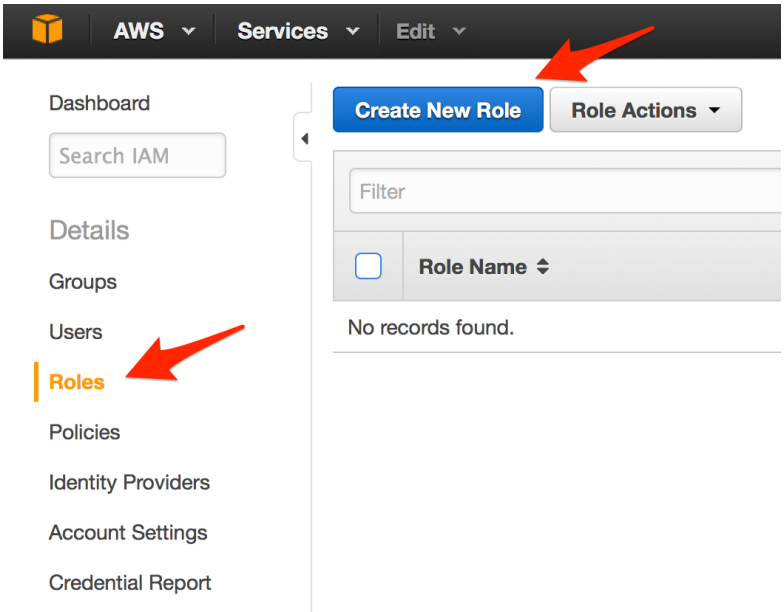
Setting up cross-account roles

As mentioned above, we will illustrate the Identity Account Structure scenario with 2 named users, John (john@acmecorp.me) and Jane (jane@acmecorp.me). John will be assigned to cross-account roles that make him EC2 administrator on both the Sales and Retail account, while Jane will be assigned to one cross-account role that makes her S3 administrator on the Retail account only.

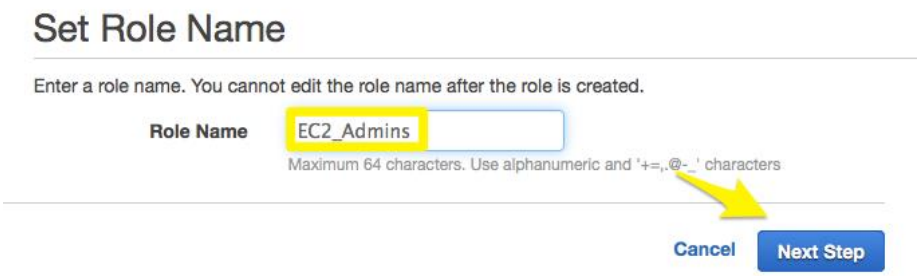
Below is a simplified version of the AWS Multiple Account Security Strategy, which we advise you to review if you need to assign more granular permissions to your users.

Important note: you will need the accounts IDs of your own Sales and Retail accounts (we use those in the table above)

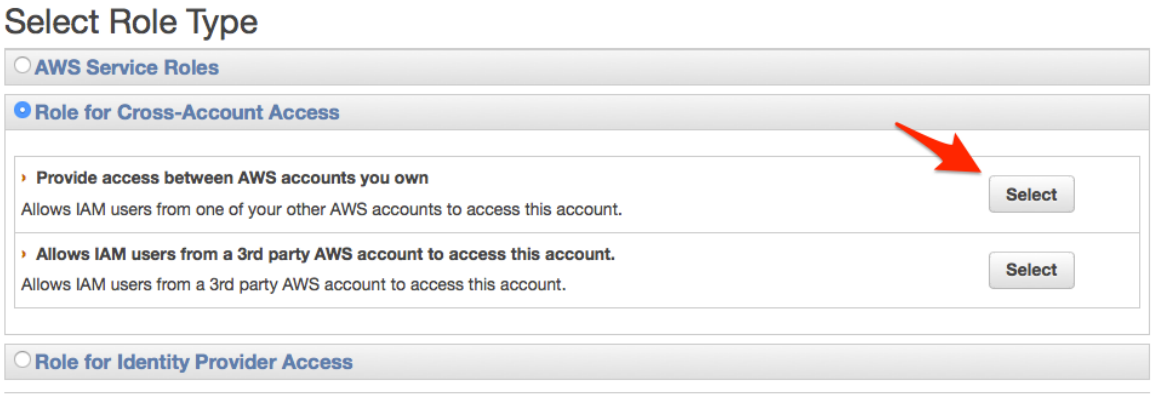
1. Sign in to the **Retail** AWS account and select **Security Credentials** in the account name pull-down menu.
2. Select **Roles** in the left navigation bar, then press the **Create New Role** button



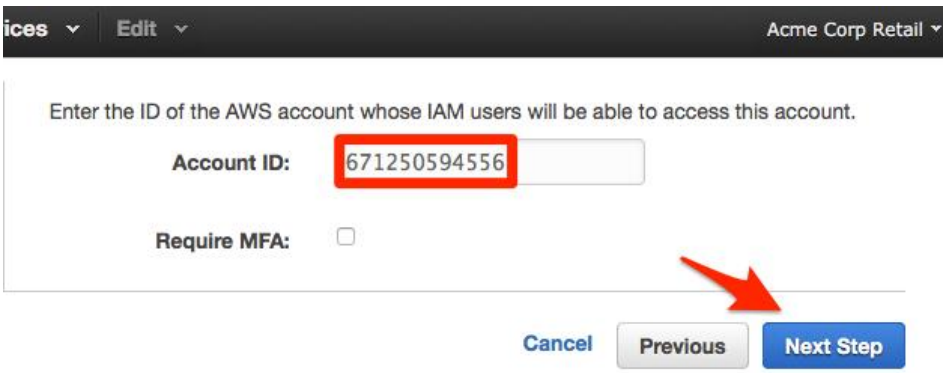
3. In the next steps, we'll create an role that will allow John to switch role as an EC2 Administrator in the Retail account. In the Set Role Name screen, enter in the field and press the button at the bottom.



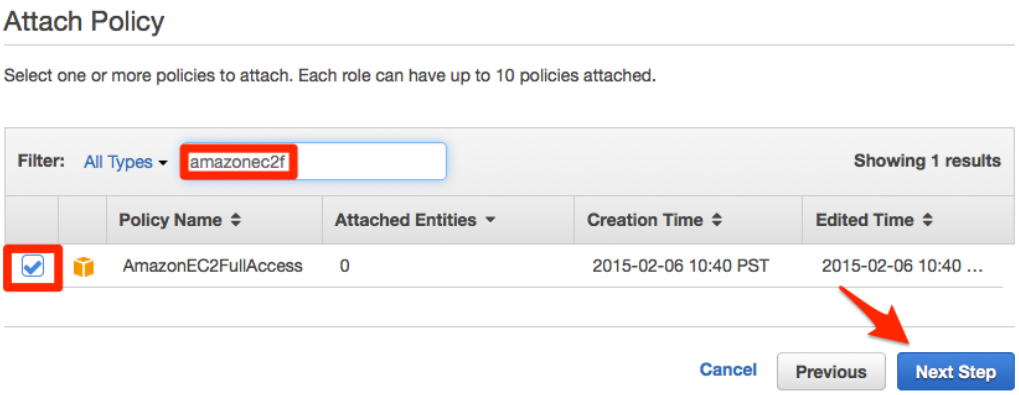
4. In the next screen, select Role for Cross-Account Access and click on the Select button next to Provide access between AWS accounts you own.



5. In the Account ID field, enter the ID of the account as documented in the AWS Accounts table (this ensures that the Retail account trusts the Identity account). Do NOT check Require MFA. Then press

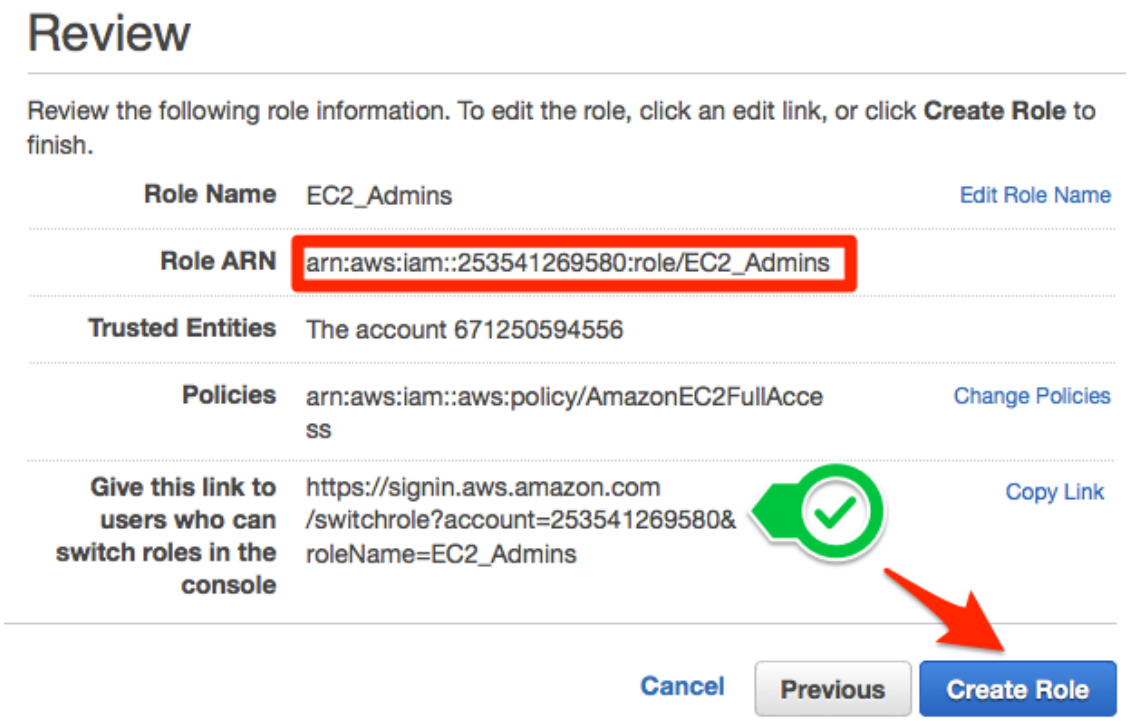


6. In the next screen, set the appropriate permissions for this cross-account role (if you previously created your own policy, you should use the "Customer Managed Policies" filter). In our specific case, we'll just type "amazonec2f" in the filter text box and check the box next to the policy. Next, press the button.



7. The Review screen appears. Please review the information in that screen and copy the Role ARN value into a separate document – we will need it for granting users access to the cross-account roles section below.

Alternatively, copy the link at the bottom if you want to provide it directly to John. When you are done, press to complete the role creation process.

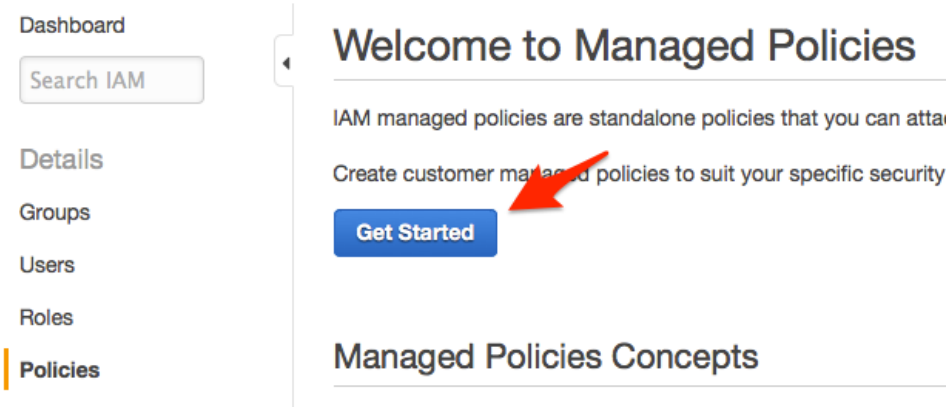


8. Repeat steps 1 to 7 to create an cross-account role in the account as well as another cross-account role in the account.

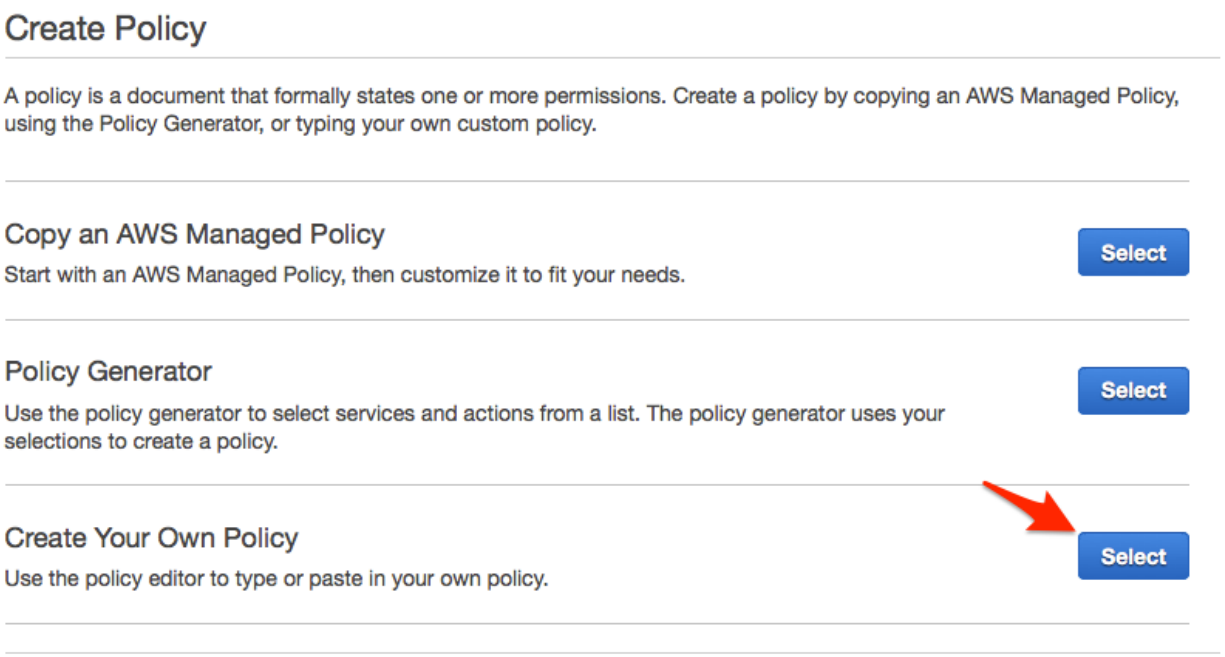
Creating custom "Assume Role" policies

Now that you created the cross-account roles in the Retail and Sales accounts, you must configure the Identity account so that users signing into it have the permission to switch roles and access the Sales or Retail account with the same identity. To do so, we will first have to create custom policies that we will later assign to IAM roles:

1. Sign in to the Identity account and navigate to .
2. Select in the left navigation bar and press (or go straight to the next step).



3. Press at the top and in the following screen, press the Select button next to

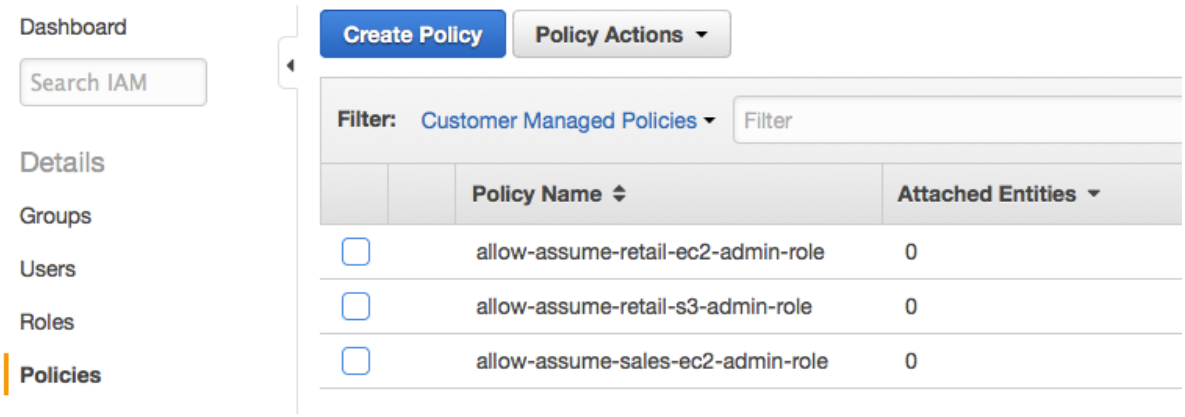


4. In the **Policy Name** field, type a policy name like allow-assume-retail-ec2-admin-role.

5. In the field, enter the following by replacing the Resource field with the Role ARN value retrieved in step #7 of Setting up cross-account roles.

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::ACCOUNT-ID:role/CrossAccountRoleName"
  }
}
```

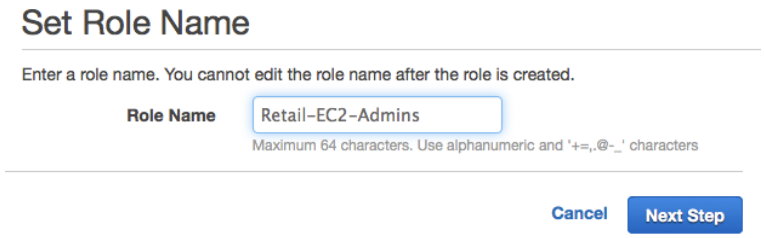
6. Press **Validate Policy** and then **Create Policy**.
7. Repeat the process for the Retail S3_Admins cross-account role and the Sales EC2_Admins cross-account role.
8. You should now see the following in the Policies screen (filtered by "Customer Managed Policies"):



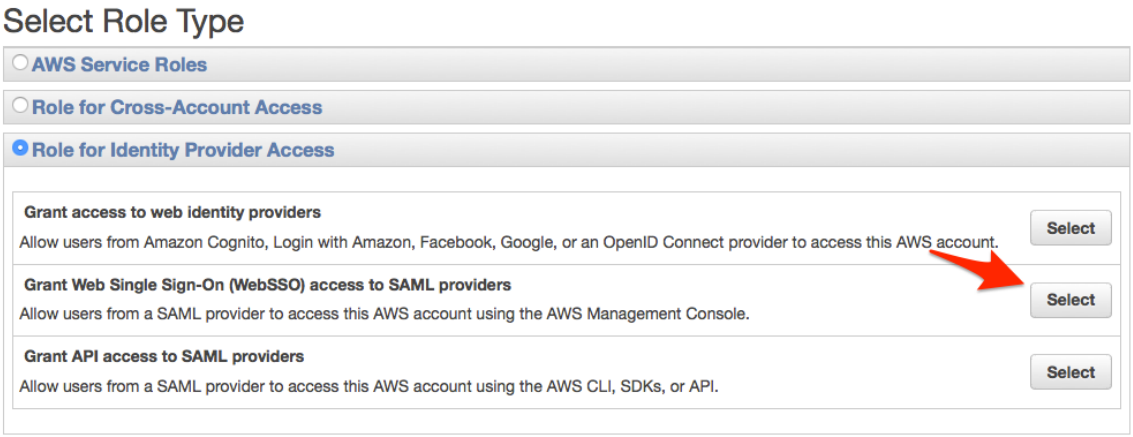
Granting users access to the cross-account roles

: only read the following section after you've gone through the Configure Okta as an Identity Provider in AWS section.

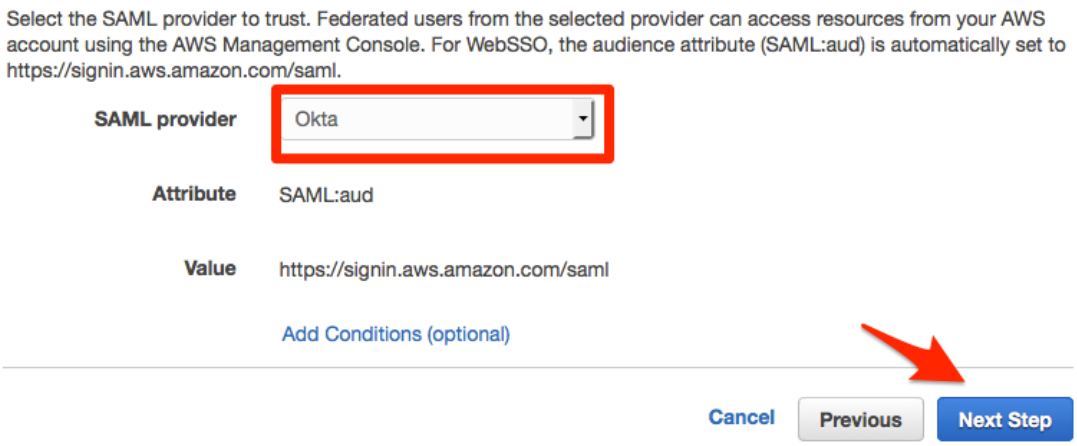
1. Once you have configured Okta as an identity provider in AWS, we can create specific IAM roles that Okta will be able to retrieve and assign to Okta users.
2. Sign in to the Identity account and select Security Credentials ➔ Roles ➔ Create New Role.
3. Set the Role Name to and press .



4. In the next screen, select Role for Identity Provider Access and Grant Web Single Sign-On (WebSSO) access to SAML providers.

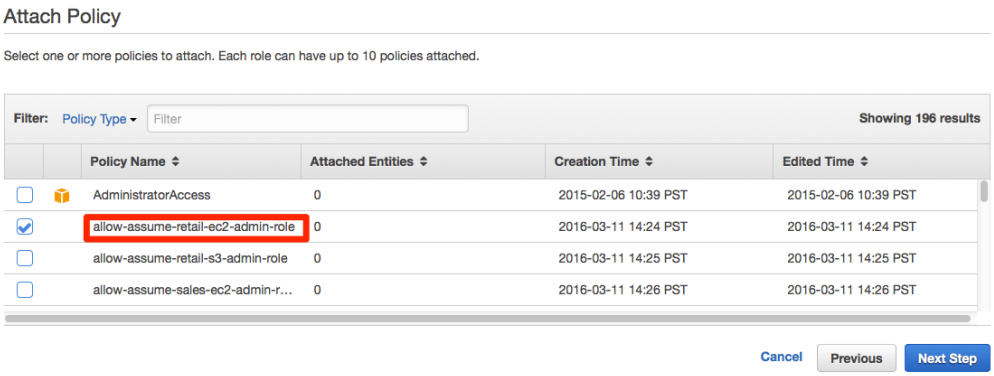


5. In the following screen, select your Okta SAML provider and press .



12.

6. In the Verify Role Trust screen, press Next Step.
7. In the screen, select the appropriate policy (allow-assume-retail-ec2-admin-role for the "Retail-EC2-Admins" role, for instance) and press Next Step.



8. In the Review screen, press .
9. Repeat steps 1 to 8 for the and roles (assigned respectively to the `allow-assume-retail-s3-admin-role` and `allow-assume-sales-ec2-admin-role` policies).

Okta Configuration

There are four ways to integrate Okta with the AWS Admin Web Console:

1. Secure Web Authentication (SWA)
2. IAM Role
3. Federated User Login
4. SAML

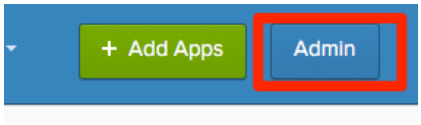
Note: IAM Role and Federated User Login are present only for backward compatibility purposes and should not be used. We also do not recommend using Secure Web Authentication.

SAML 2.0 is the preferred integration type for the cross-account role scenario. SAML 2.0 provides several benefits over SWA. SAML is more secure because there is no password or user account required in AWS. The SAML method also allows your organization to follow AWS IAM Best Practices, including adhering to the principle of least privilege[2].

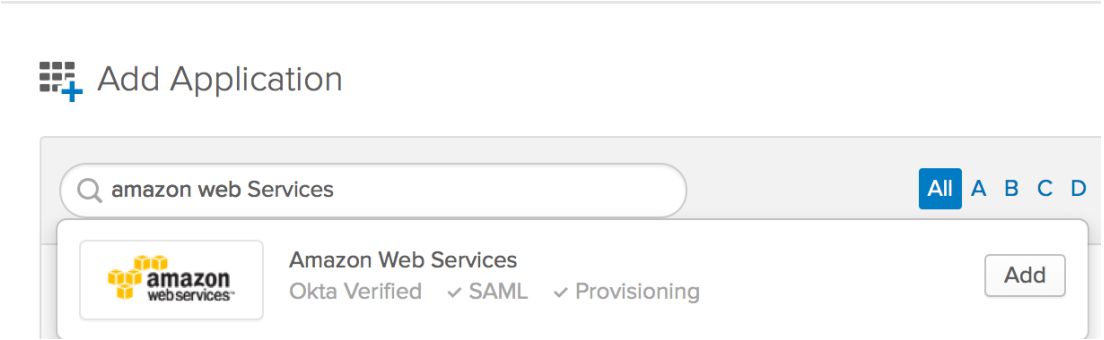
[2] Cf. https://en.wikipedia.org/wiki/Principle_of_least_privilege

Initial Configuration

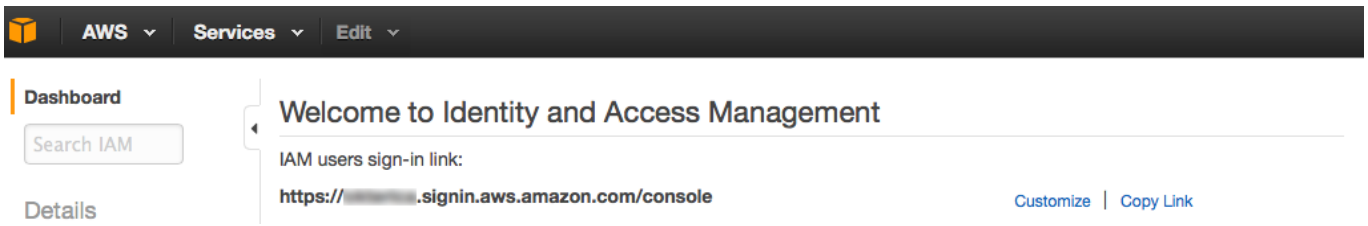
1. If you don't have an Okta organization yet, sign up for the Okta organization of your choice (Free Trial or Developer Edition) at https://www.okta.com/start-with-okta.
2. After the sign-up process is complete, sign in to your Okta tenant with an administrator account and press the Admin button in the top-right corner of the home page.



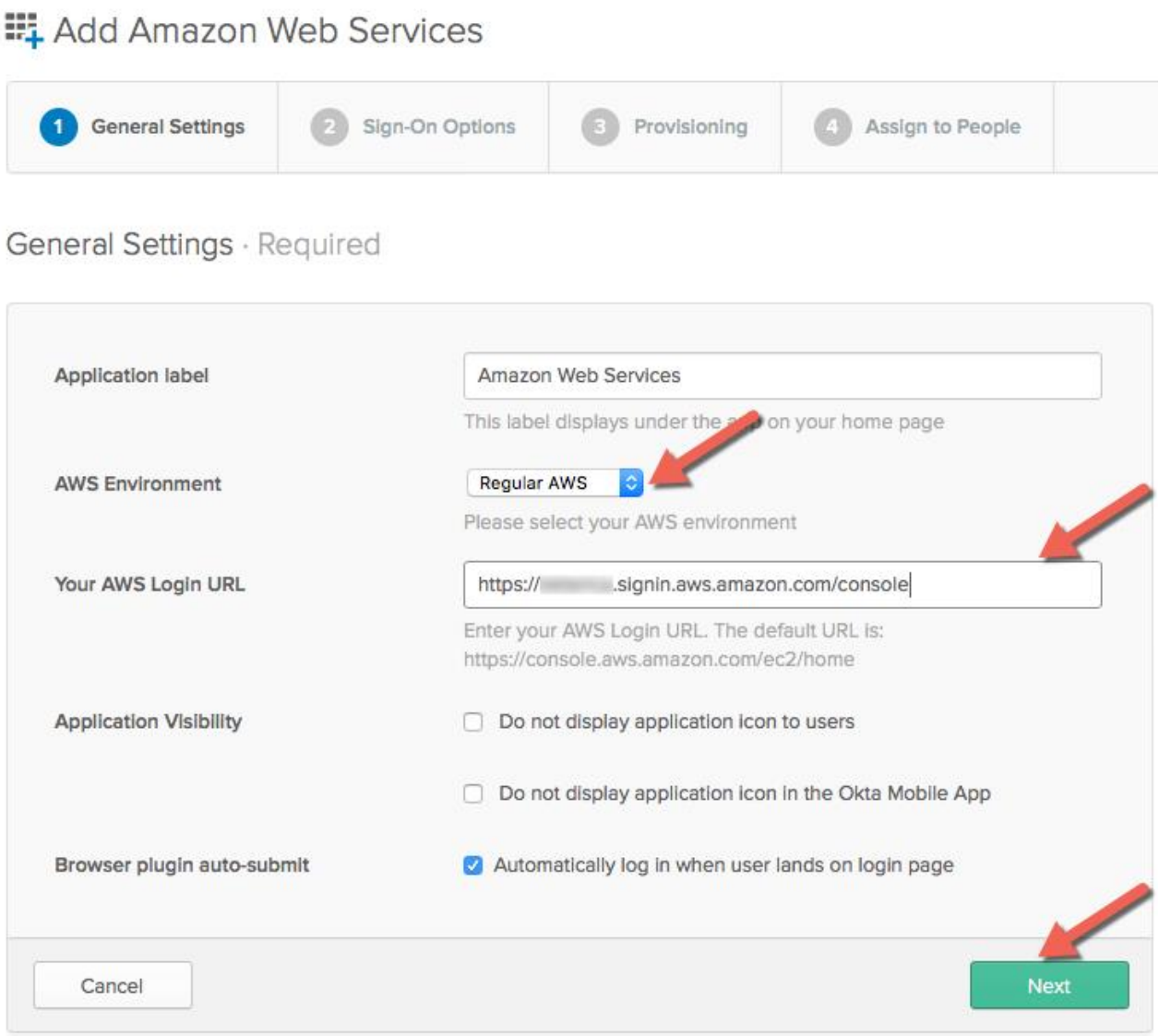
3. In the top navigation bar, select **Applications** and click **Add Application**.
4. In the search bar on the left, type "**amazon web**". This should bring up the Amazon Web Services application, as shown below.



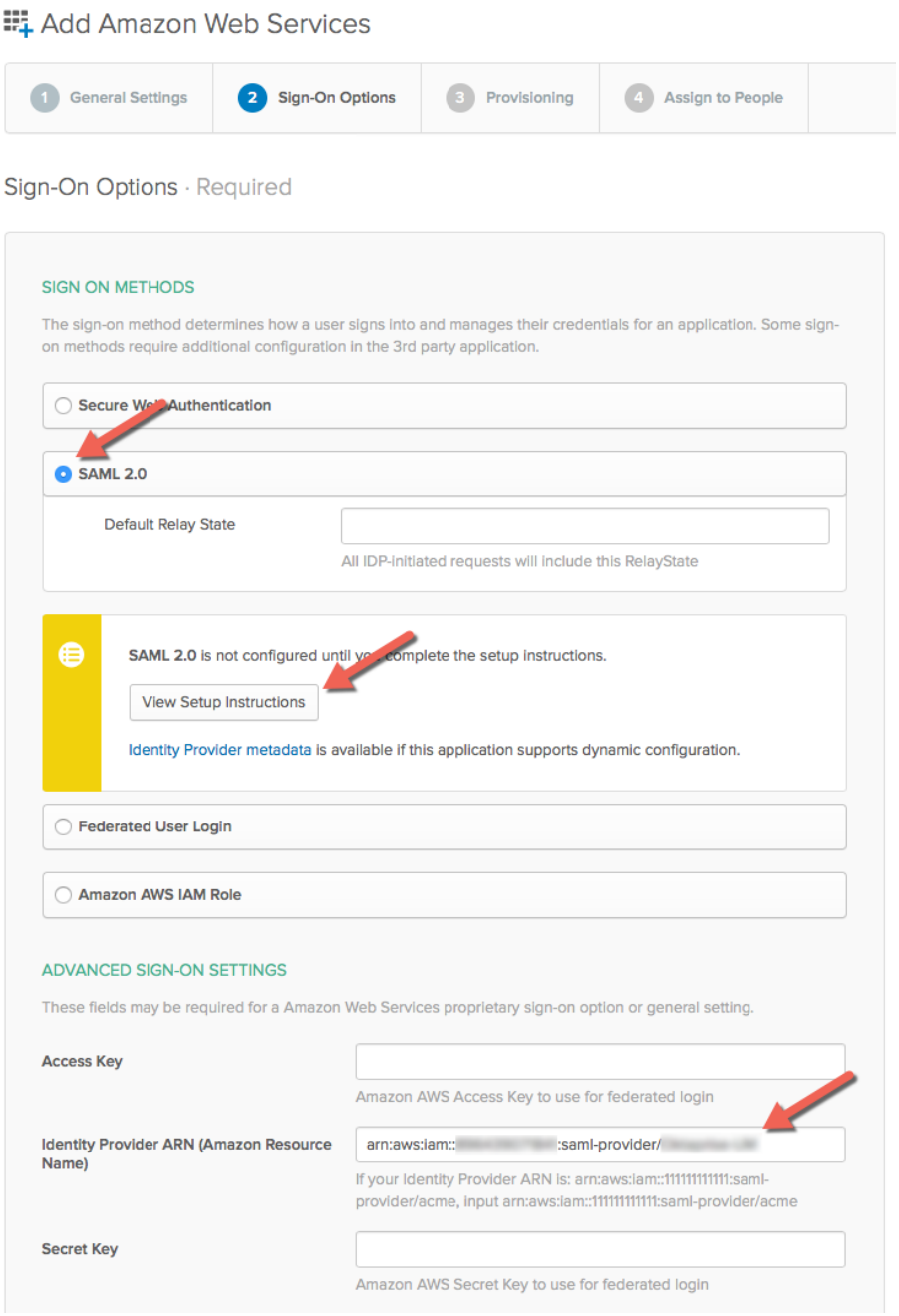
5. Press the **Add** button next to the Amazon Web Services application.
6. In the AWS application wizard, fill in the fields as appropriate for your AWS account. You can find your AWS Login URL on the dashboard of the IAM section of your AWS console (typically it is of the form https://[account-id].signin.aws.amazon.com/console):



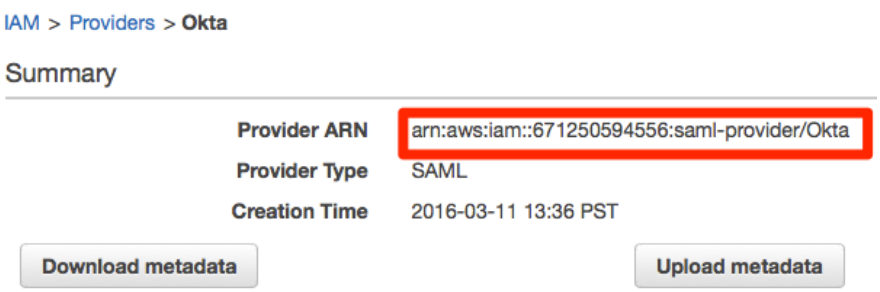
Fill out the required fields as shown in the screenshot below, then click **Next**.



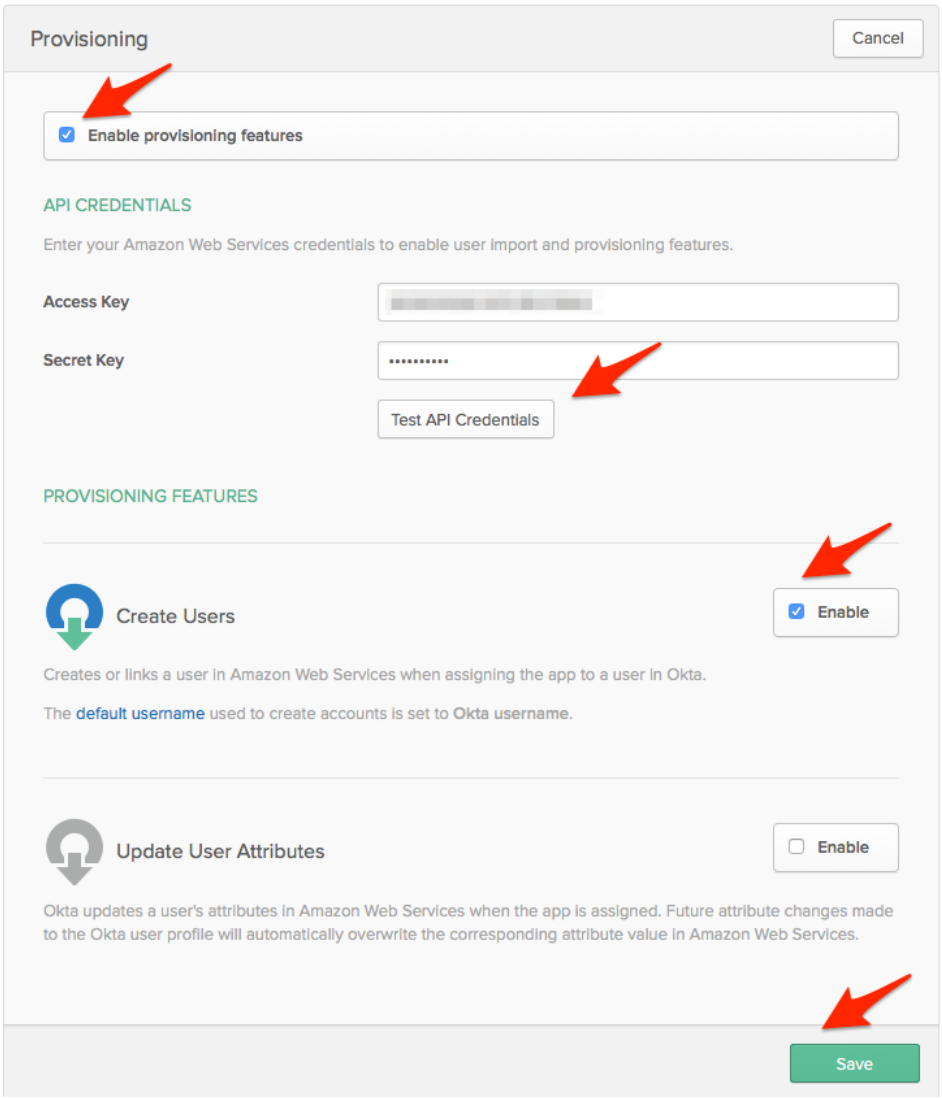
7. On the Sign-On tab, choose the **SAML 2.0** option. Then click **View Setup Instructions** and follow the customized steps for your org (or review the Configure Okta as an Identity Provider in AWS section below).
8. Fill in the **Identity Provider ARN** field with the value of the Identity Provider you created in AWS by following the SAML Setup Instructions. The typical format for this value is the following:
   arn:aws:iam::aws_account_id:saml-provider/idp_name



You can find the ARN value in the AWS Console ➔ **Security Credentials** ➔ **Identity Providers**.



9. Even though you will not be provisioning any user accounts into AWS, enabling the provisioning features initiates the API connection which allows Okta to import the available roles that end users can assume in the AWS service. Note that provisioning is enabled for AWS even if you have not purchased the Provisioning product or equivalent Edition of Okta, to facilitate this functionality. On the Provisioning Tab in Okta, check the **Enable Provisioning features** box.
10. In the **Access Key** and **Secret Key** fields, enter the values you retrieved in step #6 of Generating the API Access Key for Okta and press **Next**.
11. Press **Test API Credentials** to verify that the API credentials are working.
12. Check **Create Users** (but not **Update User Attributes**).
    Note: again, you are not creating or updating any users in AWS, but this activates necessary parts of the API and allows Okta to retrieve Okta-trusted roles from the Identity account.

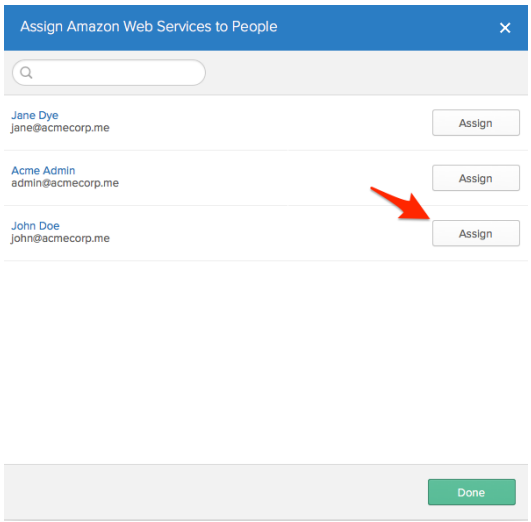


13. Press **Next** (or **Save**)

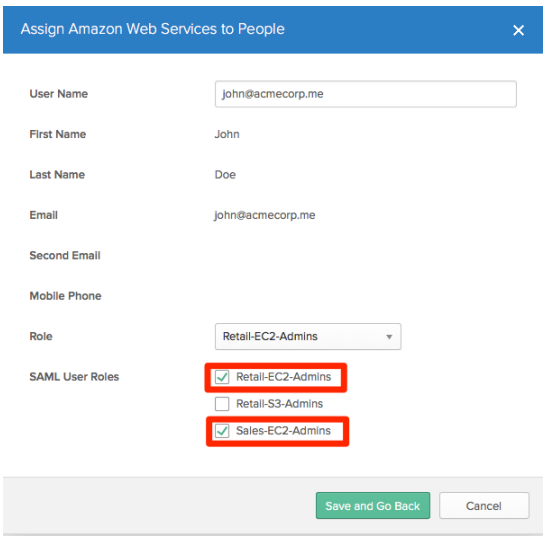
14. In the **Assign [App Name] to People** – Optional screen, don't select any user yet – we will perform that step later on once we have created IAM roles in the Identity account. Instead, just press **Next** and **Done** in the following screen.
15. Proceed with the Granting users access to the cross-account roles section to create Okta-trusted roles in the Identity account.

Assign AWS to Okta users

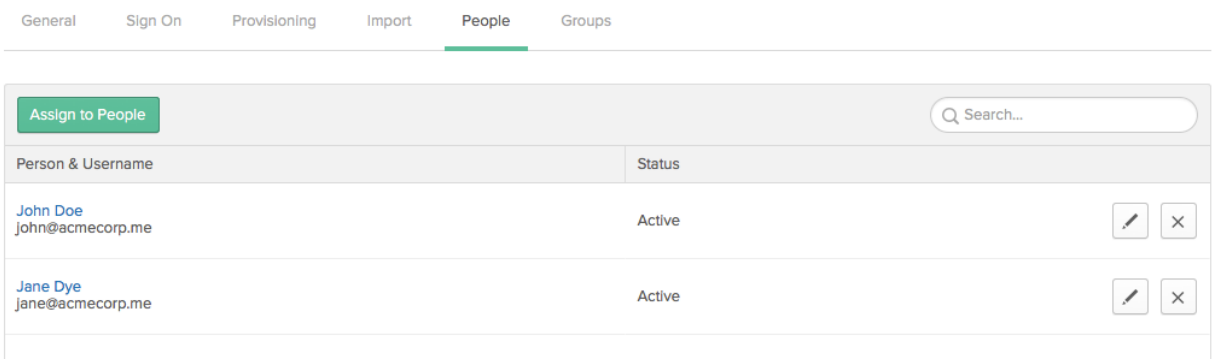
1. In Okta, Admin ➔ Applications ➔ Amazon Web Services, select the Provisioning tab.
2. Press the **Edit** button (at the top) and the **Save** button (at the bottom). This will refresh Okta with the latest list of Okta-trusted roles you created in your AWS Identity account.
   Note: you should perform that action every time you add or remove a role in AWS.
3. Select the **People** tab and press the **Assign to People** button. Press **Assign** next to John's account:



4. In the opening screen, select the **Retail-EC2-Admins** and the **Sales-EC2-Admins** SAML User Roles (ignore the Role dropdown value which only applies when selecting the "IAM User Role" sign-on option). Press **Save** and **Go Back.**



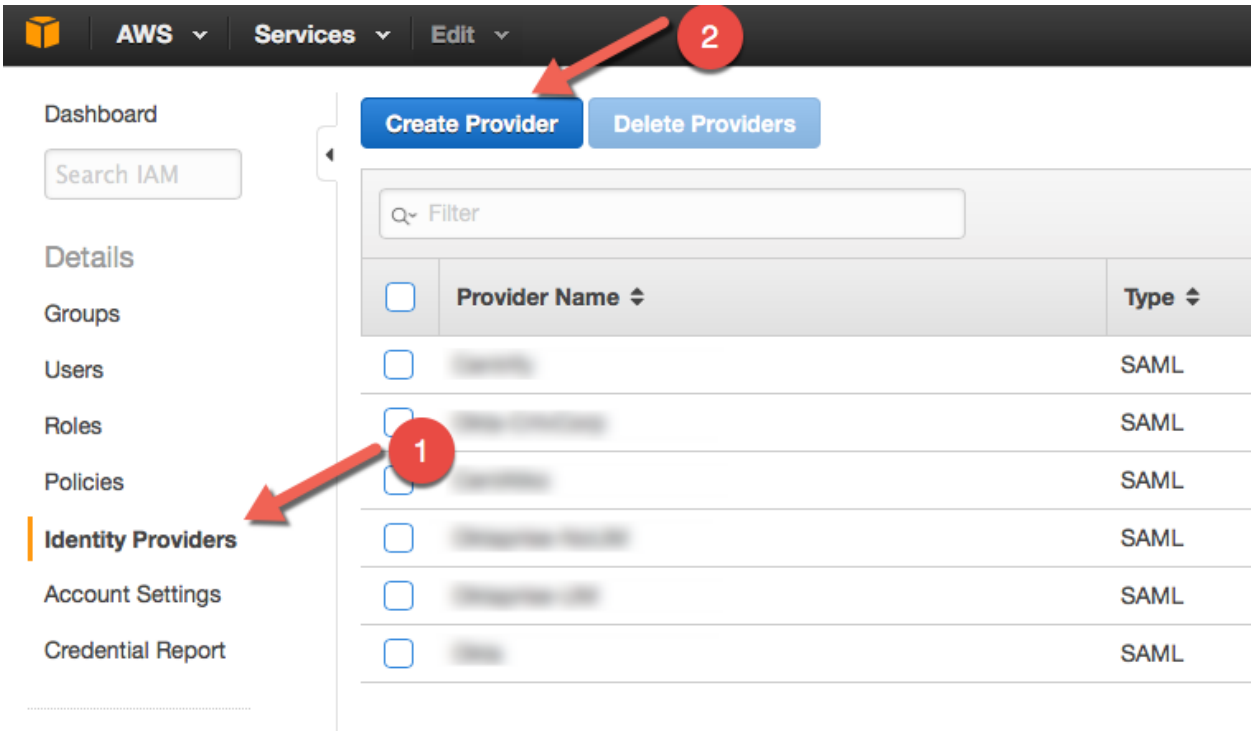
5. Repeat steps 3 and 4 for Jane's account and assign her to the **Retail-S3-Admins** role.
6. Press the **Done** button. You should see the following screen:



For more about how roles for AWS with Okta work, see Appendix B.

Configuring Okta as an Identity Provider in AWS

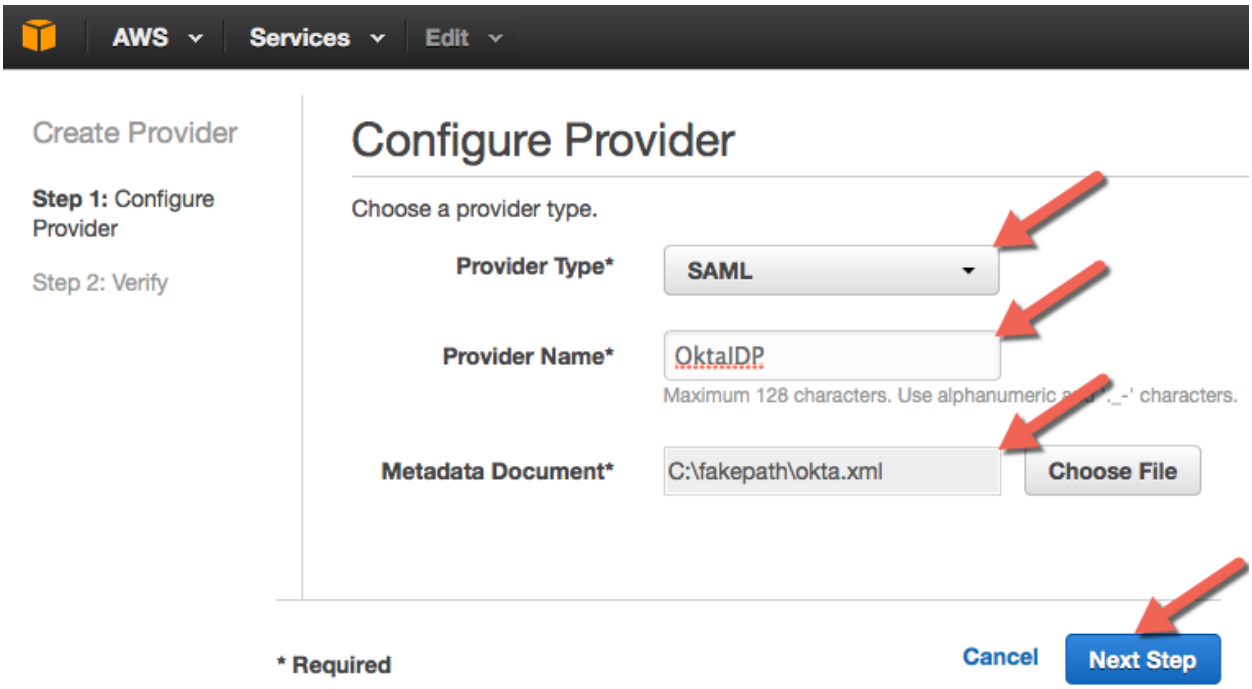
1. Sign into AWS and go to the IAM console and click on **Identity Providers**.
2. Select **Create Provider**, as shown below.



3. Go back to Okta. On the Sign-On tab for the AWS app integration, click **View Setup Instructions**. These instructions are customized for the specific instance of Okta that's being integrated. Copy the IdP metadata

from Step 3 of the instructions page and save it as okta.xml on your local computer. You will need to upload this file to finish configuring the Identity Provider in AWS.

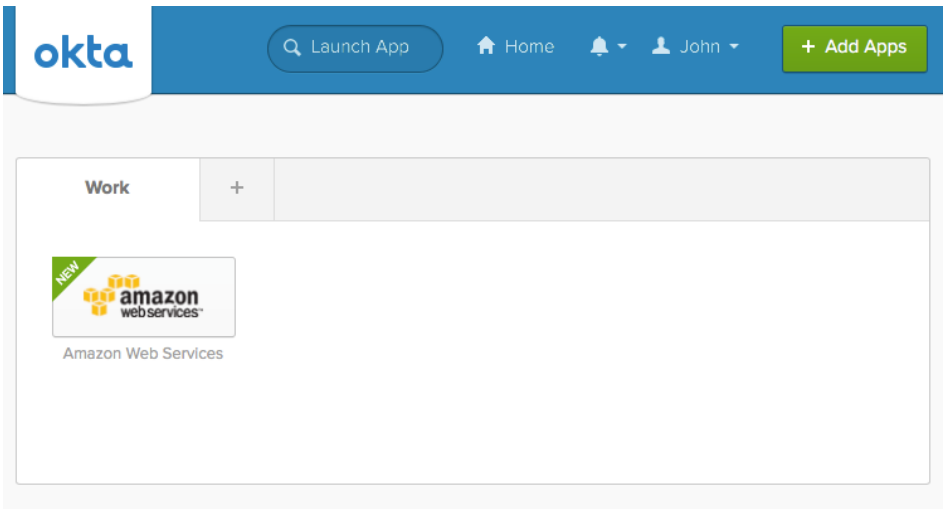
4. In AWS, in the **Configure Provider** screen, choose **SAML** as the provider type, enter a friendly name for the Provider Name field, and browse to the XML file that you just created. Then click **Next Step**.



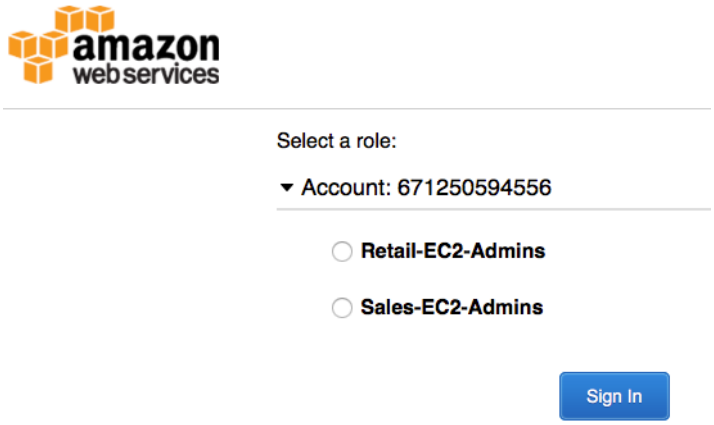
5. Click **Create** to finish.

Testing that the integration works

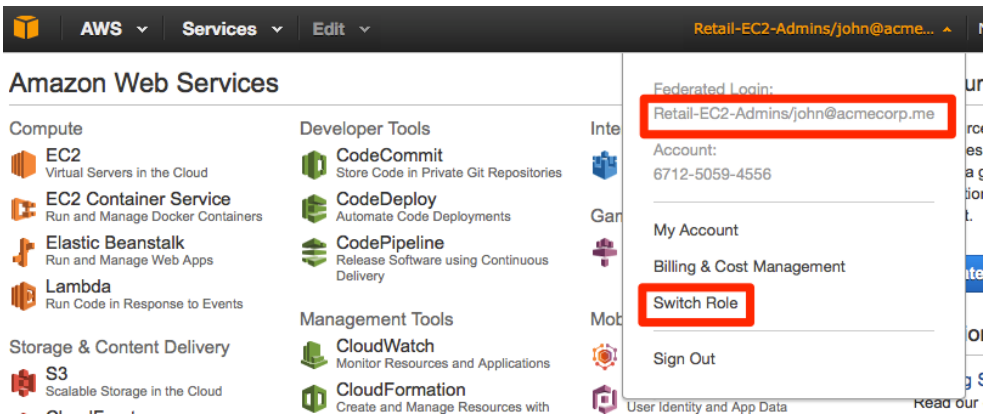
1. Navigate to the login screen of your Okta organization and sign in with John's account
2. You should see the Amazon Web Services chiclet on the dashboard. Click on the AWS icon.



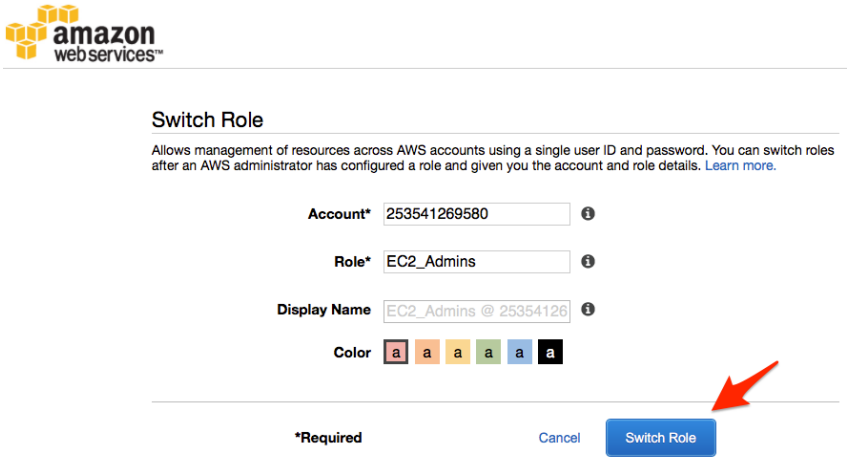
3. Upon signing in, Amazon should prompt you to choose one of the two following roles:



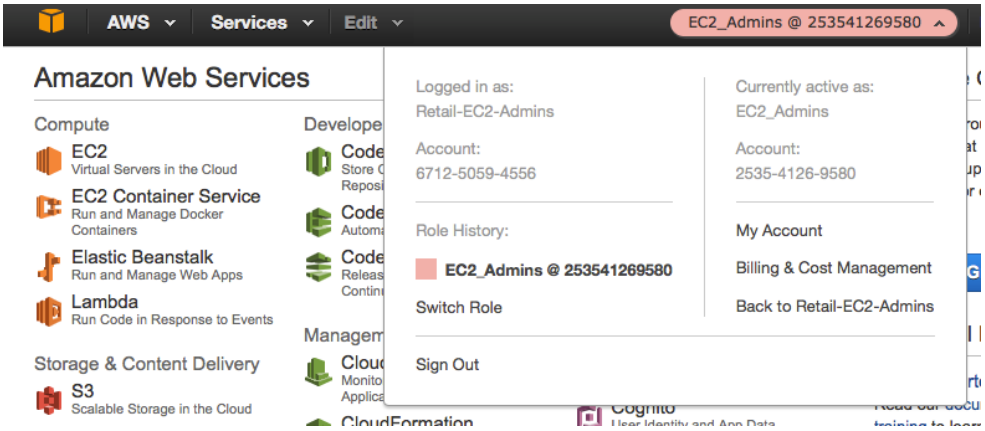
4. Select **Retail-EC2-Admins** and press the **Sign In** button. You should see the following screen and information:



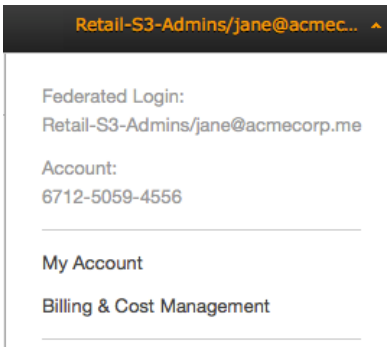
5. If you saved the Switch Role links from step #7 of Setting up cross-account roles, please enter it now in your browser. The format of the link should be similar to:
   `https://signin.aws.amazon.com/switchrole?account=[retail_account_id]&roleName=EC2_Admins`
6. The following screen appears. Press **Switch Role**.



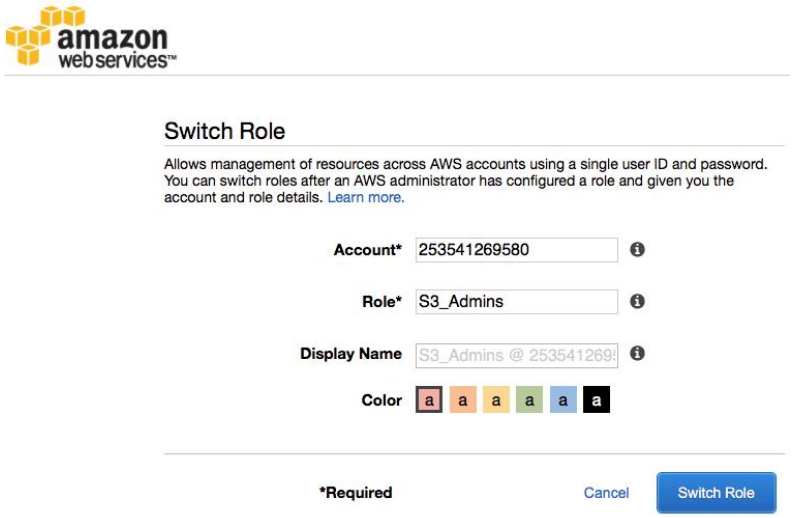
7. The following screen appears



8. Notice that you can only access the **EC2** and **EC2 Container Service** pages.
9. Sign out from AWS and sign out from Okta.
10. Sign in into Okta with Jane's account. Select the AWS icon.
11. You should be automatically signed in to AWS with the following identity:

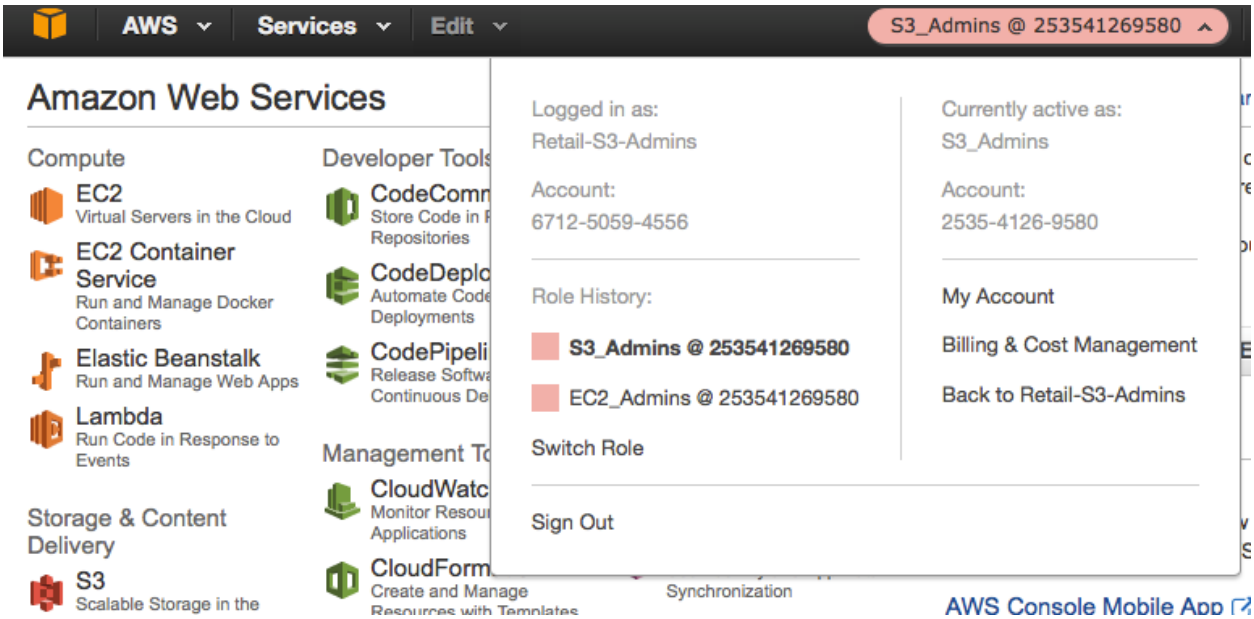


12. Navigate to the Retail-S3-Admins switch role link. The following screen appears:



14.

13. Press the **Switch Role** button and see the following screen appear:



14. Notice that you can only access the S3 console page.

Using the Okta AWS CLI Assume Role Tool

Introduction

When using the Security Assertion Markup Language (SAML) to enable Amazon Web Services (AWS), the AWS Command Line Interface (CLI) does not inherit that configuration by default. However, most customers who integrate with AWS also want a Single Sign-On (SSO) solution for the CLI.
This section shows how you can leverage the CLI with Okta and the cross-account role design previously exposed in this document.

AWS Architecture

AWS provides the ability to obtain an AWS security token in exchange for a SAML assertion, using their Security Token Service (STS)/Application Program Interfaces (API)s. The CLI can take advantage of this functionality, and, through the process outlined below, allow SAML-based authentication.

CLI Integration Process using the AWS Security Token Service API

This process consists of the following three steps:

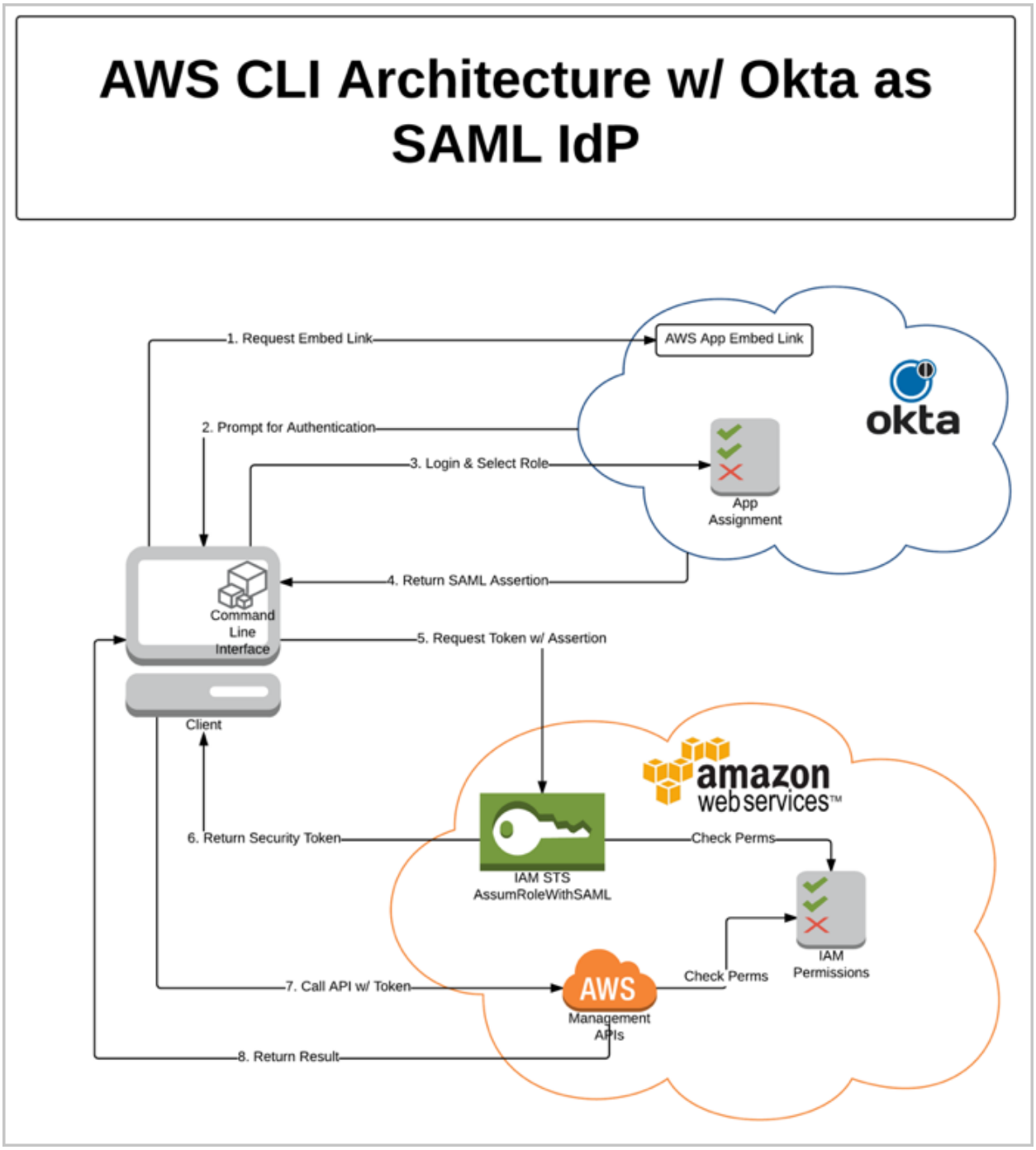
1. Call the Okta IdP to obtain a SAML assertion.
2. Call the AWS STS API and exchange the SAML assertion for a temporary, 1-hour long, security token.
3. Invoke the AWS CLI with the temporary security token.

Solution Overview

A custom tool was developed using Okta as an IdP to invoke steps 1 and 2 in the AWS architecture process above, showcasing how customers can take advantage of the solution. At a high level the custom tool performs the following actions:

1. Prompts the user for Okta credentials (which may be AD or LDAP credentials replicated with Okta)
2. Call Okta's AWS Embed URL to generate the SAML assertion and extracts the AWS IAM Roles assigned to the user from that assertion.
3. Prompts the user to select one of the available AWS IAM Roles.
4. Submits the resultant SAML assertion to the AWS STS API along with the selected role.
5. Extracts the AWS IAM Roles assigned to the user from the SAML assertion and prompts the user to choose one of them.
6. Optionally, if the user selected a cross-account role, the tool extracts the target role ARN (Amazon Resource Name, such as `arn:aws:iam::253541269580:role/EC2_Admins`) and writes a linked profile entry into the local AWS config file.

Solution Overview Diagram



Prerequisites

Before implementing the custom code sample, the following should be in place:

In your Okta tenant:

- The Okta SAML integration to AWS must be completed and functional.

Note: The only supported Sign On method is SAML. SWA, Federated User Login, and Amazon AWS IAM Role Sign On methods are not supported for CLI integration.

In your AWS account:

- Go to the Identity and Access Management page and create an IAM user with specific permissions, as mentioned in Appendix A. Generate an **Access Key ID** and **Secret Access Key** and store them in a temporary location: you will need them in order to write a specific entry into the **~/.aws/config** file.
- We strongly recommend that cross-account roles set up on the Identity account be only associated with **one** AssumeRole action. This is because the current version of the tool only picks up the first AssumeRole action it finds when it examines the selected role on the Identity account. For more information, please refer to the AWS documentation.

On your AWS Client workstation:

- Install Java
  - The custom tool requires Java 1.8 or better.
  - Verify that your AWS CLI version is 1.8 or higher (required by the tool)

Installation and Configuration

Follow these steps to install and configure the custom tool:

1. Download the package from https://github.com/oktadeveloper/okta-aws-cli-assume-role

   Note: The only files you will really need are in the `lib` and `out` directories.

15. Download the AWS SDK for Java from https://aws.amazon.com/sdk-for-java/ and put the unzipped main jar (such as `aws-java-sdk-1.10.74.jar`) in the `lib` directory.
16. In the `lib` folder, edit the `awscli.command` file if you use Linux/Mac OS X (or the `awscli.bat` file if you use Windows) and change the value of the `aws-java-sdk` library to match your value of the AWS SDK for Java (by default, the file is configured with `aws-java-sdk-1.10.74.jar`)
17. In the `out` directory, edit the `config.properties` file to represent your environment:
    - ⇒ `OKTA_ORG` is the FQDN of your Okta org (such as `acmecorp.okta.com`)
    - ⇒ `OKTA_AWS_APP_URL` is the AWS App Embed URL from the General tab of your AWS application in your Okta org.
    - ⇒ `AWS_IAM_KEY` is the Access Key ID for the IAM User you previously created
    - ⇒ `AWS_IAM_SECRET` is the Secret Access Key for the IAM User you previously created

An example `config.properties` file is shown below:

```
OKTA_ORG= acmecorp.okta.com
OKTA_AWS_APP_URL=https://acmecorp.okta.com/home/amazon_aws/0ac4qfegf372HSvKF6a3/965
AWS_IAM_KEY=AKIAJM4XALB5VEREGQJQ
AWS_IAM_SECRET=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
```

You're now ready to execute the custom tool.

Execution

Part 1: Obtain the SAML assertion and Request an AWS STS token

1. Navigate to the directory that contains the Okta AWS-CLI Assume Role tool.

18. Navigate inside the `out` sub-directory.

19. Run the following command from a command line:

    `./awscli.command` (or awscli.bat on Windows)

    or

    `java -cp oktaawscli.jar:../lib/aws-java-sdk-1.10.74.jar com.okta.tools.awscli`

    (or on Windows:

    java -cp oktaawscli.jar;..\lib\aws-java-sdk-1.10.74.jar com.okta.tools.awscli)

20. Enter the username and password of a valid Okta user assigned to the AWS Okta app.

```
Username: jane
Password:
```

21. If applicable, select an MFA Factor

```
Multi-Factor authentication required. Please select a factor to use.
Factors:
[ 1 ] : Security Question
[ 2 ] : Google Authenticator
Selection: 1

Security Question Factor Authentication
Enter 'change factor' to use a different factor

What was your dream job as a child?
Answer: pompier
```

22. Select the AWS role you would like to assume (among all the AWS roles the user was assigned to in Okta)

```
Please choose the role you would like to assume:
[ 1 ]: arn:aws:iam::671250594556:role/Retail-EC2-Admins
Selection: 1
```

The tool acquires temporary, 1-hour long credentials from the AWS Security Token Service (STS) and stores them in the local `~/.aws/credentials` file. Optionally, if the tool detects that the selected AWS role is mapped to an AssumeRole action, it will also write a corresponding entry in the `~/.aws/config` file that will allow the user to automatically access the permissions assigned to the AssumeRole in question. It will then output a message similar to the following one:

> *Your new access key pair has been stored in the aws configuration file with the following profile name:*
> ***671250123543/Retail-EC2-Admins/jane@company.com***
> *The AWS Credentials file is located in /Users/username/.aws/credentials.*
> *Note that it will expire at X/X/XX 0:00 PM*
> *After this time you may safely rerun this script to refresh your access key pair.*
> *To use these credentials, please call the aws cli with the --profile option (e.g. aws --profile 671250123543/Retail-EC2-Admins/jane@company.com ec2 describe-instances)*

23. Take note of the profile name you just generated (in this case, `671250123543/Retail-EC2-Admins/jane@company.com`) as you will need it to call the AWS Command Line Interface (as shown in the next section).

Part 2: Use the token

1. Open a command line.
2. Run the following command (for example):

   aws –-profile [your profile name] ec2 describe-instances

   by specifying the profile name you generated in the previous section. If you don't remember your profile name, you can look up the credentials file and try to identify the proper profile to use. As shown at the end of the previous section, the Assume Role tool also outputs your specific profile name in the message it displays at the end.

Appendix A: How to create an IAM User for role introspection

With its new support of cross-account roles, the Assume Role tool introduces a new convenience for AWS users to enhance the usability of the AWS CLI. When using our Assume Role tool, the AWS Security Token Service will generate temporary credentials that have the permissions associated with the selected role in the Identity account. However, when used as a proxy to a cross-account role located in another account, this role in the Identity account typically only has **sts:AssumeRole** permissions which do not give it sufficient permissions to execute the AWS operations assigned to the cross-account role it is associated with (such as **ec2 describe-instances**). To solve this issue, AWS allows local users to add an entry to their ~/.aws/config file mentioning the target role ARN (cf. AWS documentation)

The Assume Role tool provides an optional convenience that allows the automation of the second option but it requires the creation of a specific IAM User with specific permissions as well as the distribution of this user's Access Key and Secret to AWS users who will want to take advantage of this convenience.

If you want to go ahead with that automation, you will need to create an IAM user with the following read-only permissions (for instance, added in an inline policy, using the Policy Generator and the Identity and Access Management service):

- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:GetRolePolicy
- iam:ListAttachedRolePolicies
- iam:ListRolePolicies

Effect	Action	Resource
Allow	iam:GetPolicy iam:GetPolicyVersion iam:GetRole iam:GetRolePolicy iam:ListAttachedRolePolicies iam:ListRolePolicies	*

Once the permissions above have been assigned to the IAM User, go to the Security Credentials tab and press the **Create Access Key** button. Then copy the resulting **Access Key ID** and **Secret Access Key** into the **AWS_IAM_KEY** and **AWS_IAM_SECRET** parameters values of the **config.properties** file located in the **out** directory.

If you do not want to use this automation, you will have to figure out a way to provide the cross-account role ARN (mentioned in the AssumeRole policy) to the AWS user so that it can be manually added to her ~/.aws/config file. The format of the expected entry in the config file is the following:

[profile {*profile_name_from_credentials_file*}] (such as 671250594556/Retail-EC2-Admins/john@acmecorp.me)
role_arn={*role arn in target AWS account*} (such as arn:aws:iam::253541269580:role/EC2_Admins)
source_profile={*profile_name_from_credentials_file*} *(same as in [profile] line above)*

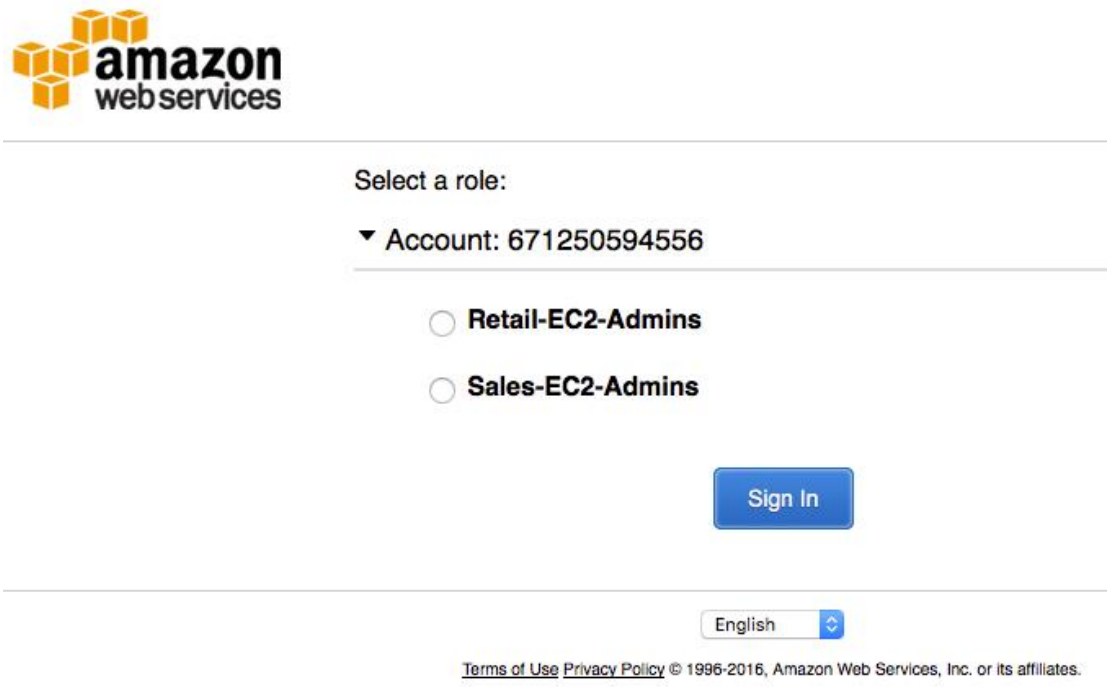
Appendix B: More about roles with AWS and Okta

The roles you select when you assign AWS to a set of users in Okta are provided to AWS as SAML attributes within the end user's SAML response. The SAML response looks something like this for the **john@acmecorp.me** account:

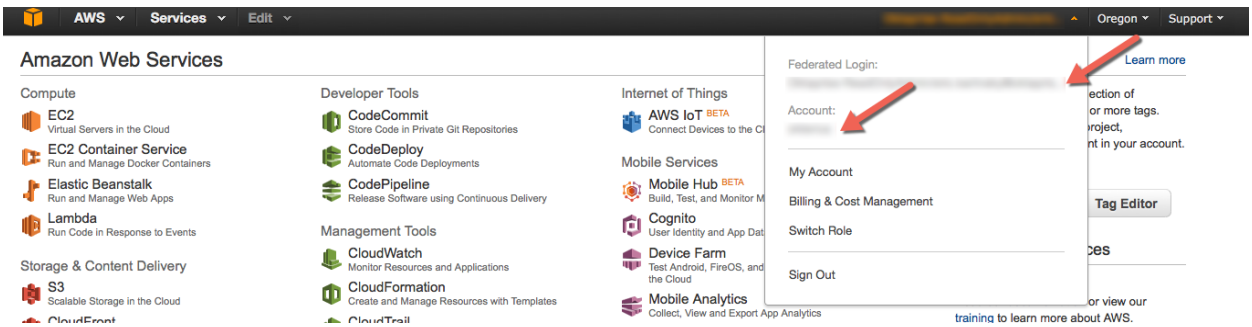
```
<saml2:Assertion>
…
<saml2:Subject xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:unspecified">john@acmecorp.me</saml2:NameID>
            <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
                <saml2:SubjectConfirmationData NotOnOrAfter="2016-03-11T23:55:27.007Z"
Recipient="https://signin.aws.amazon.com/saml"/>
            </saml2:SubjectConfirmation>
        </saml2:Subject>
        <saml2:Conditions NotBefore="2016-03-11T23:45:27.007Z" NotOnOrAfter="2016-03-
11T23:55:27.007Z"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"><saml2:AudienceRestriction>
<saml2:Audience>urn:amazon:webservices</saml2:Audience>
            </saml2:AudienceRestriction>
        </saml2:Conditions>
        <saml2:AuthnStatement AuthnInstant="2016-03-11T23:50:27.007Z"
SessionIndex="id1457740227007.1664297527"
xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:AuthnContext>
<saml2:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTrans
port</saml2:AuthnContextClassRef>
            </saml2:AuthnContext></saml2:AuthnStatement>
        <saml2:AttributeStatement xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
            <saml2:Attribute Name="https://aws.amazon.com/SAML/Attributes/Role"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
<saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">arn:aws:iam::671250594556:saml-
provider/Okta,arn:aws:iam::671250594556:role/Sales-EC2-Admins</saml2:AttributeValue>
<saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="xs:string">arn:aws:iam::671250594556:saml-
provider/Okta,arn:aws:iam::671250594556:role/Retail-EC2-Admins</saml2:AttributeValue>
            </saml2:Attribute>
            <saml2:Attribute
Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName"
```

```
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:basic">
                <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string"
>john@acmecorp.me</saml2:AttributeValue></saml2:Attribute>
</saml2:AttributeStatement>
</saml2:Assertion>
```

Amazon Web Services consumes the SAML Assertion and prompts the end user to select from a list of all available roles. So in this example, the end user would see a screen like the following:



The end user then selects the role whose rights are required for the current task, and Okta signs the user into the AWS console with those permissions.



You should see the user account, the Role and the account information when you click on the down arrow next to the username in the top-right corner of the AWS console.