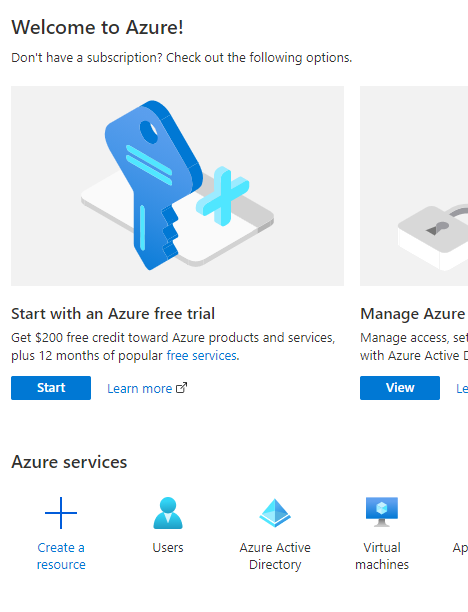
<https://aws.amazon.com/blogs/aws/the-next-evolution-in-aws-single-sign-on/>

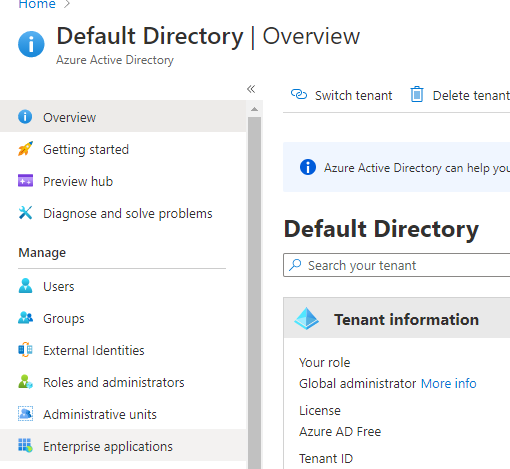
<https://pages.awscloud.com/-How-to-Use-Azure-Active-Directory-with-AWS-SSO_2020_0219-SID_OD.html>

login into azure portal

click azure active directory

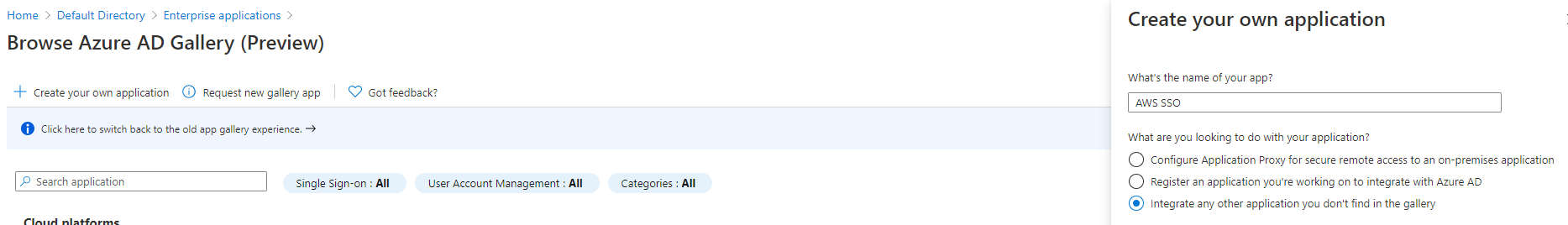


Click enterprise application



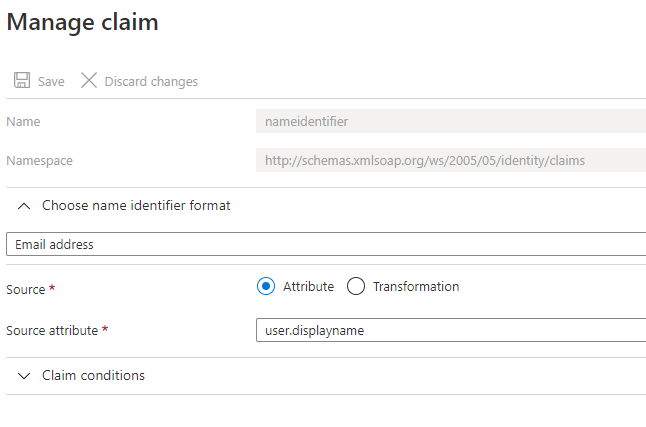
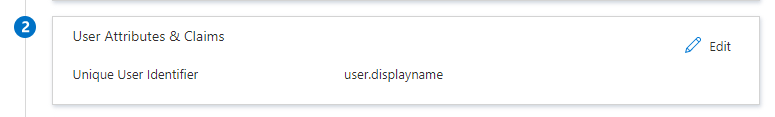
Click new application

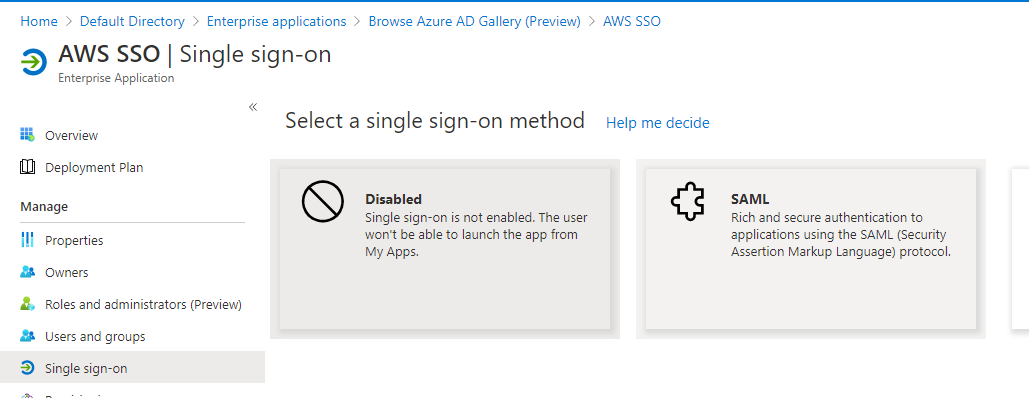
Click “Create your own application”

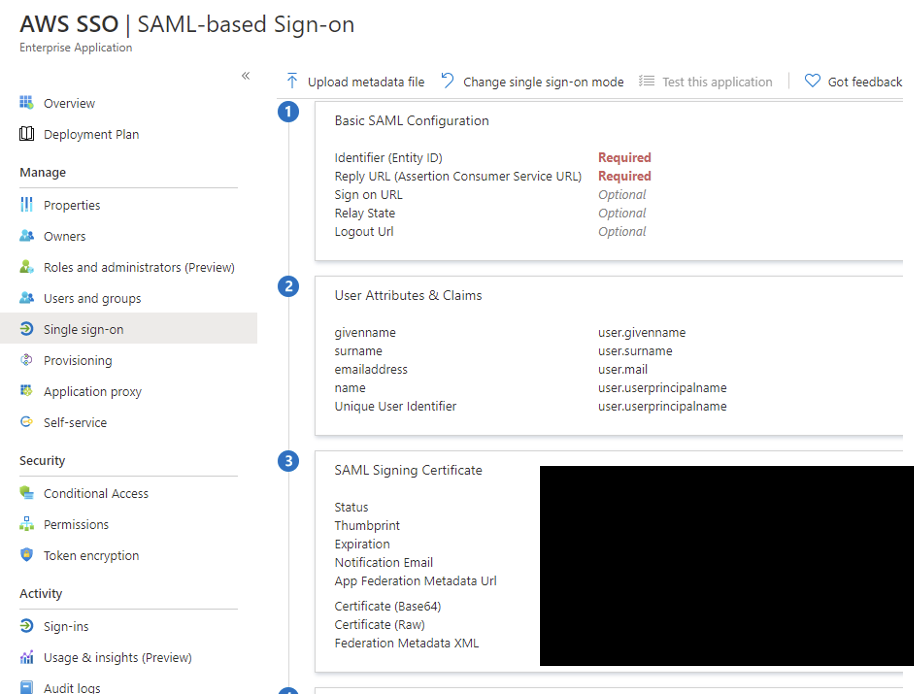


This is where I will configure the settings to enable single sign-on and exchange federation metadata between Azure AD and AWS Single Sign-On. I select **Single sign-on** from the navigation panel and then click the **SAML** option. From the settings page that then opens, I need to download the **Federation Metadata XML** file, which is shown in the **SAML Signing Certificate** panel. This file, which will be named **AWS SSO.xml** by default, will be used to configure AWS Single Sign-On in the next steps.

**\*IMPORTANT!!! On “user Attributes \* Claims”, unique user identifier somehow I need to put as displayName, and format MUST BE EMAIL**

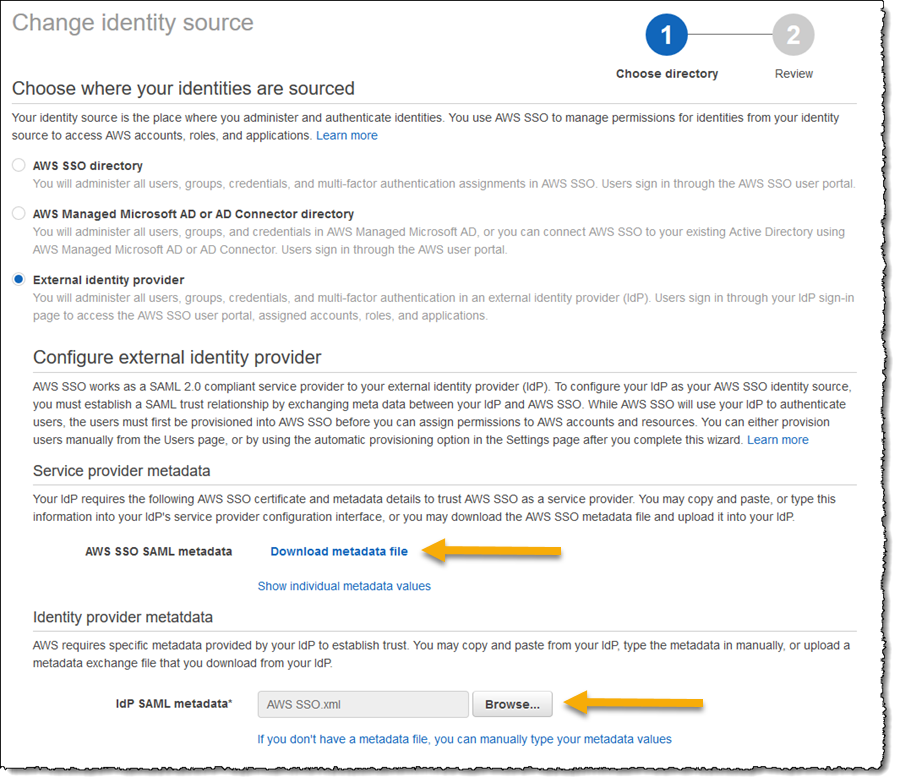
****





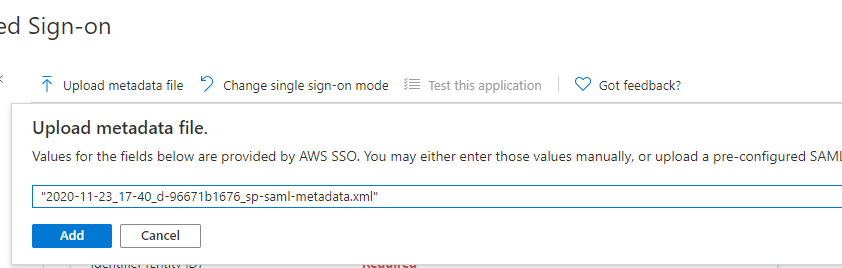
Having downloaded the file, I open another browser tab (I leave the Azure AD tab open, as I need to come back to it), and sign into the [AWS Single Sign-On Console](https://console.aws.amazon.com/singlesignon). From the dashboard home, I click the **Enable AWS SSO** button. After a few seconds, my account is enabled for single sign-on and I can proceed to configure the connection with Azure AD.

Clicking **Settings** in the navigation panel, I first set the **Identity source** by clicking the **Change** link and selecting **External identity provider** from the list of options. I now have two things to do. First, I need to download the **AWS SSO SAML metadata** file using the link on the page – I will need this back in the Azure AD portal. Second, I browse to and select the XML file I downloaded from Azure AD in the **Identity provider metadata** section.



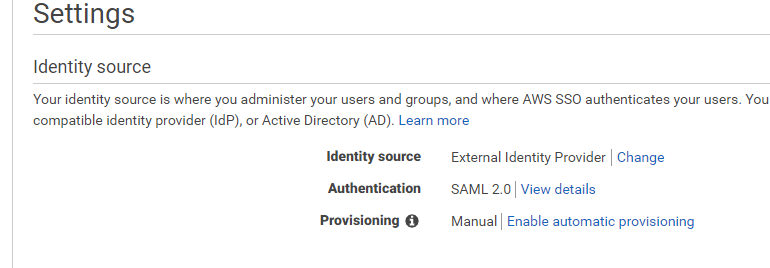
 click **Next: Review**, enter **CONFIRM** in the provided field, and finally click **Change identity source** to complete the AWS Single Sign-On side of the process.

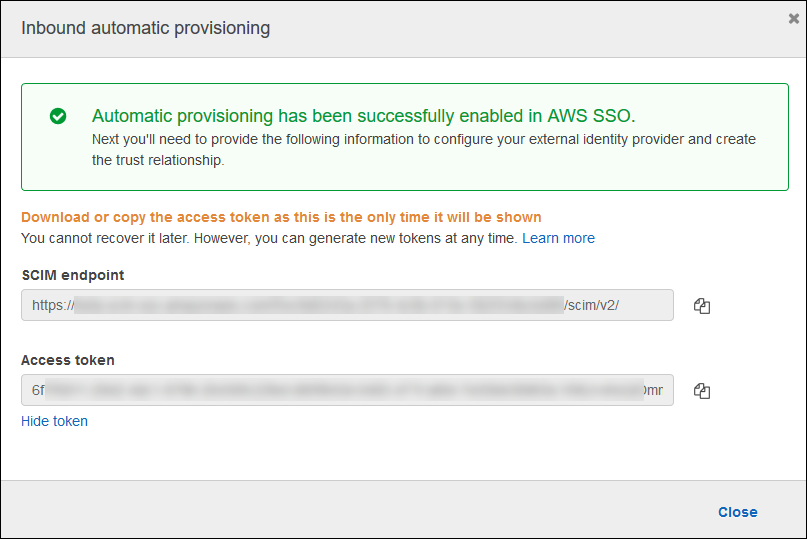
Returning to the tab I left open to my Azure AD **Set up Single Sign-On with SAML** settings page, I click the **Upload metadata file** button at the top of the page, navigate to and select the file I downloaded from the **AWS SSO SAML metadata** link in the AWS Single Sign-On settings and then, in the **Basic SAML Configuration** fly-out that opens, click **Save**. At this point, if I already have a user account in AWS Single Sign-On with a username that matches to a user in Azure AD, I can click the **Test** button to verify my setup.



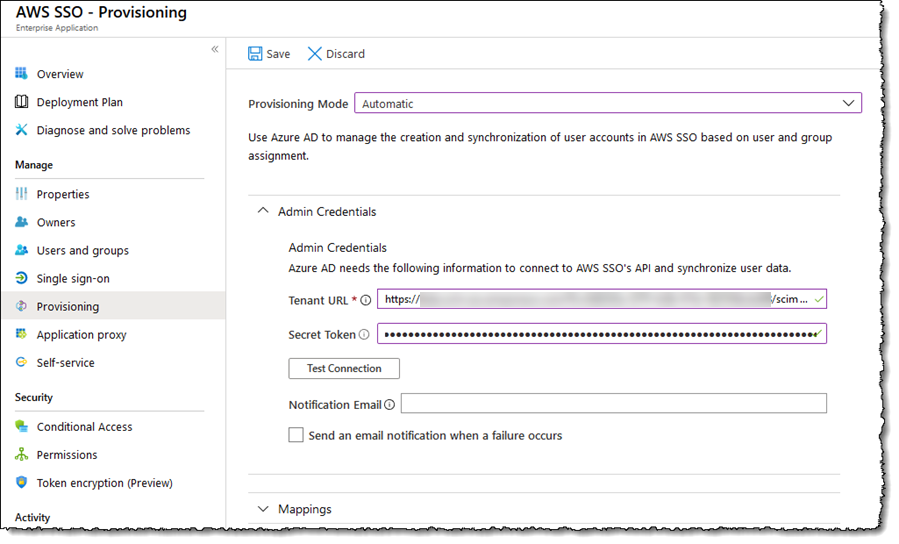
**Enabling Automatic Provisioning**  
Now that my Azure AD is configured for single sign-on for my users to connect using AWS Single Sign-On I’m going to enable automatic provisioning of user accounts. As new accounts are added to my Azure AD, and assigned to the **AWS SSO** application I created, when those users sign into [AWS](https://aws.amazon.com/), a corresponding AWS Single Sign-On user will be created automatically. As an administrator, I do not need to do any work to configure a corresponding account in [AWS](https://aws.amazon.com/) to map to the Azure AD user.

From the [AWS Single Sign-On Console](https://console.aws.amazon.com/singlesignon) I navigate to **Settings** and then click the **Enable identity synchronization** link. This opens a dialog containing the values for the **SCIM endpoint** and an OAuth bearer **Access token** (hidden by default). I need both of these values to use in the Azure AD application settings so either make note of both values, or use multiple browser tabs and copy/paste as you go.



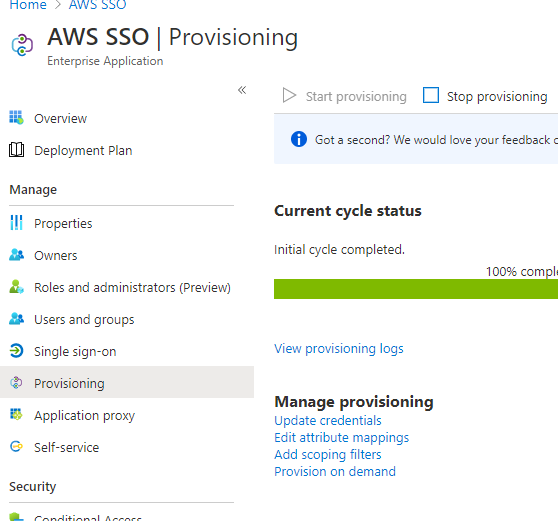


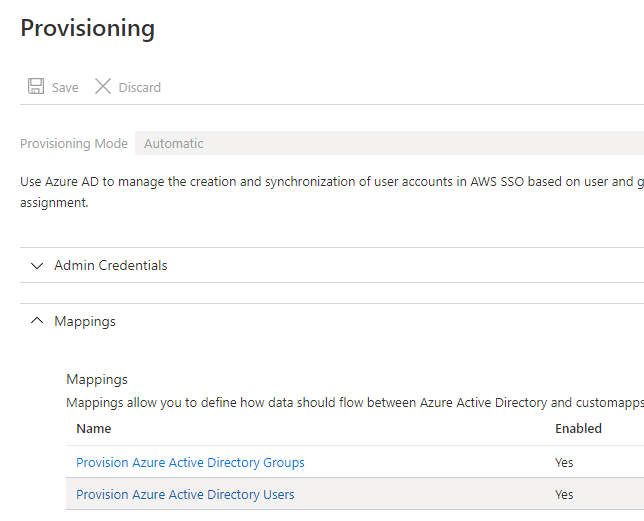
Switching back to my Azure AD browser tab and the **AWS SSO** application, I click **Provisioning** in the navigation panel and set **Provisioning Mode** to **Automatic**. This triggers display of the fields I need to complete with values from the dialog in the [AWS Single Sign-On Console](https://console.aws.amazon.com/singlesignon). First, I paste the **SCIM endpoint** value into the **Tenant URL** field. Then I paste the **Access token** into the **Secret Token** field.

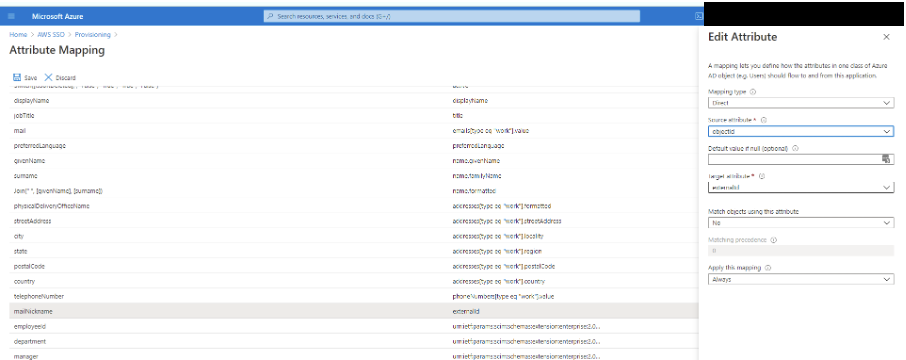


I complete the settings by entering a value for **Notification Email**, and opt in to receive an email when provisioning failures occur, then click the **Test Connection** button to verify everything is working as expected. Assuming everything is good, I click **Save** in the page toolbar and then just a couple of small steps remain to configure mapping of attributes and I am done!

Expanding **Mappings**, I click the **Synchronize Azure Active Directory Users to customappsso** link (your link may read ‘**…to AWS SSO**‘). That takes me to a page where I control attribute mappings. In that section I deleted the **facsimileTelephoneNumber** and **mobile** attributes as I won’t be using them, and I click on and edit the **mailNickname** attribute, changing the **Source attribute** to be **objectId**. Also change username to displayName. Clicking **Save** returns me to the **Settings** screen, where I then turn on synchronization by clicking **On** in **Provisioning Status**, and clicking **Save** one more time.





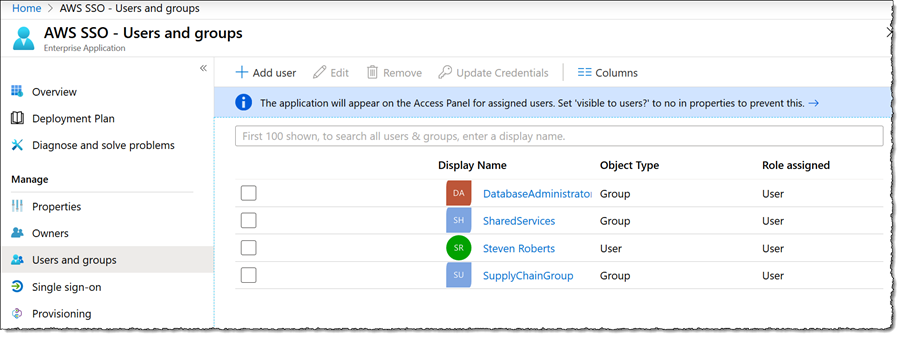


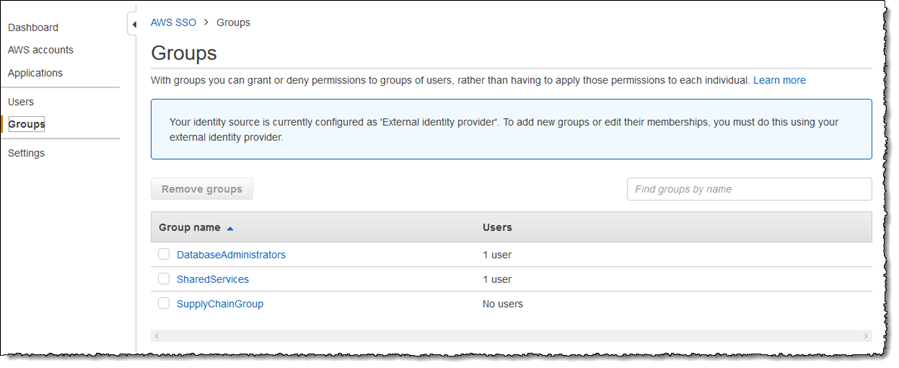
With provisioning enabled, my final task is to assign the users and groups that I want to synchronize from Azure AD to AWS SSO. To do this, I click **Users and groups** in the navigation panel. I click the **Add user** button, and click on the names of the users and/or groups that I want to synchronize from the right hand pane. I click **Select**, and then **Assign**. I can now see the users and groups assigned to the AWS SSO application. These users will be synchronized to AWS SSO, and the users will now see the AWS SSO application appear in their Azure My Apps portal.

Note that the first sync will take longer than subsequent ones, which happen around every 40 minutes. To check progress I can either view the **Synchronization Details** or the **Audit Logs** in Azure AD, or in the [AWS Single Sign-On Console](https://console.aws.amazon.com/singlesignon) I can select **Users** from the navigation panel.

**I Configured Single Sign-On, Now What?**  
Azure AD will now be my single source of truth for my user identities and their assignment into groups, and periodic synchronization will automatically create corresponding user identities in AWS Single Sign-On, enabling my users to sign into their AWS accounts and applications with their Azure AD credentials and experience, and not have to remember an additional username and password. However, as things stand my users will only have access to sign in. To manage permissions in terms of what they can access once signed into AWS, I need to set up permissions in AWS Single Sign-On which I do using the [AWS Single Sign-On Console](https://console.aws.amazon.com/singlesignon). AWS Single Sign-On uses a concept of **permission sets** for assignments. A permission set is essentially a standard role definition to which I attach [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/) policies. Once I define a permission set, I then assign a group, or user, to the permission set in a specified account. AWS Single Sign-On then creates the underlying [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/) role in the designated account, including all the right information that grants access to that group or user. You can read more about permissions sets in the AWS Single Sign-On [User Guide](https://docs.aws.amazon.com/singlesignon/latest/userguide/permissionsets.html).

In the screenshots below, you can see the effect of automatic synchronization. In Azure AD I have created three groups, and assigned my user account into two of them (for the sake of this blog post). Switching to the [AWS Single Sign-On Console](https://console.aws.amazon.com/singlesignon) once synchronization completes, I see the three groups have been automatically created, with my user account assigned into the ones I chose.





Using **Permission Sets**, available from the **AWS Accounts** and **Applications** links in the navigation panel, I can associate one or more access control policies (custom policies I have created, or [AWS Identity and Access Management (IAM)](https://aws.amazon.com/iam/) managed policies), to both groups and users, thus controlling permissions once users sign in. Sign in is accomplished using the familiar Azure AD experience, and users will be able to choose the account(s) and role(s) to assume. Sign in can also be done using the AWS console and CLI. Details on using single sign-on with the version 2 of the [AWS Command Line Interface (CLI)](https://aws.amazon.com/cli/) is detailed in [this blog post](https://aws.amazon.com/blogs/developer/aws-cli-v2-now-supports-aws-single-sign-on/).

Assign groups accordingly to aws accounts

Graphical user interface, text, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated