

# The Next Evolution in AWS Single Sign-On

by Steve Roberts | on 27 NOV 2019 | in [Launch](#), [News](#) | [Permalink](#) | [Share](#)

*Update Feb 23, 2021 – For the latest information on how to set up Azure AD with AWS SSO for automatic provisioning please see [our documentation here](#).*

Efficiently managing user identities at scale requires new solutions that connect the multiple identity sources that many organizations use today. Customers want to establish a single identity and access strategy across their own apps, 3rd party apps (SaaS), and [AWS](#) cloud environments.

Today we announced the next evolution of [AWS Single Sign-On](#), enabling enterprises that use Azure AD to leverage their existing identity store with AWS Single Sign-On. Additionally, automatic synchronization of user identities, and groups, from Azure AD is also supported. Users can now sign into the multiple accounts and applications that make up their AWS environments using their existing Azure AD identity – no need to remember additional usernames and passwords – and they will use the sign-in experience they are familiar with. Additionally, administrators can also focus on managing a single source of truth for user identities in Azure AD while having the convenience of configuring access to all AWS accounts and apps centrally.

Let's imagine that I am an administrator for an enterprise that already uses Azure AD for user identities and now wants to enable simple and easy use of our AWS environments for my users using their existing identities. I do not want to duplicate my Azure AD group and user membership setup by hand in AWS Single Sign-On, and maintain two identity systems, so I am also going to enable automatic synchronization. My users will sign in to the AWS environments using the experience they are already familiar with in Azure AD. You can read more about enabling single sign-on to applications in Azure AD [here](#). To learn more about managing permissions to [AWS](#) accounts and applications, refer to the [AWS Single Sign-On User Guide](#).

## Connecting Azure AD as an Identity Source for AWS Single Sign-On

My first step is to connect Azure AD with AWS Single Sign-On. First, I sign into the Azure Portal for my account and navigate to the **Azure Active Directory** dashboard. From the left-hand navigation panel I then select **Enterprise Applications**.

Microsoft Azure

Home > Default Directory > Enterprise applications - All applications

## Enterprise applications - All applications

Default Directory - Azure Active Directory

+ New application | Columns

Application Type: Enterprise Applications | Applications status: Any | Application visibility: Any | Apply | Reset

First 50 shown, to search all of your applications, enter a display name or the application ID.

Name	Homepage URL	Object ID
Office 365 Exchange Online	http://office.microsoft.com/outlook/	65822608-08
Office 365 Management APIs		9a595ea8-9d
Office 365 SharePoint Online	http://office.microsoft.com/sharepoint/	adf0095e-d7
Outlook Groups		039b2b92-2
Skype for Business Online		173c85d7-a

Next, I click **+ New application**, and select **Non-gallery application**. In the **Add your own application** page that opens, I enter **AWS SSO** in the **Name** field – you can use whatever name you like – and click **Add**. After a few seconds my new application will be created and I am redirected to the settings overview for the application.

Microsoft Azure

Home > Default Directory > Enterprise applications - All applications > Categories > Add an application > AWS SSO - Overview

## AWS SSO - Overview

Enterprise Application

Overview

Deployment Plan

Diagnose and solve problems

Manage

- Properties
- Owners
- Users and groups
- Single sign-on
- Provisioning
- Application proxy
- Self-service

Security

- Conditional Access
- Permissions
- Token encryption (Preview)

Activity

### Properties

Name: AWS SSO

Application ID: [Redacted]

Object ID: [Redacted]

### Getting Started

- 1. Assign users and groups**  
Provide specific users and groups access to the applications  
[Assign users and groups](#)
- 2. Set up single sign on**  
Enable users to sign into their application using their Azure AD credentials  
[Get started](#)
- 3. Provisioning**  
Automatically provision users and groups  
[Get started](#)
- 4. Conditional Access**  
Secure access to this application with a customizable access policy.  
[Create a policy](#)
- 5. Self service**  
Enable users to request access to the application using their Azure AD credentials

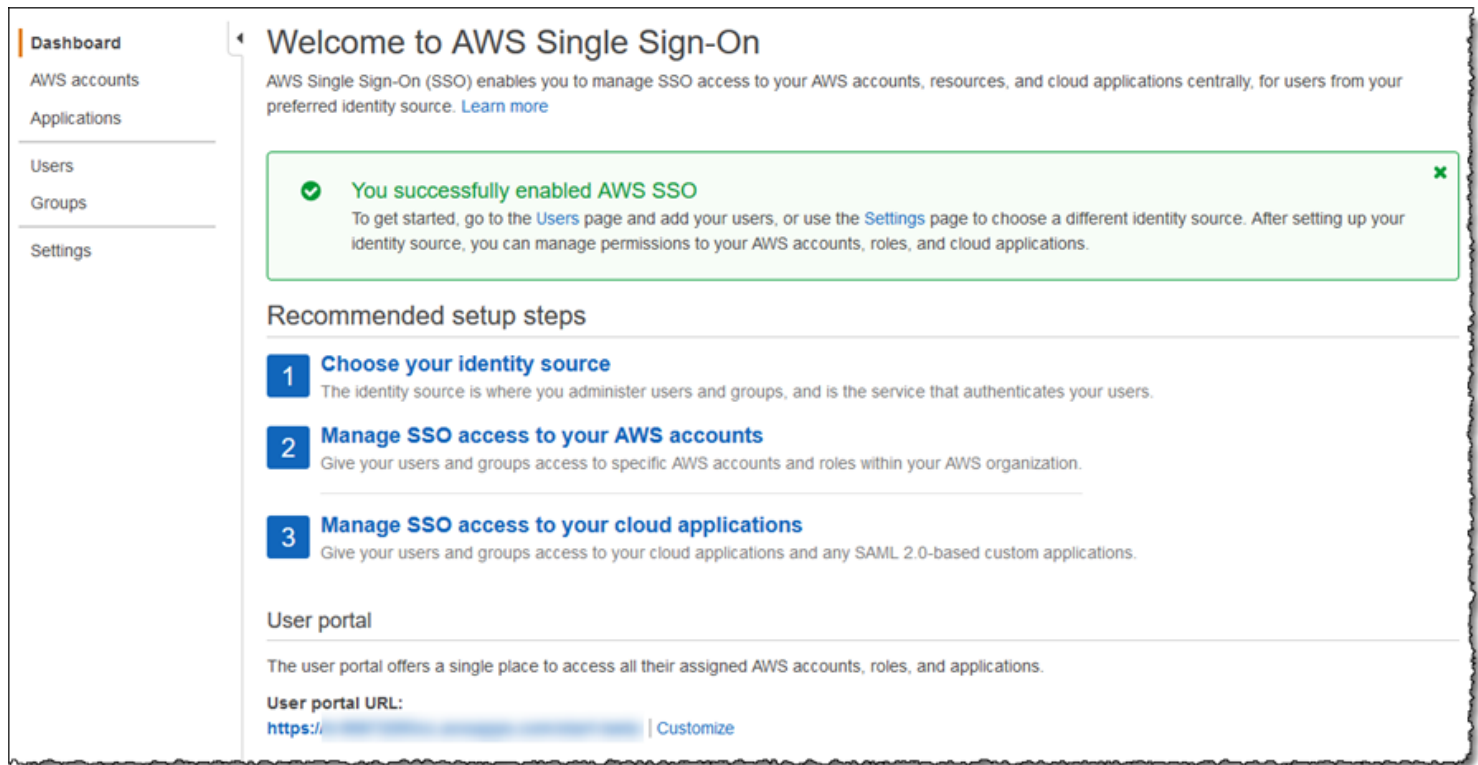
This is where I will configure the settings to enable single sign-on and exchange federation metadata between Azure AD and AWS Single Sign-On. I select **Single sign-on** from the navigation panel and then click the **SAML** option. From the settings page that then opens, I need to download the **Federation 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.

The screenshot shows the AWS SSO console interface. At the top, there's a table with columns 'Name' and 'Value'. The first row shows 'Unique User Identifier' with the value 'user:userprincipalname'. Below this is a section labeled '3 SAML Signing Certificate' with a pencil icon. It contains the following details: Status is 'Active'; Thumbprint is '4[redacted]5'; Expiration is '11/15/2022, 9:04:38 AM'; Notification Email is '[redacted].com'; App Federation Metadata Url is 'https://login.microsoftonline.com/aefbe204-353d- ...' with a download icon. Below these are three 'Download' links for 'Certificate (Base64)', 'Certificate (Raw)', and 'Federation Metadata XML'. A yellow arrow points to the 'Federation Metadata XML' download link. Below this is a section labeled '4 Set up AWS SSO' with the text 'You'll need to configure the application to link with Azure AD.'

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](#). 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.

The screenshot shows the AWS Single Sign-On (SSO) dashboard. At the top is a logo featuring a cloud, a person icon, and a key. Below the logo is the heading 'AWS Single Sign-On (SSO)'. The main text reads: 'AWS Single Sign-On is a cloud service that makes it easy to manage SSO access to multiple AWS accounts and business applications.' Below this is a white button with the text 'Enable AWS SSO'. At the bottom, there's a note with an information icon: 'When you enable AWS SSO, you allow it to create IAM roles for each AWS account in your AWS organization. [Learn more](#)'. The footer contains links for 'Getting Started Guide' and 'AWS SSO Prerequisites'.



The screenshot shows the AWS Single Sign-On dashboard. On the left is a navigation panel with links: Dashboard, AWS accounts, Applications, Users, Groups, and Settings. The main content area is titled 'Welcome to AWS Single Sign-On' and includes a brief description of SSO. A green success message states 'You successfully enabled AWS SSO' with instructions to go to the Users or Settings page. Below this is a 'Recommended setup steps' section with three numbered steps: 1. Choose your identity source, 2. Manage SSO access to your AWS accounts, and 3. Manage SSO access to your cloud applications. At the bottom is a 'User portal' section with a URL and a 'Customize' link.

**Dashboard**

- AWS accounts
- Applications
- Users
- Groups
- Settings

## Welcome to AWS Single Sign-On

AWS Single Sign-On (SSO) enables you to manage SSO access to your AWS accounts, resources, and cloud applications centrally, for users from your preferred identity source. [Learn more](#)

✓ **You successfully enabled AWS SSO**

To get started, go to the [Users](#) page and add your users, or use the [Settings](#) page to choose a different identity source. After setting up your identity source, you can manage permissions to your AWS accounts, roles, and cloud applications.

### Recommended setup steps

- 1 Choose your identity source**  
The identity source is where you administer users and groups, and is the service that authenticates your users.
- 2 Manage SSO access to your AWS accounts**  
Give your users and groups access to specific AWS accounts and roles within your AWS organization.
- 3 Manage SSO access to your cloud applications**  
Give your users and groups access to your cloud applications and any SAML 2.0-based custom applications.

### User portal

The user portal offers a single place to access all their assigned AWS accounts, roles, and applications.

**User portal URL:**  
<https://...> | [Customize](#)

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.

## Change identity source

1

Choose directory

2

Review

### Choose where your identities are sourced

Your identity source is the place where you administer and authenticate identities. You use AWS SSO to manage permissions for identities from your identity source to access AWS accounts, roles, and applications. [Learn more](#)

☐ **AWS SSO directory**

You will administer all users, groups, credentials, and multi-factor authentication assignments in AWS SSO. Users sign in through the AWS SSO user portal.

☐ **AWS Managed Microsoft AD or AD Connector directory**

You will administer all users, groups, and credentials in AWS Managed Microsoft AD, or you can connect AWS SSO to your existing Active Directory using AWS Managed Microsoft AD or AD Connector. Users sign in through the AWS user portal.

☒ **External identity provider**

You will administer all users, groups, credentials, and multi-factor authentication in an external identity provider (IdP). Users sign in through your IdP sign-in page to access the AWS SSO user portal, assigned accounts, roles, and applications.

### Configure external identity provider

AWS SSO works as a SAML 2.0 compliant service provider to your external identity provider (IdP). To configure your IdP as your AWS SSO identity source, you must establish a SAML trust relationship by exchanging meta data between your IdP and AWS SSO. While AWS SSO will use your IdP to authenticate users, the users must first be provisioned into AWS SSO before you can assign permissions to AWS accounts and resources. You can either provision users manually from the Users page, or by using the automatic provisioning option in the Settings page after you complete this wizard. [Learn more](#)

#### Service provider metadata

Your IdP requires the following AWS SSO certificate and metadata details to trust AWS SSO as a service provider. You may copy and paste, or type this information into your IdP's service provider configuration interface, or you may download the AWS SSO metadata file and upload it into your IdP.

**AWS SSO SAML metadata**

[Download metadata file](#)

[Show individual metadata values](#)

#### Identity provider metadata

AWS requires specific metadata provided by your IdP to establish trust. You may copy and paste from your IdP, type the metadata in manually, or upload a metadata exchange file that you download from your IdP.

**IdP SAML metadata\***

AWS SSO.xml

[Browse...](#)

[If you don't have a metadata file, you can manually type your metadata values](#)

I 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.

Customers with a relatively small number of users may prefer to continue maintaining the user accounts in AWS Single Sign-On rather than rely on automatic provisioning and can stop here, as it is possible to use just the sign-in aspect. We do however recommend enabling automatic provisioning for convenience. New users you add to an Azure AD group automatically get access with no additional action needed, making it convenient for administration and

productive for the end user. Users who get removed from a group in Azure AD will automatically lose access to associated permissions in AWS Single Sign-On, a security benefit.

### **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](#), 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](#) to map to the Azure AD user.

From the [AWS Single Sign-On Console](#) 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.

Inbound automatic provisioning

✔

Automatic provisioning has been successfully enabled in AWS SSO.  
Next you'll need to provide the following information to configure your external identity provider and create the trust relationship.

Download or copy the access token as this is the only time it will be shown

You cannot recover it later. However, you can generate new tokens at any time. [Learn more](#)

SCIM endpoint

https://[REDACTED]/scim/v2/

Access token

6f[REDACTED]mr

[Hide token](#)

Close

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](#). First, I paste the **SCIM endpoint** value into the **Tenant URL** field. Then I paste the **Access token** into the **Secret Token** field.



**AWS SSO - Provisioning**  
Enterprise Application

Save Discard

Provisioning Mode: Automatic

Use Azure AD to manage the creation and synchronization of user accounts in AWS SSO based on user and group assignment.

**Admin Credentials**

Admin Credentials  
Azure AD needs the following information to connect to AWS SSO's API and synchronize user data.

Tenant URL \*  ✓

Secret Token  ✓

Test Connection

Notification Email

☐ Send an email notification when a failure occurs

**Mappings**

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**. 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.

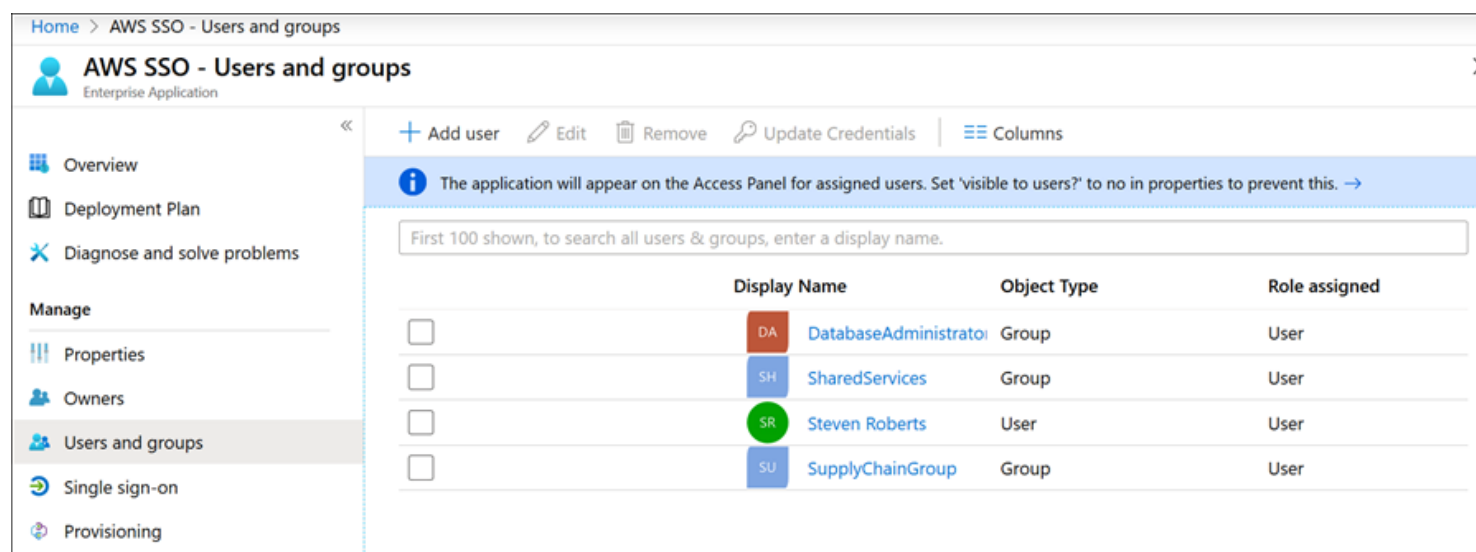
Note that the first sync will take longer than subsequent ones, which happen around every 40 minutes. To check progress I can either view the **Synchronization Details** or the **Audit Logs** in Azure AD, or in the [AWS Single Sign-On Console](#) I can select **Users** from the navigation panel.

### 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

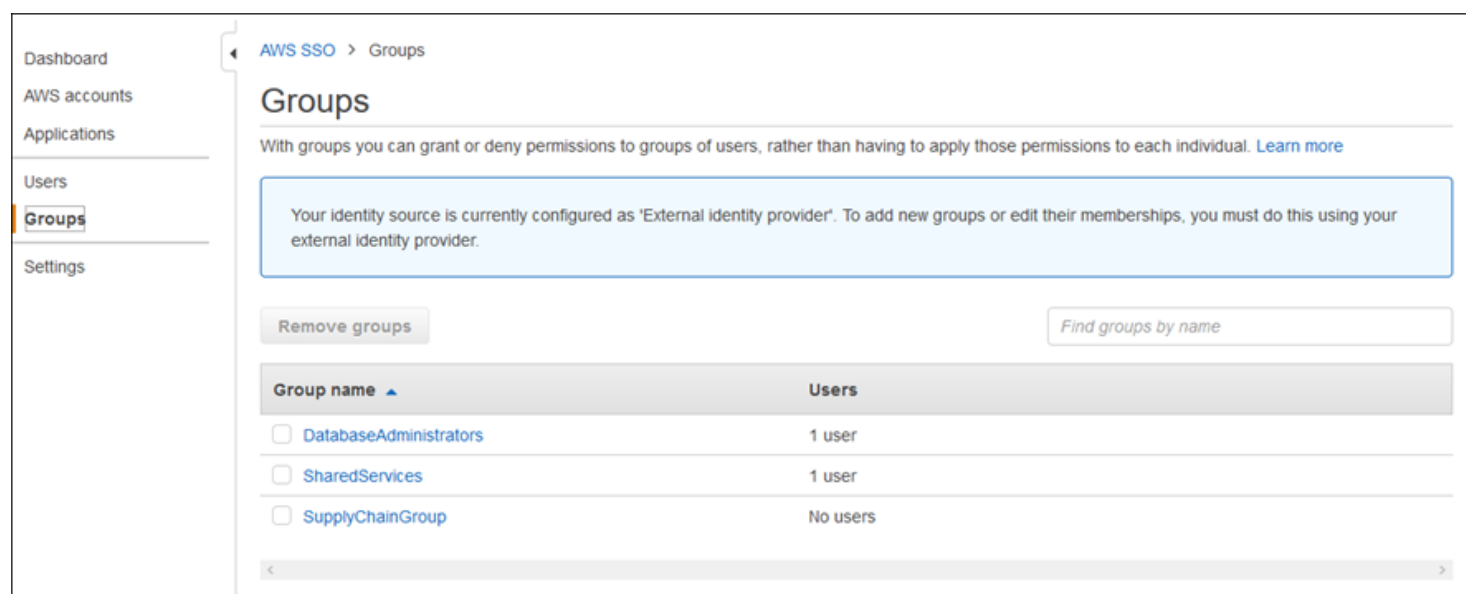
sign into their AWS accounts and applications with their Azure AD credentials and experience, and not have to remember an additional username and password. However, as things stand my users will only have access to sign in. To manage permissions in terms of what they can access once signed into AWS, I need to set up permissions in AWS Single Sign-On which I do using the [AWS Single Sign-On Console](#). 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 \(IAM\)](#) 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 \(IAM\)](#) 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](#).

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](#) once synchronization completes, I see the three groups have been automatically created, with my user account assigned into the ones I chose.



The screenshot shows the AWS SSO - Users and groups console. The left sidebar contains navigation links: Overview, Deployment Plan, Diagnose and solve problems, Manage, Properties, Owners, Users and groups (selected), Single sign-on, and Provisioning. The main content area displays a table of users and groups. At the top, there are buttons for Add user, Edit, Remove, Update Credentials, and Columns. A message states: "The application will appear on the Access Panel for assigned users. Set 'visible to users?' to no in properties to prevent this. →". Below this is a search bar with the text "First 100 shown, to search all users & groups, enter a display name." The table has columns for Display Name, Object Type, and Role assigned. The data rows are:

	Display Name	Object Type	Role assigned
<input type="checkbox"/>	DA DatabaseAdministrator	Group	User
<input type="checkbox"/>	SH SharedServices	Group	User
<input type="checkbox"/>	SR Steven Roberts	User	User
<input type="checkbox"/>	SU SupplyChainGroup	Group	User



The screenshot shows the AWS SSO - Groups console. The left sidebar contains navigation links: Dashboard, AWS accounts, Applications, Users, Groups (selected), and Settings. The main content area displays the Groups page. At the top, there is a message: "Your identity source is currently configured as 'External identity provider'. To add new groups or edit their memberships, you must do this using your external identity provider." Below this is a search bar with the text "Find groups by name". The table has columns for Group name and Users. The data rows are:

Group name	Users
<input type="checkbox"/> DatabaseAdministrators	1 user
<input type="checkbox"/> SharedServices	1 user
<input type="checkbox"/> SupplyChainGroup	No users



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 \(IAM\)](#) 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 \(CLI\)](#) is detailed in [this blog post](#).

In this post I showed how you can take advantage of the new [AWS Single Sign-On](#) capabilities to link Azure AD user identity into [AWS](#) accounts and applications centrally for single sign-on, and make use of the new automatic provisioning support to reduce complexity when managing and using identities. Administrators can now use a single source of truth for managing their users, and users no longer need to manage an additional identity and password to sign into their AWS accounts and applications.

This next evolution is now available to all users in all AWS Single Sign-On supported regions. You can check the regional availability of [AWS Single Sign-On here](#).

— Steve