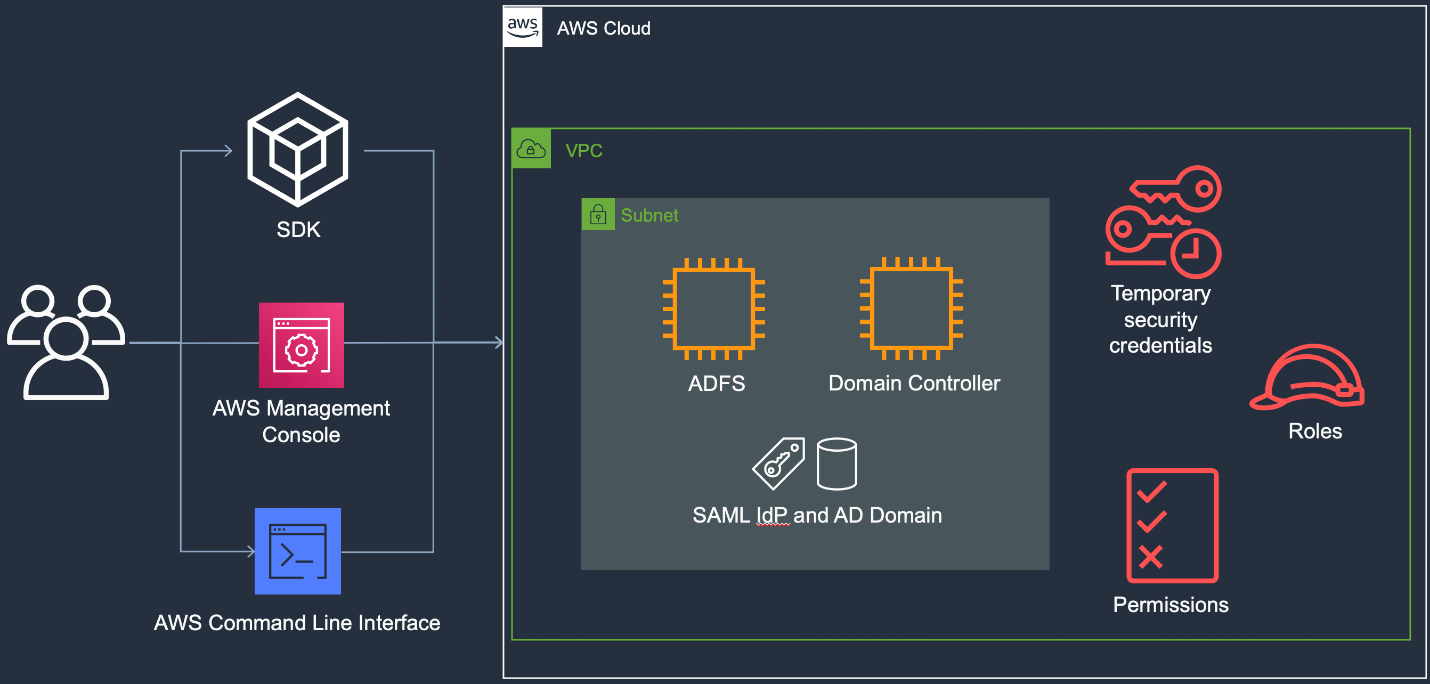
AWS Identity Federation Mastery - Microsoft Active Directory with SAML solution.

**AWS Service/Feature used during this project**

* AWS IAM [users & roles](https://docs.aws.amazon.com/IAM/latest/UserGuide/id.html)
* AWS IAM Microsoft ADFS
* EC2

Architecture

The following image provides a visual representation of what you are about to construct during this initial exercise.



Setup Instructions

**Important**

There are several important things to note as you progress through the exercise:

* The first EC2 instance, named DC, has been configured as a Domain Controller (DC) with Active Directory and DNS.
* The DNS domain for the AD domain is example.com.
* The NetBIOS name for the domain is EXAMPLE.
* The EXAMPLE\Administrator password will be set to the CloudFormation input parameter you specify below.
* The second EC2 instance, named adfsserver, has not been altered from the base AMI.
* This instance is not yet a member of the domain.
* The local Administrator password will be set to the CloudFormation input parameter you specify below.
* You will use two users throughout this lab. Their uids are bob and alice, and their password will be set to a random value exposed as an output variable below.
* You will use a service account for ADFS, the uid of this service account will be adfssvc and the password will be set to a random value exposed as an output variable below.

You will have a chance to review the outcome of these configurations in subsequent steps.

Provided is Adfs-setup.yml, a CloudFormation stack that automates many of the more trivial and time-consuming steps so that you can focus your time on the true learning objectives. To setup your environment deploy the adfs-setup CloudFormation and then move on to **Task 1**. You can work on **Task 1** while you wait for the CloudFormation stack to complete.

Task 1 Configuring federated access to the AWS CLI and SDKs

**Info**

Later in this initial exercise, you will be implementing federated API/CLI access. This enables your users (in this case bob and alice) to access your AWS environment using their AD Domain (example.com) credentials through the [AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html) or AWS SDKs. **While you wait for the CloudFormation stack above to complete**, let's start this process by implementing the specific configurations and utilities that are needed for this capability.

Configure a minimal AWS credentials file (You should have this already if not complete this)

From the command line run:

aws configure

The AWS CLI prompts you for four pieces of information (access key, secret access key, Default region name, and output format). You only need to enter the **default region name** at this point. By default we are using us-east-1 in this workshop. You can just hit **enter** for the other entries. The AWS CLI stores this information in a profile (a collection of settings) named default in the credentials file:

Linux/OSX: ~/.aws/credentials

Windows: C:\Users\USERNAME\.aws\credentials

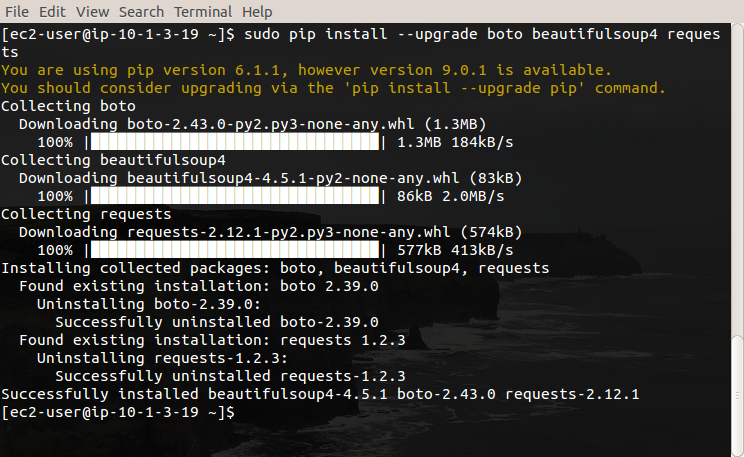
**Note**

An AWS access key pair is not configured in the above structure because the initial AWS STS call is authenticated by the SAML assertion returned by the trusted IdP. All subsequent API/CLI calls are authenticated by the key pair contained within the returned AWS STS token.

Install three Python modules: boto3, beautifulsoup4, and requests

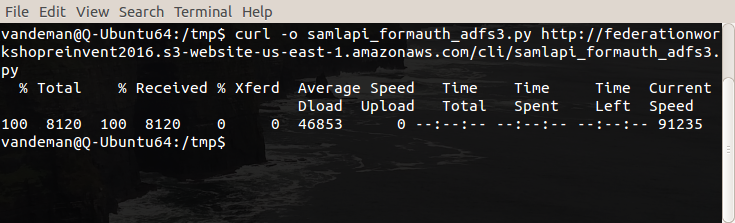
From the command line run:

pip install --upgrade boto3 beautifulsoup4 requests



[Download](https://identity-federation.awssecworkshops.com/cli/samlapi_formauth_adfs3.py) the script that will be used for federated API/CLI access:

curl -o samlapi\_formauth\_adfs3.py https://identity-federation.awssecworkshops.com/cli/samlapi\_formauth\_adfs3.py



**Info**

The version of this script is unique for this workshop:

* The value of idpentryurl in the script has been set to the [IdP initiated login URL](https://idp1.example.com/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices) for AD FS that you will use in this exercise.
* The value of sslverification has been set to False because we will be utilizing a self-signed certificate.

Task 2 Review template outputs

After the CloudFormation stack creation has completed, view the Outputs tab

Determine the public IP addresses of the Domain Controller (DC) and AD FS instances, and the randomly generated password value for **bob**, **alice**, and **adfssvc**. Copy down these values as you will need them in the following steps.

Navigate to the EC2 Instance dashboard

Examine the newly created DC and AD FS instances. Wait for the green checkmarks under Status Checks for the two instances before proceeding to the next step.

Task 3 Configure users and groups in Active Directory

**Info**

Now that you have successfully deployed the CloudFormation stack, you need to configure some users and groups in Active Directory.

Login to your Domain Controller via RDP

Login using RDP to your newly provisioned Domain Controller (DC) instance using the public IP address that you noted in the previous step. Use the Example\Administrator credential. See [directions](http://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/connecting_to_windows_instance.html) for how to remotely connect to Windows according to your client platform.

**Note**

Recall that the password for **Example\Administrator** has been set to the value you provided in the AdminPassword CloudFormation input parameter.

If prompted to verify the RDP certificate, click **Continue**.

Configure Users/Groups in Active Directory

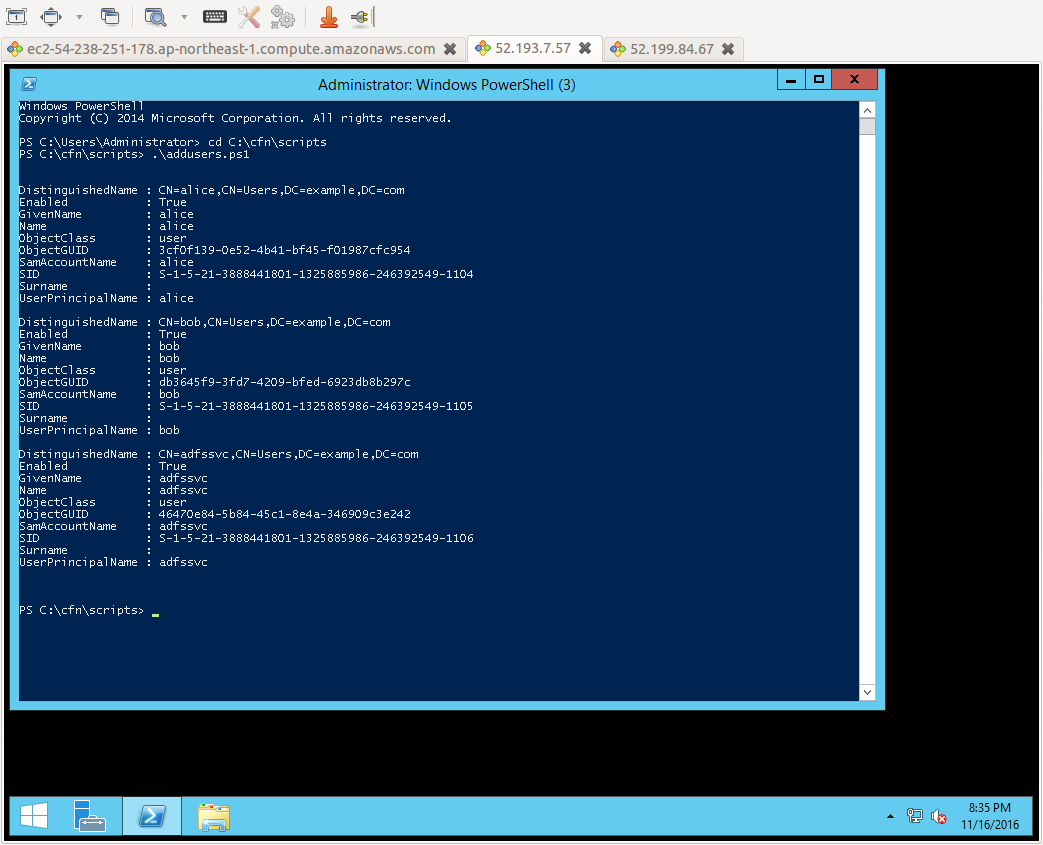
**Info**

First, we need to create two user accounts, **alice** and **bob**. We have created a script to quickly create these two user accounts for you. This script also creates the service account adfssvc.

Open a new PowerShell window and run the following commands:

cd C:\cfn\scripts

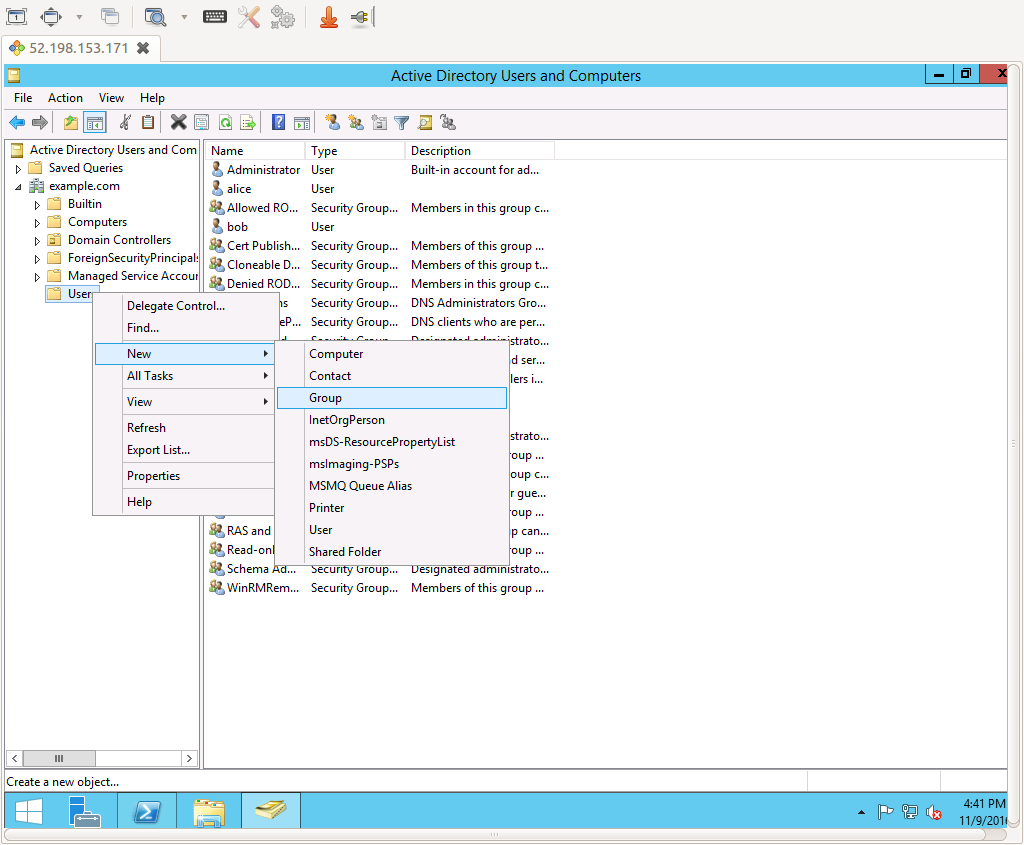
.\addusers.ps1



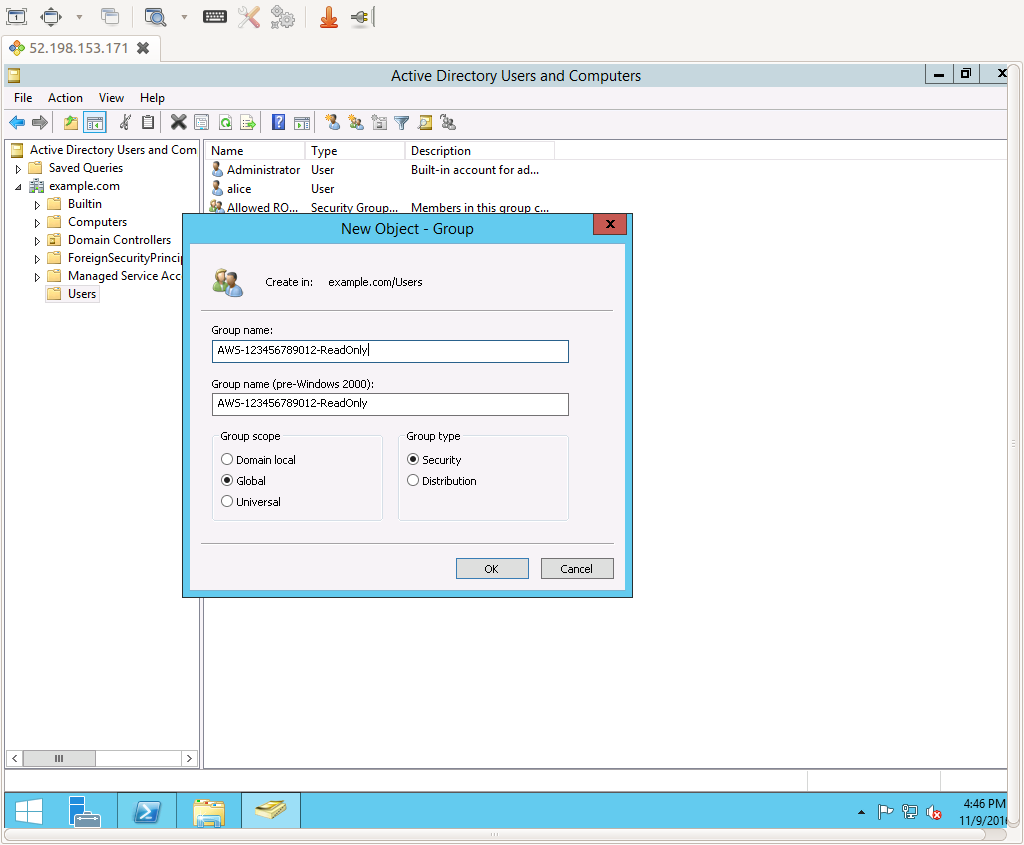
**Info**

Next, we need to create a set of groups using the naming convention of AWS-<Account Number>-<Role Name>. The following section describes this naming convention. As you will see later on, the membership of these groups will control which users can access which accounts and roles within your AWS environment.

Click the **Start Menu**, type user, and select **Active Directory Users and Computers**. Expand example.com, and then click **Users**. Right click, click **New**, and then click **Group** as shown in the following screenshot:



Create a new group for the role **ReadOnly** using the following naming convention: AWS-<Account Number>-ReadOnly. Replace <Account Number> your AWS account number (If you need help finding your account number, [see this documentation](http://docs.aws.amazon.com/general/latest/gr/acct-identifiers.html).) Then click OK to finish creating the group, as shown in the following screenshot:



Repeat the same process to create a second group for the **PowerUser** role: AWS-<Account Number>-PowerUser



**Info**

Understanding the AWS-<Account Number>-<Role Name> naming convention

In the preceding section, we utilized a naming convention of AWS-<Account Number>-<Role Name> for the groups stored in Active Directory. To understand why this was chosen, we must first consider several aspects of how AWS environments are commonly segregated:

* Most AWS customers use **multiple AWS accounts**. These accounts are organized using common dimensions (e.g. line of business, environment, application, etc) that are often combined. For example, one account might be used for Finance-Prod while another might be used for HR-NonProd.
* Most AWS customers apply permissions based on the **organizational role** of the user. For example, an organization might have roles such as 'Developer,''ProductionSupport,''Engineering,' or 'Security.'
* Most users have access only to a **subset** of their organization's AWS accounts. For example, if a developer is writing applications for the Finance department they will have access to the Finance-*set of accounts, but not to the HR-* set of accounts.
* To apply **separation of duties**, most users have different roles in different accounts. For example, an individual may have a relatively powerful role (e.g. PowerUser) in a non-production account where they perform development work, but a more constrained role (e.g. ReadOnly) in a production account where they mainly access log files for debugging purposes.

The AWS-<Account Number>-<Role Name> naming convention was specifically formulated to provide a **best practice pattern** for group naming that was able to account for all the aspects identified above. While the specific set of dimensions an organization may choose to use for account segmentation or the specific set of named roles an organization may define will certainly vary from one organization to the next, this standard works in all known cases. We highly encourage you to use this pattern within your organization, as it has been proven across a large number of customer implementations at scale.

Add users to the groups

**Info**

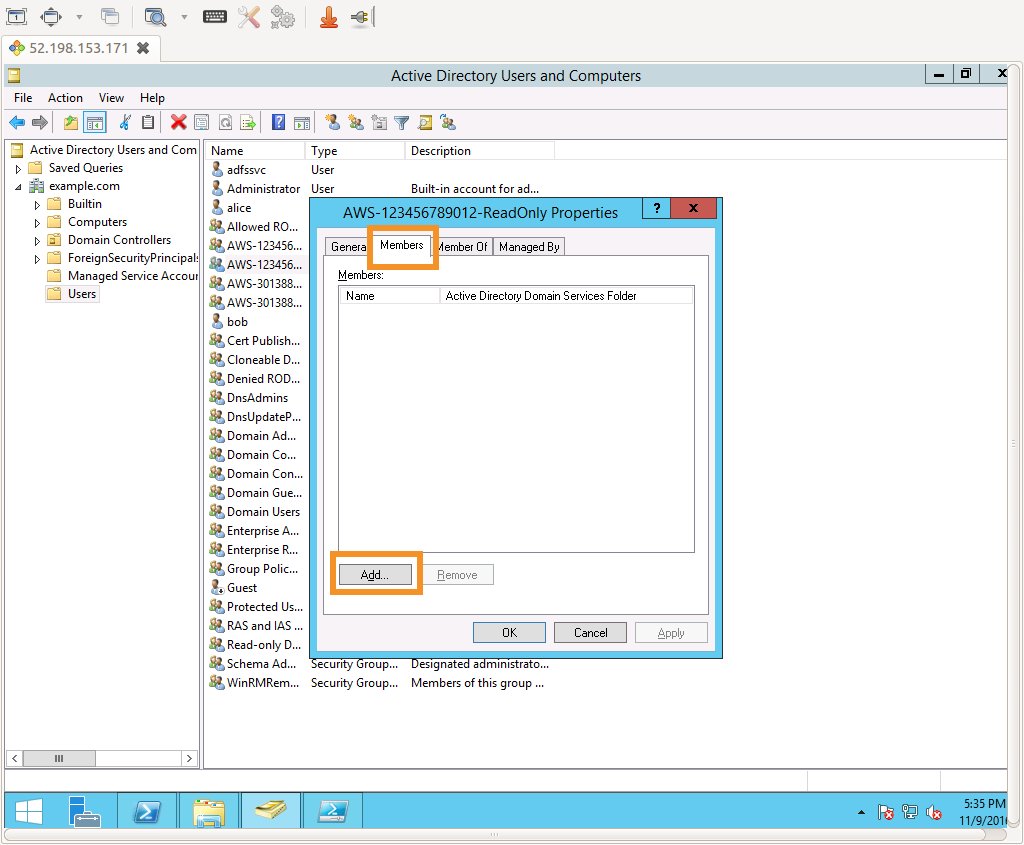
Next, we need to add our two users, bob and alice, to our newly created groups. In this exercise, you perform this directly through the Active Directory management interface. This method is intended to provide the "cooking show" version of this process. In your organization, the process for a user to be placed into one of the groups used to control AWS access should be managed through an entitlement portal, a managed workflow environment, or other trusted process.

**Important**

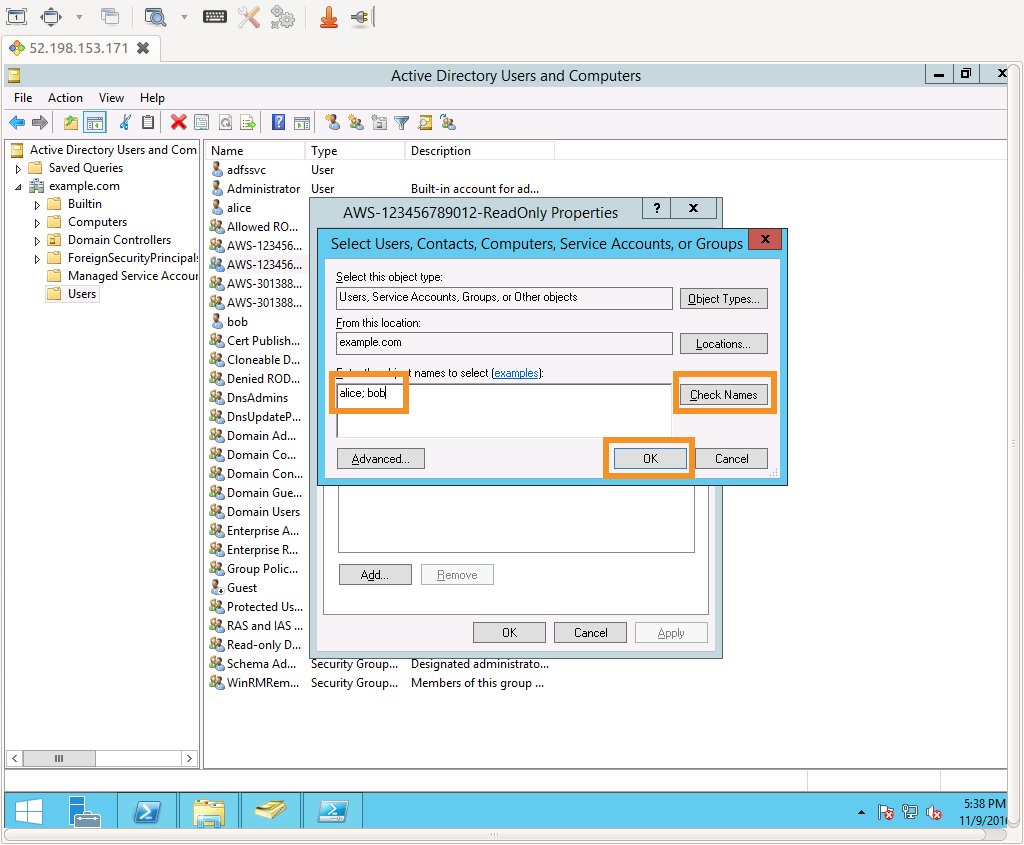
In a federated scenario the IdP is entirely responsible for authorizing which roles within AWS a given user can assume. Using the patterns and configurations shown here, the groups in Active Directory serve as the source of truth for these entitlements. If a user belongs to one of the specially named groups, they can access AWS within the confines of the corresponding AWS account and IAM role. As such, please ensure that as you adapt these learnings to your environment:

* You appropriately restrict the users and automated tools can modify these group memberships.
* You properly vet and approve users who are placed into these groups.
* You regularly baseline and audit these group memberships to ensure they contain the right users.

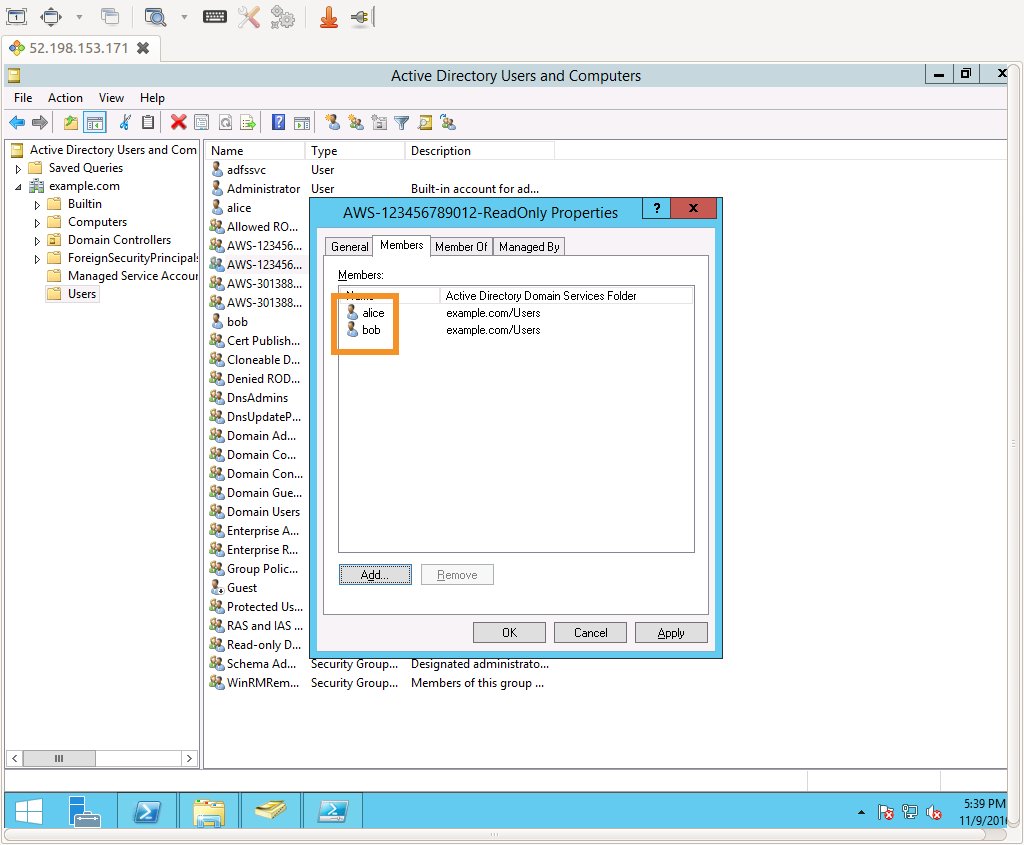
Click the AWS-<Account Number>-ReadOnly group you created above, and then click the **Members** tab, and finally click **Add**, as shown in the following screenshot:



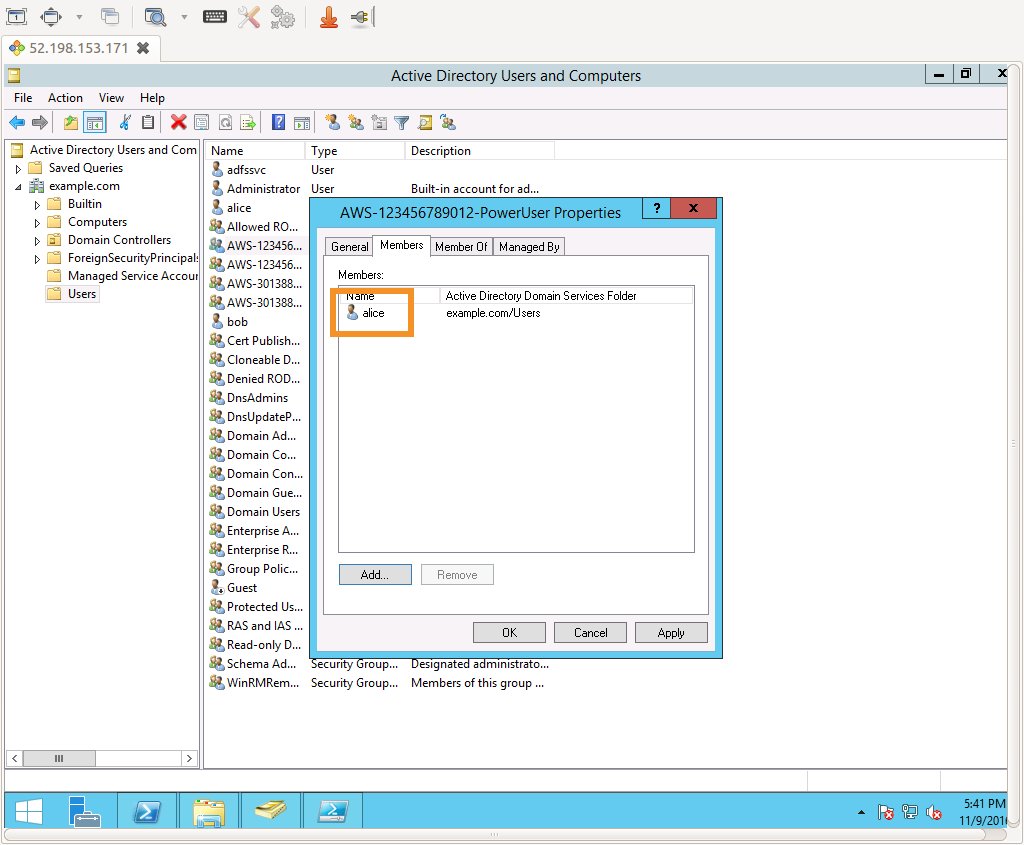
In the **object names** box, type **alice;bob**, and click **Check Names** and then **OK**, as shown in the following screenshot:



You will see alice and bob reflected as **group members**, as shown in the following screenshot.



Repeat the same process for the AWS-<Account Number>-PowerUser group. However, this time only add **alice** as a group member.



Task 4 Join the AD FS instance to the domain

**Info**

For our next step, we add the AD FS instance to the example.com Active Directory domain served by the domain controller instance. This will allow the AD FS service that will be configured in subsequent steps to use the user and group information you configured above for authentication and coarse authorization purposes.

Login to your AD FS instance via RDP

Login using **RDP** to your **AD FS instance** using the public IP address that you noted earlier (which you copied from the CloudFormation stack's Output section). Use the **local Administrator** credential (which you entered into the Parameters of the CloudFormation stack).

Update DNS configuration

**Info**

First, we need to configure the AD FS instance to use the domain controller instance for DNS. Start by determining the private IP address of the domain controller (DC) instance.

You can obtain the private IP address of the DC from the CloudFormation stack's Output section. To update the DNS configuration, open a new Windows PowerShell window, and enter the following command.

Set-DnsClientServerAddress -InterfaceAlias "Ethernet" -ServerAddresses <PrivateIPaddressofDC>

**Hint**

If you receive an error stating that the interface "Ethernet" cannot be found, use a value of "Ethernet 2" in the above command.

Now test DNS resolution by pinging the domain controller using the following command:

ping example.com

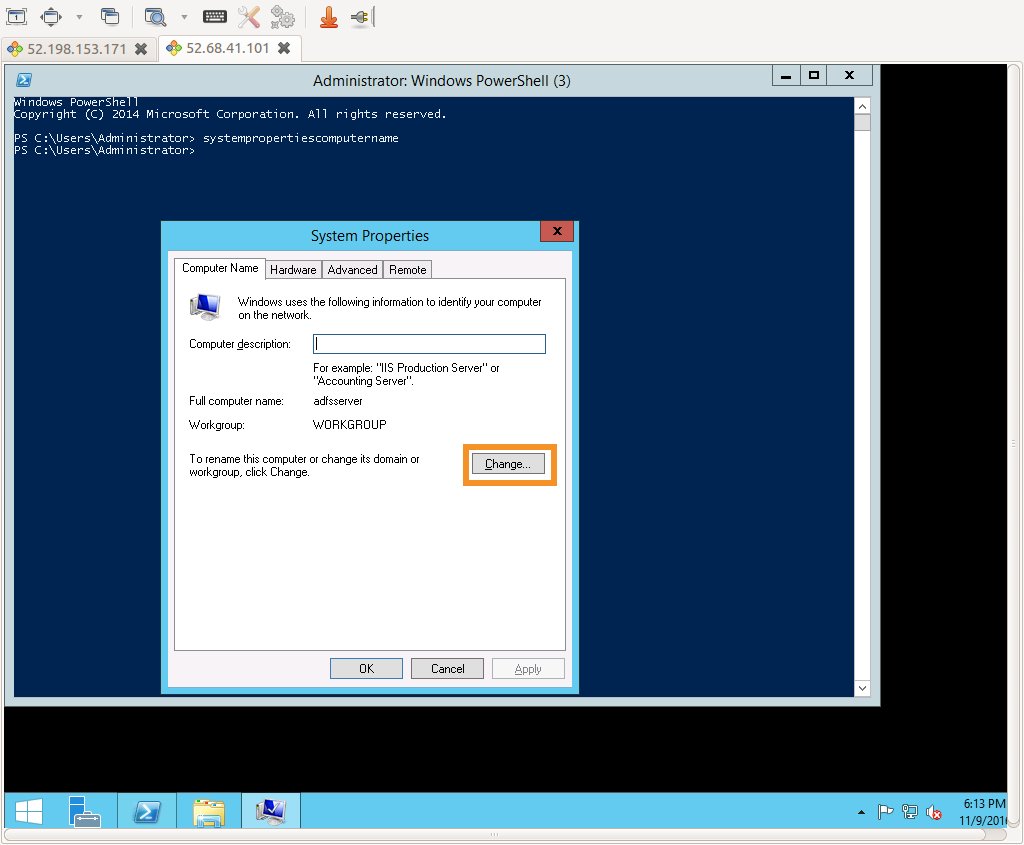
You should receive replies from the private IP address of the domain controller instance.

Join the AD FS instance to the example.com domain

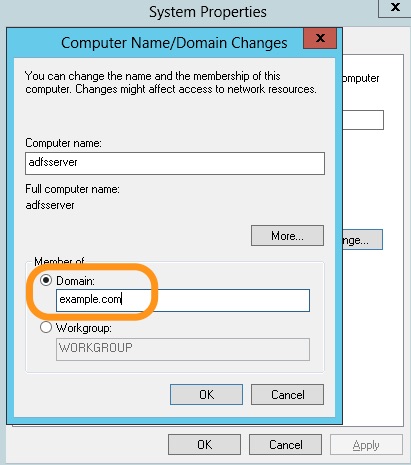
To complete the domain-joining activity, enter the following command into the open Windows PowerShell window:

systempropertiescomputername

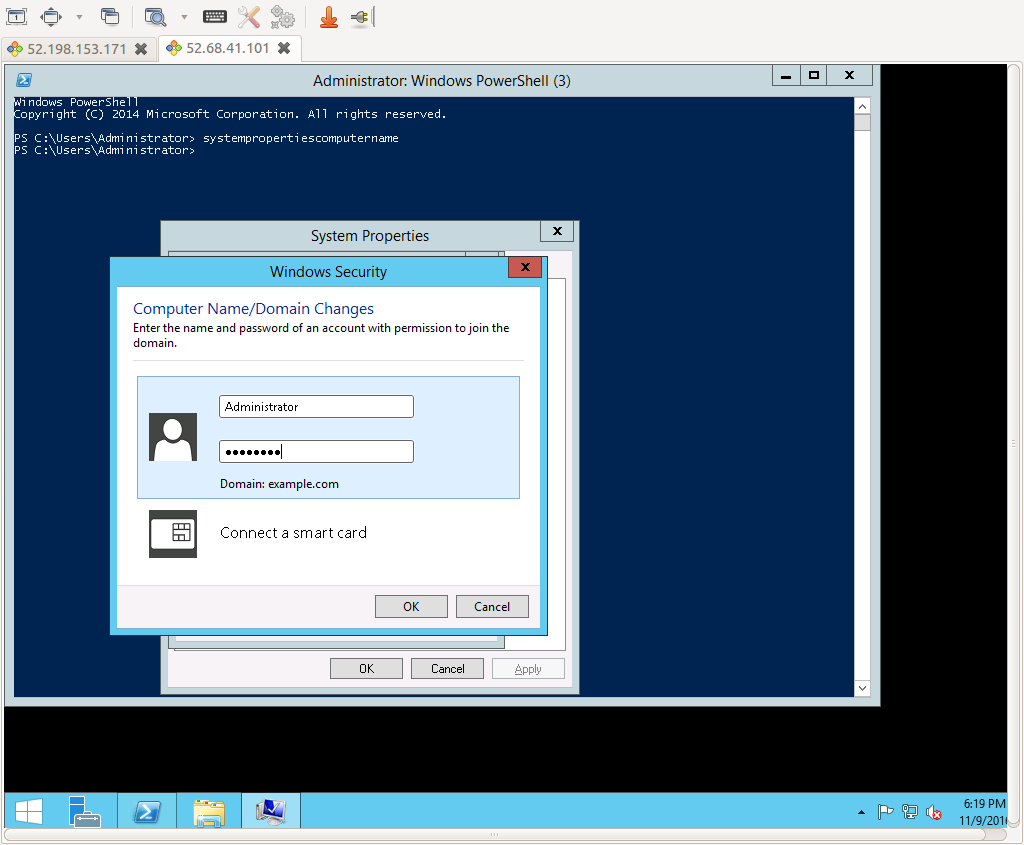
In the **System Properties** dialog, click **Change**.



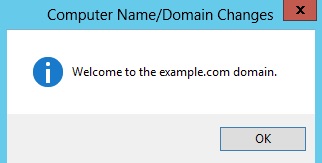
Under **Member of**, click **Domain**, type example.com, and click **OK**.



In the **Windows Security** dialog, authenticate with a user name **Administrator** and password you provided in the AdminPassword CloudFormation input parameter.



You should see the pop up window: "Welcome to the example.com domain."



Close the **System Properties** dialog, and then click **Restart now** to complete the domain-join process.

**Hint**

From this step onwards, log in to the AD FS Server as **Example\Administrator** with the password you provided in the AdminPassword of the Parameters of the CloudFormation stack.

Task 5 Install and Configure Active Directory Federation Services

**Info**

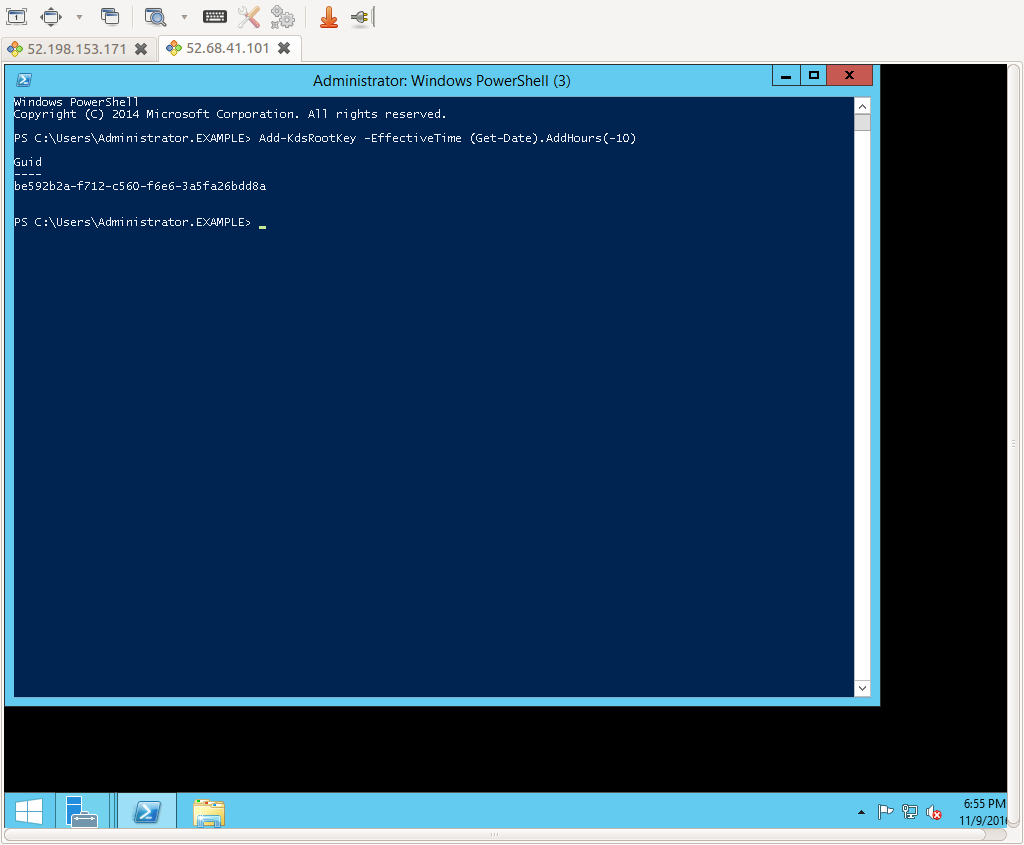
Now that you have joined the AD FS instance to the domain, you are ready to configure Active Directory Federation Services.

Once your instance has finished rebooting, reconnect using RDP, paying attention to log in with the domain credential **Example\Administrator** instead of the local account.

Set the Kds Root Key

To avoid a benign message during the following AD FS setup, open a Windows PowerShell window and enter the following command:

Add-KdsRootKey -EffectiveTime (Get-Date).AddHours(-10)

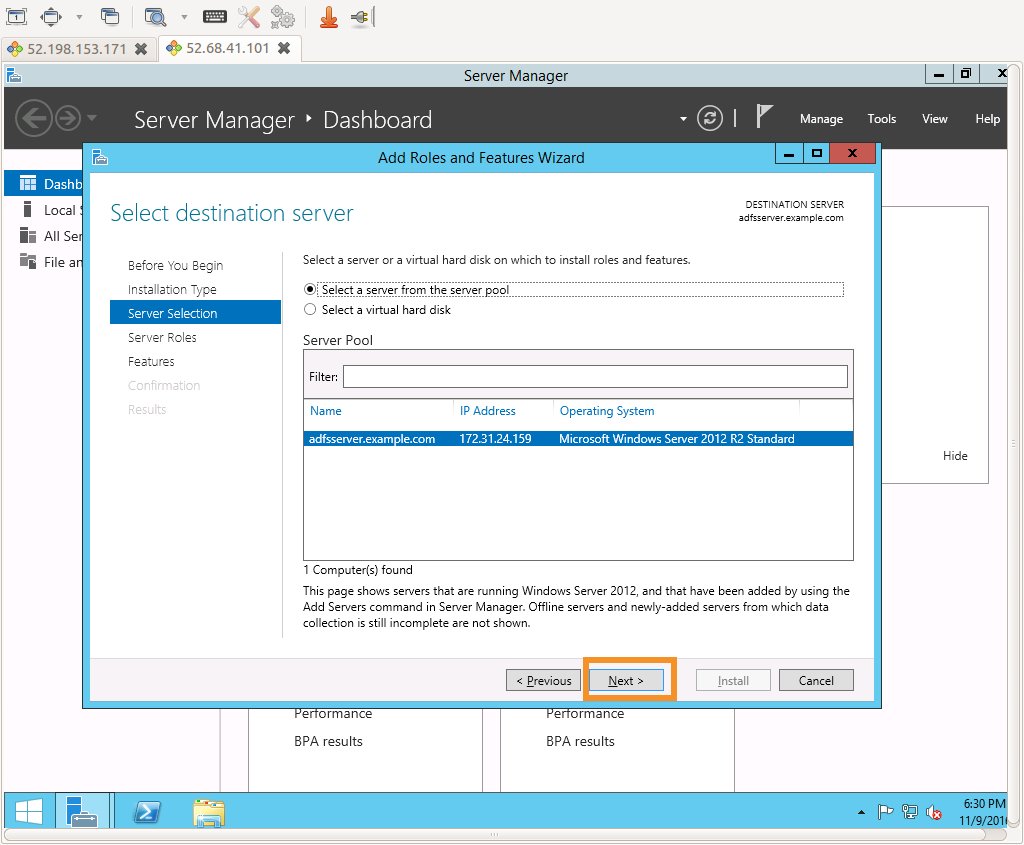


Add Server Role

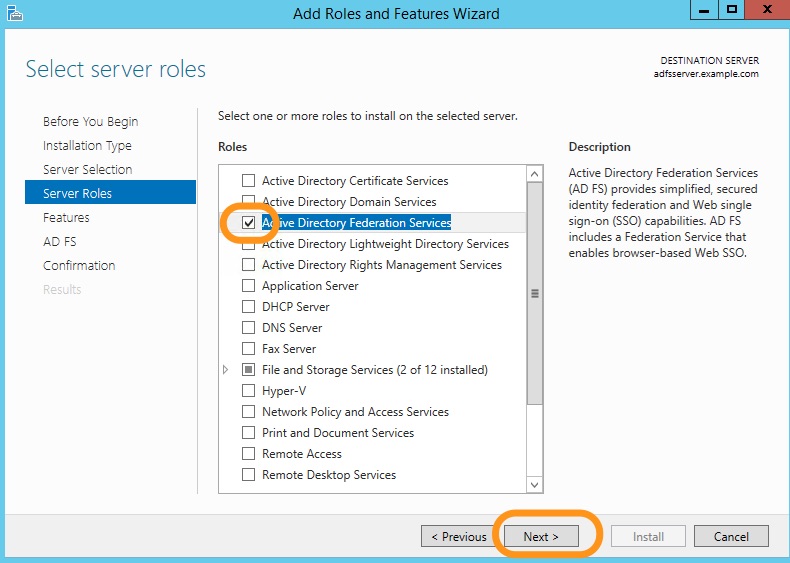
Open **Server Manager**, and click **Add roles and features**



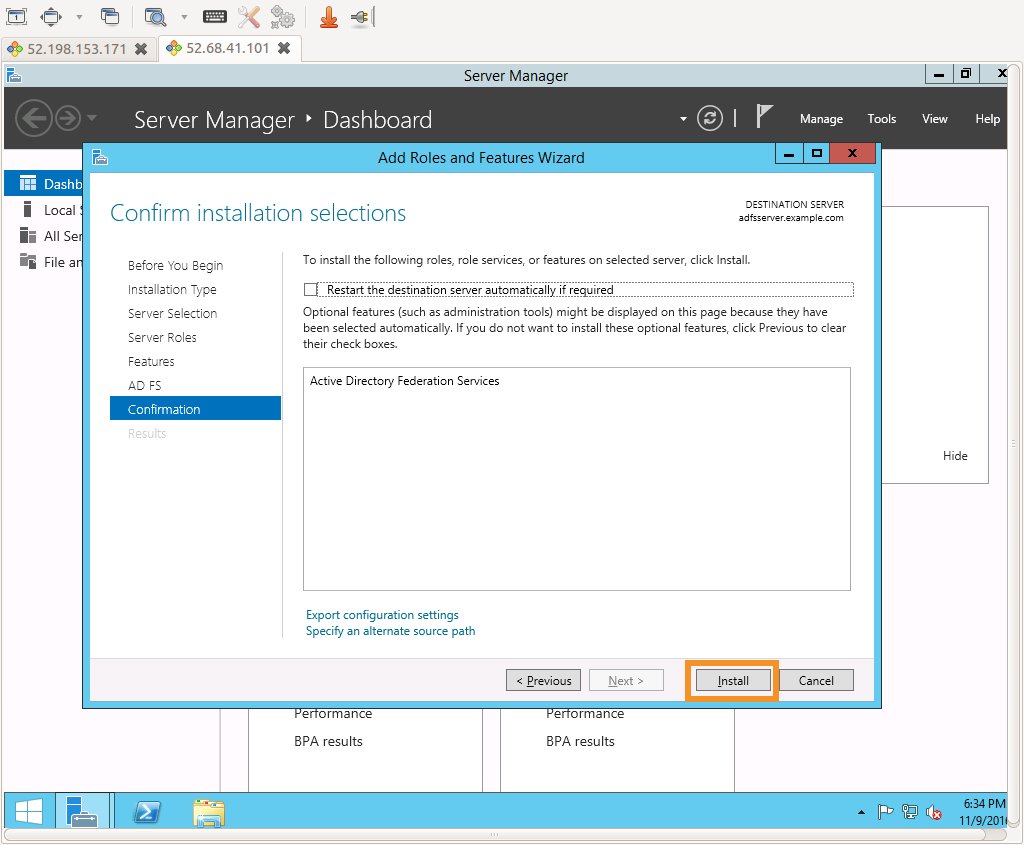
Accept the defaults for **Before You Begin**, **Installation Type**, and **Server Selection** by clicking **Next** three times.



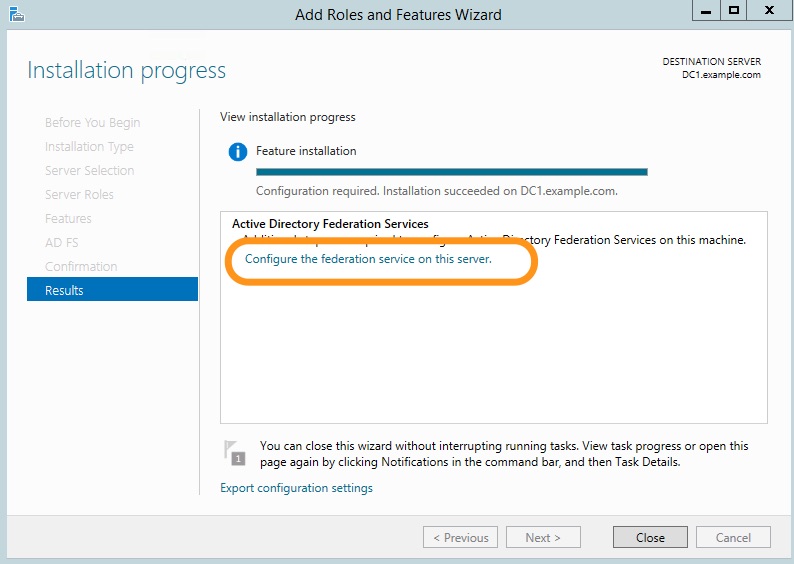
On step four, **Server Roles**, click **Active Directory Federation Services**, and click **Next**.



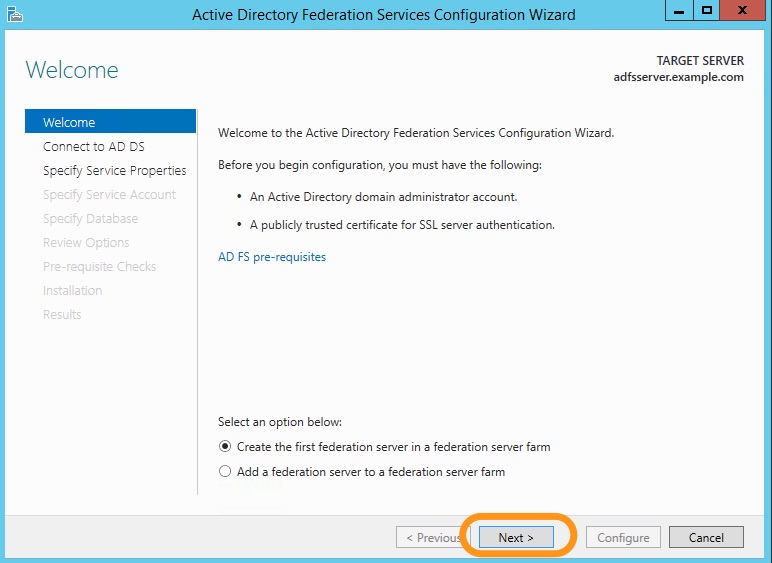
Accept the defaults for **Features** and **AD FS** by clicking **Next** two times. Leave **Restart the destination server automatically if required** unchecked, click **Install**.



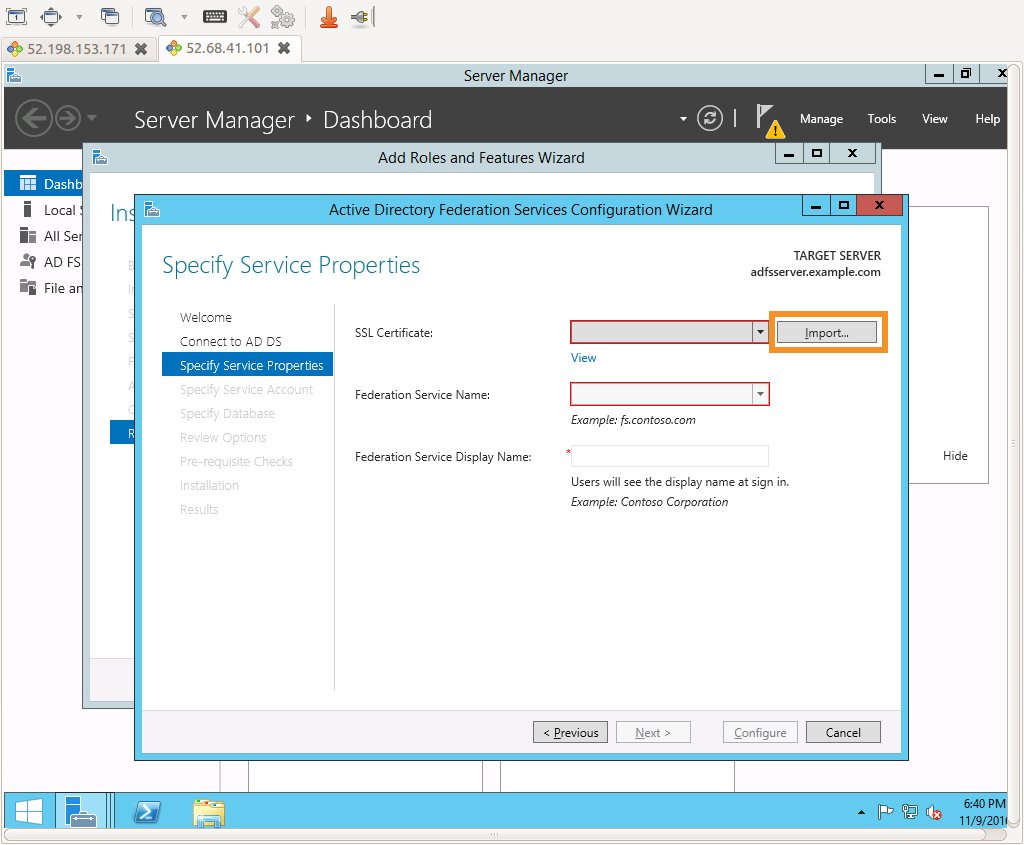
When the installation is complete, click **Configure the federation service on this server**. (leave **Restart the destination server automatically if required** unchecked)



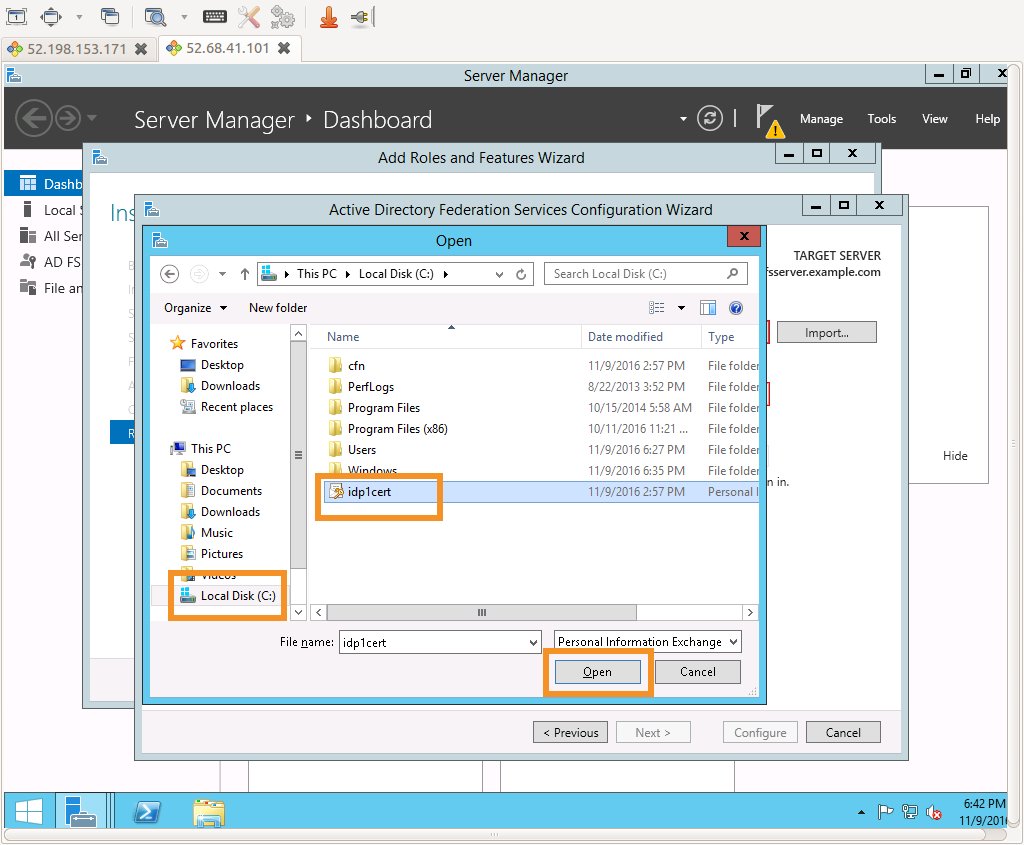
Accept the defaults for **Welcome** and **Connect to AD DS** by clicking **Next** two times.



On step three, **Specify Service Properties**, click **import**.



Navigate to **C:\*\*, click the file**idp1cert**, and then click**Open\*\*.



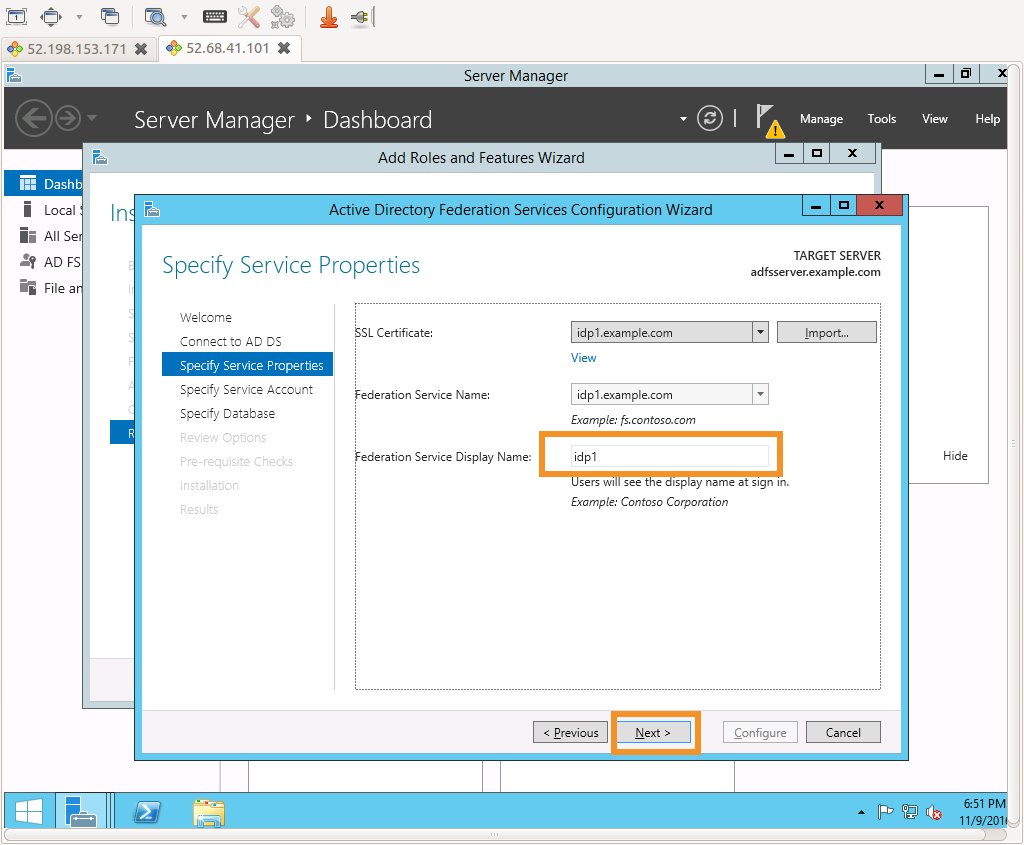
Type **Pass@123** in the password dialog for the certificate, and click **OK**.



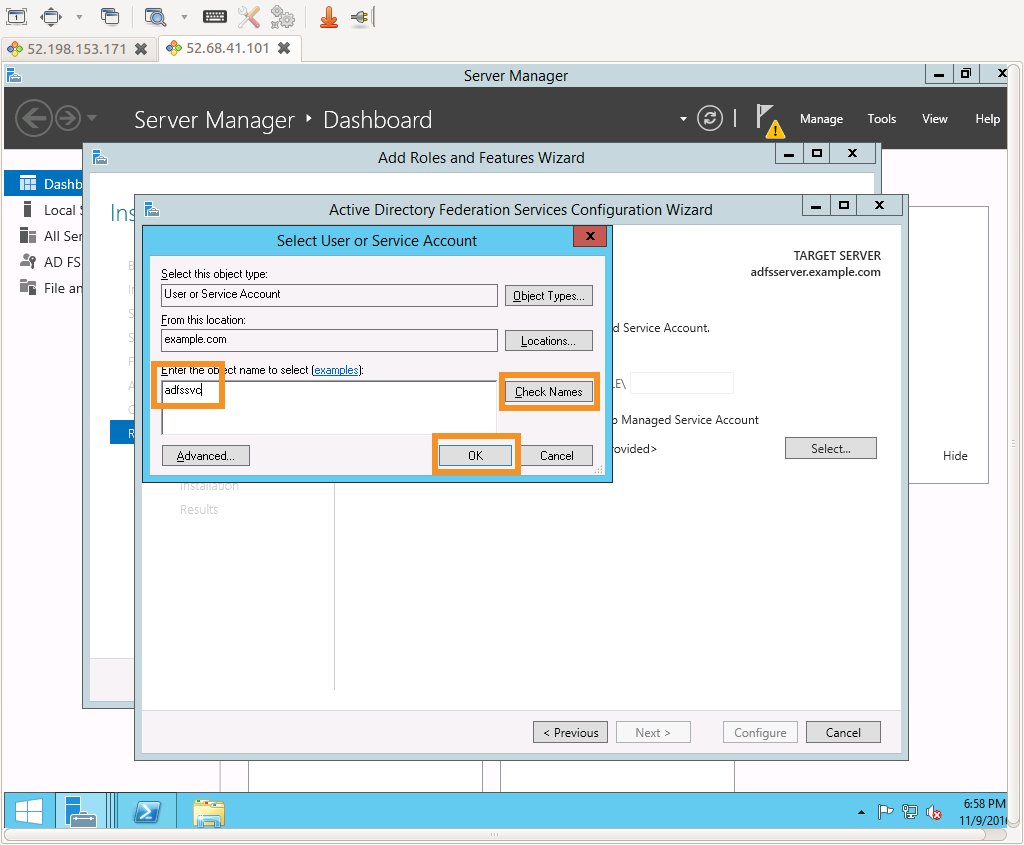
**Important**

This certificate was created and signed using a fictitious example.com certificate authority (CA). In practice, you would utilize a proper SSL certificate issued by the trusted CA of your choice. This simplification was done so that you can focus your time on the true learning objectives. Not appropriate for production use.

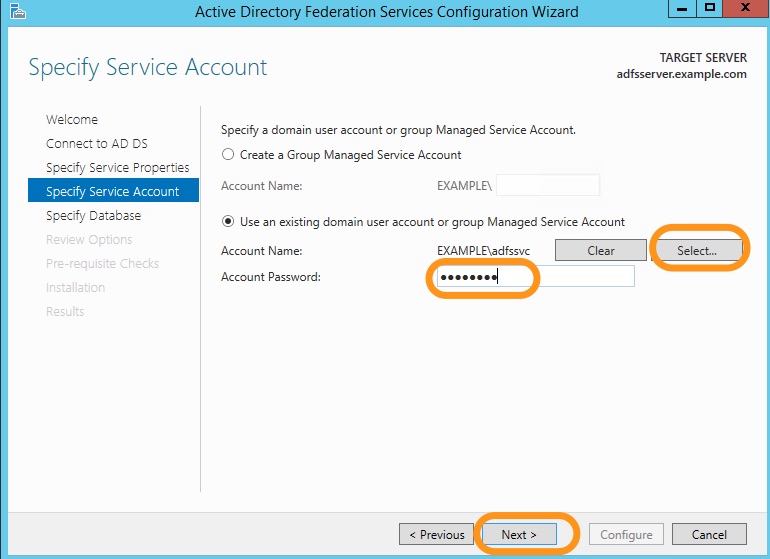
Type **idp1** as the **Federation Service Display Name**, and click **Next**.



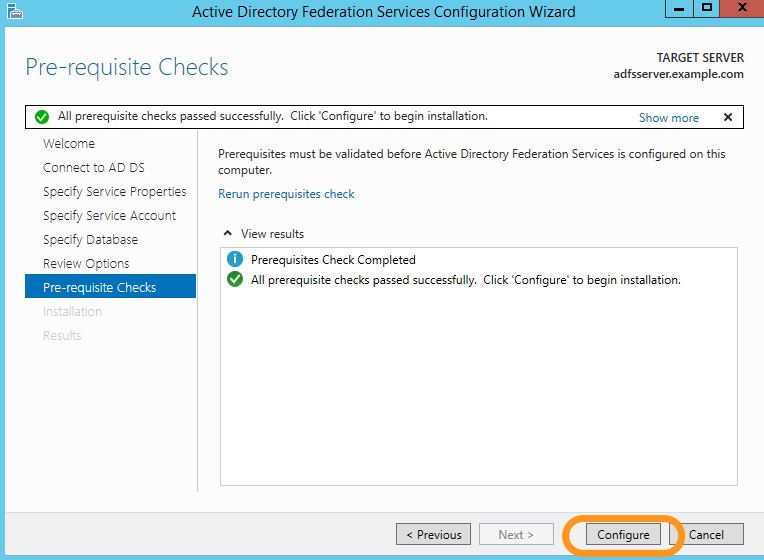
On the **Specify Service Account** page, click the radio button next to **Use an existing domain user account or group Managed Service Account** and then click **Select**. In the object name box type **adfssvc**, which is the service account that was created earlier in the exercise. Then click **Check Names**, and then **OK**.



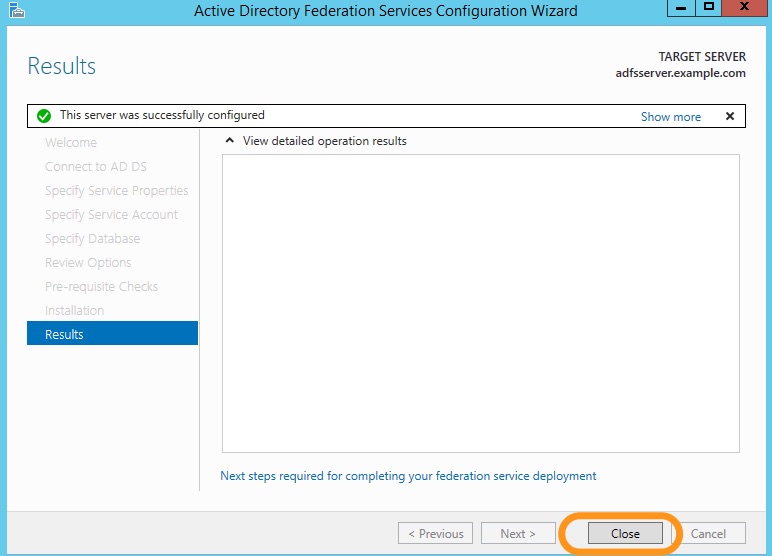
Recall that the password for **adfssvc** was set to a random value and can be found in the Outputs section of the CloudFormation stack. Type this value as the **Password**, and click **Next**.



Accept the defaults for **Specify Configuration Database** and **Review Options** by clicking click two times. On the **Pre-requisite Checks** page, click **Configure**.



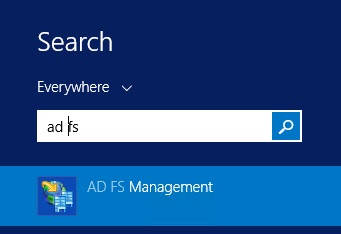
Click **Close** on the **Results** page to complete the configuration wizard.



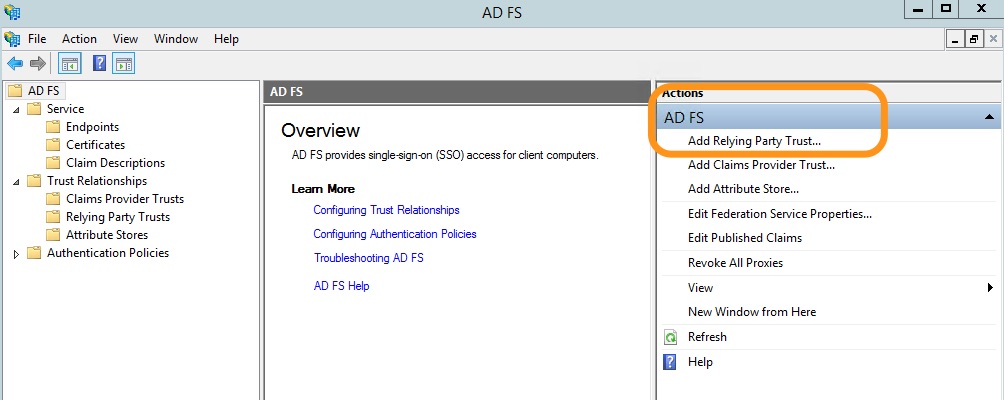
Configure AWS as a Trusted Relying Party

Next, we need to tell AD FS that it should offer authentication services for AWS. The SAML configuration for doing so is known as a relying party.

To begin, click the **Start menu**, type **ad fs**, and click **AD FS Management**.

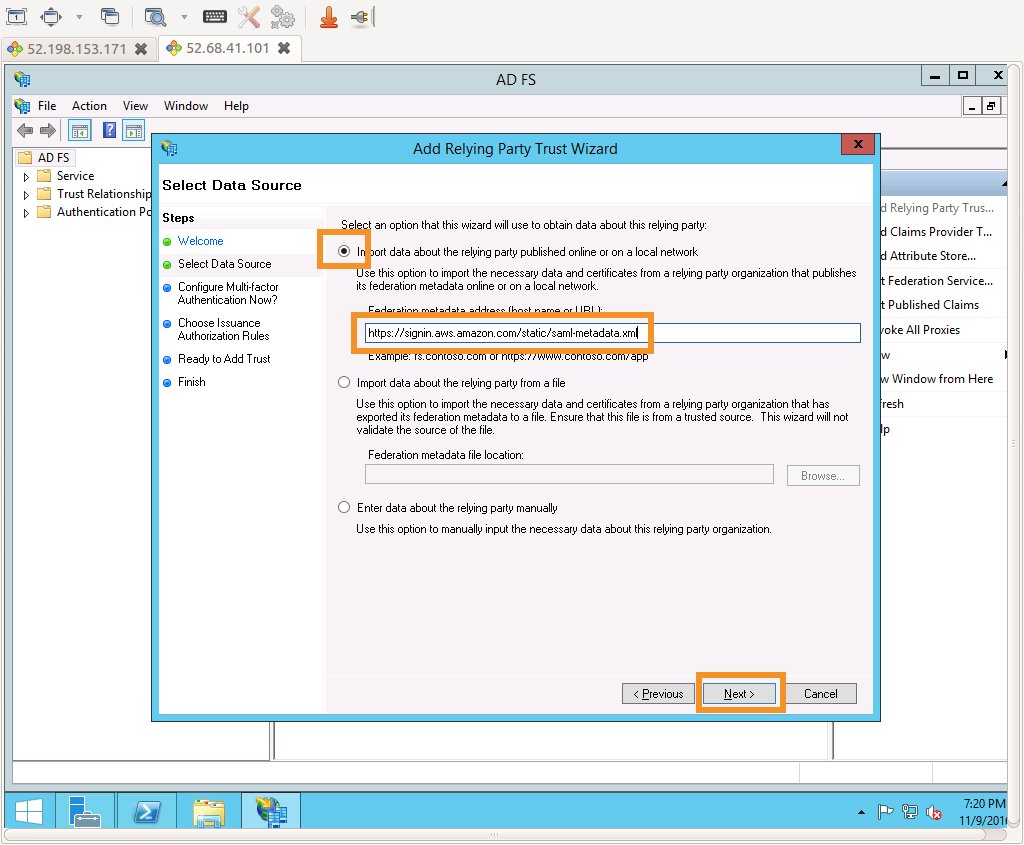


Click **Add Relying Party Trust** to start the **Add Relying Party Trust Wizard**.



Click **Start** to begin. Then, on **Select Data Source**, type the following URL for the AWS SAML metadata, and then click **Next**.

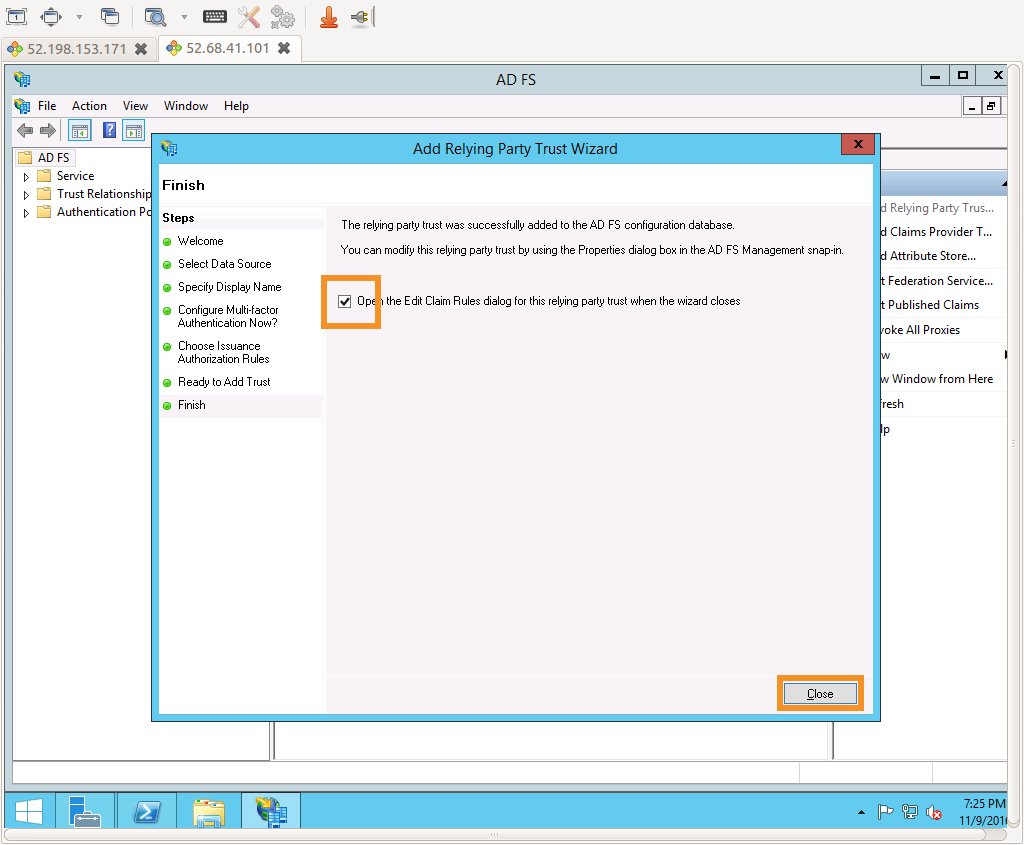
https://signin.aws.amazon.com/static/saml-metadata.xml



**Info**

SAML federation use metadata documents to maintain information about the public keys and certificates that each party utilizes. At run time, each member of the federation can then use this information to validate that the cryptographic elements of the distributed transactions come from the expected actors and haven't been tampered with. Since these metadata documents do not contain any sensitive cryptographic material, AWS publishes federation metadata at https://signin.aws.amazon.com/static/saml-metadata.xml.

Accept the defaults for **Specify Display Name**, **Configure Multi-factor Authentication Now?**, **Choose Issuance Authorization Rules** and **Ready to Add Trust** by clicking **Next** fours times. Click **Close** on the **Finish** page to complete the **Add Relying Party Trust Wizard**.

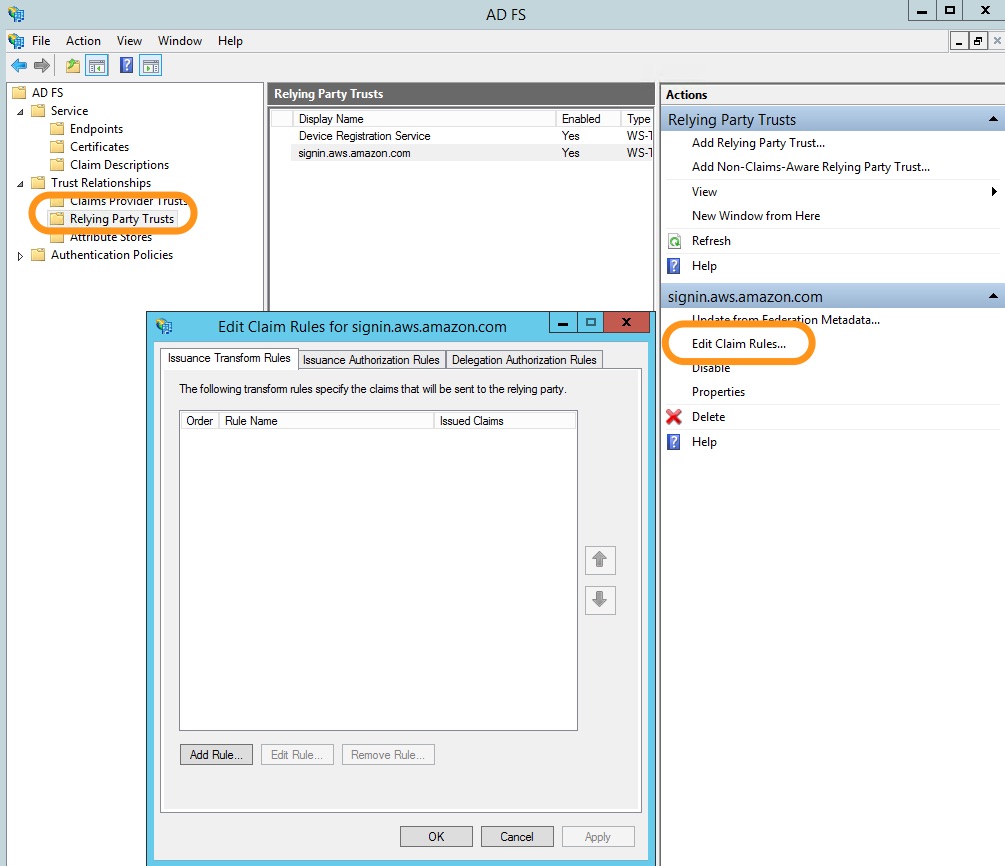


Configure claim rules for the AWS relying party

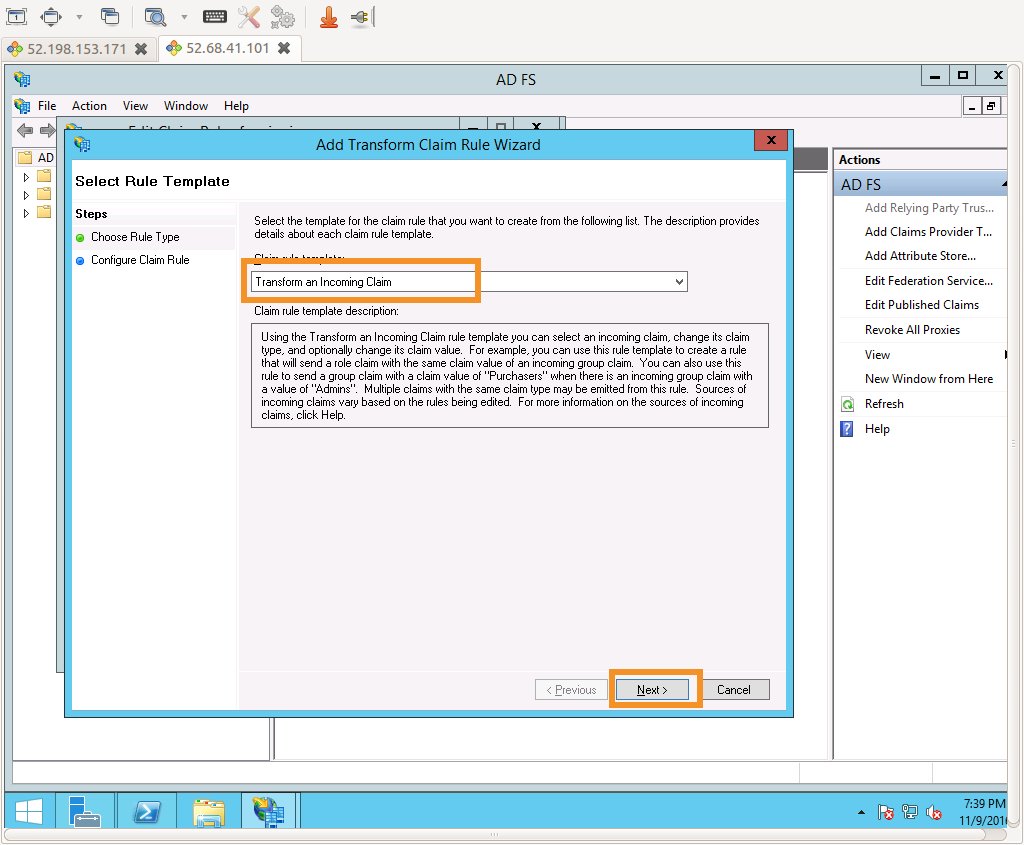
**Info**

In a SAML federation, the IdP can pass various attributes about the user, the authentication method, or other points of context to the service provider (in this case AWS) in the form of SAML attributes. In order for SAML federation for AWS to function properly, [several attributes](http://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_create_saml_assertions.html) are required. In AD FS, **Claim Rules** are used to assemble these required attributes using a combination of Active Directory lookups, simple transformations, and regular expression based custom rules. You will configure a total of **four claim rules**.

The **Edit Claim Rules** window should be already open, if not, you can re-open it from **Relying Party Trusts**, by choosing signin.aws.amazon.com, and then choosing **Edit Claim rules** as shown in the following screenshot.

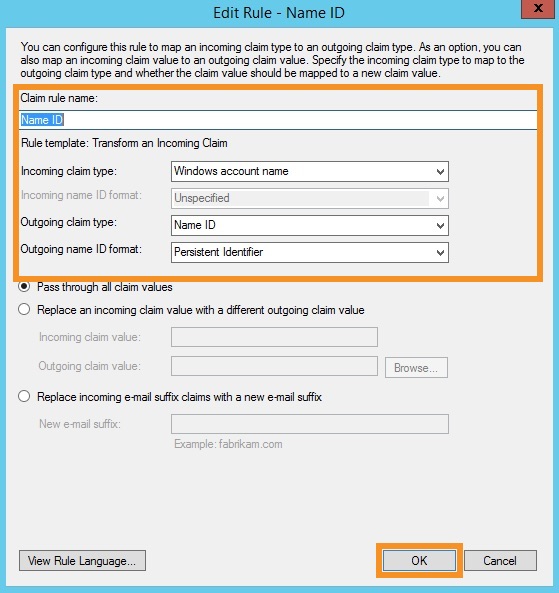


Click **Add Rule**, and then select **Transform an Incoming Claim** for the **Claim rule template**. Finally, click **Next**.

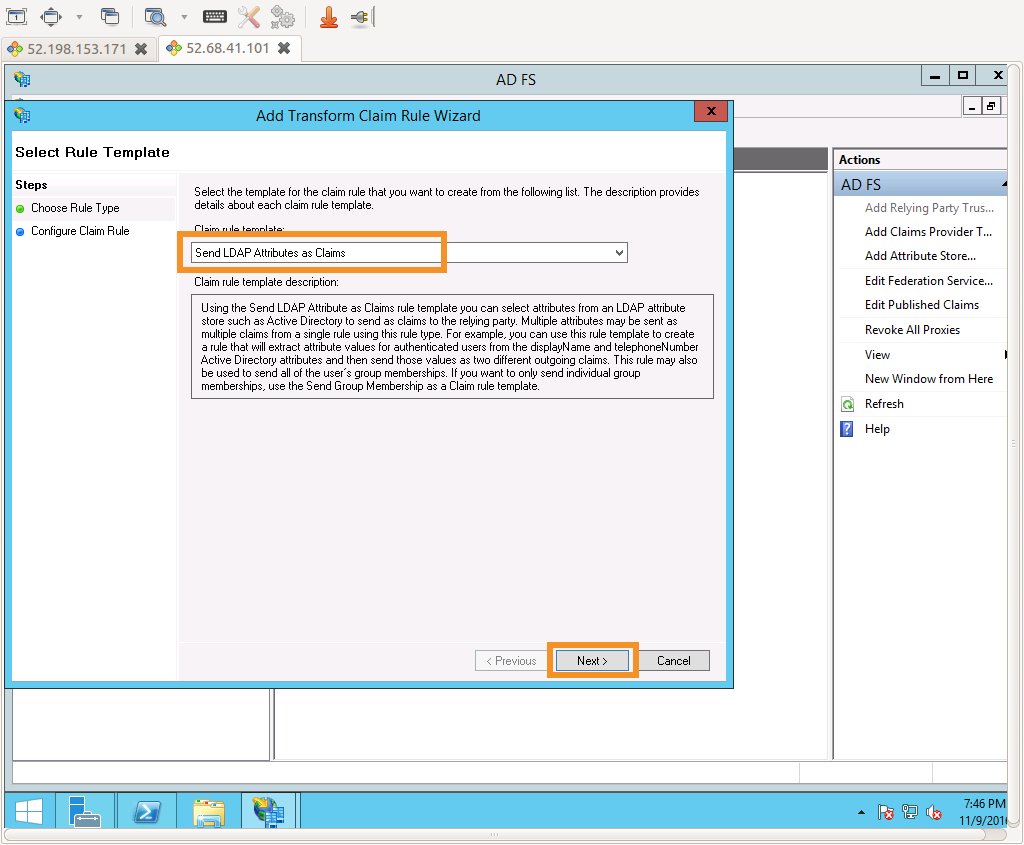


On the **Configure Claim Rule** page, type the following settings, and click **Finish**:

| Configuration Element | Value |
| --- | --- |
| Claim rule name | Name ID |
| Incoming claim type | Windows account name |
| Outgoing claim type | Name ID |
| Outgoing name ID format | Persistent Identifier |

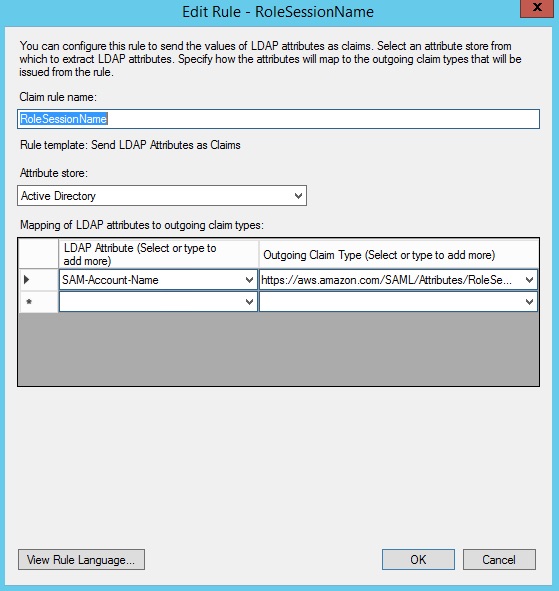


Click **Add Rule** to configure the second rule, and then select **Send LDAP Attributes as Claims**. Finally, click **Next**.

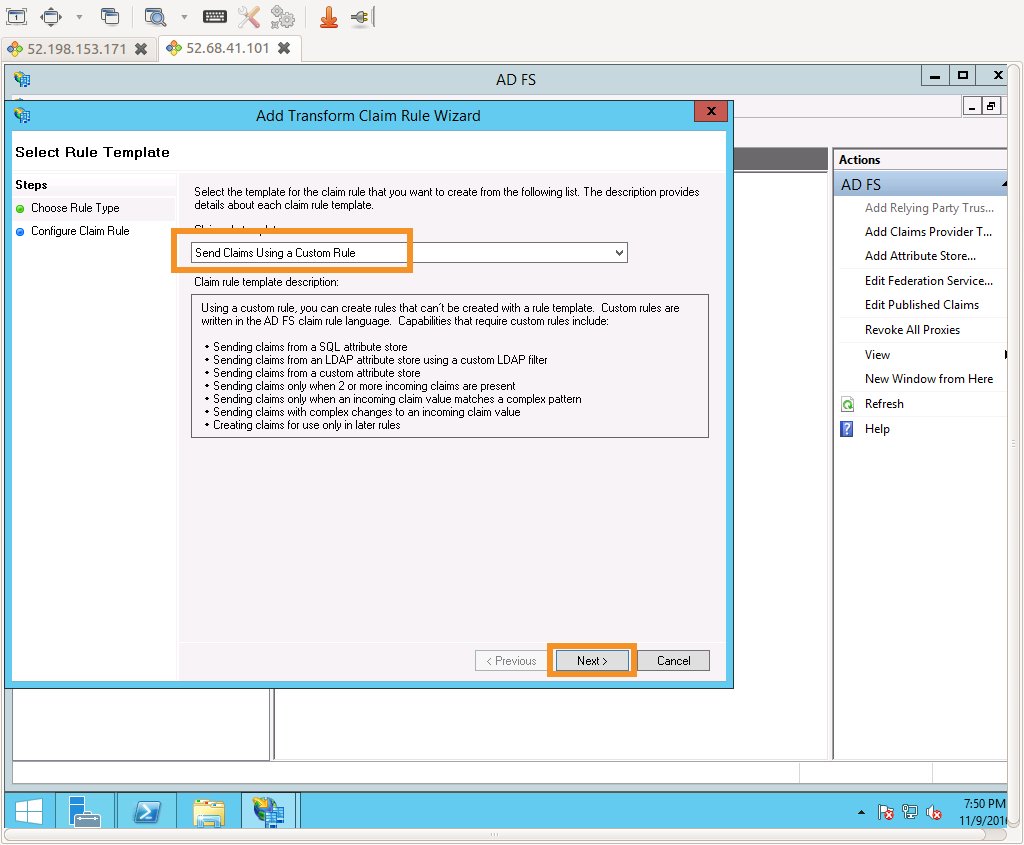


On the **Configure Claim Rule** page, type the following settings, and click **Finish**:

| Configuration Element | Value |
| --- | --- |
| Claim rule name | RoleSessionName |
| Attribute store | Active Directory |
| LDAP Attribute | SAM-Account-Name |
| Outgoing Claim Type | https://aws.amazon.com/SAML/Attributes/RoleSessionName |



Click **Add Rule** to configure the third rule, and then select **Send Claims Using a Custom Rule**. Finally, click **Next**.



In the **Claim rule name** box, type **Get AD Groups**. In the **Custom rule** box, enter the following, and then click **Finish**.

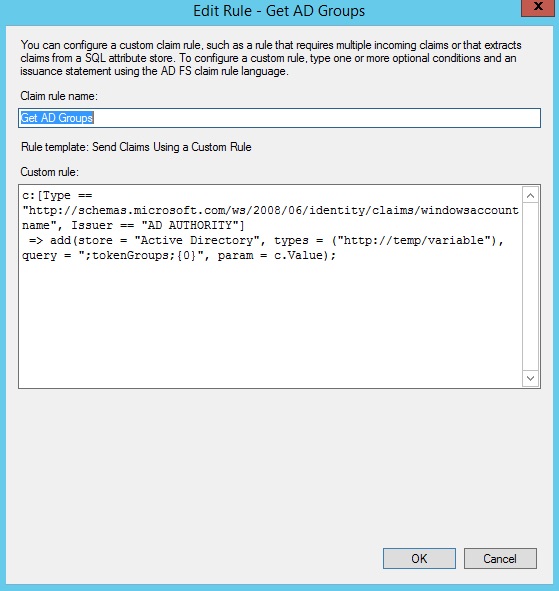
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => add(store = "Active Directory", types = ("http://temp/variable"), query = ";tokenGroups;{0}", param = c.Value);

**Hint**

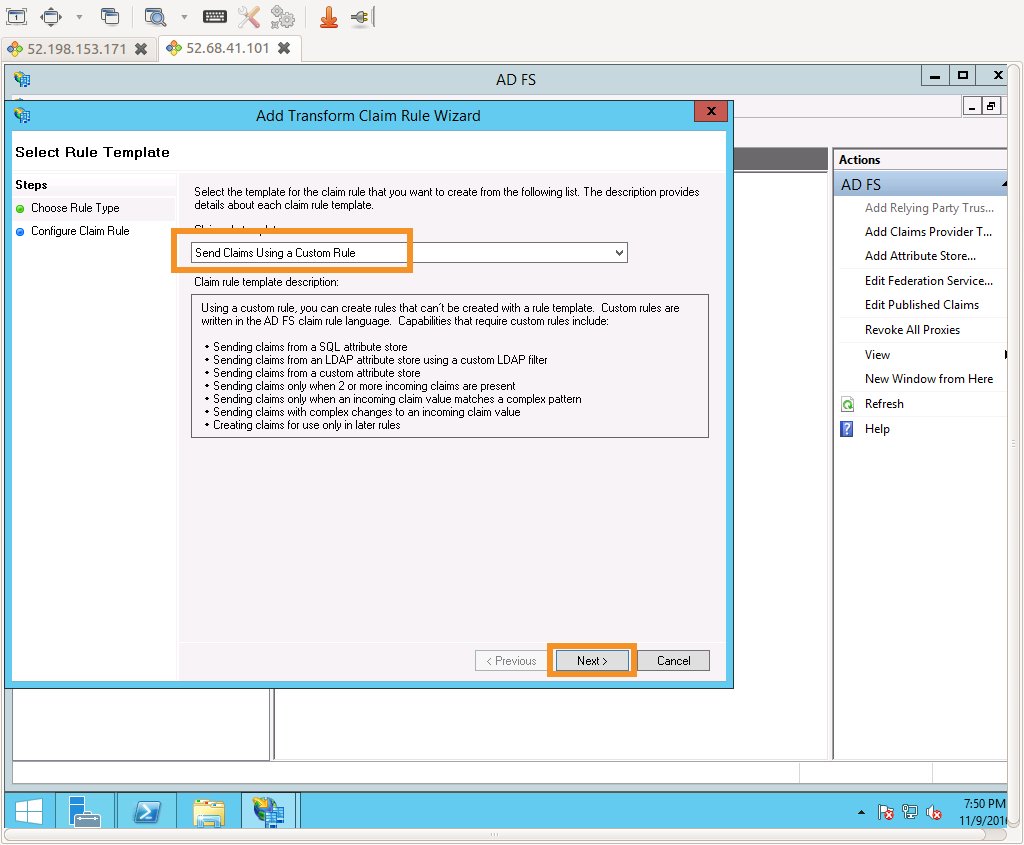
Ensure that there isn't any trailing whitespace, as this may cause unexpected results.

**Info**

This custom rule uses a script in the claim rule language to retrieve all the AD Groups the authenticated user is a member of and places them into a temporary claim named http://temp/variable. Think of this as a variable you can access later.



Click **Add Rule** to configure the fourth and final rule, and then select **Send Claims Using a Custom Rule**. Finally, click **Next**.



In the **Claim rule name** box, type **Roles**. In the **Custom rule** box, enter the following, and then click **Finish**.

c:[Type == "http://temp/variable", Value =~ "(?i)^AWS-([^d]{12})-(\w\*)"] => issue(Type = "https://aws.amazon.com/SAML/Attributes/Role", Value = RegExReplace(c.Value, "AWS-([^d]{12})-(\w\*)", "arn:aws:iam::$1:saml-provider/idp1,arn:aws:iam::$1:role/FederationWorkshop-$2"));

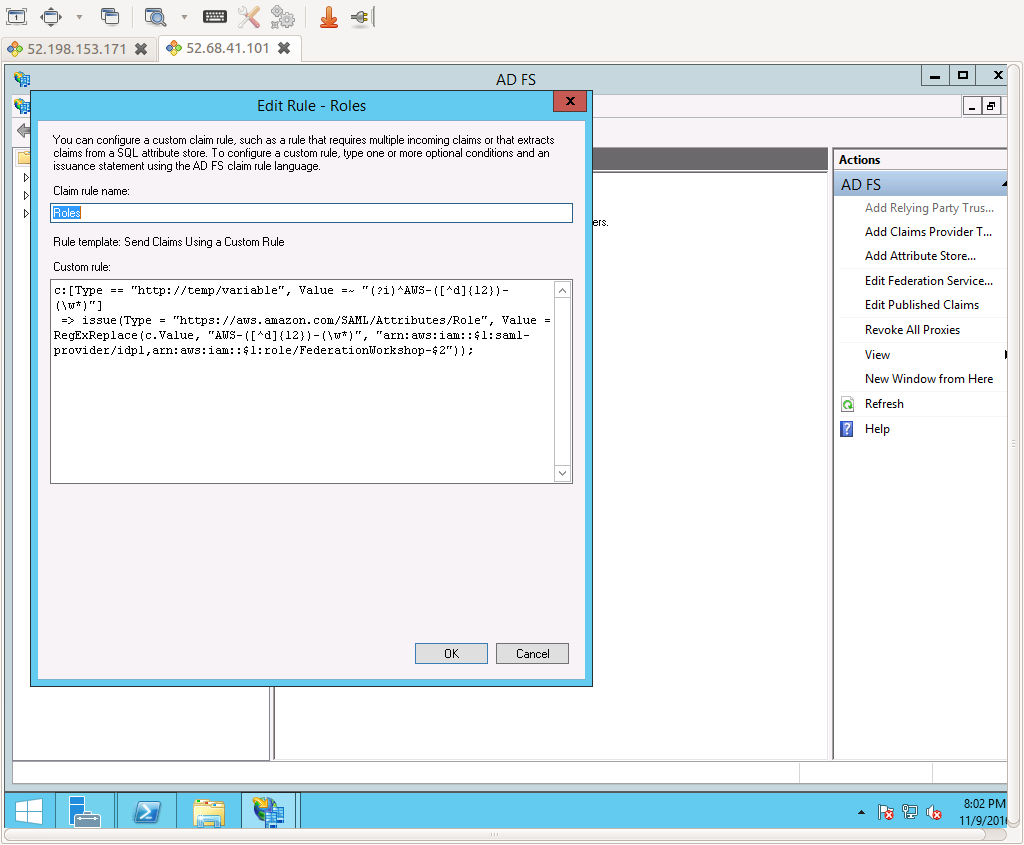
**Hint**

Ensure that there isn't any trailing whitespace, as this may cause unexpected results.

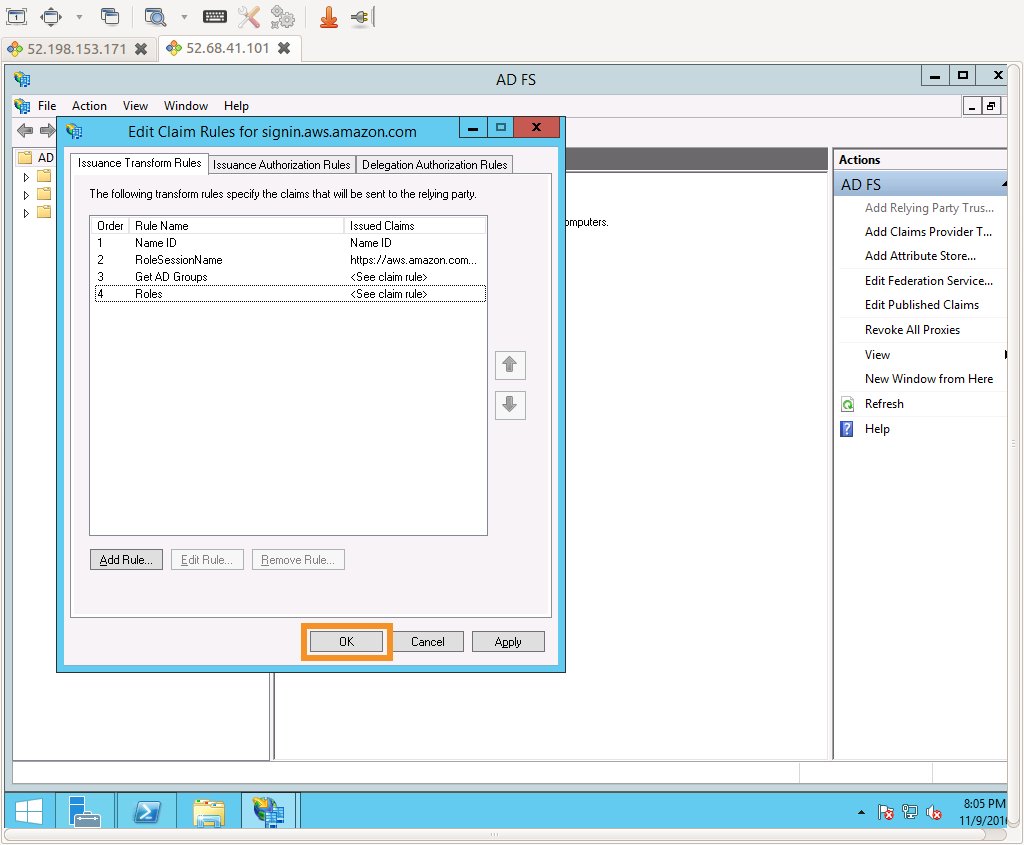
**Info**

This custom rule uses regular expressions to transform any AD Group membership of the form \*\*AWS-<Account Number>-<Role Name>\*\* into in the "IAM role ARN, IAM federation provider ARN" form AWS expects. It does so by matching the first pattern match (\d{12}) to $1 and the second pattern match (\w\*) to $2 for each entry in the temp variable.

This attribute makes better sense later in the exercise when you see it in action. For now, the key takeaway is that you are defining the resulting value for the AWS Role attribute in a dynamic way (by mapping AD Group memberships) instead of a static way (by explicitly defining ARNs). This dynamic resolution allows the IdP configuration to scale to support virtually any number of AWS accounts and any number of IAM roles without further configuration.



Click **OK** to complete the Claim rules setup.



Task 6 Configure host file entry for AD FS on your local workstation

**Note**

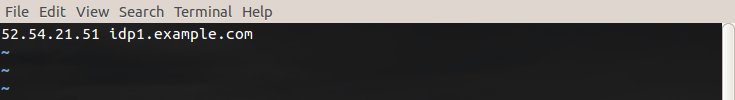
You should perform this configuration on your local workstation, not the EC2 instance where AD FS was installed.

Since example.com is only an illustrative domain, you will configure a local host file entry instead of configuring a DNS record for your identity provider. Choose the direction set below based on your workstation's operating system (Linux, Mac, or Windows).

Click here for a Windows host file entry



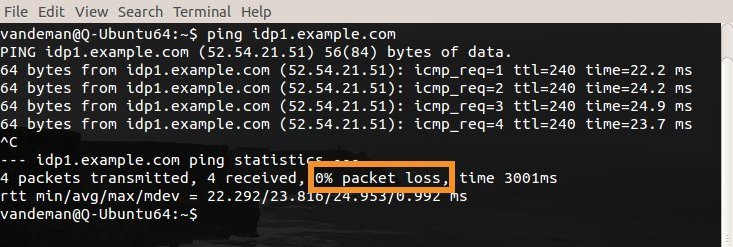
Click here for a Linux/Mac host file entry



Test the host file entry

After configuring the host file entry, make sure you can **ping** your AD FS instance:

ping idp1.example.com



Task 7 Configure the IAM Identity Provider

**Info**

Now that we have finished the installation and configuration of AD FS, we need to configure AWS to trust the authentication and authorization information that AD FS that provides. During this process you will provide AWS with the SAML metadata from your ADFS installation, which details all of the encryption and signing certificates that AD FS will use. This exchange provides the preestablished cryptographic trust needed for AWS to utilize the assertions generated by AD FS.

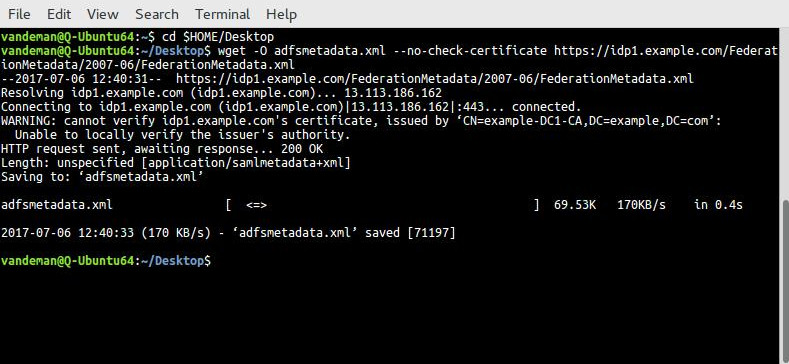
To start this process, we need to configure an IAM Identity Provider within the AWS console:

Download IdP metadata

Start by downloading the AD FS federation metadata from your newly configured IdP. This is the companion to the AWS federation metadata that you installed in the IdP itself.

cd $HOME/Desktop (or another temporary directory of your choice)

wget -O adfsmetadata.xml --no-check-certificate https://idp1.example.com/FederationMetadata/2007-06/FederationMetadata.xml



**Hint**

You can also use a web browser to download the metadata.

Create AWS IAM IdP

Now that the metadata is downloaded, open the [**AWS IAM Console**](https://console.aws.amazon.com/iam/home?region=us-east-1#/home), and click **Identity providers** from the left pane.

**Hint**

Certain versions of Firefox produce errors during the creation of the Identity Provider, consider using Chrome if possible.

Next, click **Create Provider**. Select SAML as the **provider type**. Enter the following values to configure the provider, and click **Next Step**.

| Provider Type | SAML |
| --- | --- |
| Provider Name | idp1 |
| Metadata Document | Browse to file downloaded in previous step |

Verify the information is correct and then click **Create**.

**Info**

The banner at the top of the screen provides confirmation that your identity provider was successfully created.

Create AWS IAM roles for IdP access

In our final step before initial testing, we need to create a set of IAM roles that correspond one to one to the AD Groups we created and populated. These roles will be named such that their resulting ARNs will match the regular expression transformation we configured in the AD FS claim rules earlier.

* Start by clicking **Roles** from the left pane and then click **Create Role**
* Under **Selet type of trusted entity**, choose the **SAML 2.0 federation** option
* For the **SAML provider** select idp1 and the select the radio button next to **Allow programmatic and AWS Management Console access**. This will populate **Attibute** and **Value** for you. Click **Next: Permissions**
* In the **Search** box enter **ReadOnlyAccess** and then search for and select the check next to the Policy **ReadOnlyAccess**. Click **Next: Tags**
* You don't need to enter any tags here, just click **Next: Review**
* Enter **FederationWorkshop-ReadOnly** for the **Role name**
* Click **Create role**

**Note**

The policy that you attach here specifically controls the AWS actions that the federated user is authorized to perform. In practice, you would attach a more meaningful and appropriate policy. The ReadOnlyAccess policy selected here has been chosen simply to facilitate this learning exercise.

Repeat the entire sequence of the preceding steps for a second role named FederationWorkshop-PowerUser. All other inputs remain the same except you will select **PowerUserAccess** policy.

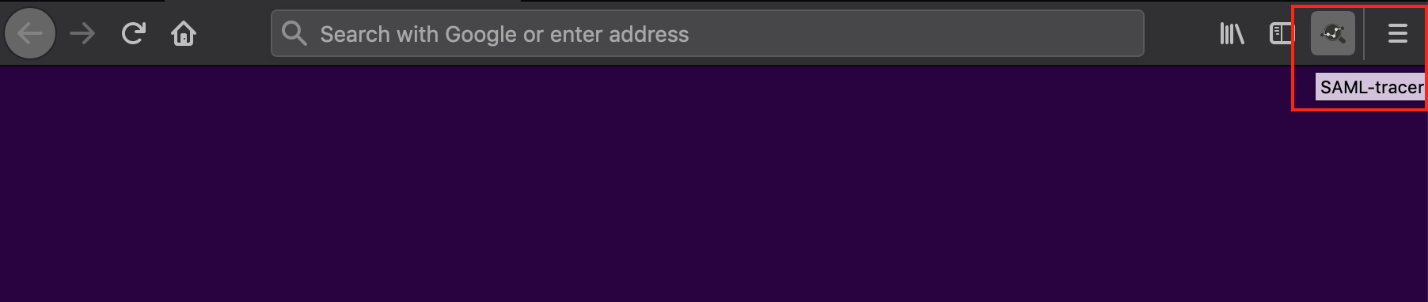
**Hint**

After creating this second role, type FederationWorkshop in the filter box on the IAM Roles page. You should see both of your newly created roles listed.

Task 8 Test federated access to the AWS Management Console

**Info**

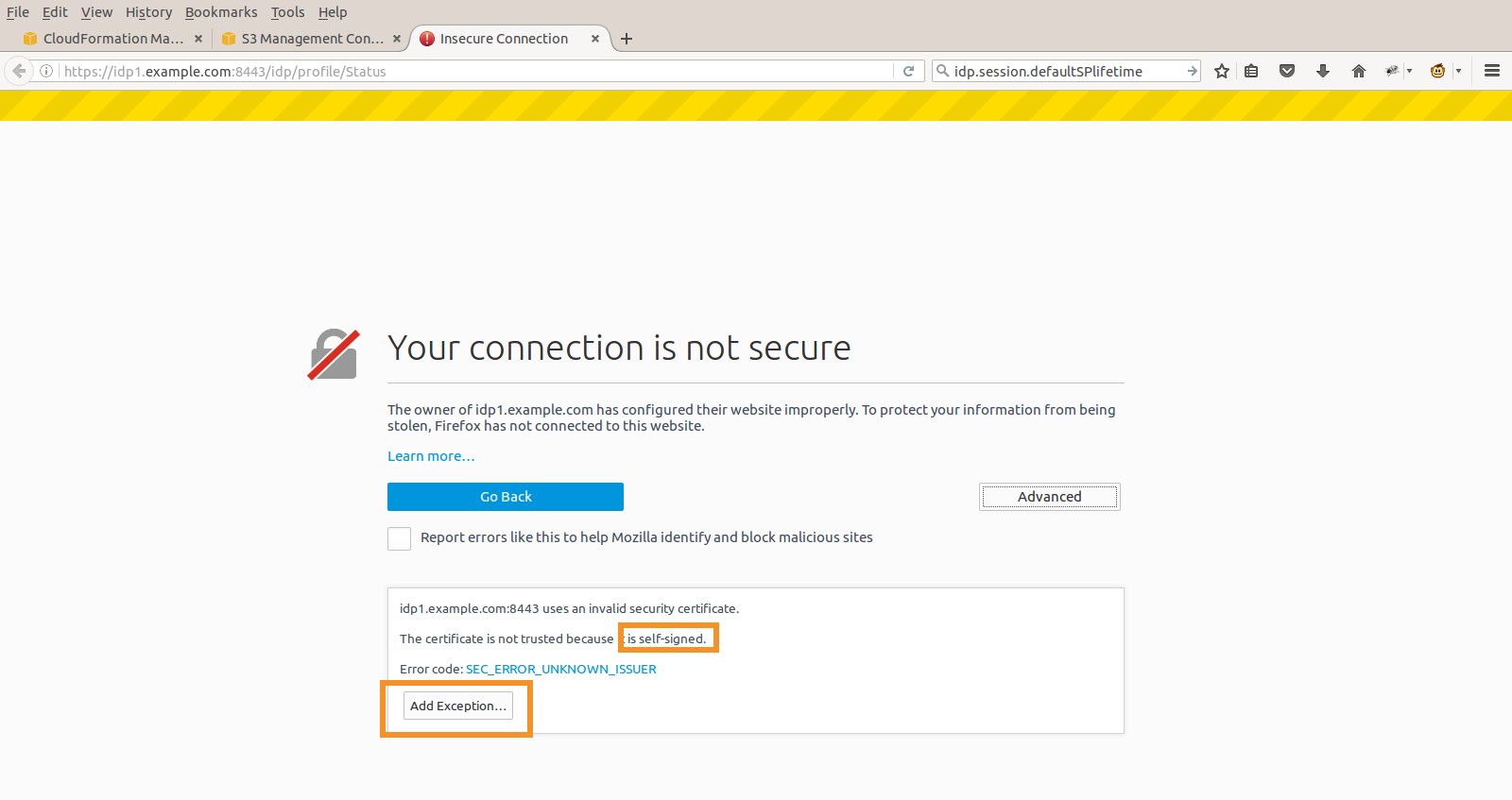
You are now ready to test federated access to the **AWS Management Console** using your newly configured Identity Provider. To do so you will need to install the **SAML-Tracer** browser add on (there is a Firefox and Chrome version). Then open a new browser window and open **SAML-tracer**. We'll use **SAML-tracer** to inspect the contents of the SAML assertions as they flow from the IdP to AWS. This allows you to see the results of your configurations and understand the information that AWS is consuming from your identity provider. See the following screenshot for help enabling SAML tracer.



With **SAML-tracer** enabled, switch back to the main browser window and enter the IdP initiated login URL for ADFS:

https://idp1.example.com/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices

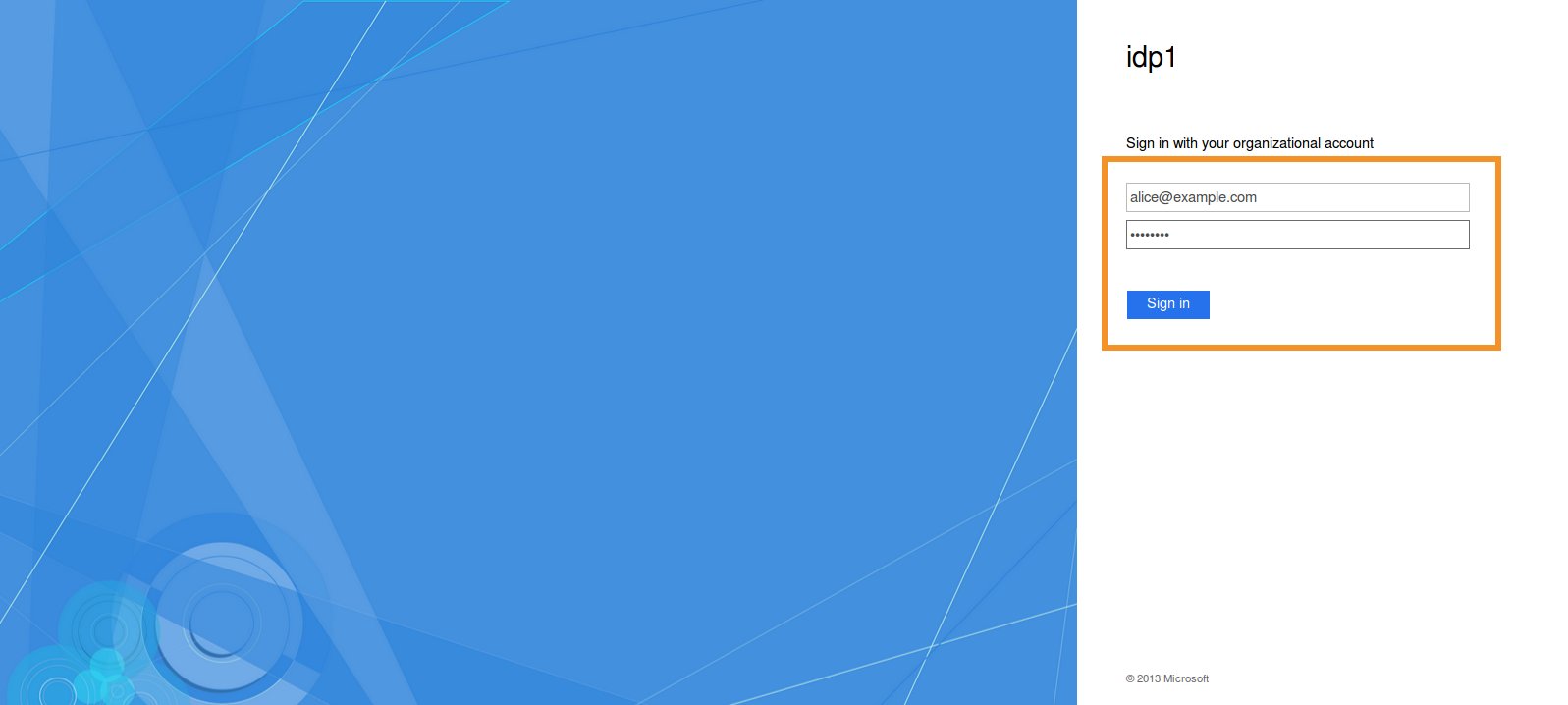
Because we configured ADFS with a self-signed certificate, you will be prompted to add a security exception.



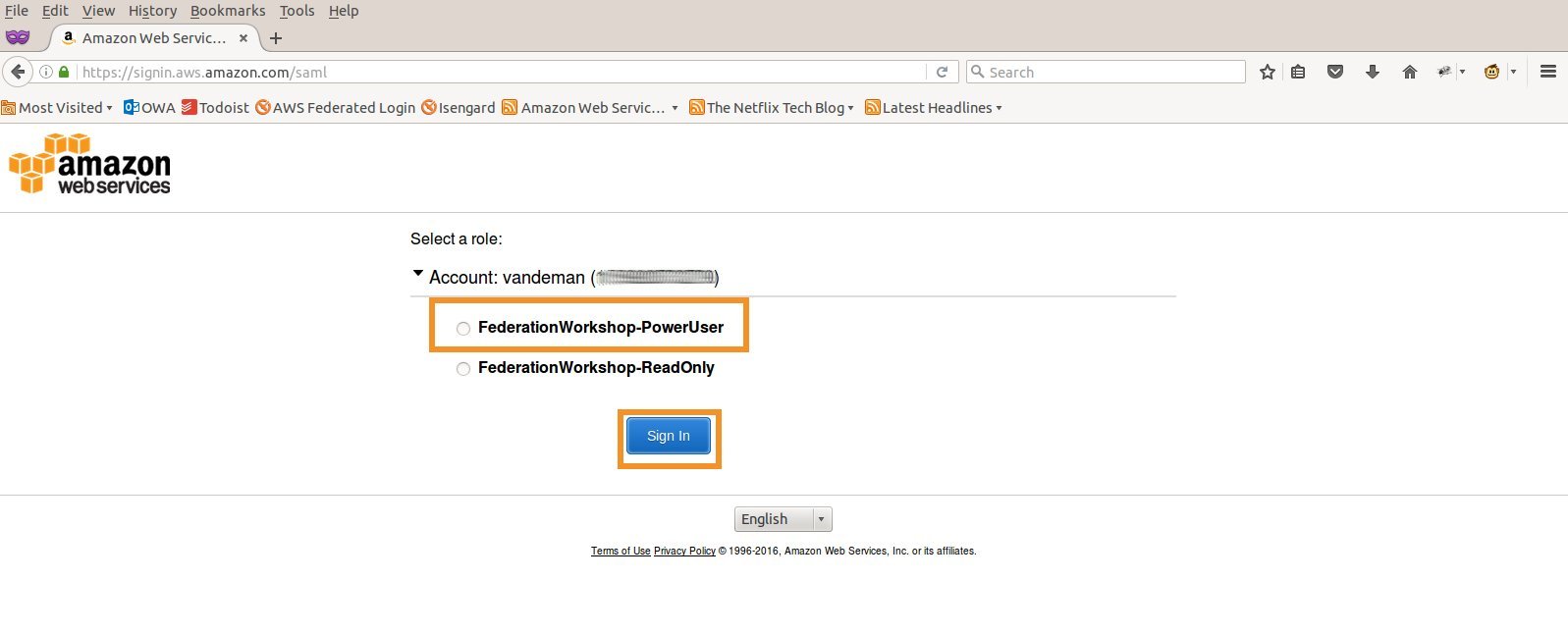
After adding the exception, AD FS will return the login page. Log in using **alice's credentials**.

**Hint**

Recall that alice's password has been set to a random value exposed as an output variable in the CloudFormation template.

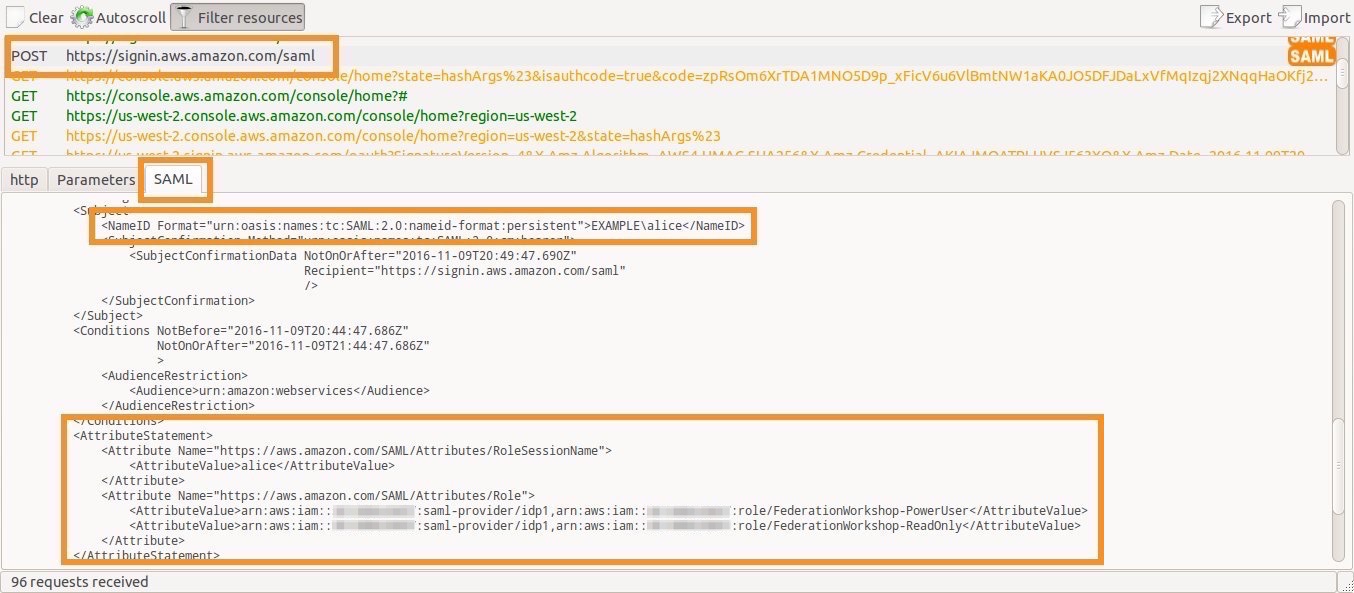


After successfully authenticating, you will then be redirected to the AWS role chooser page. This page is presented because alice was granted more than one role based on her group membership in Active Directory. Select the **FederationWorkshop-PowerUser** role and click **Sign In**.



You are then signed in to the **AWS Management Console** and provisioned into the role you selected. The selected IAM role and the associated attached policies control the set of AWS actions that your federated user can perform.

Before moving on, switch back to the **SAML-tracer** window, and scroll upward until you find the entry for https://signin.aws.amazon.com/saml. Choose that entry, and select the **SAML** tab in the lower pane. As you look through the assertion, look for the \*\*<AttributeStatement>\*\*. This will allow you to see how the configurations you applied above translate into the actual SAML attributes that are passed to AWS. See the following screenshot for reference.



Task 9 Configure federated access to the AWS CLI and SDKs

**Info**

In the final step of this initial exercise, you need to complete the implementation and testing of the federated API/CLI access that you started earlier in the exercise. This will enable your users (in this case bob and alice) to access your AWS environment using their example.com credentials through the AWS CLI or one of the AWS SDKs.

Return to the directory on your local workstation where you downloaded the script, and **execute** it using either alice or bob's credentials.

**Hint**

On Linux and Mac you may need to give the script execute permissions if it doesn't currently have it. Run sudo chmod 744 samlapi\_formauth\_adfs3.py

$ python ./samlapi\_formauth\_adfs3.py

Username: alice@example.com

Password: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Please choose the role you would like to assume:

[ 0 ]: arn:aws:iam::012345678987:role/FederationWorkshop-ReadOnly

[ 1 ]: arn:aws:iam::012345678987:role/FederationWorkshop-PowerUser

Selection: 1

---------------------------------------------------------------

Your new access key pair has been stored in the AWS configuration file /home/alice/.aws/credentials under the saml profile.

Note that it will expire at 2016-10-16T17:16:20Z.

After this time, you may safely rerun this script to refresh your access key pair.

To use this credential, call the AWS CLI with the --profile option (e.g., aws --profile saml ec2 describe-instances).

---------------------------------------------------------------

Simple API example listing all S3 buckets:

[<Bucket: mybucket1>, <Bucket: mybucket2>, <Bucket: mybucket3>, <Bucket: mybucket4>, <Bucket: mybucket5>]

After you've done so, let's utilize the profile for a few additional CLI calls. First, let's retrieve the list of SAML providers.

alice@Ubuntu64:/tmp$ aws --profile saml iam list-saml-providers

{

"SAMLProviderList": [

{

"CreateDate": "2016-10-18T20:26:17Z",

"ValidUntil": "2116-10-18T20:26:17Z",

"Arn": "arn:aws:iam::012345678987:saml-provider/idp1"

}

]

}

Next, let's inspect the FederationWorkshop-ReadOnly using a CLI filter.

alice@Ubuntu64:/tmp$ aws --profile saml iam list-roles --query 'Roles[?RoleName==`FederationWorkshop-ReadOnly`]'

[

{

"AssumeRolePolicyDocument": {

"Version": "2012-10-17",

"Statement": [

{

"Action": "sts:AssumeRoleWithSAML",

"Effect": "Allow",

"Condition": {

"StringEquals": {

"SAML:aud": "https://signin.aws.amazon.com/saml"

}

},

"Principal": {

"Federated": "arn:aws:iam::012345678987:saml-provider/idp1"

}

}

]

},

"RoleId": "AROAJZN6LB2TJXXXXXXXX",

"CreateDate": "2016-10-18T20:51:27Z",

"RoleName": "FederationWorkshop-ReadOnly",

"Path": "/",

"Arn": "arn:aws:iam::012345678987:role/FederationWorkshop-ReadOnly"

}

]

Depending on the AWS resources available in your account, try to formulate one or two additional calls (describe or list) using the saml profile. The [AWS CLI Command Reference](http://docs.aws.amazon.com/cli/latest/reference/) can help you construct these calls.

Exercise complete

**Congratulations!** You've successfully completed the first half of the exercise and now have:

* A working AD FS 3.x IdP backed by Active Directory.
* Integration between this identity infrastructure and your AWS account.
* Federated access to the AWS Management Console, CLI, and SDKs for two federated users: bob & alice.

With this setup complete, you are now ready to advance your journey into the more advanced use cases. As described in the introduction, you may complete these advanced use cases in the order and combination that you choose: [**Click here to go to the Advanced use cases (Microsoft variant)**](https://identity-federation.awssecworkshops.com/microsoft/build-advanced/)

Cleanup

After you have completed all of your desired use cases, complete the following clean up steps.

* If you completed one of the S3 use cases, delete all of the test S3 objects.
* Delete all of the FederationWorkshop CloudFormation stacks.
* Delete all of the FederationWorkshop IAM roles and the idp1 IAM identity provider