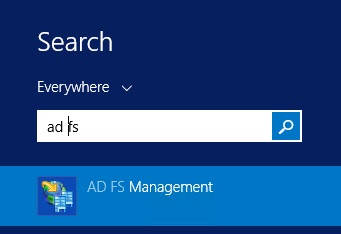
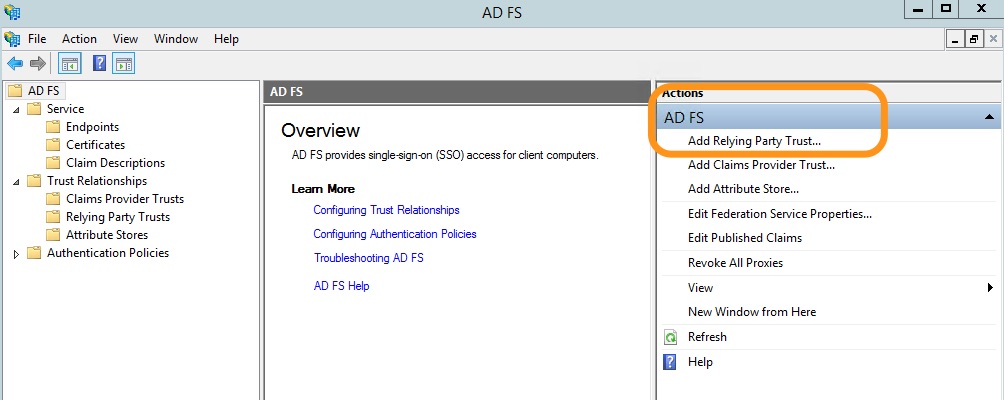
**Configure AWS as a Trusted Relying Party**

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 start, choose the **Start menu**, type **ad fs**, and choose **AD FS Management**.

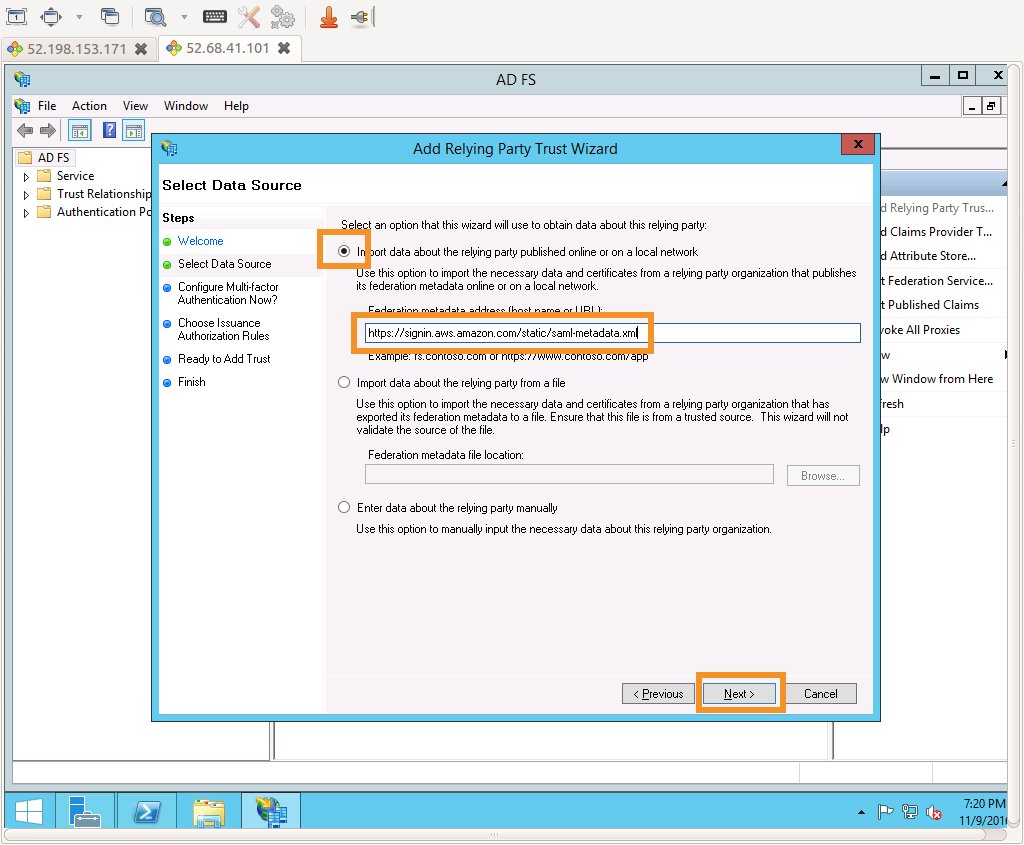


Choose **Add Relying Party Trust** to start the **Add Relying Party Trust Wizard**.



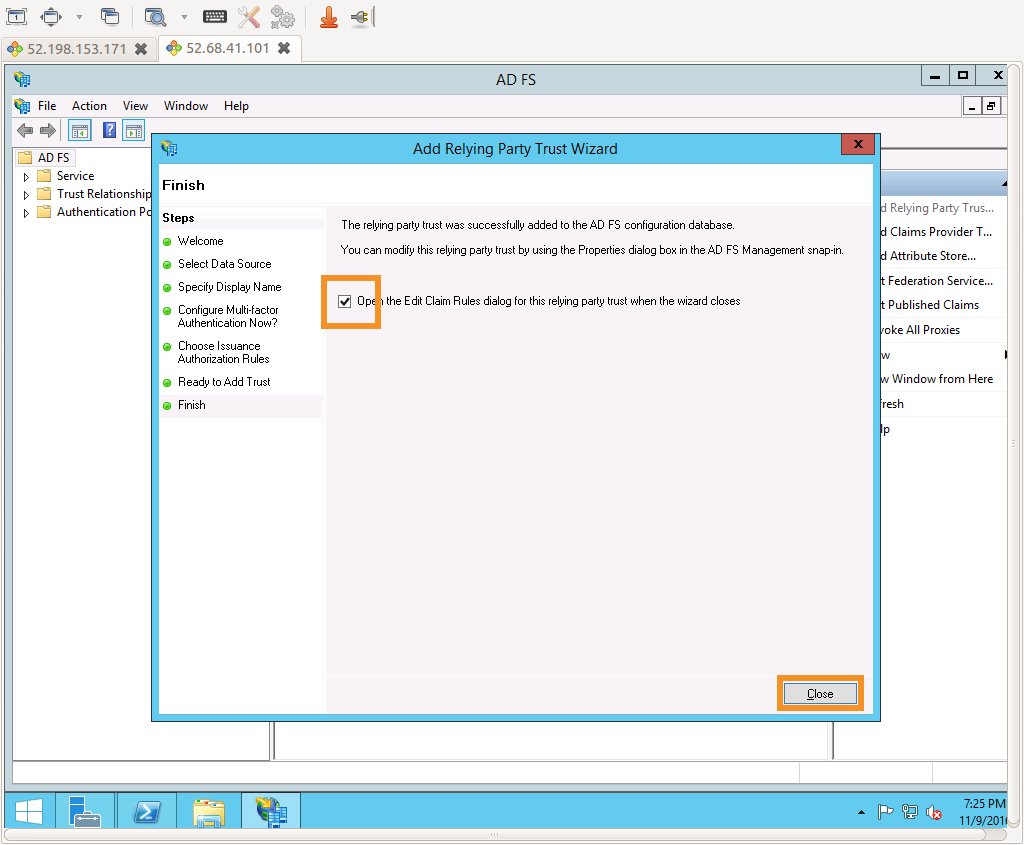
Choose **Start** to begin. Then, on **Select Data Source**, type the following URL for the AWS SAML metadata, and then choose **Next**.

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



**Note:** SAML federations 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.

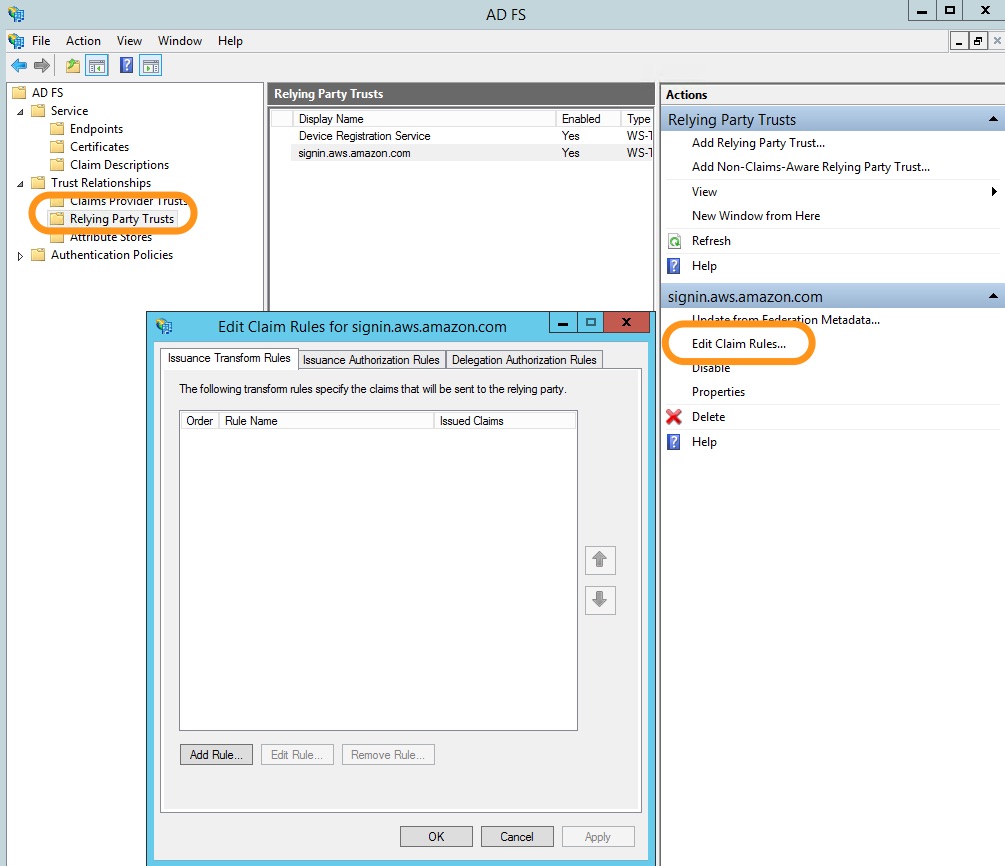
Accept the defaults for **Specify Display Name**, **Configure Multi-factor Authentication Now?**, and **Choose Issuance Authorization Rules** by choosing **Next** three times. Choose **Close** on the **Finish** page to complete the **Add Relying Party Trust Wizard**.



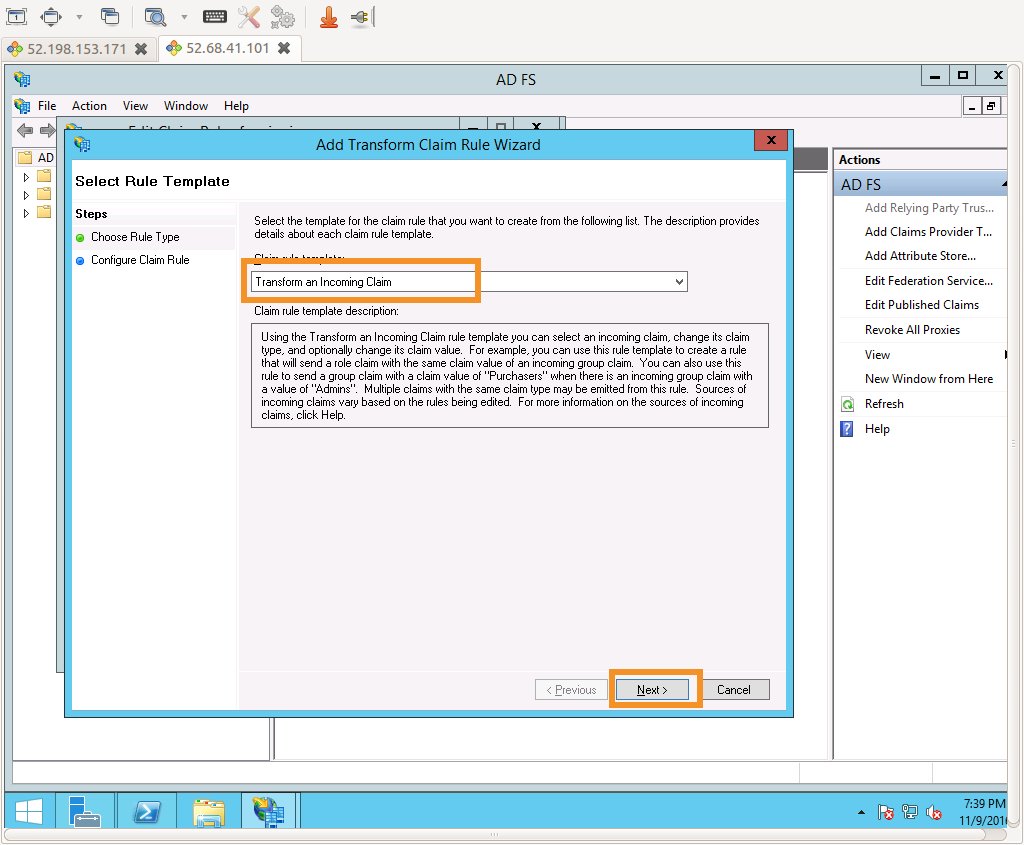
**Configure claim rules for the AWS relying party**

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.

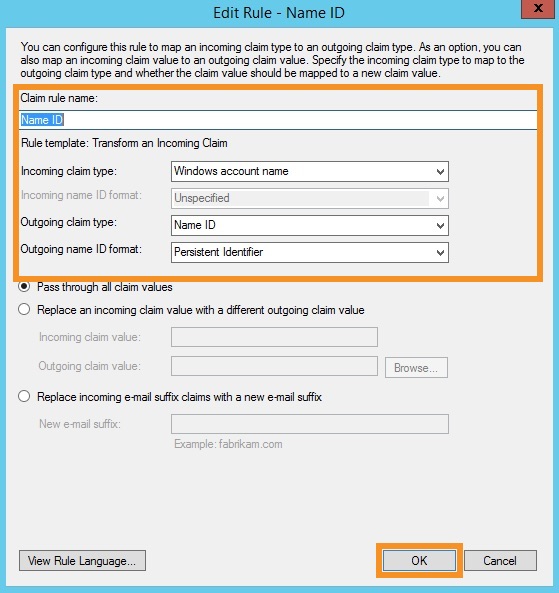


Choose **Add Rule...** to configure the first rule, and then choose **Transform an incoming claim**. Finally, choose **Next**.

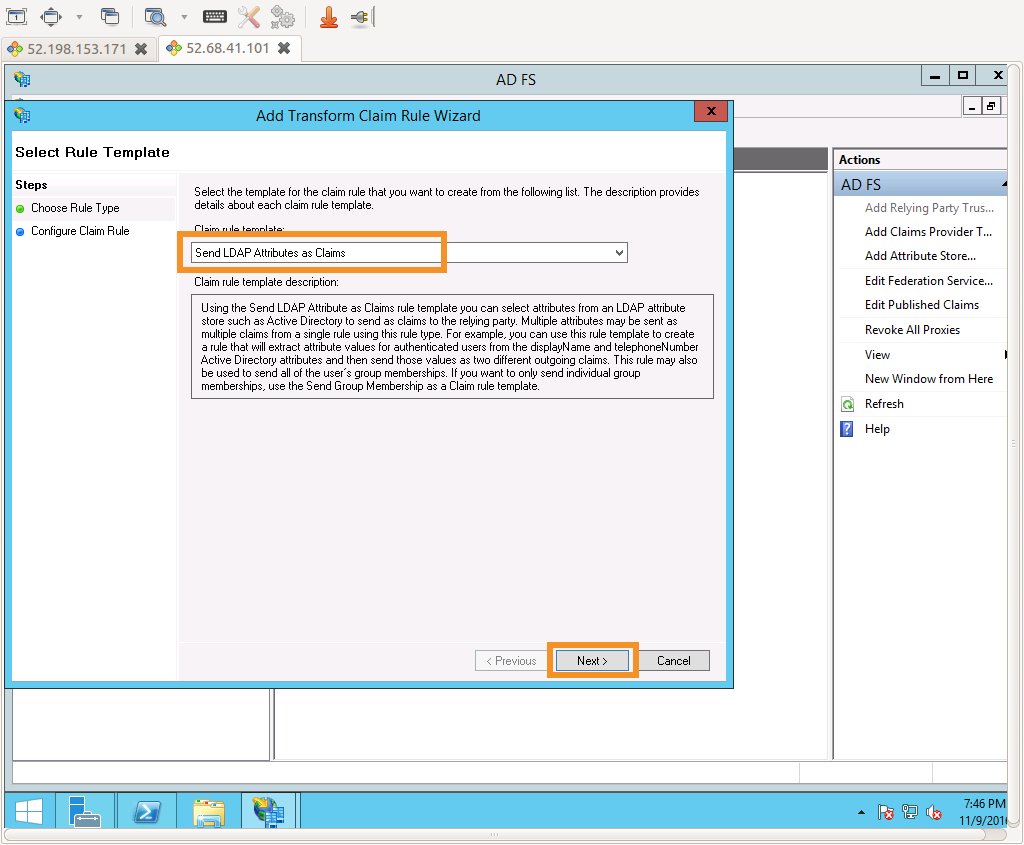


On the **Configure Claim Rule** page, type the following settings, and choose **OK**, as shown in the following screenshot.

|  |  |
| --- | --- |
| **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 |

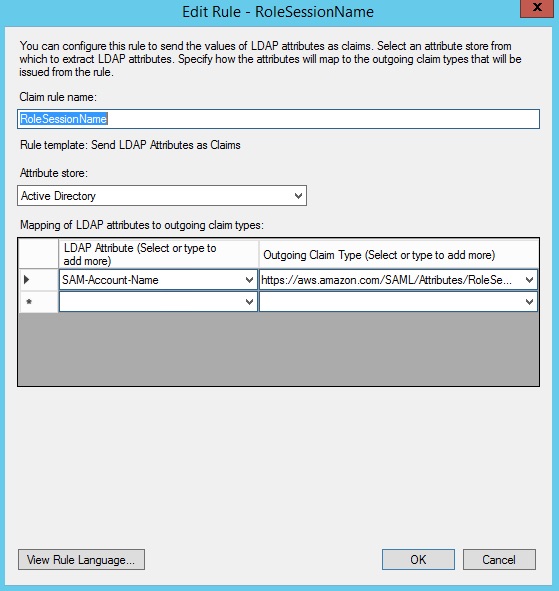


Choose **Add Rule** to configure the second rule, and then choose **Send LDAP Attributes as Claims**. Finally, choose **Next**.

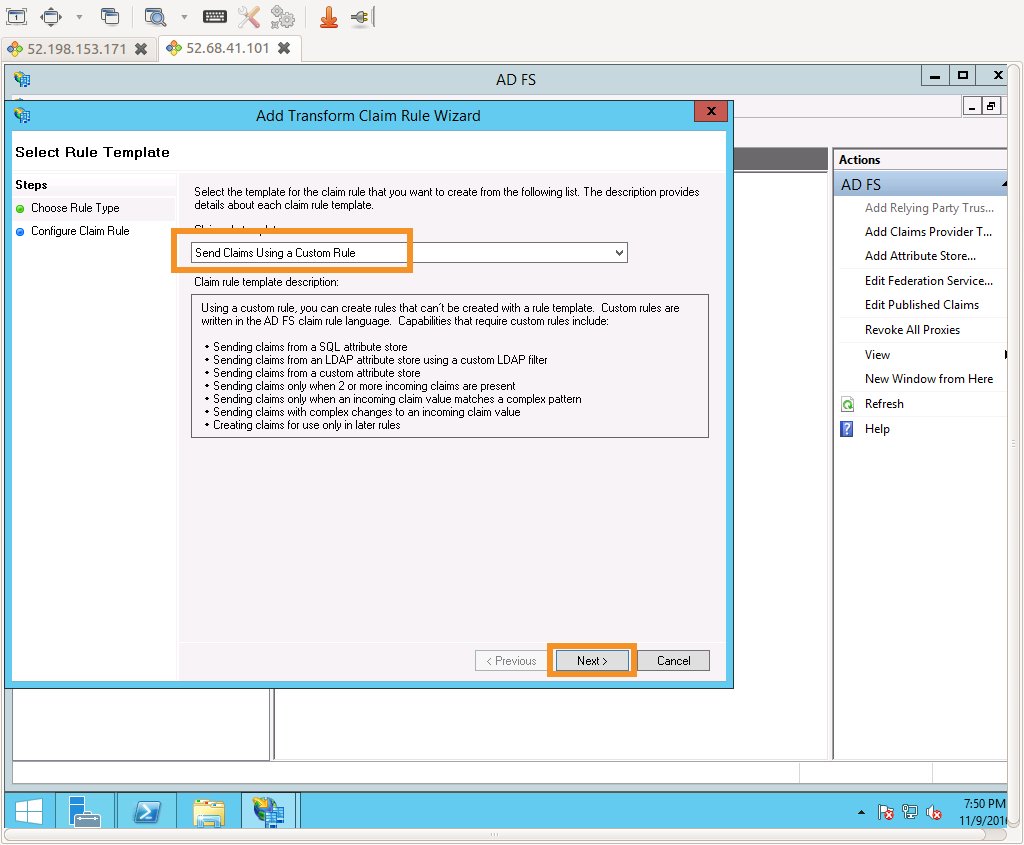


On the **Configure Claim Rule** page, type the following settings, and choose **OK**, as shown in the following screenshot.

|  |  |
| --- | --- |
| **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 |



Choose **Add Rule** to configure the third rule, and then choose **Send claims using a custom rule**. Finally, choose **Next**.

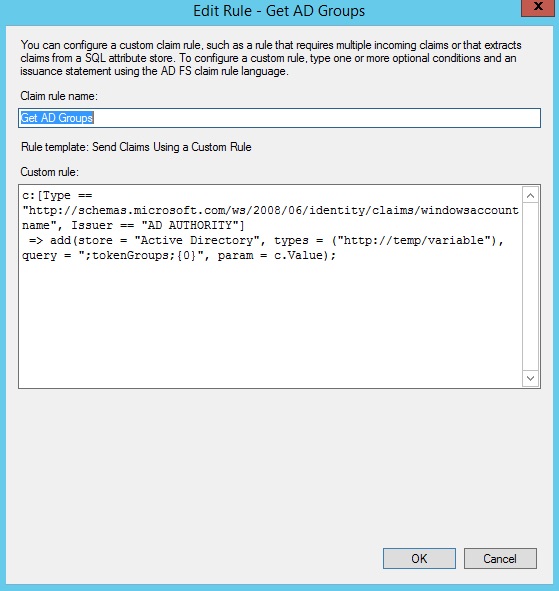


In the **Claim rule name** box, type **Get AD Groups**. In the **Custom rule** box, enter the following, and then choose **OK**.

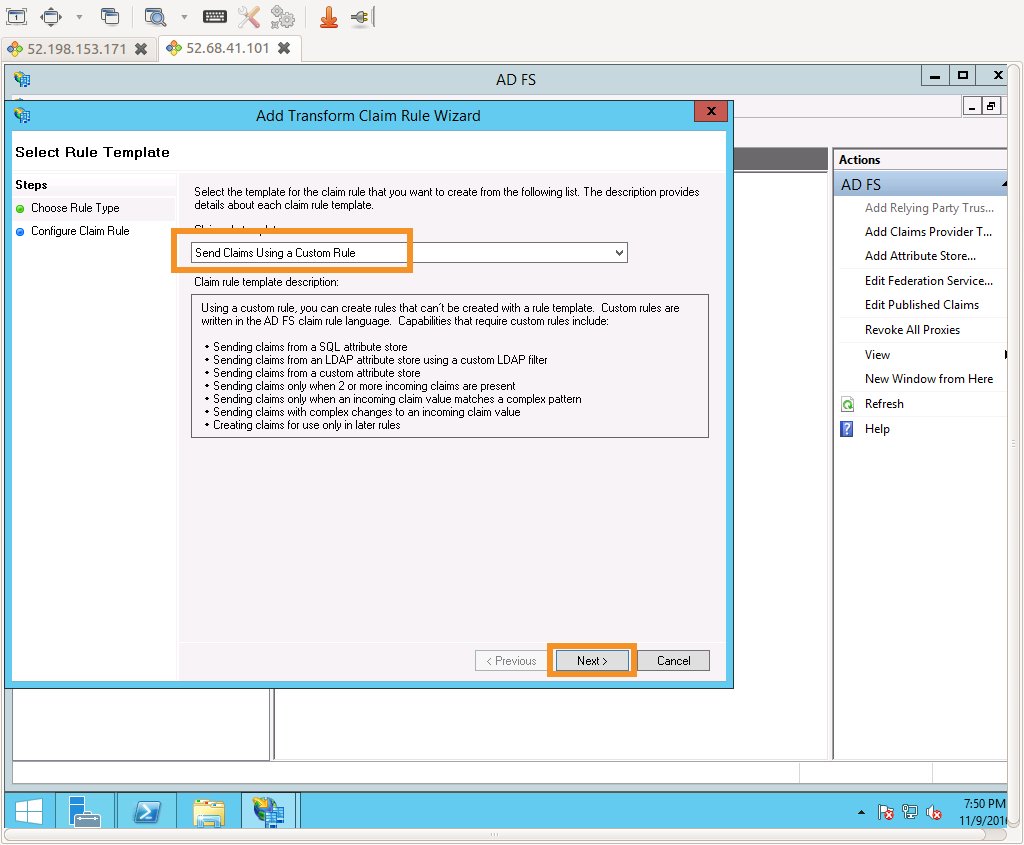
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"] => add(store = "Active Directory", types = ("http://temp/variable"), query = ";tokenGroups;{0}", param = c.Value);

This custom rule uses a script in the claim rule language that retrieves all the 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.

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



Choose **Add Rule** to configure the fourth and final rule. Then choose **Send claims using a custom rule**, and choose **Next**.



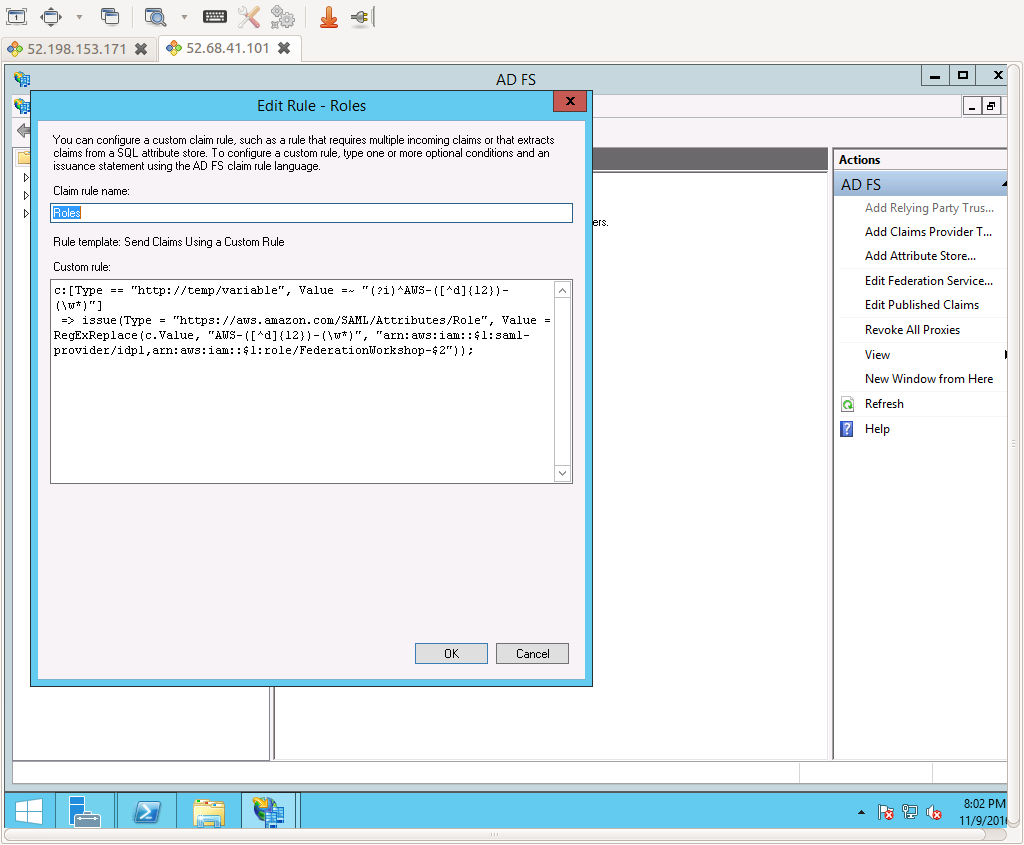
In the **Claim rule name** box, type **Roles**. In the **Custom rule** box, enter the following, and then choose **OK**.

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"));

This custom rule uses regular expressions to transform each of the group memberships 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 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.

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



Choose **OK** to complete the **Edit Claim Rules** wizard.

