Experiences with AWS and Shibboleth

Adventures in the use of Shibboleth and Amazon Web Service, with a little SAML and OpenLDAP thrown in for good measure

Mike Jones

Keywords: AWS; SAML; Cloud; Federated Access; IAM; IdP

Contents

1 Preamble

Mimas **http://mimas.ac.uk** has been involved in the Janet AWS Cloud pilot as a means to quickly migrate a number of our services onto AWS. Currently I have delegated to two people, the authority to provision AWS resources. This artificial limit was imposed, due to our need to instantiate many machines and keep abreast of the number of resources we have used and are using during the early phase of adoption of AWS.
Now we have our foot in the door, I want to enable Mimas service leaders and service technicians to access portions of the AWS console without my direct involvement. I want managers to allow service technicians to bring up and tear down instances at need, to alter the AWS firewall layer to their resources and perhaps to obtain their own API and SMTP credentials for monitoring and mail transport etc.
I can set up various users in the Arcus Portal **https://janet.arcusglobal.com/** but really this is a billing layer above the technical aspects of day to day AWS access, and I should be able to provide more fine grained access to AWS if I use AWS's access control mechanisms.
Here are some early notes on where we are in this process using V1.0 of the instructions provided by Chris Franks chris.franks@ncl.ac.uk available here: **https://community.ja.net/blogs/amazon-web-services-aws/document/guide-shibbolising-aws**. This exercise was to create a SAML provider for login to AWS using Mimas's Shibboleth Identity Provider (IdP). The rest of this document should not be read without reference to Chris' instructions.
Mimas has an IdP to help its staff, contractors and other associates authenticate to services (primarily also run at Mimas) which are protected by Shibboleth. This is a fairly straightforward set-up using OpenLDAP (for static-ish attributes) and MySQL (for on-the-fly transient attributes) as data sources. The basic configuration of OpenLDAP, MySQL and the IdP can be found elsewhere with some additional notes below.
After reading Chris' instructions I decided to create the relevant attributes in the IdP first, and test them on a known Service Provider (SP). Mimas runs an IdP and SP on the same host for just such an eventuality. This SP is not registered in UK Federation allowing us to tweak it without having to wait for Metadata refreshes (a trick I would heartily recommend for the community). Thus I started (Phase 1) with points 5, 6 and 7, with 7 adapted to the already existing `AttributeFilterPolicy` for our local SP. Once happy that the attributes were able to be obtained and sent correctly. I completed the configuration with Phase 2.
Phase 1 is tested and is working. Phase 2 required a little debugging, which I have described at the end of this document.

2 Phase 1, Release attributes to known test SP

2.1 Step 5, Add an `awsRoleSessionName` attribute

Add an `awsRoleSessionName` attribute into the IdP's configuration file `attribute-resolver.xml`
Note the id of the DataConnector for LDAP, in our case this is "MimasLDAP" from `<resolver:DataConnector id="MimasLDAP" xsi:type="LDAPDirectory" ... >`, a child node of `<resolver:AttributeResolver ... >`.

- `Add the following child to @<resolver:AttributeResolver ... >:`

```
<!-- Fow AWS session we'll use uid -->
<resolver:AttributeDefinition id="awsRoleSessionName"
                              xsi:type="ad:Simple"
                              sourceAttributeID="uid">
  <resolver:Dependency ref="MimasLDAP"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                             name="https://aws.amazon.com/SAML/Attributes/RoleSessionName"
                             friendlyName="RoleSessionName" />
</resolver:AttributeDefinition>
```

We use `uid` instead of the `mail` example, as in our case `uid` is unique and mail is not in our LDAP (we use the unique overlay in our LDAP configuration to force this uniqueness). Folk may choose any unique attribute which makes sense.

2.2 Step 6, Add an `awsRoles` attribute

Add an `awsRoles` attribute configuration also to the IdP's configuration file `attribute-resolver.xml`.
I did this directly after the `awsRoleSessionName AttributeDefinition` created in step 5. (Again note the `id` of the `DataConnector` for LDAP as in step 5).

```
<resolver:AttributeDefinition id="awsRoles" xsi:type="ad:Mapped" sourceAttributeID="memberOf">
  <resolver:Dependency ref="MimasLDAP"/>
  <resolver:AttributeEncoder xsi:type="enc:SAML2String"
                          name="https://aws.amazon.com/SAML/Attributes/Role"
                          friendlyName="Role" />
  <ad:ValueMap>
    <ad:ReturnValue>arn:aws:iam::NNNN:saml-provider/MimasIdP,arn:aws:iam::NNNN:role/$1</ad:
        ReturnValue>
    <ad:SourceValue>cn=AWS ([^,]*),.*</ad:SourceValue>
  </ad:ValueMap>
</resolver:AttributeDefinition>
```

where `NNNN` is my AWS account number.
See **http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html#arn-syntax-iam** for the ARN attribute format for the AWS Identity and Access Management (IAM) service.
Additional Notes. We use the attribute `memberOf` to provide this attribute. For this we needed to create an `ou=groups` branch in our Ldap Directory. I expect that most organisations will have this setup already. We needed to create various groups e.g. `cn=AWS admin,ou=groups,dc=mimas,dc=ac,dc=uk`
Chris used the example `AWS_Admin`, and created under (I think) an `AWS_Groups` organisational unit. For `CN=AWS_Admin,AWS_Groups,Web,Example,Org` I read `CN=AWS_Admin,ou=AWS_Groups,dc=web,dc=example,dc=org`
Thus I created new groups starting with my `AWS admin` group and populated it (I used a space instead of an underscore; just a personal preference):

```
dn: ou=groups,dc=mimas,dc=ac,dc=uk
objectclass: organizationalunit
ou: groups

dn: cn=AWS admin,ou=groups,dc=mimas,dc=ac,dc=uk
objectclass: groupofnames
cn: AWS admin
member: cn=Mike Jones,ou=people,dc=mimas,dc=ac,dc=uk
member: cn=Ann Other,ou=people,dc=mimas,dc=ac,dc=uk

# etc. etc.
```

OpenLDAP does not publish the attribute `memberOf` by default. In order for the Shibboleth attribute resolver to see this, two things need to be configured:

1. The member overlay needs to be active. I followed the instructions here:
   http://www.schenkels.nl/2013/03/how-to-setup-openldap-with-memberof-overlay-ubuntu-12-04/
2. The ldap query must explicitly request memberOf attribute. Therefore the `<resolver:DataConnector>` must have the child node `<ReturnAttributes>` and this must contain the string `memberOf` and as a consequence all of the other attributes used by shibboleth (I think) e.g.

```
<resolver:DataConnector id="MimasLDAP" ... >
    <FilterTemplate>...</FilterTemplate>
    <ReturnAttributes>uid cn mail memberOf
    eduPersonAffiliation eduPersionEntitlement</ReturnAttributes>
  </resolver:DataConnector>
```

I also changed the saml-provider name specifically to Mimas's IdP.

2.3 Step 7, Release the Amazon Attributes to (in this case my own Shibboleth SP)

I revisited this step later when I had ascertained that attributes were flowing correctly. On the first pass I added the two extra `<afp:AttributeRule>` nodes to the local SP's `<afp:AttributeFilterPolicy>` (child of `<afp:AttributeFilterPolicyGroup>`) thus:

```
  <afp:AttributeFilterPolicy id="release3attributesMimasIdPSP">
    <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
                           value="https://idp.mimas.ac.uk/shibboleth" />
    <afp:AttributeRule attributeID="eduPersonScopedAffiliation">
      <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>
    <afp:AttributeRule attributeID="eduPersonTargetedID">
      <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>
    <afp:AttributeRule attributeID="eduPersonTargetedID.old">
      <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>
    <afp:AttributeRule attributeID="eduPersonPrincipalName">
      <afp:PermitValueRule xsi:type="basic:ANY" />
    </afp:AttributeRule>
<!-- Put the AWS attributes here for Mimas SP -->
    <afp:AttributeRule attributeID="awsRoles">
      <afp:PermitValueRule xsi:type="basic:ANY"/>
    </afp:AttributeRule>
    <afp:AttributeRule attributeID="awsRoleSessionName">
      <afp:PermitValueRule xsi:type="basic:ANY"/>
    </afp:AttributeRule>
  <afp:AttributeFilterPolicy>
```

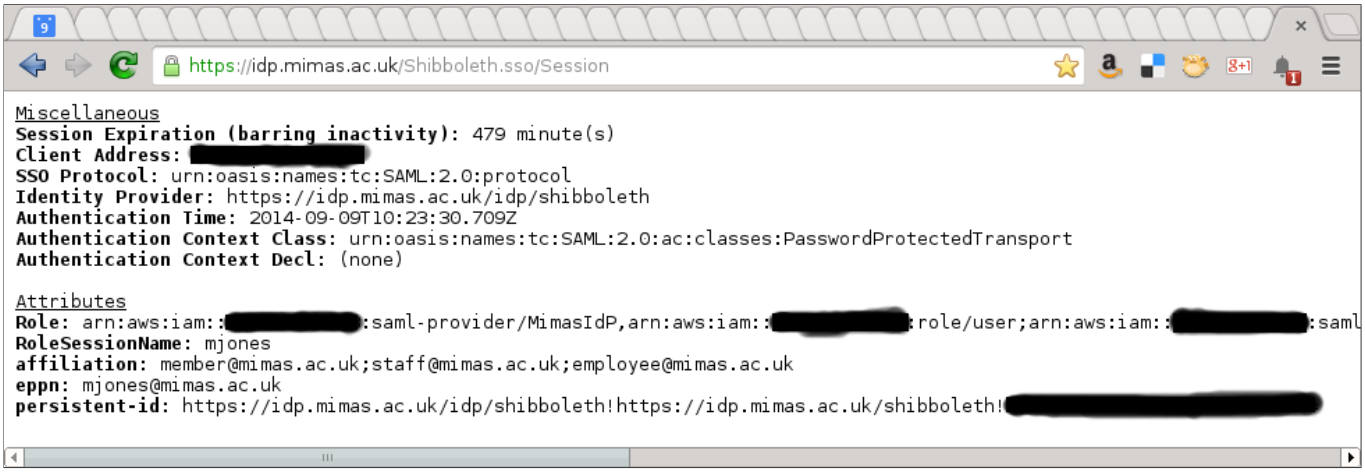
2.4 Finalisation of Phase 1

I restarted the IdP service to load the new configuration.
In my SP I needed to add two new `<Attribute>` node attribute mappings in the `attribute-map.xml` file under the root `<Attributes>` node.

```
  <Attribute name="https://aws.amazon.com/SAML/Attributes/RoleSessionName"
             id="RoleSessionName">
    <AttributeDecoder xsi:type="StringAttributeDecoder"
                      caseSensitive="false"/>
  </Attribute>
```

```
<Attribute name="https://aws.amazon.com/SAML/Attributes/Role"
           id="Role">
  <AttributeDecoder xsi:type="StringAttributeDecoder"
                    caseSensitive="false"/>
</Attribute>
```

I restarted `shibd`.
Logging in and visiting https://idp.mimas.ac.uk/Shibboleth.sso/Session I saw:



3 Phase 2, Release attributes to AWS

3.1 Step 1, Add the AWS relying party

Add a new <rp:RelyingParty> node child of <rp:RelyingPartyGroup>) to relying-party.xml in the IdP configuration (after the <DefaultRelyingParty> child).
(I was confused for a time by the term "under" in Chris' instructions, so I reworded it above, as you can see).
Various attributes are set to their default value so the following sufficed for our configuration:

```
<!-- RP Config for AWS -->
<rp:RelyingParty id="urn:amazon:webservices"
                 provider="https://idp.mimas.ac.uk/idp/shibboleth"
                 defaultSigningCredentialRef="IdPCredential">
  <rp:ProfileConfiguration xsi:type="saml:SAML2SSOProfile"
                           signResponses="never"
                           signAssertions="always"
                           encryptAssertions="never"
                           maximumSPSessionLifetime="PT1H" />
</rp:RelyingParty>
```

3.2 Step 2, Add AWS Metadata provider to the relying-party.xml

I followed the instructions changing the backingFile path.

3.3 Step 3, Ensure the Unsolicited Login Handler is setup in handler.xml

Look for

```
/ph:ProfileHandlerGroup/ph:ProfileHandler[ph:RequestPath="/SAML2/Unsolicited/SSO"]
```

In my set-up this is already defined and active.

3.4 Step 4, Ensure the Unsolicited Login Handler is defined

Ensure the corresponding entry for the Unsolicited Login Handler is defined in internal.xml.
(again "underneath" was a little ambiguous, I've tried to make a bit more explicit; below)
Make sure the <bean> node with id="shibboleth.UnsolicitedSSODecoder" exists in an <entry> node under the <util:map id="shibboleth.MessageDecoders"> node. Make sure it has a sibling <key> node which has the <value> node with text urn:mace:shibboleth:2.0:profiles:AuthnRequest.
In other words (and forgive my amateur xpath query), this element needs to exist:

```
/beans/util:map[@id="shibboleth.MessageDecoders"]/entry
 [key/value="urn:mace:shibboleth:2.0:profiles:AuthnRequest" and
   bean/@id="shibboleth.UnsolicitedSSODecoder"]
```

3.5 Step 5, Add an awsRoleSessionName attribute

Add an awsRoleSessionName attribute into the IdP's configuration file attribute-resolver.xml.
Nothing needs changing from the configuration in Phase 1 above.

3.6 Step 6, Add an `awsRoles attribute`

Add an `awsRoles attribute` configuration also to the IdP's configuration file `attribute-resolver.xml`. Nothing needs changing from the configuration in Phase 1 above.

3.7 Step 7, Release the Amazon Attributes to (in this case AWS)

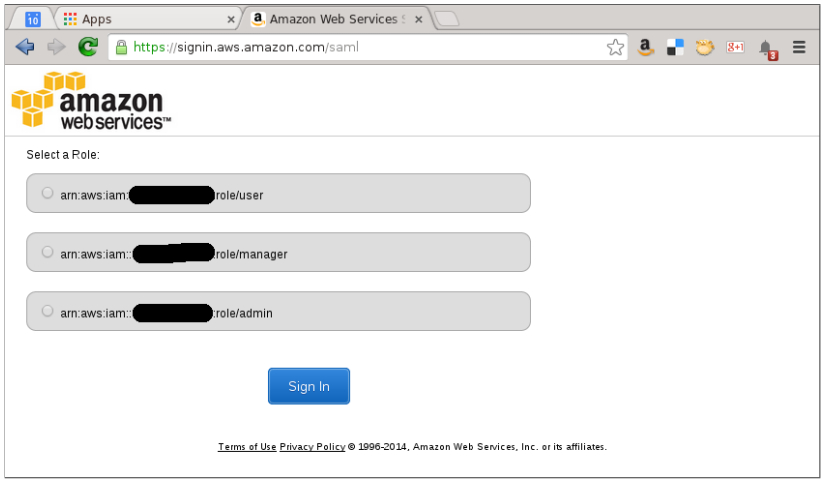
Add the a new child node `<afp:AttributeFilter>` to `<AttributeFilterPolicyGroup>`.

```
<!-- Put the AWS attributes here for AWS -->
    <afp:AttributeFilterPolicy id="release2attributesAWS">
      <afp:PolicyRequirementRule xsi:type="basic:AttributeRequesterString"
                            value="urn:amazon:webservices" />
      <afp:AttributeRule attributeID="awsRoles">
        <afp:PermitValueRule xsi:type="basic:ANY"/>
      </afp:AttributeRule>
      <afp:AttributeRule attributeID="awsRoleSessionName">
        <afp:PermitValueRule xsi:type="basic:ANY"/>
      </afp:AttributeRule>
    </afp:AttributeFilterPolicy>
```

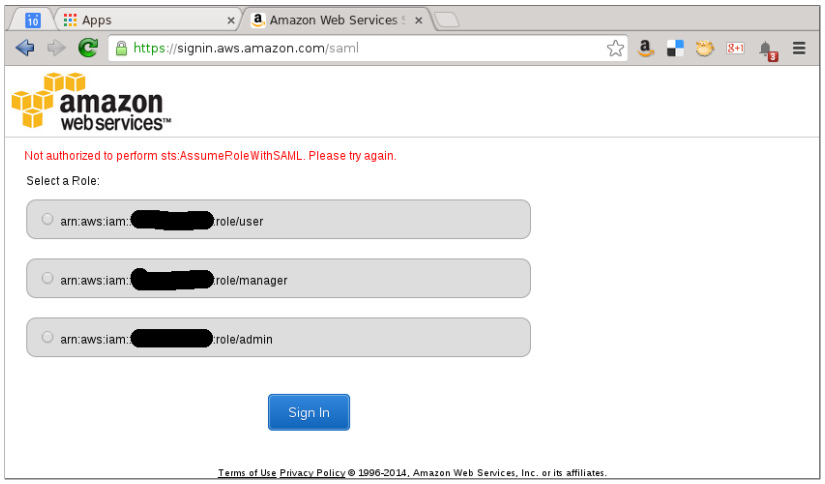
NB if you have an `id` attribute in the `AttributeFilterPolicy`, do not replicate it by accident. This will cause the IdP to overwrite the previously defined `AttributeFilterPolicy` with the same `id`.
At this point the IdP needs to be restarted.

3.8 Steps 8, 9 and 10

Steps 8, 9 and 10 were followed more or less as per Chris' instructions.
In Step 6, in my `<ad:ReturnValue>` I had specified `saml-provider/MimasIdP`. Therefore in step 8 I specified the provider name as "MimasIdP".
Steps 8 and 9 merged into one step as the AWS console suggested defining the roles immediately. I defined "manager", "admin" and "user", setting up initial policies appropriately to access EC2 (I guessed them) **this turned out to be a mistake, see below**.
Step 10. This initially failed for me as I hadn't replaced `AWSAccountID` (`NNNN` in the notes above) in all places in step 6. The error message was fairly obvious, IIRC it said that something like the Amazon Account Number was not valid. I checked on my local SP and indeed I had changed one occurrence but not both in the attribute definition. After changing this I now am able to authenticate.



but in assuming a role I receive "Not authorized to perform sts:AssumeRoleWithSAML. Please try again."



... See below for the solution to this ...

3.9 The SAML message

If I examine the POSTed SAML message to AWS I see the following (I've sanitized prettified it):

```
<?xml version="1.0" encoding="UTF-8"?>
<saml2p:Response xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol"
                 Destination="https://signin.aws.amazon.com/saml"
                 ID="_7bf1ca4c75a128ee636c4b3371137791"
                 IssueInstant="2014-09-09T15:21:58.548Z"
                 Version="2.0">
  <saml2:Issuer xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
                Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://idp.mimas.ac.uk/idp/shibboleth
  </saml2:Issuer>
  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    <ds:SignedInfo>
      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
      <ds:Reference URI="#_dc9d8ec0ea1d77a23a489ea20e8f7839">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
            <ec:InclusiveNamespaces xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"
                                    PrefixList="xs"/>
          </ds:Transform>
        </ds:Transforms>
        <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
        <ds:DigestValue>AOuO7ZHOa7V2m1cPOAxFrOA6CkQ=</ds:DigestValue>
      </ds:Reference>
    </ds:SignedInfo>
    <ds:SignatureValue /> <!-- removed my hand -->
    <ds:KeyInfo>
      <ds:X509Data>
```

```
      <ds:X509Certificate>
        MIIDJzCCAg+gAwIBAgIUExVE+BsgxiwDLoOHU9bL+UpPdVEwDQYJKoZIhvcNAQEFBQAwGjEYMBYG
        A1UEAxMPaWRwLm1pbWFzLmFjLnVrMB4XDTEwMTAwNDE3MDczOFoXDTMwMTAwNDE3MDczOFowGjEY
        MBYGA1UEAxMPaWRwLm1pbWFzLmFjLnVrMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA
        4w7U5gd6/ifZvu0vwr4rK9zY08DOuxfZDkOAoXnfL6w31loHJQaM1ezLKs9RxVgHk++2vzFUn5Y7
        0Bj7vOEOtaevjsKUVrnJZGpj7pwRtRZinpVUjlT/a8sQUhjDjnag1RceLYthPG8BTd5whlWGLl0+
        aQHDQbwdPcRqEml+6TZG8sxTZIukWqtdTCzVMl+EYfdclCqDKUFMXm3Uz15OzUje53BvWIaX2Thm
        UyqiDPHPGCfknwtcsKYcc3ztyokCamTH7U5FuxJMkTlzrieXJvHGTRjZuDHQhN6ulvr7/5Zt1v9O
        ngZxptFc5ICUe8YqLelBgpWfeekXwLgP2AOzVQIDAQABo2UwYzBCBgNVHREEOzA5gg9pZHAubWlt
        YXMuYWMudWuGJmh0dHBzOi8vaWRwLm1pbWFzLmFjLnVrL2lkcC9zaGliYm9sZXRoMB0GA1UdDgQW
        BBRoLGk6jr37RIvGJJ9mNu2cx/xIRDANBgkqhkiG9w0BAQUFAAOCAQEAxv3qV8i0hLx/UCzgEerr
        Ds0mrMQi2ZZYQQAAfrZmMSAryZu2yBc87Ibku5dbKps89fA47KYG+Fro/ixdx1DTrkod2Gwz28MM
        Ra3WTp3KxOq7NMCSrsk7OEEwyeYo3Ozw1Evmm4LRh3i0MpMNo9SunlU9XfjkUO2RBp6YWrNL3uOW
        rdskey+qbXzhnQNGnLqpsL1MDI/tmrWxsyxcjnEyG4F7FSkufvrpFwCGZkEH+4xIDCEGFTxBWBy7
        Ex/MJ7ltPBeOwej+omCGkJxJcAbzaBsM+5AMfRhXAFKoBmWmKAGSKfZDAIj7HXfN/kBwjy0PyOt8
        kPDXrvqGmveft0zUvg==
      </ds:X509Certificate>
    </ds:X509Data>
  </ds:KeyInfo>
</ds:Signature>
<saml2p:Status>
  <saml2p:StatusCode Value="urn:oasis:names:tc:SAML:2.0:status:Success"/>
</saml2p:Status>
<saml2:Assertion xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion"
                 ID="_dc9d8ec0ea1d77a23a489ea20e8f7839"
                 IssueInstant="2014-09-09T15:21:58.548Z"
                 Version="2.0"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <saml2:Issuer Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity">
    https://idp.mimas.ac.uk/idp/shibboleth
  </saml2:Issuer>
  <saml2:Subject>
    <saml2:NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-format:transient"
                  NameQualifier="https://idp.mimas.ac.uk/idp/shibboleth"
                  SPNameQualifier="urn:amazon:webservices">
      _d5af4d9bf13857f0118bfea5ce1bfc49
    </saml2:NameID>
    <saml2:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
      <saml2:SubjectConfirmationData Address="NNN.NNN.NNN.NNN"
                                     NotOnOrAfter="2014-09-09T15:26:58.548Z"
                                     Recipient="https://signin.aws.amazon.com/saml"/>
    </saml2:SubjectConfirmation>
  </saml2:Subject>
  <saml2:Conditions NotBefore="2014-09-09T15:21:58.548Z"
                    NotOnOrAfter="2014-09-09T15:26:58.548Z">
    <saml2:AudienceRestriction>
      <saml2:Audience>
        urn:amazon:webservices
      </saml2:Audience>
    </saml2:AudienceRestriction>
  </saml2:Conditions>
  <saml2:AuthnStatement AuthnInstant="2014-09-09T15:21:58.459Z"
```

```
                        SessionIndex="_454f8e98e056537400ae9e62069c4b2f"
                        SessionNotOnOrAfter="2014-09-09T16:21:58.546Z">
      <saml2:SubjectLocality Address="NNN.NNN.NNN.NNN"/>
      <saml2:AuthnContext>
        <saml2:AuthnContextClassRef>
          urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
        </saml2:AuthnContextClassRef>
      </saml2:AuthnContext>
    </saml2:AuthnStatement>
    <saml2:AttributeStatement>
      <saml2:Attribute FriendlyName="Role"
                       Name="https://aws.amazon.com/SAML/Attributes/Role"
                       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:type="xs:string">
          arn:aws:iam::MMMMMMMMMMM:saml-provider/MimasIdP,arn:aws:iam::MMMMMMMMMMM:role/user
        </saml2:AttributeValue>
        <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:type="xs:string">
          arn:aws:iam::MMMMMMMMMMM:saml-provider/MimasIdP,arn:aws:iam::MMMMMMMMMMM:role/manager
        </saml2:AttributeValue>
        <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:type="xs:string">
          arn:aws:iam::MMMMMMMMMMM:saml-provider/MimasIdP,arn:aws:iam::MMMMMMMMMMM:role/admin
        </saml2:AttributeValue>
      </saml2:Attribute>
      <saml2:Attribute FriendlyName="RoleSessionName"
                       Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName"
                       NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                              xsi:type="xs:string">
          myuid
        </saml2:AttributeValue>
      </saml2:Attribute>
    </saml2:AttributeStatement>
  </saml2:Assertion>
</saml2p:Response>
```

4 Fixing the stsAssumeRole Error

It turns out that during the creation of roles it is necessary to allow SAML assertions explicit access to the `AssumeRole` operation. This can be done at the time of the role's creation or afterwards.

4.1 Amending an existing Trust Policy to allow SAML based authorisation

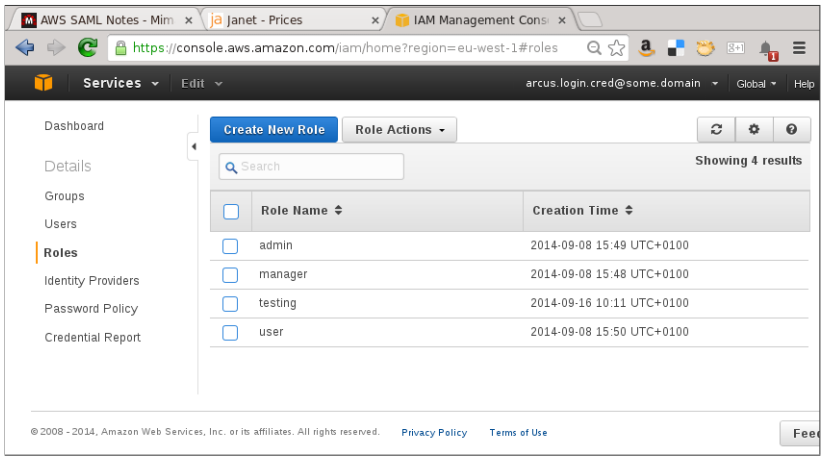
In the IAM section of the console, after selecting a role, the trust policy which may look like mine:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

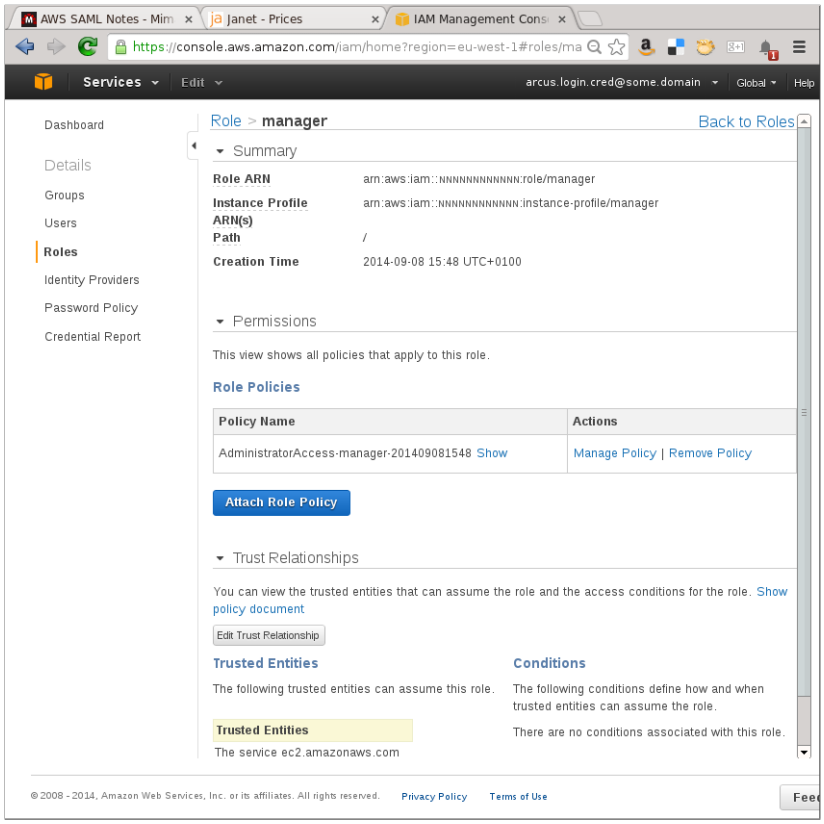
should be edited to look something more like this:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::NNNNNNNNNNNN:saml-provider/MimasIdP"
      },
      "Action": "sts:AssumeRoleWithSAML",
      "Condition": {
        "StringEquals": {
          "SAML:aud": "https://signin.aws.amazon.com/saml"
        }
      }
    }
  ]
}
```

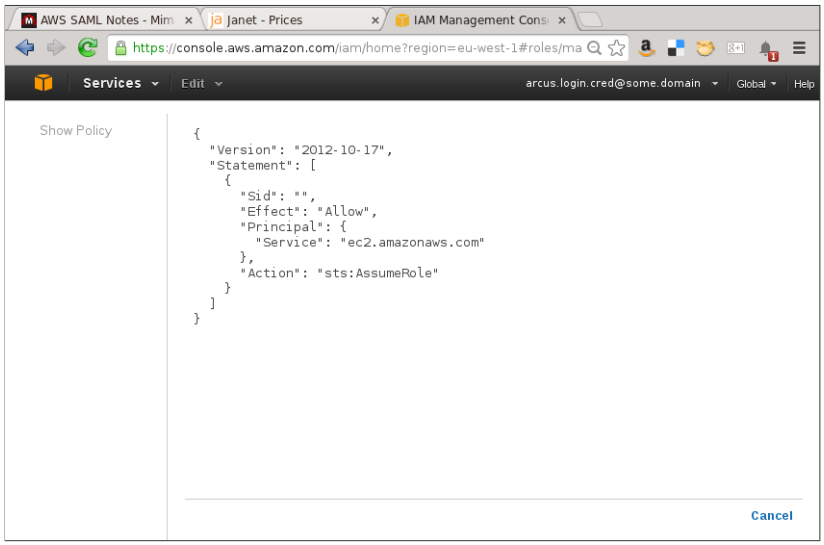
In the IAM AWS console under Roles you should see any roles that have been defined for your AWS account.



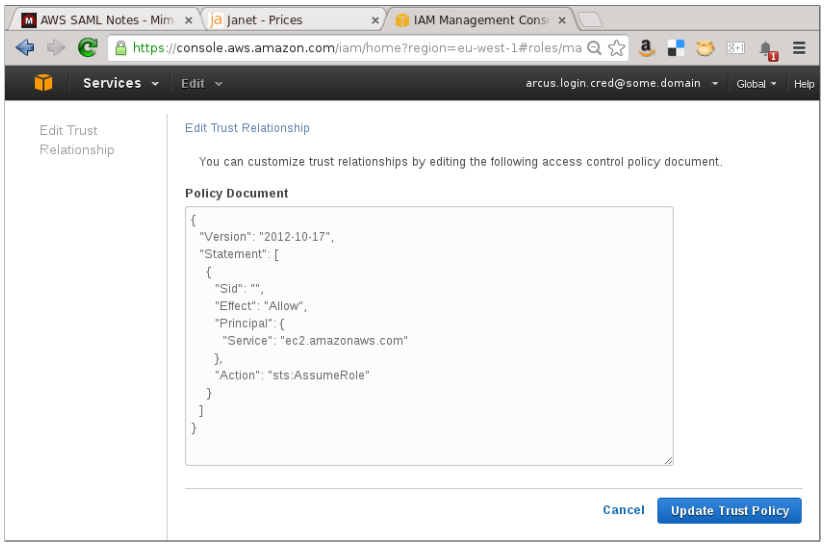
Clicking on a role reveals more information and perhaps (hopefully) the possibility to edit the role



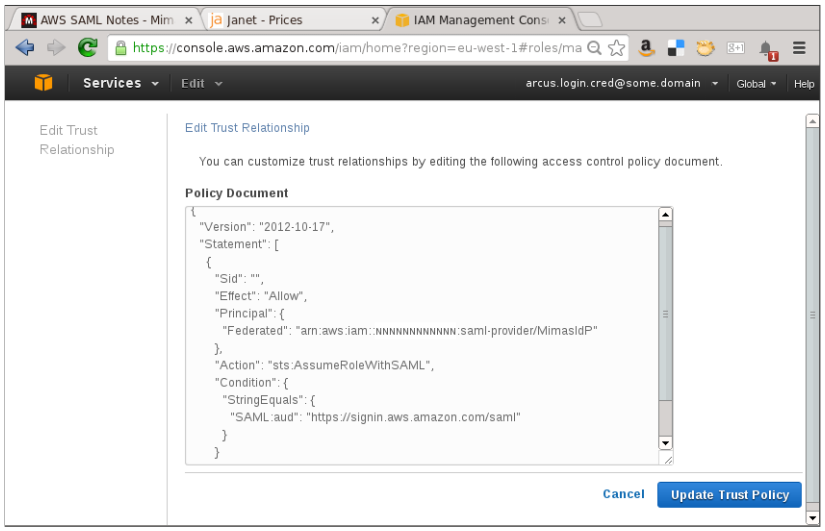
Here we are interested in the Trust Relationships section. Click on the "Show Policy Document" link:



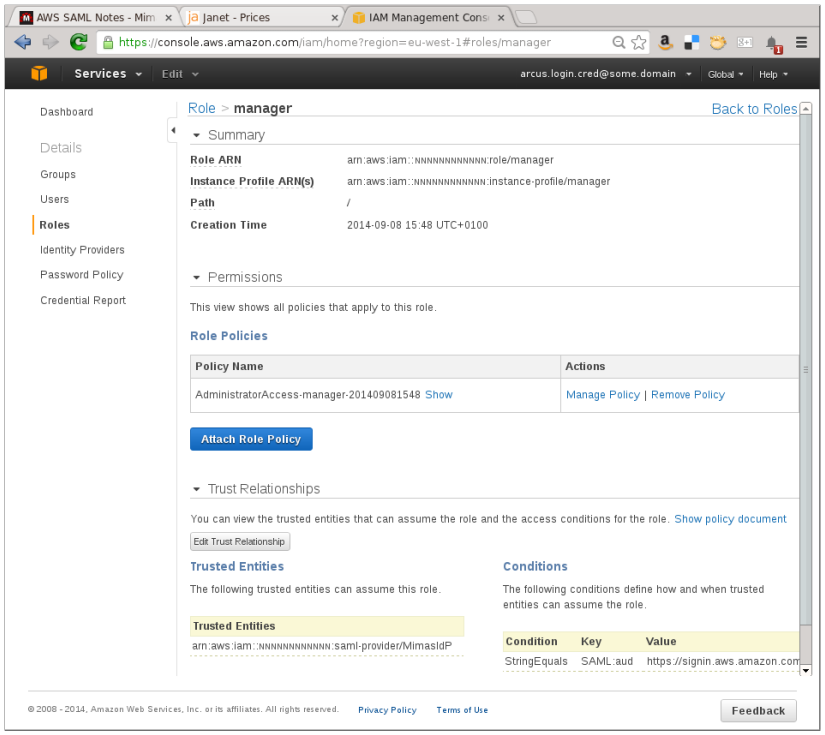
To change this, return to the Roles tab, click the role and then click on the Edit Trust Relationship button.



Replace the incorrect JSON policy

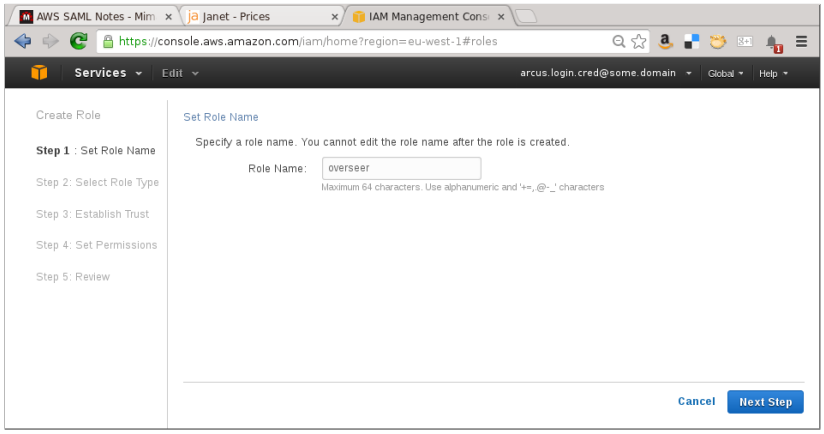


Save (then also reload the page) and you should see a different set of trusted entities at the bottom of the page.

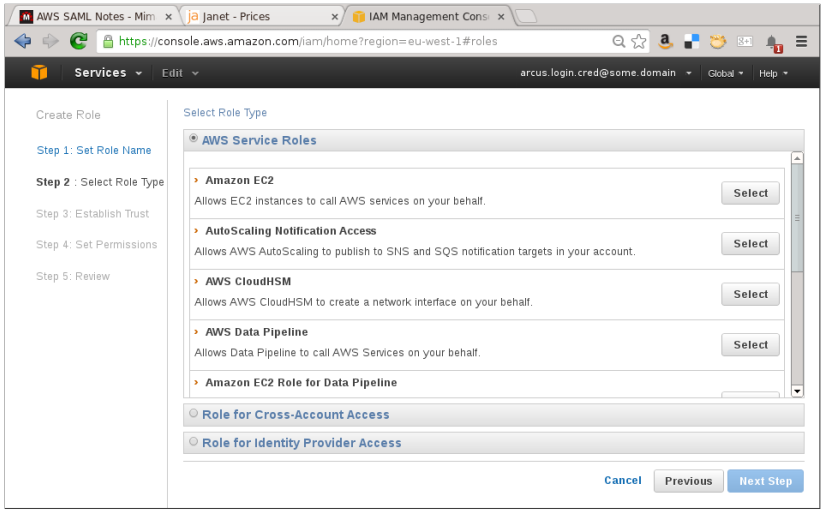


4.2 Using the AWS console to create an appropriate SAML Trust Policy at role creation.

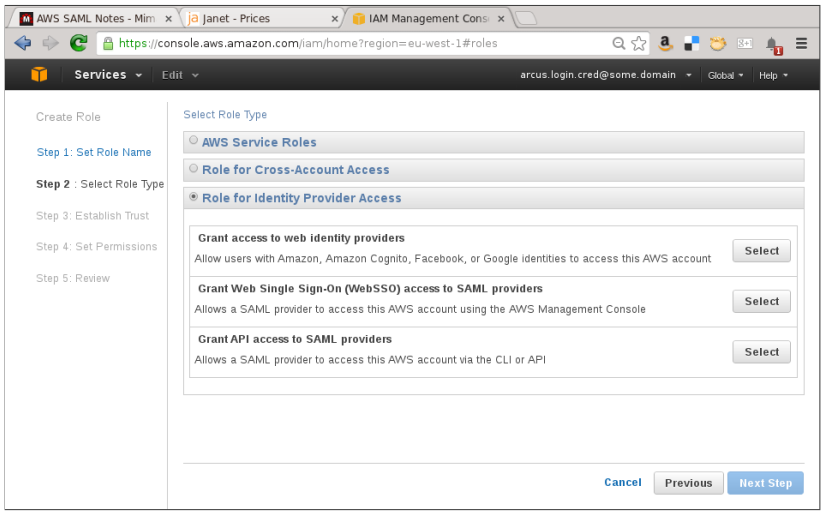
We'll create a role called "overseer", by entering "overseer" in the text entry box and clicking "Next step". This role is to allow management safe viewing of our activity (I hope).



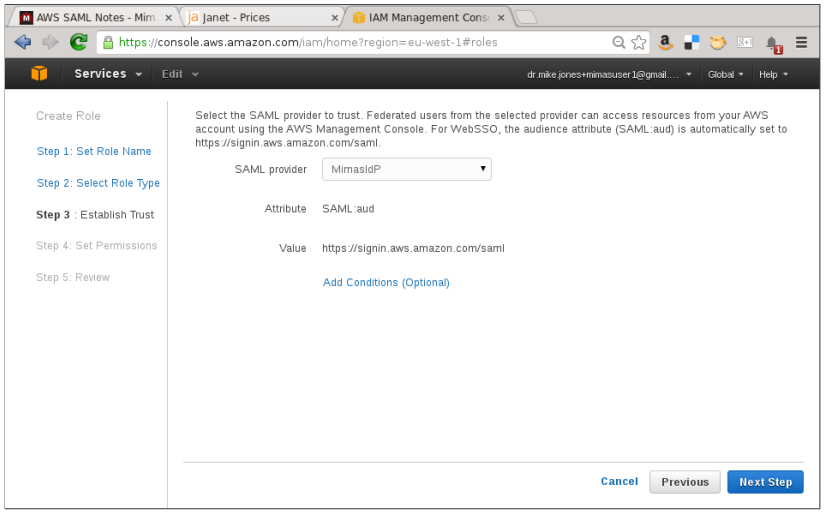
We see a set of different entities/roles (the entities' *principals* if you prefer.). This is where I went wrong. I thought that I should select one or more of these as the roles for an incoming SAML user to assume. What AWS mean here is "select the roles which may assume the role defined above". Initially, I selected Amazon EC2; I wished to grant the rights for my coworkers to instantiate to EC2 VMs using the newly created role. If you did this then follow instructions here: 4.1 to rectify this.
For incoming Shibboleth authentications to use this role we need to select the Role for "Identity Provider Access" menu



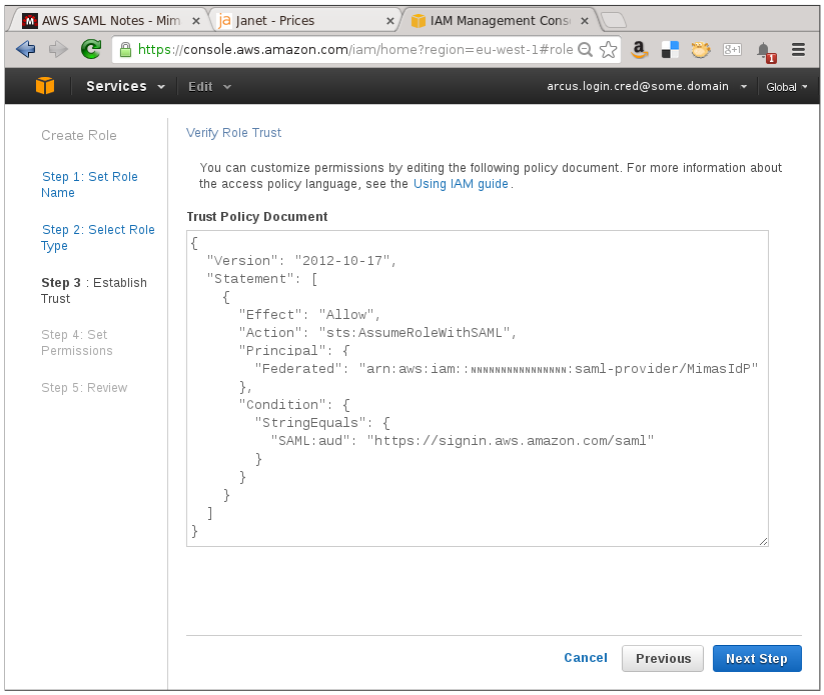
Next, grant the rights for our Shibboleth users to assume this role by selecting "Grant Web Single Sign-On (WebSSO) Access to SAML Providers".



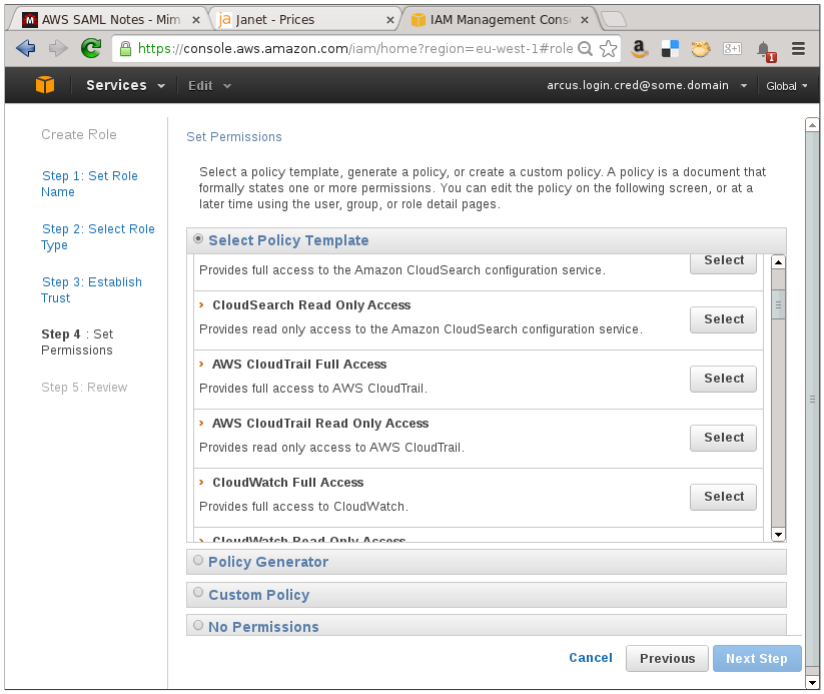
You now have the option to apply conditions. I didn't apply conditions at this point (Not sure what the conditional trust and scope would look like yet. We're just setting how we trust the Shibboleth IdP for this role here, not the access rights. Perhaps e.g. the IdP is not trustable outside the hours of 0900-1700).



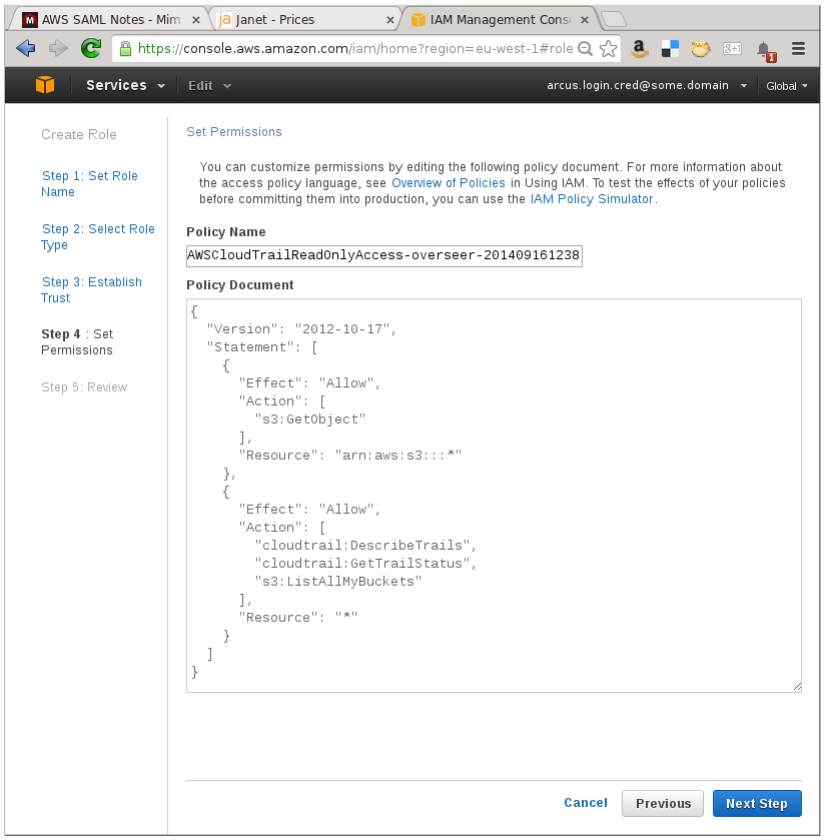
AWS shows you the trust policy you have created, which is nice.



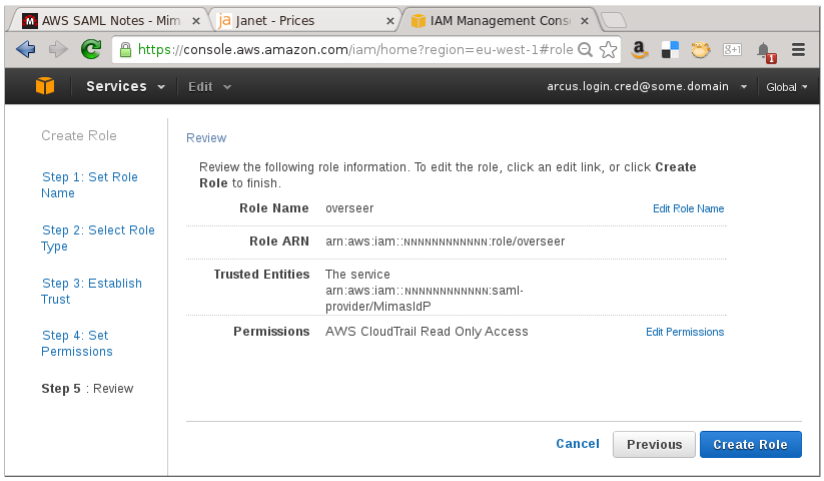
Next you can set the permissions from a template, via a wizard (Policy Generator), or by hand. I chose the AWS CloudTrail Read only for this role.



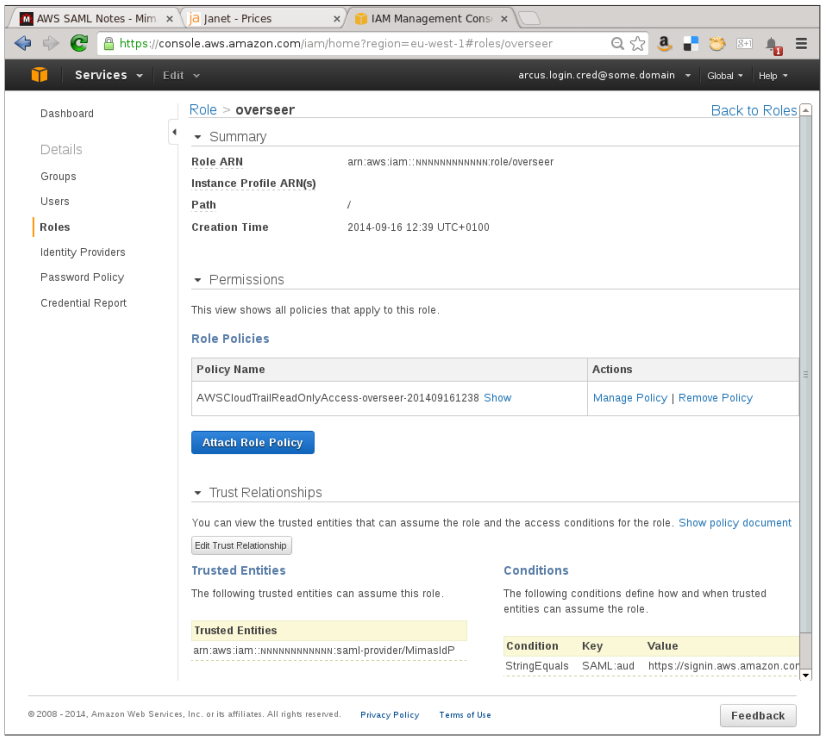
We get to see this policy, which is nice too.



We get to review a summary before applying it.



On completion, we are presented with the role overview/edit page. The role is now active and if your IdP asserts a SAML assertion which contains this role the user may assume this role when logging in and for the duration of the session (for up to 1 hour at a time).


AWS SAML Notes - Mim
Janet - Prices
IAM Management Cons
https://console.aws.amazon.com/iam/home?region=eu-west-1#roles/overseer
Services
Edit
arcus.login.cred@some.domain
Global
Help
Dashboard
Details
Groups
Users
Roles
Identity Providers
Password Policy
Credential Report
Role > overseer
Back to Roles
Summary
Role ARN
arn:aws:iam::NNNNNNNNNNN:role/overseer
Instance Profile ARN(s)
Path
/
Creation Time
2014-09-16 12:39 UTC+0100
Permissions
This view shows all policies that apply to this role.
Role Policies
Policy Name
Actions
AWSCloudTrailReadOnlyAccess-overseer-201409161238 Show
Manage Policy | Remove Policy
Attach Role Policy
Trust Relationships
You can view the trusted entities that can assume the role and the access conditions for the role. Show policy document
Edit Trust Relationship
Trusted Entities
The following trusted entities can assume this role.
Trusted Entities
arn:aws:iam::NNNNNNNNNNN:saml-provider/MimasIdP
Conditions
The following conditions define how and when trusted entities can assume the role.
Condition
Key
Value
StringEquals
SAML:aud
https://signin.aws.amazon.com
© 2008 - 2014, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
Feedback