Contents

[Developing Java Application for Windows Azure 2](#_Toc322543797)

[Software environment 2](#_Toc322543798)

[Developing Java application 2](#_Toc322543799)

[Deploy application on Windows Azure 2](#_Toc322543800)

[Installing Windows Azure plugin for eclipse 2](#_Toc322543801)

[Windows Azure Project 4](#_Toc322543802)

[Getting source code 6](#_Toc322543803)

[Importing projects 6](#_Toc322543804)

[A) Using Eclipse’s File -> Import 6](#_Toc322543805)

[B) Using Git plugin 8](#_Toc322543806)

[Creating certificate files 9](#_Toc322543807)

[ACS Workflow 11](#_Toc322543808)

[Deploy CloudNinja STS 12](#_Toc322543809)

[Configure ACS 13](#_Toc322543810)

[Creating WAR files 20](#_Toc322543811)

[ACS Configuration 20](#_Toc322543812)

[Other (non-ACS) configuration changes are as below: 25](#_Toc322543813)

[Deploying application 28](#_Toc322543814)

# Developing Java Application for Windows Azure

## Software environment

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. no.** | **Name of the software** | **Version** | **Type** |
| 1 | Windows | 7 | OS |
| 2 | Windows Azure AppFabric SDK | 1.5 | SDK |
| 3 | Windows Azure SDK | 1.6 | SDK |
| 4 | JDK | 1.7.0 | SDK |
| 5 | .Net Framework | 4.0 | SDK |
| 6 | Visual Studio | 2010 | IDE |
| 7 | Apache Tomcat | 7.0 | Server |
| 8 | IIS | 7.0 | Server |
| 9 | SQL management console | 2008 R2 | IDE |
| 10 | Github Code repository |  | Repository |
| 11 | jqPlot | jquery.jqplot.1.0.0b2\_r1012 | Chart library |
| 12 | Eclipse Helios | 3.6 | IDE |
| 13 | Spring MVC | 3.0 | Framework |
| 14 | ORM Hibernate | 3.0 | Framework |
| 15 | PMD static memory analytics |  | Tool for analysis |

## Developing Java application

The java application will be developed using eclipse helios IDE, that can be downloaded from

<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/SR2/eclipse-jee-helios-SR2-win32.zip>

If JDK is not already installed, you must install JDK and add it to the PATH in environment variables before launching Eclipse.

1. Set JAVA\_HOME=‘<JDK\_install\_dir>’
2. Add %JAVA\_HOME%\bin to PATH

## Deploy application on Windows Azure

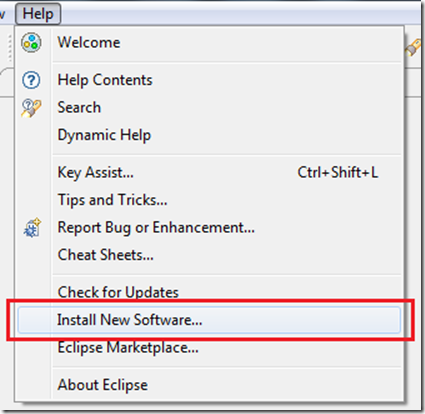
To deploy application on azure we need to package it.

We can package the application u*sing windows azure plugin for eclipse.*

Installing Windows Azure plugin for eclipse

This plugin can be used to create a project which makes use of windows azure starter kit.

1. Go to Help > Install New Software

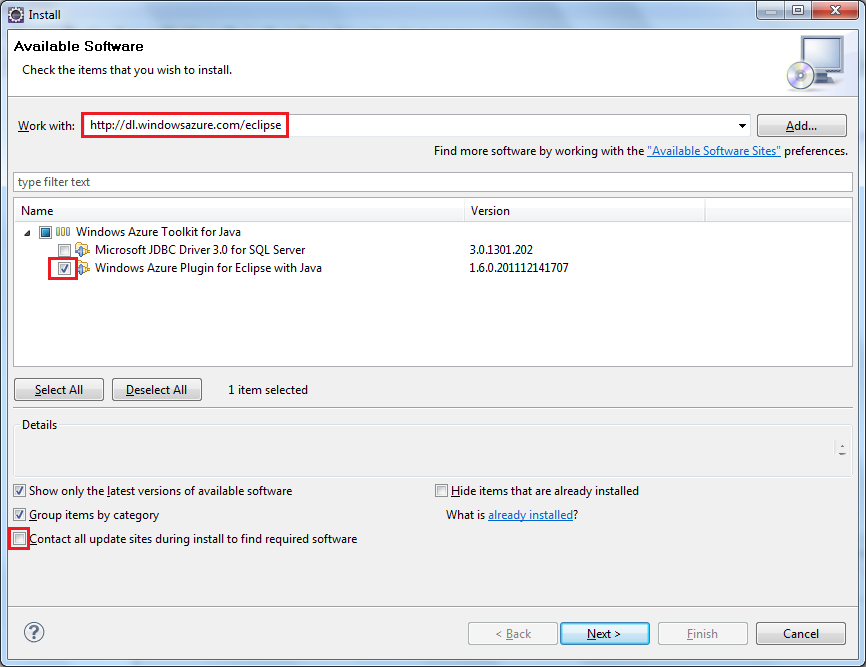


1. In ‘Work with’ box provide URL as below

<http://dl.windowsazure.com/eclipse>

1. The installer will check Windows Azure Plugin for Eclipse with Java and click Next –

Note - Please uncheck "*Contact all update sites during install to find required software*”.



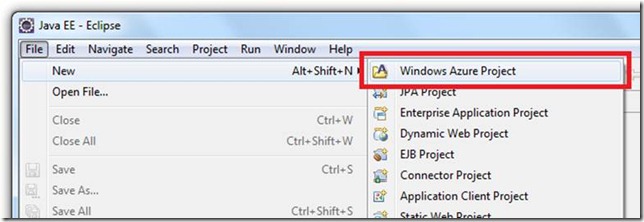
1. Select the checkbox for azure plugin click on Next
2. Accept license to install plugin
3. Restart Eclipse after installation.

## Windows Azure Project

1. You can create a new Windows Azure project or can use the one provided at

<https://github.com/PersistentSys/cloudninja-for-java/tree/domain/WindowsAzureProject>. This project has Remote Access disabled by default.

1. If you already have installed the plugin, you will see new Project Type – “*Windows Azure project*”. Create new project with this template



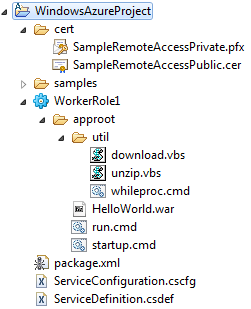
If this project type is not visible, then you can find it in *File -> New -> Other -> type “windows azure” in Wizards text box*. You will see *Windows Azure Project-> Windows Azure Project* option. Select it and click on Finish button.

Second option is to reset the perspective, *Window -> Reset Perspective*. After doing this you may see the project type.

1. This template will have all required project structure and settings required for the Windows Azure deployment along with the default role.

For Java application, the best type is Worker Role. Role provides mechanism to run Java application and communication with Azure runtime. If you are targeting for different role, you can configure it here. We can change three settings for the role – name, VM size and instances.

1. We will be using “WorkerRole1” as worker role for deploying the project (default settings).
2. The project structure contains following main components
   1. .cspack.jar - This contains the Java implementation of the windowsazurepackage, Ant builder task tag referenced in package.xml.
   2. ServiceConfiguration.cscfg - This is a sample Azure service configuration file.
   3. ServiceDefinition.csdef - This is a sample Azure service definition file.
   4. cert\SampleRemoteAccessPrivate.pfx - the sample certificate used by Windows Azure project. Replace it with your own certificate using the functionality in the *“Remote Access”* page of the project properties dialog (in the Windows Azure section).
   5. cert\SampleRemoteAccessPublic.cer - this based on the sample certificate (SampleRemoteAccessPrivate.pfx), having the public key only. You should replace this with your own certificate.
   6. startup.cmd: This Windows script will run each time your Worker Role starts.
3. If you create a new Windows Azure Project then the structure would be as below :



## Getting source code

The source code is available at –

<https://github.com/PersistentSys/cloudninja-for-java/tree/domain>

Get latest code from the github. For detailed steps please visit following URLs

<http://help.github.com/win-set-up-git/>

<http://www.vogella.de/articles/Git/article.html>

<http://kylecordes.com/2008/git-windows-go>

Github plugin for eclipse can be installed from:

<http://download.eclipse.org/egit/updates>

This can be installed by following the steps from Installing Windows Azure plugin for eclipse section.

Repository link:

[username@github.com:PersistentSys/cloudninja-for-java.git](mailto:username@github.com:PersistentSys/cloudninja-for-java.git)

*(Here username will be replaced by your github username.)*

Make sure that you connect via https to the github repository.

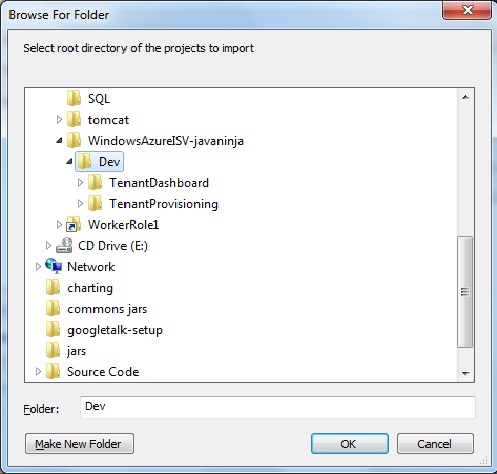
## Importing projects

This can be done by executing any one of the below methods:

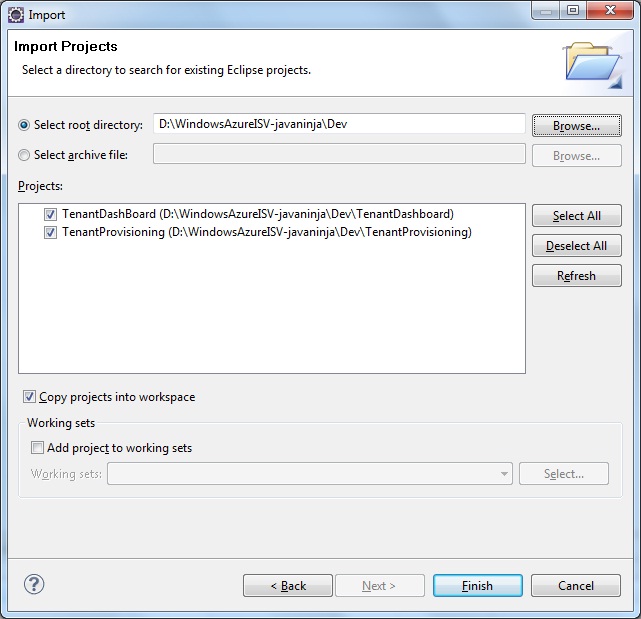
## Using Eclipse’s File -> Import

\*\* (**For simplicity we have assumed that source code is downloaded at location – D:\WindowsAzureISV-javaninja\Dev**).

1. Start Eclipse
2. Go to *File -> Import -> Existing Projects into workspace*.
3. Click Next button.
4. By default Select root directory option is selected.
5. Use Browse button to point directory *D:\WindowsAzureISV-javaninja\Dev \*\**



1. This will import projects TenantDashboard and TenantProvisioning into Eclipse.



1. Click on ok button for importing projects.

## Using Git plugin

If Github plugin is installed in eclipse, you can import projects directly as

File -> Import Git -> Projects from Git.

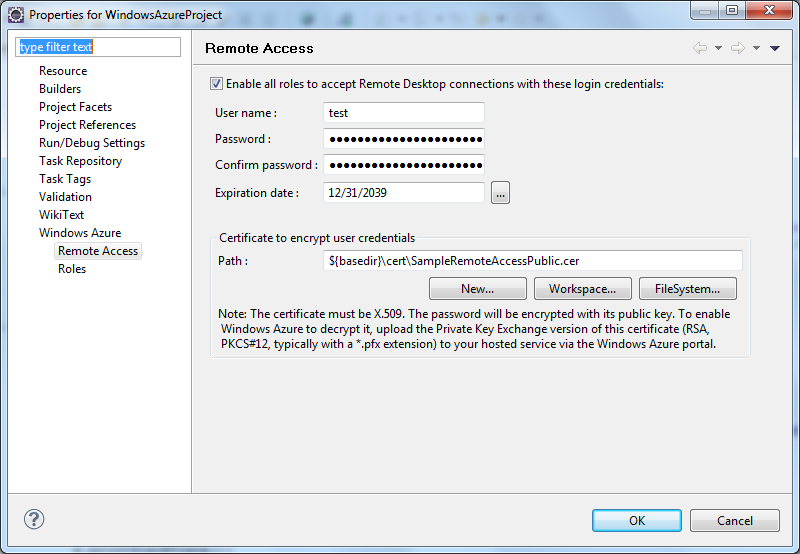
After importing projects if you get any error of ‘JRE for project build path’, then try adding JRE7 to Java Build Path again. (Right click on project, Properties -> Java Build Path -> Libraries)

## Creating certificate files

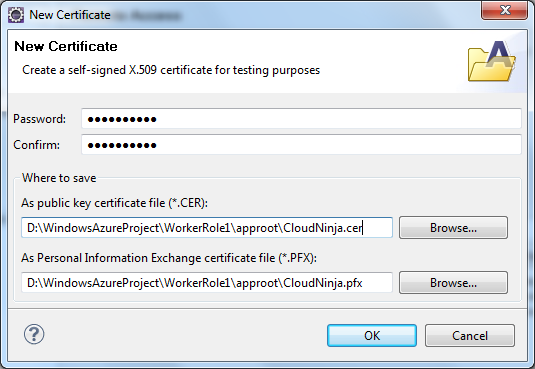
Create .pfx and .cer files which will be used to enable Remote Access and while making a call to service management API to get the deployment information such as total number of roles and their corresponding instances.

These files can be created using the Windows Azure plugin for eclipse as follows:

1. Go to Windows Azure Project’s properties. (Right click on project and click Properties).
2. Enable the remote access if not enabled.



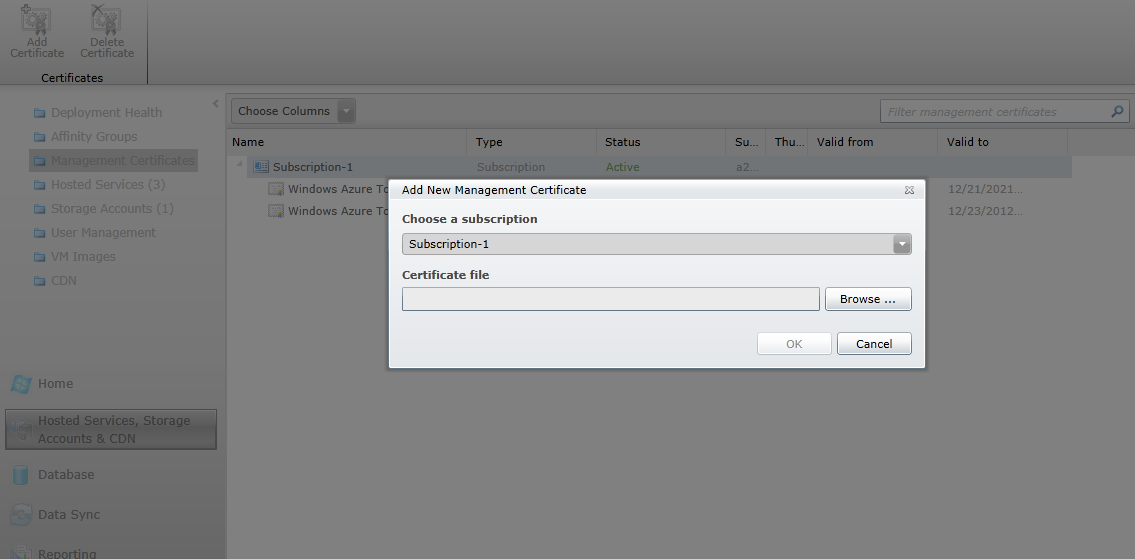
1. Click on New button.



1. Provide the password and file names as “***CloudNinja.cer***” and “***CloudNinja.pfx***” with the location <your\_windows\_azure\_project\_location>\WorkerRole1\approot in ***Where to save*** section.
2. “***CloudNinja.cer***” should also be added to your azure subscription using [Windows Azure](https://windows.azure.com) portal.
   1. Login to azure portal and go to Hosted Services, Storage Accounts & CDN

then select Management Certificates.

* 1. Select the subscription and Add Certificate. A certificate dialog will appear as shown below :

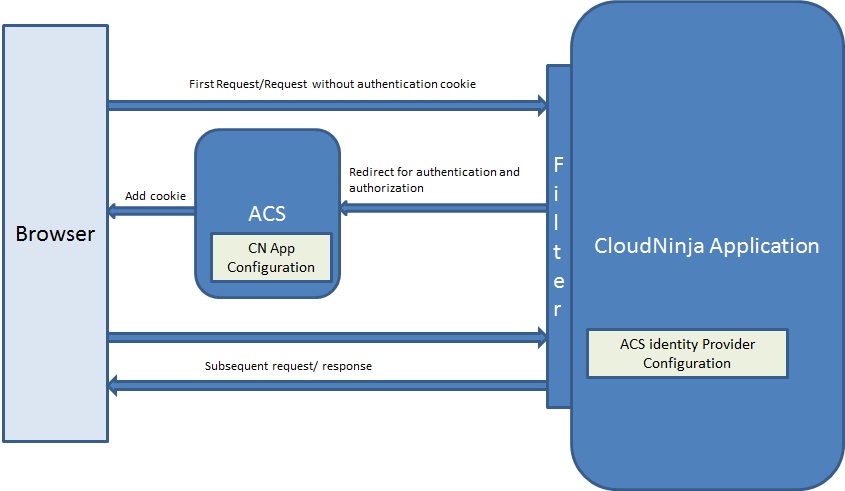


* 1. Browse to *<your\_windows\_azure\_project\_location>\WorkerRole1\approot* and select CloudNinja.cer file.

## ACS Workflow

Request- response cycle for CloudNinja application can be depicted as below:

1. When the request comes in for CloudNinja application, a filter will check if it contains a valid cookie. The cookie will not be present if the request is new.
2. If a valid cookie is not present, then the request is redirected to a SAML identity provider such as ACS, for authentication and authorization..
3. If a valid response is received from ACS, then it is parsed for tokens of interest and a cookie is created.
4. Subsequent requests will contain this cookie until it gets expired or user logs off.
5. The filter is applied for every request to check for the cookies and then decide on authentication/authorization.
6. Similar filter intercepts all requests to see if the user is authorized to access the application he/she is trying to access.



## Deploy CloudNinja STS

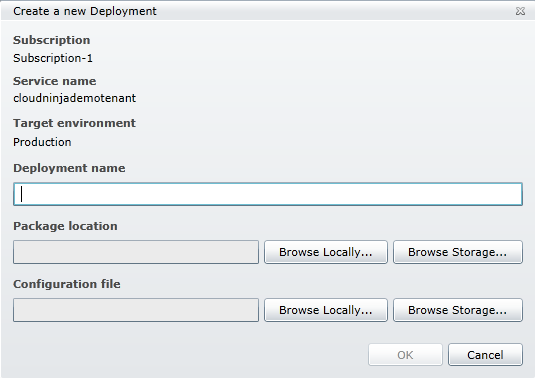
1. In order to deploy **CloudNinjaSTS**, download the source code from <http://cloudninja.codeplex.com/releases>. Open the **CloudNinja.Cloud.Sts** project, replace the ServiceConfiguration.cscfg with the ServiceConfiguration.cscfg file in CloudNinjaSTS folder (<https://github.com/PersistentSys/cloudninja-for-java/tree/domain/CloudNinjaSts> ). The certificate used in this ServiceConfiguration.cscfg can be found under CloudNinjaSTS folder.

Build the project to generate CloudNinja.Cloud.Sts.cspkg

1. Edit the **ServiceConfiguration.cscfg** to change following property :

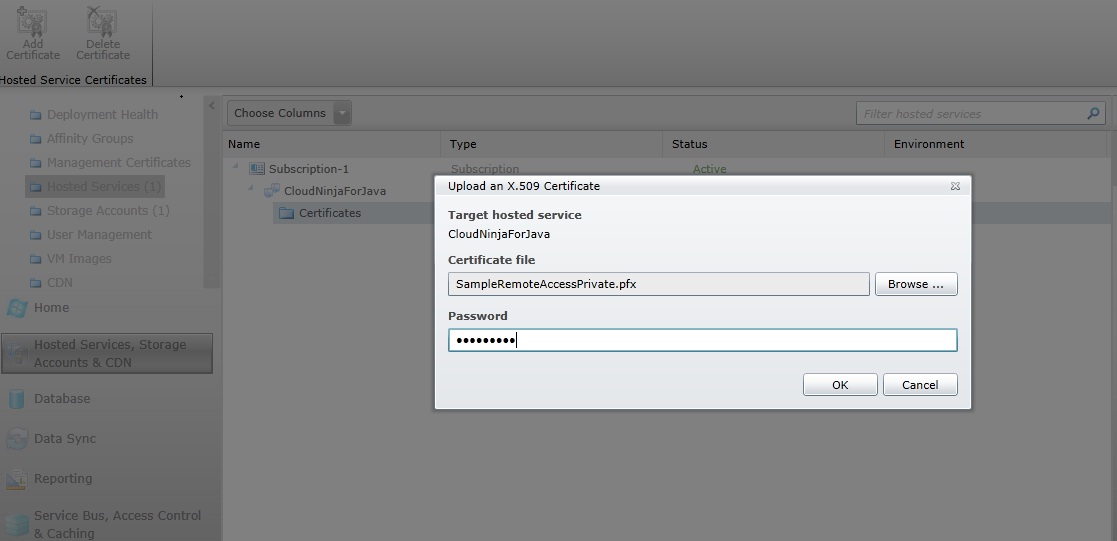
<Setting name="SystemConnectionString" value="Server=tcp:xnifcmw0bm.database.windows.net;Database=CloudNinjaPrimary;User ID=psladmin@xnifcmw0bm;Password=<databasepassword>;Trusted\_Connection=False;Encrypt=True;" />

1. Create a new hosted service and deployment by making use of Package file, the edited Configuration file and Certificate file (**cloudninjasts.pfx** from the downloaded CloudNinjaSTS folder). The provided certificate file (**cloudninjasts.pfx**) must be used for the deployment. Password for the Certificate file is ‘**abcd**’.
   1. Select Hosted services, storage services and CDN. Then select "Hosted services".
   2. Select Subscription - create a new Hosted service.
   3. Select the Hosted service which would be used for deployment and then create a new production deployment.



Provide the deployment name. Browse using *Browse Locally* button for the location <CloudNinjaSTS\_folder> for Package location i.e. "CloudNinja.Cloud.Sts.cspkg”.

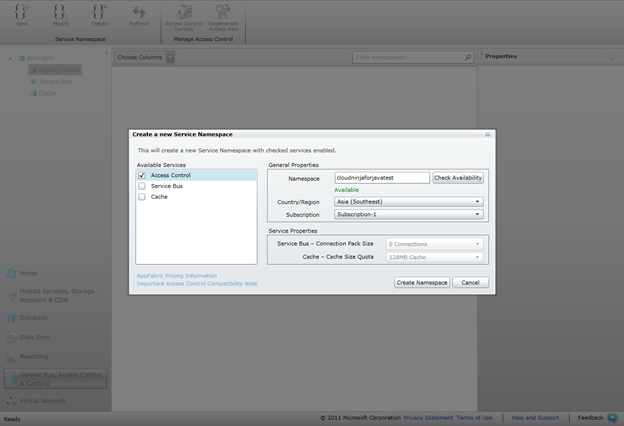
* 1. Select configuration file “ServiceConfiguration.cscfg” from the above specified location.
  2. You might be asked to upload a certificate. Please specify the certificate file (‘**cloudninjasts.pfx’** with password ‘**abcd**’).



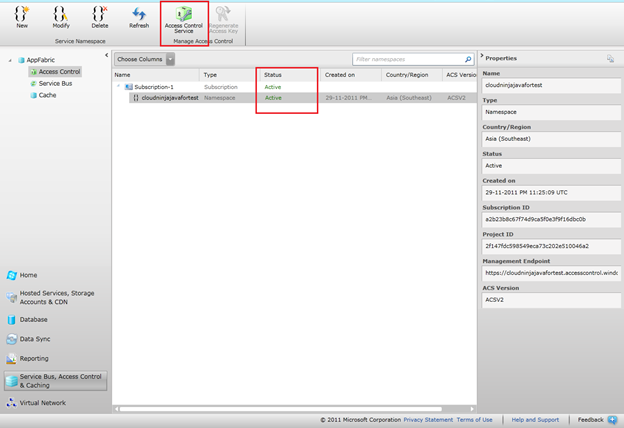
* 1. After this step, it will start uploading the package to the cloud environment.
  2. You may get a warning if you are deploying a single instance. Please ignore this warning.

## Configure ACS

1. Login to windows azure management [portal](https://windows.azure.com/).
2. On the lower-left-hand side of this page, click ‘Service Bus, Access Control & Caching’. To create a new service namespace, select ‘Access Control’ on the tree shown in the left side of the page and click ‘New Namespace’ from the toolbar at the top of the page. Provide a unique value for namespace, choose the desired region and click on ‘Create Namespace’. (Uncheck ‘Service Bus’ and ‘Cache’ since we are using only ACS)



1. Select the created namespace and then click on ‘Access Control Service’ on the toolbar at the top of the page so as to be redirected to the ACS portal.

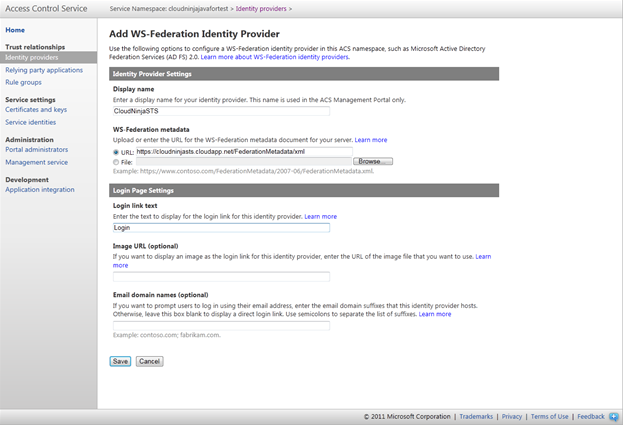


1. Upon redirected to the ACS portal, click on the ‘Identity Providers’ link on left side of the page. By default ‘Windows Live ID’ would have been added.

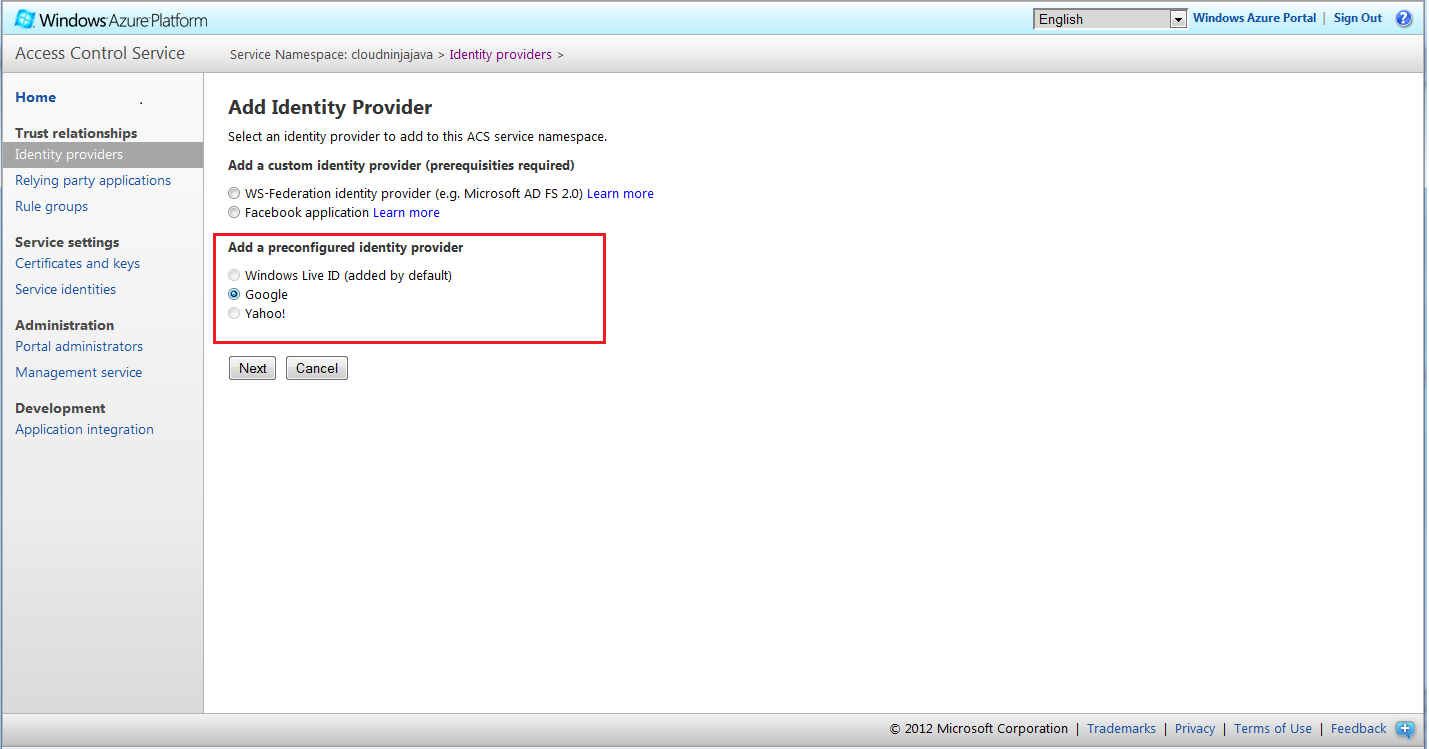
We need to add the custom STS (written in .NET) which will be used to verify user identity by checking the ‘Member’ table in control database. Click on the link ‘Add’. On the next page select “**WS-Federation identity provider**” and then click on the button ‘Next’.

Note: - At this point, the custom STS must have been already hosted onto windows azure.

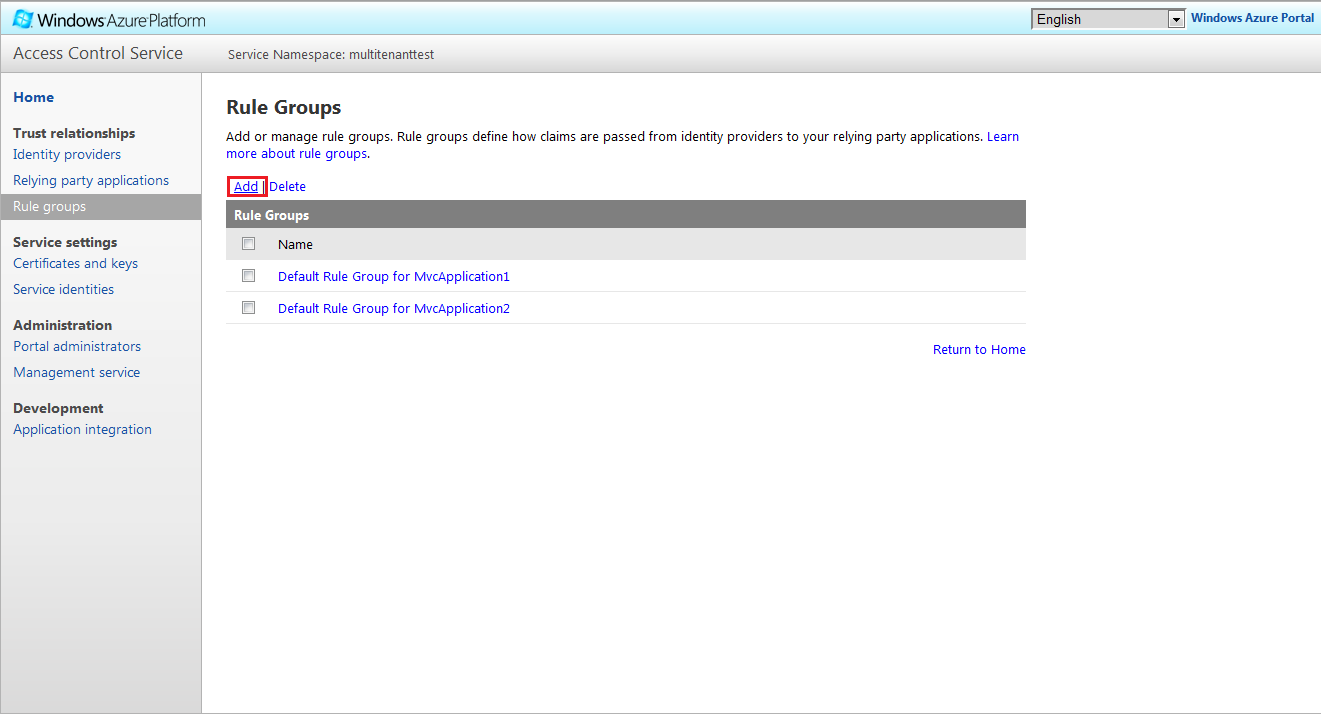
The value for URL must be ‘https://<custom STS base url>/FederationMetadata/xml’. Here <custom STS base url> would be replaced with the URL of hosted service for STS we deployed in previous section (***Deploy CloudNinja STS***). Use an appropriate friendly name as ‘Display Name’ and use ‘Login’ for ‘Login link text’. Click on ‘save’.



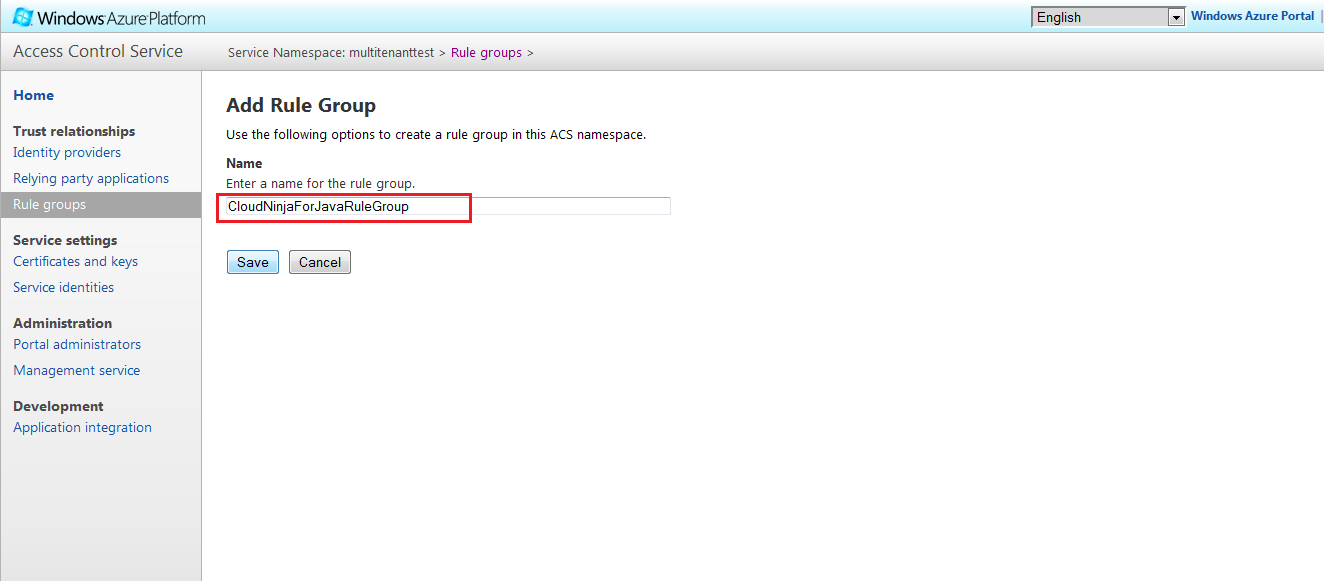
1. We can also add Yahoo and Google in identity providers which are preconfigured in ACS. Click on the link “Identity Provider” on the left side of the page and click on “Add”. Now select Google from “Add a preconfigured identity provider” and say next.



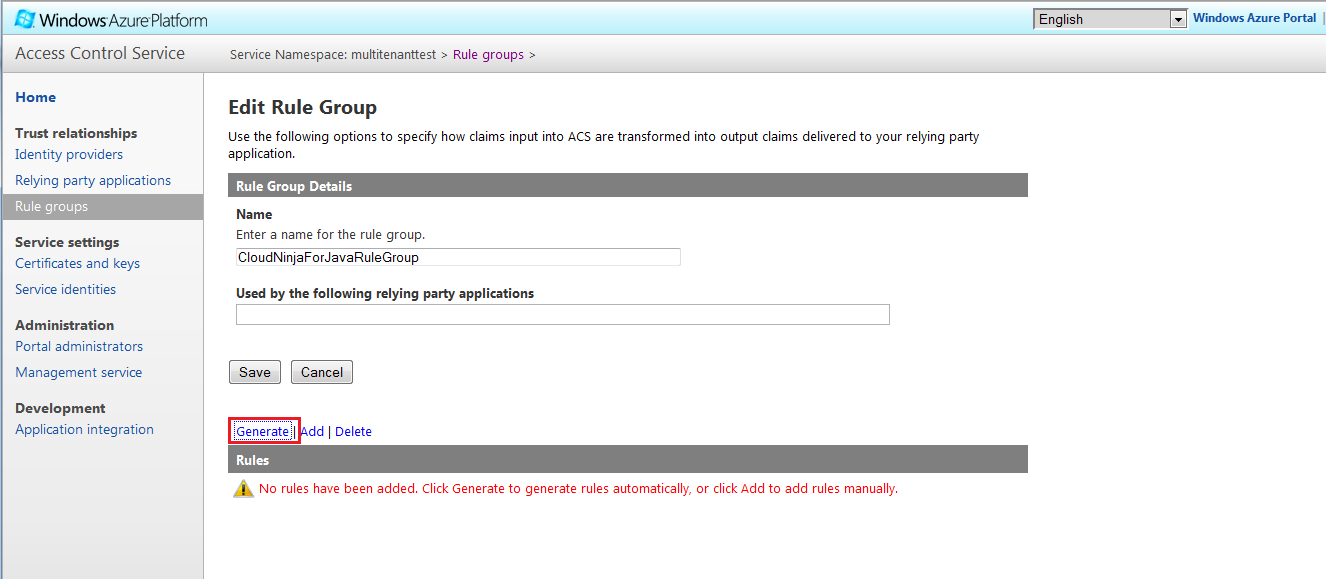
1. Save the Identity Provider. This way add Yahoo as Identity Provider.
2. Create “Rule Group”. To create a rule group Click on the link ‘Rule Groups’ on the left side of the page say “Add”



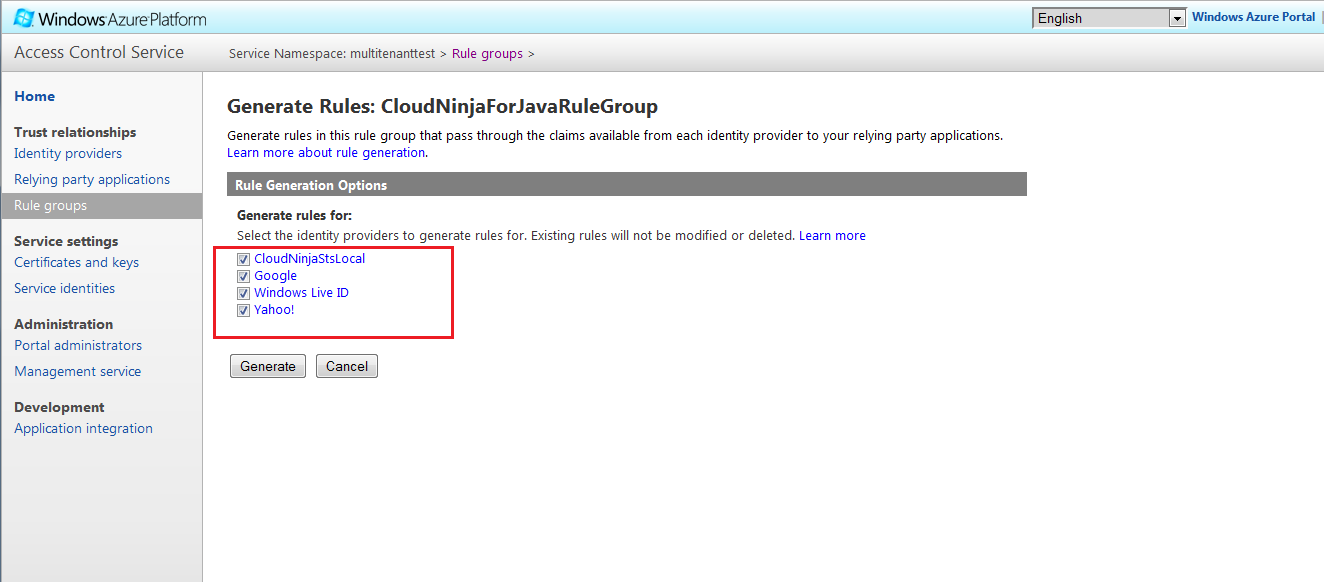
Enter the name. We have used “CloudNinjaForJavaRuleGroup” here. Click on Save

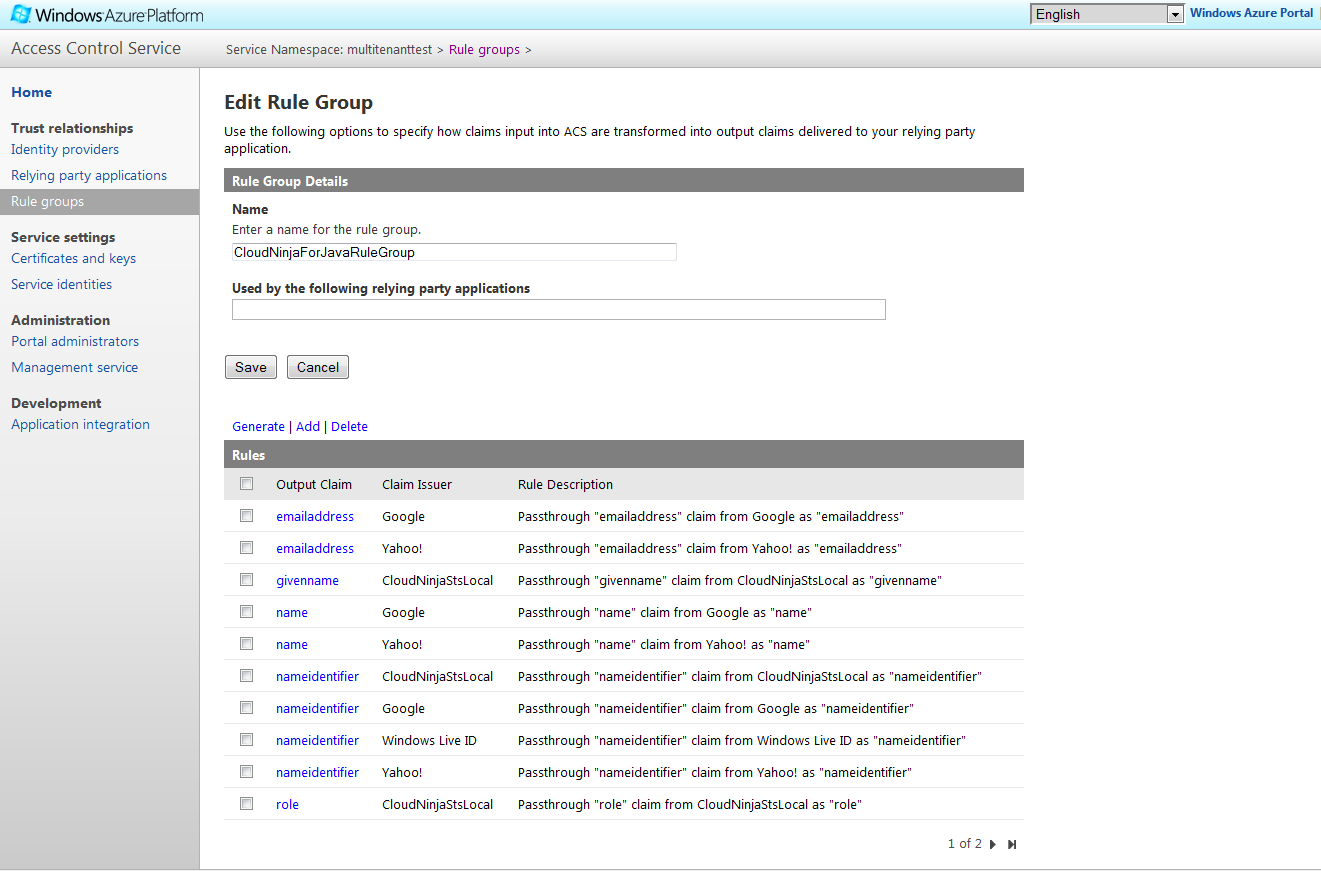


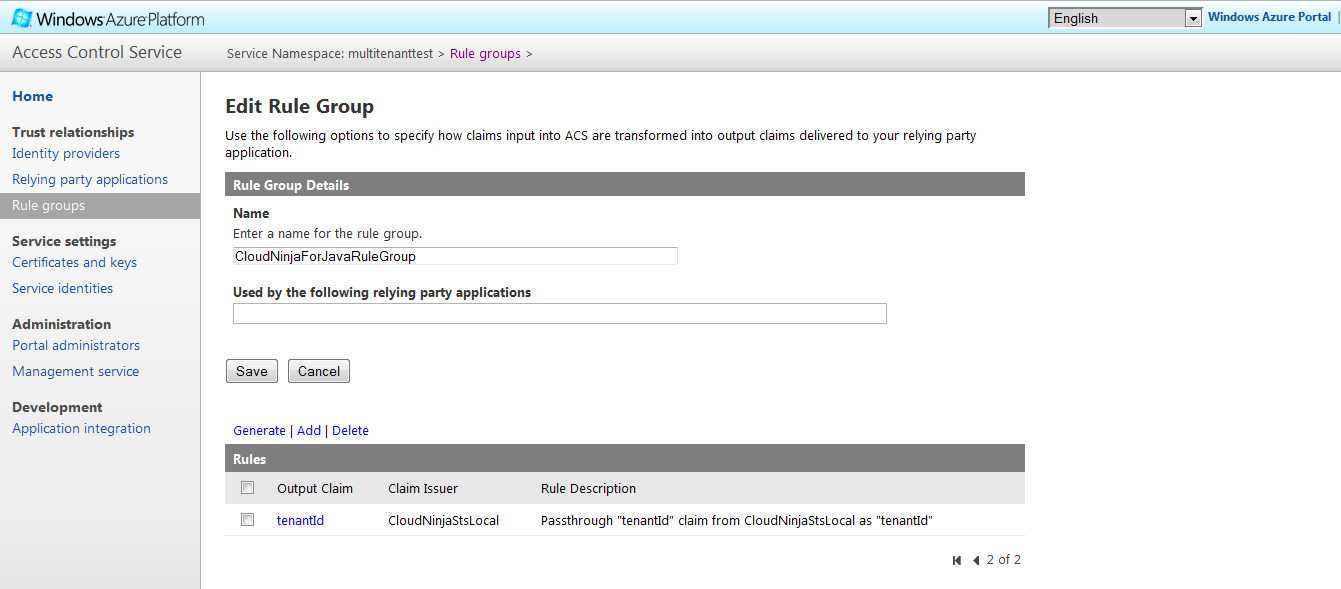
1. Click the link ‘Generate’ to create the default rules for transforming the claims that can be understood by the relying party application. On the next page, click ‘Generate’ to generate default transformation.



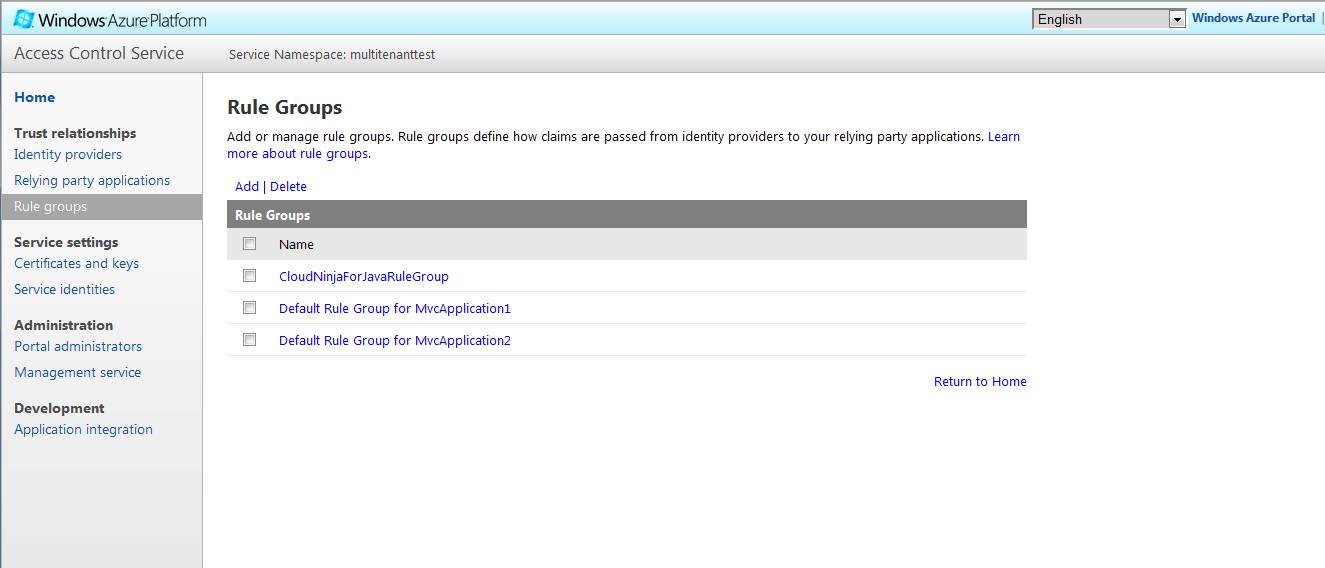
Here the entire identity provider list should be selected.







Save it.



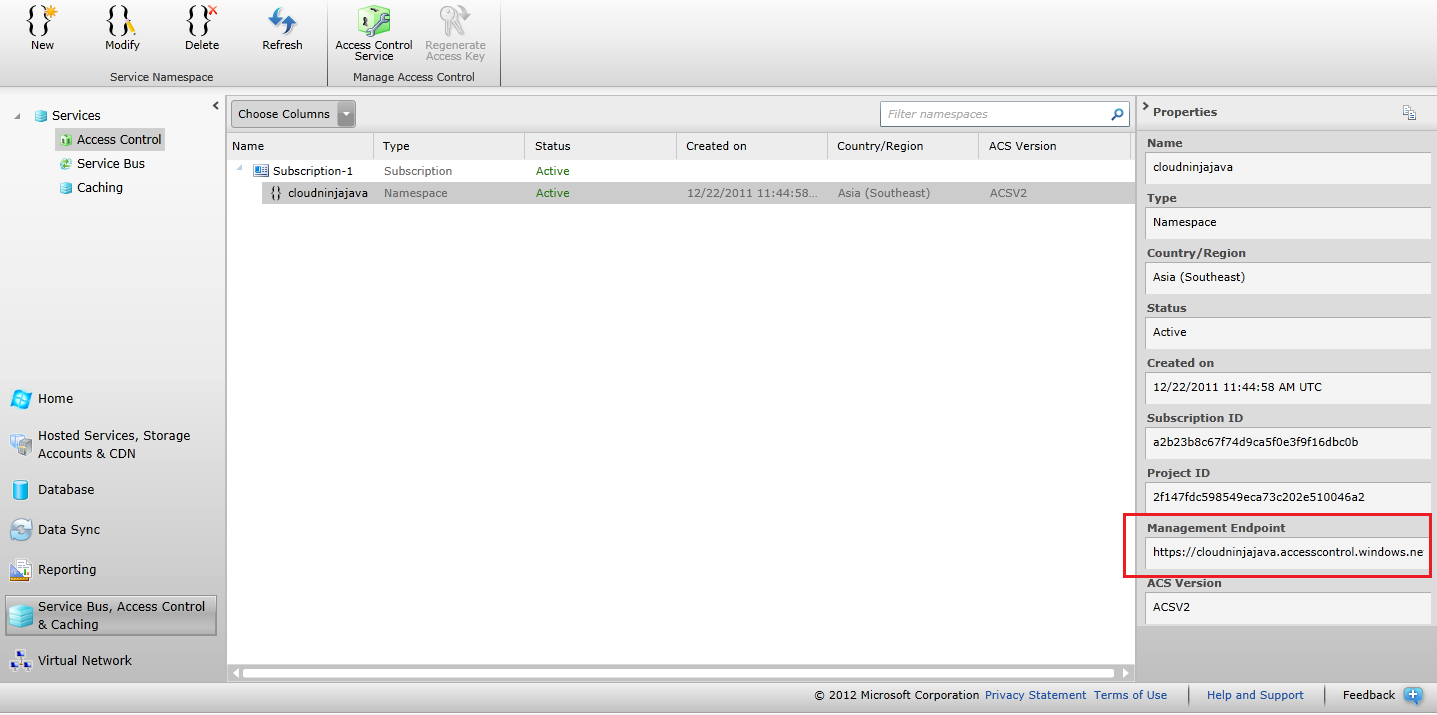
## Creating WAR files

### ACS Configuration

**ACS Configuration in Tenant Dashboard application**

1. We have created a custom identity provider and configured in ACS as explained in previous section.
2. The ACS url is generally something like <https://cloudninjajava.accesscontrol.windows.net/v2/wsfederation> where “cloudninjajava.accesscontrol.windows.net” is the Management Endpoint of ACS and it ends with “/wsfedaration”.

Where Management endpoint can be taken from Windows Azure Portal (In Service Bus, Access control and Caching, select the Namespace and check the properties on right side of window).



1. Make an entry of this url against **acs.idp.url** key in WebContent/WEB-INF/resources/acs.properties file.
2. The application will send an identity validation request to the ACS site. The actual url with various attributes is created in the code using the values supplied above.
3. The URL looks like: <https://cloudninjajava.accesscontrol.windows.net/v2/wsfederation?wa=wsignin1.0&wtrealm=http://cloudninjademo.cloudapp.net/TenantDashBoard>
4. To verify if the url is correct, try accessing it using a browser. ACS login page should be displayed.
5. The response returned by the ACS will be parsed and stored in the browser cookie. The max age of this authentication cookie should be set in the acs.properties file. The key for this property is **browser.cookie.max.age**.in.seconds.
6. The application does not need to be compiled again.

For example, a sample configuration (acs.properties) might look like:

* acs.url=https://cloudninjajava.accesscontrol.windows.net/
* acs.idp.url=https://cloudninjajava.accesscontrol.windows.net/v2/wsfederation
* wa.acs=wsignin1.0
* browser.cookie.max.age.in.seconds=600

**ACS Configuration in Tenant Provisioning application**

Modify the acs.properties as:

acs.namespace=cloudninjajava

acs.management.username=ManagementClient

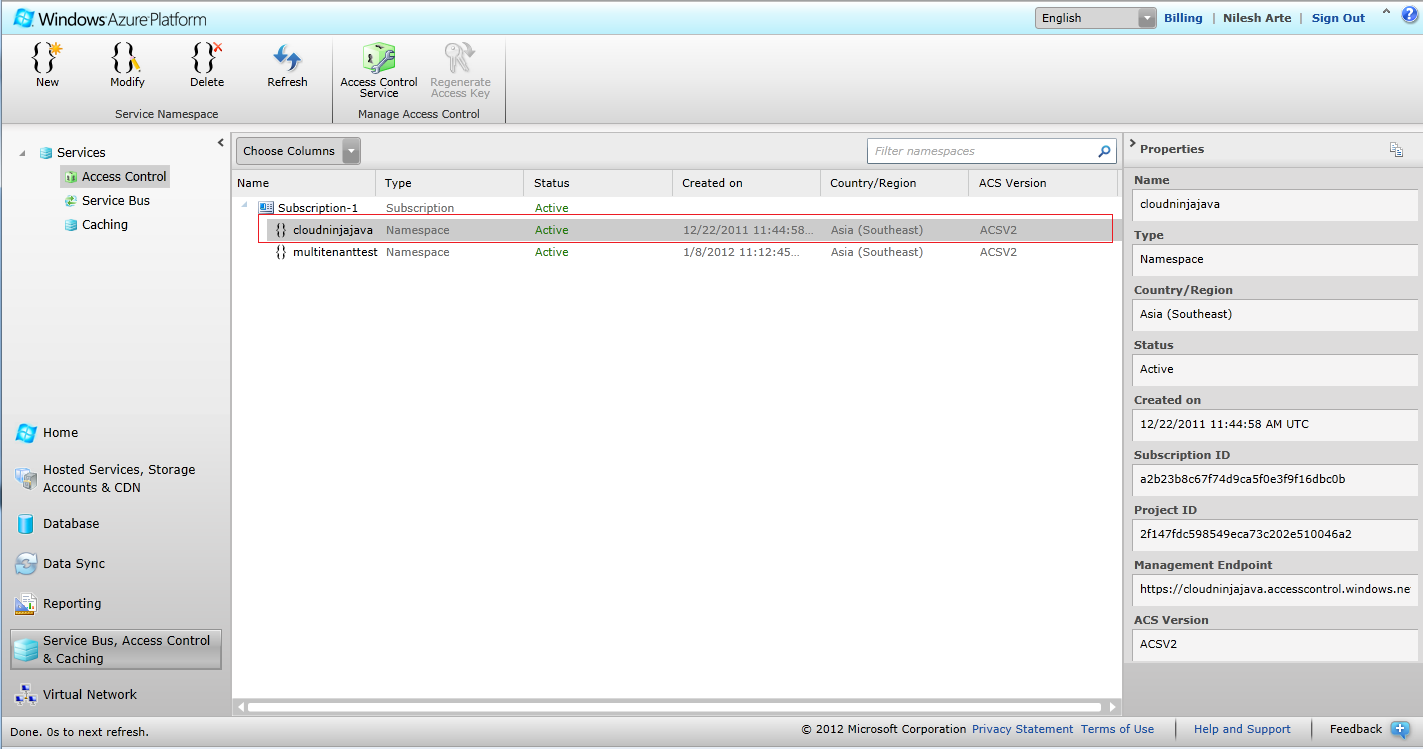
acs.management.password=SOnfPKyGnWjQtsg/38IzWHP7cQxDY8rwBI95KFlv4NQ=

acs.rulegroup=CloudNinjaForJavaRuleGroup

acs.returnurl=http://%s.%s/TenantDashBoard/showTenantHomePage.htm

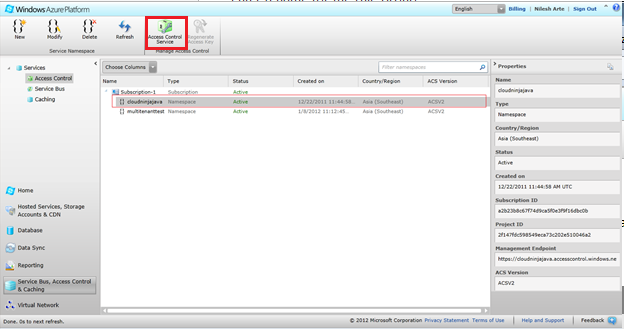
acs.applicationdomain=cloudninjajava.com

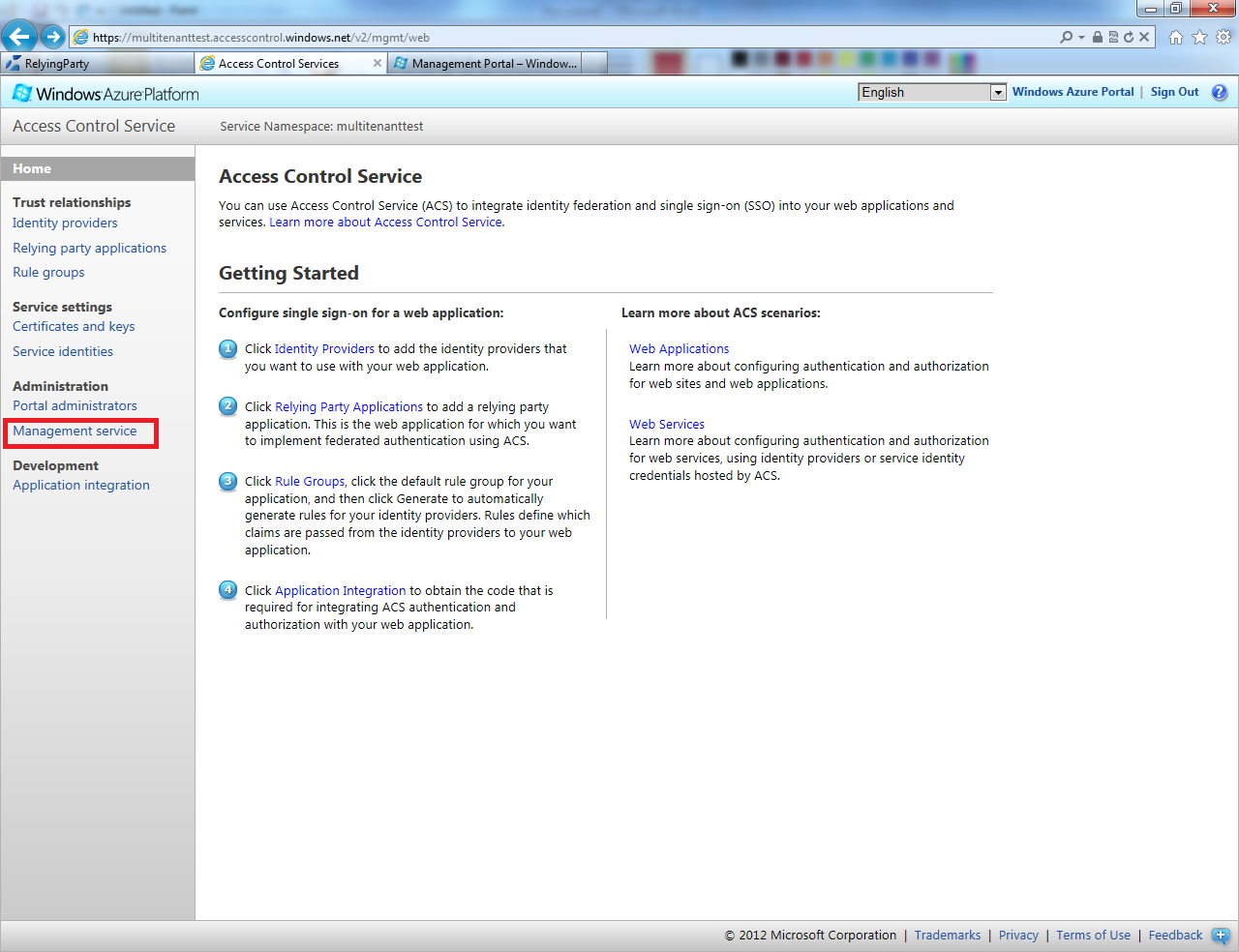
* acs.namespace is the namespace under Service Bus, Access control & caching ->Access Control

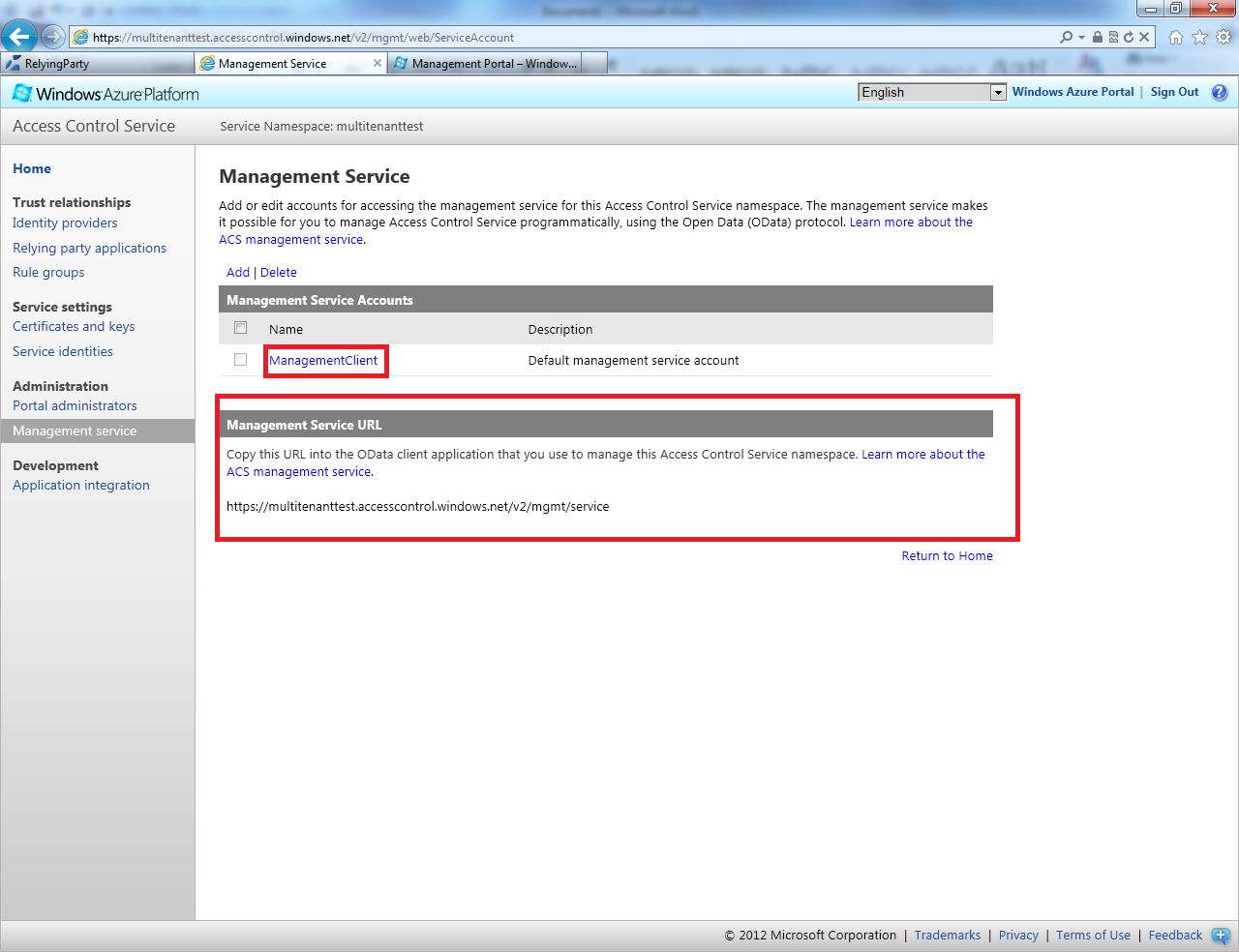


* acs.management.username is the Management username which we can get from Service Bus, Access control & caching ->Access Control, select the namespace and select “Access Control Service”
* acs.management.username and acs.management.password can be obtained as follows:

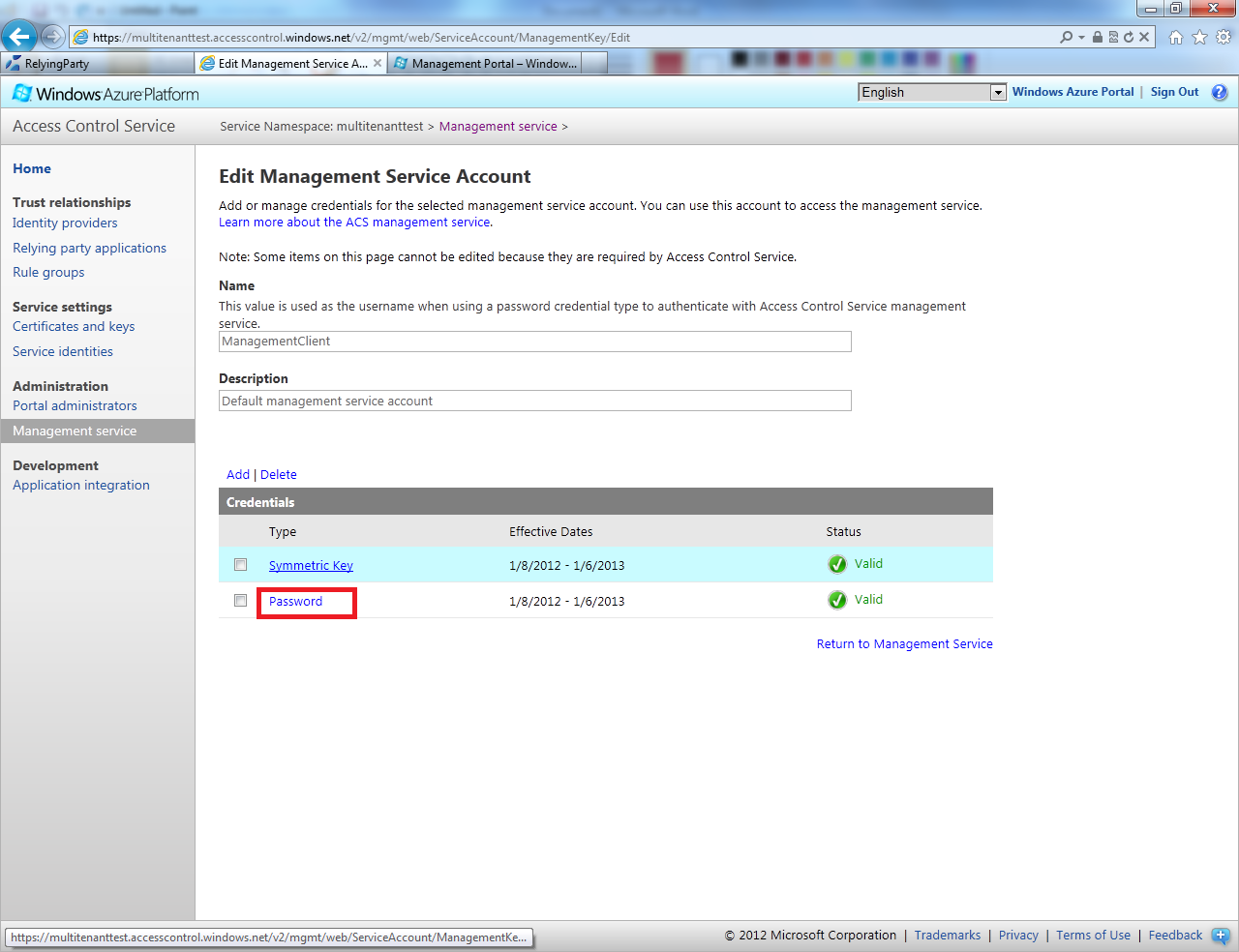
Navigate to the ACS namespace portal.



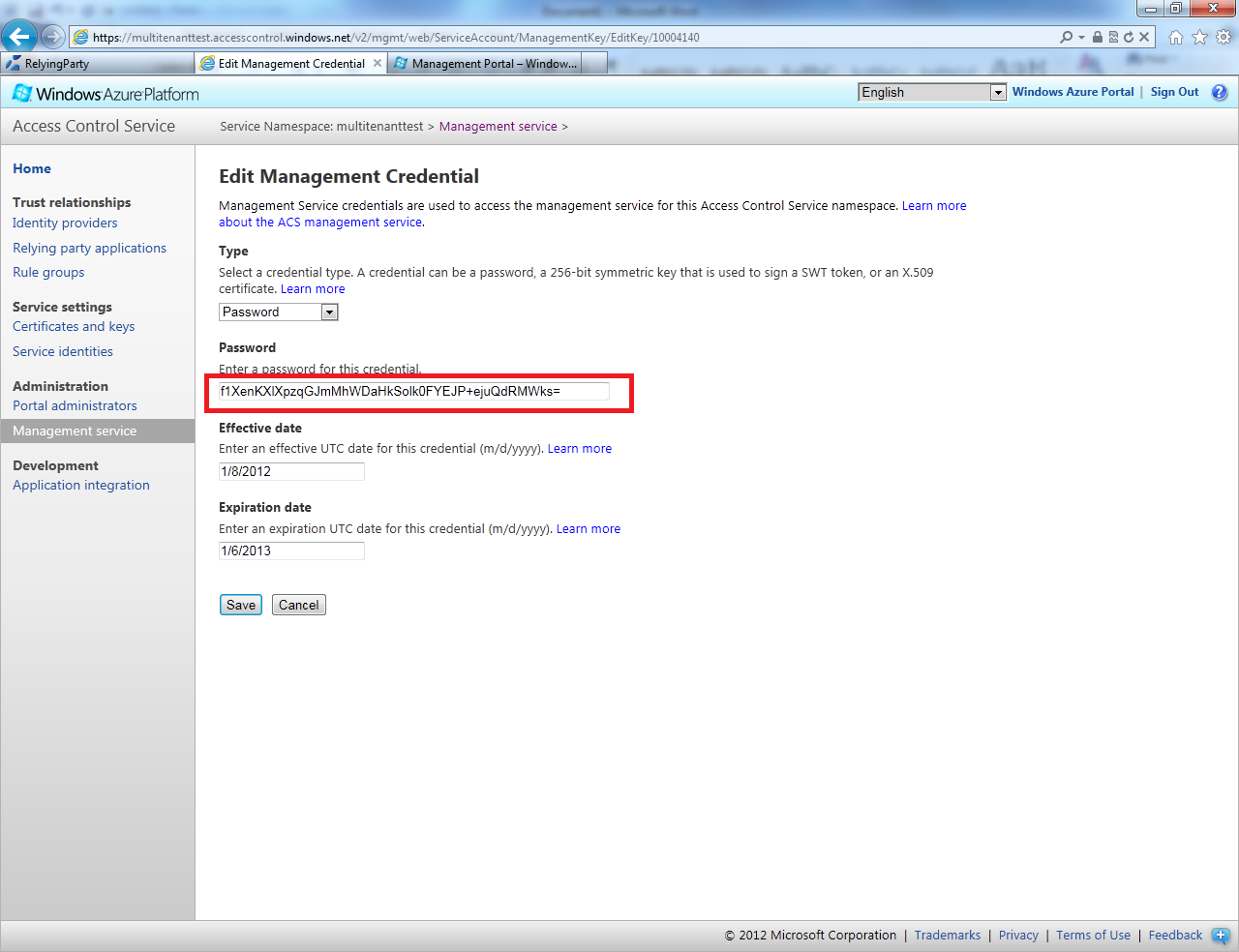




Note the credential name ‘ManagementClient’ and assign it to acs.management.username. This value cannot be changed in the portal.



Note the credential password and assign it to acs.management.password. This value can be changed through the portal.



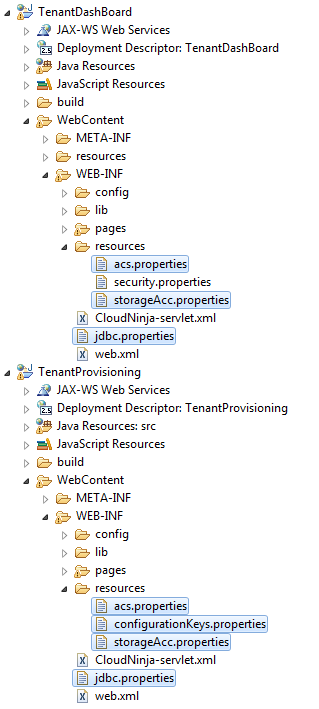
* acs.rulegroup is the Rule group created above that is

”CloudNinjaForJavaRuleGroup”

* acs.applicationdomain will be replaced by deployment URL e.g. **cloudninjademo.cloudapp.net** or if a domain name e.g. **cloudninjajava.com**  is registered for deployment URL then it will be **cloudninjajava.com**.

### Other (non-ACS) configuration changes are as below:

1. These configuration changes are done in the following selected files in snapshot below



1. Please make sure you are pointing to correct database and storage account by editing Web-INF\jdbc.properties and Web-INF\Resources\storageAcc.properties files.

Database name MUST be ‘**CloudNinjaPrimary’**.

1. Modify the **configurationKeys.properties** as :

#certificate password

deployment.keystorepassword=<password you used while creating the CloudNinja.cer as explained earlier in **Creating certificate files** section>

#java keystore password

deployment.truststorepassword=<your java keystore("%JRE\_HOME%\lib\security\cacerts") password (Default is ***changeit***)>

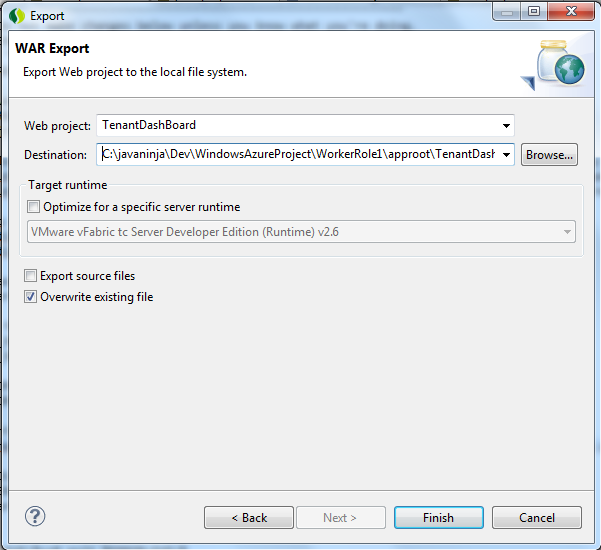
deployment.host=https://management.core.windows.net/

deployment.subscriptionid=<subscription ID of your azure account>

deployment.hostedservicename=<name of the hosted service on which application will be deployed>

deployment.deploymenttype=<type of deployment, valid values ‘production’ or ‘staging’ without quotes>

1. Now we need to generate WAR files for TenantProvisioning and TenantDashBoard Project. So follow the below steps for each project.
2. Right click on the project and select Export from the menu.
3. We have to select WAR format here which can be found at *Web->War* file from the “Select an Export Destination” option window.
4. Select WAR and click on the Next button.



1. If the Windows Azure Project is already created or imported then select destination folder as approot folder under Windows Azure Project (In gitHub approot is located at <https://github.com/PersistentSys/cloudninja-for-java/tree/domain/WindowsAzureProject/WorkerRole1/approot> ) else specify any destination and move WAR files to approot folder. After selecting “Overwrite existing file”, click Finish button.

## Deploying application

1. Before deploying application make sure that you run the database scripts on SQL Azure for ‘CloudNinjaPrimary’ database.

Before running script take care of changes in **Primary.sql** script.

**INSERT** **INTO** [IdentityProvider] ([Name])

**VALUES** ('CloudNinjaSts')

**INSERT** **INTO** [IdentityProvider] ([Name])

**VALUES** ('Google')

**INSERT** **INTO** [IdentityProvider] ([Name])

**VALUES** ('WindowsLiveID')

**INSERT** **INTO** [IdentityProvider] ([Name])

**VALUES** ('Yahoo!')

The above statements add the Name of Identity providers in table “IdentityProvider”.

So if a new Identity Provider is to be supported or removed then the above script is to be modified.

**INSERT** **INTO** [DataServer] ([Server], [LocationId], [User] , [Password])

**VALUES** (CAST(SERVERPROPERTY('ServerName') **AS** **VARCHAR**(64)) + '.database.windows.net', 1, '<username>@<SQL-Azure-server-name>', 'password')

The above statement has the name and password of the database server on which these scripts are executed.

So if the database server changes the Id and password should be updated in the above statement.

These scripts can be found at <https://github.com/PersistentSys/cloudninja-for-java/tree/domain/DBScripts> and which should be run in the following order :

1. Primary.sql
2. Members.sql
3. Metering.sql
4. Monitoring.sql.
5. Replace the startup.cmd contents with the following if you have created new Windows Azure Project.

**Startup.cmd contents:**

@REM To use the sample, follow these steps:

@REM \*\*\* 1) Copy all this content into approot/startup.cmd in the role folder, close this file, and edit the copy

@REM \*\*\* 2) Update SERVER\_DIR\_NAME below as appropriate:

@REM \*\*\* (IMPORTANT: There must be no trailing nor leading whitespace around the setting)

SET SERVER\_DIR\_NAME=apache-tomcat-7.0.22

@REM \*\*\* 3) To deploy your own WAR file, place it in approot and update WAR\_NAME below:

@REM \*\*\* (IMPORTANT: There must be no trailing nor leading whitespace around the setting)

SET WAR\_NAME=TenantDashBoard.war

SET WAR\_NAME\_1=TenantProvisioning.war

@REM \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

@REM \*\*\* Do not make changes below unless you know what you're doing.

rd "\%ROLENAME%"

mklink /D "\%ROLENAME%" "%ROLEROOT%\approot"

cd /d "\%ROLENAME%"

@REM Remove tomcat and jre directories if already exists

IF EXIST %SERVER\_DIR\_NAME% rmdir /S /Q %SERVER\_DIR\_NAME%

IF EXIST jre7 rmdir /S /Q jre7

@REM Download apache tomcat and jre from the blob container. Following URLs will be replaced by your storage account.

cscript "util\download.vbs" "https://<storage-account-name>.blob.core.windows.net/binaries/apache-tomcat-7.0.22-windows-x64.zip" "apache-tomcat-7.0.22.zip"

cscript "util\download.vbs" "https:// <storage-account-name>.blob.core.windows.net/binaries/jre7-64bit.zip" "jre7.zip"

@REM Unzip the zip files

cscript util\unzip.vbs jre7.zip "%CD%"

cscript util\unzip.vbs apache-tomcat-7.0.22.zip "%CD%"

@REM Replace the existing server.xml

copy /Y server.xml "%SERVER\_DIR\_NAME%\conf\server.xml"

@REM Copy the war files to webapps

copy %WAR\_NAME% "%SERVER\_DIR\_NAME%\webapps\%WAR\_NAME%"

copy %WAR\_NAME\_1% "%SERVER\_DIR\_NAME%\webapps\%WAR\_NAME\_1%"

set JRE\_HOME=\%ROLENAME%\jre7

cd %JRE\_HOME%\bin

@REM Import the CloudNinja.cer into the JAVA keystore

keytool -noprompt -import -v -alias cloudninjacert -file "\%ROLENAME%\CloudNinja.cer" -keypass cloudninja -keystore "%JRE\_HOME%\lib\security\cacerts" -storepass changeit

copy "\%ROLENAME%\CloudNinja.pfx" %JRE\_HOME%\lib\security\CloudNinja.pfx

cd "\%ROLENAME%\%SERVER\_DIR\_NAME%\bin"

cmd /c startup.bat

@ECHO OFF

if %ERRORLEVEL%==0 exit %ERRORLEVEL%

choice /d y /t 5 /c Y /N /M "\*\*\* Windows Azure startup failed - exiting..."

exit %ERRORLEVEL%

(This startup.cmd is located at

<https://github.com/PersistentSys/cloudninja-for-java/tree/domain/WindowsAzureProject/WorkerRole1/approot> )

1. Change the contents of startup.cmd marked with yellow color (Refer above step).
   1. [https:// <storage-account-name>.blob.core.windows.net/binaries/apache-tomcat-7.0.22-windows-x64.zip](https://cloudninjaforjavastorage.blob.core.windows.net/binaries/apache-tomcat-7.0.22-windows-x64.zip) is the zip file for apache tomcat (64 bit version) hosted on your storage account’s blob container. Create a blob container with name ***binaries*** and upload the tomcat zip to that location. Then replace above URL in **startup.cmd** with your modified URL.
   2. [https:// <storage-account-name>.blob.core.windows.net/binaries/jre7-64bit.zip](https://cloudninjaforjavastorage.blob.core.windows.net/binaries/jre7-64bit.zip) is the zip file for JRE (64 bit version) hosted on your storage account’s blob container named ***binaries***. Upload the JRE zip to ***binaries*** container. Then replace above URL in **startup.cmd** with your modified URL.
   3. keytool -noprompt -import -v -alias cloudninjacert -file "\%ROLENAME%\CloudNinja.cer" -keypass cloudninja -keystore "%JRE\_HOME%\lib\security\cacerts" -storepass changeit

Replace ***cloudninja*** with the password you used while creating the CloudNinja.cer as explained earlier in **Creating certificate files** section. And replace ***changeit*** with your java keystore password (Default is ***changeit***).

1. Add both WAR files TenantDashBoard.war and TenantProvisioning.war to the approot directory of WorkerRole1.
2. Delete Helloworld.war file as we will not need it.
3. Perform below mentioned steps for WindowsAzureProject to ensure that the following things are in place.
   1. copy the Azure Diagnostics related files (<https://github.com/PersistentSys/cloudninja-for-java/tree/domain/AzureDiagnosticsTool> ) and the dependencies specified in **Readme.txt** (<https://github.com/PersistentSys/cloudninja-for-java/blob/domain/AzureDiagnosticsTool/Readme.txt> ) into approot folder.
   2. and also copy the **server.xml** file from tomcat’s **conf** directory to approot folder, after modifying its Valve contents as follows :

<Valve className=*"org.apache.catalina.valves.AccessLogValve"*

directory=*"acslogs"*

pattern=*"%t,%U,%{Content-Length}i,%{Content-Length}o"*

prefix=*"access\_log."* suffix=*".txt"* resolveHosts=*"false"*/>

(An example of such server.xml is located at <https://github.com/PersistentSys/cloudninja-for-java/blob/domain/WindowsAzureProject/WorkerRole1/approot/server.xml> )

* 1. modify the Startup node in **ServiceDefinition.csdef** by adding the following highlighted entry as follows :

<Startup>

<!-- Sample startup task calling startup.cmd from the role's approot folder -->

<Task commandLine=*"util/.start.cmd startup.cmd"* executionContext=*"elevated"* taskType=*"simple"*>

</Task>

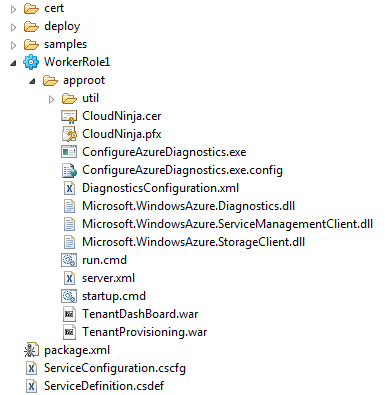
<Task commandLine=*"util/.start.cmd ConfigureAzureDiagnostics.exe"* executionContext=*"elevated"* taskType=*"background"*/>

</Startup>

1. Open the **DiagnosticsConfiguration.xml** which is inside approot and replace the following **StorageAccountConnectionString** with your storage account details.

<DiagnosticsConfig StorageAccountConnectionString=*"DefaultEndpointsProtocol=https;*

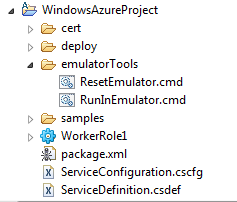
*AccountName=storage-account-name;AccountKey=storage-account-key"*>

1. The Windows Azure project should have the following structure after completing all the mentioned steps till now :
2. 
3. Build the project.
4. To run the project in local environment –
   1. Open package.xml. Make sure the value for <Windowsazurepackage> is “local”.
   2. Build the azure project.

To build the azure project in eclipse, select the project in Project Explorer View and then click on Project menu on menu bar and say build project.

While building, sometimes we may get an error saying “Cannot run the program ResetEmulator.cmd” which is ignorable if the project gets built successfully.

* 1. For local build, emulatorTools folder will be created in the project.

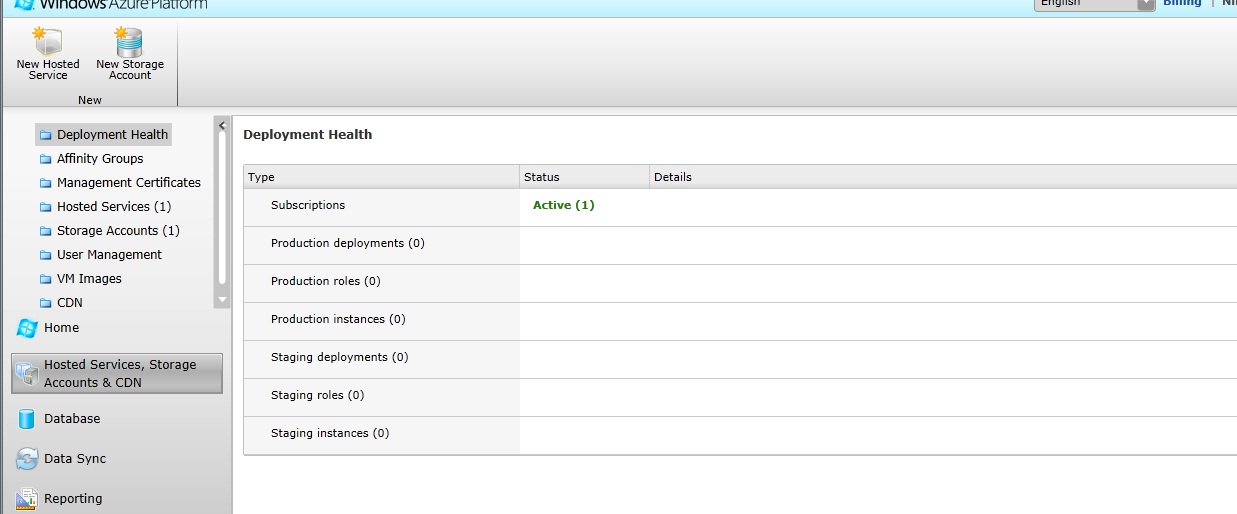


* 1. Start emulator tool by launching emulatorTools -> RunInEmulator.cmd

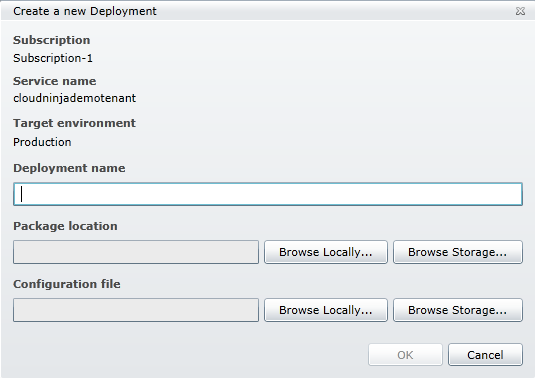
1. To run the project in cloud environment –
   1. Open the package.xml and change <Windowsazurepackage> package type from "local" to "cloud"
   2. Run ResetEmulator.cmd.
   3. Build the azure project. To build the azure project in eclipse, select the project in Project Explorer View and then click on Project menu on menu bar and say build project.

While building, sometimes we get an error saying “Cannot run the program ResetEmulator.cmd” which is ignorable if the project gets built successfully.

* 1. Go to portal (<https://windows.azure.com> ).

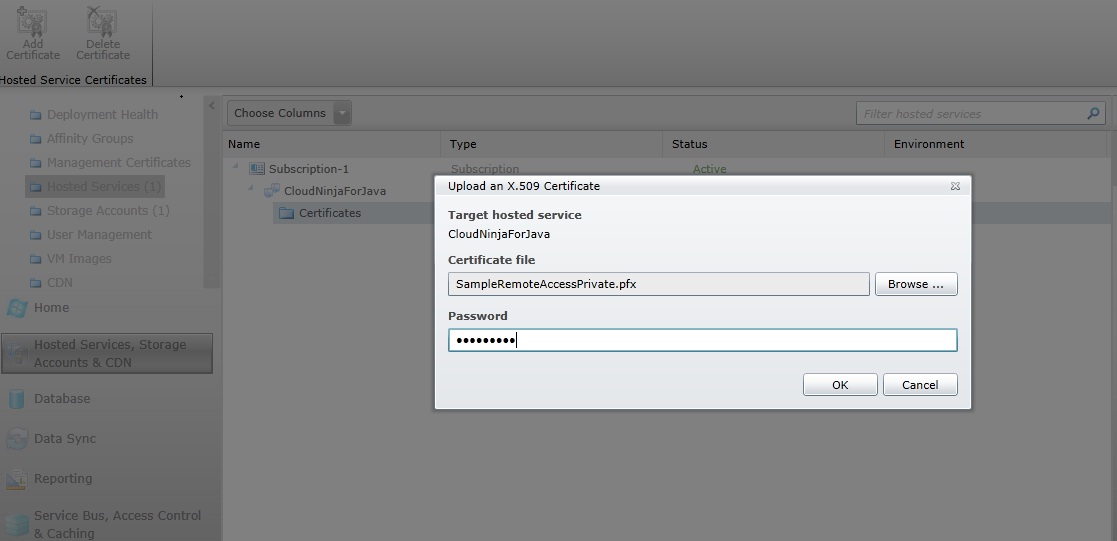


* 1. Select Hosted services, storage services and CDN. Then select "Hosted services".
  2. Select Subscription -. If you are using it for first time, create a new Hosted service using the name you specified in **configurationKeys.properties** in “***Creating WAR files***” section.
  3. Select the Hosted service which would be used for deployment and then create a new production deployment.



Provide the deployment name. Browse using *Browse Locally* button for the location <your\_windows\_azure\_project\_location>\deploy for Package location i.e. "WindowsAzurePackage.cspkg”.

* 1. Select configuration file “ServiceConfiguration.cscfg” from the above specified location.
  2. You might be asked to upload a certificate. Please specify the certificate file (the file extension is ”.pfx”). If you have used the same certificate file created in “***Creating certificate files***” section to enable remote access then specify that file only here.



* 1. After this step, it will start uploading the package to the cloud environment.

1. You may get a warning if you are deploying a single instance. Please ignore this warning.