Hybris On Azure – For All

Setup guide (Version 1.5)

1. Introduction

This document describes how to set up an installation of hybris on the Windows Azure platform. It guides through the necessary steps in order to create required cloud services as well as how to prepare the hybris platform in order to be properly deployed. After completing this setup guide you should have a basic installation of hybris running on Azure ready to be customized and populated with data.

In this document the hybris commerce suite 5.4.0.0 is used with Java 1.7.0 Update 65.

Architecture

HybrisOnAzure For All provides a two tier setup:

1st Tier: Application request routing tier consisting of two or more servers running IIS assuring requests of one customer are always forwarded to the same machine and protecting administrative interfaces from public access.

2nd Tier: Two or more application servers hosting hybris providing web content using a Tomcat server. Additionally this tier contains one application server called "BackOfficeWorker" exposing administrative interfaces.

VM size considerations

Due to massive file access when using hybris with detailed logging (e.g. while debugging) experience shows VMs equipped with HDD drives lack performance to handle logging and user requests. Thus SSD equipped VMs (Standard_D4) are used for the servers running hybris.

For the application request routing tier Medium sized VMs are sufficient to handle SSL offloading and session affinity.

Database considerations

The sample setup provided by this guide uses SQL Azure as a database tier. You might consider using a SQL Server Always On setup provided by Windows Azure. Please respect the virtual network setup information about SQL Server Always on in chapter 2.4 Windows Azure Virtual Network.

Table of contents

2. Prerequisites

This chapter guides you through all the necessary steps to setup a Windows Azure environment to host the hybris platform. If you haven't signed up for a Windows Azure account, go to http://windowsazure.com.

To get started login to the Windows Azure management portal at https://manage.windowsazure.com.

2.1 Windows Azure Cloud Service

First of all you need to have a Windows Azure Cloud Service that is going to run the virtual servers hosting the hybris platform.

1. At the bottom of the page click the "new" button and navigate to "compute" > "cloud service" and select "custom create".
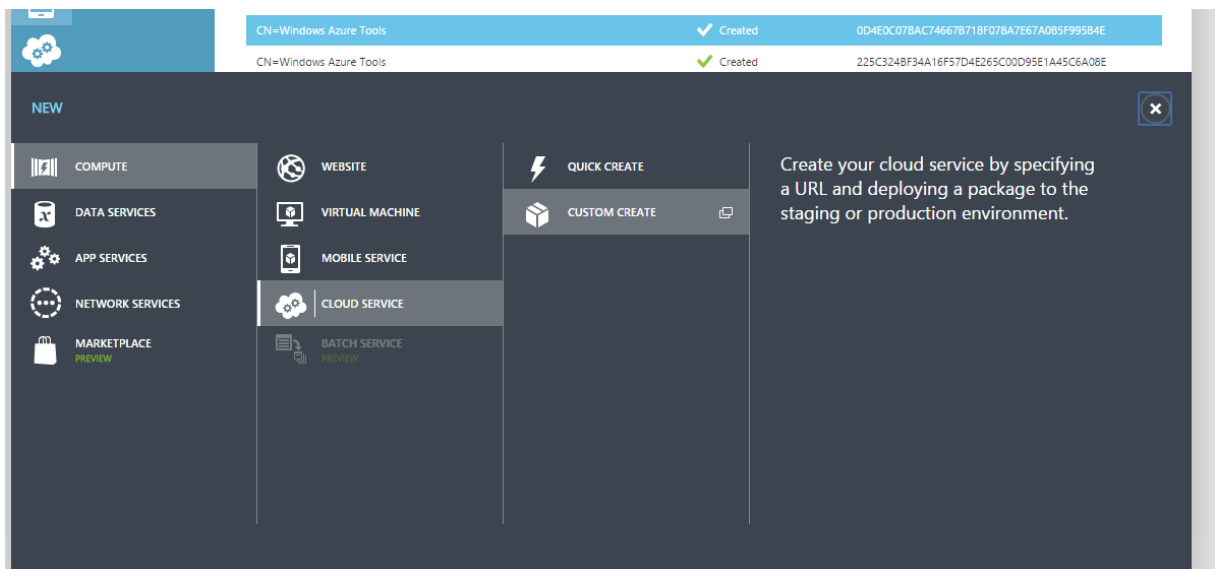
2. Enter a name for your cloud service and select a region in which the servers will be deployed.

> **Sample Configuration:** The cloud service in this sample is named "hybrisdemo" and is hosted in North Europe
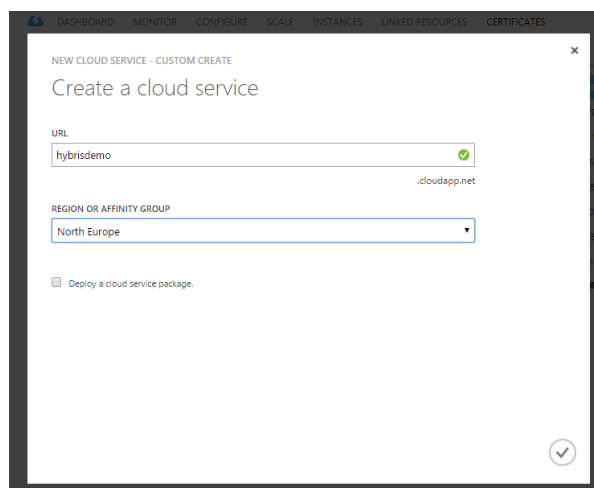
3. Do not check the "Deploy a cloud service package" checkbox.
4. Click the OK button and have Windows Azure create the Cloud Service for you.

You are going to deploy the cloud service package containing the hybris platform later. First there are other assets to be prepared.
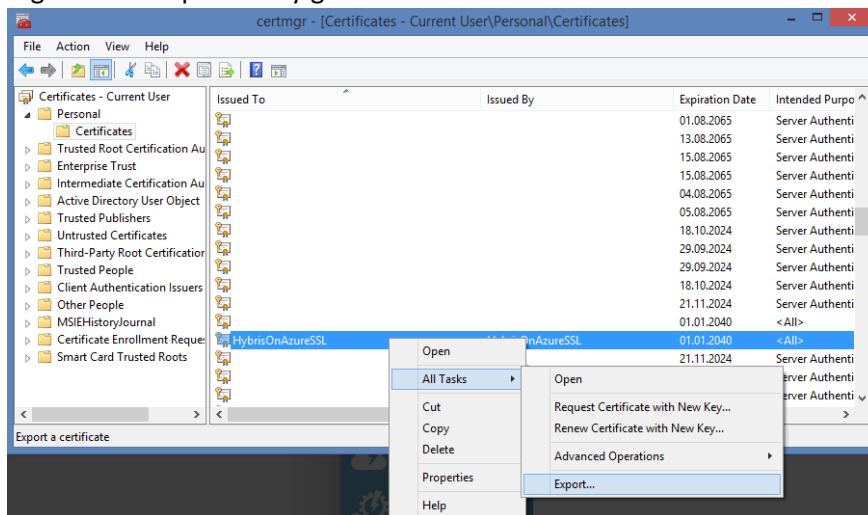
2.2 Windows Azure SSL Certificate

If you want your hybris deployment to support SSL connections to customers, you need to provide a SSL certificate. For test purposes a self-signed certificate will suffice.

Create the certificate

1. Open a Visual Studio command prompt as an administrator
2. Run the following command
   ```
   makecert –sky signature –r –pe –n "CN=HybrisOnAzureSSL" –ss my
   ```
3. Open the certificate manager (certmgr.msc)
4. Navigate to "Personal" > "Certificates" in the tree view
5. Right click the previously generated certificate and choose "All Tasks" > "Export"

6. Export the private key with the certificate, assign a password and export the certificate as a .pfx file.
7. Save the file e.g. on your desktop so you can find it easily in the next step.

4

Upload the certificate

1. In the windows Azure management portal navigate to the Cloud Service you created in chapter 2.1 Windows Azure Cloud Service" and select the "certificates" tab.
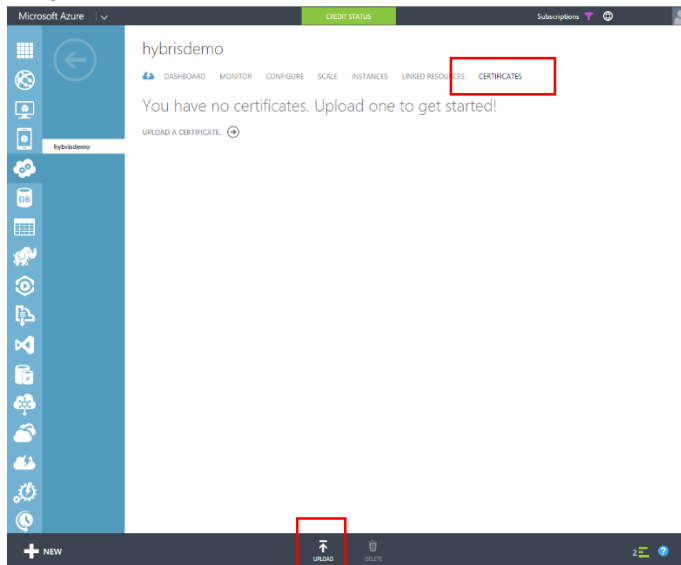
2. Click the "upload" button in the bottom menu bar.
3. Select the .pfx file, enter the password for the private key and click the "Complete" button.
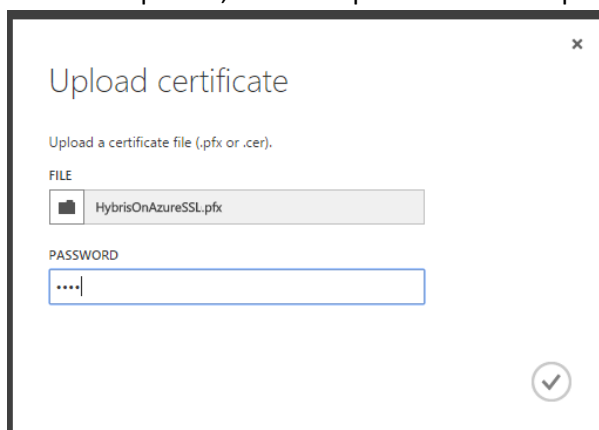
4. After the upload task finished the SSL certificate is displayed in the certificate list for the cloud service. Note the thumbprint of the certificate for later use.
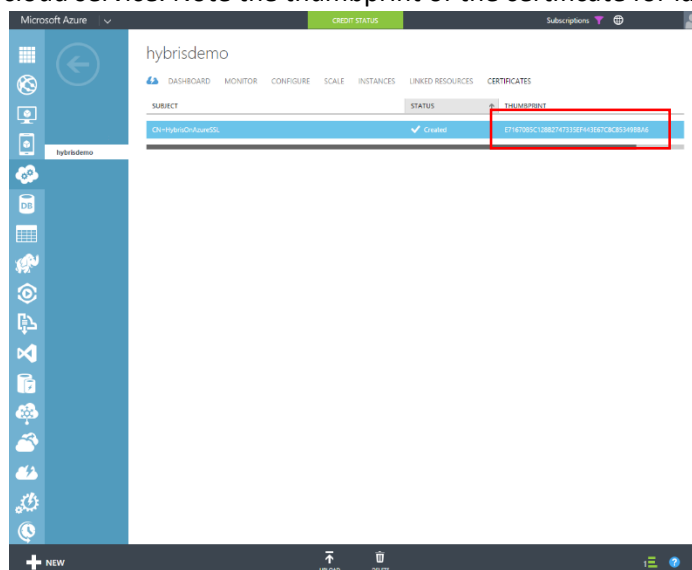
2.3 Windows Azure Management Certificate

In order to deploy from Visual Studio and maintain the deployment using administrative interfaces from the BackOfficeWorker a Management certificate is needed. A detailed guide how to create such a certificate can be found here: https://msdn.microsoft.com/en-us/library/azure/gg551722.aspx

Create the certificate

1. Open a Visual Studio command prompt as an administrator
2. Run the following command
   ```
   makecert –sky exchange –r –n “CN=HybrisOnAzureManagementCertificate”
   –pe –a sha1 –len 2048 –ss My “HybrisOnAzureManagementCertificate.cer”
   ```
3. Open the certificate manager (certmgr.msc)
4. Navigate to "Personal" > "Certificates" in the tree view
5. Right click the previously generated certificate and choose "All Tasks" > "Export"

6. Do NOT export the private key and save the certificate as a DER encoded binary X.509
7. Save the file e.g. on your Desktop so you can find it easily in the next step
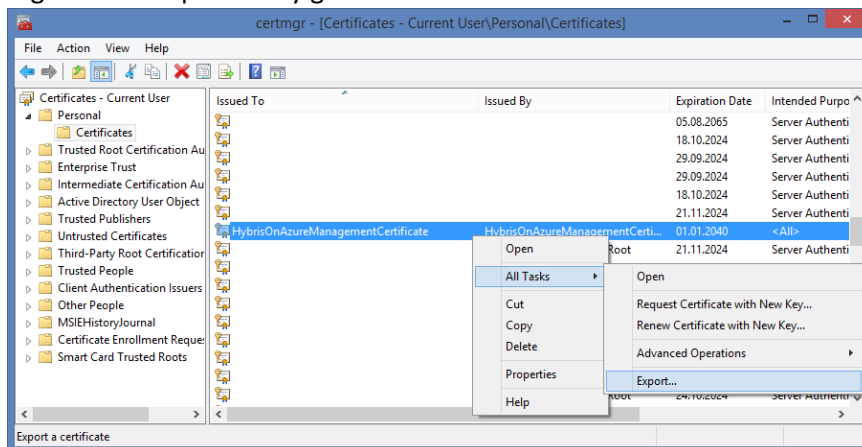
Upload the certificate

1. Open the Windows Azure portal
2. In the left bar navigate to "settings" and select the "Management Certificates" tab

6
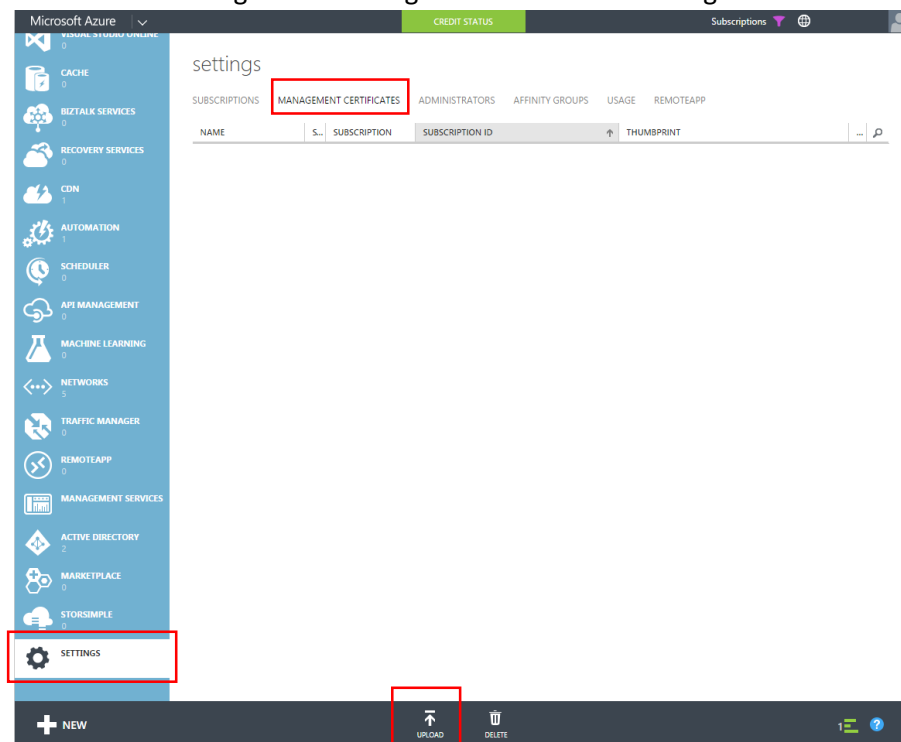
3. Click on the "upload" button and select your .cer certificate
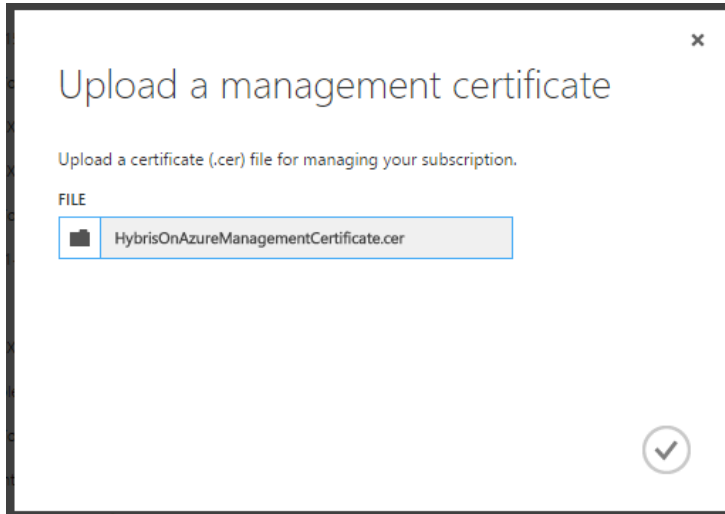
4. Note the thumbprint and subscriptionId of the certificate for later use
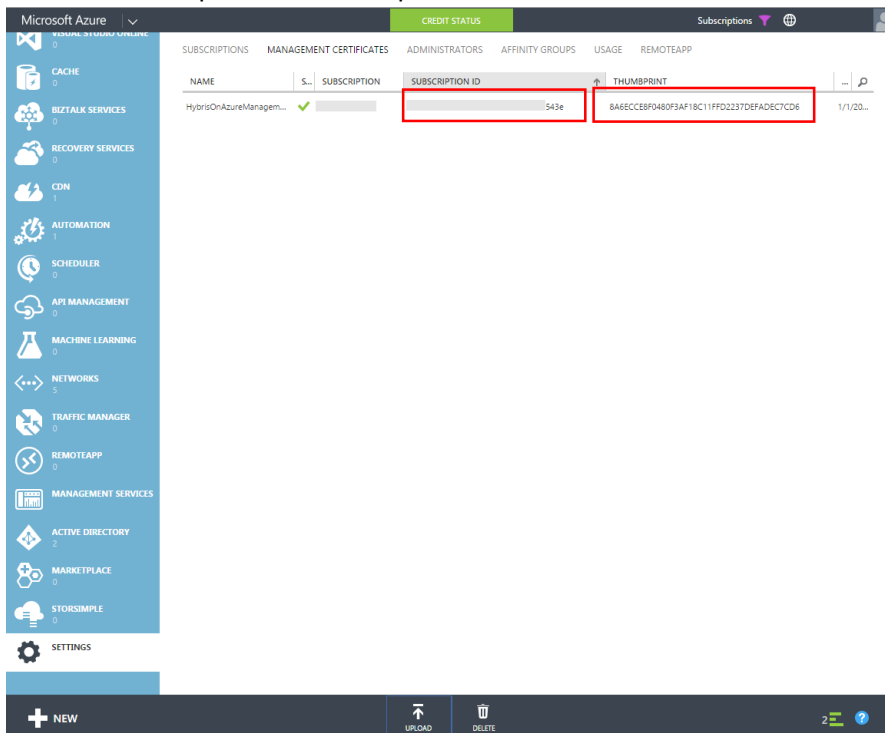
In order to use the management certificate from within the CloudService you also need to upload it and its private key to the CloudService as done with the SSL certificate in chapter 2.2 Windows Azure SSL Certificate.

5. Export the management certificate with its private key and password protect it.
6. In the Windows Azure management portal navigate to your CloudService you created in chapter 2.1 Windows Azure Cloud Service and select the "Certificates" tab.

7. Upload the management certificate

2.4 Windows Azure Virtual Network

In order to communicate with each other a virtual network is needed. This network needs to be created in the same location as the cloud service you created in chapter 2.1 Windows Azure Cloud Service.

SQL Server AlwaysOn:
When you are planning to use the SQL Server AlwaysOn setup provided by Windows Azure, you need to setup a virtual network when creating the SQL Server AlwaysOn setup. Please make sure the virtual network you setup meets the requirements of this section, so that the CloudService can create its machines in that network as well.
For a SQL Server AlwaysOn setup please refer to the latest Azure Portal at http://portal.azure.com.

If you are not planning to use SQL Server AlwaysOn, follow these steps to create a Virtual Network:

1. In the Windows Azure management portal, click the "new" button and navigate to "network services" > "virtual network" and select "custom create".
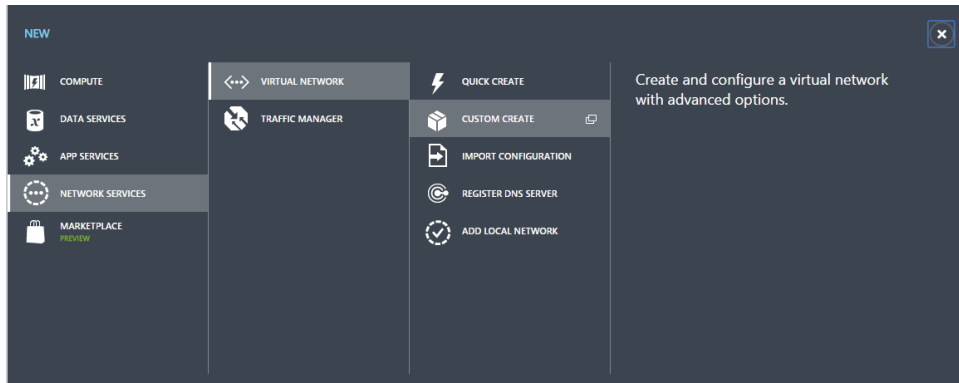
2. Enter a name for the virtual network and select the same location as for the cloud service created in chapter 2.1 Windows Azure Cloud Service. Then click on the "next" button.

Sample Configuration: The virtual network in this sample is named "hybrisdemo_net" and is hosted in North Europe

---

3. On the second page of the wizard you don't need to setup a special configuration. Just click on the „next" button.

4. On page three of the wizard, create an address space with 4096 addresses starting at 10.11.0.0. Within this space create two subnets:
"ArrNet" containing 256 addresses starting at 10.11.4.0 which will host the first tier
"FrontendNet" containing 256 addresses starting at 10.11.5.0 hosting hybris servers

5. Click the "complete" button to create the virtual network.

2.5 Windows Azure ServiceBus Namespace

For debugging and monitoring this solution uses a live trace feature based on the Windows Azure ServiceBus. But this feature uses an ACS Token instead of a SAS signature. Azure disabled this feature when creating a new ServiceBus namespace from within the Windows Azure Portal.

Sample Configuration: ServiceBus Namespace = `hybrisdemotrace`

In order to create a new ServiceBus namespace with ACS enabled, the following steps need to be performed:

1. Install Windows Azure PowerShell (see http://azure.microsoft.com/en-us/documentation/articles/install-configure-powershell/#Install)
2. Have a Windows Azure Management certificate (.pfx) uploaded to the Windows Azure Portal to manage subscriptions and install it locally in the CurrentUser\My storage.
   A detailed HowTo can be found in chapter "2.3 Windows Azure Management Certificate".
   Note the Thumbprint (e.g. `AAAA3217F1E66E4CD3DD7273CEC0D2FCB2DCBBBB`)
3. Open the Windows Azure PowerShell
4. Select the management certificate
   ```
   PS C:\> $cert = Get-Item
   Cert:\CurrentUser\My\AAAA3217F1E66E4CD3DD7273CEC0D2FCB2DCBBBB
   ```
5. Set the Windows Azure subscription where you want to create the Namespace
   ```
   PS C:\> Set-AzureSubscription -SubscriptionName "mysubscription" -
   SubscriptionId "8AF94362-921D-48F2-BB65-B7DC24179975" -Certificate
   $cert
   ```
6. Select the Windows Azure subscription where you want to create the Namespace
   ```
   PS C:\> Select-AzureSubscription -SubscriptionName "mysubscription"
   ```

7. Create a new ServiceBusNamespace
    ```
    PS C:\> New-AzureSBNamespace -Name "newnamespace" -Location "North
    Europe" –NamespaceType "Messaging"
    ```

    ```
    PS C:\> New-AzureSBNamespace -Name "hybrisdemotrace" -Location "North Europe" -NamespaceType "Messaging"

    Name                  : hybrisdemotrace
    Region                : North Europe
    DefaultKey            : oS9bQwH6IdKQul7w3BpPKR+dr5zigmOqzCEpT+2jUYg=
    Status                : Active
    CreatedAt             : 06.02.2015 14:17:59
    AcsManagementEndpoint : https://hybrisdemotrace-sb.accesscontrol.windows.net/
    ServiceBusEndpoint    : https://hybrisdemotrace.servicebus.windows.net/
    ConnectionString      : Endpoint=sb://hybrisdemotrace.servicebus.windows.net/;SharedSecretIssuer=owner;SharedSecretValu
                            e=oS9bQwH6IdKQul7w3BpPKR+dr5zigmOqzCEpT+2jUYg=


    PS C:\>
    ```

8. Note the DefaultKey for later use.

2.6 Windows Azure Storage Account

Before creating Windows Azure Storage accounts please visit the Windows Azure Preview Portal at http://azure.microsoft.com/en-us/services/preview/ and sign up for the "Windows Azure Files Preview" Feature using the administrative Account for the Azure Subscription you are using.

In order to provide an automatic server setup the hybrisOnAzure for all solution uses a Windows Azure blob storage containing the hybris platform and configuration files. This also allows an update of the hybris platform without having to recreate the cloud service. Additionally media files need to be accessible for all hybris servers simultaneously, so they will be stored in a separate blob storage.

Create **two** Windows Azure storage account using the following steps:

1. In the Windows Azure management portal click the "new" button and navigate to "data services" > "storage" and select "quick create".
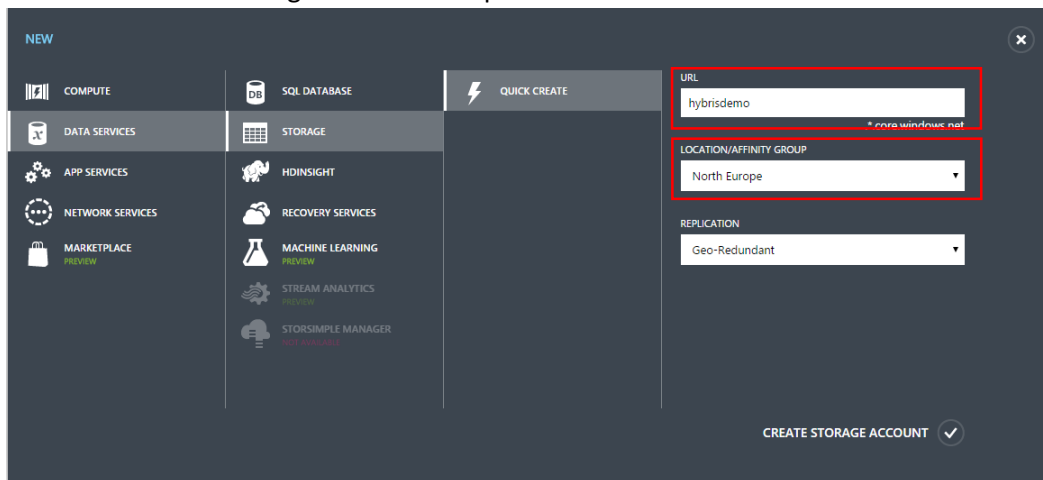
2. Enter a name for the storage account and select the same location as for the cloud service created in chapter 2.1 Windows Azure Cloud Service.

> **Sample Configuration:** In this sample the storage account is named "hybrisdemo". The storage account providing media files is named "hybrisdemocontent".
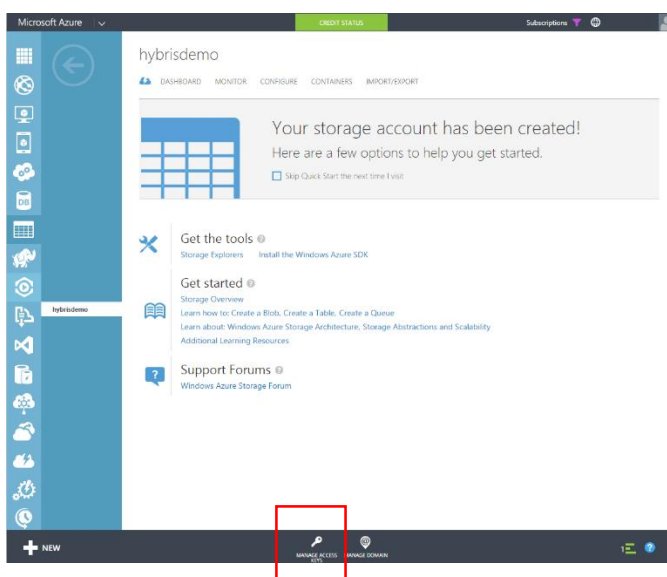
3. Click on the "Manage access keys" button in the bottom menu bar and note the primary access key for later use.
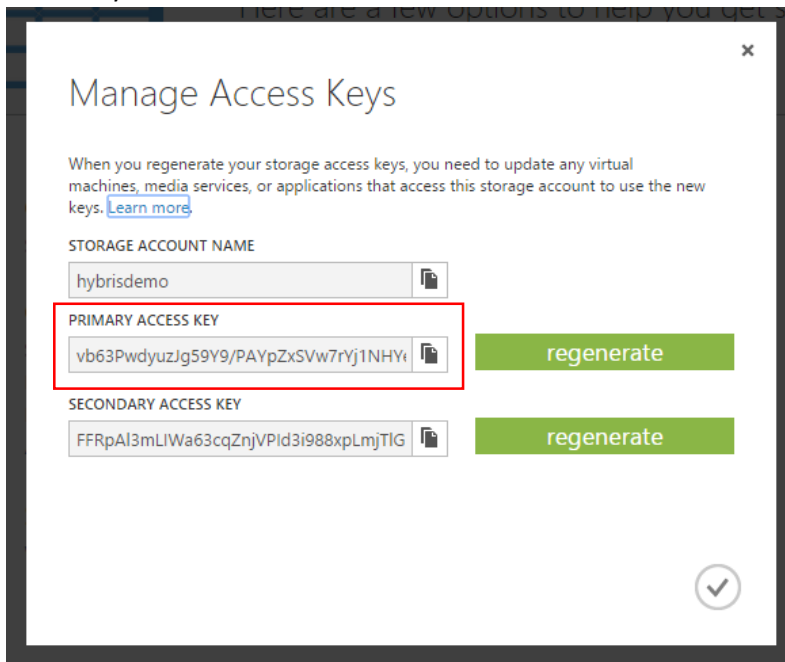
All files required for the automated server setup need to be stored within a blob container. You can create a container within the Windows Azure management portal but managing blobs within containers require a separate tool. There are lots of free tools available. This sample uses the "Cloud Berry Storage Explorer".

4. Connect to the Windows Azure Storage account that is to be used to deploy the hybris platform using the account's name and primary access key from step 3 as credentials.
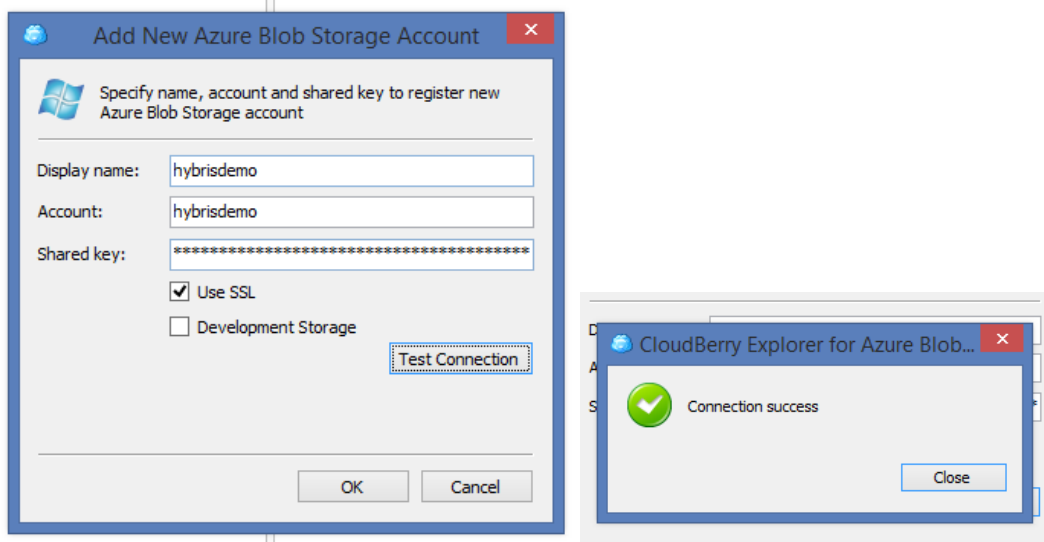
5. Create a new container by clicking the "new container" symbol in the menu bar and name it "deployment".
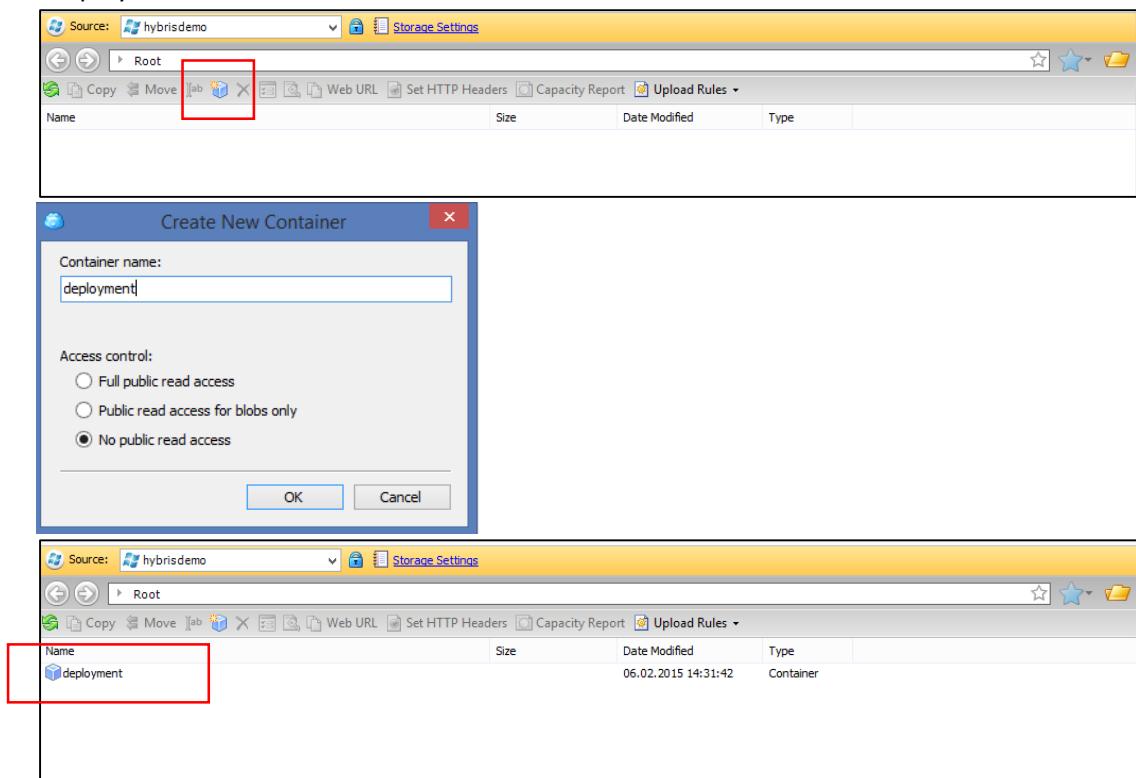
2.7 Windows Azure Files

Before you can setup the Windows Azure Files feature, please ensure you signed up for the Windows Azure File Preview feature in the Preview portal at http://azure.microsoft.com/en-us/services/preview/ before creating the Storage Accounts in chapter 2.6 Windows Azure Storage Account.

In the hybrisOnAzure Architecture there is one server instance hosting hybris designated as a "BackOfficeWorker". It provides all the administrative interfaces on a different port for security reasons.

This master server gets access to a Windows Azure Files share where import/export data for hybris can be stored and integrated into the hybris setup.

Create a Windows Azure Files share using the following instructions:

1. Install Windows Azure PowerShell (see http://azure.microsoft.com/en-us/documentation/articles/install-configure-powershell/#Install)
2. Have the account name and key of the Windows Azure storage account created in chapter 2.6 Windows Azure Storage Account at hand.
3. Open the Windows Azure PowerShell
4. Create a context object for the storage account (replace name and key)
    ```
    PS C:\> $ctx = New-AzureStorageContext <account name> <account key>
    ```
5. Create a new share using that context
    ```
    PS C:\> New-AzureStorageShare <share name> -Context $ctx
    ```

Sample Configuration: In this sample the storage account is named "hybrisdemo" and the share name "backofficeshare".

2.8 SQL Azure

A convenient way to have a database in Windows Azure is to use SQL Azure. SQL Azure provides a scalable and high available SQL database without having to care for the servers yourself. Using a hybris version 5.4.0.0 or higher enables the usage of SQL Azure.

Use the following steps in order to create a new SQL Azure database for your deployment:

1. In the Windows Azure Management Portal navigate to "new" > "Data Services" > "SQL Database" > "Custom Create"
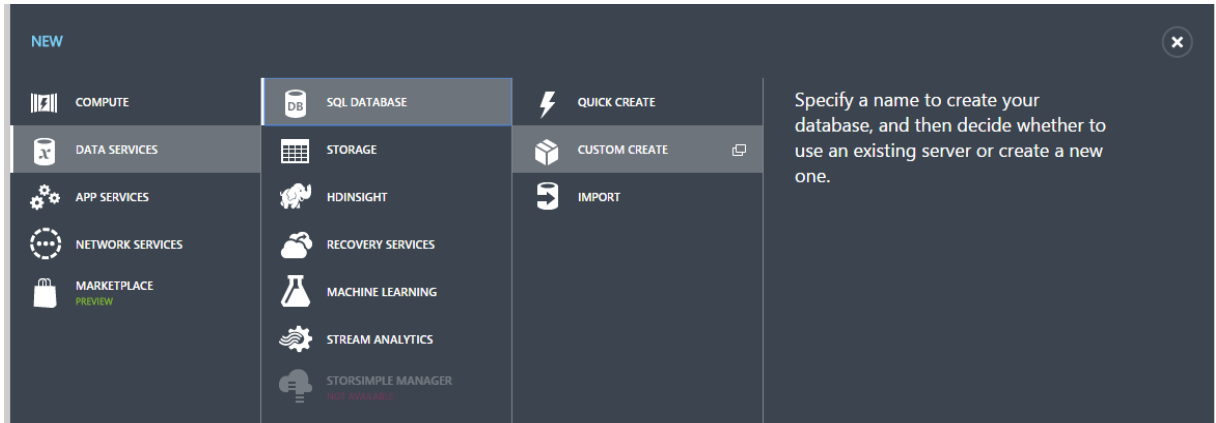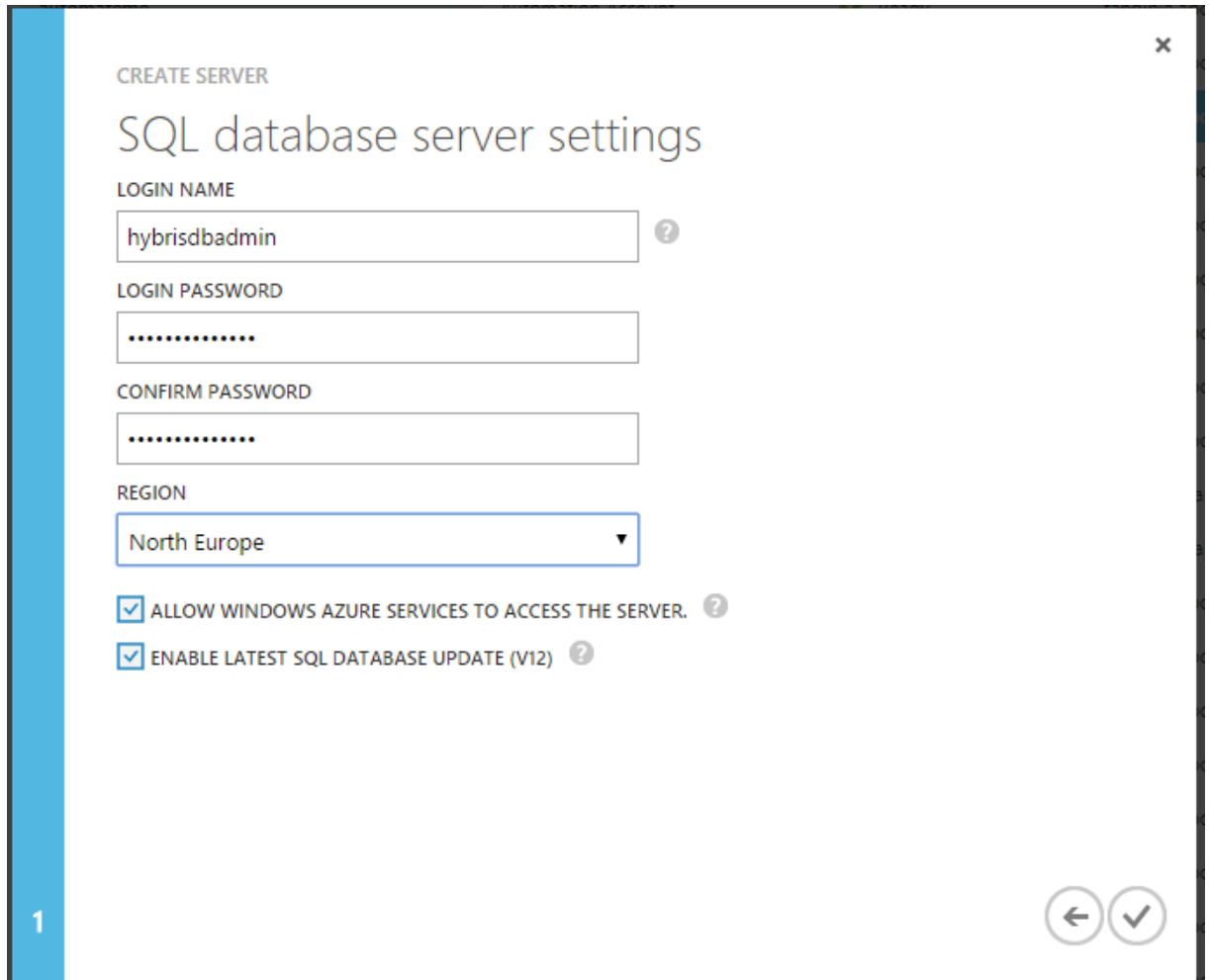
2. On the first page of the wizard: enter a name for the database, select "Premium" as service tier, "P3" as Performance level and have Azure create a new SQL Server.

Using a performance level of P3 will speed up the database initialization process. After initialization you might consider reducing costs by scaling down to P1.

3. Configure the new SQL Server on the second page of the wizard. Specify a user name and a password and select the region in which the database should be created. The region should match with the region you created the Cloud Service in.
Make sure both check boxes are checked to ensure the cloud service may access the database. hybris requires the latest SQL database update (V12).

4. Click the "OK" button to create the SQL Azure database and server.

Sample Configuration:
Database name: hybrisdemodb
Server name: mi67xuny
Login name: hybrisdbadmin
Login Password: P4ssw0rd
Location: North Europe

2.9 Java Runtime

The Java Runtime is required on your local machine in order to prepare the hybris platform as well as on all servers that will host hybris on Windows Azure.

> **Sample Configuration:** Java 1.7.0 Update 65 is used.
>
> Download here: http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html#jdk-7u65-oth-JPR

Install Java on your local machine

1. Download Java from the oracle website (link in the box above)
2. Install java using the setup wizard

Provision Java for Windows Azure

Java must be installed on each Windows Azure server on server startup. Best practice is to have a ZIP file inside the Windows Azure Blob Storage that is downloaded during the server startup. The hybrisOnAzure solution looks for a "java.zip" archive in the "deployment" container and will unzip it on each machine into the D:\Program Files\Java\ directory.

To prepare and provision this ZIP file, use the following instructions:

1. Have Java (64 bit) installed on your local machine
2. Navigate to the installation folder (e.g. C:\Program Files\Java\jdk1.7.0_65)
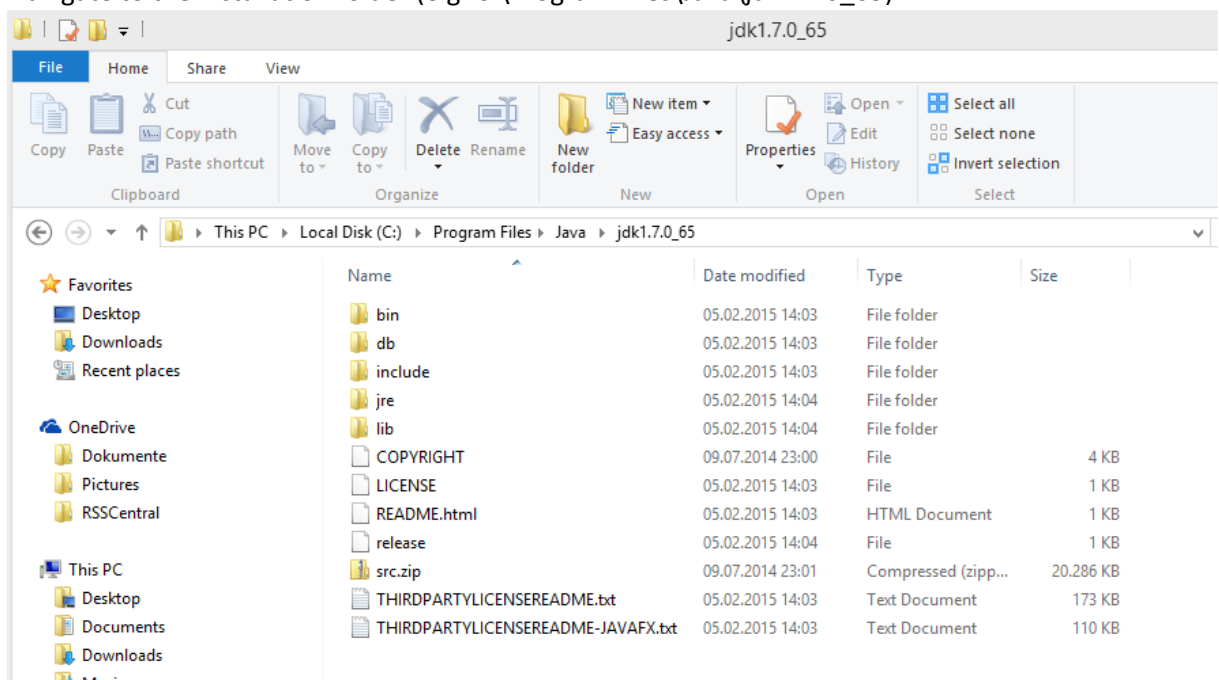
3. Compress all files and directories into a ZIP file named "java.zip"
   CAVE: be sure to only use small letters, because BLOB URIs are case sensitive.
4. Upload "java.zip" to the blob storage into container "deployment"

2.10 Application Request Routing

Using multiple server instances in the cloud requires running a web farm in which the servers can route request to one another. This ensures that requests from one session are always routed to the same machine, no matter where the request is routed to by the Azure Load Balancer in the first place.

To enable these features additional setup files are needed. Follow these steps to create a ZIP file that contains all required setup files and configuration commandlets:

1. Download the WebFarm Framework 1.1 setup file from
   http://download.microsoft.com/download/5/7/0/57065640-4665-4980-A2F1-4D5940B577B0/webfarm_v1.1_amd64_en_US.msi
   and rename it to "**webfarm_amd64_en-US.msi**"

2. Download the Application Request Routing 3.0 setup file from
   http://www.microsoft.com/en-us/download/details.aspx?id=39715
   and rename it to "**requestRouter_amd64_en-US.msi**"

3. Download the External Disk Cache V1 setup file from
   http://download.microsoft.com/download/3/4/1/3415F3F9-5698-44FE-A072-D4AF09728390/ExternalDiskCache_amd64_en-US.msi

4. Download the Patch for the External Disk Cache from
   http://download.microsoft.com/download/D/E/9/DE90D9BD-B61C-43F5-8B80-90FDC0B06144/ExternalDiskCachePatch_amd64.msp

5. Put these file alongside with the cmd-files "ConfigWebFarm.cmd" and "startup.cmd" from the Directory "Solution Items\Application Request Routing" shipped with the HybrisOnAzure - For All package in a Zip file named "**ARR30.zip**".

   Your Zip file should look like this:

PC ▸ Data (D:) ▸ hybrisOnAzure-ForAll ▸ Solution Items ▸ Application Request Routing ▸ ARR30.zip				
Name	Type	Compressed size	Password ...	Size
ConfigWebFarm.cmd	Windows Command Script	1 KB	No	
ExternalDiskCache_amd64_en-US....	Windows Installer Package	2.932 KB	No	
ExternalDiskCachePatch_amd64.msp	Windows Installer Patch	67 KB	No	
requestRouter_amd64_en-US.msi	Windows Installer Package	1.661 KB	No	
startup.cmd	Windows Command Script	1 KB	No	
webfarm_amd64_en-US.msi	Windows Installer Package	319 KB	No	

6. Upload the "ARR30.zip" to the blob storage container "deployment".

3. Preparing hybris

Hybris needs to be built and configured before shipping to the cloud. This is where the customization of hybris should take place. Before hybris can be built Java must be installed on your machine.

> **Sample Configuration:** Hybris Commerce Suite 5.4.0.0 is used in this sample

In order to build and configure hybris, use the following steps:

1. unzip commerce suite to `C:\Applications`
   This directory should be used to assure it matches the later installation path on Windows Azure
2. Open a command shell
3. Navigate to the platform directory:
   `cd C:\Applications\hybris\bin\platform`
4. Set variables for the ANT environment:
   `setantenv.bat`
5. Make a clean build to create a dummy configuration folder:
   `ant clean`
6. adapt C:\Applications\hybris\config\local.properties file to fit your needs
   Use the SQL Azure information you created in chapter 2.8 SQL Azure
   Use the information of the storage account dedicated for static hybris content you created in chapter 2.6 Windows Azure Storage Account

> **Sample Configuration:**
>
> ```
> # Official "Microsoft JDBC Driver for SQL Server" connection settings
> db.url=jdbc:sqlserver://mi67xuny.database.windows.net:1433;database=hybr
> isdemodb;encrypt=true;hostNameInCertificate=*.database.windows.net;login
> Timeout=30;
> db.driver=com.microsoft.sqlserver.jdbc.SQLServerDriver
> db.username=hybrisdbadmin
> db.password=P4ssw0rd
> db.tableprefix=pre_
> db.pool.testWhileIdle=false
> db.pool.testOnReturn=true
> db.pool.maxActive=90
> db.pool.maxIdle=5
> db.pool.timeBetweenEvictionRunsMillis=5000
> db.pool.minEvictableIdleTimeMillis=60000
>
> #azure cloud connection
> media.default.storage.strategy=windowsAzureBlobStorageStrategy
> media.default.url.strategy=windowsAzureBlobURLStrategy
>
> media.globalSettings.windowsAzureBlobStorageStrategy.local.cache=true
> media.globalSettings.windowsAzureBlobStorageStrategy.cleanOnInit=true
> media.globalSettings.windowsAzureBlobStorageStrategy.connection=DefaultE
> ndpointsProtocol=http;AccountName=hybristestcontent;AccountKey=pgxi61U2w
> NBegGJgXjYT9kuTduthIphZXAQ0KKvdqpR68C2GtwxuEgQUT5jVSo4ciX6tDvQsggTqEGM0O
> Kxc+g==
> media.globalSettings.windowsAzureBlobStorageStrategy.public.base.url=hyb
> ristestcontent.blob.core.windows.net
> ```

7. adapt C:\Applications\hybris\config\localextensions.xml to include desired extensions
   make sure the "azurecloud" extension is include in your configuration

Sample Configuration: This sample uses the following extensions

```xml
<extensions>
        <path dir="${HYBRIS_BIN_DIR}" />

        <!-- ext-platform -->
        <extension name="admincockpit" />
        <extension name="backoffice" />
        <extension name="cockpit" />
        <extension name="hmc" />
        <extension name="mcc" />
        <extension name="platformhmc" />

        <!-- ext-commerce -->
        <extension name="btg" />
        <extension name="btgcockpit" />
        <extension name="commercesearch" />
        <extension name="commercesearchbackoffice" />
        <extension name="commercesearchhmc" />
        <extension name="commerceservices" />
        <extension name="basecommerce" />
        <extension name="commercefacades" />

        <!-- ext-data -->
        <extension name="electronicsstore" />

        <!-- ext-content -->
        <extension name="productcockpit" />
        <extension name="cms2" />
        <extension name="cms2lib" />
        <extension name="cmscockpit" />

        <!-- ext-template -->
        <extension name="ycommercewebservices" />
        <extension name="ycommercewebserviceshmc" />
        <extension name="yacceleratorfulfilmentprocess" />
        <extension name="yaddon" />
        <extension name="yacceleratortest" />
        <extension name="yacceleratorstorefront" />
        <extension name="yacceleratorinitialdata" />
        <extension name="yacceleratorcockpits" />

        <!-- ext-accelerator -->
        <extension name="acceleratorservices" />
        <extension name="acceleratorfacades" />
        <extension name="acceleratorcms" />
        <extension name="acceleratorstorefrontcommons" />

        <!-- ext-accelerator -->
        <extension name="azurecloud" />
</extensions>
```

8. Build the hybris platform including your extensions:
   `ant clean all production` (builds platform)

With this build hybris creates ready-to-ship .zip files that can be copied to each Windows Azure Server. The hybrisForAll platform will download those files automatically from the "deployment" container in the Blob Storage.

9. upload C:\Applications\hybris\temp\hybris\hybrisServer\hybrisServer-AllExtensions.zip into the "deployment" container
10. upload C:\Applications\hybris\temp\hybris\hybrisServer\hybrisServer-Platform.zip into the "deployment" container
11. Navigate to C:\Applications\hybris
12. Create a new directory named "hybris" and copy the "config" directory into the "hybris" subdirectory
13. Right click the "hybris" directory you created in setp 12 and select "Send to" > "Compressed (zipped) folder".
14. Name it "hybrisServer-Config.zip"
15. Upload the hybrisServer-Config.zip into the "deployment" container

If you want to include a standalone solr server with your deployment, use these additional steps:

16. Navigate to C:\Applications\hybris\bin\ext-commerce\solrfacetsearch\resources\solr\server\
17. Compress all files of this folder into "solr.zip"
18. Upload the solr.zip into the "deployment" container.

4. Preparing deployment

4.1 Upload additional files

In Addition to what was prepared in the chapters 2 and 3 some other files need to be placed inside the "deployment" container in the deployment Blob Storage. These files are available at the hybris on azure GitHub page in the "Solution Items" directory:

https://github.com/Microsoft/microsoft-deployment-accelerator-for-hybris-on-azure/tree/master/Solution%20Items

- **JDBCSetup.cmd** (directory: \JDBC Driver)
  This script applies changes to the Windows Registry regarding the JDBC database driver. This file does not need to be changed.

- **RunJar.zip** (directory: \Run a JAR on startup)
  Put "MyJavaProgram.jar" and "RunJar.cmd" into a new ZIP File named "RunJar.zip".
  This is a sample of how to run a user defined Java application on server startup.
  See chapter 7.1 Run custom Java application on startup.

- **ConnectAzureFiles.cmd** (directory: \AzureFiles)
  This script connects the BackOfficeWorker to the Windows Azure Files share created in chapter 2.7 Windows Azure Files.

- **Solrconfig.xml** and **solrserver.bat** (directory: \Solr Search)
  These files configure the standalone solr server.

- **StopTanukiWrapper.jar** (directory: \StopTanukiWrapper)

Your "deployment" container should now look like this:
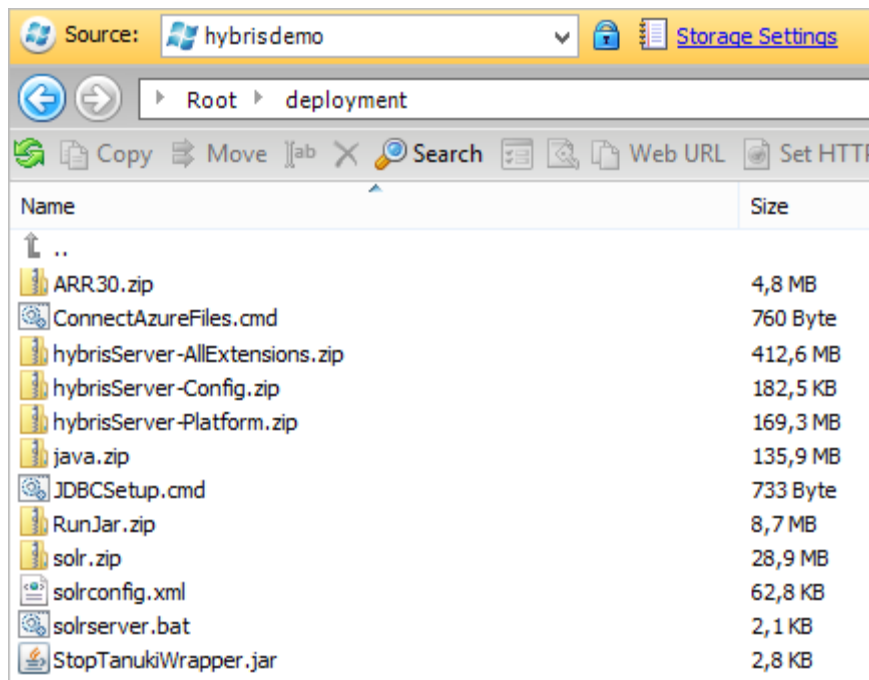
In anticipation of chapter 6. Maintenance you need to upload the following file to another container named "config":

- **Configuration.xml** (directory: \Additional Configuration)
  This file stores configuration information for each single server in your deployment and allows you to determine reboots or maintenance modes for each server.
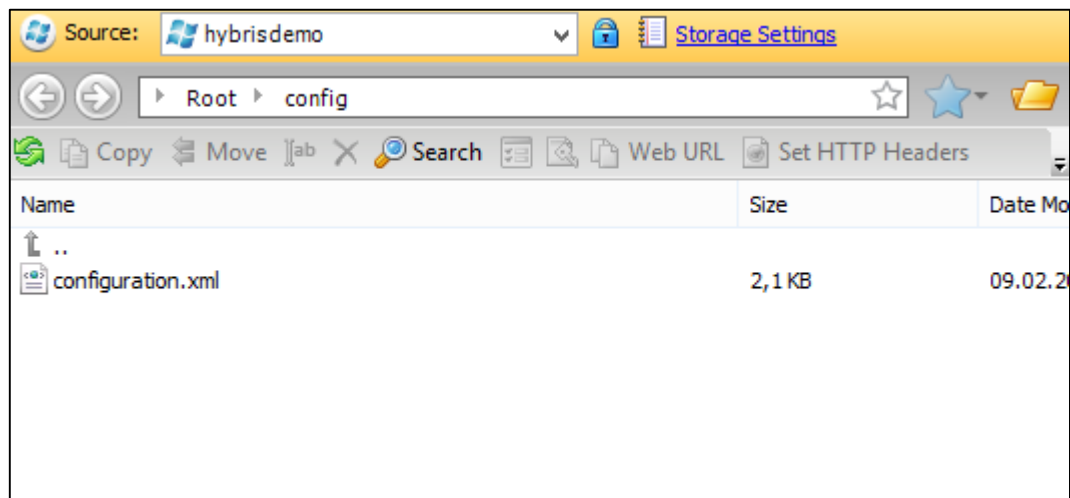
4.2 Prepare Windows Azure package

In this chapter required changes need to be made to the configuration files to match the previously created Windows Azure features. This is done by altering the ServiceConfiguration.cscfg file using a Text editor.

Set certificates

Locate the XML Elements `<Certificate name="SSL-Certificate" … />` and replace the thumbprint value with the thumbprint of the SSL Certificate created in chapter 2.2 Windows Azure SSL Certificate. There are two XML Elements: one for the role `"tangible.HybrisOnAzure.BackOfficeWorkerRole"` and one for the `"tangible.HybrisOnAzure.ArrRole"`.

NOTE: It is highly recommended not to copy&paste values from the certificate manager directly into the .cscfg file, because it may insert hidden characters which will render the .cscfg invalid.
For reference see: this form entry at microsoft.com

Set configuration settings

- **Blob Storage Connection:**
  For each role replace AccountName and AccountKey inside the Values of settings `"StorageConnectionString"` and `"Microsoft.WindowsAzure.Plugins.Diagnostics.ConnectionString"` with values from the Blob Storage created in chapter 2.6 Windows Azure Storage Account.

- **Service Bus Connection:**
  For each role replace the values of `"tangible.Azure.Trace.ServiceNamespace"` and `"tangible.Azure.Trace.IssuerSecret"` with values from the ServiceBus Namespace created in chapter 2.5 Windows Azure ServiceBus Namespace.

- **Administration Page settings:**
  For management purposes the administration page needs access to the Cloud Service that hosts the hybris platform. Provide the following information:

  `<Setting name="HybrisOnAzure.DeploymentSlot" value="…" />`: set value to "production" if you are deploying to the Production slot of the Cloud service, "staging" otherwise.

  `<Setting name="HybrisOnAzure.SubscriptionId" value="…" />`: set value to the Id of your subscription (e.g. as seen in chapter 2.3 Windows Azure Management Certificate).

  `<Setting name="HybrisOnAzure.ManagementCertThumb" value="…" />`: set value to the thumbprint of the management certificate you created and uploaded in chapter 2.3 Windows Azure Management Certificate.

  `<Setting name="HybrisOnAzure.HostedServiceName" value="…" />`: set value to the name of the Cloud Service you created in 2.1 Windows Azure Cloud Service (e.g. hybrisdemo)

- **Windows Azure File share (BackOfficeWorker):**
  `<Setting name="HybrisOnAzure.BackOfficeShare.AccountName" value="…" />`: set value to the name of the Storage Account you created in chapter 2.6 Windows Azure

Storage Account.

```
<Setting name="HybrisOnAzure.BackOfficeShare.AccountKey" value="…" />:
```
set value to the primary access key of the Storage Account you created in chapter 2.6
Windows Azure Storage Account.

```
<Setting name="HybrisOnAzure.BackOfficeShare.ShareName" value="…" />:
```
set the value to the name of the share you created in chapter 2.7 Windows Azure Files.

Set virtual network configuration

Locate the `<NetworkConfiguration>` Element and its child `<VirtualNetworkSite>`. Replace
the value of the attribute `"name"` with the name of the virtual network created in chapter 2.4
Windows Azure Virtual Network.

5. Deployment

This chapter covers the actual deployment process after all preparation has been done. The hybrisOnAzure platform is delivered as a ready-to-upload Windows Azure package (.cspkg file). It comes with a separate cloud configuration file (.cscfg file) that needs to be adapted to your setup. These adoptions have been made in chapter 4.2 Prepare Windows Azure package.

Now the package needs to be uploaded and after deployment the hybris platform needs to be initialized. In order to debug the deployment process please refer to chapter 6.1 Debug deployment.

5.1 Deploy from File System

To initiate the deployment process, use the following steps:

1. In the Windows Azure management portal navigate to the Cloud Service you created in chapter 2.1 Windows Azure Cloud Service.
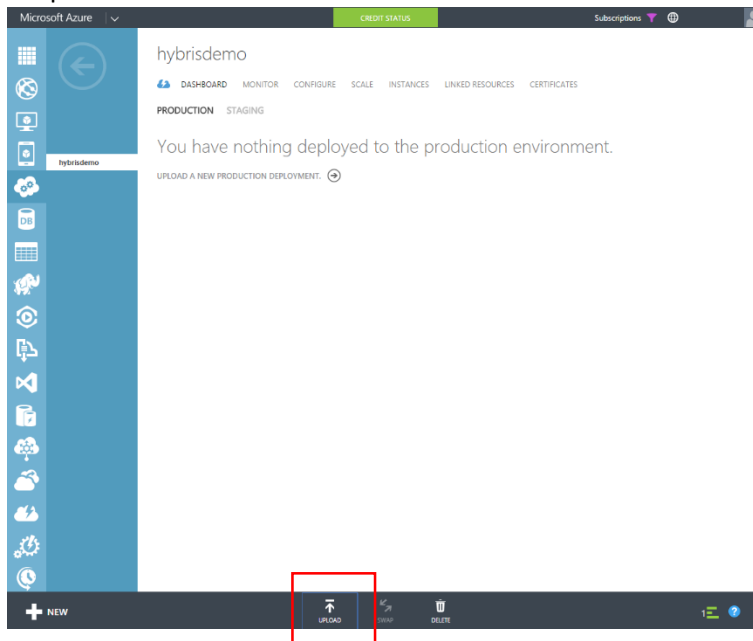
2. Click the "upload" button in the bottom menu bar.
3. Label your deployment and select the package file (.cspkg) and configuration file (.cscfg) from your local disk. The files are contained in the "Windows Azure Package" directory.

4. Make sure the "Deploy even if one or more roles contain a single instance" option is checked.
5. Click the "complete" button and wait for the deployment process to finish.

5.2 Initialize hybris platform

It may take several minutes for the deployment to finish. But after that process your hybris setup will be available at http://<cloudservice>.cloudapp.net/.

Sample Configuration: the base url for this sample is http://hybrisdemo.cloudapp.net/

Although at the beginning the administration console is visible, the management consoles are not accessible via the default url. Only the BackOfficeWorker deliveres those administrative endpoints and can be found at port 8080.
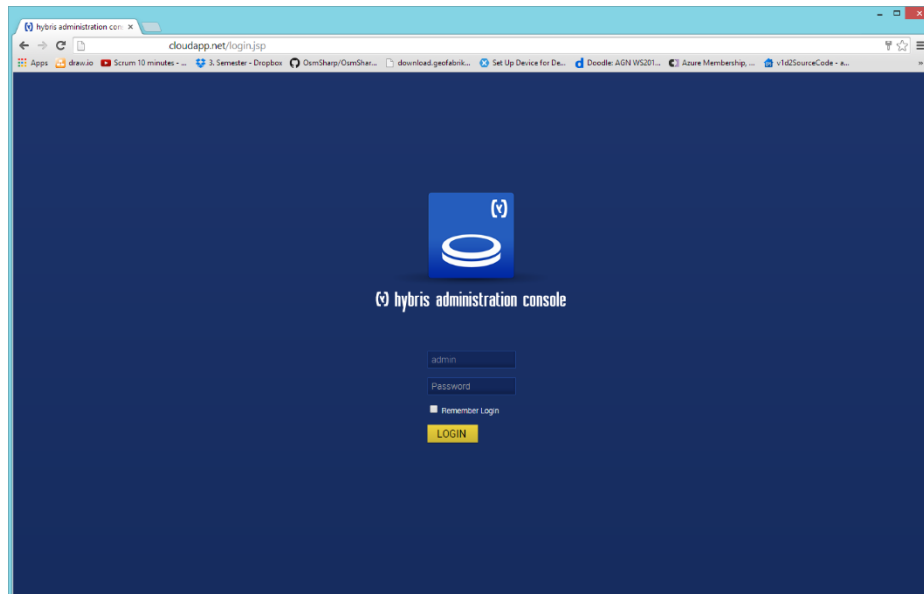
Sign in to the hybris administration console at https://<cloudservice>.cloudapp.net:8080/ using the user name "admin" and password "nimda".

Hybris detects that the database you provided has not been initialized yet and offers you to initialize it now. Check that the database connection is the one you configured in chapter Preparing hybris and click the "initialize" button.
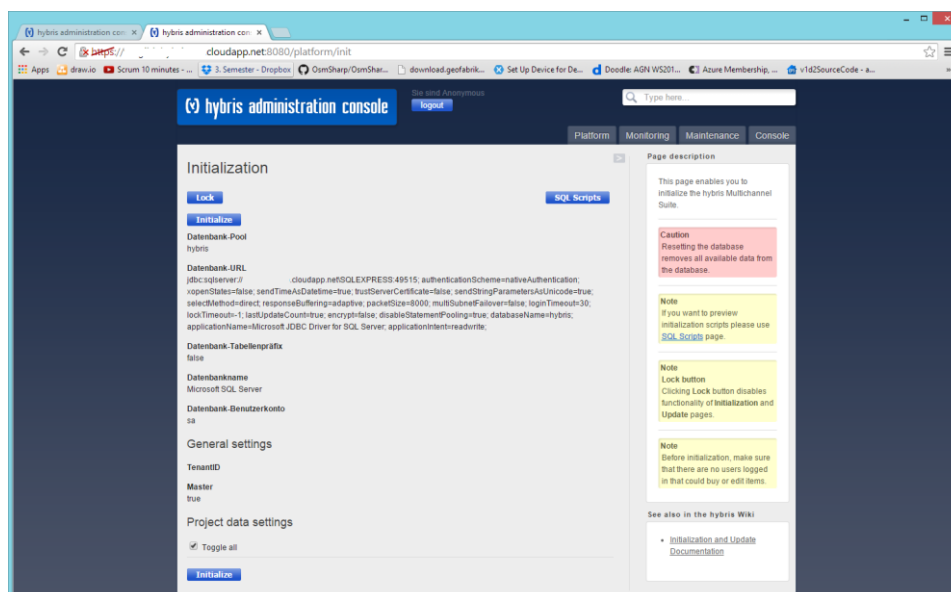
Depending on the extension you selected this may take a while.

After the initialization process finished, you need to make the media files accessible for your customers. When hybris created the container for the media files it restricted its access. In the Windows Azure management portal, navigate to the storage account for the media files you created in chapter 2.6 Windows Azure Storage Account.
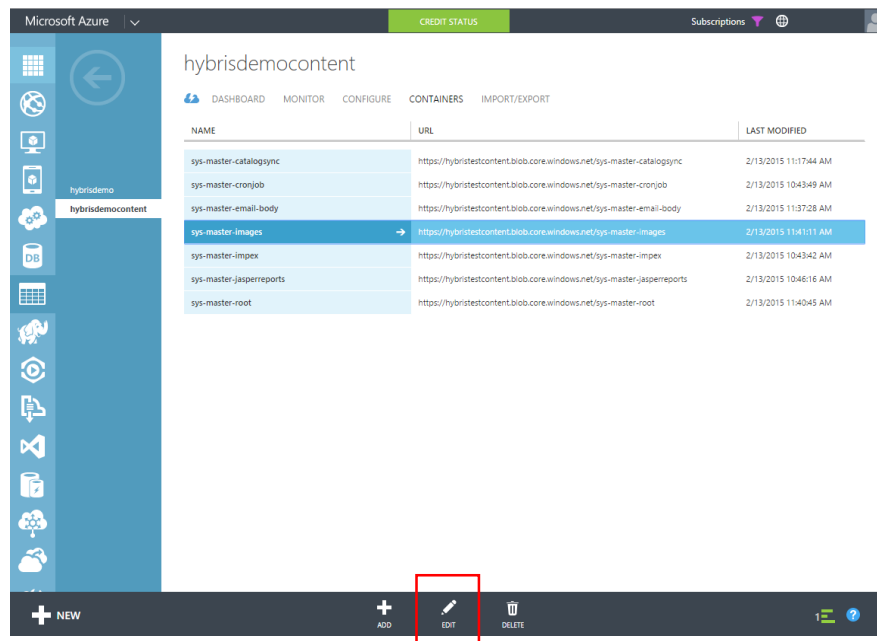
Click on the "containers" tab, and select the container created by hybris. Click "Edit" in the bottom tool bar.
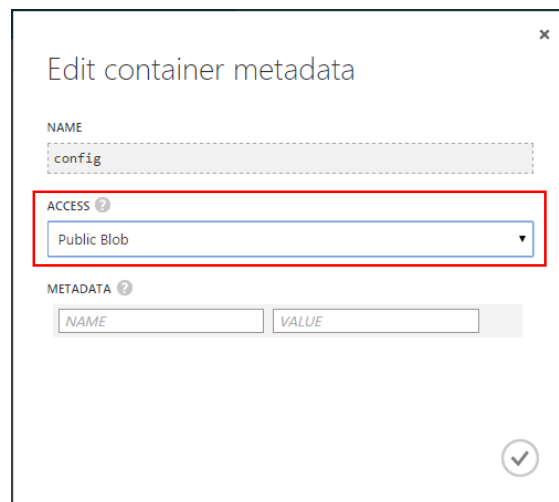
In the "Edit container metadata" dialog select "Public blob" as an access rule for this container. This rule grants access to all blobs (given you have the proper URL which hybris delivers) but denies access to browse containers and subdirectories.

Click the "OK" button.

5.3 Open the browser

If you used the localextensions.xml sample provided in chapter Preparing hybris", one of the default stores will be available at

http://<cloudservice>.cloudapp.net/yacceleratorstorefront/?site=electronics&clear=true
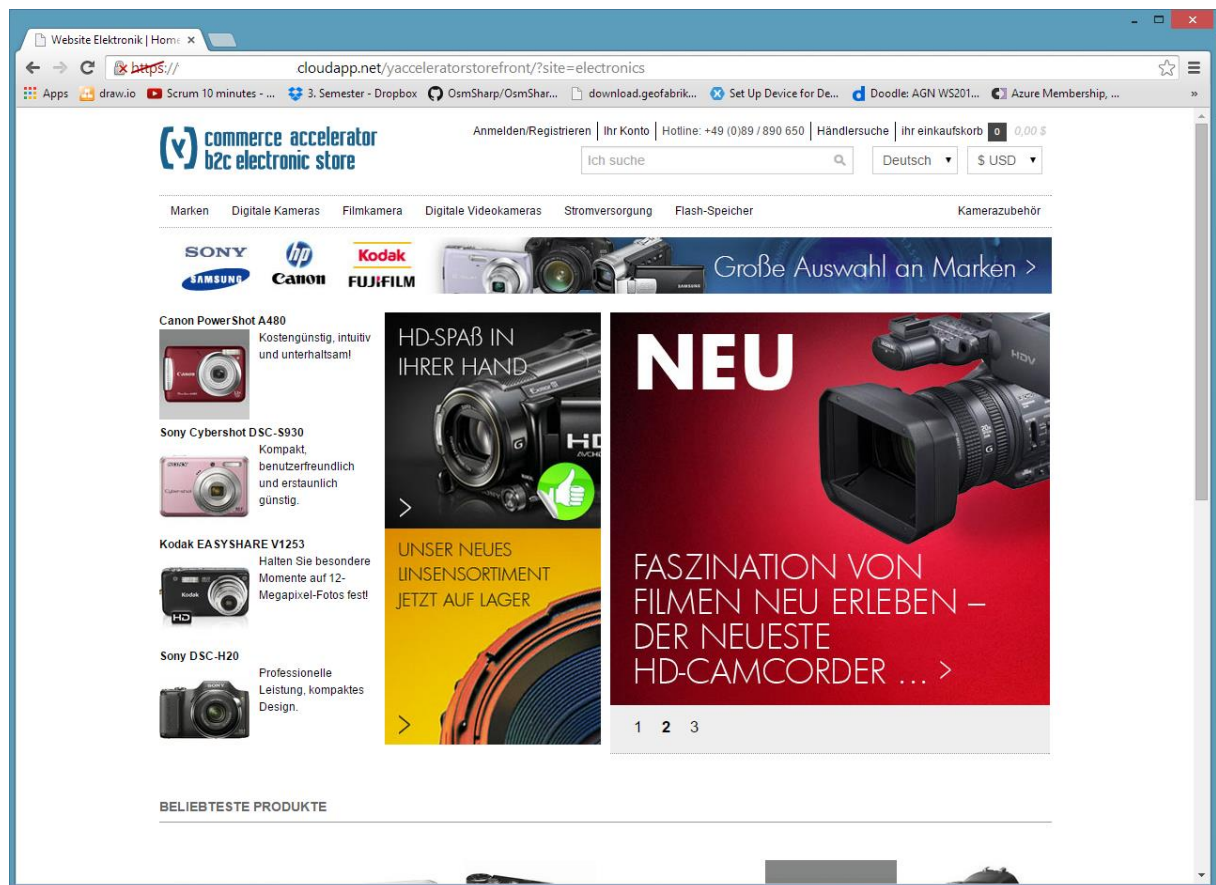
6. Maintenance

This chapter covers how to trace down and debug issues, make use of the admin page provided by the BackOfficeWorker and updating java or hybris packages on your deployment.

6.1 Debug deployment

If your deployment seems to take forever, the result is not what you expected, or you're just curious what happens behind the scenes, the hybrisOnAzure solution comes with a possibility to read the servers' Trace output via the Windows Azure ServiceBus.

Download the tangible.Azure.Tracing.TraceConsole for hybrisOnAzure from https://tangibletransfer.blob.core.windows.net/public/tangible.Azure.Tracing.TraceConsole.zip. Also download the free DebugView tool (https://technet.microsoft.com/en-us/library/bb896647.aspx) and put it into the "DebugView" directory next to the tangible.Azure.Tracing.TraceConsole.exe.

Edit the tangible.Azure.Tracing.TraceConsole.exe.config file and enter your ServiceBus credentials (namespace and default key) which you created in chapter 2.5 Windows Azure ServiceBus Namespace.

```
Sample Configuration:

<?xml version="1.0" encoding="utf-8">
<configuration>
  ...
  <add key="tangible.Azure.Trace.ServiceNamespace"
       value="hybrisdemotrace" />
  <add key="tangible.Azure.Trace.Credentials.IssuerName"
       value="owner" />
  <add key="tangible.Azure.Trace.Credentials.IssuerSecret"
       value="oS9bQwH6IdKQul7w3BpPKR+dr5zigmOqzCEpT+2jUYg=" />
  ...
</configuration>
```

Save the .config file and run the tangible.Azure.Tracing.TraceConsole.exe. It will open up a command window and run a DebugView window. In this window all Traces written to the Azure servers will be displayed and you can have a detailed look at what's going on there:
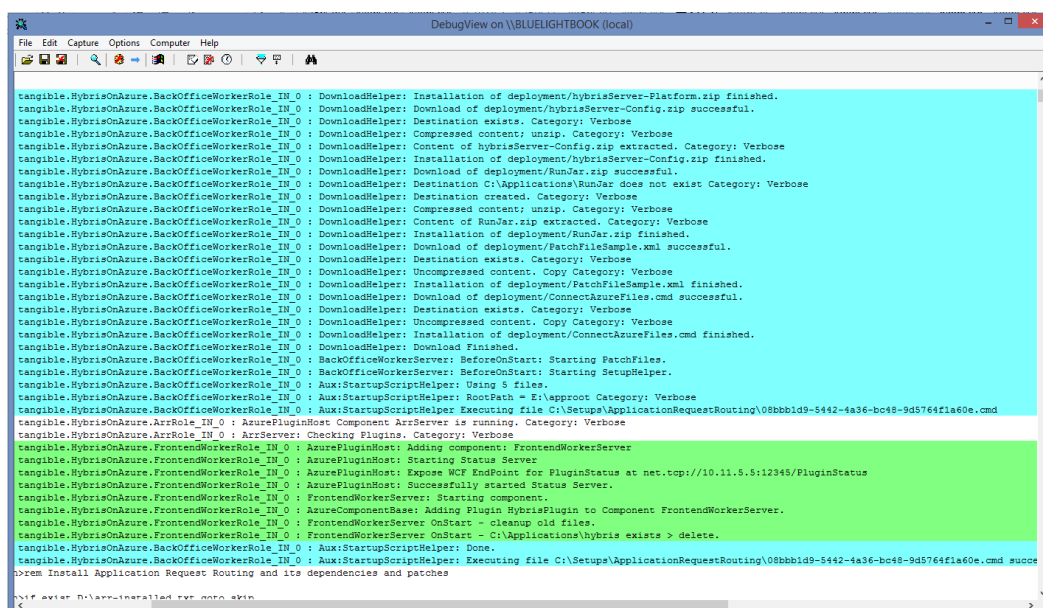
6.2 Using the Admin page

The BackOfficeWorker server in the hybrisOnAzure package provides an Administrative page available at http://<cloudservice>.cloudapp.net:8080/Admin.aspx. This page provides information about the current state of the deployment and allows several administrative features.

Each square containing an IP Address represents a server within the hybrisOnAzure deployment and displays its current status. When applying changes you need to provide a user name to be stored with this configuration change. These configurations are stored in the blob container "config".

Application Request Routing Tier

This area contains all servers of the Application Request routing tier. By checking the "Set maintenance" checkbox beneath one server enables a rewrite rule on that server causing all customer requests to be redirected to http://<cloudservice>.cloudapp.net/maintenance.html.

Using this feature you can reduce traffic to the Frontend Worker servers e.g. for a "warming up" phase after deployment.

When all ARR server are in maintenance mode, no traffic will reach the Frontend Worker server and you could for example do a hybris update.

You can also change the amount of ARR servers in your deployment causing new ones to be created or surplus ones to be shut down.

Frontend Worker

This area contains all servers of the Frontend Worker tier. You can cause single instances to reboot (and thus reload and rebuild the hybris platform) or stop hybris and solr processes on a single instance causing them to be taken out of traffic.

You can also change the amount of Frontend Worker server in your deployment causing new ones to be created or surplus ones to be shut down.

BackOffice Worker

This area contains the single instance of the BackOffice Worker server in your deployment. Like with the Frontend Workers you can cause it to reboot and reload and rebuild the hybris platform or stop its hybris and solr processes.

You cannot alter the amount of BackOffice Worker servers because there may be only one solr master running per hybris deployment.

6.3 Updating Java and hybris packages

The Admin page also allows you to specify which version of Java of hybris are to be deployed when a server instance boots.

Changing Java version

In order to change the Java version deployed, you need to package the new Java version the same way described in chapter 2.9 Java Runtime into a ZIP file. Then it needs to be uploaded into a subdirectory of the "deployment" container named "JavaPackages".

In the admin page you can now specify the name of the .zip file which should be deployed when a server reboots or starts up the first time. When you click the "Submit changes" button on the admin page, the selected .zip file is copied to "deployment/java.zip" which will be downloaded at server startup.
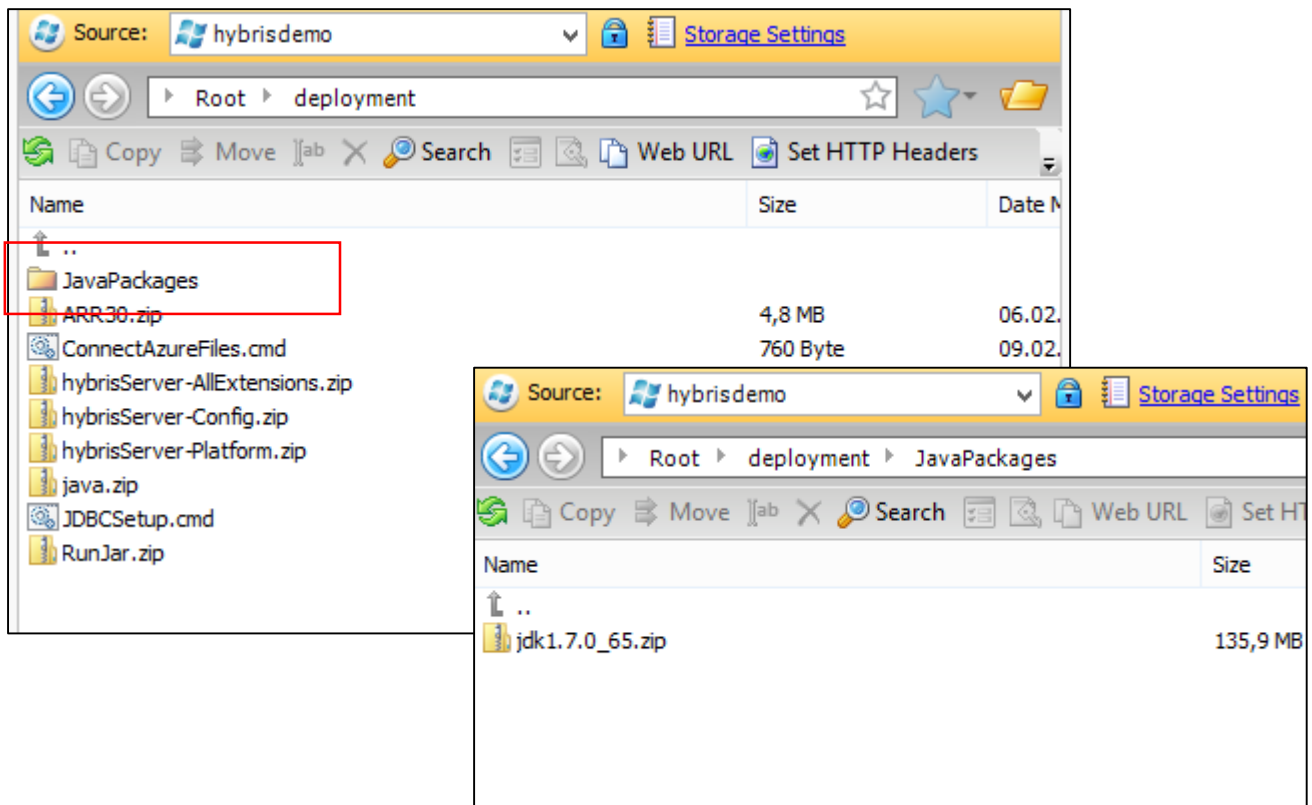
Changing hybris version

In order to change the hybris version deployed, you need to package the new hybris version the same way described in chapter Preparing hybris. Then you need to upload the three zip files (hybrisServer-AllExtensions.zip, hybrisServer-Platform.zip and hybrisServer-Config.zip) in a subdirectory of the "deployment" container named "HybrisPackages". The "HybrisPackages" subdirectory can contain multiple hybris versions as long as all are contained within another subdirectory.
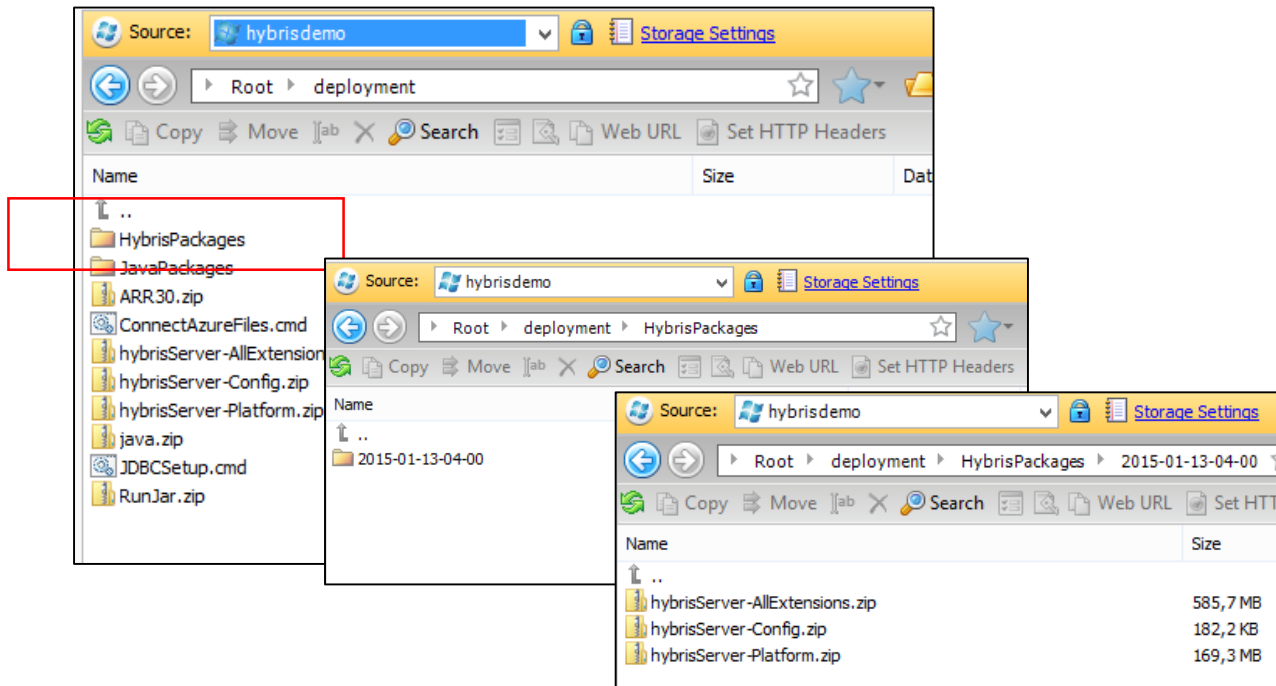
In the admin page you can specify the name of the package subdirectory you want to be deployed on the next server reboot or when a new server starts up (e.g. 2015-01-13-04-00). When you click the "Submit changes" button the three .zip files within the subdirectory of the "HybrisPackages" directory will be copied to "deployment/*.zip" which will be downloaded at server startup.

7. Customization

This chapter gives some possibilities to customize your hybrisOnAzure for all deployment. The features used to setup the hybris platform enables you to customize many things. For example the execution of a custom java program at server startup and a custom maintenance page are presented.

7.1 Run custom Java application on startup

The possibility to download files from the blob storage and execute scripts at server startup enables you run a custom Java application at server startup. A sample is included with the hybrisOnAzure package.

Create a Java application

Using eclipse you can create and export a runnable JAR file. The sample consists of a Java program that reads all environment variables available to the java process.

```java
Sample Java application:

package source;
import java.util.Map;

public class Main {

  public static void main(String[] args) {
    // Sysout will be visible in tangible Azure Trace Console
    System.out.println("Enered java code..");

    // Write environment variables to a local file
    File file = new File("C:\\javaout.txt");
    if (file.exists()) {
      file.delete();
    }
    file.createNewFile();

    Map<String, String> env = System.getenv();
    for (String envName : env.keySet()) {
      fw.write(envName);
      fw.write(" = ");
      fw.write(env.get(envName));
      fw.write(System.getProperty("line.separator"));
    }

    fw.flush();
    fw.close();

    System.out.println("Exiting Java code");
}
```

Access Azure Information

Inside your Java application you can use the Windows Azure SDK for Java (http://azure.microsoft.com/de-de/develop/java/) to interact with Windows Azure resources. But you cannot use the RoleEnvironment interfaces. Those interfaces are only available to the first assembly initialized by the Windows Azure platform.

To grant you access to those information you can provide necessary values using the batch script that is used to startup your Java application in the following section.

Create a Batch script

The hybrisOnAzure For All package allows you to run batch scripts at startup. Such a batch script can be used to run your JAR file using the JRE downloaded at server startup. Before executing a batch script the file is searched by the hybris on Azure package for place holders that can be replaced with information from the RoleEnvironment or the CloudService configuration.

This is the sample script delivered with the hybrisOnAzure package to run the sample Java application.

Sample Java application:

```
@REM -------------------------------------------------------------------------
@REM This command file executes a given JAR file
@REM -------------------------------------------------------------------------

@REM If you want the JAR to be executed only once, enable the next line
@REM if exist %SystemDrive%\jar-executed.txt goto skip

@REM Read a value from the .cscfg file and store it in an environment variable
@REM that can be read from within Java
set
CSCFG_HybrisOnAzure.JavaHomeDirectory="<#%HybrisOnAzure.JavaHomeDirectory%#>"

@REM Read a public endpoint from the RoleEnvironment
set
RE_TomcatIP=<#%RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["TomcatHtt
p"].IPEndpoint.Address%#>
set
RE_TomcatPort=<#%RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["TomcatH
ttp"].IPEndpoint.Port%#>
set
RE_TomcatEndpoint=<#%RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["Tom
catHttp"].IPEndpoint%#>

@REM start the java program
call "<#%HybrisOnAzure.JavaHomeDirectory%#>"\bin\java.exe -jar
%~dp0MyJavaProgram.jar

time /t >> %SystemDrive%\jar-executed.txt
:skip
```

Before calling the Java application this script sets environment variables with values from the CloudService configuration and the RoleEnvironment:

`<#%HybrisOnAzure.JavaHomeDirectory%#>` will be read from the .cscfg file

`<#%RoleEnvironment.CurrentRoleInstance.InstanceEndpoints["TomcatHttp"].IPEndpoint %#>` will be read from the RoleEnvironment.

Using the C# reference will help you get other information from the RoleEnvironment when needed:
https://msdn.microsoft.com/en-us/library/microsoft.windowsazure.serviceruntime.roleenvironment_members.aspx

Run at startup

The hybrisOnAzure For All package contains a RunJar.zip file. This file contains the sample Java application and the Sample batch file to run the application. In order to have it available on server startup you need to perform the following steps:

1. Upload the .zip file to the "deployment" container within the Blob Storage (e.g. RunJar.zip).
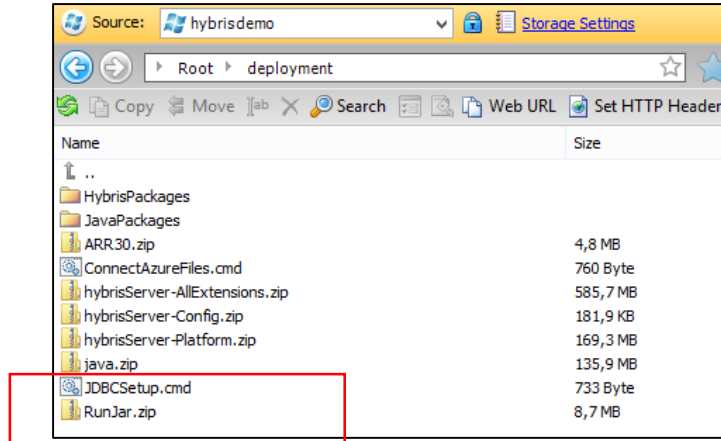
2. In the Cloud Configuration file (.cscfg) locate the `<Role>` elements representing the Server type on which you want the Java application to run.

3. Within the `<Role>` element locate the
   `<Setting name="tangible.Azure.Startup.Download.Downloads" />` element.
   This element contains a semicolon separated list of download tasks to be performed on server startup. A download task consists of three information parts: a file to download from blob storage, a directory to download the file to and a boolean parameter with value "true" if the downloaded file needs to be unzipped.

4. Extend the value by adding your download task
   `"deployment/RunJar.zip|C:\Applications|true;"`

After finishing all downloads the hybrisOnAzure platform executes the command scripts and batch files listed in the
`<Setting name="tangible.Azure.Startup.StartupCommands.Commands" />`. Its value consists of a semicolon separated list of execution tasks. A task consists of three information parts: a file to execute, a number representing a timeout in milliseconds and a boolean parameter with value "true" if the scripts needs to be run in-place.

5. Within the `<Role>` element locate the `<Setting name="tangible.Azure.Startup.StartupCommands.Commands" />` element.

6. Extend the value by adding your java task
   `"C:\Applications\RunJar.cmd|30000|true;"`

Please note that the Java application you run at startup must terminate within the given timeout. Otherwise the process will be killed. You can try to choose a larger timeout.

7.2 Custom maintenance page

When setting your hybris deployment into maintenance mode, you might want to have a custom look and feel of the maintenance page. In order to have custom content for your maintenance page you need to create the custom design maintenancepage.html and an adapted web.config file. Both files need to be zipped with all images or design files needed into a .zip file, uploaded to the blob storage and downloaded into the Request Routing servers' web root directory.

1. Create a file named "maintenance.html" and fill it with your custom html content.

Sample custom maintenance page:

```
<html>
  <head>
    <title>Maintenance mode</title>
  </head>

  <body>
    <h1>This is a custom maintenance page.</h1>
        <p>Please be patient until we have everything running smoothly
again.</p>

        <image src="construction.jpeg" width="200" height="200" />
  </body>
</html>
```

2. Adapt the web.config file to include a rule for each file of your custom page layout

Sample adapted web.config:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please
visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.diagnostics>
    <trace>
      <listeners>
        <add
type="Microsoft.WindowsAzure.Diagnostics.DiagnosticMonitorTraceListener,
Microsoft.WindowsAzure.Diagnostics, Version=2.4.0.0, Culture=neutral,
PublicKeyToken=31bf3856ad364e35" name="AzureDiagnostics">
          <filter type="" />
        </add>
      </listeners>
    </trace>
  </system.diagnostics>
  <system.web>
    <compilation debug="true" targetFramework="4.0" />
    <httpRuntime />
  </system.web>
  <system.webServer>
```

```
<!-- Rules for Application Request routing -->
<rewrite>
  <rules>
      <!-- Deliver Maintenance Page directly -->
      <rule name="MaintenancePage" stopProcessing="true" enabled="true">
        <match url="^maintenance.html$" />
        <action type="None" />
      </rule>
     <rule name="ConstructionImage" stopProcessing="true" enabled="true">
        <match url="^construction.jpeg$" />
        <action type="None" />
      </rule>
      <!-- create a rule for each additional custom maintenance page file -->
      <!--<rule name="MaintenancePage" stopProcessing="true" enabled="true">
        <match url="^myfile.ext$" />
        <action type="None" />
      </rule>-->

      <!-- Rule to redirect all traffic to the maintenance page -->
      <rule name="IsInMaintenance" stopProcessing="true" enabled="false">
        <match url="(.*)" />
        <action type="Rewrite" url="/maintenance.html"
redirectType="Temporary" />
      </rule>

      <!-- Rules to deny access to cockpit urls -->
      <rule name="Block_AdminCockpit" stopProcessing="true" enabled="true">
        <match url="^admincockpit($|/.*)" />
        <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
      </rule>
      <rule name="Block_CmsCockpit" stopProcessing="true" enabled="true">
        <match url="^cmscockpit($|/.*)" />
        <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
      </rule>
      <rule name="Block_CmsCockpitRegular" stopProcessing="true"
enabled="true">
        <match url="^cmscockpitRegular($|/.*)" />
        <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
      </rule>
      <rule name="Block_AcceleratorServices" stopProcessing="true"
enabled="true">
        <match url="^acceleratorservices($|/.*)" />
        <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
      </rule>
      <rule name="Block_CsCockpit" stopProcessing="true" enabled="true">
        <match url="^cscockpit($|/.*)" />
        <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
```

```
            <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_hac" stopProcessing="true" enabled="true">
          <match url="^hac($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_hmc" stopProcessing="true" enabled="true">
          <match url="^hmc($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_instore" stopProcessing="true" enabled="true">
          <match url="^instore($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_mcc" stopProcessing="true" enabled="true">
          <match url="^mcc($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_ws410" stopProcessing="true" enabled="true">
          <match url="^ws410($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_ProductCockpit" stopProcessing="true" enabled="true">
          <match url="^productcockpit($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_ReportCockpit" stopProcessing="true" enabled="true">
          <match url="^reportcockpit($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_SolrFacetSearch" stopProcessing="true"
enabled="true">
          <match url="^solrfacetsearch($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
```

```xml
supplied." />
        </rule>
        <rule name="Block_VirtualJDBC" stopProcessing="true" enabled="true">
          <match url="^virtualjdbc($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>
        <rule name="Block_Test" stopProcessing="true" enabled="true">
          <match url="^test($|/.*)" />
          <action type="CustomResponse" statusCode="403"
statusReason="Forbidden: Access is denied." statusDescription="You do not have
permission to view this directory or page using the credentials that you
supplied." />
        </rule>

        <!-- Rule to redirect all traffic to the webfarm -->
        <rule name="WebFarm" stopProcessing="true">
          <match url="(.*)" />
          <action type="Rewrite" url="http://AzureWebFarm/{R:0}" />
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <dependentAssembly>
        <assemblyIdentity name="Newtonsoft.Json"
publicKeyToken="30ad4fe6b2a6aeed" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-6.0.0.0" newVersion="6.0.0.0" />
      </dependentAssembly>
      <dependentAssembly>
        <assemblyIdentity name="System.Net.Http"
publicKeyToken="b03f5f7f11d50a3a" culture="neutral" />
        <bindingRedirect oldVersion="0.0.0.0-2.2.20.0" newVersion="2.2.20.0" />
      </dependentAssembly>
    </assemblyBinding>
  </runtime>
</configuration>
```
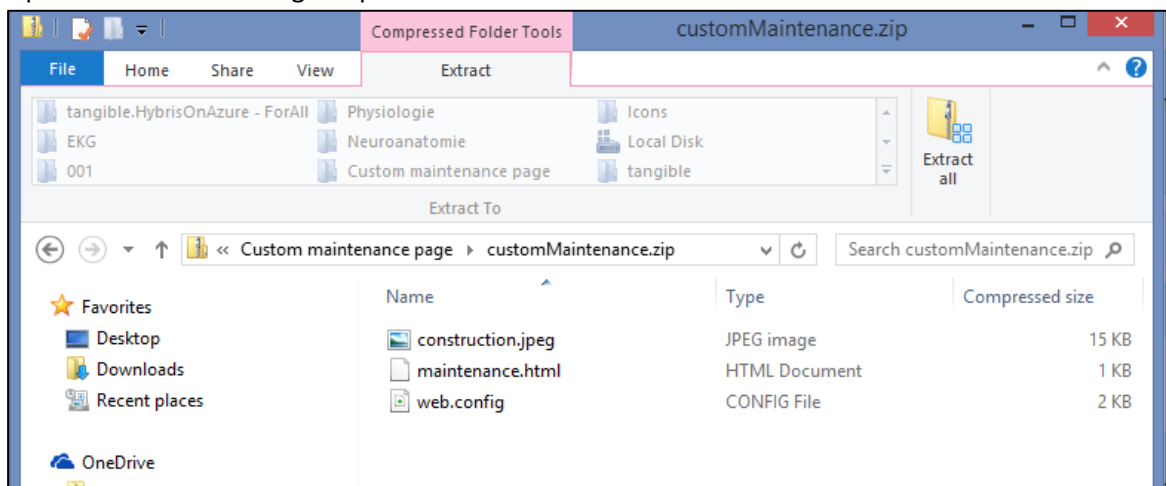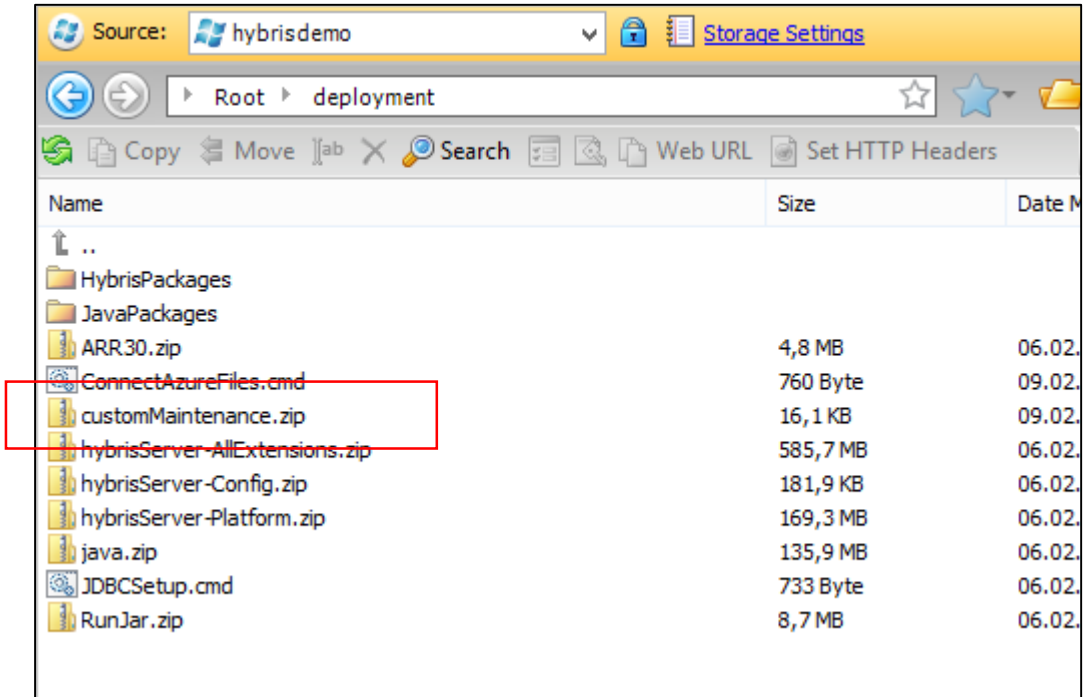
3. Zip all the files into a single .zip file

4. Upload the zip file into the "deployment" container

5. Edit the Cloud Service configuration file to have the Application Request Routing servers download the .zip file at server startup

```
<ServiceConfiguration ...>
  <Role name="tangible.HybrisOnAzure.ArrRole">
    <Instances count="1" />
    <ConfigurationSettings>
      ...
      <Setting name="tangible.Azure.Startup.Download.Downloads"
value="deployment/ARR30.zip|C:\Setups\ApplicationRequestRouting|true;deployment/cus
tomMaintenance.zip|E:\sitesroot\0|true" />
      ...
    </ConfigurationSettings>
    ...
  </Role>
</ServiceConfiguration>
```

7.3 Solr Standalone Configuration

In Chapter 4.1 Upload additional files the configuration of a solr standalone installation was prepared by copying the solrconfig.xml and solserver.bat in the "deployment" container of the Windows Azure Storage Account. This chapter covers the contents of those files and how they can be adapted for special customer needs.

Solconfig.xml

This file is a standard configuration file for solr and may be customized. In order to place proper master/slave information for solr clustering special placeholders were put into the replication section:

```xml
...
<requestHandler name="/replication" class="solr.ReplicationHandler">
    <!--
        To enable simple master/slave replication, uncomment one of the
        sections below, depending on whether this solr instance should be
        the "master" or a "slave".  If this instance is a "slave" you will
        also need to fill in the masterUrl to point to a real machine.
    -->
    <lst name="master">
      <str name="replicateAfter">commit</str>
      <str name="replicateAfter">startup</str>
      <str name="confFiles">schema.xml,stopwords.txt</str>
      <str name="enable"><#%HybrisOnAzure.IndexMaster%#></str>
    </lst>
    <lst name="slave">
        <str
name="masterUrl">http://<#%RoleEnvironment.Roles["tangible.HybrisOnAzure.BackOffice
WorkerRole"].Instances[0].InstanceEndpoints["Solr"].IPEndpoint.Address%#>:8983/solr
</str>
        <str name="pollInterval">00:00:60</str>
        <str name="enable"><#%HybrisOnAzure.IndexSlave%#></str>
    </lst>
  </requestHandler>
...
```

Please refer to chapter 7.1 Run custom Java application on startup to learn more about these placeholders. These placeholders should not be changed.

The solrcongif.xml file is downloaded at server startup and downloaded into the solr config directory.

Solrserver.bat

This file is used to start the Solr server on each machine. It contains different parameters for the solr server depending if it is used as the index master or an index slave. You may customize this file in order to adapt solr server behavior.

```bat
@echo off

:: This script start, stops and restarts standalone solr for hybris on azure
:: (c) tanigble engineering GmbH, 2015

:: the value of this variable will be replaced on startup by the hybris on azure
platform
set ISMASTER="<#%HybrisOnAzure.IndexMaster%#>"
call :UpCase ISMASTER

:: determine if this server needs master/slave configuration
if %ISMASTER%=="TRUE" (
   :: set options for solr master
   set JAVA_OPTIONS=-server -Xms2048m -Xmx2048m -jar -
Dcom.sun.management.jmxremote.port=9883 -Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote -
DSTOP.PORT={solrstopport} -DSTOP.KEY={solrstopkey} -Dsolr.solr.home=. -
Denable.master=true -Djetty.port={solrport} -Dsolr.data.dir={solrdatadir} start.jar
   )
if %ISMASTER%=="FALSE" (
   :: set options for solr slave
   set JAVA_OPTIONS=-server -Xms2048m -Xmx2048m -jar -
Dcom.sun.management.jmxremote.port=9883 -Dcom.sun.management.jmxremote.ssl=false -
Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote -
DSTOP.PORT={solrstopport} -DSTOP.KEY={solrstopkey} -Dsolr.solr.home=. -
Denable.slave=true -Djetty.port={solrport} -Dmaster.host={solrmasterip}  -
Dsolr.data.dir={solrdatadir} start.jar
   )

set LOG_FILE=%SOLR_DIR%\logs\solr.log


if ""%1"" == ""restart"" goto doRestart
if ""%1"" == ""start"" goto doStart
if ""%1"" == ""stop"" goto doStop
if ""%1"" == """" goto doUsage


goto EOF

:doStart

echo "Starting Solr"
cd %SOLR_DIR%
cmd /c java %JAVA_OPTIONS%

goto EOF

:doStop

echo "Stoping Solr"
cd %SOLR_DIR%
cmd /c java %JAVA_OPTIONS% --stop

goto EOF
```

```
    :doRestart
    echo "Starting Solr"
    cd %SOLR_DIR%
    cmd /c java %JAVA_OPTIONS% --stop
    timeout 2
    cmd /c java %JAVA_OPTIONS%

    goto EOF

    :doUsage
    @echo off
    echo Usage: "%0 {start|stop|restart}"

    goto EOF


    :UpCase
    :: Subroutine to convert a variable VALUE to all UPPER CASE.
    :: The argument for this subroutine is the variable NAME.
    FOR %%i IN ("a=A" "b=B" "c=C" "d=D" "e=E" "f=F" "g=G" "h=H" "i=I" "j=J" "k=K" "l=L"
    "m=M" "n=N" "o=O" "p=P" "q=Q" "r=R" "s=S" "t=T" "u=U" "v=V" "w=W" "x=X" "y=Y"
    "z=Z") DO CALL SET "%1=%%1:%%~i%%"
    GOTO:EOF

    :EOF
```

7.4 Changing VM sizes

There might be scenarios in which the default sizes of the VMs created in Windows Azure do not fit your needs. The VM sizes that are selected by default are chosen based on customer experience. If you still need them to be changed, follow these steps:

1. In Visual Studio locate the "ServiceDefinition.csdef" file inside the solution explorer
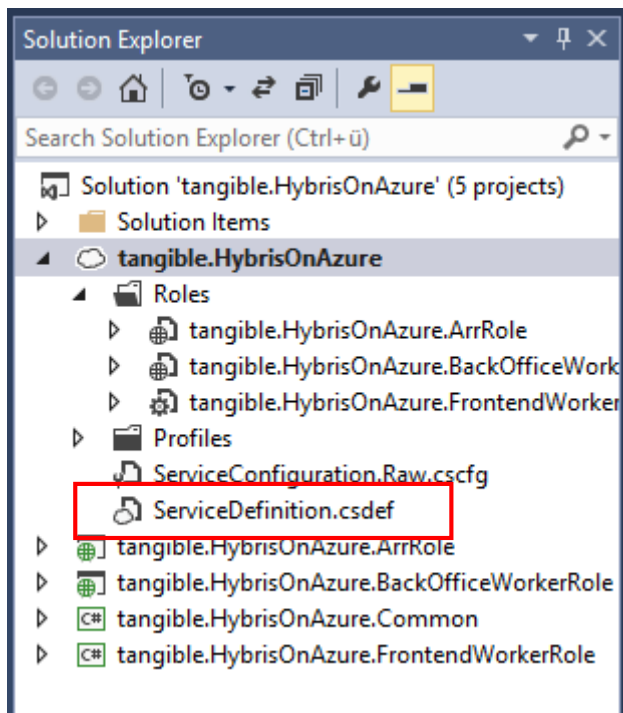
2. Inside this XML file there is a section for each server type that is deployed in Windows Azure: the ARR tier, the Frontend workers and the BackOffice worker. Change the value of the "vmsize" attribute to your desired value.

A list of valid VM sizes can be found here:
https://msdn.microsoft.com/en-us/library/azure/dn197896.aspx

8. Troubleshooting

These chapter covers learnings from attempts made to port an existing customer hybris setup into the hybrisOnAzure – For All platform:

SQL Server Drivers
Ensure that the JAR file containing the Microsoft Java SQL Server database drivers exists in the platform/lib directory: e.g. "sqljdbc_4.0.2206.100.jar".
The Microsoft JDBC driver can be downloaded here: http://www.microsoft.com/de-de/download/details.aspx?id=11774

At hybris build time, maven tries to find database drivers
This may result in a missing .lastupdate file and can be resolved by adding the setting "maven.update.dbdrivers=false".

At hybris build time, error "corrupt config folder" is thrown
This may be caused by missing or wrong "license" and "tomcat" folders.

Document History

February 6, 2015   Document created

February 16, 2015   Version 1.0

March 3, 2015   Version 1.1
(includes standalone solr search, azure files preview and troubleshooting)

March 25, 2015   Version 1.2
(includes Application Request Routing Zip generation)

May 26, 2015   Version 1.3
(includes VM size changing)

June 24, 2015   Version 1.4
(includes SQL Azure documentation, separate DebugView download, separate TanukiWrapper)

August 22, 2015   Version 1.5
(includes feedback from customers)