

[Company name]

# Sitecore on Azure PaaS services Technical Workshop

Hands-on Lab – Step by step guide

[Author name]

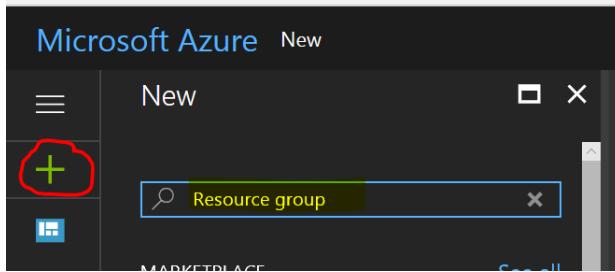
10-2-2017

## Contents

Exercise 2 - ARM Templates (85 min) .....	3
Task 2: Deploy the default Sitecore XM ARM Templates (45 min).....	6
Task 3: Configure an Azure KeyVault to store secrets (10 min).....	9
Task 4: Deploy a custom Sitecore XM ARM Templates (15 min).....	12
Exercise 3 – Azure SQL Database (20 min).....	20
Objectives .....	20
Task 1: Scale up/down your databases (5 min) .....	20
Task 2: Local connection with Visual Studio (7 min).....	21
Task 3: Add Alert Rules (3 min) .....	25
Task 4: Configure a Geo-replication (5 min) .....	27
Takeaways.....	30
Exercise 4 - Azure Web App (65 min).....	32
Objectives .....	32
Task 1: Verify the Sitecore solution previously provisioned (5 min) .....	32
Task 2: Add IP restriction on the CM instance (5 min).....	36
Task 3: Setup Azure Web App Backup (10 min).....	37
Task 4: Create a Staging Slot for the CD instance (15 min).....	40
Task 5: Testing in Production (5 min).....	46
Task 6: Slot Swap (5 min) .....	48
Task 7: Sitecore Publishing Services on Azure Web App (20 min).....	49
Takeaways.....	58
Exercise 5 - Azure Search and Application Insights (35 min) .....	59
Objectives .....	59
Task 1: Azure Search (15 min).....	59
Task 2: Application Insights (20 min) .....	65
Takeaways.....	73
Exercise 6 – QnA Maker and Bot Service (20 min).....	75
Objectives .....	75
Task 1: QnA Maker (10 min) .....	75
Task 2: Bot Service (10 Min).....	78
Takeaways.....	84

Exercise 7 – Visual Studio Team Services - VSTS.....	84
Objectives .....	84
Task 1: Create a new VSTS project (5 min) .....	84
Task 2: Code – Version Controlling with Git (15 min) .....	85
Task 3: Build – Continuous Integration.....	93
Task 4: Release – Continuous Deployment.....	93
Takeaways.....	93

## Exercise 2 - ARM Templates (85 min)



- Once the Resource Group created, go to its “Overview” blade and click on the “Add” button.

A screenshot of the Azure Resource Group 'Overview' blade for a group named 'sitecorelab'. The left sidebar has links for 'Overview' (which is highlighted in blue), 'Activity log', 'Access control (IAM)', and 'Tags'. The main area has a header with '+ Add', 'Columns', 'Delete', and 'Refresh' buttons. Below that is a section titled 'Essentials' with 'Subscription name (change)' and 'Deployments' (with 'No deployments'). At the bottom is a 'Filter by name...' input field.

- Create a new Azure Storage account looking for “blob” and fill out all the fields like illustrated below with a unique name.

A screenshot of the Microsoft Azure Marketplace search results. The search bar at the top contains the word 'blob'. Below the search bar, a list of results is shown, with the first item highlighted in yellow: 'Storage account - blob, file, table, queue'. The left sidebar shows icons for 'Create', 'Marketplace', and 'Search'.

Storage account - blob, file, table, queue

PUBLISHER Microsoft

USEFUL LINKS Service overview Documentation Pricing

Create storage account

Name: mabenotsitecorelab.core.windows.net

Deployment model: Resource manager

Account kind: General purpose

Performance: Standard

Replication: Read-access geo-redundant storage (RA...)

Storage service encryption: Disabled

Subscription: Microsoft Azure Internal Consumption

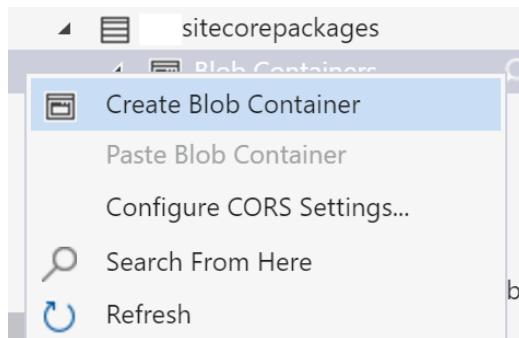
Resource group: sitecorelab

Location: East US

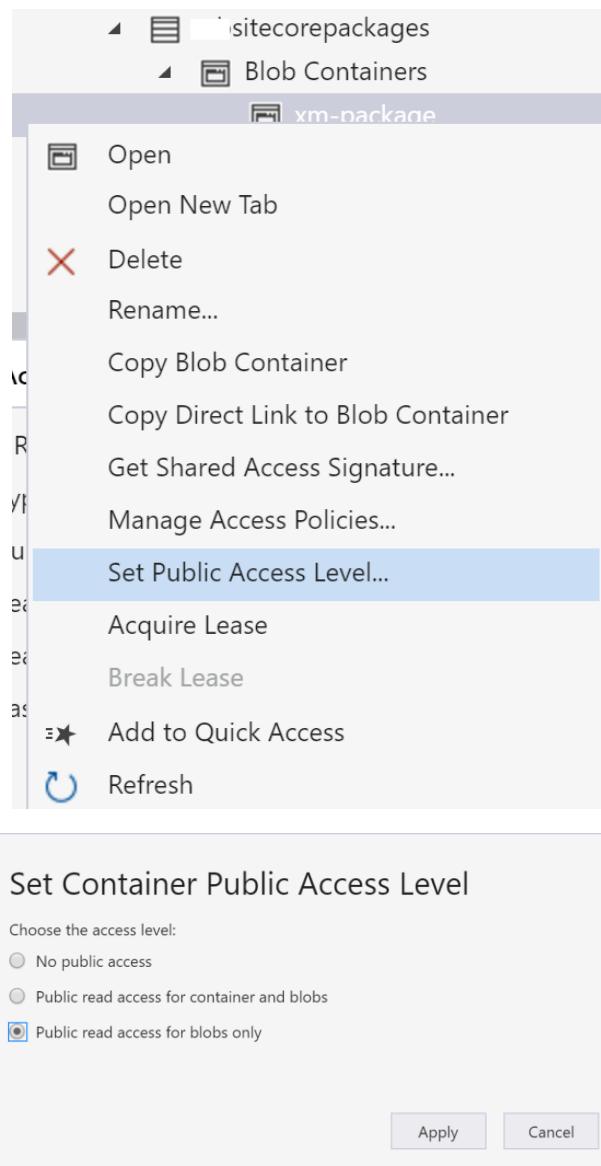
Pin to dashboard

**Create** Automation options

3. Open Microsoft Azure Storage Explorer locally on your VM, and sign-in with your credentials.
4. Locate your Azure Storage account previously created and right click on it to “Create Blob Container” named “xm-packages”.



5. “Set Public Access Level” to this Blob container “xm-packages” by right-clicking on it and then select “Public read access for blobs only”.



**Note:** To make easier the manipulation of the files stored on Blob Storage we are making them publicly available, but in real life you will set more security around their access with for example the "Get Shared Access Signature" feature.

6. You could now upload the 2 Sitecore packages files from this local directory: "C:\SitecoreLab\Packages\XM\Sitecore 8.2 rev. 170407 (WDP XM1 packages)\Sitecore 8.2 rev. 170407 (WDP XM1 packages)" to this Azure Blob Container "xm-packages" by drag-and-dropping the file or clicking on the "Upload" button directly within the Microsoft Azure Storage Explorer (be sure to open the "xm-packages" container).

The screenshot shows the Azure Storage Explorer interface. At the top, there's a toolbar with various file operations like Upload, Download, Open, New Folder, Copy URL, Select All, Copy, Paste, Rename, Delete, Make Snapshot, and Manage Snapshot. Below the toolbar is a breadcrumb navigation bar showing the path: xm-packages. A search bar labeled "Search by prefix (case-sensitive)" is also present. Underneath, there's a table with columns: Name, Last Modified, Blob Type, Content Type, Size, Lease State, and Disk Name. Two items are listed:

Name	Last Modified	Blob Type	Content Type	Size	Lease State	Disk Name
Sitecore 8.2 rev. 170407_cd.scwdp.zip	Wed, 12 Apr 2017 01:32:28 GMT	Block Blob	application/zip	262.0 MB		
Sitecore 8.2 rev. 170407_cm.scwdp.zip	Wed, 12 Apr 2017 01:32:28 GMT	Block Blob	application/zip	262.0 MB		

At the bottom left, it says "Showing 1 to 2 of 2 cached items". Below the table, there's a section for "Activities" with links to "Clear completed" and "Clear successful". A log entry is shown:

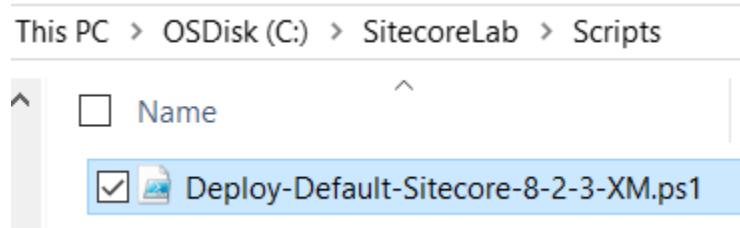
- Uploaded 2 blobs from 'C:\SitecoreLab\Packages\XM\Sitecore 8.2 rev. 170407 (WDP XM1 packages)\Sitecore 8.2 rev. 170407 (WDP XM1 packages)\' to 'mbsitecorepackages/xm-packages' 2 completed

Here we are, we now have a clear setup to run the Sitecore ARM Templates with the coming tasks below.

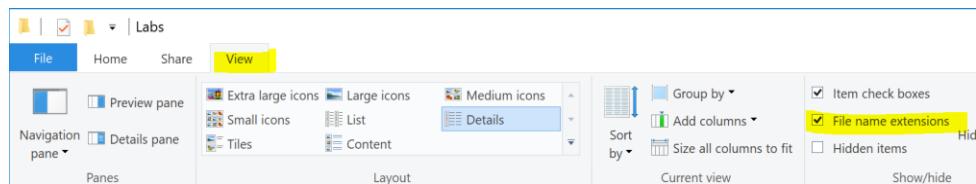
## Task 2: Deploy the default Sitecore XM ARM Templates (45 min)

In this section, the attendee will deploy the default Sitecore XM ARM Templates via PowerShell by using the files previously prepared.

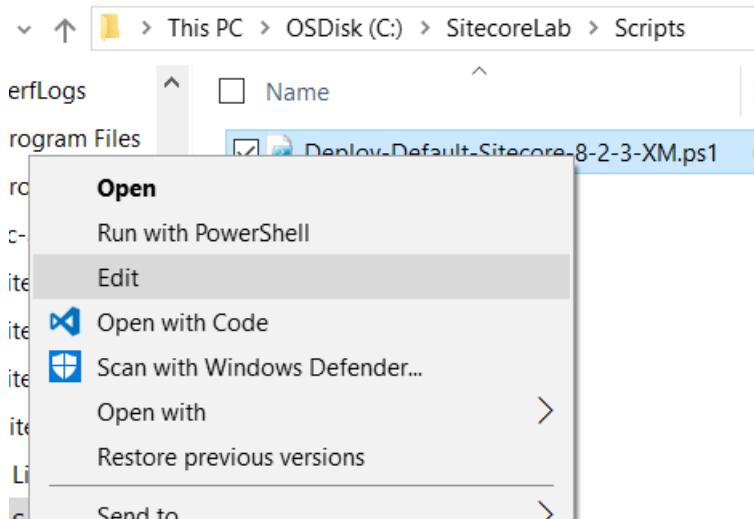
1. Create a new subfolder “**C:\SitecoreLab\Scripts**”.
2. Go to <https://gist.githubusercontent.com/mathieu-benoit/4d5479b024aa2154bb5e91b19bb1e255/raw/554e779c0f9bce48da00d282fdc2ecd0fe85db2c/Deploy-Default-Sitecore-8-2-3-XM.ps1>
3. Copy/paste the code as “**Deploy-Default-Sitecore-8-2-3-XM.ps1**” file into “**C:\SitecoreLab\Scripts**”.



4. To change the extension of your file, you could activate the “**File name extensions**” option from the “**View**” tab of your Windows Explorer.



5. Right click on the “Deploy-Default-Sitecore-8-2-3-XM.ps1” file and select “Edit”.



6. The PowerShell ISE tool is now opened.

```

# PowerShell script to deploy Sitecore 8.2.3 XM
# Located here: https://raw.githubusercontent.com/Sitecore/Sitecore-Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm

Param(
    [string] [Parameter(Mandatory=$true)] $SubscriptionId,
    [string] [Parameter(Mandatory=$true)] $ResourceGroupName,
    [string] $ResourceGroupLocation = "East US",
    [string] $TemplateFile = "https://raw.githubusercontent.com/Sitecore/Sitecore-Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm/azuredploy",
    [string] [Parameter(Mandatory=$true)] $LicenseFile,
    [string] $CDMsDeployPackageUrl = 'TO_REPLACE',
    [string] $CMsDeployPackageUrl = 'TO_REPLACE',
    [string] [Parameter(Mandatory=$true)] $SqlServerLogin,
    [string] $SitecoreSku = 'XM2',
    [securestring] [Parameter(Mandatory=$true)] $SqlServerPassword,
    [securestring] [Parameter(Mandatory=$true)] $SitecoreAdminPassword
)
$licenseFileContent = Get-Content -Raw -Encoding UTF8 -Path $LicenseFile | Out-String;
$parameters = New-Object -TypeName Hashtable;

```

7. On the top window of the PowerShell ISE tool (PowerShell Editor with the white background), replace the ‘TO\_REPLACE’ values for the 2 package url parameters accordingly. You could get the URL value of each Blob file by clicking on the “Copy URL” button via the Microsoft Azure Storage Explorer:

The screenshot shows the Azure Storage Explorer interface. At the top, there's a toolbar with various icons for file operations like Upload, Download, Open, New Folder, Copy, Rename, Delete, and Make Snapshot. Below the toolbar is a search bar containing "xm-packages". The main area displays a list of files in the "xm-packages" container. The files listed are:

Name	Last Modified	Blob Type	Content Type	Size
Sitecore 8.2 rev. 170407_cd.scwdp.zip	Wed, 12 Apr 2017 01:32:28 GMT	Block Blob	application/zip	262.0 MB
Sitecore 8.2 rev. 170407_cm.scwdp.zip	Wed, 12 Apr 2017 01:32:28 GMT	Block Blob	application/zip	262.0 MB

For the URLs, make sure to have the CD and CM values at the right place, like illustrated on the image below.

```

Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Deploy-Default-Sitecore-8-2-3-XM.ps1
1 # PowerShell script to deploy Sitecore 8.2.3 XM
2 # Located here: https://raw.githubusercontent.com/Sitecore/Sitecore-Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm
3
4 Param(
5     [string] [Parameter(Mandatory=$true)] $SubscriptionId,
6     [string] [Parameter(Mandatory=$true)] $ResourceGroupName,
7     [string] $TemplateFile = "https://raw.githubusercontent.com/Sitecore/Sitecore-Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm/azuredeploy.json",
8     [string] [Parameter(Mandatory=$true)] $LicenseFile,
9     [string] $cdmsDeployPackageUrl = "https://sitecoresmsdeploypackages.blob.core.windows.net/xm-packages/Sitecore%208.2%20rev.%20170407_cd.scwdp.zip",
10    [string] $cmmsDeployPackageUrl = "https://sitecoresmsdeploypackages.blob.core.windows.net/xm-packages/Sitecore%208.2%20rev.%20170407_cm.scwdp.zip",
11    [string] [Parameter(Mandatory=$true)] $sqlServerLogin,
12    [string] $sitecoreSku = 'XM2',
13    [securestring] [Parameter(Mandatory=$true)] $sqlServerPassword,
14    [securestring] [Parameter(Mandatory=$true)] $sitecoreAdminPassword
15 )
16
17 $licenseFileContent = Get-Content -Raw -Encoding UTF8 -Path $LicenseFile | Out-String;
18
19 $parameters = New-Object -TypeName Hashtable;
20 $parameters.Add("cdmsDeployPackageUrl", $cdmsDeployPackageUrl);
21 $parameters.Add("cmmsDeployPackageUrl", $cmmsDeployPackageUrl);
22 $parameters.Add("sqlServerLogin", $sqlServerLogin);
23 $parameters.Add("sqlServerPassword", $sqlServerPassword);
24 $parameters.Add("sitecoreAdminPassword", $sitecoreAdminPassword);
25 $parameters.Add("sitecoreSKU", $sitecoreSku);
26 $parameters.Add("licenseXml", $licenseFileContent);
27
28
PS C:\sitecoreLab\scripts>
```

8. Save your changes by clicking on the **Save** icon:
9. On the bottom window of the PowerShell ISE tool (PowerShell console with blue background), copy/paste the following snippet and execute it with the "**Enter**" key:

```

.\ Deploy-Default-Sitecore-8-2-3-XM.ps1 -LicenseFile
"..\License\license.xml" -SqlServerLogin "demouser" -SqlServerPassword
(ConvertTo-SecureString -String "demo@pass12345" -AsPlainText -Force)
-SitecoreAdminPassword (ConvertTo-SecureString -String
"demo@pass12345" -AsPlainText -Force)

```

10. You will be prompted to enter complementary parameters value:

- **SubscriptionId**: enter the id of the Azure subscription in which you would like to deploy your Sitecore solution for this lab.
- **ResourceGroupName**: enter the name of the Resource Group you would like to create to regroup all the resources needed to manage your Sitecore solution for this lab. **We highly encourage you to choose a unique name, not too long and just alphanumerical (as tips, for the prefix, use your last name for example).**
- You will be asked to sign-in as well.
- *Then all the resources defined within the associated ARM Templates will be deployed. It will take about 30 min.*

DEPLOYMENT NAME	STATUS	TIMESTAMP	DURATION
mabenoit-test	Succeeded	6/7/2017 11:48:36 AM	29 minutes 21 seconds
mabenoit-test-empty	Succeeded	6/7/2017 11:48:31 AM	2 seconds
mabenoit-test-application	Succeeded	6/7/2017 11:48:12 AM	17 minutes 44 seconds
mabenoit-test-infrastructure	Succeeded	6/7/2017 11:30:21 AM	11 minutes 3 seconds

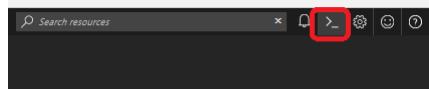
You could start the next tasks below while this deployment is running.

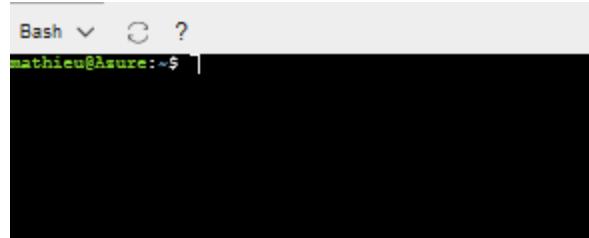
### Task 3: Configure an Azure KeyVault to store secrets (10 min)

In this section, the attendee will create an Azure KeyVault and will store some secrets into it by using the new Cloud Shell within the Azure portal, then we will update the ARM Template accordingly.

**Note:** This section could be accomplished by the Azure portal UI as well but here we would like to leverage the [Azure Cloud Shell](#). This new tool/feature became in Public Preview since May 10<sup>th</sup> 2017, [see the announce made during the Microsoft //build 2017 conference](#).

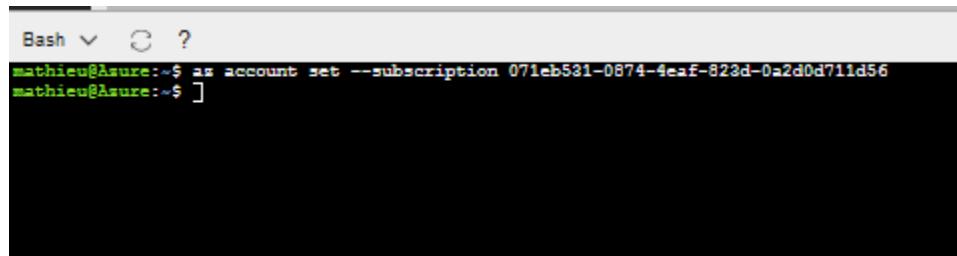
1. Go to the Azure portal at <https://portal.azure.com>.
2. On the top left corner, open the Cloud Shell by clicking on the icon highlighted below:





3. Select the right Azure subscription in which you have your “**sitecorelab**” Resource Group by running this CLI command:

```
az account set --subscription your_azure_subscription_id
```



4. Create an **Azure KeyVault** account within the “**sitecorelab**” Resource Group by running this CLI command:

```
az keyvault create --name 'sitecorelab' --resource-group 'sitecorelab' --location 'East US' --enabled-for-template-deployment true
```

```
mathieu@Azure:~$ az keyvault create --name 'sitecorelab' --resource-group 'sitecorelab' --location 'East US' --enabled-for-template-deployment true
{
  "id": "/subscriptions/071eb531-0874-4eaf-823d-0a2d0d711d56/resourceGroups/sitecorelab/providers/Microsoft.KeyVault/vaults/sitecorelab",
  "location": "eastus",
  "name": "sitecorelab",
  "properties": {
    "accessPolicies": [
      {
        "applicationId": null,
        "objectId": "f1e00000-0000-0000-0000-0c4255820a0",
        "permissions": {
          "certificates": [
            "all"
          ],
          "keys": [
            "get",
            "create",
            "delete",
            "list",
            "update",
            "import",
            "backup",
            "restore"
          ],
          "secrets": [
            "all"
          ]
        },
        "tenantId": "70200000-0000-0000-0000-d7cd011db47"
      }
    ],
    "enabledForDeployment": false,
    "enabledForDiskEncryption": null,
    "enabledForTemplateDeployment": true,
    "sku": {
      "name": "standard"
    },
    "tenantId": "70200000-0000-0000-0000-d7cd011db47",
    "vaultUri": "https://sitecorelab.vault.azure.net"
  },
  "resourceGroup": "sitecorelab",
  "tags": {},
  "type": "Microsoft.KeyVault/vaults"
}
mathieu@Azure:~$
```

5. Store 2 secrets into this KeyVault just created: 'SqlServerLogin' and 'SqlServerPassword'.

```
az keyvault secret set --vault-name 'sitecorelab' --name 'SqlServerLogin' --value 'demouser'
```

and

```
az keyvault secret set --vault-name 'sitecorelab' --name 'SqlServerPassword' --value 'demo@pass12345'
```

```
Bash | PowerShell (coming soon) Feedback
mathieu@Azure:~$ az keyvault secret set --vault-name 'sitecorelabkeyvault' --name 'SqlServerPassword' --value 'demo@pass12345'
{
  "attributes": {
    "created": "2017-04-11T02:37:35+00:00",
    "enabled": true,
    "expires": null,
    "notBefore": null,
    "updated": "2017-04-11T02:37:35+00:00"
  },
  "contentType": null,
  "id": "https://sitecorelabkeyvault.vault.azure.net/secrets/SqlServerPassword/0cb1c3f0-0000-0000-0000-000000000000",
  "kid": null,
  "managed": null,
  "tags": {
    "file-encoding": "utf-8"
  },
  "value": "demo@pass12345"
}
mathieu@Azure:~$
```

6. So, the KeyVault “**siterecorelab**” has now 2 secrets we will use in the following task below, you could check that by opening the associated “**Secrets**” blade:

The screenshot shows the 'sitecorelab - Secrets' blade in the Azure portal. On the left, there's a navigation menu with options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Keys, Secrets (which is selected), Access policies, Advanced access policies, Properties, and Locks. The main area displays a table with two rows:

NAME	TYPE	STATUS
SqlServerLogin		✓ Enabled
SqlServerPassword		✓ Enabled

**Note:** We will use these 2 secrets with the next task with a custom ARM Templates.

Furthermore, we just stored 2 secrets for this current lab, but you should store more: other SQL databases passwords and login, Sitecore Admin password, etc. per environment for example: DEV, QA, PROD, etc.

#### Task 4: Deploy a custom Sitecore XM ARM Templates (15 min)

In this section, the attendee will customize the default Sitecore XM ARM Templates by taking into the KeyVault secrets previously created and will as well update the default azuredeploy.json file to allow the re-deployment of the infrastructure with these updates. Good opportunity to manipulate a bit an ARM Template and navigate through it.

1. With a Windows Explorer, create the folder “C:\SitecoreLab\Templates”.
2. Go to <https://raw.githubusercontent.com/Sitecore/Sitecore-Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm/azuredeploy.json>
3. Copy/paste the code as “azuredeploy-custom.json” file into “C:\SitecoreLab\Templates”.
4. Open the azuredeploy-custom.json with **Visual Studio Code**.

The screenshot shows the Visual Studio Code interface with the 'azuredelay-custom.json' file open in the editor. The file content is as follows:

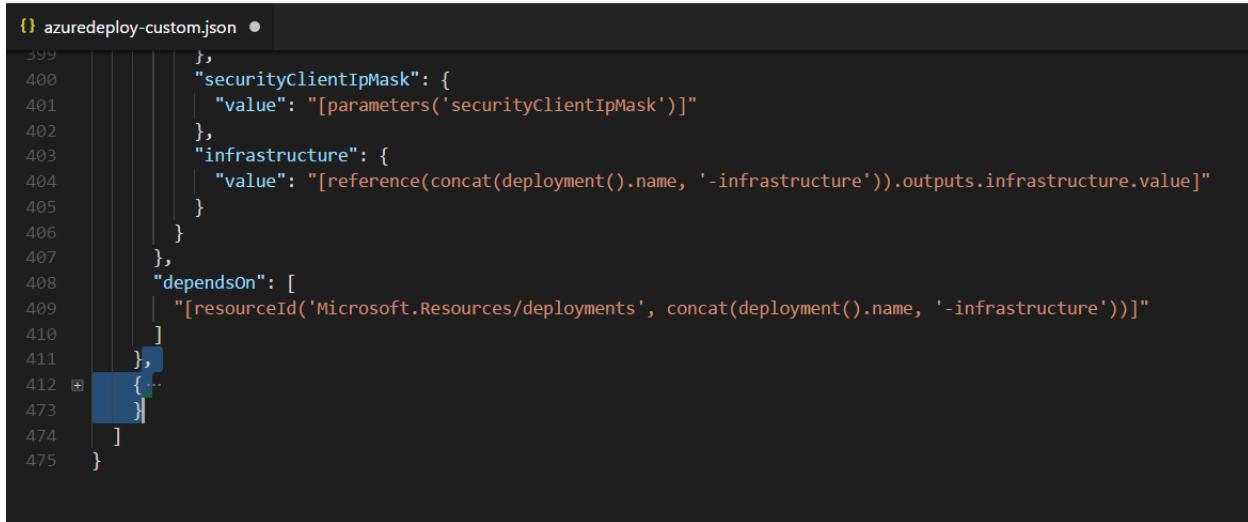
```

1  {
2   "$schema": "http://schema.management.azure.com/schemas/2015-01-01/de
3   "contentVersion": "1.0.0.0",
4   "variables": {
5     "resourcesApiVersion": "2016-09-01",
6     "defaultDependency": [
7       {
8         "name": "application"
9       }
10    ],
11    "dependencies": "[concat(var
12    ],
13    ],
14    ],
15    ],
16    ],
17    ],
18    ],
19    ],
20    ],
21    ]
  },
  "parameters": {
    "modules": {
      "type": "secureObject",
      "defaultValue": {
        "items": [
          {
            "name": "empty",
            "templateLink": "[co
            "parameters": {
  
```

The right-hand sidebar shows a context menu with the following options:

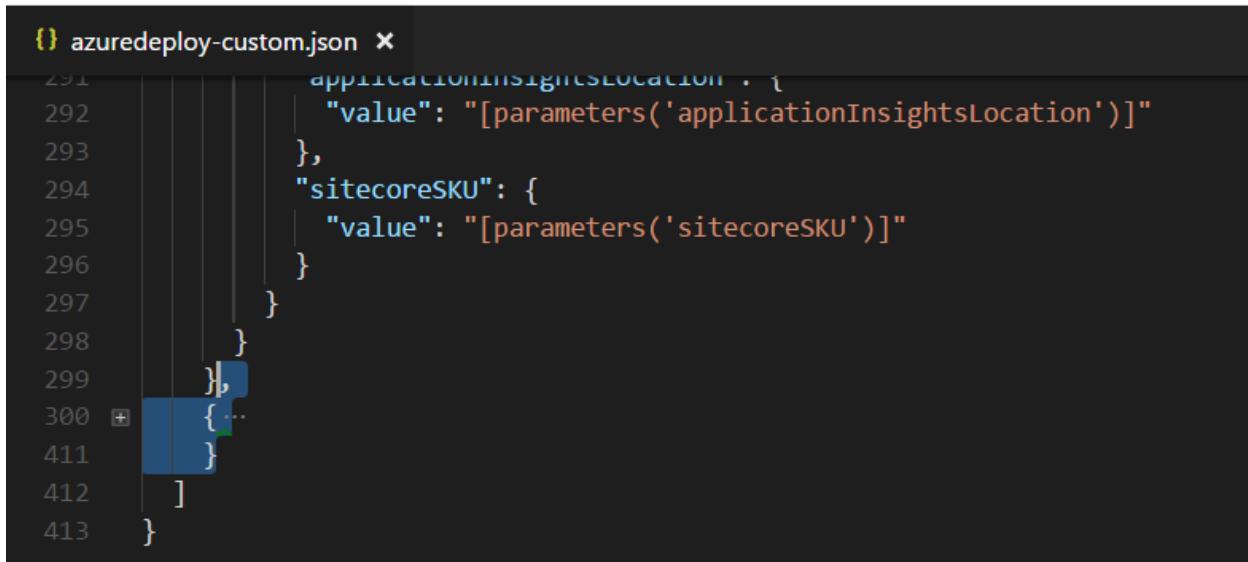
- Go to Definition F12
- Peek Definition Alt+F12
- Find All References Shift+F12
- Rename Symbol F2
- Change All Occurrences Ctrl+F2
- Format Document Alt+Shift+F
- Cut Ctrl+X
- Copy Ctrl+C

5. Right click anywhere on the file opened and click on “**Format Document**” and then **Save the changes**.
6. Go to the end of the document and collapse the line 412 like illustrated and remove the lines from line 412 to line 473 included (+ the extra ‘,’ at the end of line 411). *We are here removing the deployment of the “Sitecore modules” part, we don’t need it for this case.*



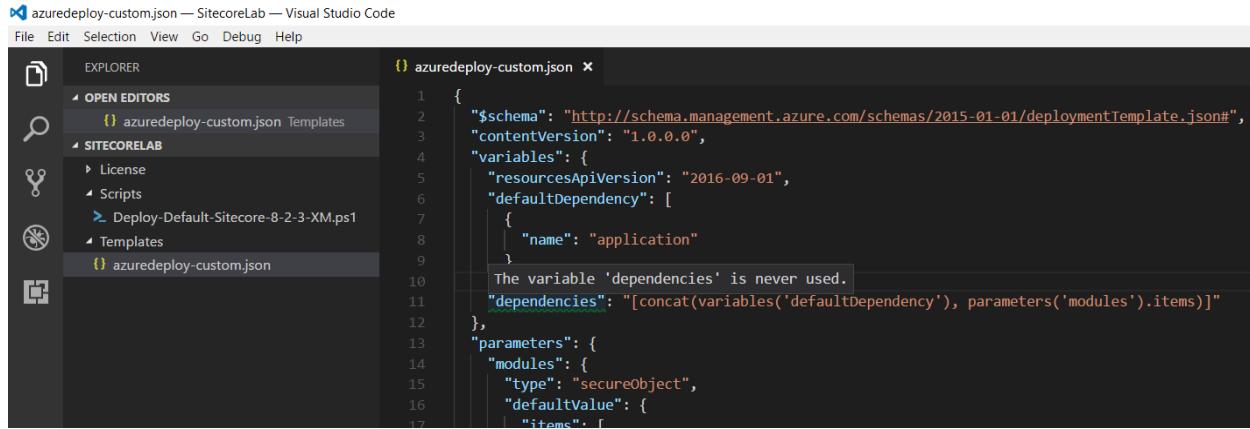
```
{
  "399": "  },
  "400": "  },
  "401": "    \"securityClientIpMask\": {
  "402": "      \"value\": \"[parameters('securityClientIpMask')]\"
  "403": "    },
  "404": "    \"infrastructure\": {
  "405": "      \"value\": \"[reference(concat(deployment().name, '-infrastructure')).outputs.infrastructure.value]\""
  "406": "    }
  "407": "  },
  "408": "  \"dependsOn\": [
  "409": "    [resourceId('Microsoft.Resources/deployments', concat(deployment().name, '-infrastructure'))]"
  "410": "  ]
  "411": "},
  "412": "  {..}
  "413": "}
  "414": "}
  "415": "}
  "416": "}
  "417": "}
  "418": "}
  "419": "}
  "420": "}
  "421": "}
  "422": "}
  "423": "}
  "424": "}
  "425": "}
  "426": "}
  "427": "}
  "428": "}
  "429": "}
  "430": "}
  "431": "}
  "432": "}
  "433": "}
  "434": "}
  "435": "}
  "436": "}
  "437": "}
  "438": "}
  "439": "}
  "440": "}
  "441": "}
  "442": "}
  "443": "}
  "444": "}
  "445": "}
  "446": "}
  "447": "}
  "448": "}
  "449": "}
  "450": "}
  "451": "}
  "452": "}
  "453": "}
  "454": "}
  "455": "}
  "456": "}
  "457": "}
  "458": "}
  "459": "}
  "460": "}
  "461": "}
  "462": "}
  "463": "}
  "464": "}
  "465": "}
  "466": "}
  "467": "}
  "468": "}
  "469": "}
  "470": "}
  "471": "}
  "472": "}
  "473": "}
  "474": "}
  "475": "}"
}
```

7. Repeat the same action by collapsing line 300 and removing lines from line 300 to line 411 (+ the extra ‘,’ at the end of line 299).



```
{
  "291": "  },
  "292": "  },
  "293": "    \"applicationInsightsLocation\": {
  "294": "      \"value\": \"[parameters('applicationInsightsLocation')]\"
  "295": "    },
  "296": "    \"sitecoreSKU\": {
  "297": "      \"value\": \"[parameters('sitecoreSKU')]\"
  "298": "    }
  "299": "  },
  "300": "  {..}
  "301": "}
  "302": "}
  "303": "}
  "304": "}
  "305": "}
  "306": "}
  "307": "}
  "308": "}
  "309": "}
  "310": "}
  "311": "}
  "312": "}
  "313": "}
  "314": "}
  "315": "}
  "316": "}
  "317": "}
  "318": "}
  "319": "}
  "320": "}
  "321": "}
  "322": "}
  "323": "}
  "324": "}
  "325": "}
  "326": "}
  "327": "}
  "328": "}
  "329": "}
  "330": "}
  "331": "}
  "332": "}
  "333": "}
  "334": "}
  "335": "}
  "336": "}
  "337": "}
  "338": "}
  "339": "}
  "340": "}
  "341": "}
  "342": "}
  "343": "}
  "344": "}
  "345": "}
  "346": "}
  "347": "}
  "348": "}
  "349": "}
  "350": "}
  "351": "}
  "352": "}
  "353": "}
  "354": "}
  "355": "}
  "356": "}
  "357": "}
  "358": "}
  "359": "}
  "360": "}
  "361": "}
  "362": "}
  "363": "}
  "364": "}
  "365": "}
  "366": "}
  "367": "}
  "368": "}
  "369": "}
  "370": "}
  "371": "}
  "372": "}
  "373": "}
  "374": "}
  "375": "}
  "376": "}
  "377": "}
  "378": "}
  "379": "}
  "380": "}
  "381": "}
  "382": "}
  "383": "}
  "384": "}
  "385": "}
  "386": "}
  "387": "}
  "388": "}
  "389": "}
  "390": "}
  "391": "}
  "392": "}
  "393": "}
  "394": "}
  "395": "}
  "396": "}
  "397": "}
  "398": "}
  "399": "}
  "400": "}
  "401": "}
  "402": "}
  "403": "}
  "404": "}
  "405": "}
  "406": "}
  "407": "}
  "408": "}
  "409": "}
  "410": "}
  "411": "}
  "412": "}
  "413": "}"
}
```

8. To clean up and explore more this file we could now remove the unused variables and parameters of this file, go to the top of the file and look for all the messages “**The variable ‘xx’ is never used**” or “**The parameter ‘xx’ is never used**”. *Remark: the more you will remove them, more will happen because of dependencies, so make sure you do another check.*



```
1  {
2   "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3   "contentVersion": "1.0.0.0",
4   "variables": {
5     "resourcesApiVersion": "2016-09-01",
6     "defaultDependency": [
7       {
8         "name": "application"
9       }
10    ]
11  },
12  "dependencies": "[concat(variables('defaultDependency'), parameters('modules').items)]"
13},
14  "parameters": {
15    "modules": {
16      "type": "secureObject",
17      "defaultValue": {
18        "items": [
19          {
20            "value": "[parameters('cdWebAppName')]"
21          },
22          {
23            "redisCacheName": {
24              "value": "[parameters('redisCacheName')]"
25            }
26          },
27          {
28            "searchServiceName": {
29              "value": "[parameters('searchServiceName')]"
30            }
31          },
32          {
33            "applicationInsightsName": {
34              "value": "[parameters('applicationInsightsName')]"
35            }
36          },
37          {
38            "applicationInsightsLocation": {
39              "value": "[parameters('applicationInsightsLocation')]"
40            }
41          },
42          {
43            "sitecoreSKU": {
44              "value": "[parameters('sitecoresku')]"
45            }
46          }
47        ]
48      }
49    }
50  }
51 }
```

9. Then you could repeat the “Format Document” by right-clicking anywhere on this file. You should have now a file with 204 lines:



```
183  "value": "[parameters('cdWebAppName')]"
184},
185  "redisCacheName": {
186    "value": "[parameters('redisCacheName')]"
187  },
188  "searchServiceName": {
189    "value": "[parameters('searchServiceName')]"
190  },
191  "applicationInsightsName": {
192    "value": "[parameters('applicationInsightsName')]"
193  },
194  "applicationInsightsLocation": {
195    "value": "[parameters('applicationInsightsLocation')]"
196  },
197  "sitecoreSKU": {
198    "value": "[parameters('sitecoresku')]"
199  }
200}
201}
202]
203]
204}
```

**Note:** So, now we have a clean ARM Template locally with “variables”, “parameters” and only one “resources” to deploy. This resource to deploy is targeting another ARM Template ‘nested/infrastructure.json’ which is stored in the official Sitecore GitHub repository [here](#). For this lab, we will use this remote version but for your own future needs, you may have

to download and edit it locally. Take few minutes to explore it and go through the variables, parameters and resources. That's this file which is going to define all the Azure resources needed (i.e. the infrastructure) and their associated settings: Azure Web App, Application Insights, SQL Database, Redis Cache, etc. Ask your proctor if you have any questions about it because we are not going deeper than that with this file during this lab.

10. Now let's integrate the KeyVault secrets. At the top of the azuredeploy-custom.json file, add a keyVaultId variable and the two keyVaultName and keyVaultResourceGroupName parameters:

```
{} azuredeploy-custom.json x
1 {
2     "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
3     "contentVersion": "1.0.0.0",
4     "variables": {
5         "resourcesApiVersion": "2016-09-01",
6         "keyVaultId": "[resourceId(parameters('keyVaultResourceGroupName'), 'Microsoft.KeyVault/vaults', parameters('keyVaultName'))]"
7     },
8     "parameters": {
9         "keyVaultName": {
10             "type": "string"
11         },
12         "keyVaultResourceGroupName": {
13             "type": "string"
14         },
15         "templateLinkBase": {
16             "type": "string",
17             "defaultValue": "[uri(replace(deployment().properties.templateLink.uri, ' ','%20'), '.')]"
18         },
19         "templateLinkAccessToken": {
20             "type": "securestring",
21             "defaultValue": ""
22         },
23         "deploymentId": {
24             "type": "string",
25             "defaultValue": "[resourceGroup().name]"
26         },
27         "location": {
28             "type": "string",
29             "defaultValue": "[resourceGroup().location]"
30         }
},
```

11. Let's remove the parameter "sqlServerPassword" and its dependency "webSqlServerPassword". We are not changing anything about the sqlServerLogin since this parameter within the nested template is a string and not a securestring. We could change the nested template to reflect that, but that's not within the scope of this lab, so we will manipulate the associated KeyVault secret by PowerShell.

```
{} azuredeploy-custom.json > Deploy-Custom-Sitecore-8-2-3-XM.ps1
  33     "defaultValue": "12.0"
  34   },
  35   "sqlDatabaseCollation": {
  36     "type": "string",
  37     "defaultValue": "SQL_Latin1_General_CI_AS"
  38   },
  39   "sqlServerName": {
  40     "type": "string",
  41     "defaultValue": "[concat(parameters('deploymentId'), '-sql')]"
  42   },
  43   "sqlServerLogin": {
  44     "type": "string",
  45     "minLength": 1
  46   },
  47   "sqlServerPassword": {
  48     "type": "securestring",
  49     "minLength": 8
  50   },
  51   "webSqlServerName": {
  52     "type": "string",
  53     "defaultValue": "[concat(parameters('deploymentId'), '-web-sql')]"
  54   },
  55   "webSqlServerLogin": {
  56     "type": "string",
  57     "minLength": 1,
  58     "defaultValue": "[parameters('sqlServerLogin')]"
  59   },
  60   "webSqlServerPassword": {
  61     "type": "securestring",
  62     "minLength": 8,
  63     "defaultValue": "[parameters('sqlServerPassword')]"
  64   },
  65   "coreSqlDatabaseName": {
  66     "type": "string",
  67     "defaultValue": "[concat(parameters('deploymentId'), '-core-db')]"
  68   },
```

12. Let's fix the errors more below in this template. You could remove the lines from 164 to 166. If you look at the nested ARM Template, this parameter will use as defaultValue sqlServerPassword.

```
{} azuredeploy-custom.json > Deploy-Custom-Sitecore-8-2-3-XM.ps1
```

```
157 },
158 "webSqlServerName": {
159     "value": "[parameters('webSqlServerName')]"
160 },
161 "webSqlServerLogin": {
162     "value": "[parameters('webSqlServerLogin')]"
163 },
164 "webSqlServerPassword": {
165     "value": "[parameters('webSqlServerPassword')]"
166 },
167 "coreSqlDatabaseName": {
168     "value": "[parameters('coreSqlDatabaseName')]"
169 },
170 "masterSqlDatabaseName": {
171     "value": "[parameters('masterSqlDatabaseName')]"
172 },
173 "webSqlDatabaseName": {
```

13. Now let's update lines from 150 and 157 to take the associated value from our Azure KeyVault secret like illustrated below:

```

{} azuredeploy-custom.json x Deploy-Custom-Sitecore-8-2-3-XM.ps1
140 },
141     "sqlServerVersion": {
142         "value": "[parameters('sqlServerVersion')]"
143     },
144     "sqlDatabaseCollation": {
145         "value": "[parameters('sqlDatabaseCollation')]"
146     },
147     "sqlServerLogin": {
148         "value": "[parameters('sqlServerLogin')]"
149     },
150     "sqlServerPassword": {
151         "reference": {
152             "keyVault": {
153                 "id": "[variables('keyVaultId')]"
154             },
155             "secretName": "SqlserverPassword"
156         }
157     },
158     "webSqlServerName": {
159         "value": "[parameters('webSqlServerName')]"
160     },
161     "webSqlServerLogin": {
162         "value": "[parameters('webSqlServerLogin')]"
163     },
164     "coreSqlDatabaseName": {
165         "value": "[parameters('coreSqlDatabaseName')]"
166     },

```

14. Duplicate the C:\SitecoreLab\PowerShell\Deploy-Default-Sitecore-8-2-3-XM.ps1 and rename it by **Deploy-Custom-Sitecore-8-2-3-XM.ps1**.
15. Right-click on it and select “**Edit**”. Then “**PowerShell ISE**” tool will be opened. Here, let’s adapt this script.
16. The result file should look like illustrated below. *Note: please make sure you have seen the highlighted parts:*

```

Windows PowerShell ISE
File Edit View Tools Debug Add-ons Help
Deploy-Custom-Sitecore-8-2-3-XM.ps1 X
1 # PowerShell script to deploy a local customized Sitecore 8.2.3 XM ARM Template
2
3 [Param(
4     [string] [Parameter(Mandatory=$true)] $SubscriptionId,
5     [string] [Parameter(Mandatory=$true)] $ResourceGroupName,
6     [string] [Parameter(Mandatory=$true)] $keyVaultName,
7     [string] $TemplateFile = "..\templates\azuredploy-custom.json",
8     [string] $nestedTemplateLinkBase = "https://raw.githubusercontent.com/Sitecore/Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm/",
9     [string] $SitecoreSKU = "xM4"
10)]
11
12 Login-AzureRmAccount;
13 Select-AzureRmSubscription -SubscriptionId $SubscriptionId;
14
15 # we need to get the value of SqlServerLogin in PowerShell instead of doing that directly from the ARM Template since the type of the parameter in the nested template is string instead of securestring.
16 # It allows to manipulate an Azure KeyVault by PowerShell. The other alternative could be to update the nested template by changing the type of the parameter by a securestring.
17 $sqlServerLoginSecret = Get-AzureKeyVaultSecret -VaultName $keyVaultName -Name $SqlServerLogin;
18
19 $parameters = New-Object -Type NameValueCollection;
20 $parameters.Add("sqlServerLogin", $sqlServerLoginSecret.SecretValueText);
21 $parameters.Add("sitecoreResourceGroup", $ResourceGroupName);
22 $parameters.Add("keyVaultName", $keyVaultName);
23 $parameters.Add("keyVaultResourceGroupName", $ResourceGroupName);
24 $parameters.Add("templateLinkBase", $nestedTemplateLinkBase);
25
26 New-AzureRmResourceGroupDeployment -Name $anotherDeployment -ResourceGroupName $ResourceGroupName -TemplateFile $TemplateFile -TemplateParameterObject $parameters -Verbose;

```

17. Now, let's execute this script with this simple command below:

```
. \ Deploy-Custom-Sitecore-8-2-3-XM.ps1
```

18. You will be prompted to enter complementary parameters value:

- a. **SubscriptionId**: enter the id of the Azure subscription in which you already deployed your Sitecore solution for this lab.
- b. **ResourceGroupName**: enter the same name of the Resource Group you provisioned earlier with all the Default ARM Templates.
- c. You will be asked to sign-in as well.

19. After ~3 min and 20 seconds, the deployment should be done. You could have a look via the Azure portal the history of the “**Deployments**” of the Resource Groups, you should now see a new entry named “infra-update”. Furthermore, because we changed the sitecoreSKU value from xM2 to xM4 you should have the Sitecore web instance App Service Plan now as Standard S3 and the Sitecore web Sql Database as Standard S2.

The screenshot shows the Azure portal interface for the 'mabenoit-test' resource group. On the left, there is a navigation sidebar with options like Overview, Activity log, Access control (IAM), Tags, Quickstart, Resource costs, Deployments, and Policies. The main area is titled 'mabenoit-test - Deployments' and shows a table of deployment history. The table has columns for Deployment Name, Status, Timestamp, Duration, and Related events. There are seven entries listed:

DEPLOYMENT NAME	STATUS	TIMESTAMP	DURATION	RELATED EVENTS
another-deployment	Succeeded	6/11/2017 1:15:09 PM	3 minutes 22 seconds	<a href="#">Related events</a>
another-deployment-infrast...	Succeeded	6/11/2017 1:14:56 PM	3 minutes 5 seconds	<a href="#">Related events</a>
mabenoit-test	Succeeded	6/7/2017 11:48:36 AM	29 minutes 21 seconds	<a href="#">Related events</a>
mabenoit-test-empty	Succeeded	6/7/2017 11:48:31 AM	2 seconds	<a href="#">Related events</a>
mabenoit-test-application	Succeeded	6/7/2017 11:48:12 AM	17 minutes 44 seconds	<a href="#">Related events</a>
mabenoit-test-infrastructure	Succeeded	6/7/2017 11:30:21 AM	11 minutes 3 seconds	<a href="#">Related events</a>

## Exercise 3 – Azure SQL Database (20 min)

### Objectives

The goal of this exercise is to be familiar with the Azure SQL Database service and see the key features.

Through this exercise, you will play/use with:

- Azure portal
- Azure SQL Database
- Visual Studio



### Task 1: Scale up/down your databases (5 min)

In this section, the attendee will scale up the pricing tier of the master SQL Database using the Microsoft Azure Portal.

1. Go to <https://portal.azure.com>
2. Open the Resource group containing all the Azure services previously deployed. Then select the “SQL database” with the suffix name “-master-db”.

A screenshot of the Microsoft Azure Resource Groups blade. The left sidebar shows a list of resource groups: 'mabenoit-sitecore', 'mabenoit', 'mabenoit-function', 'mabenoit-logicapp', 'sitecorelab', and 'VS-mabenoit-ms-Group'. The main pane shows the 'mabenoit-sitecore' resource group details. In the center, there's an 'Overview' card with a 'Subscription name (change)' field set to 'mabenoit', a 'Deployment' section showing '4 Succeeded', and a 'Resource' section listing various resources like 'mabenoit-sitecore' (Storage account), 'mabenoit-sitecore-al' (Application Insights), etc. At the bottom, a table lists resources by name and type. Two specific entries are highlighted with red boxes: 'mabenoit-sitecore-master-db' (SQL database) and 'mabenoit-sitecore-web-sql' (SQL database). Both entries have their 'TYPE' column labeled 'SQL database'.

3. Then navigate to the “Pricing tier (scale DTUs)” blade.

4. There you could change the Pricing tier value up or down, and click on the “**Apply**” button. After few seconds, a “**Scaling in progress...**” message should appear. Then, few seconds/minutes after the scaling process is done.

## Task 2: Local connection with Visual Studio (7 min)

In this section, the attendee will add a Firewall rule on the master SQL Database using the Microsoft Azure Portal to be able to connect to it from its local machine.

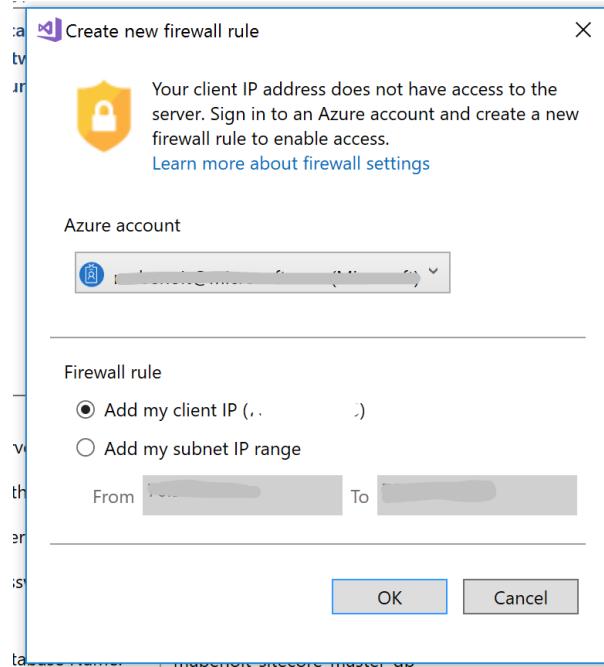
1. Go to the “**Overview**” blade of this “...-master-db” database.

The screenshot shows the Microsoft Azure portal interface. At the top, there's a search bar and a 'Report a bug' link. Below the search bar, the URL shows the resource group: 'Resource groups > mabenot-sitecore > mabenot-sitecore-master-db'. On the far right, there's a 'Search resources' bar. The main content area is titled 'mabenot-sitecore-master-db' and shows a 'SQL database' icon. A toolbar at the top has several actions: 'Copy', 'Restore', 'Export', 'Set server firewall', and 'Delete'. The 'Tools' button is highlighted with a red circle. The left sidebar has a tree view with 'Overview' (circled in red), 'Activity log', 'Tags', 'Diagnose and solve problems', 'SETTINGS' (with 'Quick start', 'Pricing tier (scale DTUs)', 'Geo-Replication', 'Auditing & Threat Detection', and 'Dynamic Data Masking'), and a 'Search (Ctrl+F)' field. The right panel displays 'Essentials' information like Resource group, Status, Location, Subscription name, and Monitoring. It also shows 'Resource utilization' at 100%.

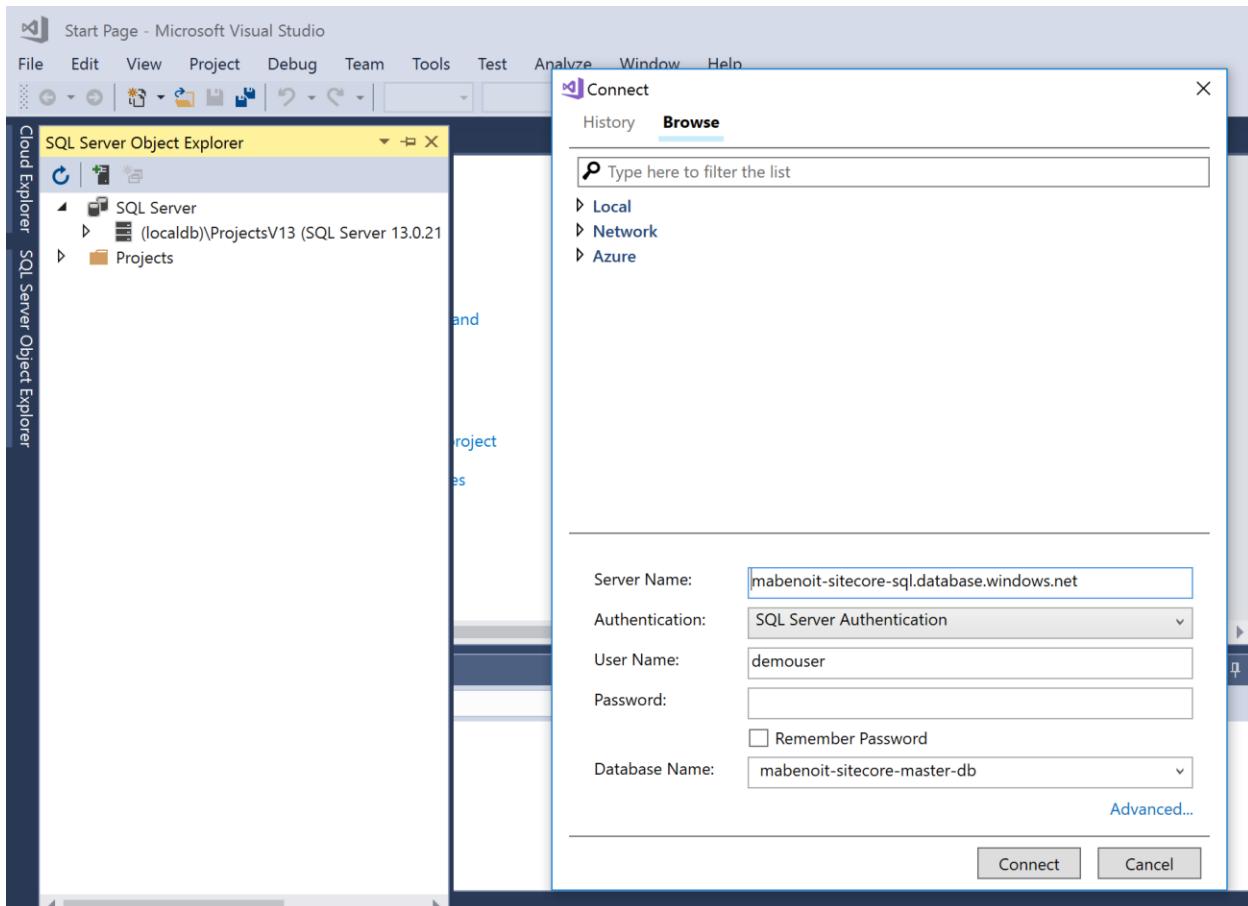
2. Then click on the “**Tools**” toolbar action and choose “**Open in Visual Studio**” to launch your local Visual Studio Community 2017.

The screenshot shows the 'Tools' blade for the 'mabenot-sitecore-master-db' resource group. The 'Open In Visual Studio' option is highlighted with a red circle. To the right, a separate window titled 'Open In Visual Studio' displays the Visual Studio logo and a large blue 'Open In Visual Studio' button. The main blade also lists 'Query editor (preview)' and other options. The right side of the screen shows a 'Configure Firewall' section with a note about setting up the machine to access the database, and links to 'Get Visual Studio' and download tools.

3. In Visual Studio, the “**Create new firewall rule**” will be prompted. You could use the 2 options proposed, but for this lab we will do that through the Azure portal. So, click on the “**Cancel**” button.



4. You will be asked to enter the associated password; all the other information will be populated automatically for you.



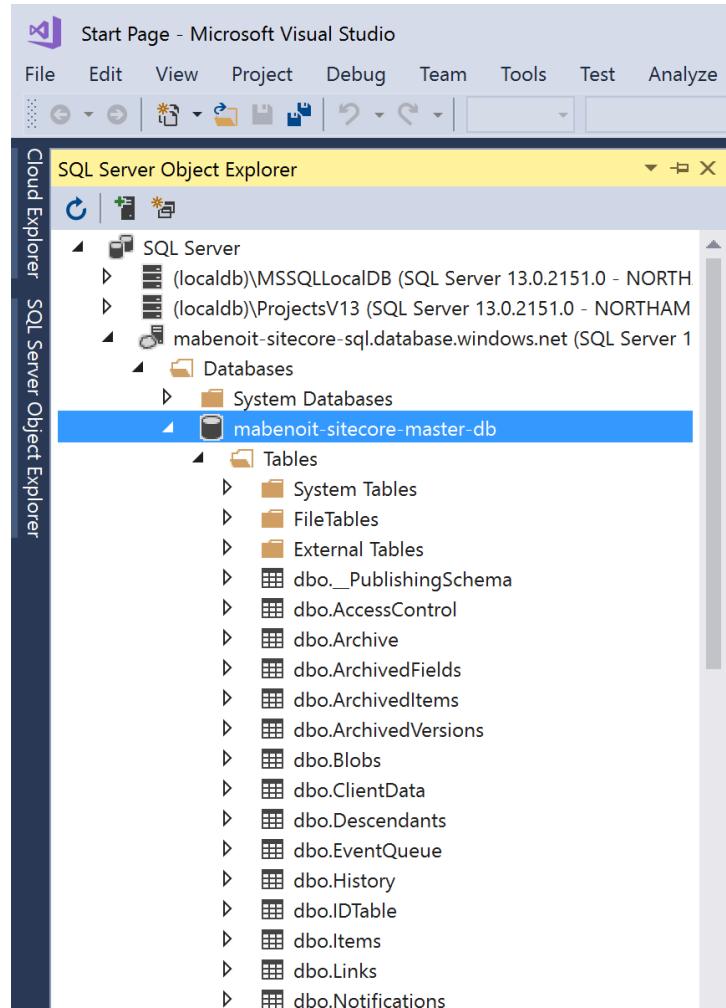
5. Enter the associated password, if you followed the instructions earlier on this lab, it should be "demo@pass12345". Then click on the "**Connect**" button.
6. Depending on which tool you are using you could have an error message saying that you don't have the permission to connect on this server for security reason. In our case with Visual Studio 2017, the "**Create new firewall rule**" dialog will be prompted again. Click on the "**Cancel**" button again.
7. Go back to the Azure portal and open the "...-master-db" resource, on the "**Overview**" blade click on the "**Set server firewall**" toolbar action.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various icons and a search bar. The main area displays the 'Overview' blade for a SQL database named 'mabenoit-sitecore-master-db'. At the top of this blade, there's a toolbar with several buttons: 'Tools', 'Copy', 'Restore', 'Export', 'Set server firewall' (which is circled in red), and 'Delete'. Below the toolbar, there's a section titled 'Essentials' with details about the resource group, status, location, and pricing tier. To the right, there's a detailed view of the server configuration, including the server name, connection strings, and geo-replication role.

8. There, click on "**Add client IP**" and then "**Save**".

This screenshot shows the 'Firewall settings' blade for the same SQL database. The toolbar at the top includes 'Save' (circled in red) and 'Discard'. Below the toolbar, there's an informational message about client IP access. The main area contains a table for defining firewall rules, with columns for 'RULE NAME', 'START IP', and 'END IP'. A single rule is listed: 'ClientIPAddress\_2017-3-0...' with 'START IP' set to '192.168.1.100' and 'END IP' set to '192.168.1.100'.

9. Go back to Visual Studio and click on the "**Connect**" button. You should now be able to successfully login on this server/database and browse the tables, etc.

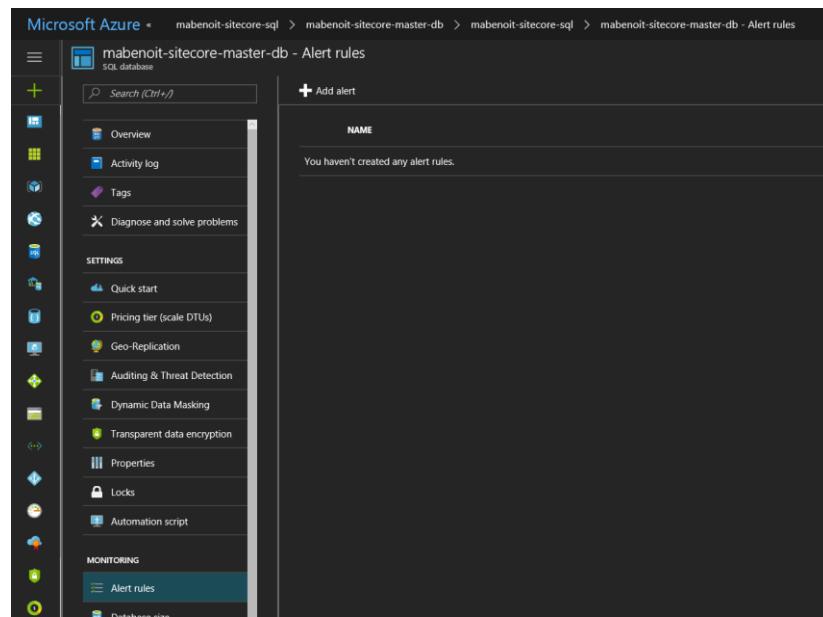


**Note:** Instead of Visual Studio you could connect to your Azure SQL Database locally via other tools like SQL Server Management Studio (SSMS). Furthermore, setting the correct Firewall rules will allow you to locally sync your Sitecore content via tools like Sitecore Rocks, TDS or Unicorn for example during the Development phase.

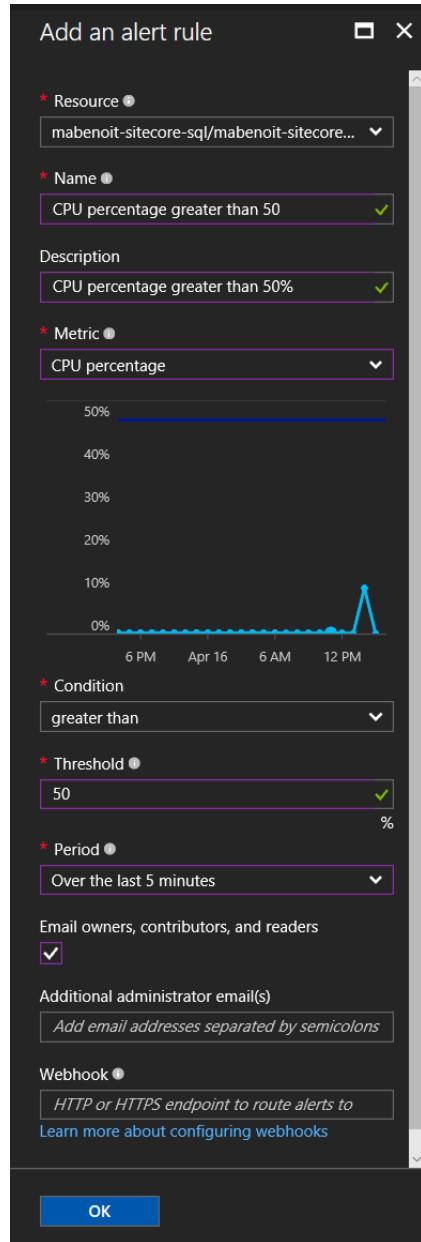
### Task 3: Add Alert Rules (3 min)

In this section, the attendee will add an Alert rule on the master SQL Database using the Microsoft Azure portal to be notified by email when the CPU percentage used is greater than 50%.

1. Go to the master SQL Database resource on the Azure portal and click on the “Alert rules” blade.



2. Click on the “Add alert” toolbar button.
3. Fill out the mandatory fields for example with the information:
  - a. **Name:** CPU percentage greater than 50
  - b. **Description:** CPU percentage greater than 50%
  - c. **Metric:** CPU percentage
  - d. **Threshold:** 50
  - e. **Email owners, contributors, and readers:** “checked”



4. Then click on the “OK” button and your alert will be in place.

#### Task 4: Configure a Geo-replication (5 min)

In this section, the attendee will provision a secondary web SQL Database and configure Geo-Replication by adding a Failover group using the Microsoft Azure Portal.

1. Go to the web SQL Server resource named “...-web-sql” on the Azure portal and click on the “Failover groups” blade.

mabenoittest-896e-web-sql - Failover groups

+ Add group Refresh

Failover group are a SQL server feature designed to automatically manage replication, connectivity and failover of a set of databases.

NAME	PRIMARY SERVER	PARTNER SECONDARY SERVER	READ/WRITE FAILOVER POLICY	DATABASE COUNT
You have no group created				

Search (Ctrl+F)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

SETTINGS

Quick start

Firewall

Failover groups

Long-term backup retention

Auditing & Threat Detection

Active Directory admin

2. Click on the “**Add group**” toolbar button and fill out all the fields by adding a new server in West US2 for example with the same login/password than the web SQL Database.

- a. **Server admin login:** demouser
- b. **Password and Confirm Password:** demo@pass12345

## Failover group

Create a failover group to automatically failover databases in it.

\* Failover group name  
web-failover-group ✓

\* Secondary server  
mabenoittest-896e-web-west-sql... >

Read/Write failover policy  
Automatic

Read/Write grace period (hours)  
1 hours

Database within the group  
1 / 1 >

**Create**

3. Then click on the “**Create**” button.
4. Once the Failover group will be created, it will appear in the list below:

NAME	PRIMARY SERVER	PARTNER SECONDARY SERVER	READ/WRITE FAILOVER POLICY	DATABASE COUNT
web-failover-group	mabenoittest-896e-web-sql	mabenoittest-896e-web-west-sql	Automatic, 1 hours	1/1

5. At this point, the two listeners end points are created, one for primary and another for secondary.
  - a. Primary: <FailoverGroupName>.database.windows.net
  - b. Secondary: <FailoverGroupName>.secondary.database.windows.net

**Note:** If at this point you go to SSMS and try to connect to your Primary/Secondary database using associated listeners, you will receive error and will not be able to login. Ideally it should have allowed, but it currently fails, as it tries to connect to the Master database which is currently not part of the group. This is currently being worked upon and should be resolved soon. Till then, workaround is to provide the database name while connecting to server. Use the “Options” button to provide database name.

6. For further considerations, not included in this lab:
  - a. You could change all your connection strings to target the primary listener for your web SQL Database.
  - b. You could use the “**Fail-over**” action to switch the primary and secondary roles. Without changing your connection strings in your web app! 😊

## Takeaways

The “**Pricing tier (scale DTUs)**” feature allows you to scale up/down each SQL Database to optimize your resource cost depending of the traffic you have at different time/moment. Furthermore, you could automate this process via CLI and PowerShell for example.

The “**Copy**”, “**Export**” and “**Restore**” features are great and could be performed via the Azure portal, CLI or PowerShell for more automation. To complete the built-in Backup feature on each SQL Database, at the SQL Server level you could configure a more advanced “**Long-term backup retention**”. [This link](#) illustrates the difference between Backup vs. Import/Export.

The “**Performance overview**”, “**Performance recommendations**”, “**Auditing & Threat Detection**” and such pro-active features are great as well to look at for more anticipation.

When you have a lot of databases on the same Azure SQL server, and you would like to decrease the price, you may study the use of an [Azure Elastic Pool](#).

The Geo-replication is great if your CDs server should be geo-located, you could use this feature to replicate your web database instead of having them as target database during your Sitecore publishing process. This feature could be used as well for your HA and DR plans and why not some content deployment scenarios.

Some other links:

- Monitoring and maintaining Azure SQL
  - [https://doc.sitecore.net/sitecore\\_experience\\_platform/setting\\_up\\_and\\_maintaining/sitecore\\_on\\_azure/analytics/monitoring\\_and\\_maintaining\\_azure\\_sql](https://doc.sitecore.net/sitecore_experience_platform/setting_up_and_maintaining/sitecore_on_azure/analytics/monitoring_and_maintaining_azure_sql)

## Exercise 4 - Azure Web App (65 min)

### Objectives

The goal of this exercise is to be familiar with the Azure App Service and Azure Web App and see the key features.

Through this exercise, you will play/use with:

- Azure portal
- Azure App Service
- Azure Web App
- PowerShell
- Blob Storage
- Sitecore Content Editor



### Task 1: Verify the Sitecore solution previously provisioned (5 min)

In this section, the attendee will verify the Sitecore solution previously provisioned with the ARM Templates is ready to use. To do so, we will browse the 2 Sitecore instances: CM and CD.

1. Go to the Azure portal - <https://portal.azure.com> and open the Resource Group containing all the services provisioned earlier. You should see all the services below:

NAME	TYPE	LOCATION
mabenoit-sitecore-ai	Application Insights	East US
mabenoit-sitecore-as	Search service	East US
mabenoit-sitecore-cd	App Service	East US
mabenoit-sitecore-cd-hp	App Service plan	East US
mabenoit-sitecore-cm	App Service	East US
mabenoit-sitecore-cm-hp	App Service plan	East US
mabenoit-sitecore-redis	Redis Cache	East US
mabenoit-sitecore-sql	SQL server	East US
mabenoit-sitecore-core-db	SQL database	East US
mabenoit-sitecore-master-db	SQL database	East US
mabenoit-sitecore-web-sql	SQL server	East US
mabenoit-sitecore-web-db	SQL database	East US

2. Open the "...-cm" Web App Service resource and there on the "Overview" blade click on the "Browse" toolbar button.

3. You should see the homepage of the Sitecore Vanilla version:

Welcome to Sitecore +

mabenoit-sitecore-cm.azurewebsites.net

 sitecore

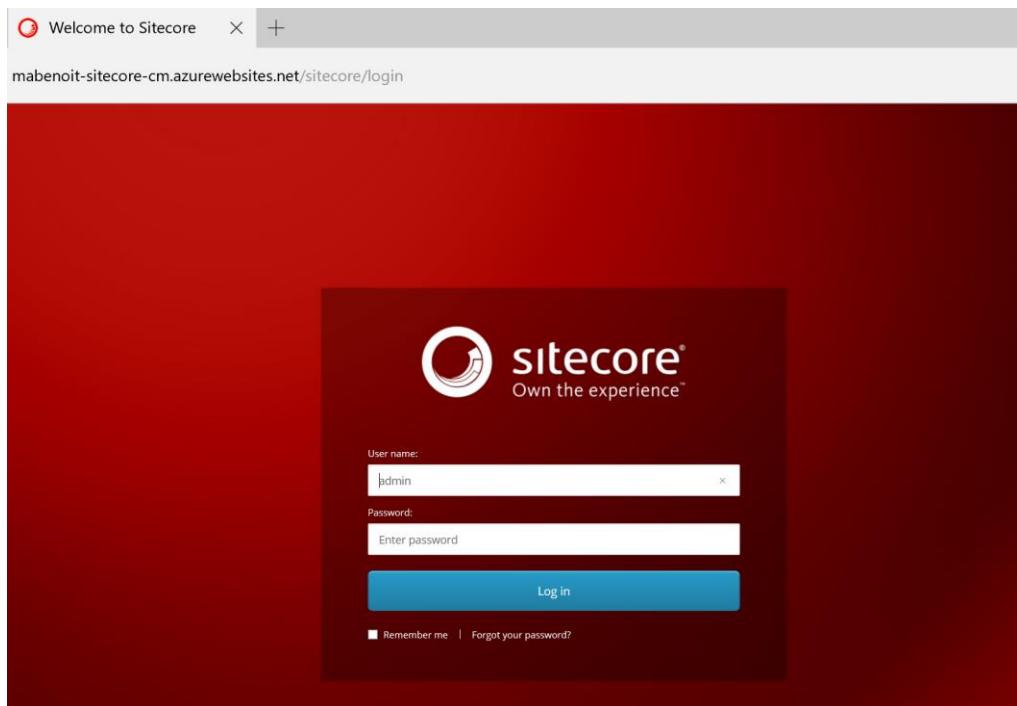
**Sitecore Experience Platform**

From a single connected platform that also integrates with other customer-facing platforms, to a single view of the customer in a big data marketing repository, to completely eliminating much of the complexity that has previously held marketers back, the latest version of Sitecore makes customer experience highly achievable. Learn how the latest version of Sitecore gives marketers the complete data, integrated tools, and automation capabilities to engage customers throughout an iterative lifecycle – the technology foundation absolutely necessary to win customers for life.

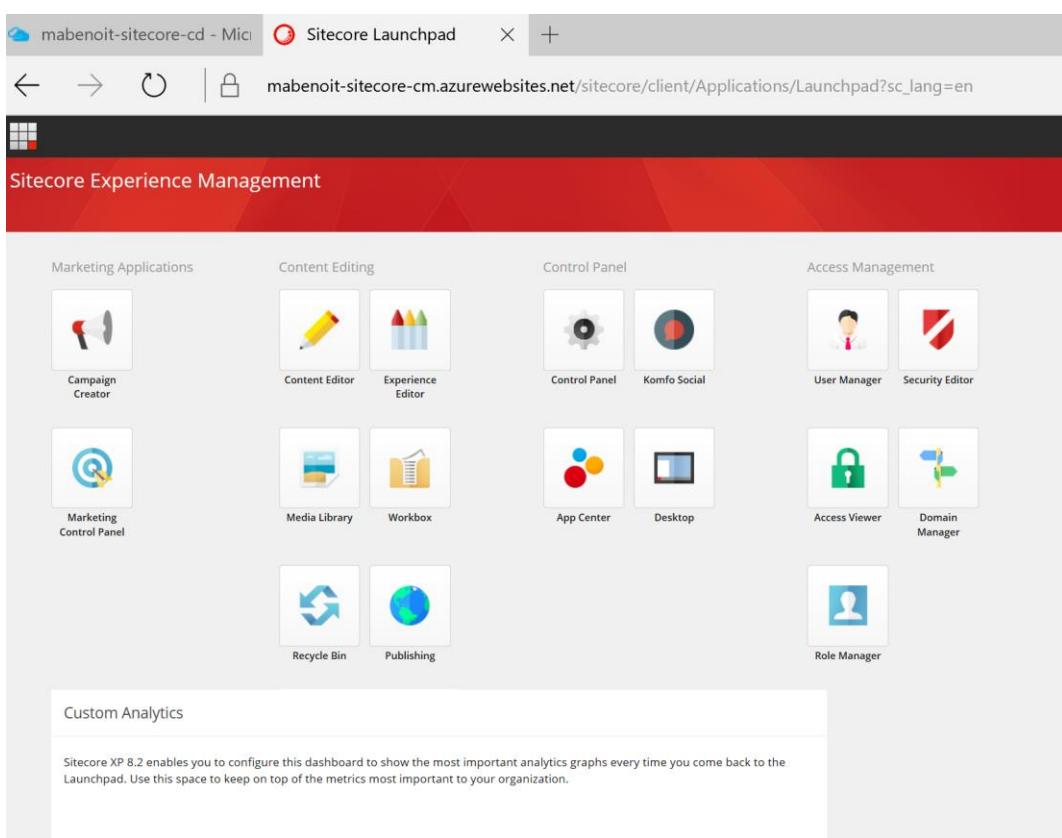
For further information, please go to the [Sitecore Documentation site](#)

© 2017 Sitecore

4. On this web page, add at the end of the URL the “**/sitecore**” suffix. We will be landed on the Sitecore administration login page.



5. If you enter the credentials below, you should be landed on the Sitecore Launchpad page.
  - a. **User name:** admin
  - b. **Password:** demo@pass12345



6. On the Azure portal, go back to the Resource Group, and this time click on the “...-cd” Web App Service resource and there on the “Overview” blade click on the “Browse” toolbar button.

The screenshot shows the Azure portal's 'Overview' blade for an App Service named 'mabenoit-sitecore-cd'. The left sidebar contains navigation links for Search, Overview (which is circled in red), Activity log, Access control (IAM), Tags, Diagnose and solve problems, Deployment (Quickstart, Deployment credentials, Deployment slots, Deployment options, Continuous Delivery (Preview)), Settings (Application settings, Authentication / Authorization, Backups, Custom domains, SSL certificates, Networking), and Monitoring. The main area displays 'Essentials' information including Resource group (mabenoit-sitecore), Status (Running), Location (East US), Subscription name (changed), and Subscription ID (redacted). It also shows 'Monitoring' data for Requests and errors, with a chart showing spikes in requests between 9:15 AM and 10 AM, and a summary of 0 HTTP SERVER ERRORS and 229 REQUESTS.

7. You should see the homepage of the Sitecore Vanilla version:

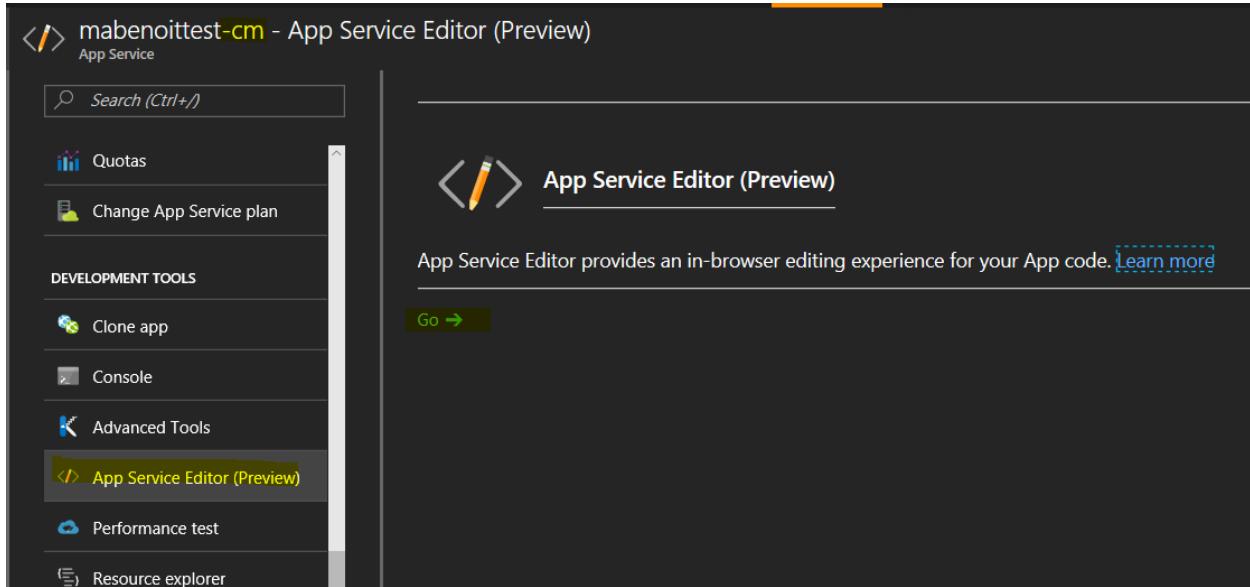
The screenshot shows the Sitecore homepage. The browser title bar reads "Welcome to Sitecore" and the URL is "mabenoit-sitecore-cd.azurewebsites.net". The page itself has a header with the Sitecore logo and a woman smiling while looking at her phone. Below the header, the text "Sitecore Experience Platform" is displayed. A paragraph explains that Sitecore is a single connected platform for customer engagement, mentioning its integration with other platforms, a big data marketing repository, and its focus on customer experience. It encourages users to learn more about Sitecore's capabilities. At the bottom, there is a copyright notice for "© 2017 Sitecore".

**Note:** This checkpoint is important for the rest of the lab if you have any issue or difference with the steps below, please ask your lab proctor.

## Task 2: Add IP restriction on the CM instance (5 min)

In this section, the attendee will add an [IP access restriction](#) for the CM instance. This allows for example to add more security about the provenance of the Sitecore authors. *Could be done as well for the CD (restrict access to QA, UAT, Staging, etc.) but for this lab it will be for the CM only.*

1. From the Azure portal, open the associated Azure Web App CM instance and click on the “**App Service Editor (Preview)**” blade.



2. From there, open the **web.config** file and change the lines between the **<security>** entry like illustrated below:

```

<configuration>
  <system.web>
    <httpHandlers>
      <add verb="*" path="*.aspx" type="Sitecore.DependencyInjection.AutowirededPageHandlerFactory" />
      <add verb="*" name="Sitecore.SpeakJS64" path="~/speak/v1/*/*.js" modules="IsapiModule" />
      <add verb="*" name="Sitecore.SpeakJS32" path="~/speak/v1/*/*.js" modules="IsapiModule" />
      <add verb="*" name="Sitecore.SpeakClassic64" path="sitecore_speak.ashx" modules="IsapiModule" />
      <add verb="*" name="Sitecore.SpeakClassic32" path="sitecore_speak.ashx" modules="IsapiModule" />
      <add verb="*" path="sitecore_speak.ashx" type="Sitecore.Resources.Scripts.ScriptHandler" />
      <add verb="*" path="sitecore_expeditor_speak_request.ashx" type="Sitecore.ExperienceEditor.ExpeditorSpeakRequestHandler" />
    </httpHandlers>
  </system.web>
  <system.webServer>
    <validation validateIntegratedModeConfiguration="false" />
    <security>
      <requestFiltering>
        <requestLimits maxAllowedContentLength="524288000" />
      </requestFiltering>
      <ipSecurity allowUnlisted="false" denyAction="AbortRequest">
        <add ipAddress="192.168.1.100" allowed="true" />
      </ipSecurity>
    </security>
    <httpProtocol>
      <customHeaders>
        <remove name="X-Powered-By" />
      </customHeaders>
    </httpProtocol>
    <rewrite>
      <rules>
        <rule name="Root Hit Force HTTPS Redirection" enabled="true" stopProcessing="true">
          <match url="^$" ignoreCase="false" />
          <conditions>
            <add input="{HTTPS}" pattern="^OFF$" />
            <add input="{HTTP_METHOD}" pattern="GET" />
          </conditions>
        </rule>
      </rules>
    </rewrite>
  </system.webServer>
</configuration>

```

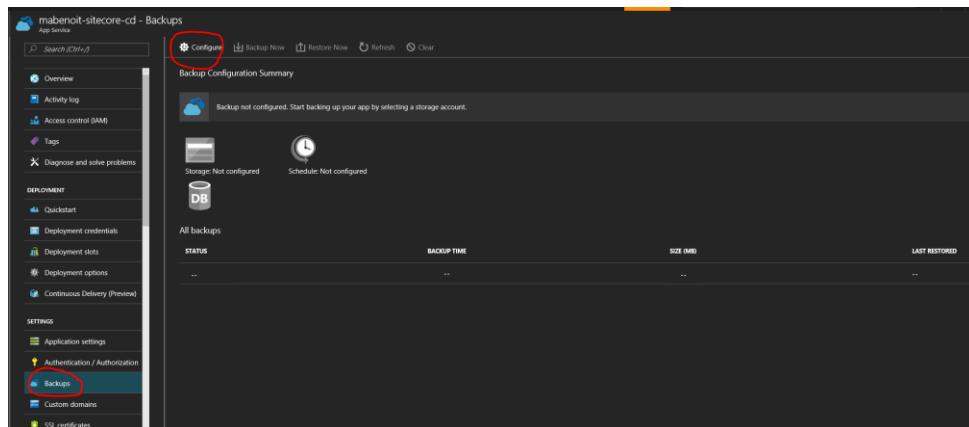
- Then, browse the CM website by playing/changing the IP address with yours and another to check that works properly.

**Note:** For more security and performance needs or concerns with the Azure Web App (App Service) you will need to consider the [App Service Environment \(ASE\)](#) as a Premium service plan option of Azure App Service.

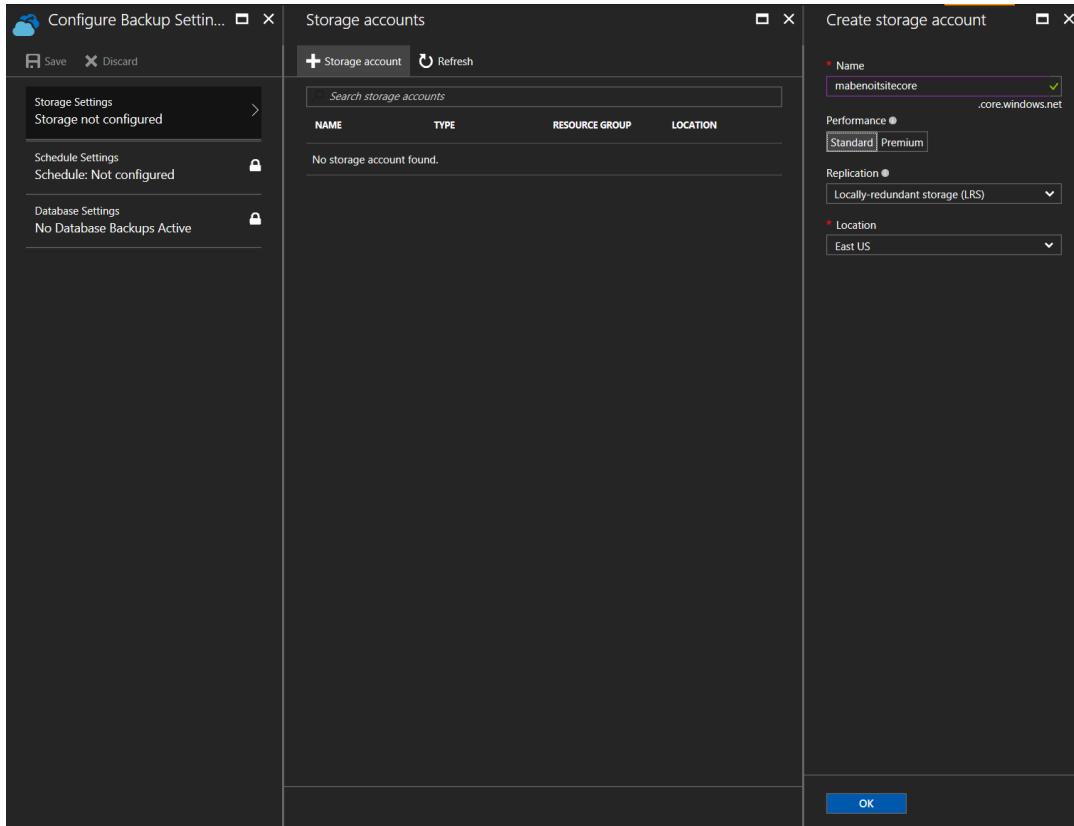
### Task 3: Setup Azure Web App Backup (10 min)

In this section, the attendee will configure a Backup configuration for the CD instance. Could be done as well for the CM, but for the purpose it will be for the CD only.

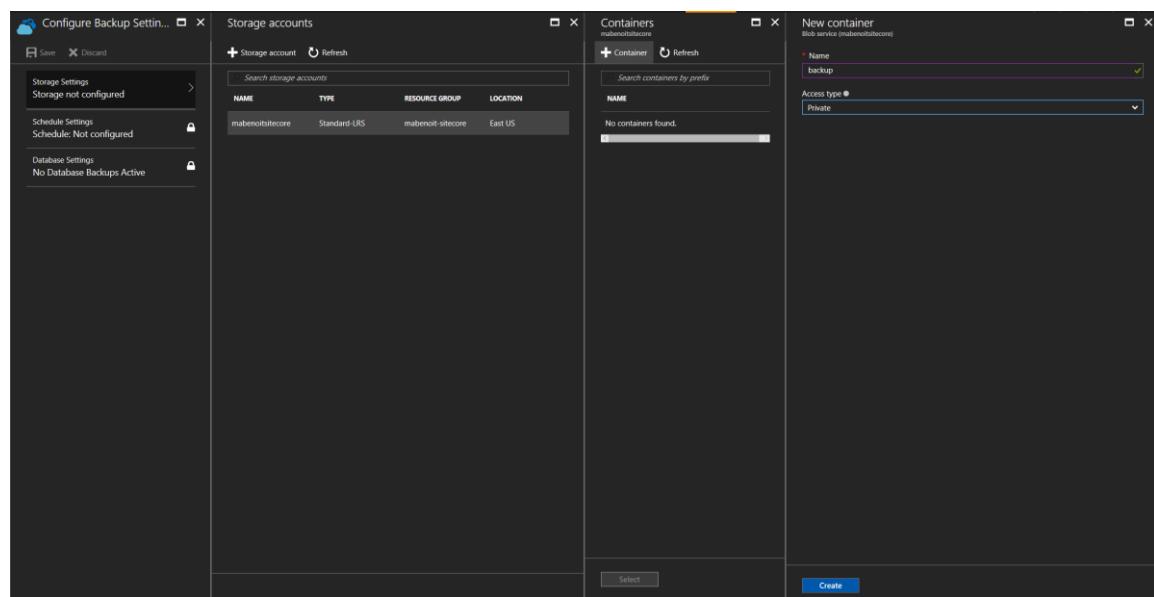
- On the Azure portal, go back to the "...-cd" Web App Service resource and there go on the "Backups" blade. And then click on the "Configure" toolbar button.



2. From there, you are invited to create a new storage account to store your backup files. You must provide a unique name (ensure the green checkmark appears) for the “Name” field and leave the other fields “as-is”. Then click on the “OK” button to submit the deployment.



3. Once the Storage account is provisioned it will appear in the list, select it, click on the “+ Container” toolbar button and set the “Name” field value as “backup”. Click on the “Create” button.



4. Then, select this “**backup**” container and click on the “**Select**” button.

The screenshot shows three windows side-by-side. The left window is titled "Configure Backup Settings" with tabs for "Save" and "Discard". The middle window is titled "Storage accounts" with tabs for "+ Storage account" and "Refresh". It lists one account: "mabenotsitecore" (Standard-LRS, mabenot-sitecore, East US). The right window is titled "Containers" with tabs for "+ Container" and "Refresh". It lists one container: "backup" (Mon Apr 17 2017 10:59:24 GMT-0400 (Eastern...)). A red circle highlights the "Select" button at the bottom of the right window.

5. Then, on the “**Configure Backup settings**” blade, you could click on the “**Save**” toolbar button.

The screenshot shows the "Configure Backup Settings" blade. The toolbar at the top has "Save" and "Discard" buttons. The "Save" button is highlighted with a red circle. Below the toolbar is a list of configuration items: "Storage Settings backup", "Schedule Settings Schedule: Not configured", and "Database Settings No Database Backups Active".

6. When you will get the “**Successfully saved Backup Configuration.**” Message, you could close this “**Configure Backup Settings**” blade.

7. On the “**Backups**” blade you are now able to click on the “**Backup now**” toolbar button.

The screenshot shows the Azure portal interface for managing backups. The main title bar indicates the resource path: Microsoft Azure > mabenot-sitecore-cd-hp - Apps > mabenot-sitecore-cd > mabenot-sitecore > mabenot-sitecore-cd - Backups. The left sidebar has sections for Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, DEPLOYMENT (Quickstart, Deployment credentials, Deployment slots, Deployment options, Continuous Delivery (Preview)), SETTINGS (Application settings, Authentication / Authorization), and Backups (which is circled in red). The main content area is titled 'Backup Configuration Summary' and includes a note: 'Backup is configured. Click Backup Now to manually start a backup or configure a schedule for automatic backups.' It shows two status icons: 'Storage: Configured' (green) and 'Schedule: Not configured' (grey). Below this is a table titled 'All backups' with columns: STATUS, BACKUP TIME, SIZE (MB), and LAST RESTORED. One row is listed with a status of 'Created' (indicated by a red circle), a backup time of '4/17/2017 11:09 AM', a size of '0', and '--' for last restored.

8. A new backup entry will appear with the “**Created**” status. We are not waiting for the end of the backup, let’s move forward with the next task below, we will use this backup file generated later on this lab.

**Note:** Here we just did a manual backup, but more advanced could be used and automated (via CLI or PowerShell). We could for example, restore a specific backup, setup a schedule for the backup (every night, etc.) and we could backup/restore the attached SQL Databases. You could find more information about this feature [here](#).

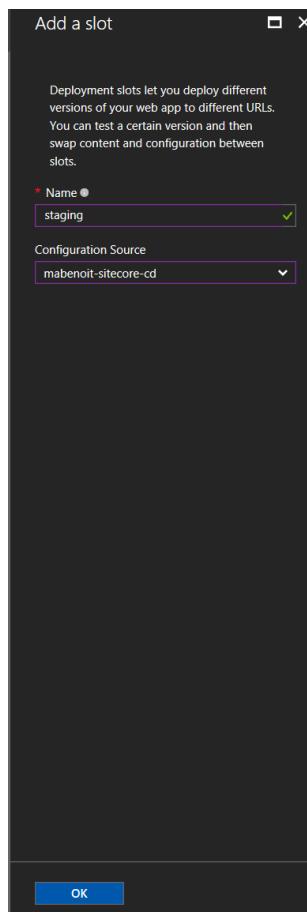
#### Task 4: Create a Staging Slot for the CD instance (15 min)

In this section, the attendee will create an empty “staging” Slot for the CD instance and then will restore the previous CD backup just made with the previous task to have a fresh Sitecore Vanilla instance. By using the “App Service Editor (Preview)” we will guarantee that the two web apps are different.

1. On the Azure portal, go back to the “...-cd” Web App Service resource and there go on the “Deployment slots” blade.

The screenshot shows the 'Deployment slots' section of the Azure App Service blade for the 'mabenoit-sitecore-cd' app. On the left, a sidebar lists various management options like Overview, Activity log, and Deployment slots. The 'Deployment slots' option is selected and highlighted in blue. The main area displays a table with columns for NAME, STATUS, and APP SERVICE PLAN. A message at the top states, 'You haven't added any deployment slots. Click ADD SLOT to get started.' A 'Swap' button is located above the table.

2. Click on the “Add Slot” toolbar button. Name the slot “**staging**” and on the “**Configuration Source**” field, select the “...-cd” web app. Click the “OK” button then.



3. Few seconds after, the “staging” slot should appear on the “Deployment slots” list.

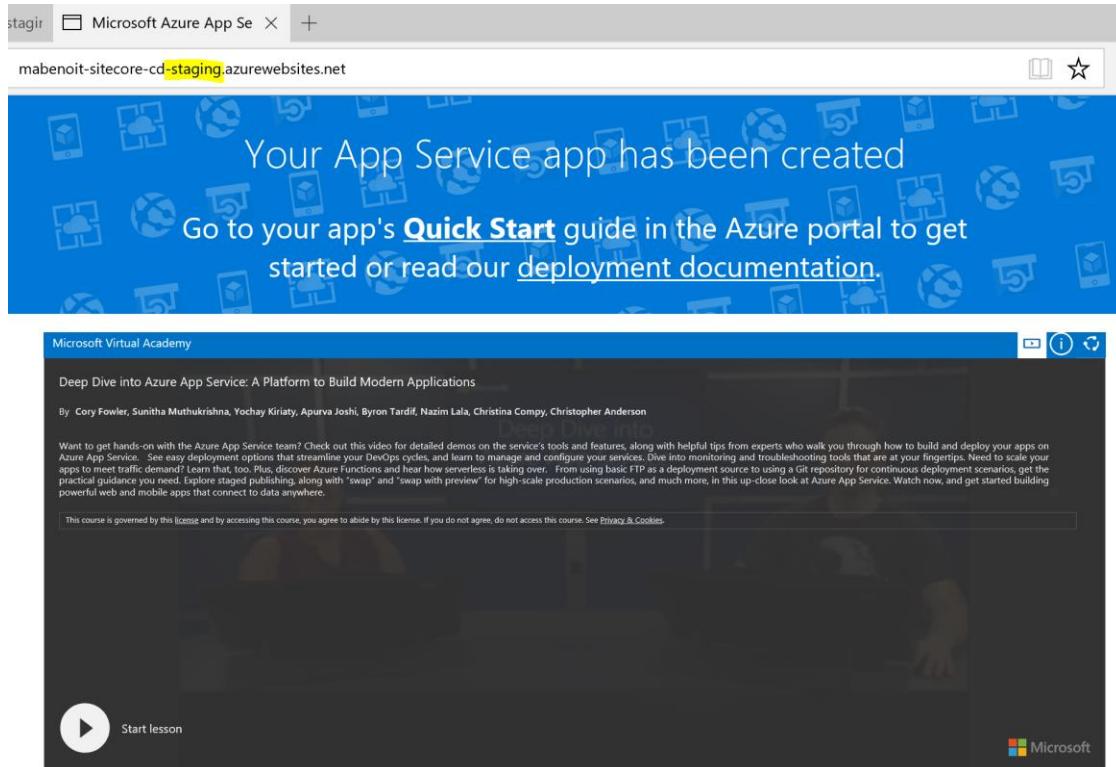
The screenshot shows the 'Deployment slots' blade for the 'mabenoit-sitecore-cd' app service. On the left, a sidebar lists various management options like Overview, Activity log, and Deployment slots. The main area displays a table with columns for NAME, STATUS, and APP SERVICE PLAN. A single row is present, representing the 'staging' slot, which is currently 'Running' on the 'mabenoit-sitecore-cd-hp' plan.

NAME	STATUS	APP SERVICE PLAN
mabenoit-sitecore-cd-staging	Running	mabenoit-sitecore-cd-hp

4. Click on it to open its associated resource blade:

The screenshot shows the resource blade for the 'staging' deployment slot. The left sidebar includes links for Overview, Activity log, and Deployment slots. The main content area has a toolbar with buttons for Browse, Stop, Swap, Restart, Delete, Get publish profile, and Reset publish profile. The 'Browse' button is circled in red. Below the toolbar, there's a purple banner with a link to a Quickstart guide. The 'Essentials' section provides basic information about the slot, including its resource group ('mabenoit-sitecore'), status ('Running'), location ('East US'), and subscription details. The 'Monitoring' section features a chart titled 'Requests and errors' showing request counts over time, with a sharp spike around 10:45 AM. At the bottom, two large numbers are displayed: 'HTTP SERVER ERRORS: 0' and 'REQUESTS: 220'.

5. Click on the “Browse” toolbar button to open the empty website (we have not yet deployed anything on it).



6. You could now open the Microsoft Azure Storage Explorer to see your new storage account, the new “backup” container and the files generated during the Azure WebApp backup process.

Name	Last Modified	Blob Type	Content Type	Size	Lease State
mabenoit-sitecore-cd_201704171509.log	Mon, 17 Apr 2017 15:37:26 GMT	Block Blob	application/octet-stream	257 B	
mabenoit-sitecore-cd_201704171509.xml	Mon, 17 Apr 2017 15:37:26 GMT	Block Blob	application/octet-stream	501 B	
mabenoit-sitecore-cd_201704171509.zip	Mon, 17 Apr 2017 15:37:26 GMT	Block Blob	application/octet-stream	265.6 MB	

7. Go to <https://gist.githubusercontent.com/mathieu-benoit/f3d6b1b2d330c4fe11dbf37b475745ec/raw/0fd2fa961b2fe970918cc2142050788048f230/db/Restore-WebApp-Backup-On-Slot.ps1>
8. Copy/paste the code as “Restore-WebApp-Backup-On-Slot.ps1” file into “C:\SitecoreLab\PowerShell”.
9. Right click on the “Restore-WebApp-Backup-On-Slot.ps1” file and select “Edit”.
10. The PowerShell ISE tool is now opened.

The screenshot shows the Windows PowerShell ISE interface. The top window displays the PowerShell script 'Restore-WebApp-Backup-On-Slot.ps1'. The bottom window shows the command being run in a PowerShell console:

```

PS C:\Users\mabenoit\Desktop> .\Restore-WebApp-Backup-On-Slot.ps1
cmdlet Restore-WebApp-Backup-On-Slot at command pipeline position 1
Supply values for the following parameters:
SubscriptionId: 0bd042ec-b6bc-4162-8385-c05a21d55572
ResourceGroupName: mabenoit-sitecore
webAppName: mabenoit-sitecore-cd
SlotName: staging
StorageAccountName: mabenouisitecore
ContainerName: backup
BlobFileName: mabenoit-sitecore-cd_201704171509.zip

```

11. On the PowerShell console at the bottom you could execute the following command:

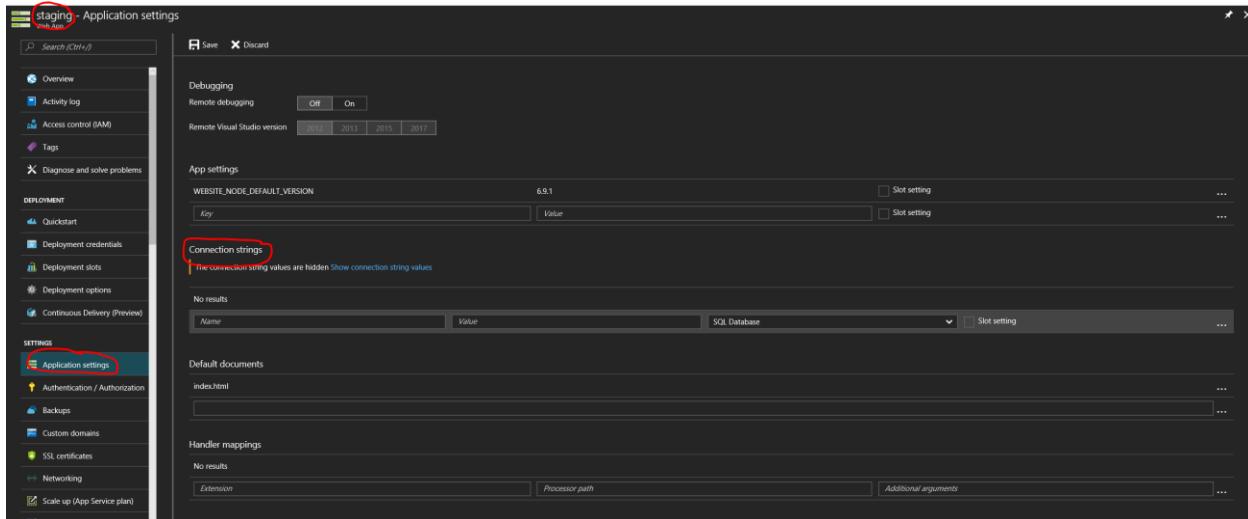
```
. \ Restore-WebApp-Backup-On-Slot.ps1
```

12. You will be prompted to enter the associated values of the following parameters:

- a. **SubscriptionId**: *associated subscription id.*
- b. **ResourceGroupName**: *associated resource group name.*
- c. **WebAppName**: *associated "...-cd" Azure WebApp name.*
- d. **SlotName**: *staging*
- e. **StorageAccountName**: *associated storage account name.*
- f. **ContainerName**: *backup*
- g. **BlobFileName**: *associated zip file name.*

13. This process will take ~10min. If you try to browse the associated website, you will get a 403 error.

14. By waiting the end of the backup restoration, go back to the Azure portal on the "**staging**" Azure WebApp Slot resource blade and from there, click on the "**Application settings**" blade.



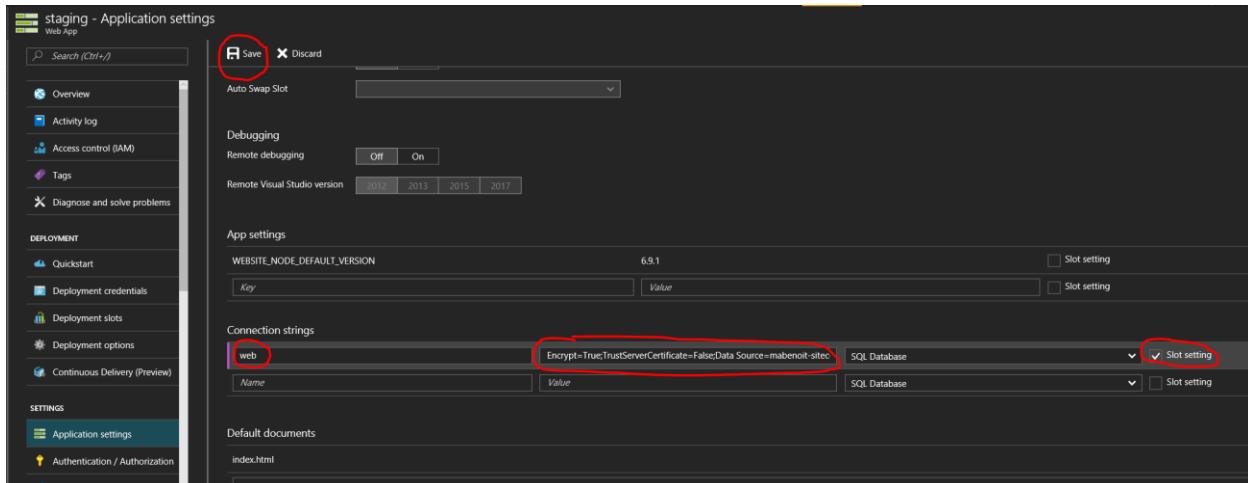
15. On the “**Connection strings**” section we would like to illustrate that you could use this place to set your connection strings – it’s a good practice while using Azure WebApp and Slot (it works as well for the section above for any “App settings”). Instead of having them in the web.config or in the App\_Config/ConnectionStringNames.config file, by *setting a key/value on this blade it will override the associated key of the .config files. So, for the example you could take the connection string of the Geo-Replicated database we setup earlier in this lab and set the associated entry here with:*

- a. **Name:** web
- b. **Value** (by replacing the):

```
Encrypt=True;TrustServerCertificate=False;Data
Source={your_prefix_token}-web-sec.database.windows.net,1433;Initial
Catalog={your_prefix_token}-web-db;User
Id=demouser;Password=demo@pass12345;
```

- c. **Type:** SQL Database
- d. **Slot setting:** checked

16. Then, click on the “**Save**” toolbar button.

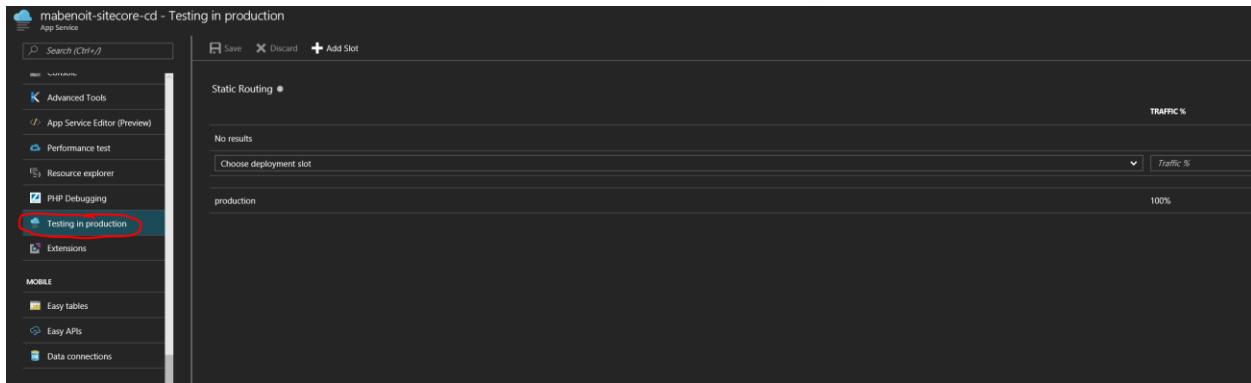


**Note:** You could perform another PowerShell command which could clone a WebApp into a Slot instead of restoring a WebApp backup in a Slot like we just did. The documentation could be found [here](#). But there is currently an issue like described [here](#) which will be fixed soon. Stay tuned!

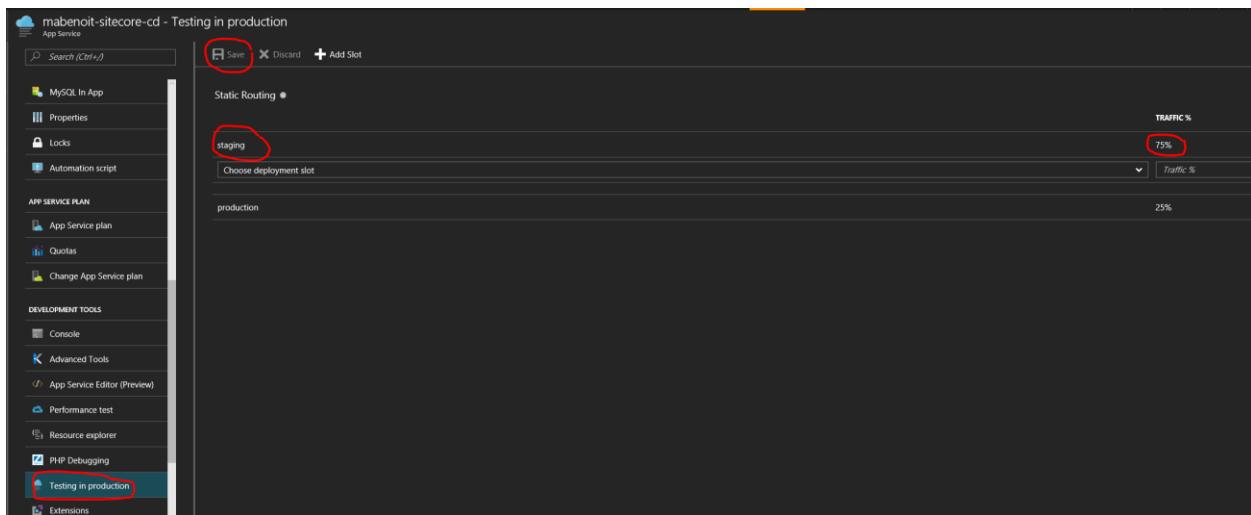
## Task 5: Testing in Production (5 min)

In this section, the attendee will use the Testing in Production feature with an Azure WebApp and its Slots and be able to set routing traffic rules.

1. Go to the “...-cd” Azure Web App resource blade and click on the “Testing in Production” blade.

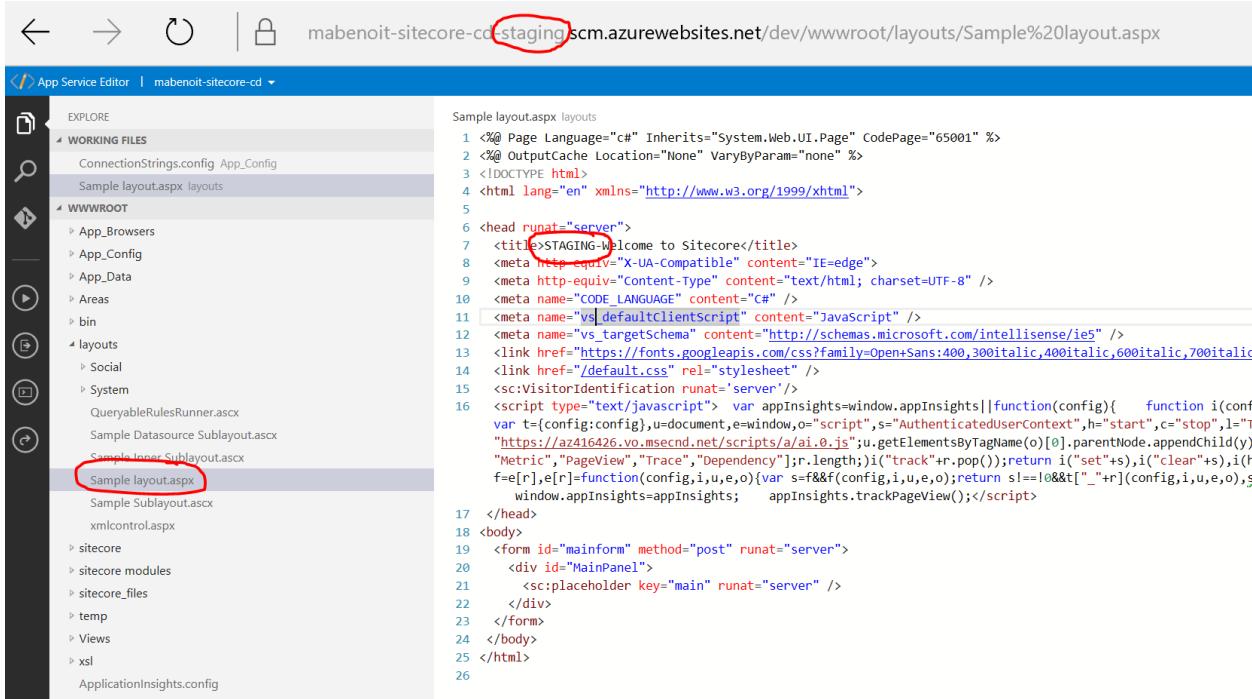


2. From there you could set the routing as:
  - a. **staging** = 75%
  - b. **production** = 25%



3. If you open a new web browser and navigate to the CD website URL, you will be redirected at 75% to the “**staging**” website (the URL is still the same).
4. You could now invert the value to have more traffic on the “**production**” site than the “**staging**” site. *You will need another web browser session to make sure you are not using the same session cookie “TiPMix” – or you could remove it.*

5. To make sure you are on the right site during these tests, you could edit the “**staging**” site by going to the “**App Service Editor (Preview)**” feature and open the “**wwwroot/layouts/Sample layouts.aspx**” file and add the “**STAGING-**” prefix to the **<title>** tag for example like illustrated below:

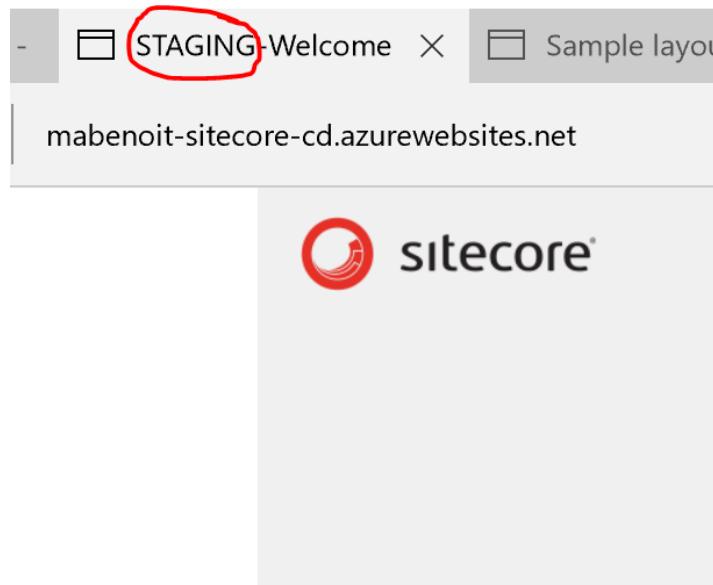


```

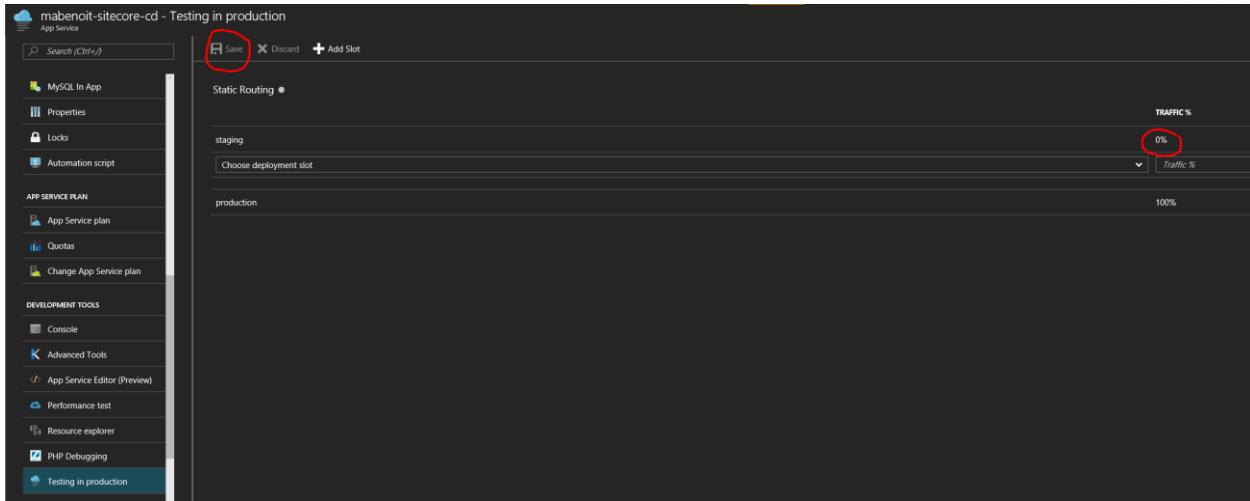
Sample layout.aspx layouts
1 <%@ Page Language="c#" Inherits="System.Web.UI.Page" CodePage="65001" %>
2 <%@ OutputCache Location="None" VaryByParam="none" %>
3 <!DOCTYPE html>
4 <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
5
6 <head runat="server">
7   <title>STAGING-Welcome to Sitecore</title>
8   <meta http-equiv="X-UA-Compatible" content="IE=edge">
9   <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
10  <meta name="CODE_LANGUAGE" content="C#" />
11  <meta name="vs_defaultClientScript" content="JavaScript" />
12  <meta name="vs_targetschema" content="http://schemas.microsoft.com/intellisense/ie5" />
13  <link href="https://fonts.googleapis.com/css?family=Open+Sans:400,300italic,400italic,600italic,700italic" rel="stylesheet" />
14  <link href="default.css" rel="stylesheet" />
15  <sc:visitorIdentification runat="server" />
16  <script type="text/javascript"> var appInsights=window.appInsights||function(config){ function i(config) {
var t=(config=config),u=document,e=window,o="script",s="AuthenticatedUserContext",h="start",c="stop",l="T
" "https://az416426.vo.msecnd.net/scripts/a/ai_.js";u.getElementsByTagName(o)[0].parentNode.appendChild(y)
"Metric","PageView","Trace","Dependency"];r.length;i("track"+r.pop());return i("set"+s),i("clear"+s),i(h
f=e[r],e[r]=function(config,i,u,e,o){var s=f&&f(config,i,u,e,o);return s!=!0&&t["_"+r](config,i,u,e,o),s
window.appInsights=appInsights; appInsights.trackPageView();</script>
17  </head>
18  <body>
19    <form id="mainform" method="post" runat="server">
20      <div id="MainPanel">
21        <sc:placeholder key="main" runat="server" />
22      </div>
23    </form>
24  </body>
25 </html>
26

```

6. With that, when hitting the “**staging**” site, you will see the title of the page like that:



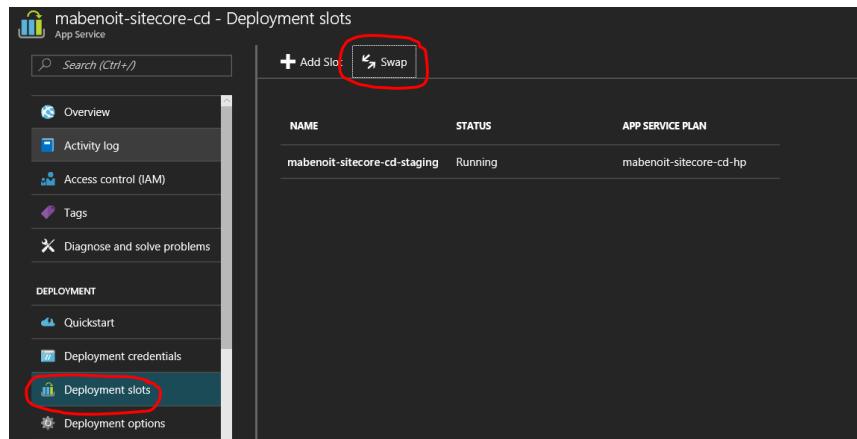
7. After these tests, you could remove the routing rules by setting the “**Traffic %**” of the “**staging**” slot to “**0**”.



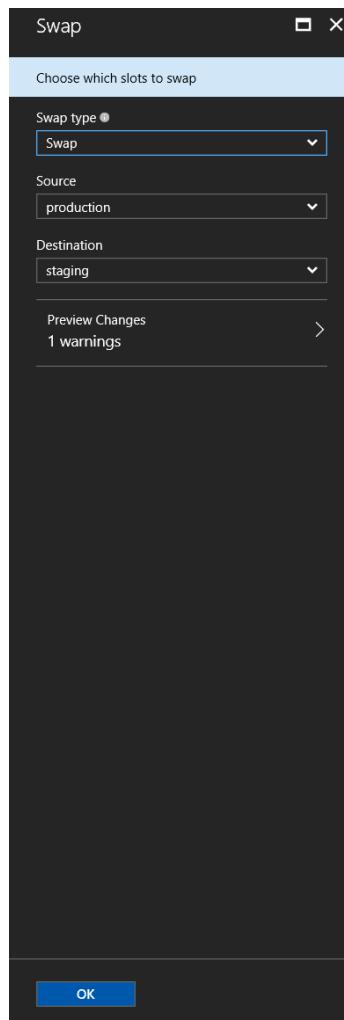
## Task 6: Slot Swap (5 min)

In this section, the attendee will use the Swap feature between an Azure WebApp and its Slots to be able to have successful deployments before going live in Production.

1. On the “...-cd” Azure WebApp resource blade, click on the “Deployment slots” blade. From there, click on the “Swap” toolbar button.



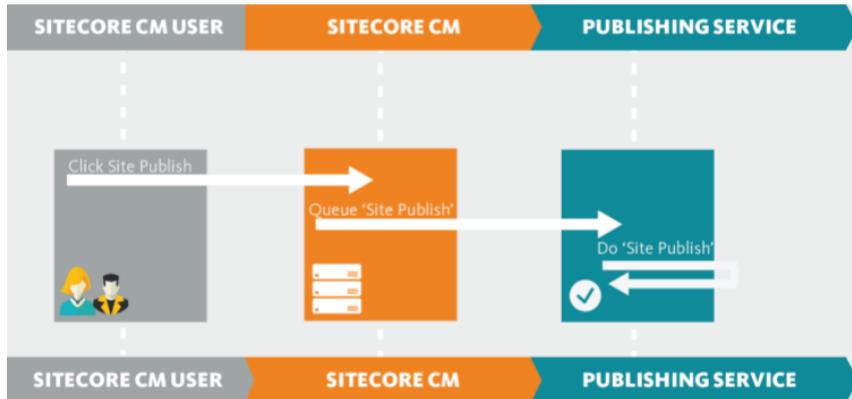
2. On the “Swap” blade, you could select your “Source” and your “Destination”; to actually perform the swap action you could click on the “OK” button. *For the purposes of this lab we are not doing and illustrating that, but be aware of this great feature for your own projects.*



**Note:** The swap with preview action applies slot specific configuration elements from the destination slot to the source slot and pauses until a selection is made to complete or cancel the swap action. The swap action directs destination slot's traffic to the source slot after source slot has been warmed up. You could have more details with the Swap with preview feature [here](#).

#### Task 7: Sitecore Publishing Services on Azure Web App (20 min)

In this section, the attendee will create a new Azure Web App to host the Sitecore Publishing Service instead of having this feature on the CM server.



*The Publishing Service module is an optional replacement for the existing Sitecore publishing methods. This module increases publishing throughput, reduces the amount of time spent publishing large volumes of items, and offers greater data consistency and reliability. The module also improves the user experience and provides better visual feedback to the user on the state of the publishing system.*

*The Publishing Service does not use any of the features, pipelines, and settings in the current publishing system. It is an entirely new way of publishing Sitecore items and media.*

*The Publishing Service runs a separate process to the Sitecore CM instance.*

Installation involves:

- Installation and configuration of the Publishing Service.
- Installation of the integration module package on your Sitecore instance. The integration module ensures that every publishing action, such as triggering a site publish, is handed on to the publishing service.

1. Download the Publishing Service and Module from this link:

[https://dev.sitecore.net/Downloads/Sitecore\\_Publishing\\_Service/20/Sitecore\\_Publishing\\_Service\\_2\\_0\\_Update1.aspx](https://dev.sitecore.net/Downloads/Sitecore_Publishing_Service/20/Sitecore_Publishing_Service_2_0_Update1.aspx)

➔ If you don't have a login to download them, please ask your Sitecore partner or rep or your lab proctor.

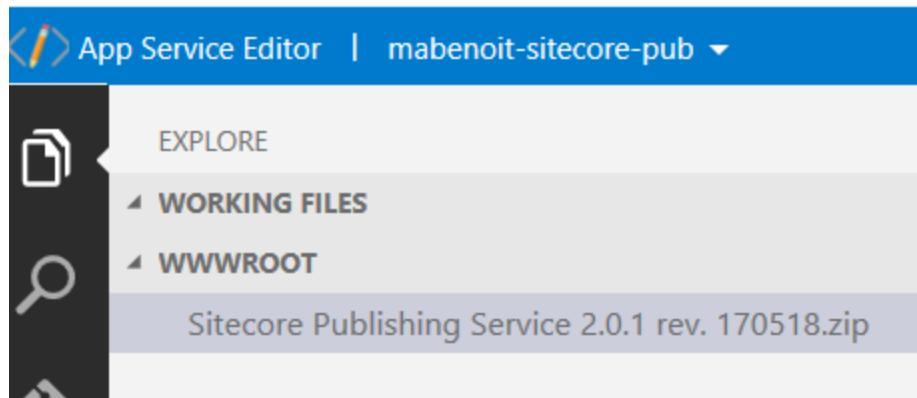
2. Go to <https://portal.azure.com/#create/Microsoft.WebSite>

3. Fill out all the fields by choosing the existing Resource Groups from the previous deployment and creating a new associated App Service Plan, like illustrated below:

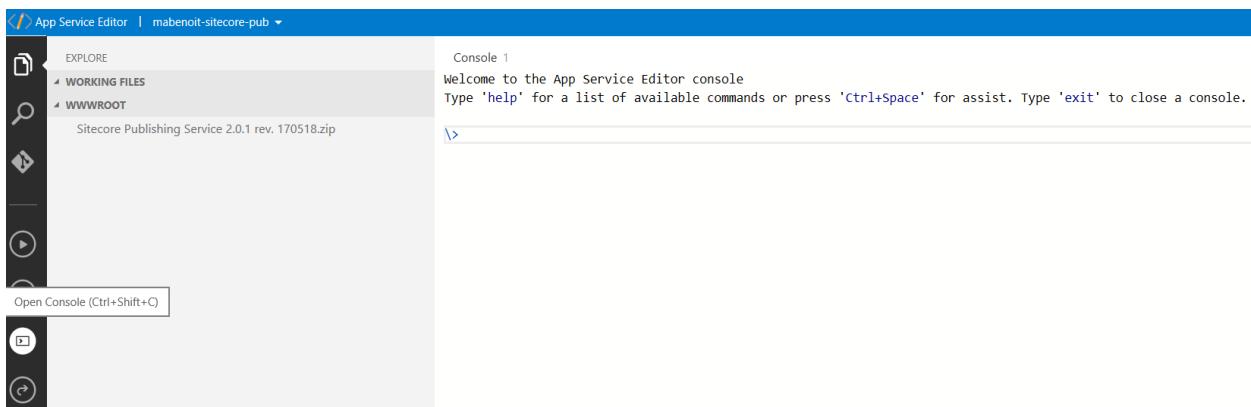
4. Click on the “Create” button. The new App Service Plan with the new Web App will be provisioned.
 

→ Remark: having a dedicated App Service Plan for this new server, will allow you to use Scale out and Scale up features without impacting other servers and take advantage of these features.
5. Once they are provisioned, open the associated Azure Web App just created click on the “App Service Editor (Preview)” blade.

6. Click on “Go →”
7. Now delete the **hostingstart.html** file and upload the publishing service zip file.

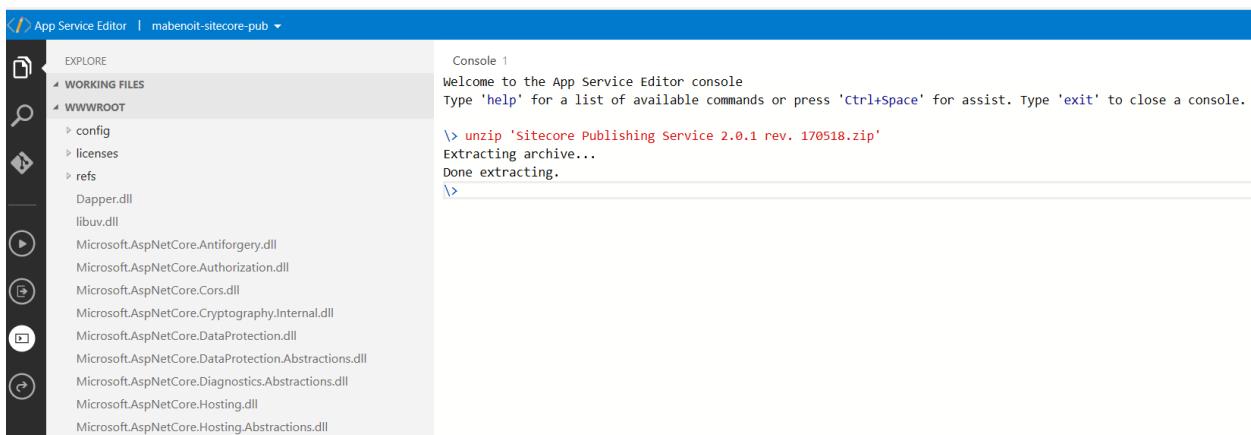


8. Open the associated **Console** like illustrated below:



9. Type the command below to unzip this archive file:

```
unzip 'Sitecore Publishing Service 2.0.1 rev. 170518.zip'
```



10. Now you will need to configure your connection strings. To do so, run the following command by replacing accordingly the {your\_prefix\_token} - parts:

```
Sitecore.Framework.Publishing.Host.exe configuration
setconnectionstring core
"Encrypt=True;TrustServerCertificate=False;Data
```

```

Source={your_prefix_token}-sql.database.windows.net,1433;Initial
Catalog={your_prefix_token}-core-db;User
Id=demouser;Password=demo@pass12345"

Sitecore.Framework.Publishing.Host.exe configuration
setconnectionstring web
"Encrypt=True;TrustServerCertificate=False;Data
Source={your_prefix_token}-web-sql.database.windows.net,1433;Initial
Catalog={your_prefix_token}-web-db;User
Id=demouser;Password=demo@pass12345"

Sitecore.Framework.Publishing.Host.exe configuration
setconnectionstring master
"Encrypt=True;TrustServerCertificate=False;Data
Source={your_prefix_token}-sql.database.windows.net,1433;Initial
Catalog={your_prefix_token}-master-db;User
Id=demouser;Password=demo@pass12345"

```

```

App Service Editor | mabenoit-sitecore-pub
EXPLORE WORKING FILES WWWROOT config
  sc.publishing.sqlazure.xml sc.publishing.sqlazure.connections.xml.example
  sc.connectionsstrings.json sc.global.xml.example
  Dapper.dll libuv.dll Microsoft.AspNetCore.Antiforgery.dll Microsoft.AspNetCore.Authorization.dll Microsoft.AspNetCore.Cors.dll
Console
  > Sitecore.Framework.Publishing.Host.exe configuration setconnectionstring master "Encrypt=True;TrustServerCertificate=False;Data Source=mabenoit-sitecore-sql.database.windows.net,1433;Initial Catalog=mabenoit-sitecore-master-db;User Id=demouser;Password=demo@pass12345;" [01:03:22 INF] Sitecore Set Connection String [01:03:22 INF] Key : Sitecore:Publishing:ConnectionStrings:master [01:03:22 INF] Value : Encrypt=True;TrustServerCertificate=False;Data Source=mabenoit-sitecore-sql.database.windows.net,1433;Initial Catalog=mabenoit-sitecore-master-db;User Id=demouser;Password=demo@pass12345; [01:03:22 INF] Environment : global [01:03:22 INF] File : sc.connectionstrings.json
  > Sitecore.Framework.Publishing.Host.exe configuration setconnectionstring core "Encrypt=True;TrustServerCertificate=False;Data Source=mabenoit-sitecore-sql.database.windows.net,1433;Initial Catalog=mabenoit-sitecore-core-db;User Id=demouser;Password=demo@pass12345;" [01:03:47 INF] Sitecore Set Connection String [01:03:48 INF] Key : Sitecore:Publishing:ConnectionStrings:core [01:03:48 INF] Value : Encrypt=True;TrustServerCertificate=False;Data Source=mabenoit-sitecore-sql.database.windows.net,1433;Initial Catalog=mabenoit-sitecore-core-db;User Id=demouser;Password=demo@pass12345; [01:03:48 INF] Environment : global [01:03:48 INF] File : sc.connectionstrings.json
  > Sitecore.Framework.Publishing.Host.exe configuration setconnectionstring web "Encrypt=True;TrustServerCertificate=False;Data Source=mabenoit-sitecore-web-sql.database.windows.net,1433;Initial Catalog=mabenoit-sitecore-web-db;User Id=demouser;Password=demo@pass12345;" [01:05:06 INF] Sitecore Set Connection String [01:05:06 INF] Key : Sitecore:Publishing:ConnectionStrings:web [01:05:06 INF] Value : Encrypt=True;TrustServerCertificate=False;Data Source=mabenoit-sitecore-web-sql.database.windows.net,1433;Initial Catalog=mabenoit-sitecore-web-db;User Id=demouser;Password=demo@pass12345; [01:05:06 INF] Environment : global [01:05:06 INF] File : sc.connectionstrings.json

```

11. Now you are ready to upgrade the database schema by running the following command:

```
Sitecore.Framework.Publishing.Host.exe schema upgrade -f
```

```

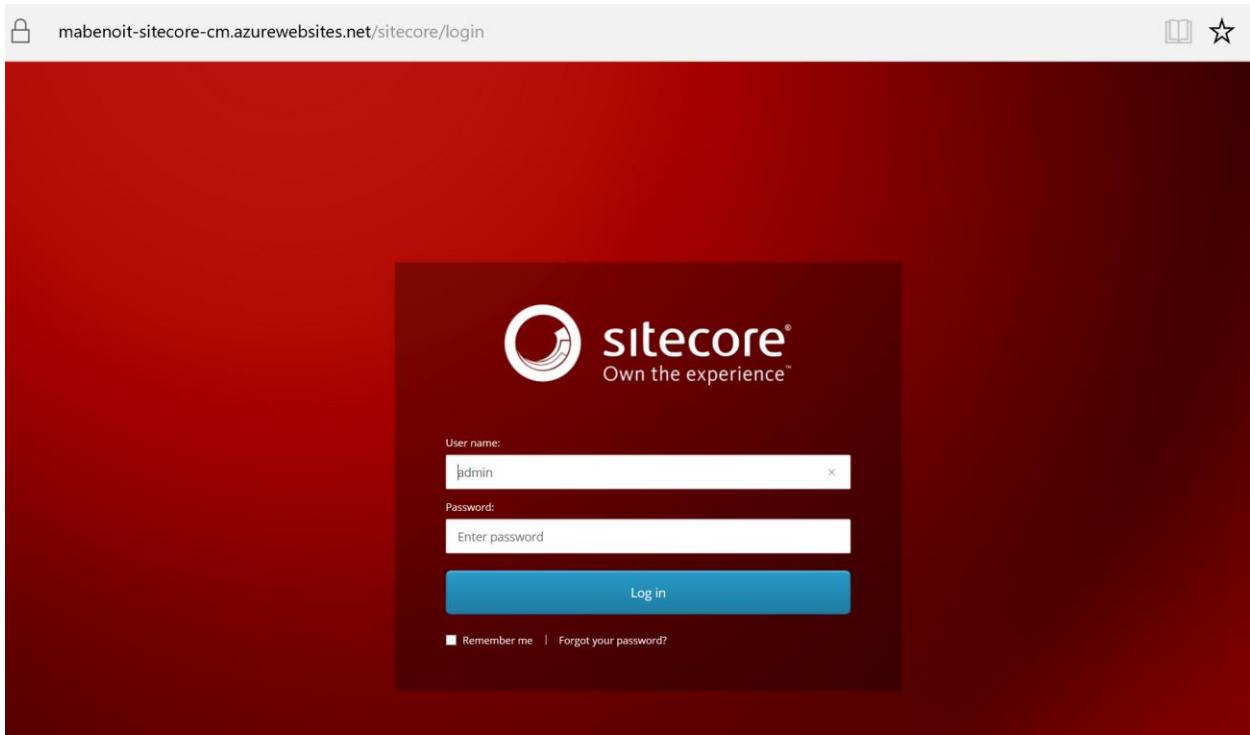
App Service Editor | mabenoit-sitecore-pub
EXPLORE WORKING FILES WWWROOT config
  sc.publishing.sqlazure.xml sc.publishing.sqlazure.connections.xml.example
  sc.connectionsstrings.json sc.global.xml.example
  Dapper.dll libuv.dll Microsoft.AspNetCore.Antiforgery.dll Microsoft.AspNetCore.Authorization.dll Microsoft.AspNetCore.Cors.dll
Console
  > Sitecore.Framework.Publishing.Host.exe schema upgrade -f [01:05:17 INF] Schema Upgrade [01:05:19 INF] Upgrading all databases to version [ 2 ]
[01:05:20 INF] Database: [ mabenoit-sitecore-sql.database.windows.net,1433|mabenoit-sitecore-master-db ] ... COMPLETE [ v0 => v2 ] [01:05:20 INF] Database: [ mabenoit-sitecore-sql.database.windows.net,1433|mabenoit-sitecore-core-db ] ... COMPLETE [ v0 => v2 ] [01:05:20 INF] Database: [ mabenoit-sitecore-web-sql.database.windows.net,1433|mabenoit-sitecore-web-db ] ... COMPLETE [ v0 => v2 ]

```

12. Now the service is ready. To test your service, try to hit this url: <http://<service host or domain name>/api/publishing/maintenance/status>. The service should return status:0 in json response.

Now we would like to install the Sitecore Publishing Module on the Sitecore CM instance.

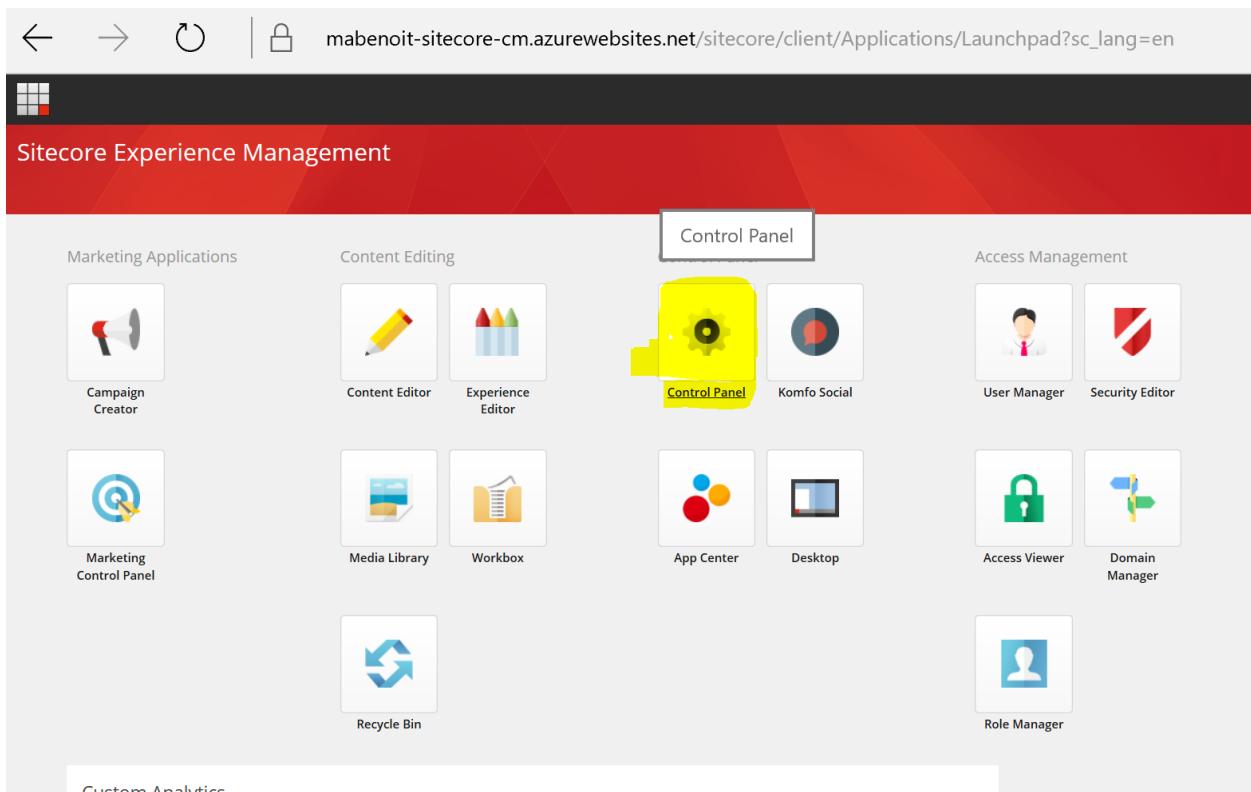
13. Go to your Sitecore CM URL by adding at the end of this URL “/sitecore”.



14. Login with these credentials:

- ➔ Login: admin
- ➔ Password: demo@pass12345

15. Once logged in, click on Control Panel.

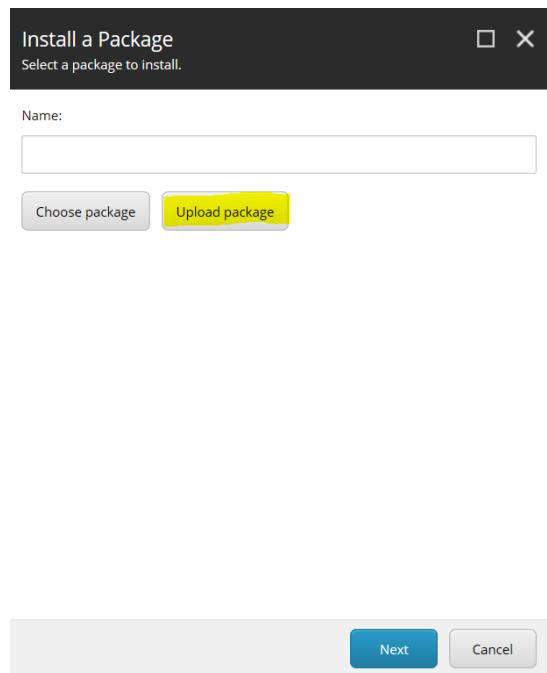


16. Click on “Install package”

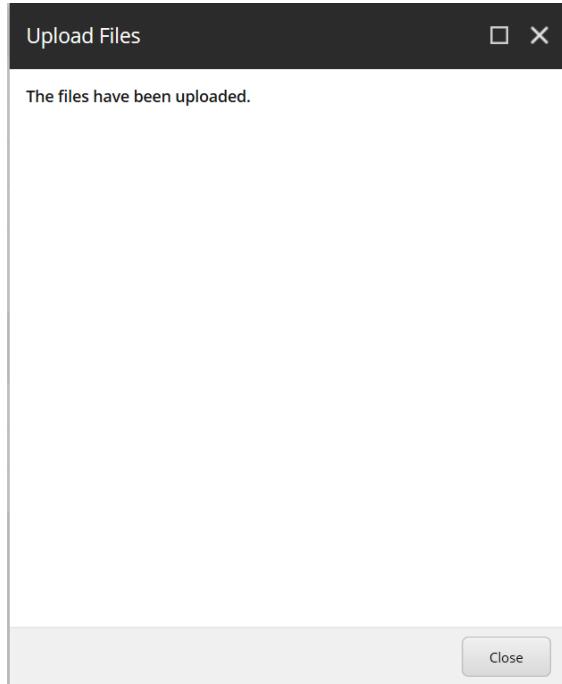
The screenshot shows the Sitecore Control Panel interface. At the top, there are navigation icons (Back, Forward, Refresh) and a lock icon. The URL in the address bar is [mabenoit-sitecore-cm.azurewebsites.net/sitecore/client/Applications/ControlPanel.aspx?sc\\_bw=1](https://mabenoit-sitecore-cm.azurewebsites.net/sitecore/client/Applications/ControlPanel.aspx?sc_bw=1). Below the header, there's a red banner with the text "Control Panel". A "Back" button is visible. The main content area is divided into several sections:

- MY SETTINGS**: Includes links for Change desktop background, Change Application options, Change password, Change personal information, Region and language options, and Reset to default settings.
- DATABASE**: Includes links for Rebuild link databases, Move an item to another database, Clean up databases, and Display database usage.
- LOCALIZATION**: Includes links for Export languages, Import languages, Add a new language, and Delete a language.
- ADMINISTRATION**: Includes links for Administration tools, License details, Installed licenses, and a yellow-highlighted "Install a package" button. It also lists "Install an update".
- REPORTS**: Includes links for Scan the database for broken links and Scan the database for untranslated fields.
- SECURITY**: Includes links for User manager and Role manager.
- INDEXING**: Includes links for Generate the Solr Schema.xml file and Indexing manager.

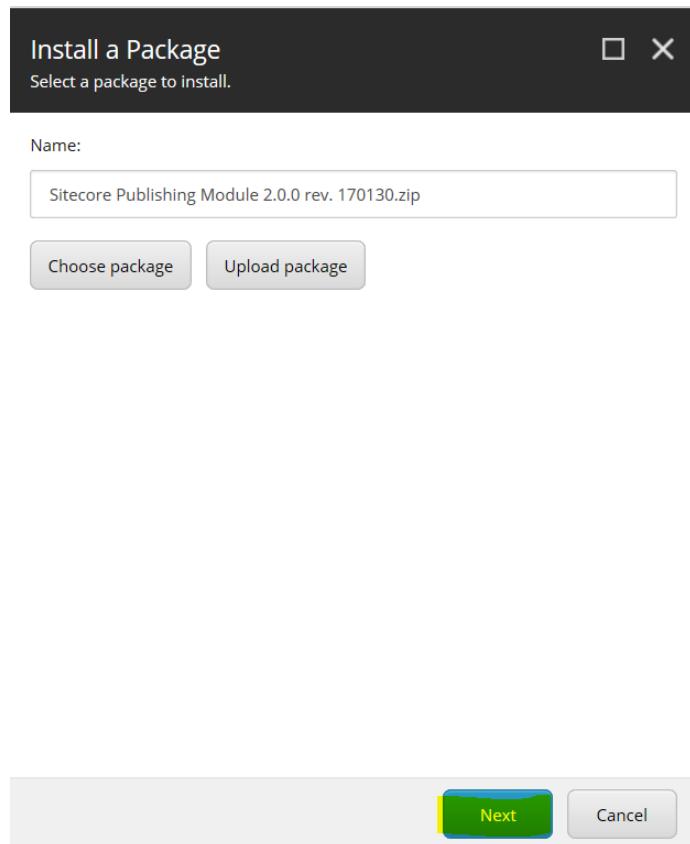
17. On the “Install a package” dialog, choose “Upload package”.



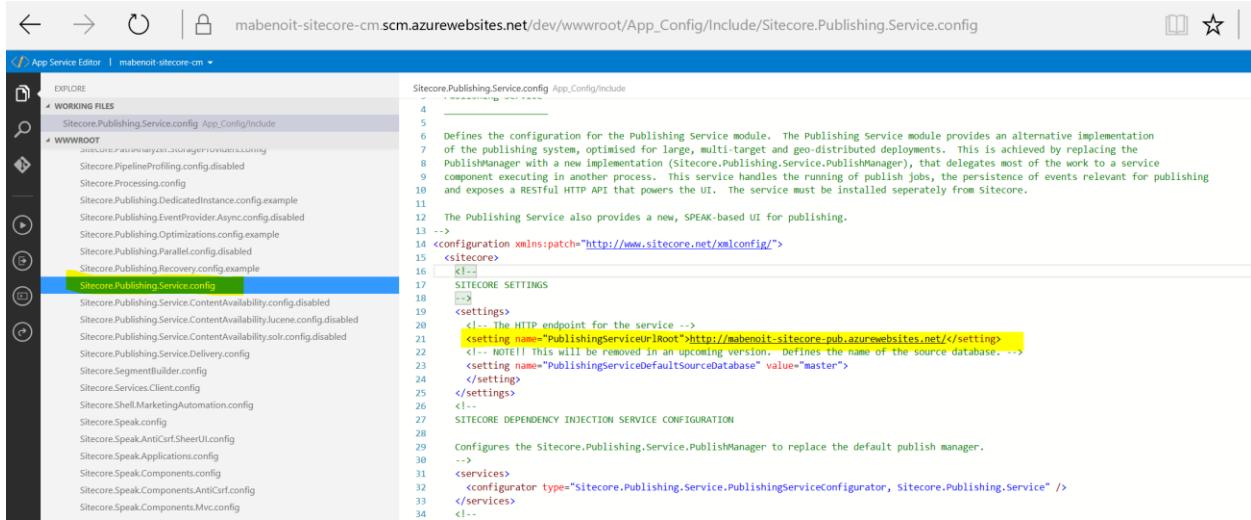
18. Select your local “**Sitecore Publishing Module 2.0.1 rev. 170518.zip**”, few seconds after the confirmation dialog should appear:



19. Click “**Close**” and you are back on the first page of the “**Install a Package**” wizard, click on “**Next**”.



20. Then, accept the terms, click on “**Next**” and finally on “**Install**”.
21. After the package is successfully installed, open a new Internet browser tab and navigate to the “**App Service Editor (Preview)**” of your CM instance - [https://\[your url\]-cm.scm.azurewebsites.net/dev/](https://[your url]-cm.scm.azurewebsites.net/dev/). And open the `wwwroot/App_Config/Include/Sitecore.PUBLISHING.Service.config` file.



```

<configuration xmlns:patch="http://www.sitecore.net/xmlconfig/">
  <sitecore>
    <!-- SITECORE SETTINGS -->
    <settings>
      <!-- The HTTP endpoint for the service -->
      <setting name="PublishingServiceUrlRoot" value="http://mabenot-sitecore-pub.azurewebsites.net/" />
      <!-- NOTE!! This will be removed in an upcoming version... Defines the name of the source database. -->
      <setting name="PublishingServiceDefaultSourceDatabase" value="master" />
    </settings>
    <!-- SITECORE DEPENDENCY INJECTION SERVICE CONFIGURATION -->
    <services>
      <!-- Configures the Sitecore.PUBLISHING.Service.PublishManager to replace the default publish manager. -->
      <service type="Sitecore.PUBLISHING.Service.PublishManager" />
      <!-- Sitecore.PUBLISHING.Service.PUBLISHINGServiceConfigurator -->
      <configurator type="Sitecore.PUBLISHING.Service.PUBLISHINGServiceConfigurator, Sitecore.PUBLISHING.Service" />
    </services>
  </sitecore>
</configuration>

```

22. Here, you must change the value of the “`PublishingServiceUrlRoot`” setting by the root URL of your Publishing server just configured few steps above. **Important: Make sure that the URL ends with a trailing slash ‘/’ and that it is formatted correctly.**
23. The new Publishing server is ready to use, you could try a quick update on the Homepage from your Content Editor and Publish the item to play with the new UI of this Module.

## Takeaways

You could find more advanced settings for the Sitecore Publishing Service in the official associated installation guide [here](#).

Once hosted on an Azure Web Apps, you could easily increase the number of instances by using the “Scale out” features with the associated App Service Plan.

We would like to highlight [this great blog post](#) about the benefits to use this Sitecore Publishing Service regarding the huge performance gain.

From the community, [Zero downtime deployments with Sitecore on Azure](#).

## Exercise 5 - Azure Search and Application Insights (35 min)

### Objectives

The goal of this exercise is to be familiar with the Application Insights and Azure Search and see the key features.

Through this exercise, you will play/use with:

- **Azure portal**
- **Application Insights**
- **Azure Search**
- **Sitecore Content Editor**
- **Azure PowerShell**



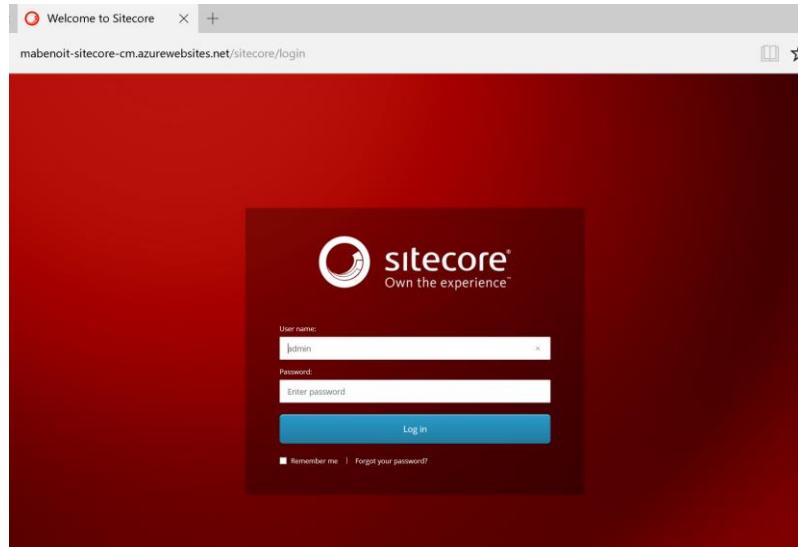
### Task 1: Azure Search (15 min)

In this section, the attendee will add 3 pages using the Sitecore Content Editor, will rebuild the master index and then will do some queries with the Azure Search Explorer using the Microsoft Azure Portal.

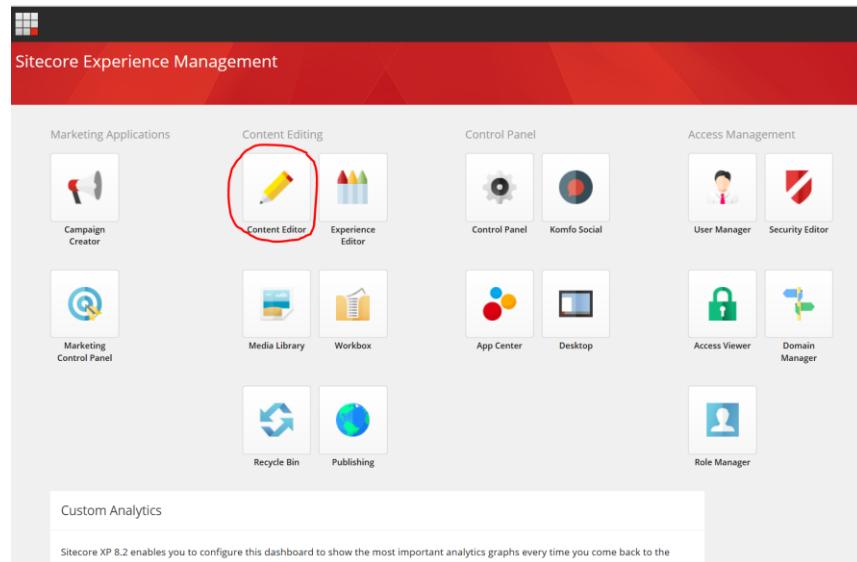
1. Go to <https://portal.azure.com/> and select your “...-cm” Azure Web App.

The screenshot shows the Microsoft Azure portal interface. On the left, there's a sidebar with various icons and a list of services. The main area shows the 'Overview' blade for an Azure Web App named 'mabenoit-sitecore-cm'. The 'Overview' link in the sidebar and the 'Browse' button in the top toolbar are both highlighted with red circles. The blade itself shows details like the resource group ('mabenoit-sitecore'), status ('Running'), location ('East US'), and subscription information. Below this is a 'Monitoring' section with a chart showing 'Requests and errors' over time, with a value of 126 requests at the end of the chart.

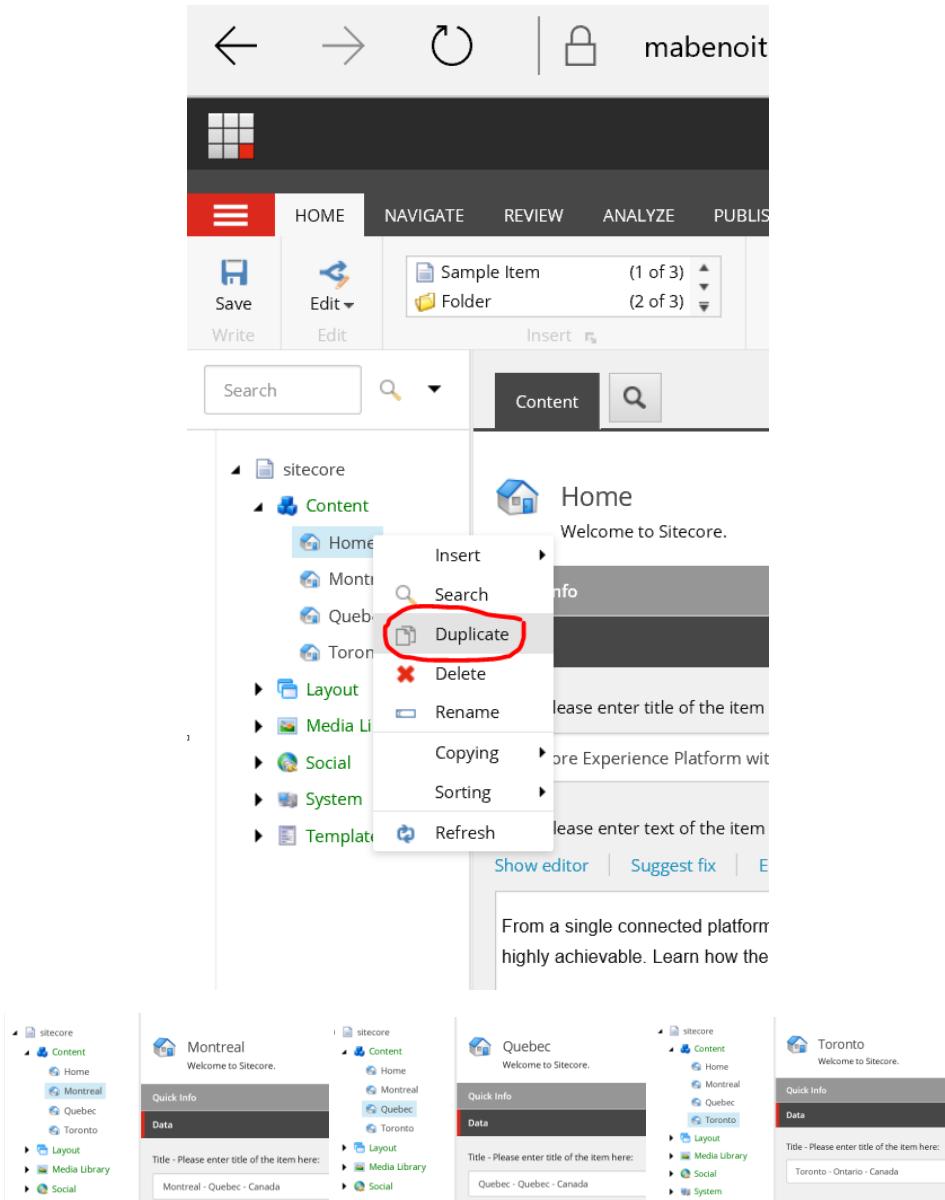
2. On the “Overview” blade, click on the “Browse” toolbar action.
3. There, login with the following credentials to the “/sitecore” page.
  - a. **User name:** admin
  - b. **Password:** demo@pass12345



4. On the Sitecore Launchpad page, click on "**Content Editor**".



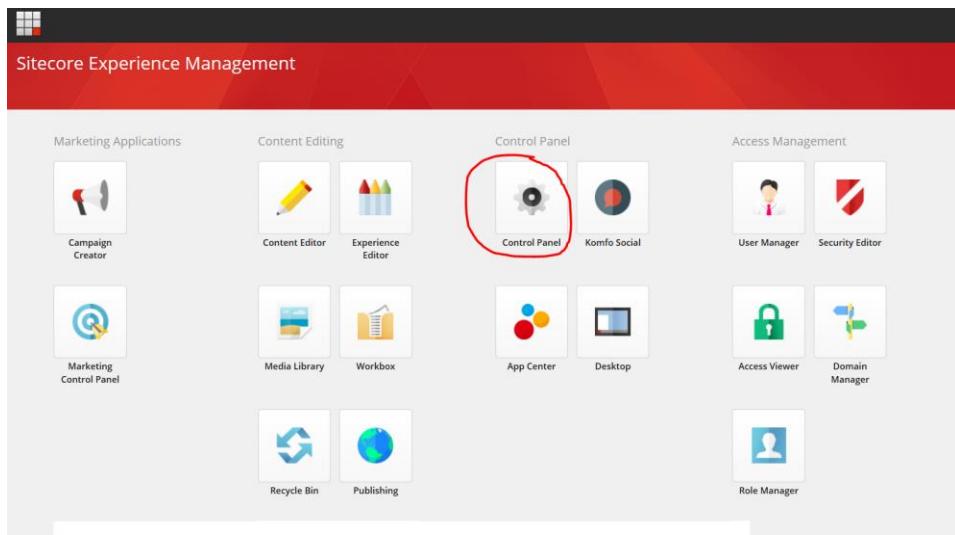
5. By default, the "**Home**" Sitecore item will be selected. Right-click on it and select "**Duplicate**" three times to add these 3 Sitecore items:
- Montreal - the field "**Title**" with "Montreal – Quebec – Canada" as value.
  - Quebec – the field "**Title**" with "Quebec – Quebec – Canada" as value.
  - Toronto – the field "**Title**" with "Toronto – Ontario – Canada" as value.



6. Save all these changes and then click on the top left icon to go to the Sitecore Launchpad page.



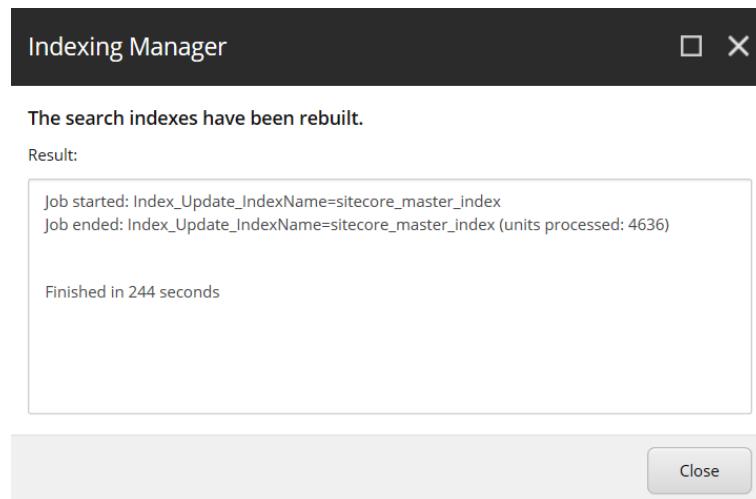
7. There, click on “**Control Panel**”.



8. From the “Control Panel” page, click on “Indexing Manager”. On the dialog box prompted select the “sitecore\_master\_index” and then click on the “Rebuild” button.

The screenshot shows the Sitecore Control Panel interface. On the left, there are sections for MY SETTINGS, ADMINISTRATION, and INDEXING. In the INDEXING section, the "Indexing manager" link is circled in red. A modal dialog box titled "Indexing Manager" is open, prompting the user to "Select the search indexes that you want to rebuild". Inside the dialog, under "Rebuild search index", the "sitecore\_master\_index" checkbox is checked and circled in red. To the right, "Index statistics" for several indexes are listed, including "sitecore\_core\_index" and "sitecore\_master\_index". At the bottom right of the dialog, the "Rebuild" button is circled in red.

9. After ~4 min the associated index will be rebuilt on the Azure Search service. While the index is rebuilding you could move forward with the following steps in parallel.

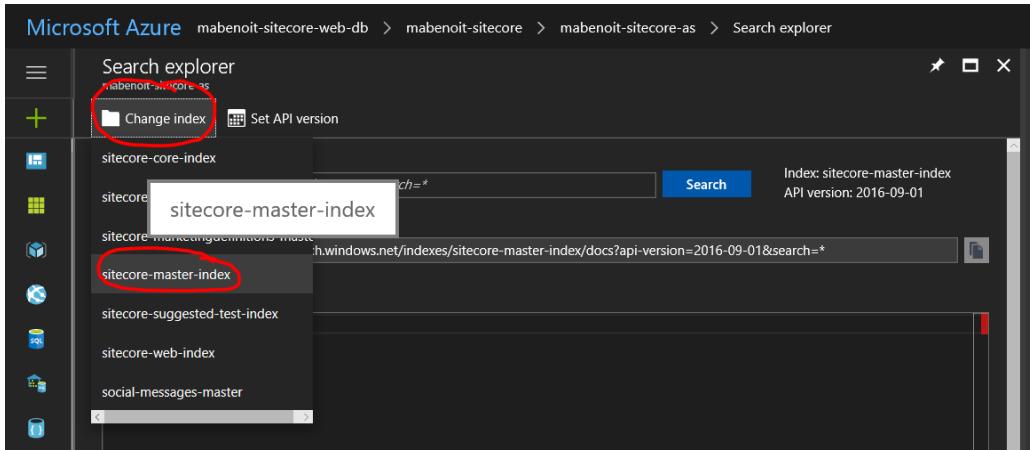


10. Go to the Azure portal <https://portal.azure.com> and navigate to the Azure Search service provisioned earlier in this lab.

The screenshot shows the Azure portal interface for the 'mabenot-sitecore-as' search service. The 'Search explorer' button in the toolbar is circled in red. The 'Indexes' section lists several indexes with their document counts and storage sizes. The 'Usage' section displays current document count (86,687) and storage size (114.4 MB) against thresholds (15,000,000 and 25 GB). The 'Scale' section shows 1 replica, 1 partition, and 1 search unit. A callout box in the 'Indexers' section says 'Create an indexer to get started'. The 'Data sources' section shows 'No data sources found'.

NAME	DOCUMENT COUNT	STORAGE SIZE
sitecore-core-index	69,640	84.55 MB
sitecore-list-index	1	20.29 KB
sitecore-marketingdefinitions-master	36	412.19 KB
sitecore-master-index	17,001	29.28 MB
sitecore-suggested-test-index	4	49.48 KB

11. On the “Overview” blade, click on the “Search explorer” toolbar button.
12. On the “Search explorer”, “Change index” to “sitecore-master-index”.



13. In the “**Query string**” search box copy/paste the following string:

```
search=quebec&$filter=parsedlanguage eq 'english'&$select=name__,
fullpath_1, title
```

14. By clicking on the “**Search**” button you should get the following result:

Rank	Score	Name	Full Path	Title
1	1.7034332	Quebec	/sitecore/content/quebec	Quebec - Quebec - Canada
2	0.22160262	Montreal	/sitecore/content/montreal	Montreal - Quebec - Canada

15. You could repeat this action by changing the value of the “**Query string**” search box by:

```
search=canada&$filter=parsedlanguage eq 'english'&$select=name__,
fullpath_1, title
```

16. By clicking on the “**Search**” button you should get the following result:

The screenshot shows the Microsoft Azure Search Explorer interface. At the top, it displays the URL `https://mabenoit-sitecore-as.search.windows.net/indexes/sitecore-master-index/docs?api-version=2016-09-01&search=canada&%24filter=pa`. Below the URL, the results pane shows a JSON array of documents. The first document is for 'Canada', followed by 'Quebec', 'Toronto', and 'Montreal'. Each document includes its score, name, full path, and title.

```

1 [
2   "@odata.context": "https://mabenoit-sitecore-as.search.windows.net/indexes('sitecore-master-index')/$metadata#docs(name_,fullpath_1,title)",
3   "value": [
4     {
5       "@search.score": 1.1946371,
6       "name_": "Canada",
7       "fullpath_1": "/sitecore/system/settings/Analytics/lookups/countries/canada",
8       "title": null
9     },
10    {
11      "@search.score": 0.22225805,
12      "name_": "Quebec",
13      "fullpath_1": "/sitecore/content/quebec",
14      "title": "Quebec - Quebec - Canada"
15    },
16    {
17      "@search.score": 0.22171894,
18      "name_": "Toronto",
19      "fullpath_1": "/sitecore/content/toronto",
20      "title": "Toronto - Ontario - Canada"
21    },
22    {
23      "@search.score": 0.22131284,
24      "name_": "Montreal",
25      "fullpath_1": "/sitecore/content/montreal",
26      "title": "Montreal - Quebec - Canada"
27    }
28  ]
29 ]

```

## Task 2: Application Insights (20 min)

In this section, the attendee will go through the different powerful features of Application Insights using the Microsoft Azure Portal and will add the Application Insight JavaScript SDK on the CD instance. By default, Sitecore collects logs via Application Insights on the server-side, here we will be able to complete that by collecting client-side information as well.

1. Go to <https://portal.azure.com/> and select your "...-ai" Application Insights resource.

Microsoft Azure mabenoit-sitecore-web-db > mabenoit-sitecore > mabenoit-sitecore-ai

mabenoit-sitecore-ai Application Insights

Search (Ctrl+F)

Overview Activity log Access control (IAM) Tags

USAGE Usage

INVESTIGATE Application map Smart Detection Live Metrics Stream Metrics Explorer Availability Failures Performance Servers Browser

CONFIGURE Getting started Properties Alerts Smart Detection settings Features + pricing

Search Metrics Explorer Analytics Time range Refresh More

Click here to see a map of KPIs for your app components and dependencies.

Resource group (change)  
mabenoit-sitecore

Type  
ASP.NET

Location  
East US

Instrumentation Key  
3D0C67...  
Subscription name (change)  
Mabenoit-Sitecore - Internal Consumption

Subscription ID  
C1174e6d-11f6-43c5-8005-000000000000

Live Stream 0 Alerts 8 Servers Smart Detection 0 Detections (24h) Availability App map

Health

Overview timeline MABENOIT-SITECORE-AI

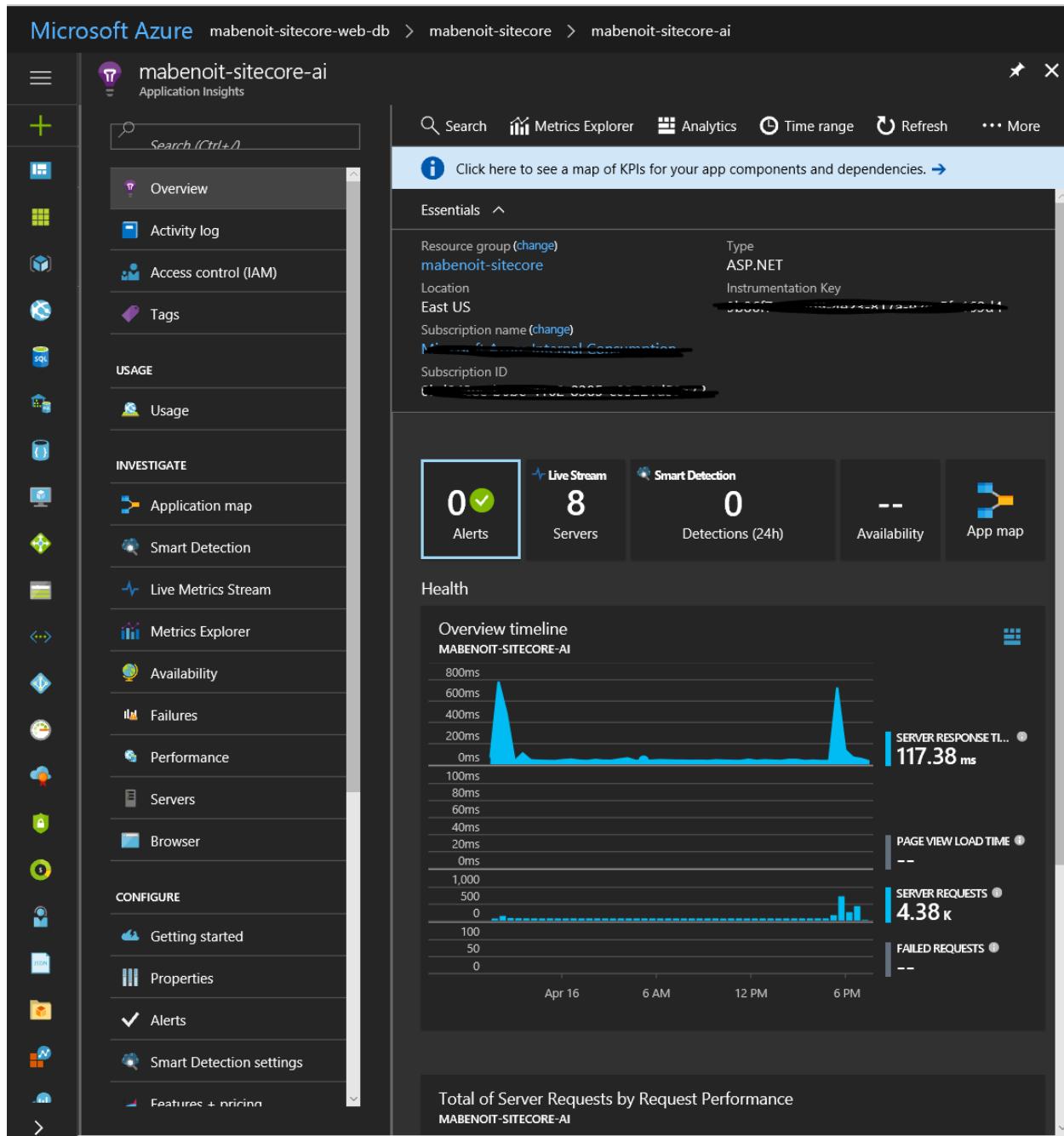
SERVER RESPONSE TIME 117.38 ms

PAGE VIEW LOAD TIME --

SERVER REQUESTS 4.38 k

FAILED REQUESTS --

Total of Server Requests by Request Performance MABENOIT-SITECORE-AI



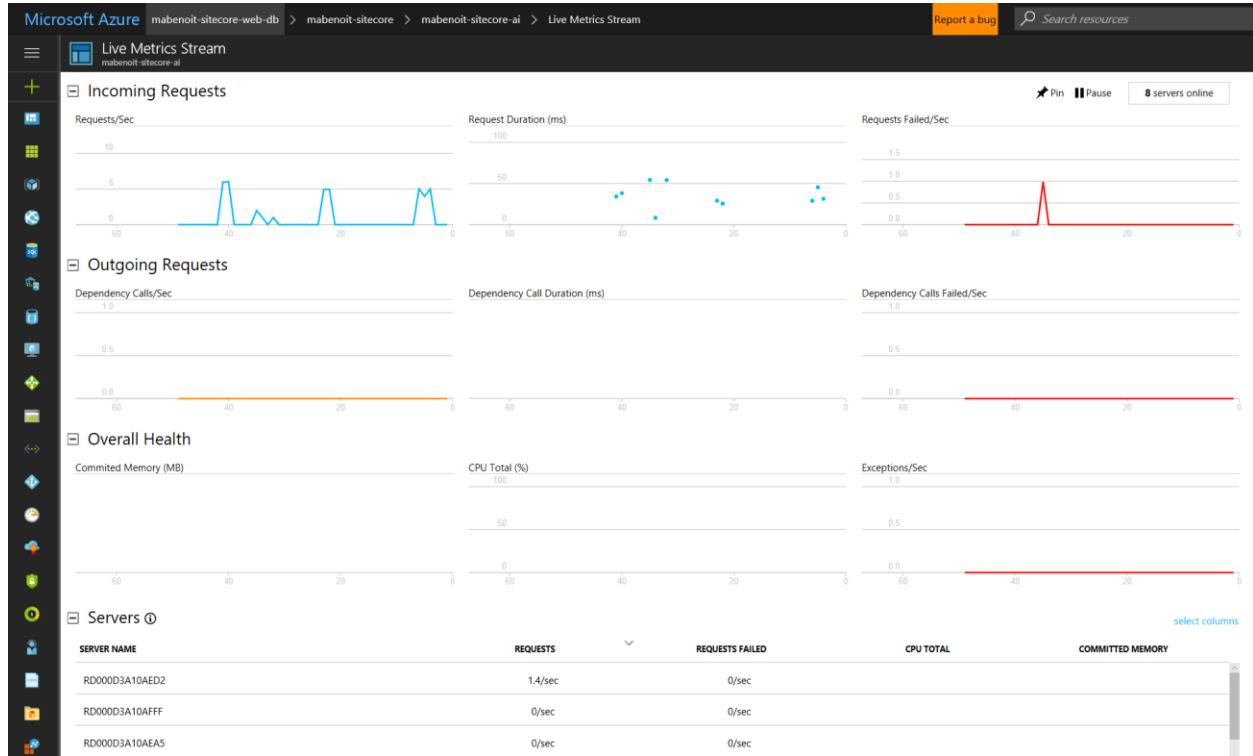
- On the Application Insights resource blade, go to the “**Getting started**” blade and copy/paste the “**Client-side telemetry**” JavaScript snippet code.

3. Open the “**App Service Editor (Preview)**” blade of your CD Azure Web App, open the “/wwwroot/layouts/Sample layout.aspx” file and edit it like explained above by inserting the Javascript just before the closing </head> tag.

4. Then click on the “**Browse Web Site**” toolbar action to navigate to your CD website. There, do some refresh on the homepage to generate some events. We will use and display them with the following steps above.

**Note:** With this quick setup on the Client side, be aware that the events will be logged in Application Insights with the Sitecore custom fields "Role" as "<undefined>" value, by default Sitecore provides the 2 values "CM" and "CD". Need to see how you would like to integrate these new client-side events within all your logs.

5. Go back on the Azure portal on the Application Insights resource blade, go to the “**Live Metrics Stream**” blade and open in another web browser your CD website URL and do some refresh on the Homepage to generate some server request.
6. On the “**Live Metrics Stream**” blade you could see the generated logs:



7. Go back on the “**Overview**” resource blade of the Application Insights resource, click on the “**Search**” toolbar action. Like mentioned earlier during this training, by default Sitecore logs all in Application Insights (not anymore as File systems like used to do). On the image below you could for example play with the “**Filters**” to display all the “**Trace**” of the “**CM**” related to the “**sitecore\_master\_index**”. You should see the associated traces we did by the previous exercise with Azure Search.

The screenshot shows two side-by-side windows from the Azure Application Insights portal.

**Left Window (Overview Blade):**

- Toolbar:** Search (Ctrl+/, Metrics Explorer, Analytics, Time range, Refresh, More).
- Resource Information:** Resource group (change) mabenoit-sitecore, Type ASP.NET, Location East US, Subscription name (change), Subscription ID [REDACTED].
- Metrics:** Alerts (0), Servers (8), Detections (24h) (0), Availability, App map.
- Health:** Overview timeline showing SERVER RESPONSE TIME (107.36 ms) and PAGE VIEW LOAD TIME (921 ms) over the last 24 hours.
- Logs:** Total of Server Requests by Request Performance.

**Right Window (Analytics Blade):**

- Toolbar:** Search (sitecore\_master\_index), Time range, Filters, Refresh, Reset, Analytics (circled in red).
- Search Results:** Filtered on Trace with CM, 43 total results between 4/15/2017 8:06 PM and 4/16/2017 8:06 PM.
- Log List:**
  - 4/16/2017 6:37:21 PM - TRACE ManagedPoolThread #5 22:37:24 INFO Job ended: Index\_Update\_IndexName:sitecore\_master\_index (units processed: 4636) PC Informational
  - 4/16/2017 6:36:42 PM - TRACE 8840 22:36:45 INFO Crawler [sitecore\_master\_index]: Processed 4000 items Device type PC Severity level Informational
  - 4/16/2017 6:35:47 PM - TRACE 15464 22:35:51 INFO Crawler [sitecore\_master\_index]: Processed 3000 items Device type PC Severity level Informational
  - 4/16/2017 6:34:52 PM - TRACE 15464 22:34:56 INFO Crawler [sitecore\_master\_index]: Processed 2000 items Device type PC Severity level Informational
  - 4/16/2017 6:34:02 PM - TRACE 15464 22:34:05 INFO Crawler [sitecore\_master\_index]: Processed 1000 items Device type PC Severity level Informational
  - 4/16/2017 6:33:15 PM - TRACE ManagedPoolThread #5 22:33:19 INFO Job started: Index\_Update\_IndexName:sitecore\_master\_index Device type PC Severity level Informational
  - 4/16/2017 6:33:15 PM - TRACE 13264 22:33:19 WARN IndexCustodian. FullRebuild triggered on index sitecore\_master\_index Device type PC Severity level Warning
  - 4/16/2017 6:02:35 PM - TRACE ManagedPoolThread #3 22:02:38 INFO Job ended: Index\_Update\_IndexName:sitecore\_master\_index (units processed: 4636) PC Informational
  - 4/16/2017 6:01:55 PM - TRACE 3188 22:01:59 INFO Crawler [sitecore\_master\_index]: Processed 4000 items

- From there, you could then click on the “Analytics” toolbar button, you will navigate to the Application Insights Analytics tools to perform more advanced queries within a powerful editor.

The screenshot shows the Application Insights Analytics blade with a query results table.

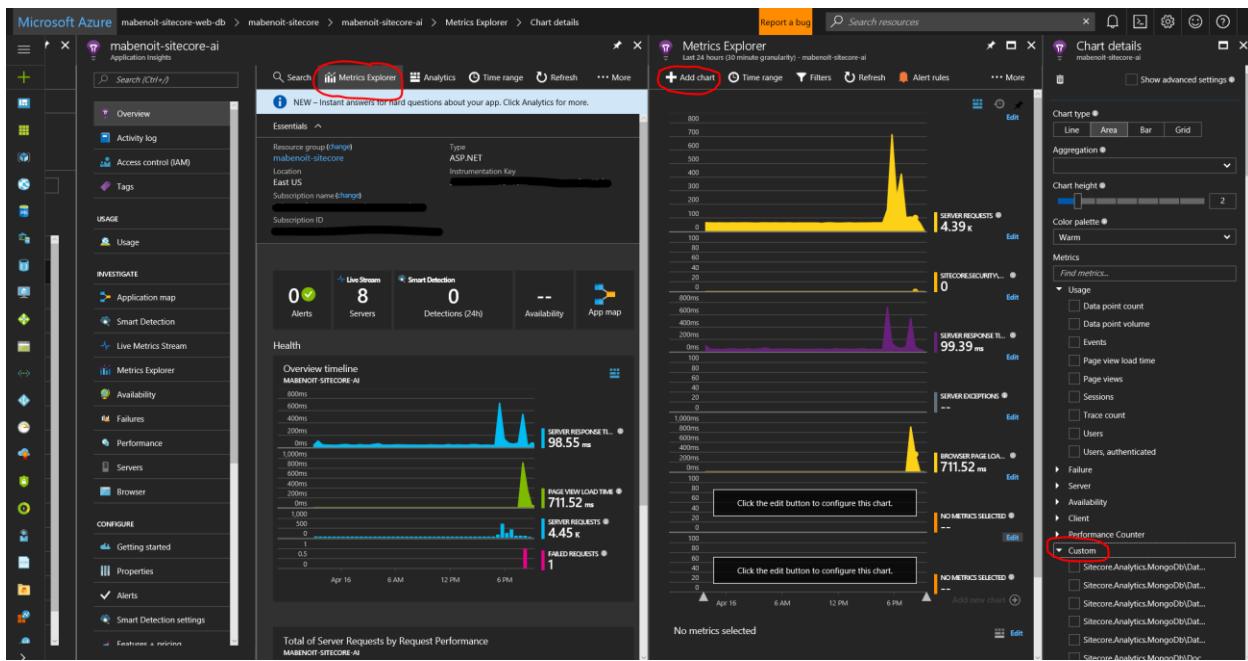
**Query:**

```
traces
| where timestamp > datetime(2017-04-16T00:59:999Z) and timestamp < datetime(2017-04-17T00:51:00.001Z)
| where (type == "trace" and ((timestamp > datetime(2017-04-16T00:51:00.000Z) and timestamp < datetime(2017-04-17T00:51:00.000Z)) and tostring(customDimensions["Role"]) == "CM"))
| top 40 by timestamp desc
```

**Table Headers:** timestamp, message, severityLevel, customDimensions, operation\_Name, operation\_Id, operation\_ParentId, application\_Version.

**Table Data:** The table lists 43 records loaded, each row representing a trace event with details like timestamp, message content, severity level (e.g., Informational, Warning), custom dimensions (e.g., Role: CM), and operation metadata.

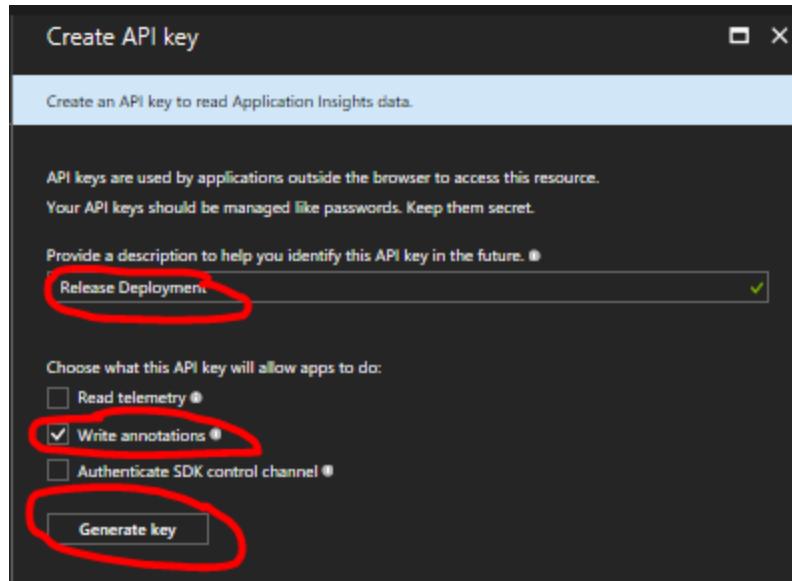
- Go back on the Azure portal, on the Application Insights resource go to the “Overview” blade, click on the “Metrics Explorer” toolbar action.



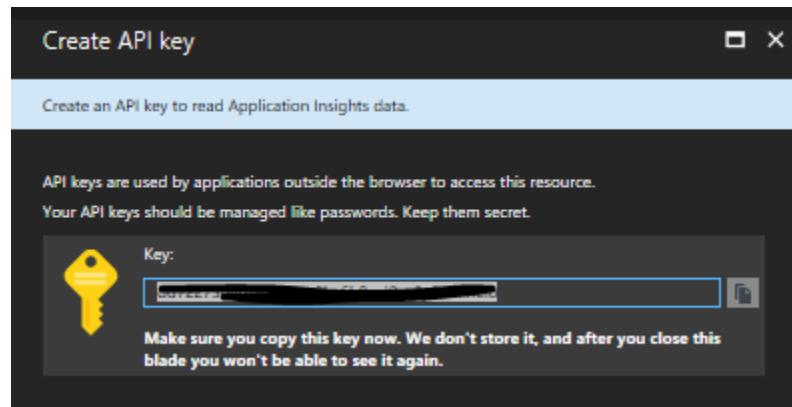
10. From there you could “Add chart” (or “Alert rules”) based on built-in metrics or Custom like illustrated provided by Sitecore.
11. Now let’s add some annotations within these charts to be able to see when a release has been deployed and the impact it could have with the metrics for example. For that, let’s first go to the “API Access” blade and click on the “Create API Key” action toolbar:

The screenshot shows the Microsoft Application Insights interface for the 'mabenoit-test - API Access' application. On the left, a sidebar lists various monitoring and configuration options. At the bottom of the sidebar, the 'API Access' option is highlighted with a red circle. The main content area displays the 'Create API key' blade. At the top of this blade, there is a red circle around the '+ Create API key' button. Below it, the 'Application ID' field contains a long GUID. A table header with columns 'API KEY DESCRIPTION', 'LAST USED', 'CREATED ON', and 'PERMISSIONS' is shown, followed by a message: 'You haven't set up any API keys. Click 'Create API key' to get started. Learn more'.

12. From the “**Create API key**” blade, provide a description, enable the “**Write annotations**” checkbox and then click on the “**Generate key**” button.



13. Then, make sure you copy this generated key. We will use it soon and like explained "after you close this blade you won't be able to see it again".



14. Make a local copy of the [Powershell script from GitHub](#).
15. Right-click on it and select "Edit" to open it with PowerShell ISE tool and run the following PowerShell command:

```
.\\CreateReleaseAnnotation.ps1 -applicationId "your application id" -  
apiKey "your api key" -releaseName "MyRelease1.0" -releaseProperties  
@{ "ReleaseDescription"="a description"; "TriggerBy"="My Name" }
```

```

# Sample usage: .\CreateReleaseAnnotation.ps1 -applicationId "cappid" -apiKey "<apiKey>" -releaseName "<releaseName>" -releaseProperties @{"ReleaseDescription"="Release with annotation"; "ReleaseName"="My Release 1.0"} -eventDateTime "2017-11-11T18:00:00Z"
param(
    [parameter(Mandatory = $true)][string]$applicationId,
    [parameter(Mandatory = $true)][string]$apiKey,
    [parameter(Mandatory = $true)][string]$releaseName,
    [parameter(Mandatory = $false)]$releaseProperties,
    [parameter(Mandatory = $false)][Datetime]$eventDateTime
)
# background info on how fblink works: After you submit a web request, many sites redirect through a series of intermediate pages before you finally land on the destination page.
# So when calling Invoke-WebRequest, the result it returns comes from the final page in any redirect sequence. Hence, I set MaximumRedirection to 0, as this prevents the call to
# be redirected. By doing this, we get a response with status code 302, which indicates that there is a redirection link from the response body. We grab this redirection link and
# construct the url to make a release annotation.
# Here's how this logic is going to work:
# 1. Client send http request, such as: http://go.microsoft.com/fwlink/?LinkId=625115
# 2. FBLINK get the request and send the destination URL for it, such as: http://www.bing.com
# 3. FBLINK gets a new http response with status code "302" and with destination URL "http://www.bing.com", send it back to client.
# 4. Client, such as a powershell script, knows that status code "302" means redirection to new a location, and the target location is "http://www.bing.com"
function GetRequestUrlFromFwLink($fwLink)
{
    $request = Invoke-WebRequest -uri $fwLink -MaximumRedirection 0 -UseBasicParsing -ErrorAction Ignore
    if ($request.StatusCode -eq "302") {
        return $request.Headers.Location
    }
    return $null
}
function CreateAnnotation($grpEnv)
{
    $retries = 1
    $success = $false
    while ($success -and $retries -lt 6) {
        $location = "$grpEnv/applications/$applicationId/Annotations?api-version=2015-11"
        Write-Host "Invoke a web request for $location to create a new release annotation. Attempting $retries"
        Set-Variable -Name createResultStatus -Force -Scope Local -Value $null
        Set-Variable -Name createResultStatusDescription -Force -Scope Local -Value $null
        Set-Variable -Name result -Force -Scope Local
        try {
            $request = Invoke-WebRequest -uri $location -Method Post -Body $body -ContentType "application/json" -Headers $headers -MaximumRedirection 0 -UseBasicParsing -ErrorAction Ignore
            if ($request.StatusCode -eq "201") {
                $success = $true
                $result = $request.Content
            }
            else {
                $retries++
            }
        }
        catch {
            $retries++
        }
    }
    return $result
}

```

PS C:\Users\mabenoit\Documents\Sitecore\workshop\My workshop\Lab Materials> .\CreateReleaseAnnotation.ps1 -applicationId "cappid" -apiKey "dd72275b40iqbasyqw0ku6b8y" -releaseName "My Release 1.0" -releaseProperties @{"ReleaseDescription"="Release with annotation"; "ReleaseName"="My Release 1.0"} -eventDateTime "2017-11-11T18:00:00Z"

Invoke a web request for https://aigsl1.asvc.visualstudio.com/applications/cappid/annotations?api-version=2015-11 to create a new release annotation. Attempting 1

Release annotation created. Id: before11/11/2017 18:00:00 2f403d8d71d.

PS C:\Users\mabenoit\Documents\Sitecore\workshop\My workshop\Lab Materials>

16. You could verify your annotation has been successfully added by navigating to the “Metrics Explorer” blade (on the “Overview” blade, click on the “Metrics Explorer” toolbar action):

Name	MyRelease1.0
Kind	Deployment
Event Time	5/11/2017 5:50 PM
ID	b7104371-1234-422d-a8d71d
ReleaseDescription	a description
ReleaseName	MyRelease1.0
TriggerBy	My Name

**Note:** You can also create annotations from any process or tool, you like by running the PowerShell script itself like we just did or adapt it according your need. Furthermore, if you use VSTS, you would like to use [its associated extension](#).

## Takeaways

- Analyze Sitecore logs with Application Insights
  - [https://doc.sitecore.net/sitecore\\_experience\\_platform/setting\\_up\\_and\\_maintaining/sitecore\\_on\\_azure/analytics/analyze\\_sitecore\\_logs\\_with\\_application\\_insights](https://doc.sitecore.net/sitecore_experience_platform/setting_up_and_maintaining/sitecore_on_azure/analytics/analyze_sitecore_logs_with_application_insights)

- Annotations on metric charts in Application Insights
  - <https://docs.microsoft.com/en-us/azure/application-insights/app-insights-annotations>
- Sitecore Azure Search Overview
  - [https://doc.sitecore.net/sitecore\\_experience\\_platform/setting\\_up\\_and\\_maintaining/search\\_and\\_indexing/sitecore\\_azure\\_search\\_overview](https://doc.sitecore.net/sitecore_experience_platform/setting_up_and_maintaining/search_and_indexing/sitecore_azure_search_overview)
- Monitoring and Maintaining Azure Search
  - [https://doc.sitecore.net/sitecore\\_experience\\_platform/setting\\_up\\_and\\_maintaining/sitecore\\_on\\_azure/analytics/monitoring\\_and\\_maintaining\\_azure\\_search](https://doc.sitecore.net/sitecore_experience_platform/setting_up_and_maintaining/sitecore_on_azure/analytics/monitoring_and_maintaining_azure_search)
- Support reference for Azure Search
  - [https://doc.sitecore.net/sitecore\\_experience\\_platform/setting\\_up\\_and\\_maintaining/search\\_and\\_indexing/support\\_reference\\_for\\_azuresearch](https://doc.sitecore.net/sitecore_experience_platform/setting_up_and_maintaining/search_and_indexing/support_reference_for_azuresearch)
- Sitecore SXA Search implementation
  - <https://doc.sitecore.net/sitecore%20experience%20accelerator/working%20with%20content/adding%20and%20changing%20content/walkthrough%20adding%20search%20functionality%20for%20your%20site>

## Exercise 6 – QnA Maker and Bot Service (20 min)

### Objectives

The goal of this exercise is to be familiar with the Cognitive Services (more specifically in this case with the QnA Maker) and the Bot Framework/Service and see the key features.

Through this exercise, you will play/use:

- **Azure portal**
- **Cognitive Services**
- **QnA Maker**
- **Bot Framework**



### Task 1: QnA Maker (10 min)

In this section, the attendee will be able to provision a QnA Maker service from the Cognitive Services and add some key/value entries for example. He/she will have the opportunity to interact with Web Bot for test purposes.

1. Go to <https://qnamaker.ai/>
2. You will be asked to sign-in.
3. Then go to the “**Create new service**” tab and fill out all the fields below and click on the “**Create**” button:
  - **Service Name:** MyFirstBotExperience
  - **FAQ URL(s):** <https://docs.botframework.com/en-us/faq>

qnamaker.ai/Create

Microsoft

Mathieu

QnA Maker PREVIEW My services Create new service Documentation Feedback

## Creating a QnA service

Add sources which contain question and answer pairs you would like to include in your knowledge base.

**What would you like to name your service?**  
The service name is for your reference and you can change it at anytime.  
SERVICE NAME  
MyFirstBotExperience

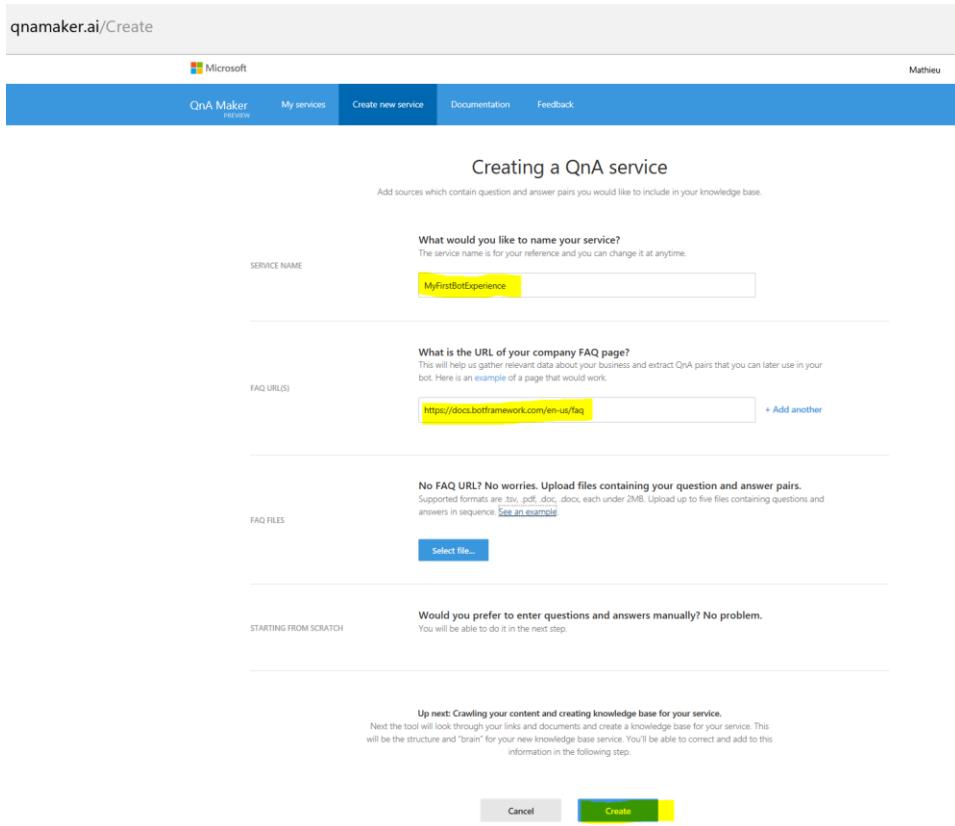
**What is the URL of your company FAQ page?**  
This will help us gather relevant data about your business and extract QnA pairs that you can later use in your bot. Here is an [example](#) of a page that would work.  
FAQ URL(S)  
https://docs.botframework.com/en-us/faq + Add another

**No FAQ URL? No worries. Upload files containing your question and answer pairs.**  
Supported formats are **.tsv, .pdf, .doc, .docx**, each under 2MB. Upload up to five files containing questions and answers in sequence. [See an example](#)  
FAQ FILES  
Select file...

**Would you prefer to enter questions and answers manually? No problem.**  
You will be able to do it in the next step.  
STARTING FROM SCRATCH

Up next: Crawling your content and creating knowledge base for your service.  
Next the tool will look through your links and documents and create a knowledge base for your service. This will be the structure and "brain" for your new knowledge base service. You'll be able to correct and add to this information in the following step.

Cancel Create



4. After few seconds, your service is provisioned, and you could see the different key/value pairs prepopulated (you landed on the “**Knowledge Base**” tab after the creation):

qnamaker.ai/Edit/KnowledgeBase?kbId=0c9ddeb2-b5a7-4304-9c48-c6c8bfd29c48

Microsoft

OnA Maker PREVIEW My services Create new service Documentation Feedback

MyFirstBotExperience Download Knowledge Base Replace Knowledge Base

Save and retrain Publish Retrained a few seconds ago

Knowledge Base KNOWLEDGE BASE | 30 QnA pairs + Add new QnA pair

	Question	Answer
1	What is the Microsoft Bot Framework?	<p>Original source: Editorial</p> <p>Original source: <a href="https://docs.botframework.com/en-us/faq">https://docs.botframework.com/en-us/faq</a></p> <p>The Microsoft Bot Framework is a comprehensive offering to build and deploy high-quality bots for your users to enjoy in their favorite conversation experiences. Developers writing bots all face the same problems: bots require basic I/O; they must have language and dialog skills; they must be performant, responsive and scalable; and they must connect to users - ideally in any conversation experience and language the user chooses. Bot Framework provides just the tools you need to build, connect, manage and publish intelligent bots that interact naturally wherever your users are talking - from mobile and web, Office 365 and Azure, Facebook Messenger, Slack, Office 365 Mail and many other popular services.</p> <p>The Bot Framework consists of a number of components including the Bot Builder SDK, Developer Portal and the Bot Directory.</p> <p>The Bot Builder SDK is an open source SDK hosted on GitHub (<a href="https://github.com/Microsoft/BotBuilder">https://github.com/Microsoft/BotBuilder</a>) that provides everything you need to build great dialogs within your Node.js-, .NET- or REST API-based bot.</p> <p>The Bot Framework Developer Portal (<a href="https://dev.botframework.com">https://dev.botframework.com</a>) is a free service that lets you connect your bot(s) seamlessly to Skype, Slack, Facebook Messenger, Kik, Office 365 Mail, Office 365 Groups, Yammer, and the Bot Directory.</p> <p>All bots registered with Bot Framework are auto-configured to work with Skype and the Web.</p> <p>The Bot Directory is a public directory of all reviewed bots registered through the Developer Portal. Users can discover, try, and add bots to their favorite conversation experiences from the Bot Directory.</p>
2	Why should I write a bot?	<p>The Conversational User Interface, or GUI, has arrived! A plethora of chit-chat bots are offering to do things for us in our various communication channels like Skype and Facebook Messenger. A series of personal agent services have emerged that leverage machine, humans or both to complete tasks for us (x.ai, Clara Labs, Fancy Hands, Task Rabbit, Facebook "M" to name a few).</p> <p>The primary interface for these experiences is emoji, text, cards and images. It's a natural way for us to interact with machines. It's a fast way to pay the electric bill or sending money to a friend. Offerings such as Siri, Google Now and Cortana demonstrate value to millions of people every day, particularly on mobile devices where the GUI can be superior to the GUI or complements it.</p> <p>Bots and conversation agents are rapidly becoming an integral part of one's digital experience - they are as vital a way for users to interact with a service or application as is a web site or a mobile experience.</p>
3	Who are the target users for the Bot Framework? How will they benefit?	<p>The Bot Framework is targeted at developers who want to create a new service with a great bot interface or enable an existing service with a great bot interface.</p> <p>Developers writing bots all face the same problems: bots require basic I/O; they must have language and dialog skills; they must be performant, responsive and scalable; and they must connect to users - ideally in any conversation experience and language the user chooses.</p> <p>The Bot Framework provides tools to address these problems while also providing a way for users to discover, try, and add bots to the conversation experiences they love via the Bot Directory.</p> <p>As a participant in the Bot Framework, you may also take advantage of the auto-configured Skype channel and Web channel.</p> <p>The Direct Line API which can be used to test your bot's dialogging logic including the Bot Framework Emulator (<a href="https://github.com/Microsoft/BotFrameworkEmulator">https://github.com/Microsoft/BotFrameworkEmulator</a>), and/or powerful services provided by making your bot smarter through Cognitive Services (<a href="https://www.microsoft.com/cognitive">https://www.microsoft.com/cognitive</a>) such as LUIS for language understanding, Translation for automatic translation to more than 30 languages, and FormFlow for reflection generated bots.</p>
4	I'm a developer, what do I need to get started?	<p>You can get started by visiting the [Bot Framework site] (<a href="https://botframework.com">https://botframework.com</a>). To register a bot in the Developer Portal, you'll need a Microsoft account. The [Bot Builder SDK] (<a href="https://github.com/Microsoft/BotBuilder">https://github.com/Microsoft/BotBuilder</a>) is</p>

## 5. Go to the “Test” tab and ask the bot with some questions like:

- Any roadmap?
- How to get started?
- What's a bot?

qnamaker.ai/Edit/Test?kbId=0c9ddeb2-b5a7-4304-9c48-c6c8bfd29c48

Microsoft

OnA Maker PREVIEW My services Create new service Documentation Feedback

MyFirstBotExperience Download Knowledge Base Replace Knowledge Base

Save and retrain Publish Retrained 5 minutes ago

Knowledge Base TEST

Chat

For a quick start in the Developer Portal, you'll need a Microsoft account. The Bot Builder SDK is open source and available to all on GitHub. We also have guides to get started building a bot using Node.js, .NET or REST API:

- Get started with the Bot Builder - Node.js.
- Get started with the Bot Builder - .NET.
- Get started with the Bot Builder - REST API.

We are excited to provide initial availability of the Bot Framework at [Build 2016](#) and plan to continuously improve the framework with additional tools, samples, and channels. The Bot Builder SDK is an open source SDK hosted on GitHub and we look forward to the contributions of the community at large. [Feedback](#) as to what you'd like to see is welcome.

Choose the most appropriate answer from these alternatives:

We are excited to provide...

None of the above

Any roadmap?

Provide multiple alternative answers to the question to broaden the knowledge base.

Add alternative here

Any roadmap?

Type your message...

Download chat logs | Upload chat logs

6. You could see on the left side, the ability to select the right answer or indicate “**None of the above answer**” are relevant to train your QnA Maker service. Furthermore, on the right side, you could add manually “alternative answer”. With these 2 options, you could “**Save and retrain**” your QnA Maker service.
7. Then, on the top right end corner, click on the “**Publish**” button. You will see the summary of your service, click on the “**Publish**” button there.

The screenshot shows the Microsoft QnA Maker service interface. At the top, it says "qnemaker.ai/Publish/Preview?kbid=0c9ddeb2-b5a7-4304-9c48-c6c8bfd29c48". Below that is a Microsoft logo and a navigation bar with "QnA Maker PREVIEW", "My services", "Create new service", "Documentation", and "Feedback". On the right, it says "Mathieu". The main content area is titled "MyFirstBotExperience" and displays the message "Your service has never been deployed.". Under "Review your changes", there is a table comparing "Source" (https://docs.botframework.com/en-us/faq) against "Editorial". The table shows counts for QnA in production (0 vs 0), QnA in current draft (29 vs 1), QnA added (29 vs 1), and QnA deleted (0 vs 0). A "Download Diff File" link is available. At the bottom are "Cancel" and "Publish" buttons, with "Publish" being highlighted in yellow.

8. Few seconds after, your QnA Maker service is exposed via its URL.

The screenshot shows the Microsoft QnA Maker service interface. At the top, it says "qnemaker.ai/Publish?kbid=0c9ddeb2-b5a7-4304-9c48-c6c8bfd29c48". Below that is a Microsoft logo and a navigation bar with "QnA Maker PREVIEW", "My services", "Create new service", "Documentation", and "Feedback". On the right, it says "Mathieu". The main content area displays a yellow success message: "Success! Your service has been deployed. What's next?". It also says "You can always find the deployment details in your service's settings.". Below this, there is a "Sample HTTP request" box containing sample code for generating an answer. The code includes a POST request to https://westus.api.cognitive.microsoft.com/qnamaker/v1.0/generateAnswer with headers like Host, Ocp-Apim-Subscription-Key, and Content-Type, and a JSON payload with a question. At the bottom, it says "Need to fine-tune and refine? Go back and keep editing your service." and has an "Edit Service" button.

## Task 2: Bot Service (10 Min)

In this section, the attendee will provision a Bot Service/Framework through the Azure portal and will link the QnA Maker service just provisioned. With that, he/she will be able to expose his/her bot through different channel: Skype, Slack, etc.

1. Go to the Azure portal at <https://portal.azure.com>
2. Click on the add button to provision a new service and look for “**Bot Service**”, then click on the “**Create**” button.

**Results**

NAME	PUBLISHER	CATEGORY
Bot Service (preview)	Microsoft	Intelligence + analytics
aiProtect Denial of Service Protection (BYOL)	aiScaler	Compute
aiProtect Denial of Service Protection (hourly)	aiScaler	Compute
Imperva Incapsula	Imperva Incapsula	Security + Identity
F5 BIG-IP ADC+SEC BEST 1G Hourly	F5 Networks	Compute
F5 BIG-IP ADC+SEC BEST 25M Hourly	F5 Networks	Compute
F5 BIG-IP ADC+SEC BEST 200M Hourly	F5 Networks	Compute
F5 BIG-IP ADC+SEC BEST – BYOL	F5 Networks	Compute
Check Point vSEC (BYOL)	Check Point	Compute
Check Point vSEC Cluster - 2 NIC	Check Point	Compute
Check Point vSEC Autoscale	Check Point	Compute
Check Point vSEC - 2 NIC	Check Point	Compute
Check Point vSEC - Threat Prevention (PAYG)	Check Point	Compute
Voxibot 1.0.132	Ulex Innovative Systems	Compute

**Related to your search**

- aiMobile Mobile Acceleration -BYOL
- aiScaler Application Delivery Controller (BYOL)
- aiScaler Application Delivery Controller (hourly)

**Bot Service (preview)**

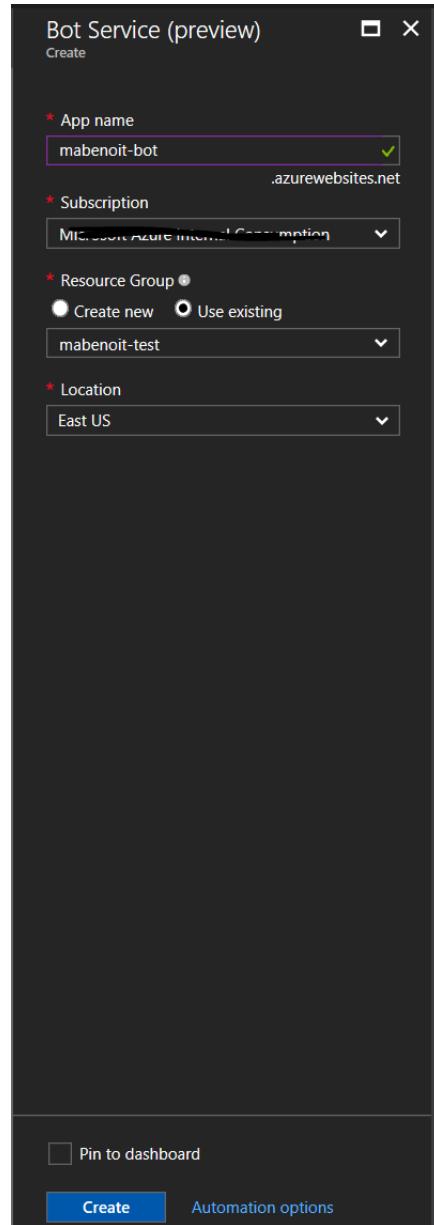
Azure Bot Service enables rapid intelligent bot development powered by the Microsoft Bot Framework, and runs in a serverless environment on Azure. Build, connect, deploy and manage intelligent bots that interact naturally wherever your users are talking – from your app or website to text/sms to Skype, Slack, Teams, Facebook Messenger, Kik, Office 365 mail and other popular services. Allow your bots to scale based on-demand and you pay only for the resources you consume.

[Documentation](#) [Solution Overview](#) [Pricing Details](#)

**PUBLISHER** Microsoft

**USEFUL LINKS**

3. On the “**Bot Service (preview)**” creation blade, fill out the fields accordingly and then click on the “**Create**” button:



4. Few seconds after the provisioning will be completed, you will land on the default blade of your new Bot Service. Click on the “**Create Microsoft App ID and password**” button.

#### Choose a language

We'll be creating some files to start with so we need to know what language you'll be developing your bot in. We currently support Node and C# but are working to add more languages soon.

5. You will be redirected through a sign-in process to an associated “**Generate App ID and password**” page. Click on the “**Generate an app password to continue**” button.

6. Then copy the “only time generated” password.

7. Then click on the “Finish and go back to the Bot Framework” button.

8. You will land on the Azure portal, copy/paste your generated password in the associated field, choose C# as the language and select the “**Question and Answer**” template. Then click on the “**Create bot**” button.

9. Select the associated (previously created) QnA Maker service and check the agreement part and click on the “**OK**” button.

10. Few minutes after, your Bot Service will be associated to your QnA Maker. You could type some question on the chat on your right to interact with your QnA Maker service via the new Bot just created.

```

Microsoft Azure mabenot-bot
mabenot-bot
|m .gitignore
|m Bots.sln
|m commands.json
|m debughost.cmd
|m host.json
|m messages
| | BasicQnAMakerDialog.cs
| | function.json
| | projection.json
| | projectlock.json
|m run.csx
|m PostDeployScripts
|m README.md

Report a bug Search resources
|m When did the Bot framework start?
|m You
|m Welcome You!
|m mabenot-bot
|m The core Bot Framework work has been underway since the summer of 2015.
|m mabenot-bot
|m Any roadmap?
|m You
|m We are excited to provide initial availability of the Bot Framework at Build 2016 and plan to continuously improve the framework with additional tools, samples, and channels. The Bot Builder SDK is an open source SDK hosted on GitHub and we look forward to the contributions of the community at large.
|m Feedback as to what you'd like to see is welcome.
|m mabenot-bot at 7:04:20 PM
|m Type your message...

```

11. Not included in the purpose of this lab but you could go to the “**Channels**” tab on the top left corner to configure your bot as an add-in for Skype, Slack, Microsoft Teams, etc. You could as well get the associated “Web Chat” snippet to embed it on a web site.

Channels	
	Skype <a href="#">Add to Skype</a>
	Web Chat
<b>Add another channel</b>	
	Direct Line
	Email
	Facebook Messenger
	GroupMe
	Kik
	Microsoft Teams
	Slack
	Telegram
	Twilio (SMS)

## Takeaways

Let's imagine you send the content of your FAQ from Sitecore and expose your Q&A Module with this setup: QnA Maker and Bot Service?

During the Sitecore Hackathon 2017, you could find [7 projects/ideas which have used the Bot Framework](#).

You could find more documentation about the Azure Bot Service here:

<https://docs.botframework.com/en-us/azure-bot-service/>

You could find the free Bot Framework Emulator for local purposes as well:

<https://github.com/Microsoft/BotFramework-Emulator#download>

## Exercise 7 – Visual Studio Team Services - VSTS

### Objectives

The goal of this exercise is to be familiar with Visual Studio Team Services and see the key features.

**Through this exercise, you will play/use with:**

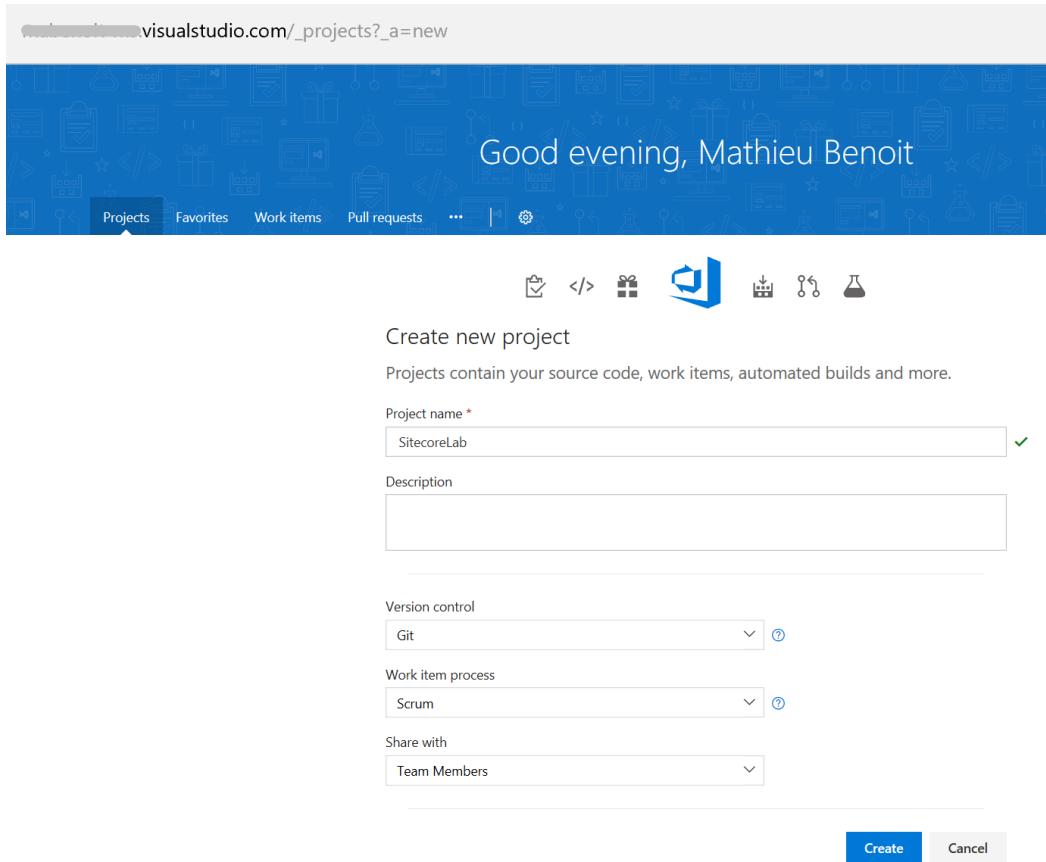
- **Visual Studio**
- **ARM Templates**
- **PowerShell**
- **Visual Studio Team Services**
- **Git**



### Task 1: Create a new VSTS project (5 min)

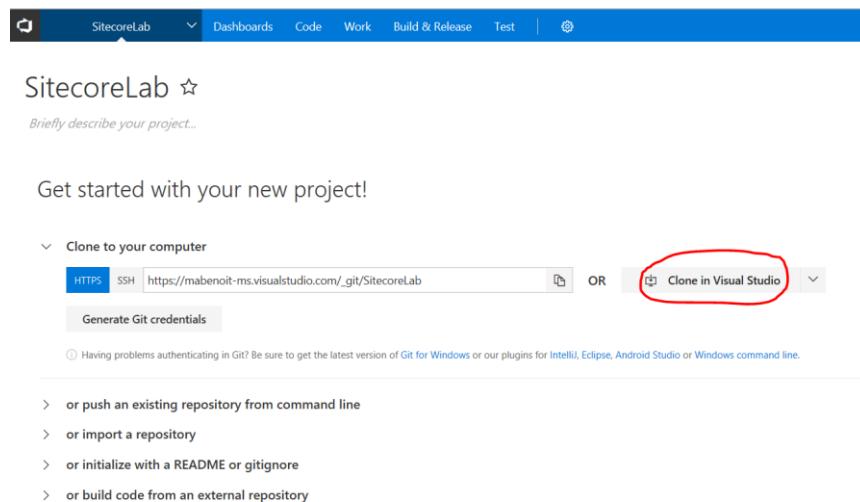
In this section, the attendee will create a new VSTS account and his/her first VSTS project.

1. You will need a Visual Studio Team Services account. If you don't have one, you can create from [here](#).
2. Go to your VSTS account home page and from there click on the “**Create new project**” button. Fill out all the field like illustrated below and click on the “Create” button

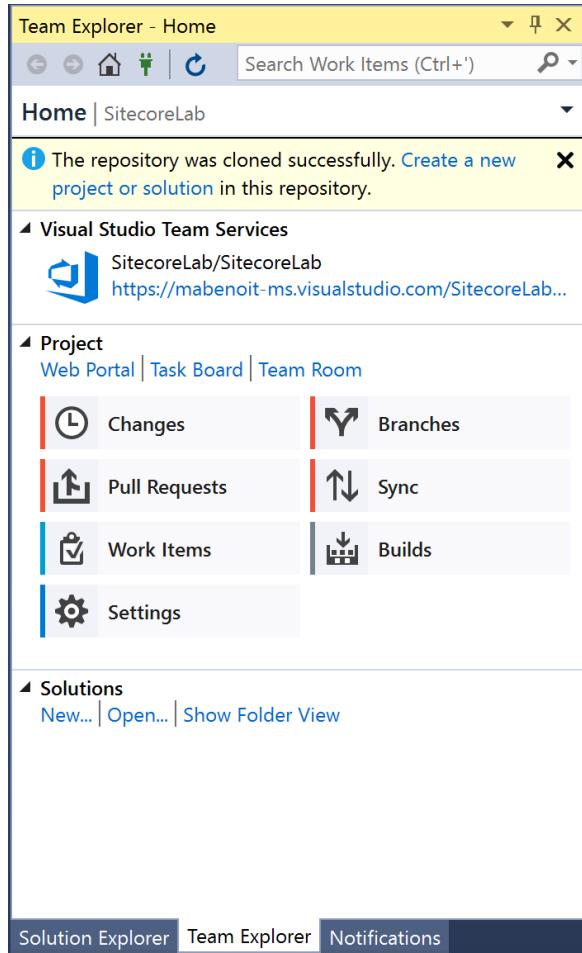


## Task 2: Code – Version Controlling with Git (15 min)

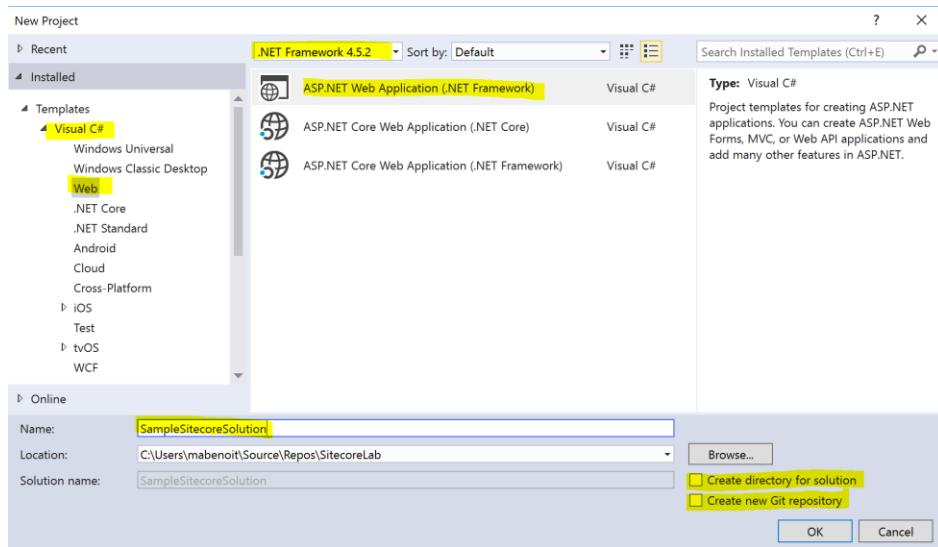
- Once created, you could click on the “Clone in Visual Studio” button.

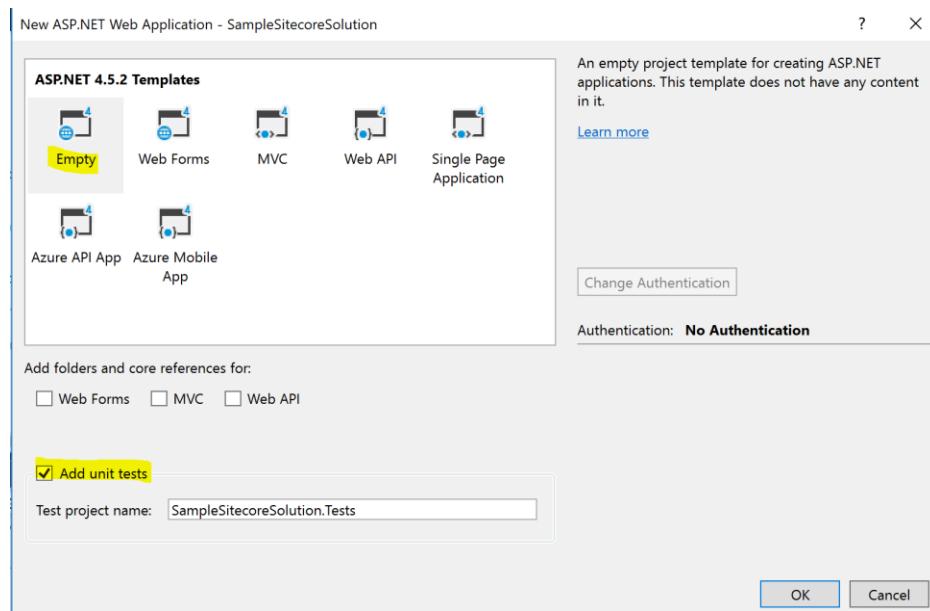


- Once successfully cloned, you should see this message within the Visual Studio View “Team Explorer”:

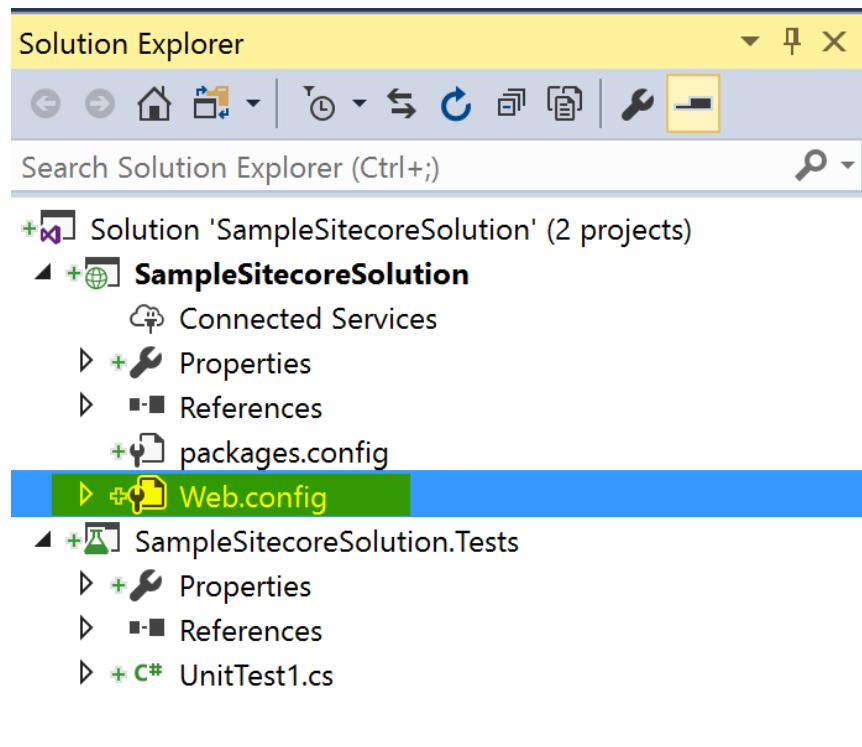


3. You could click on “Create a new project or solution”.





4. Then, the solution with the 2 projects “**SampleSitecoreSolution**” and “**SampleSitecoreSolution.Tests**” will be created.



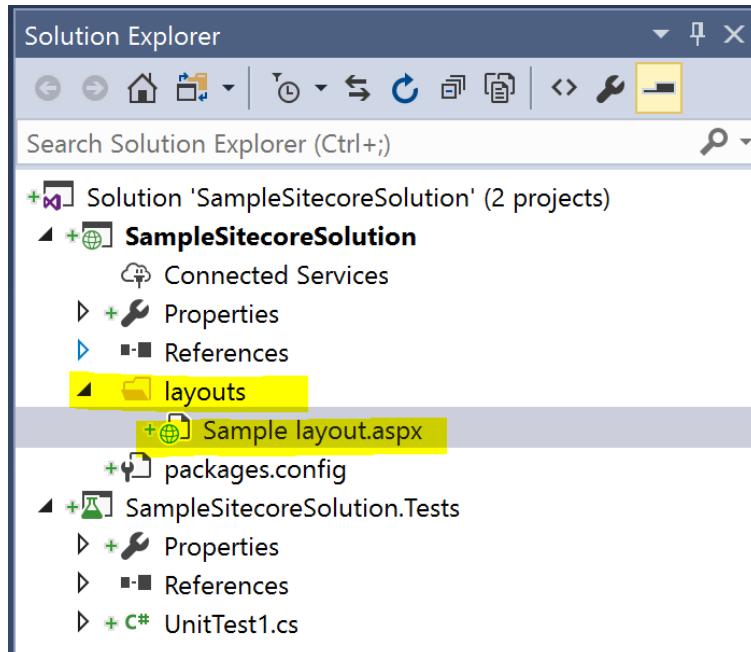
5. In the “**SampleSitecoreSolution**” project, remove the **web.config** file (to avoid overriding the Sitecore one when we will deploy the solution later).
6. In the “**SampleSitecoreSolution.Tests**” project, update the **UnitTest1.cs** file by inserting the code below into the **TestMethod1()** method:

```
Assert.IsTrue(true);
```

```

1  using System;
2  using Microsoft.VisualStudio.TestTools.UnitTesting;
3
4  namespace SampleSitecoreSolution.Tests
5  {
6      [TestClass]
7      public class UnitTest1
8      {
9          [TestMethod]
10         public void TestMethod1()
11         {
12                 Assert.IsTrue(true);
13         }
14     }
15 }
```

7. In the “**SampleSitecoreSolution**” project, create a “**layouts**” folder and create an empty “**Sample layout.aspx**” file.

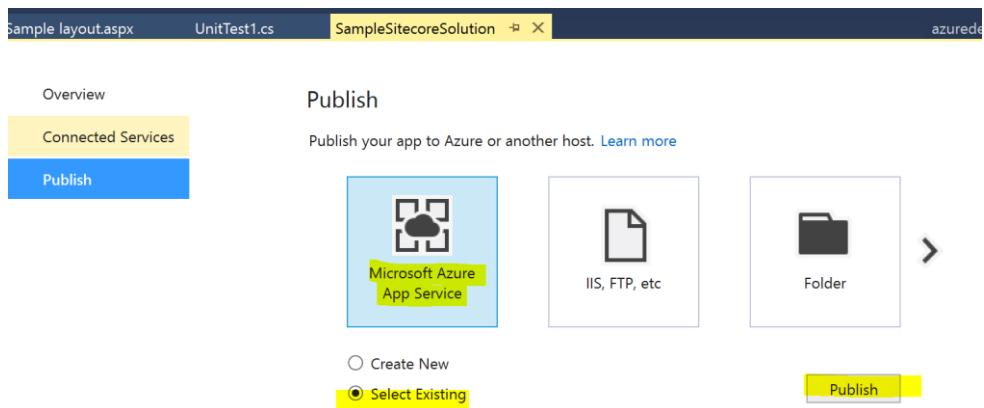


8. Retrieve the content of this file from the “**App Service Editor (Preview)**” feature of the Azure Web App (CD) to populate this file and change the <title> tag by adding the suffix “**from VSTS**” for example.

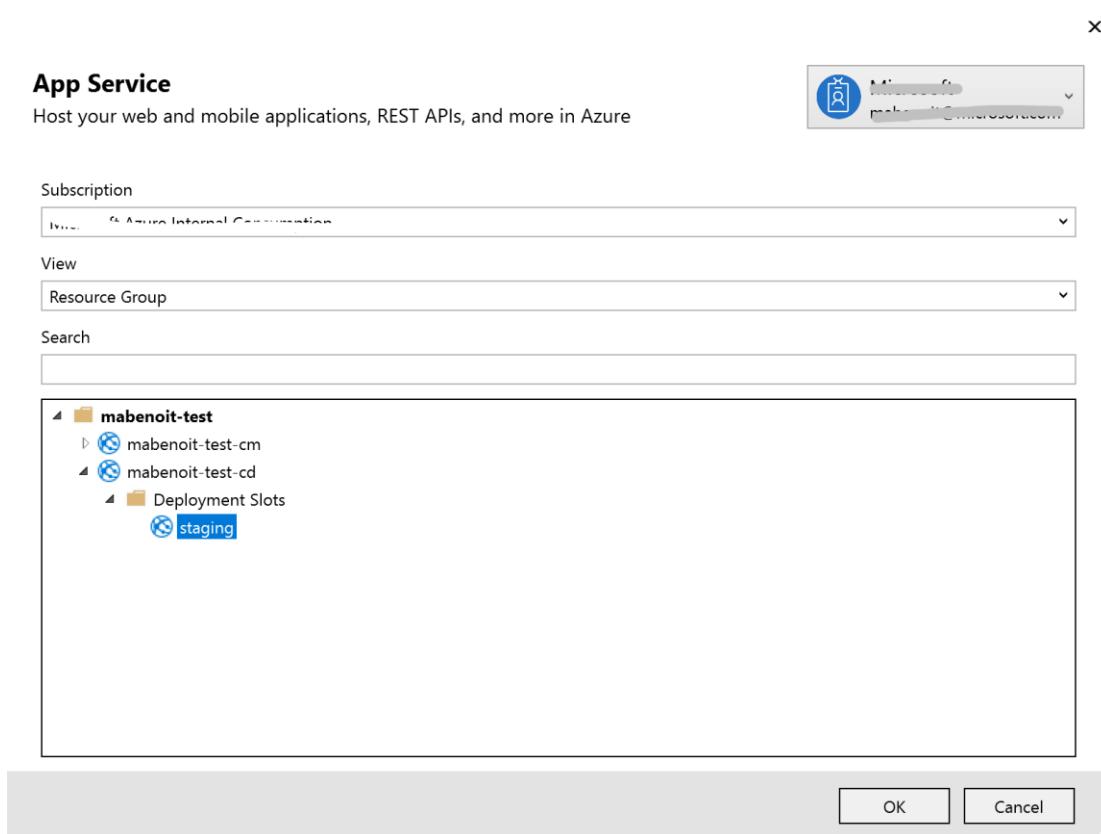
```

1  <%@ Page Language="C#" Inherits="System.Web.UI.Page" CodePage="65001" %>
2  <%@ OutputCache Location="None" VaryByParam="none" %>
3  <!DOCTYPE html>
4  <html lang="en" xmlns="http://www.w3.org/1999/xhtml">
5
6  <head runat="server">
7      <title>Welcome to Sitecore from Visual Studio</title>
8      <meta http-equiv="X-UA-Compatible" content="IE=edge" />
9      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
10     <meta name="CODE_LANGUAGE" content="C#" />
11     <meta name="vs_defaultClientScript" content="JavaScript" />
12     <meta name="vs_targetSchema" content="http://schemas.microsoft.com/intellisense/ie5" />
```

9. Right-click on the “**SampleSitecoreSolution**” project, and choose “**Publish**”. Then choose “**Microsoft Azure App Service**” and “**Select existing**” and click on the “**Publish**” button.



10. Select the appropriate resources to deploy this application into the CD Staging slot. Click then on the “OK” button.

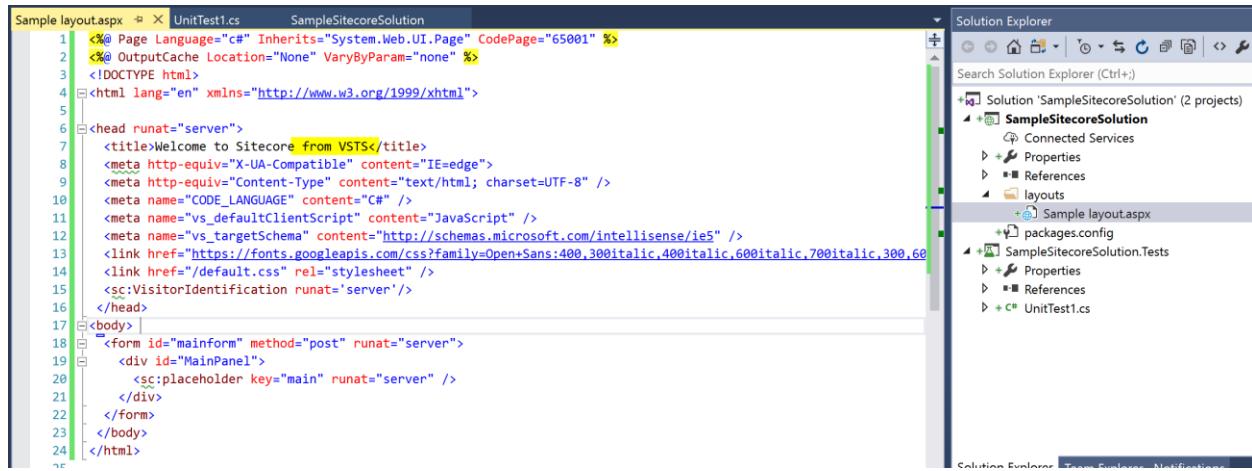


11. After the application is deploying and warming up again, the web browser will automatically be opened, and you could see your new title:

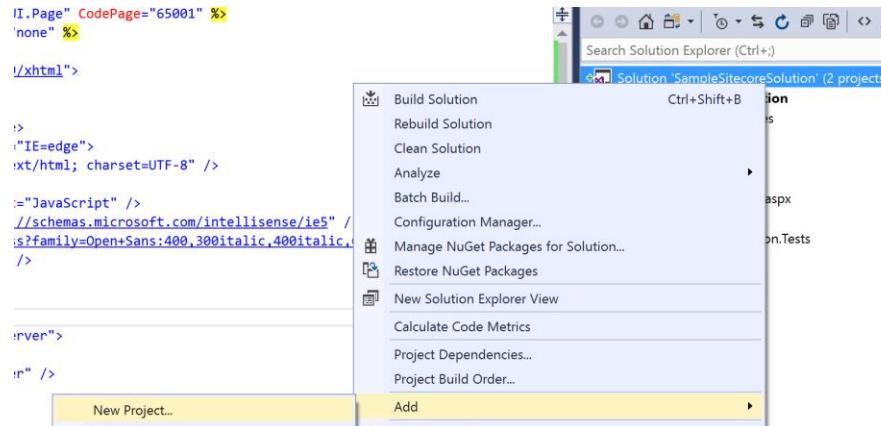


## Sitecore Experience Platform

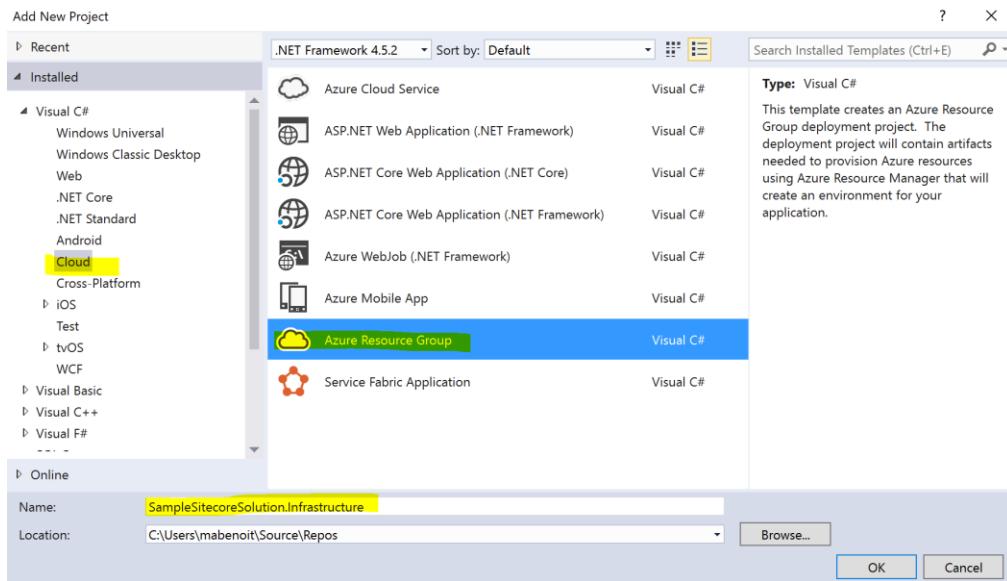
12. In the “layouts/Sample layout.aspx” file change the prefix “from Visual Studio” by “from VSTS” within the <title> tag.



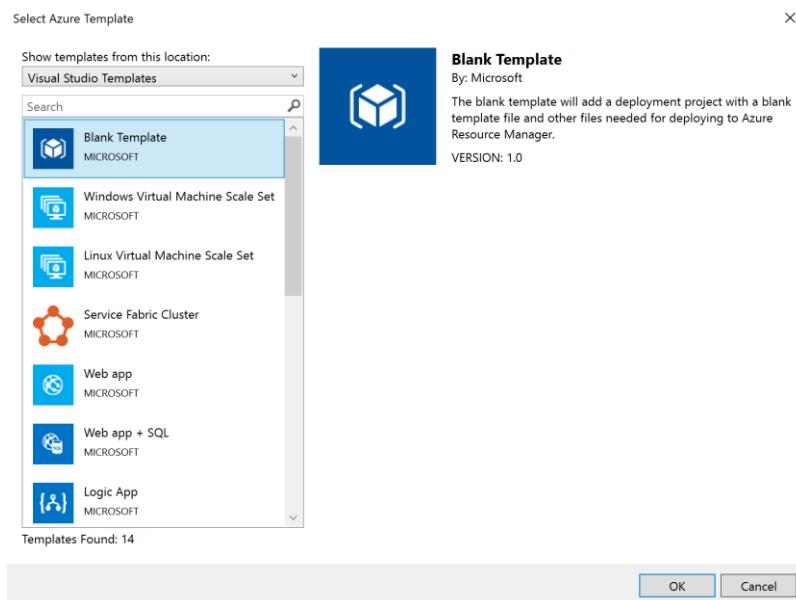
13. Now, let's create a new project within the solution by right-clicking on it and go to “Add > New Project...”:



14. Choose “Cloud > Azure Resource Group” and name the project, for example “**SampleSitecoreSolution.Infrastructure**”.



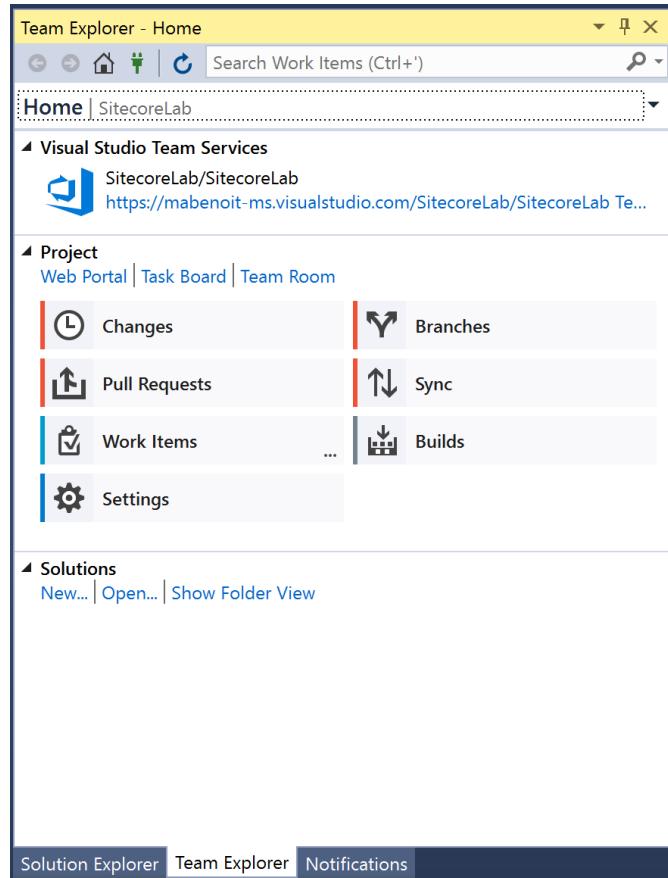
15. Then select the “**Blank Template**”:



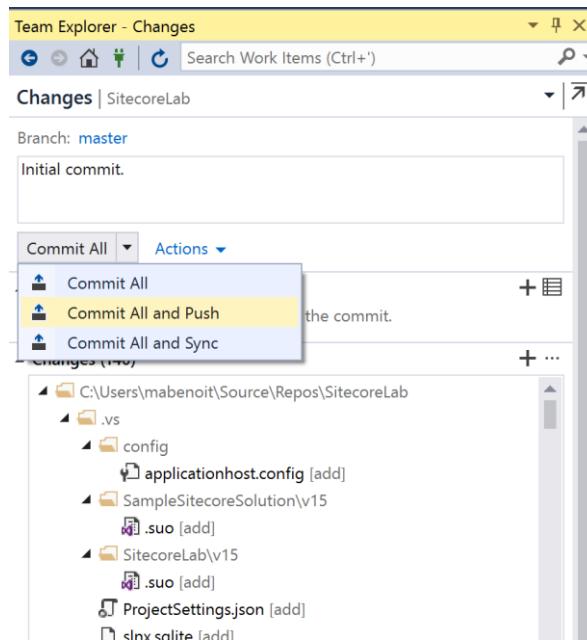
16. Retrieve the content of these 3 files (**.json**, **.parameters.json** and **.ps1**) based on what we did earlier in this lab with the Custom ARM Templates section to replace the content of these default files created accordingly.

**Note:** For the purpose of this lab we are not using them but with Azure Resource Group project you could use the JSON Outline View on hitting the ARM Template json file and furthermore you could [Deploy your PowerShell/ARM Templates within from Visual Studio](#).

17. Go to the **Team Explorer** view on the right-hand panel and click on “**Changes**”.



18. You will see all the changes made, enter a comment for the commit, for example “**Initial commit**” and select “**Commit All and Push**”.



19. After this action is performed you could go to your VSTS project and see in the “Code” tab all your code in the Git repository.

The screenshot shows the VSTS interface with the 'Code' tab selected. The repository 'SitecoreLab' is shown with its branches. The 'master' branch is selected, and the file list is displayed. The files listed are .vs, packages, SampleSitecoreSolution, SampleSitecoreSolution.Tests, and SampleSitecoreSolution.sln. Each file has a timestamp of '2 minutes ago' and a comment 'Initial commit.' by 'Mathieu Benoit'.

**Note:** Here some best practices could consist to add a gitignore for the bin, obj, etc. folders and furthermore instead of pushing directly the code in the master branch, create a dedicated branch with a Pull Request could be recommended.

### Task 3: Build – Continuous Integration

Get inspiration from [this lab](#). Ask your proctor if you are here.

The goal here is to create a Build definition to build the web application previously set up, run the unit tests and provide 2 artifacts: web deploy package and the ARM Templates/PowerShell files.

### Task 4: Release – Continuous Deployment

Get inspiration from [this lab](#). Ask your proctor if you are here.

The goal here is to create a Release definition to deploy the previous artifacts accordingly on different Environments. You could then deploy the WebDeploy package on a specific infrastructure just deployed with ARM Templates for example, trigger some LoadTest, trigger automatic action or require manual approval, etc.

## Takeaways

- Visual Studio Team Services - Hands-on-Labs
  - <https://almvm.azurewebsites.net/labs/vsts/>
- Considerations on using Deployment Slots in your DevOps Pipeline
  - <https://blogs.microsoft.com/visualstudioalm/2017/04/10/considerations-on-using-deployment-slots-in-your-devops-pipeline/>
- From the community, TDS on VSTS
  - <https://www.geekhive.com/buzz/post/2017/03/sitecore-artifact-promotion-deployments/>

