[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

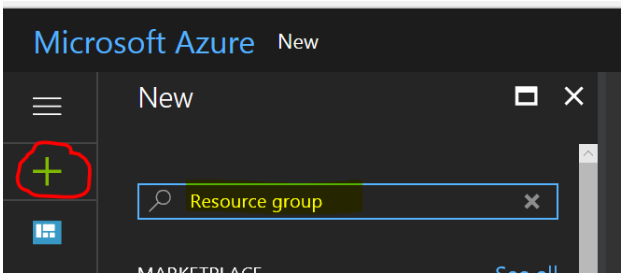
Task 1: QnA Maker (10 min) 75

Task 2: Bot Service (10 Min) 78

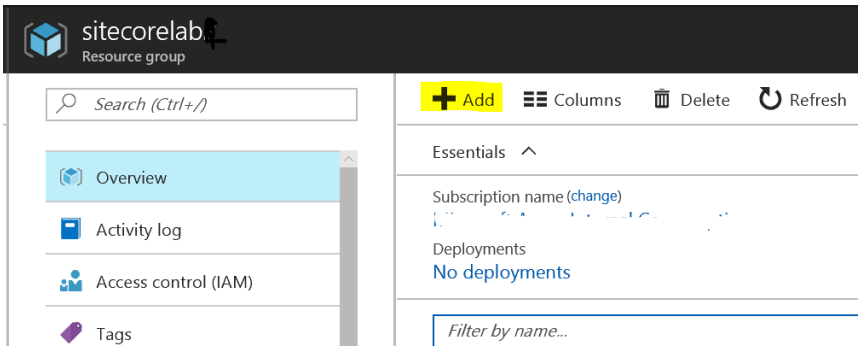
Takeaways 84

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

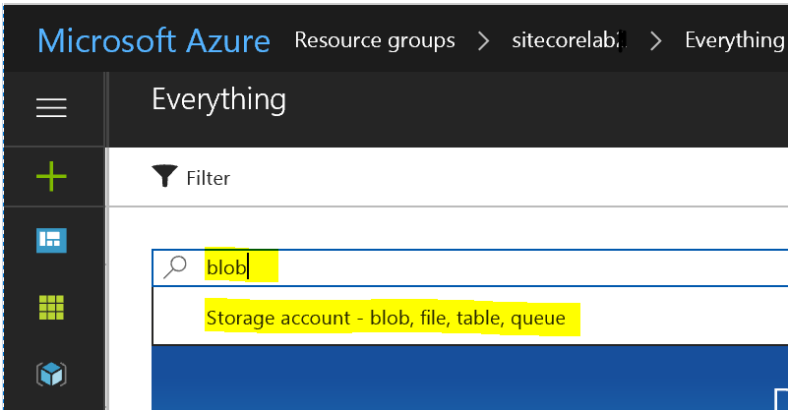
Exercise 2 - ARM Templates (85 min)

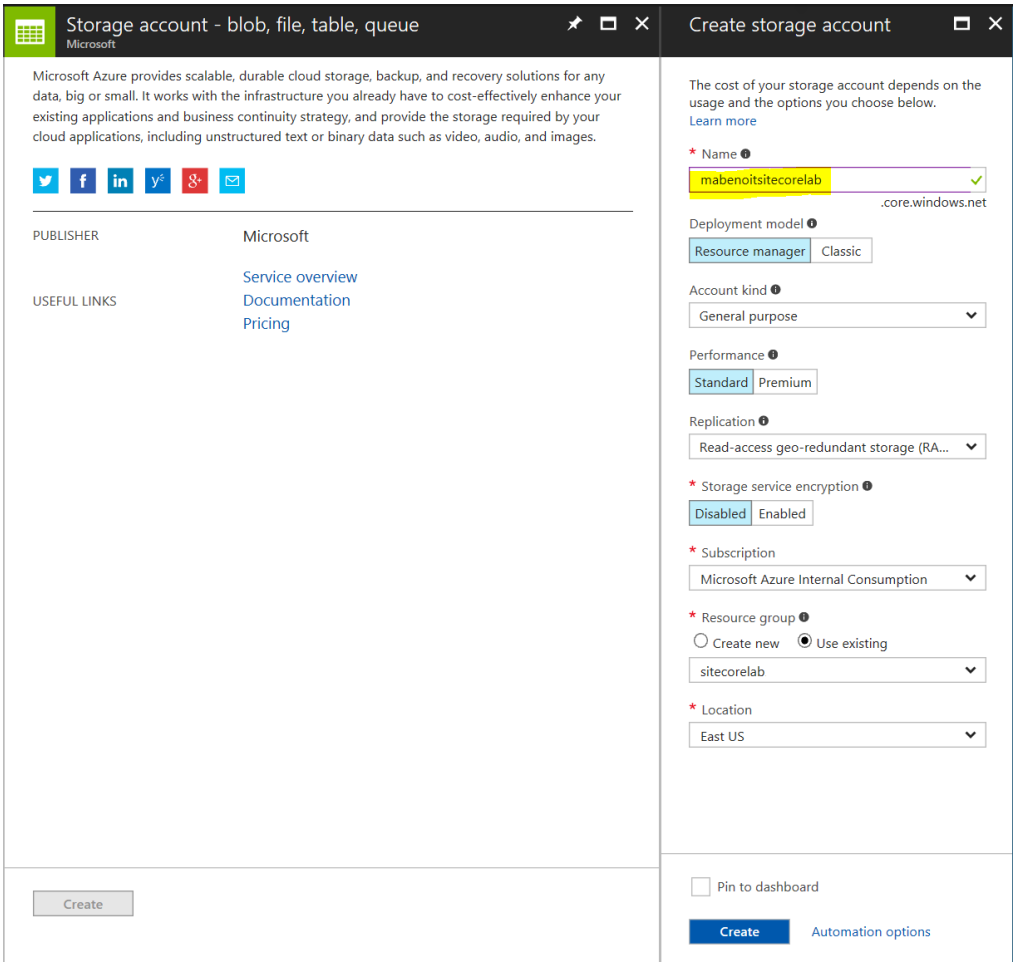


1. Once the Resource Group created, go to its "**Overview**" blade and click on the "**Add**" button.

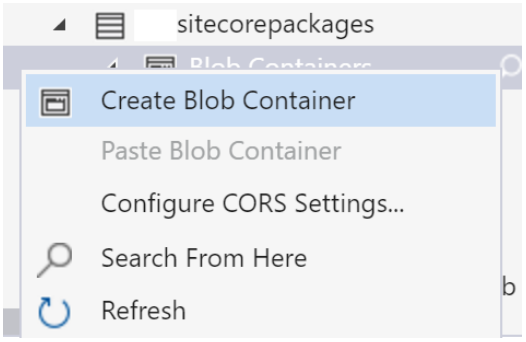


2. Create a new Azure Storage account looking for "blob" and fill out all the filed like illustrated below with a unique name.

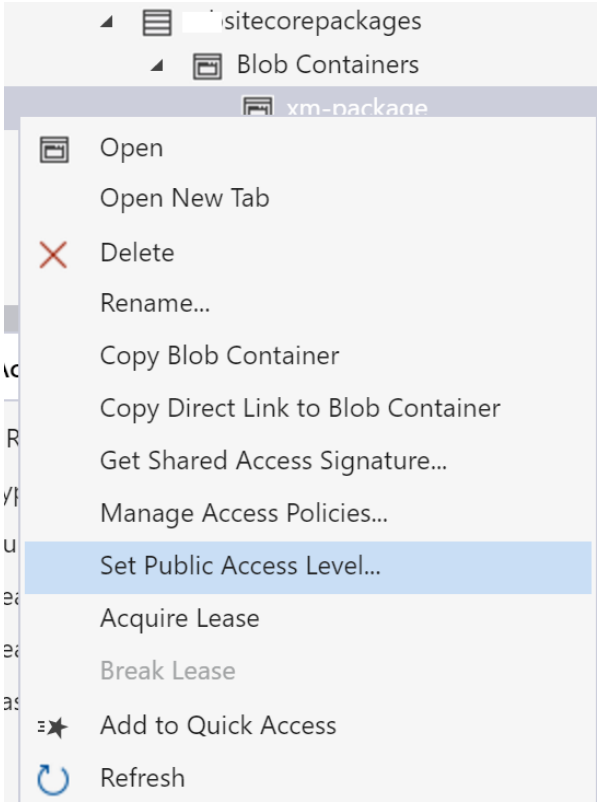


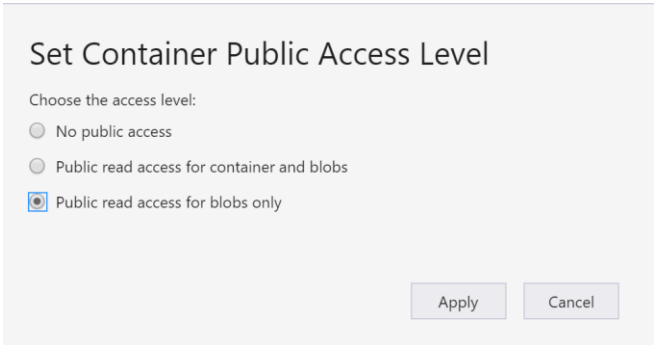


3. Open Microsoft Azure Storage Explorer locally on your VM, and sign-in with your credentials.
4. Locate your Azure Storage account previously created and right click on it to "Create Blob Container" named "**xm-packages**".



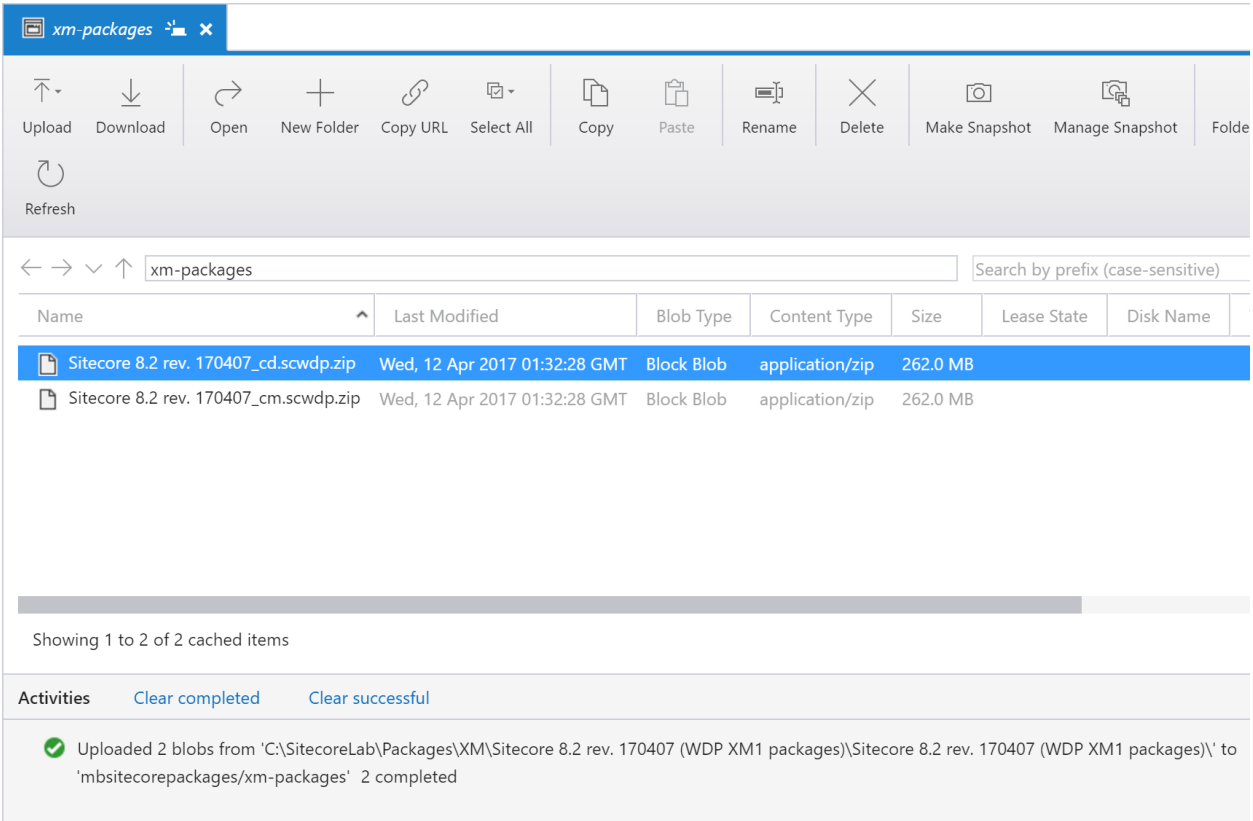
5. "**Set Public Access Level**" to this Blob container "xm-packages" by right-clicking on it and then select "**Public read access for blobs only**".





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).

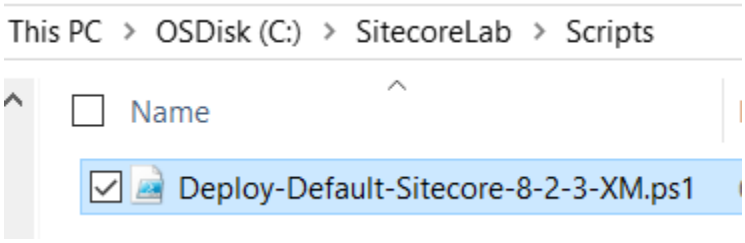


Here we are, we now have a clear setup to run the Sitecore ARM Templates with the coming tasks below.

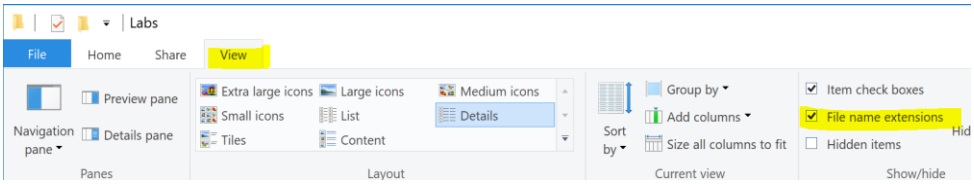
Task 2: Deploy the default Sitecore XM ARM Templates (45 min)

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

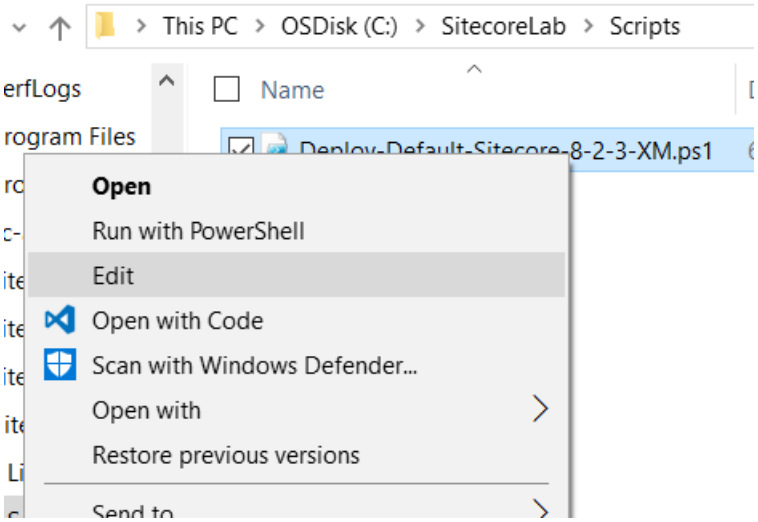
1. Create a new subfolder "**C:\SitecoreLab\Scripts**".
2. Go to https://gist.githubusercontent.com/mathieu-benoit/4d5479b024aa2154bb5e91b19bb1e255/raw/554e779c0f9bce48da00d282fdc2ecd0fe85db2c/Deploy-Default-Sitecore-8-2-3-XM.ps1
3. Copy/paste the code as "**Deploy-Default-Sitecore-8-2-3-XM.ps1**" file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



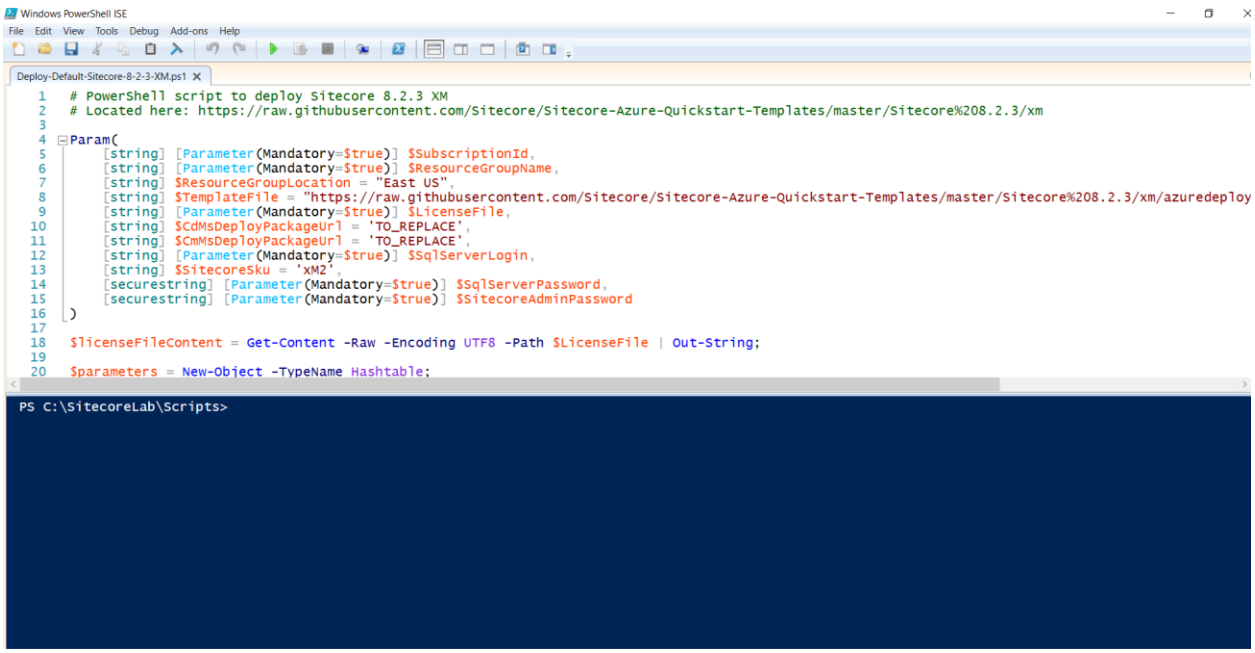
4. *To change the extension of your file, you could activate the "**File name extensions**" option from the "**View**" tab of your Windows Explorer.*



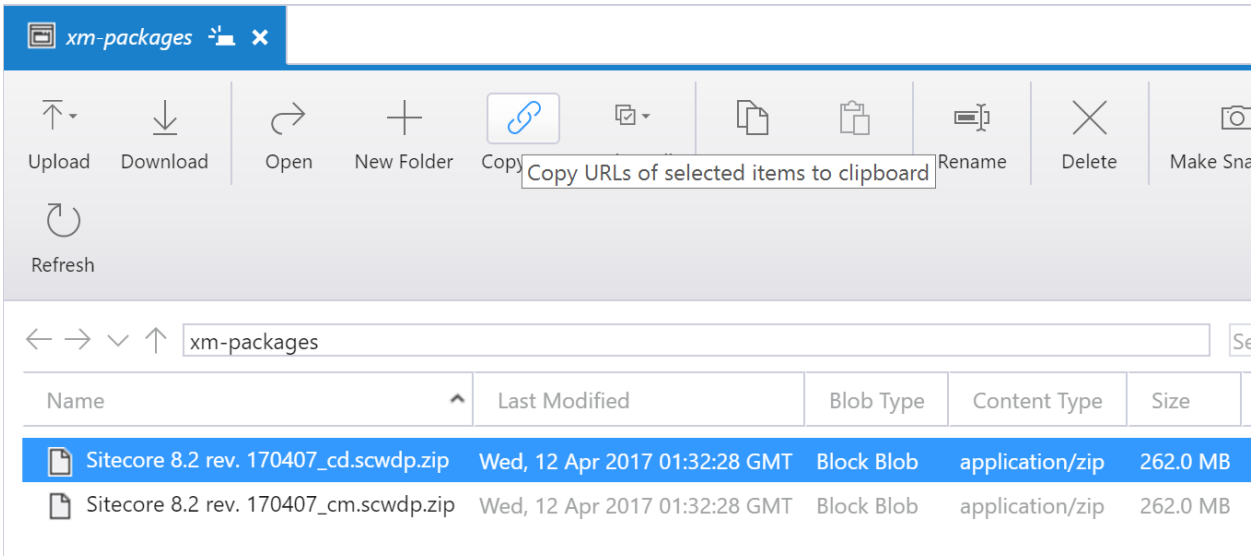
5. Right click on the "**Deploy-Default-Sitecore-8-2-3-XM.ps1**" file and select "**Edit**".



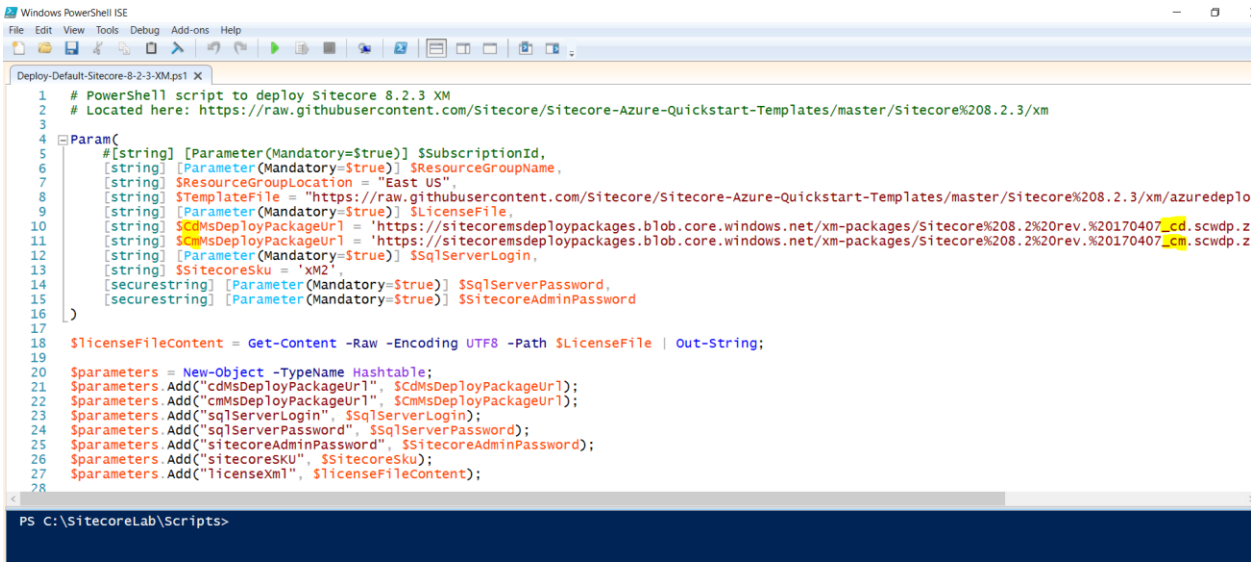
6. The **PowerShell ISE** tool is now opened.



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**:



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

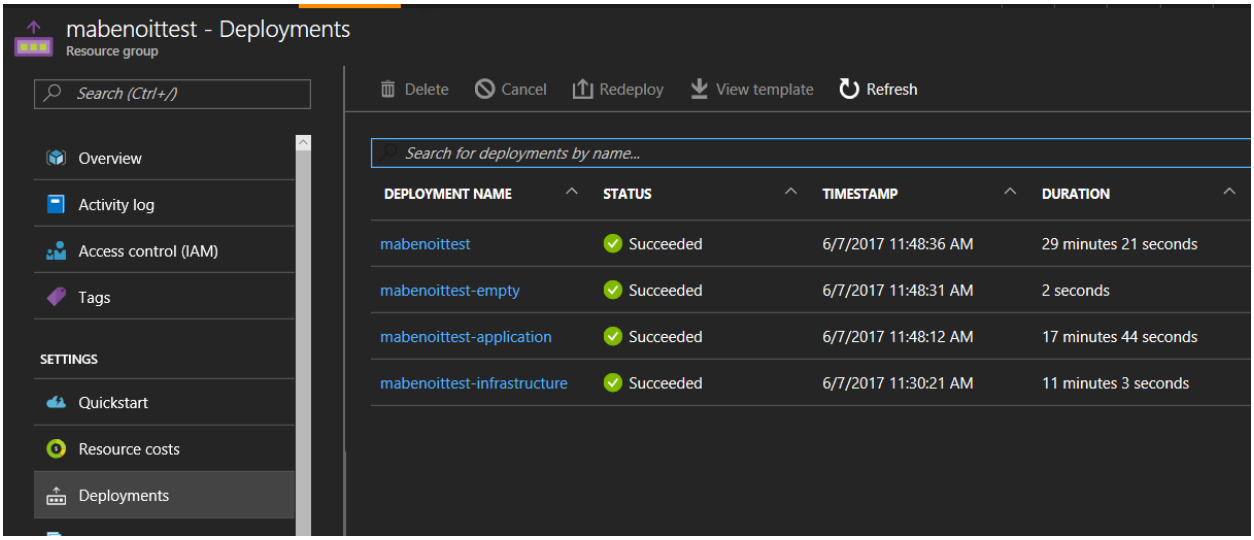


8. Save your changes by clicking on the **Save** icon: 
9. On the bottom window of the PowerShell ISE tool (PowerShell console with blue background), copy/paste the following snippet and execute it with the "**Enter**" key:

```
.\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.*



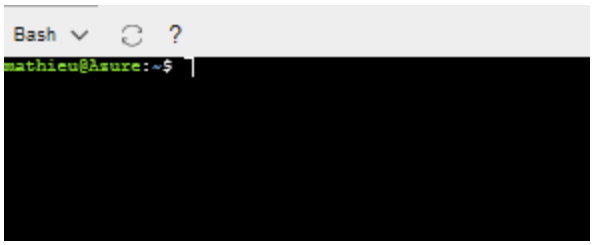
You could start the next tasks below while this deployment is running.

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.

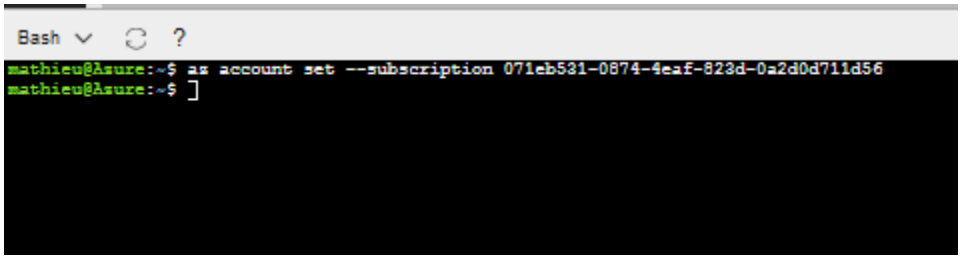
> **Note:** This section could be accomplished by the Azure portal UI as well but here we would like to leverage the Azure Cloud Shell. This new tool/feature became in Public Preview since May 10th 2017, see the announce made during the Microsoft //build 2017 conference.

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:



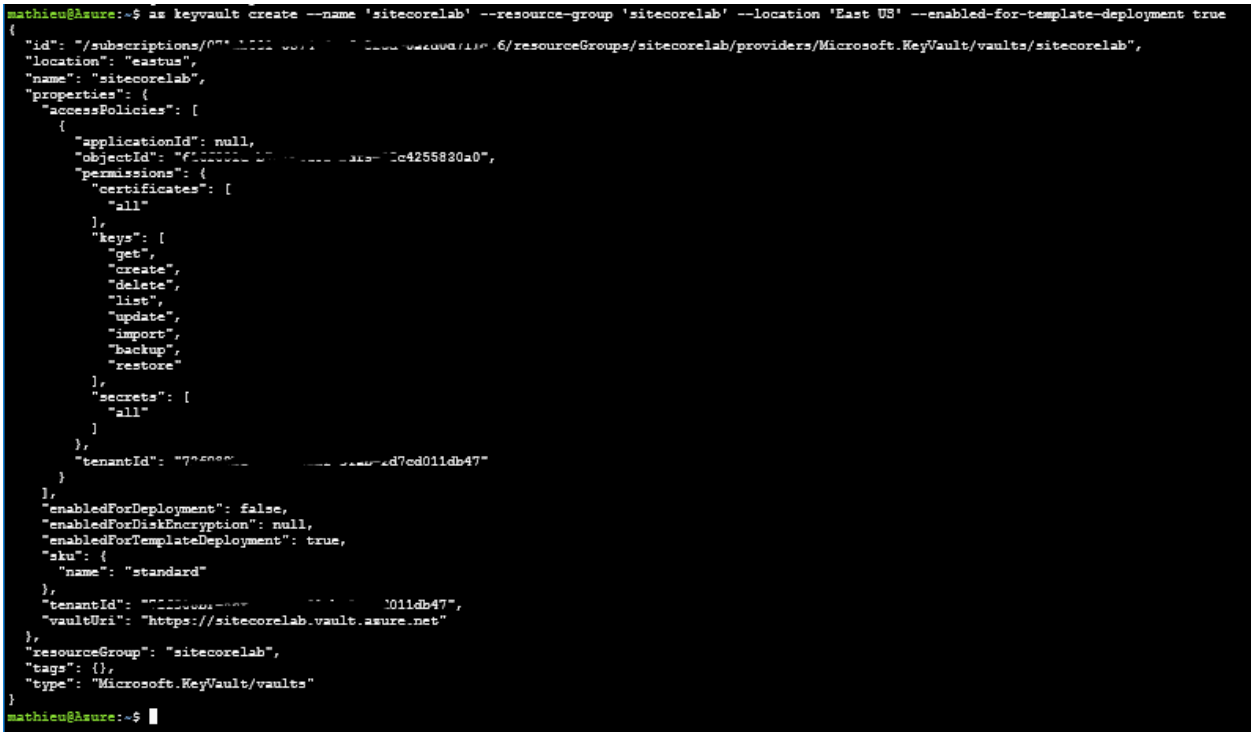
3. Select the right Azure subscription in which you have your "**sitecorelab**" Resource Group by running this CLI command:

```
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
```

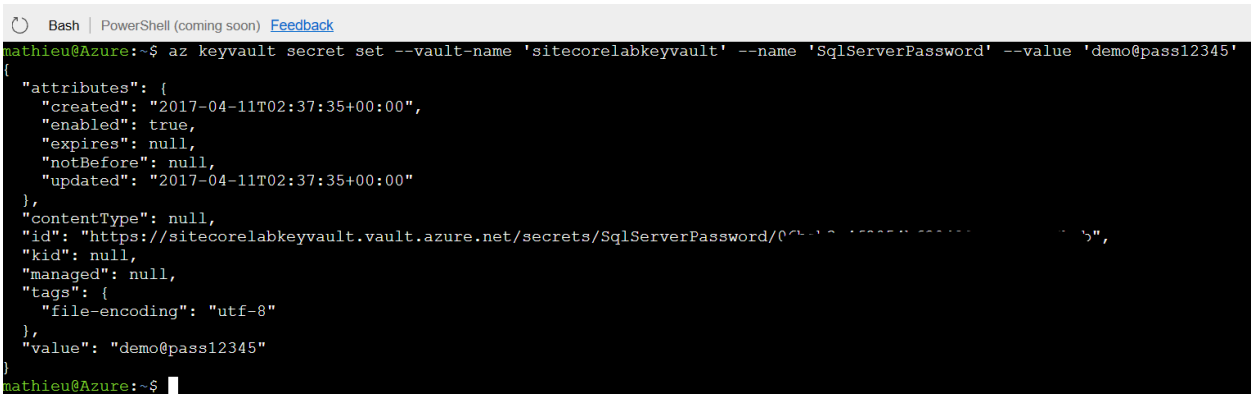


5. Store 2 secrets into this KeyVault just created: `'SqlServerLogin'` and `'SqlServerPassword'`.

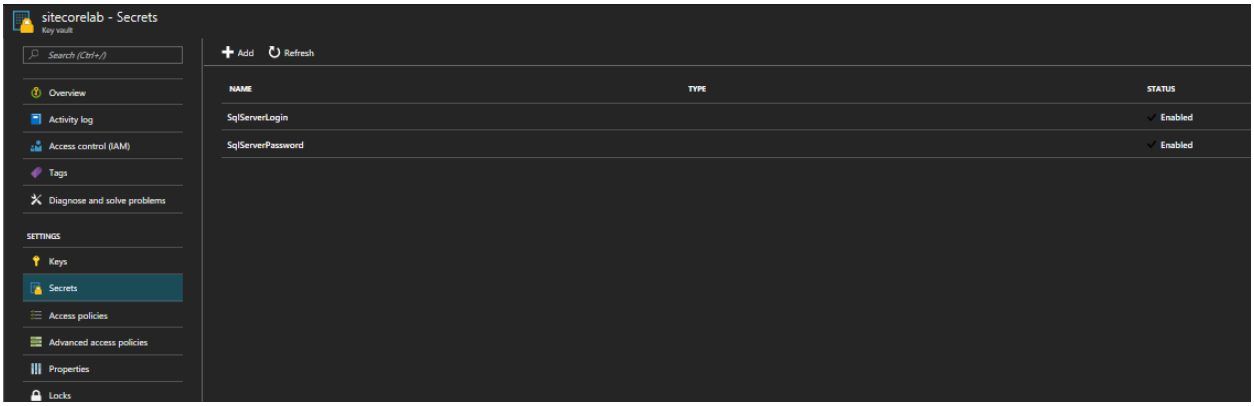
```
az keyvault secret set --vault-name 'sitecorelab' --name
'SqlServerLogin' --value 'demouser'
```

and

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



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



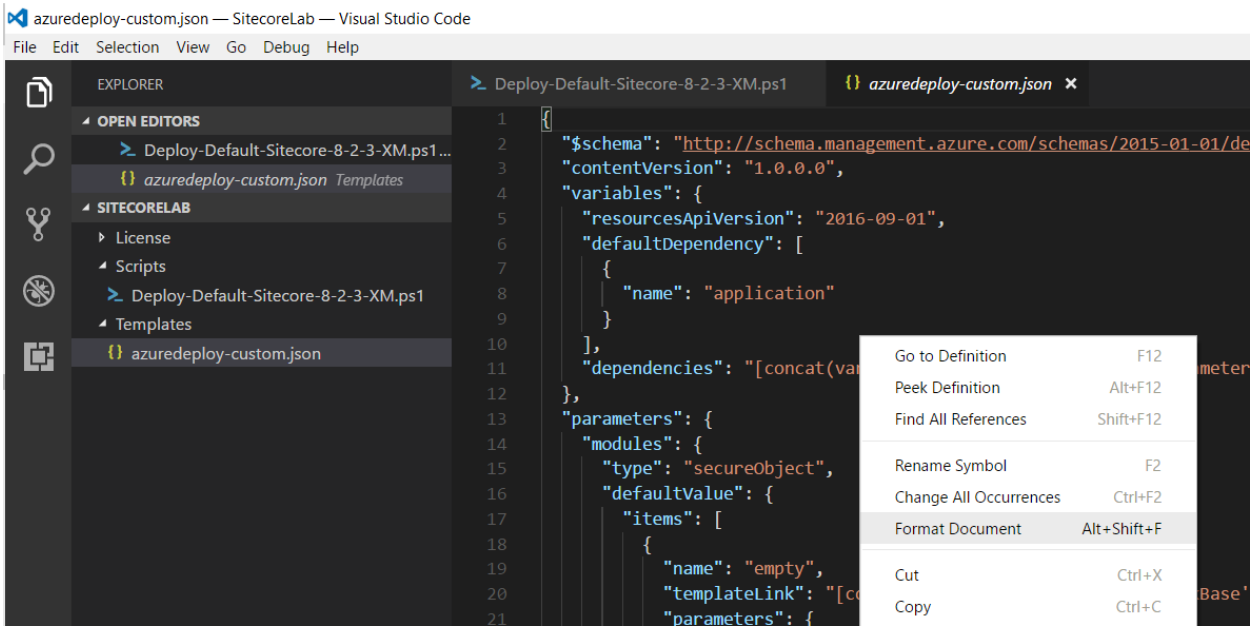
Note: We will use these 2 secrets with the next task with a custom ARM Templates.

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.

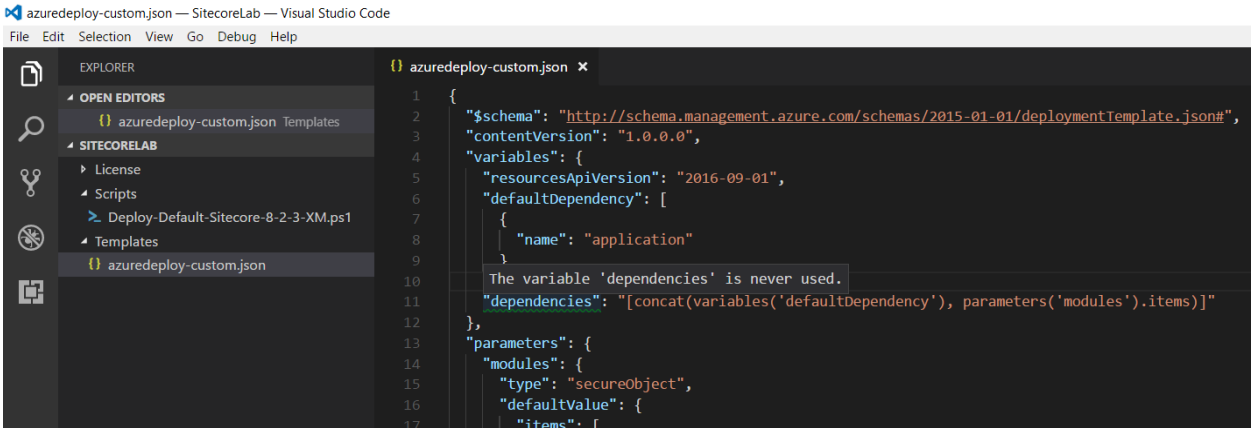
1. With a Windows Explorer, create the folder "**C:\SitecoreLab\Templates**".
2. Go to https://raw.githubusercontent.com/Sitecore/Sitecore-Azure-Quickstart-Templates/master/Sitecore%208.2.3/xm/azuredeploy.json
3. Copy/paste the code as "**azuredeploy-custom.json**" file into "**C:\SitecoreLab\Templates**".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



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.*

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).

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.*

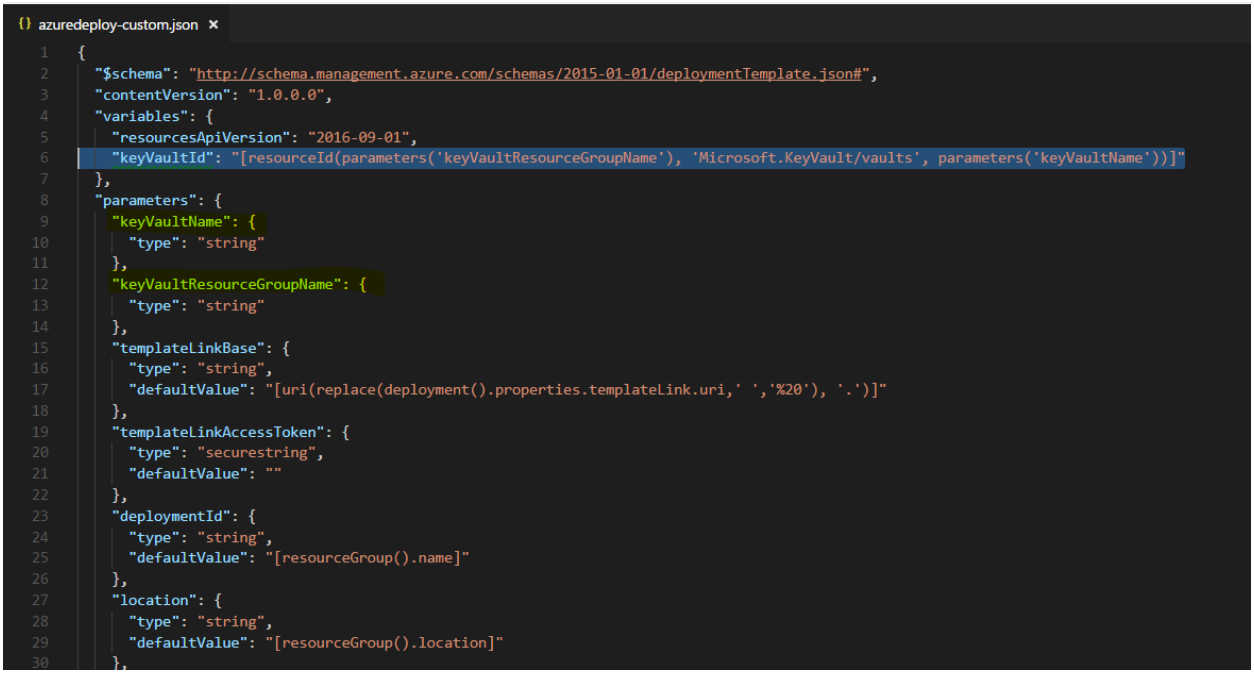


9. Then you could repeat the "**Format Document**" by right-clicking anywhere on this file. You should have now a file with 204 lines:

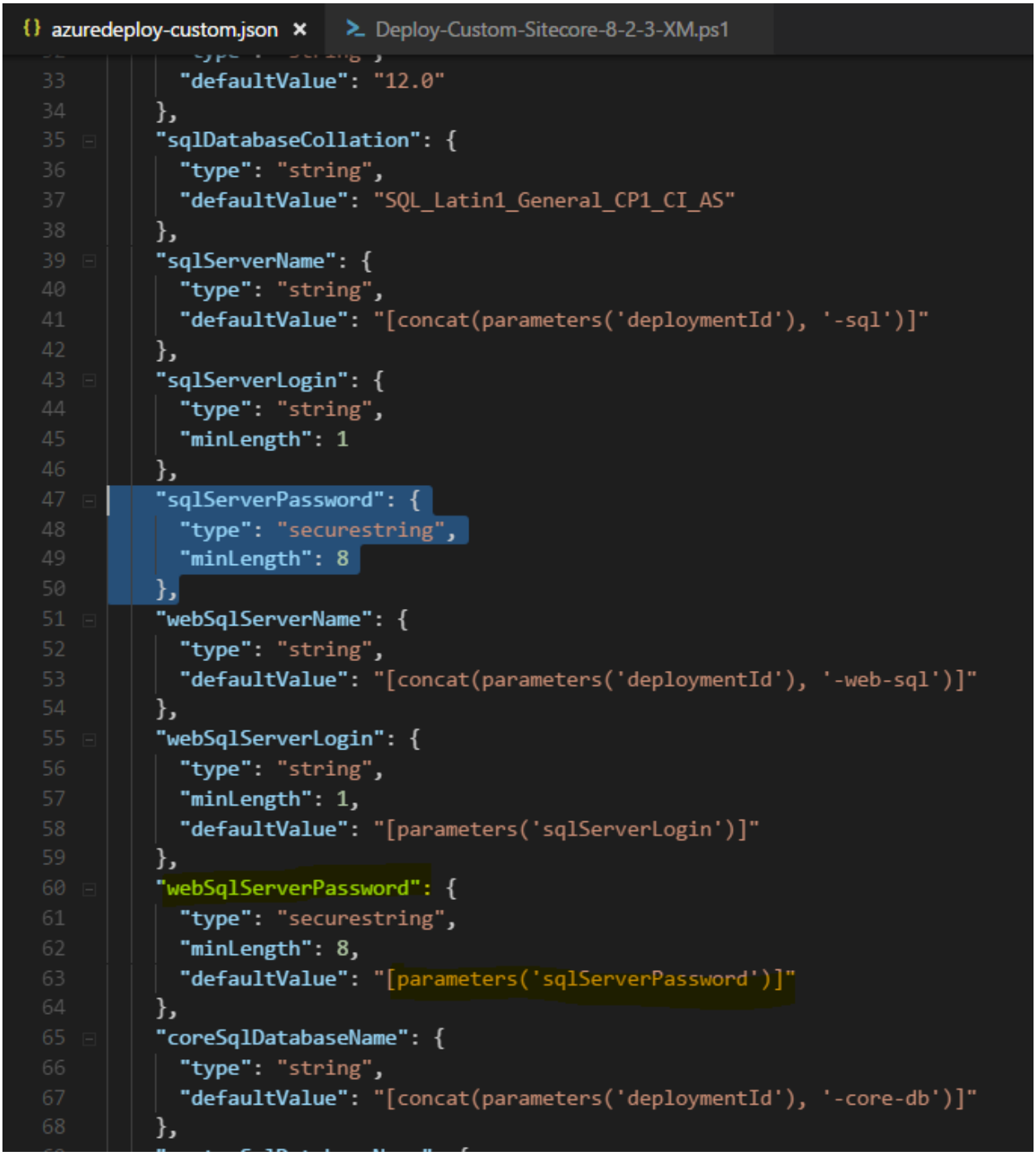
Note: So, now we have a clean ARM Template locally with "`variables`", "`parameters`" and only one "`resources`" to deploy. This resource to deploy is targeting another ARM Template `'nested/infrastructure.json'` which is stored in the official Sitecore GitHub repository here. For this lab, we will use 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.

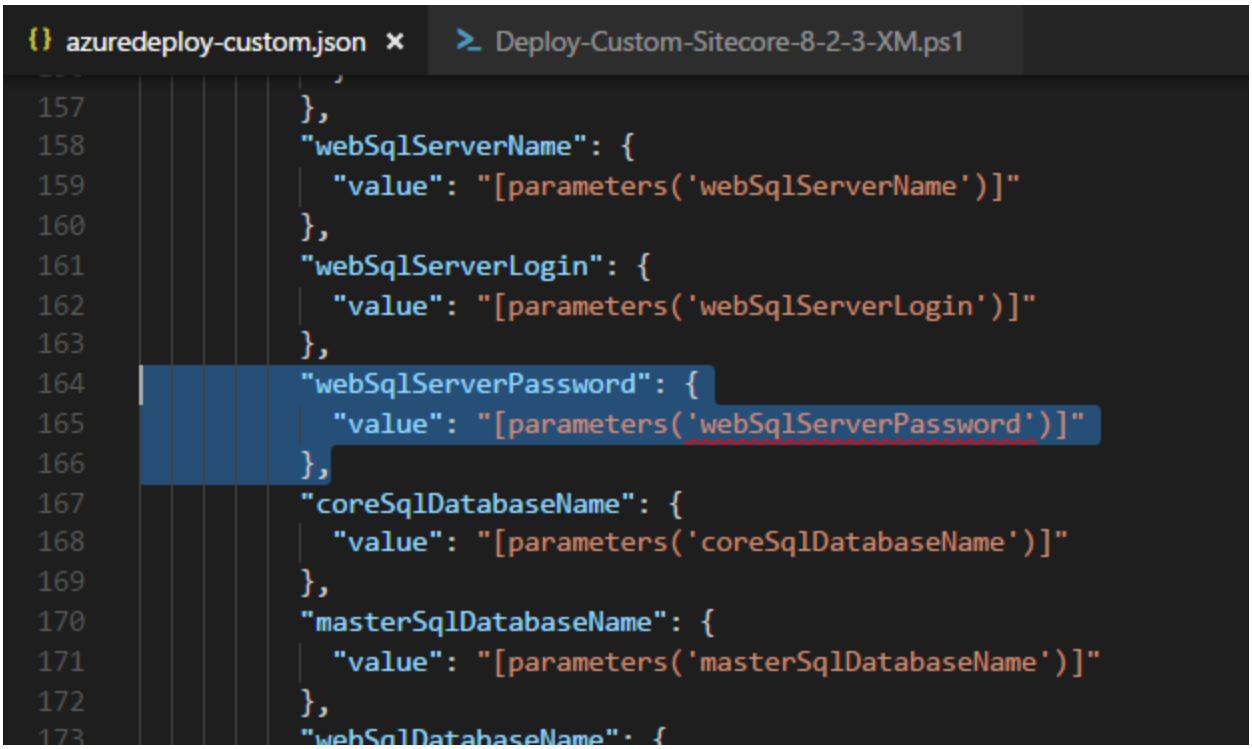
10. *Now let's integrate the KeyVault secrets.* At the top of the azuredeploy-custom.json file, add a `keyVaultId` variable and the two `keyVaultName` and `keyVaultResourceGroupName` parameters:



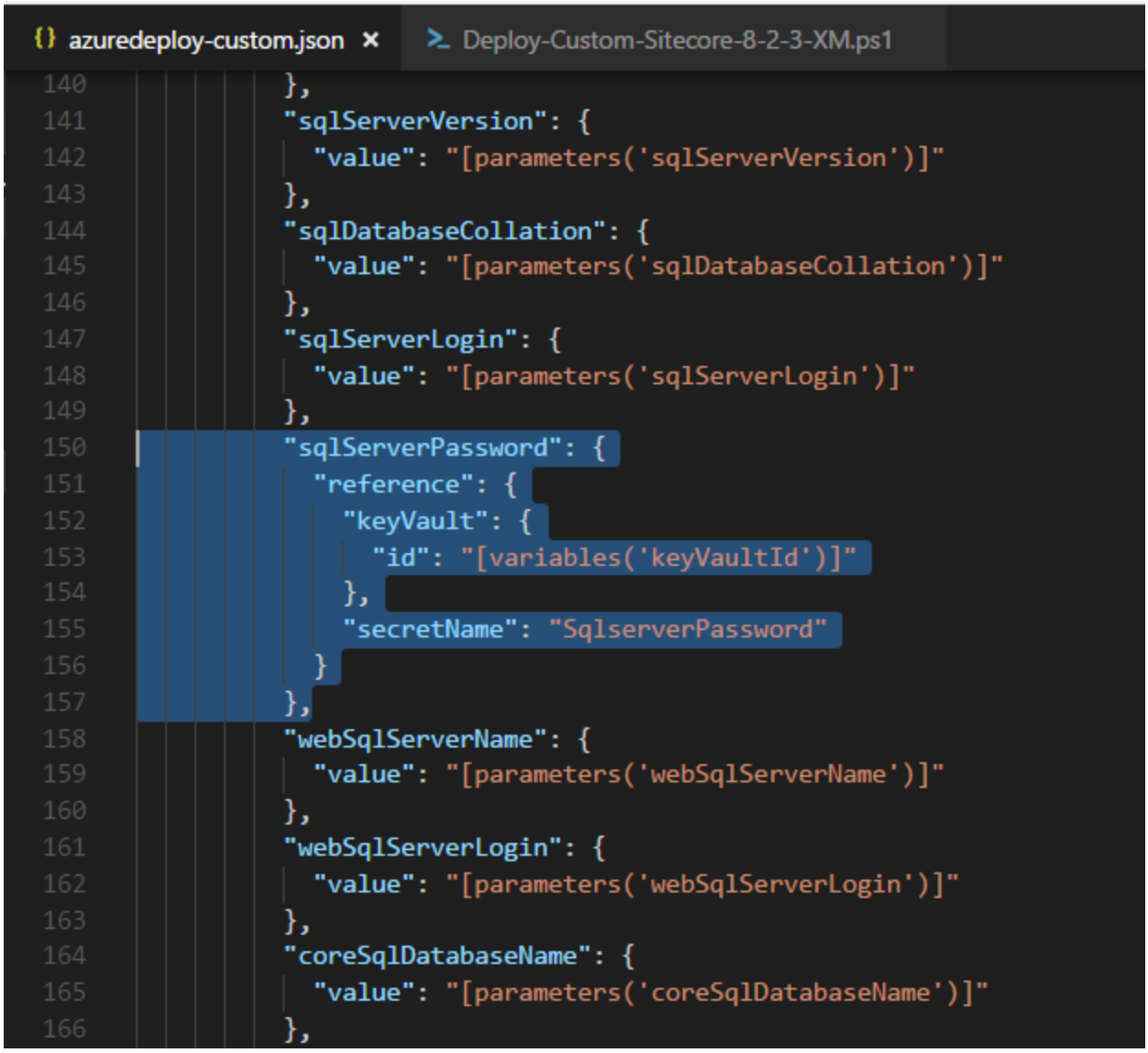
11. Let's remove the parameter "`sqlServerPassword`" and its dependency "`webSqlServerPassword`". *We are not changing anything about the `sqlServerLogin` since this parameter within the nested template is a `string` and not a `securestring`. We could change the nested template to reflect that, but that's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



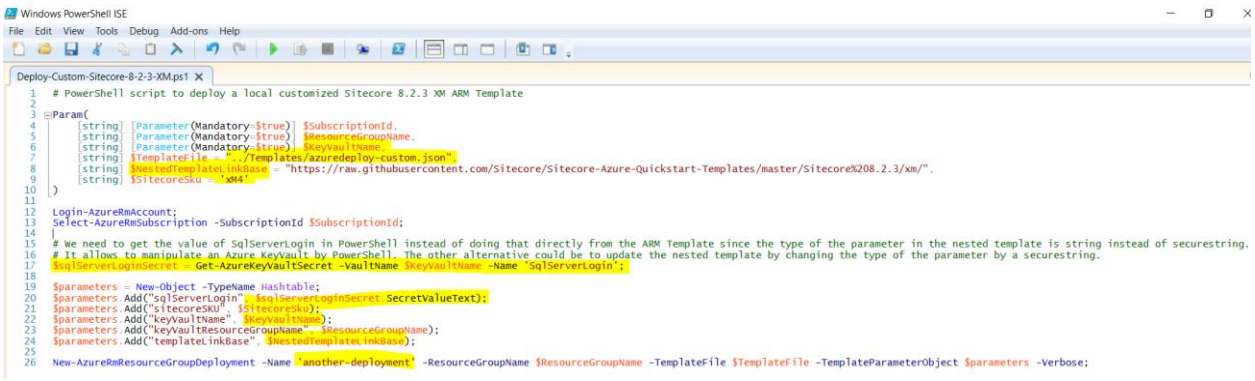
12. Let's fix the errors more below in this template. You could remove the lines from 164 to 166. If you look at the nested ARM Template, this parameter will use as defaultValue `sqlServerPassword`.



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



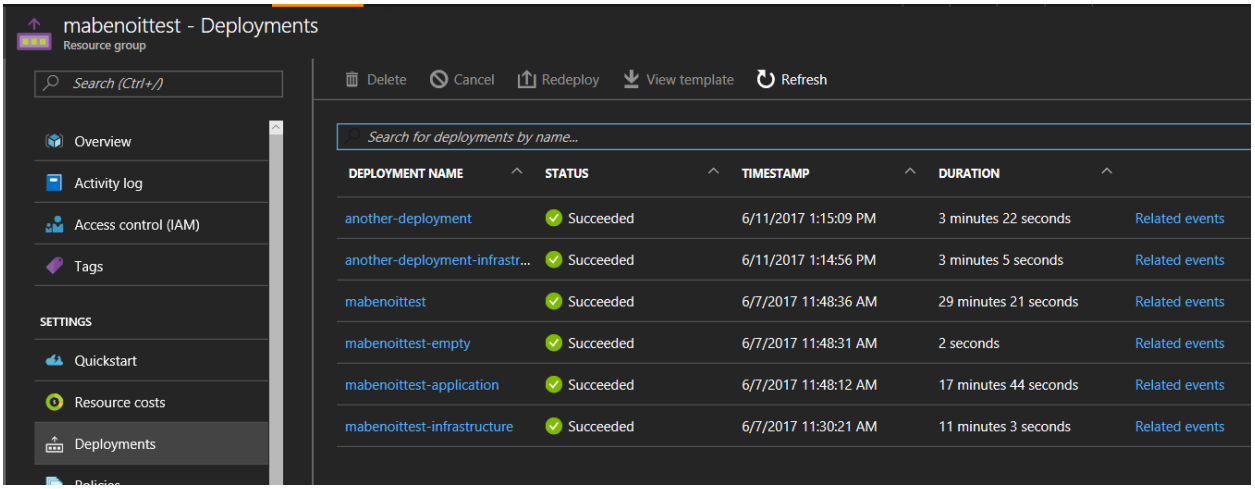
14. Duplicate the C:\SitecoreLab\PowerShell\Deploy-Default-Sitecore-8-2-3-XM.ps1 and rename it by **Deploy-Custom-Sitecore-8-2-3-XM.ps1**.
15. Right-click on it and select "**Edit**". Then "**PowerShell ISE**" tool will be opened. Here, let's adapt this script.
16. The result file should look like illustrated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



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.*



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.

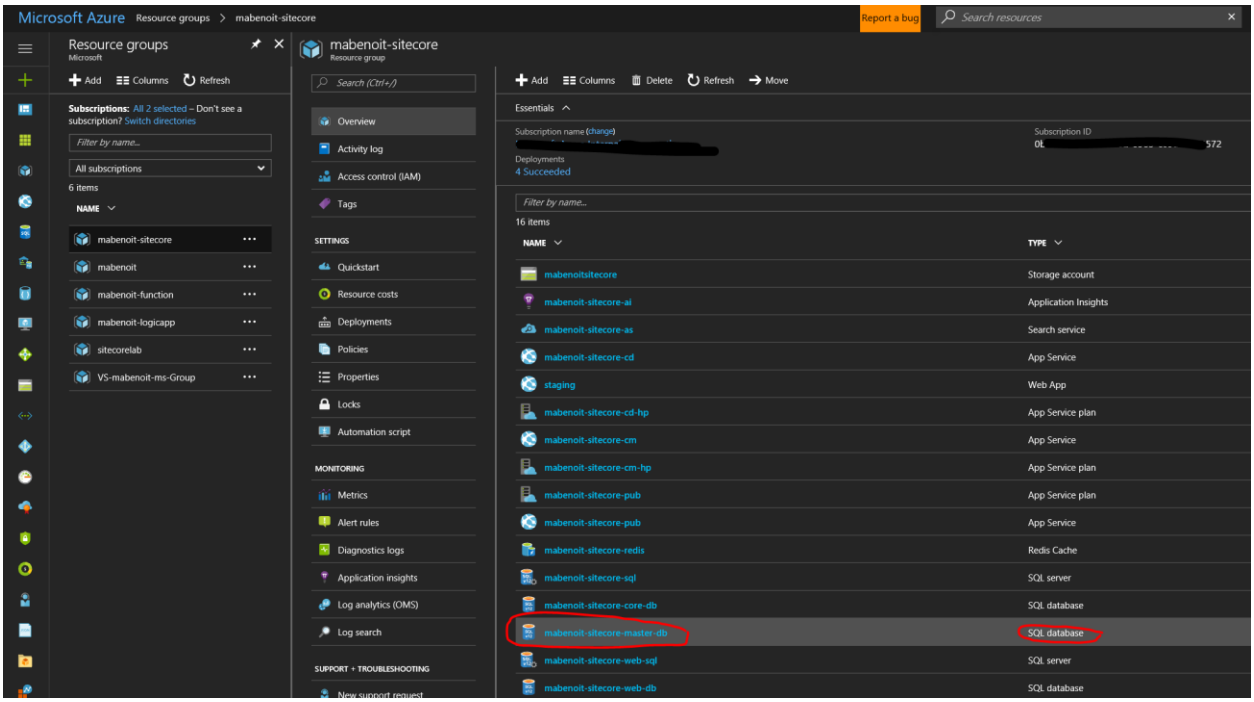
Through this exercise, you will play/use with:

- **Azure portal**
- **Azure SQL Database**
- **Visual Studio**

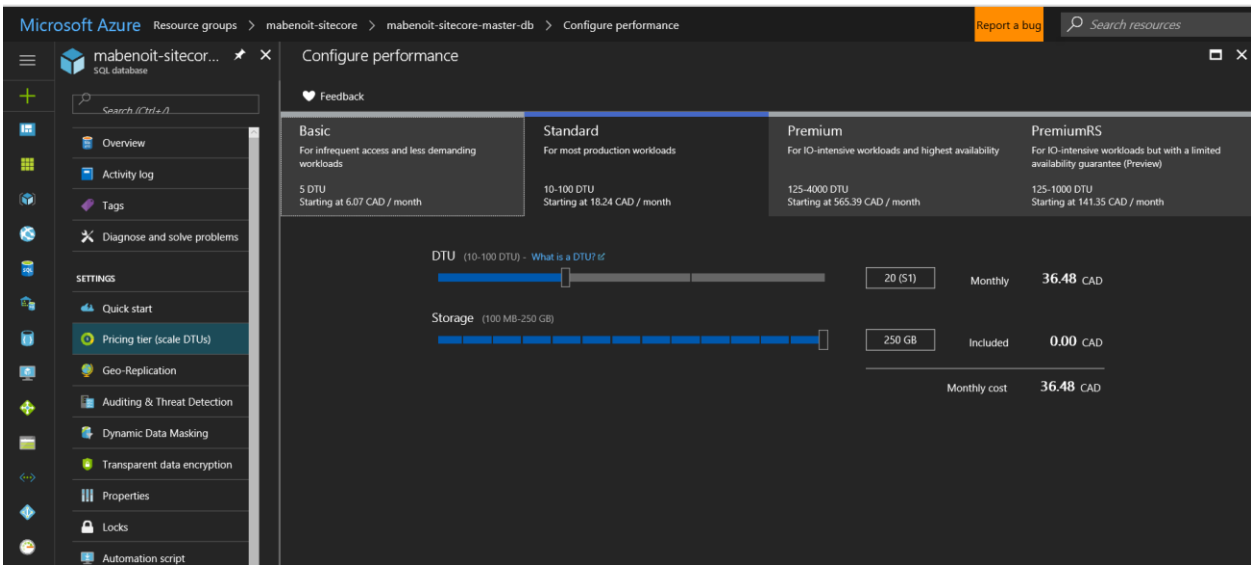

Task 1: Scale up/down your databases (5 min)

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

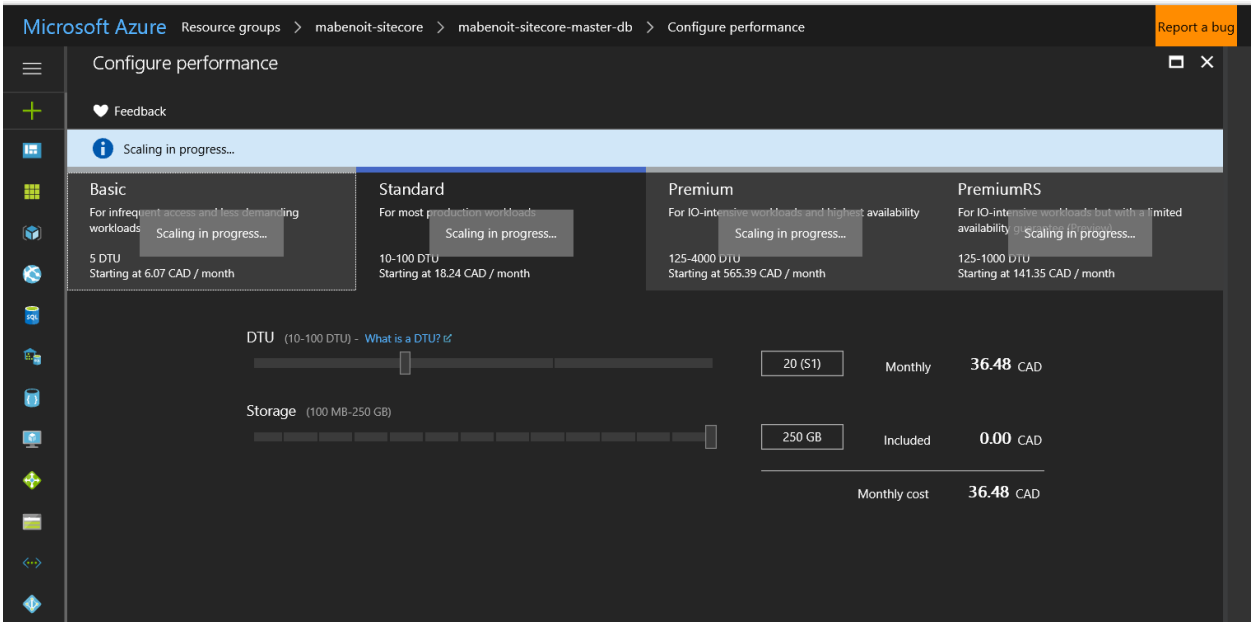
1. Go to https://portal.azure.com
2. Open the Resource group containing all the Azure services previously deployed. Then select the "**SQL database**" with the suffix name "-master-db".



3. Then navigate to the "**Pricing tier (scale DTUs)**" blade.



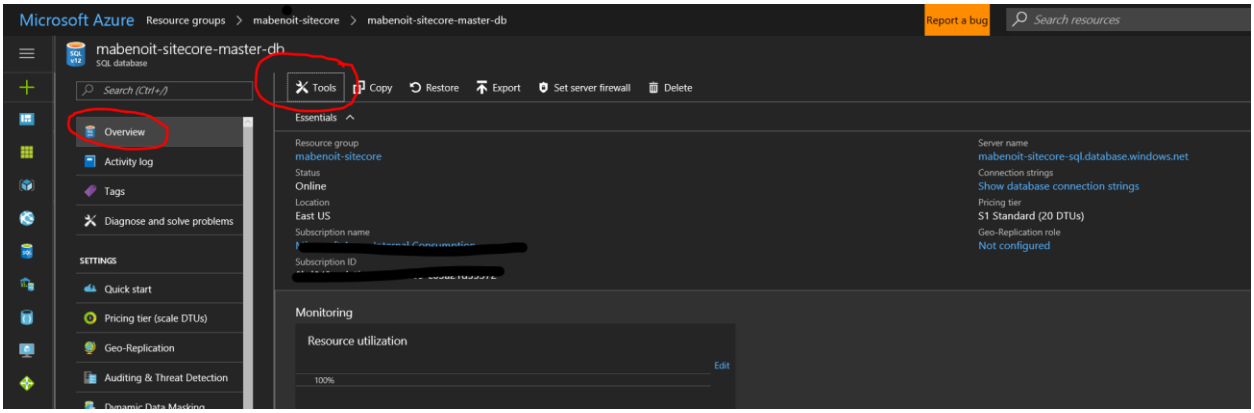
4. There you could change the Pricing tier value up or down, and click on the "**Apply**" button. After few seconds, a "**Scaling in progress…**" message should appear. Then, few seconds/minutes after the scaling process is done.



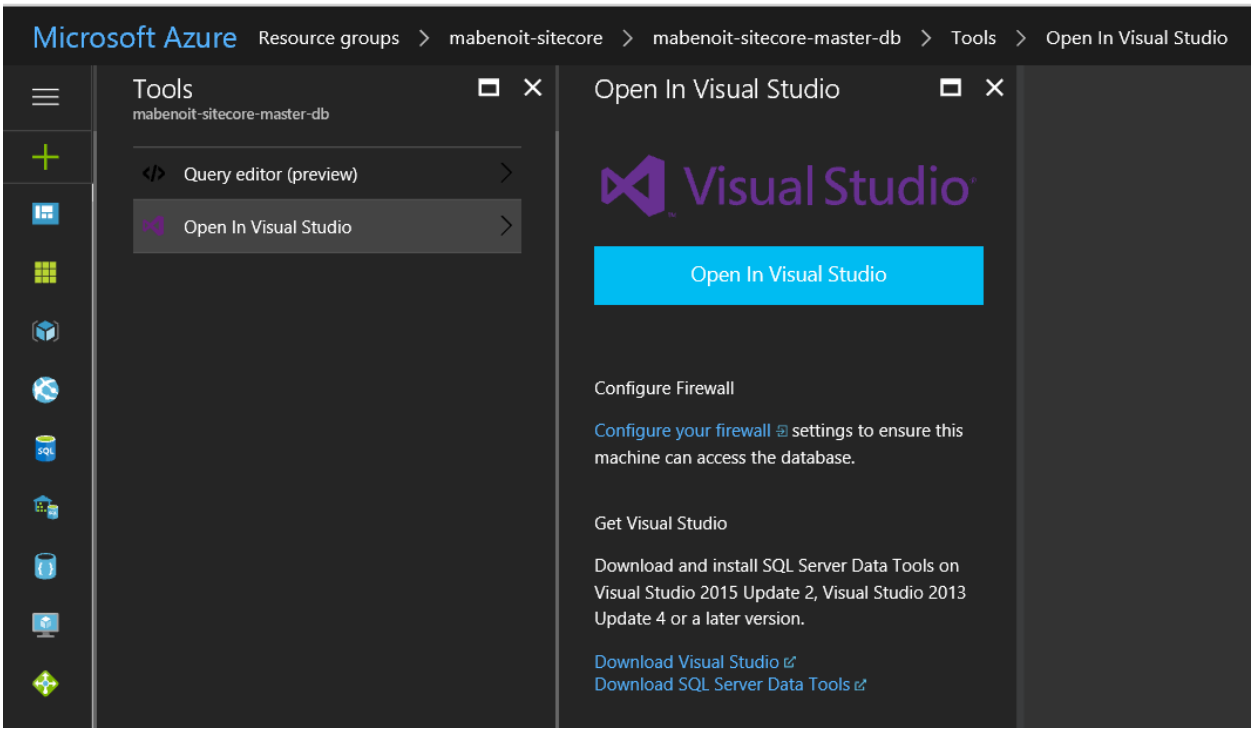
Task 2: Local connection with Visual Studio (7 min)

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.

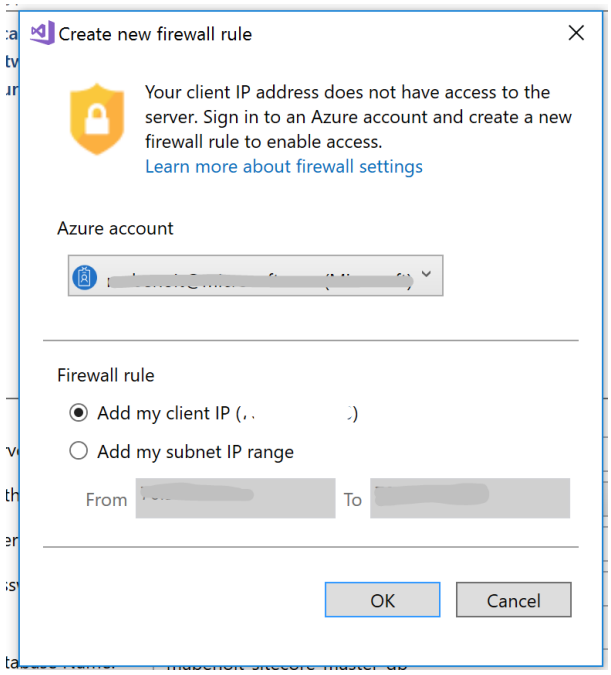
1. Go to the "**Overview**" blade of this "…-master-db" database.



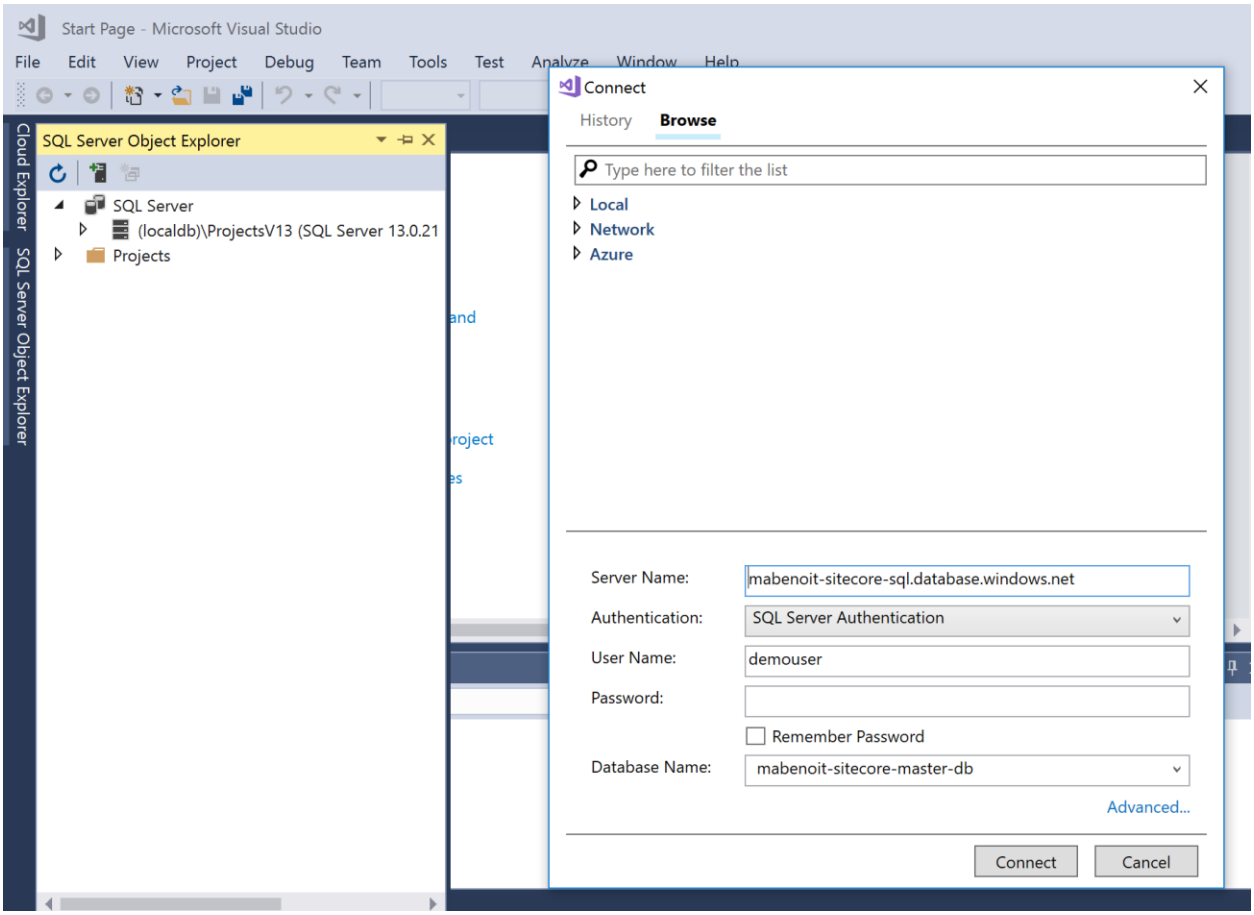
2. Then click on the "**Tools**" toolbar action and choose "**Open in Visual Studio**" to launch your local Visual Studio Community 2017.



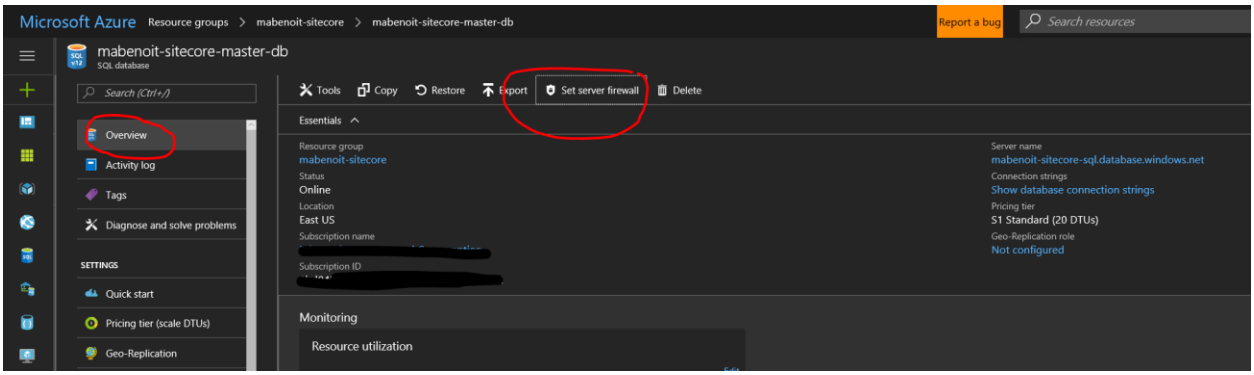
3. In Visual Studio, the "**Create new firewall rule**" will be prompted. You could use the 2 options proposed, but for this lab we will do that through the Azure portal. So, click on the "**Cancel**" button.



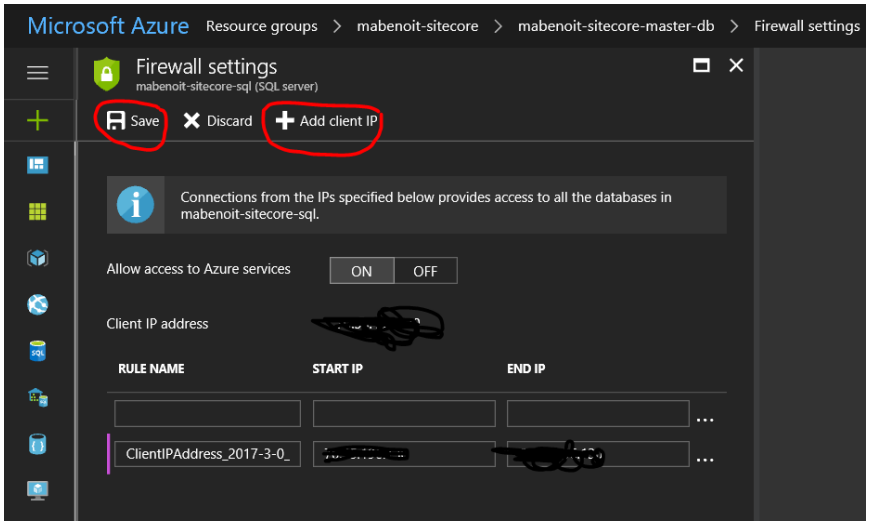
4. You will be asked to enter the associated password; all the other information will be populated automatically for you.



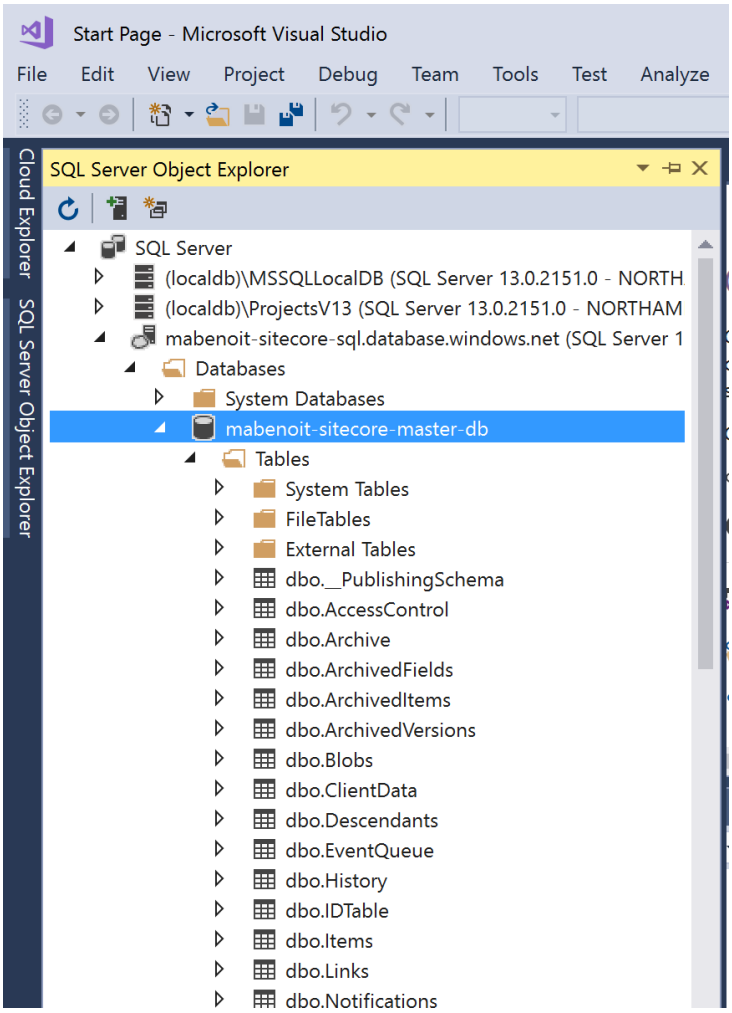
5. Enter the associated password, if you followed the instructions earlier on this lab, it should be “demo@pass12345”. Then click on the “**Connect**” button.
6. Depending on which tool you are using you could have an error message saying that you don’t have the permission to connect on this server for 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.



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



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.

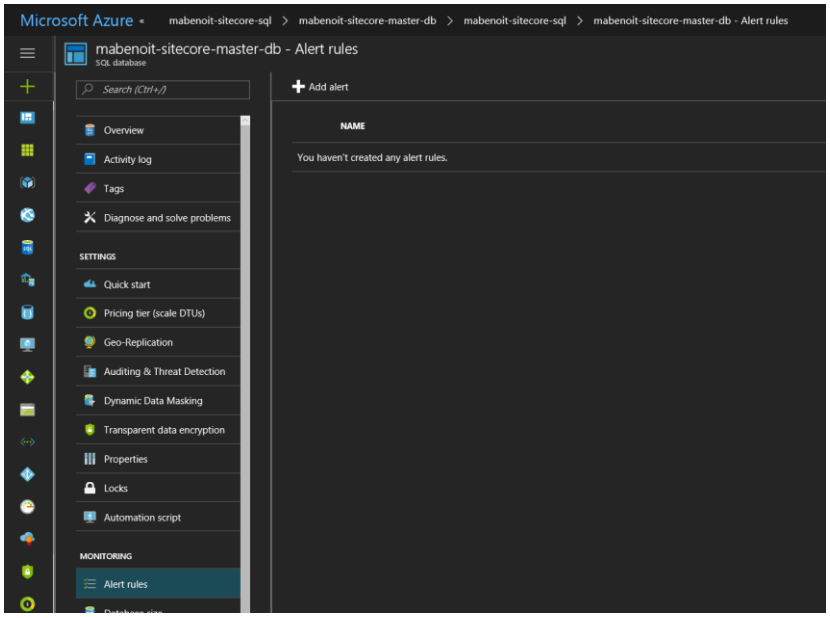


Note: Instead of Visual Studio you could connect to your Azure SQL Database locally via other tools like SQL Server Management Studio (SSMS). Furthermore, setting the correct Firewall rules will allow you to locally sync your Sitecore content via tools like Sitecore Rocks, TDS or Unicorn for example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

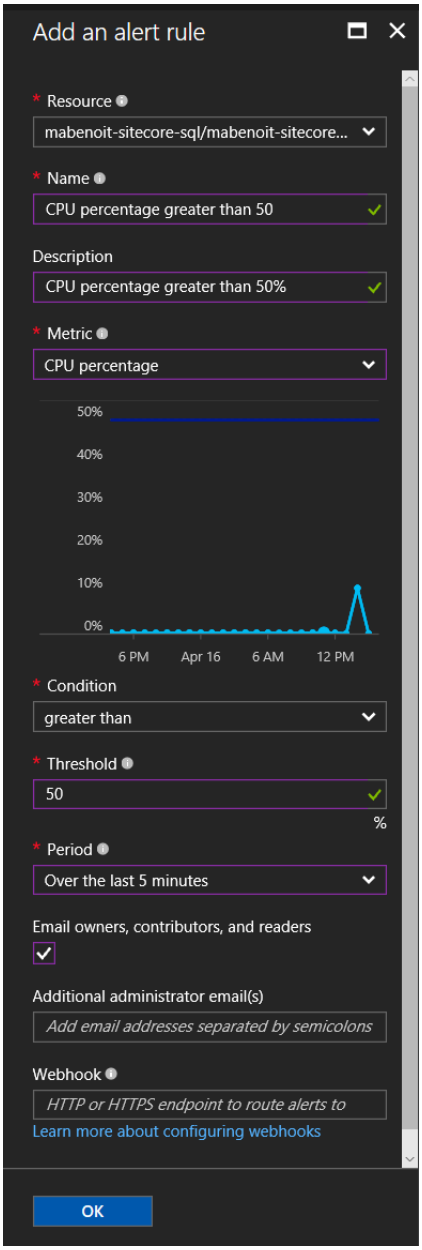
Task 3: Add Alert Rules (3 min)

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*"

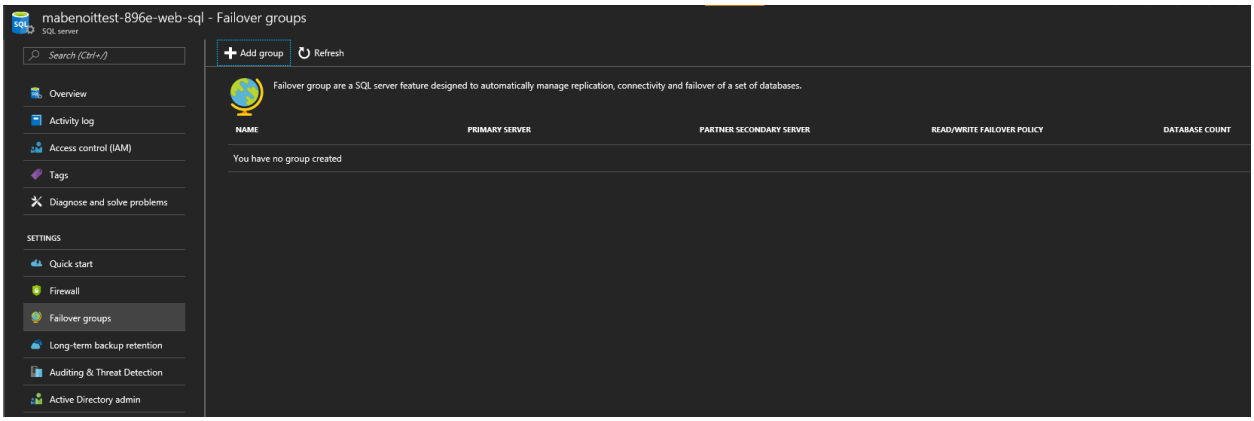


4. Then click on the "**OK**" button and your alert will be in place.

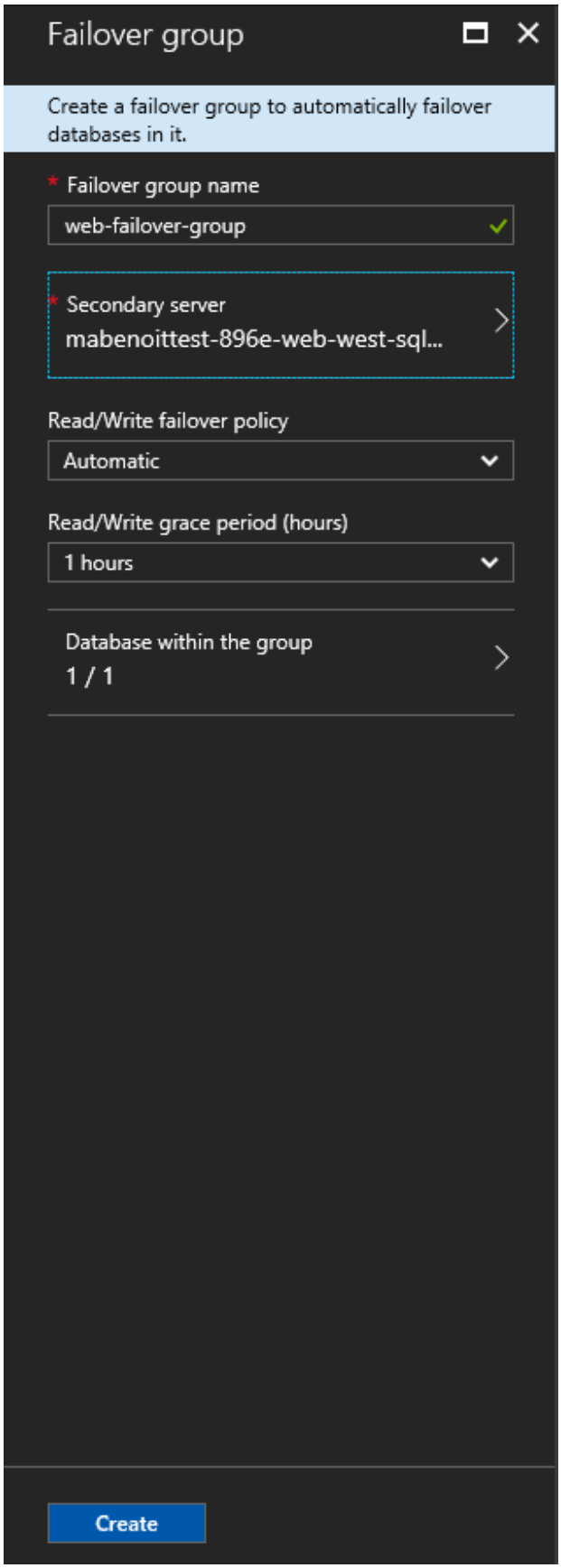
Task 4: Configure a Geo-replication (5 min)

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.

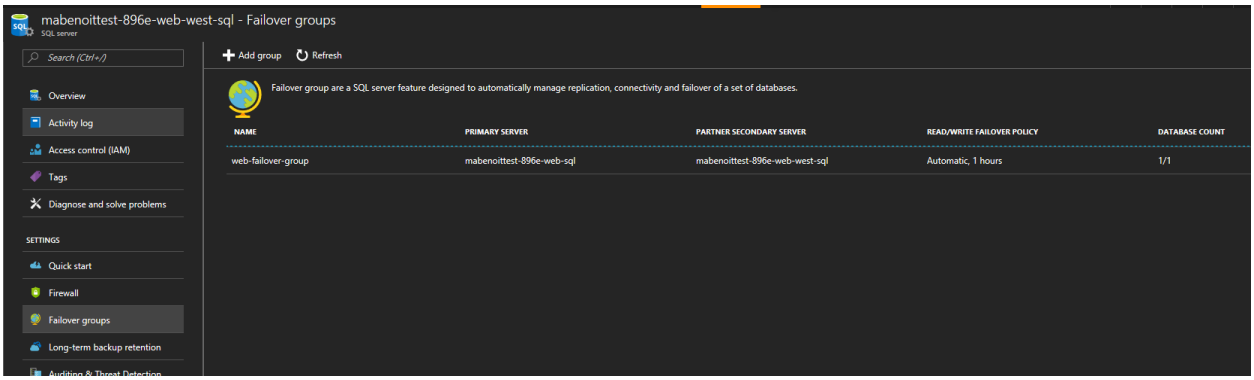
1. Go to the web SQL Server resource named "…-web-sql" on the Azure portal and click on the "**Failover groups**" blade.



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:



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

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.

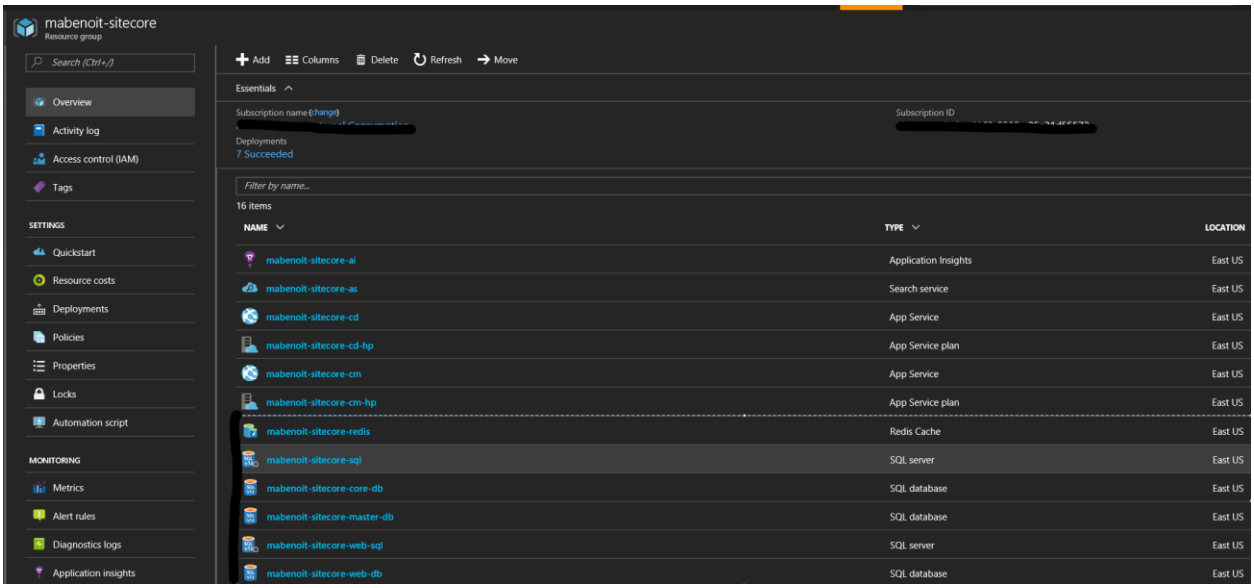
Through this exercise, you will play/use with:

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

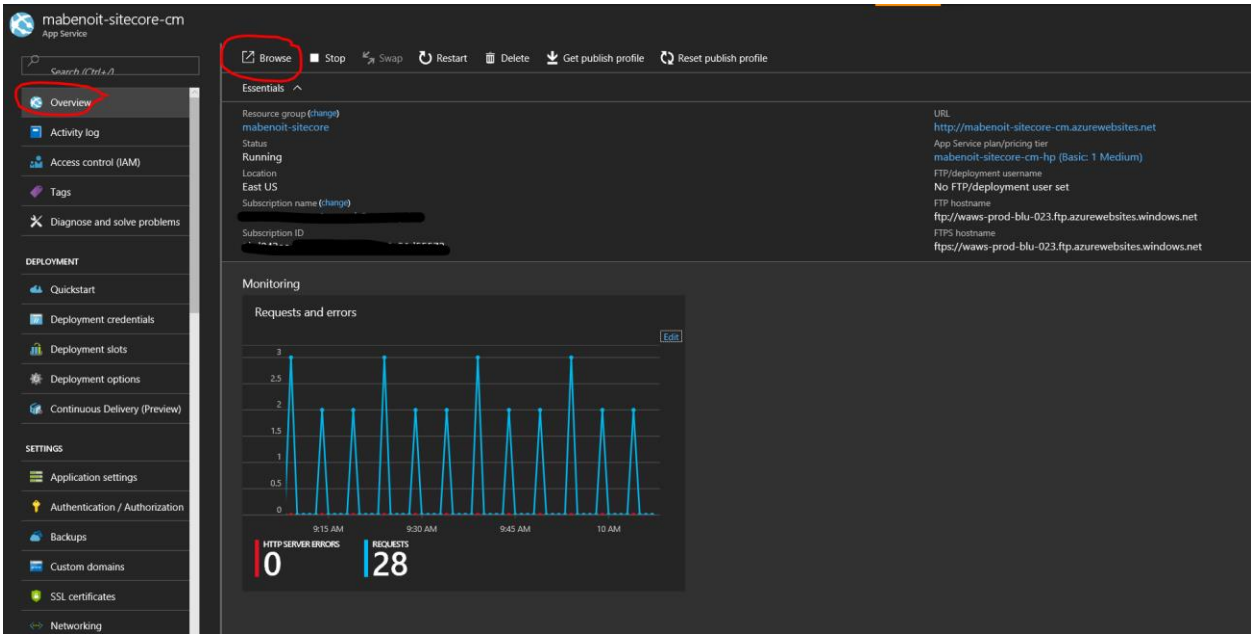
Task 1: Verify the Sitecore solution previously provisioned (5 min)

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.

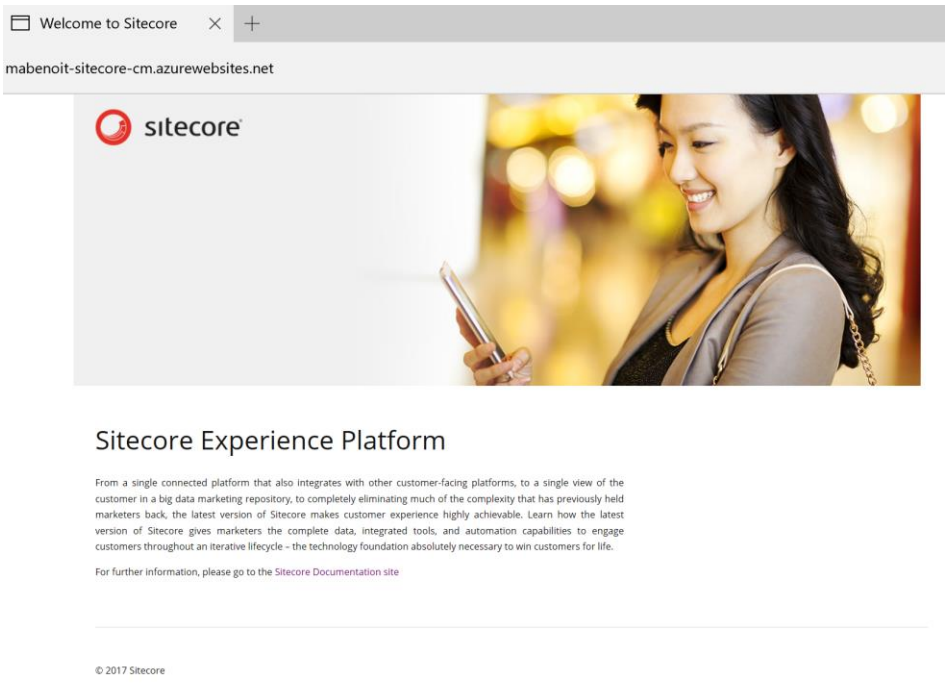
1. Go to the Azure portal - https://portal.azure.com and open the Resource Group containing all the services provisioned earlier. You should see all the services below:



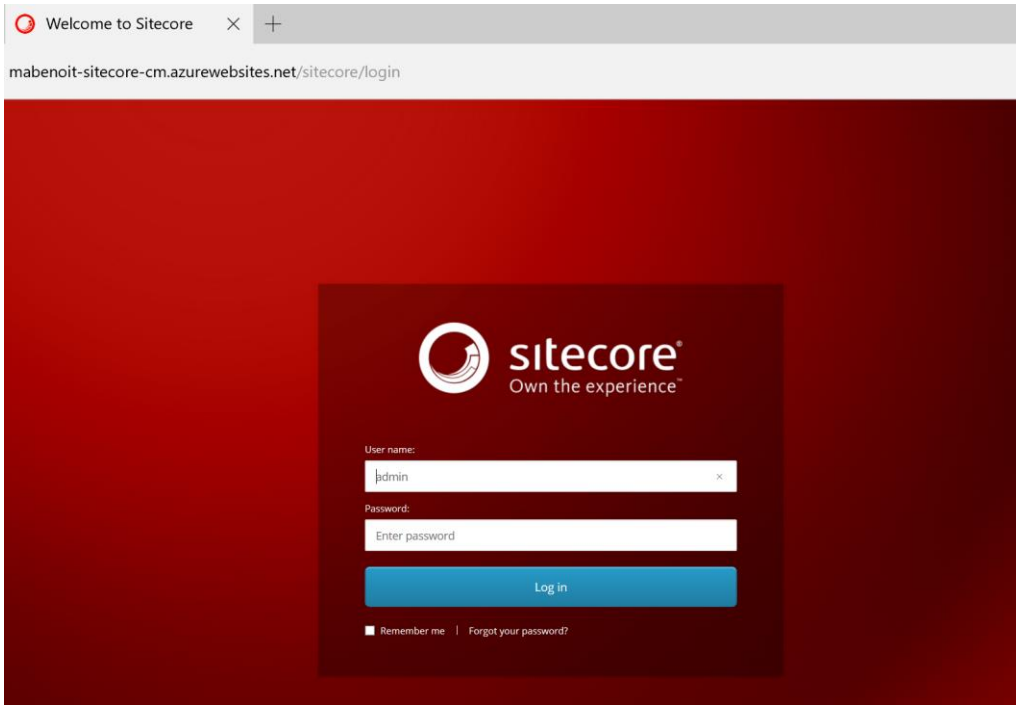
2. Open the "…-cm" Web App Service resource and there on the "**Overview**" blade click on the "**Browse**" toolbar button.



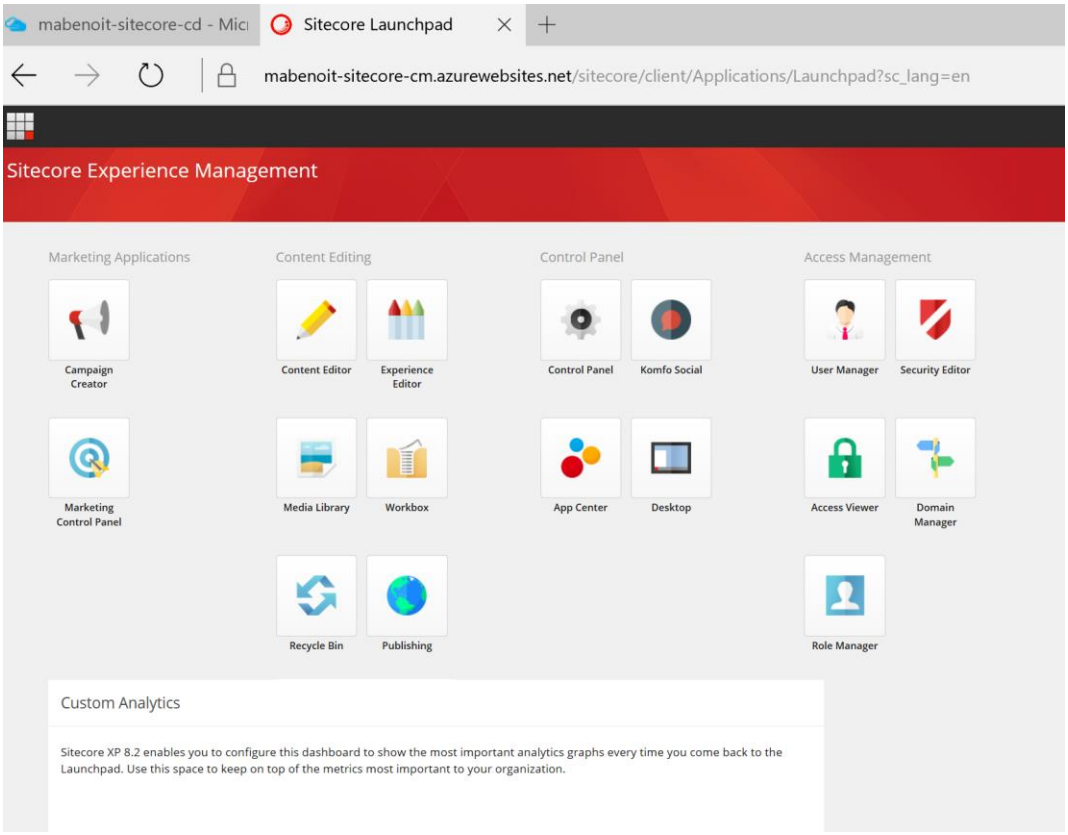
3. You should see the homepage of the Sitecore Vanilla version:



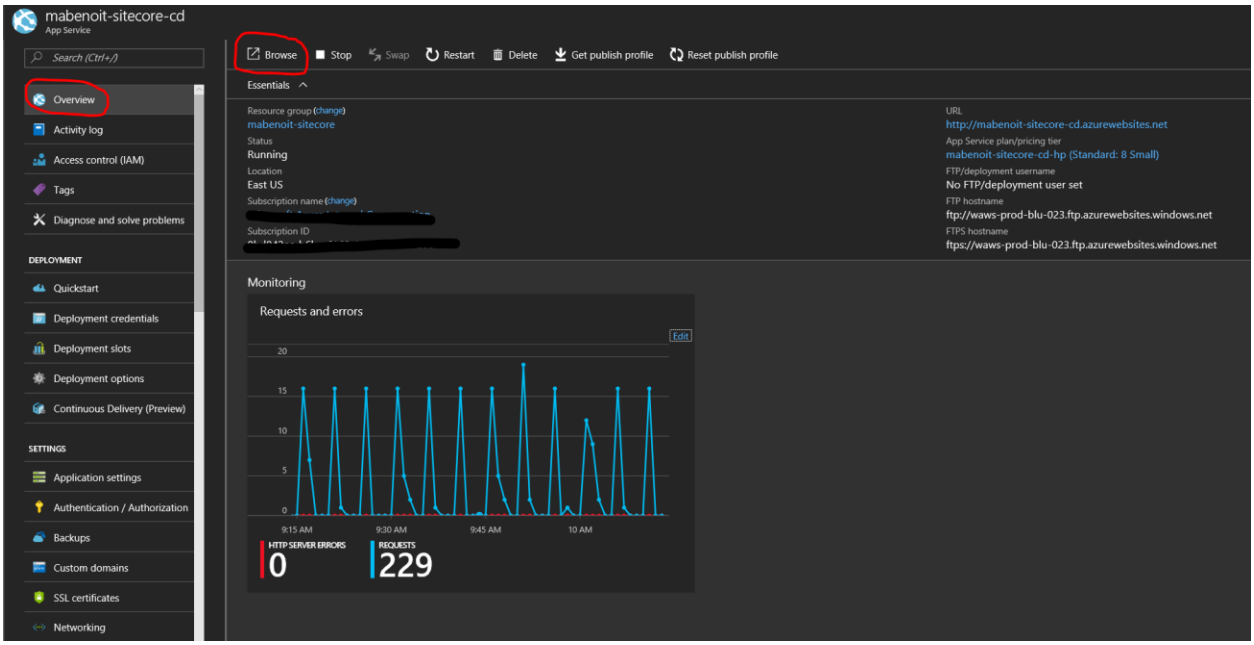
4. On this web page, add at the end of the URL the **"/sitecore**" suffix. We will be landed on the Sitecore administration login page.



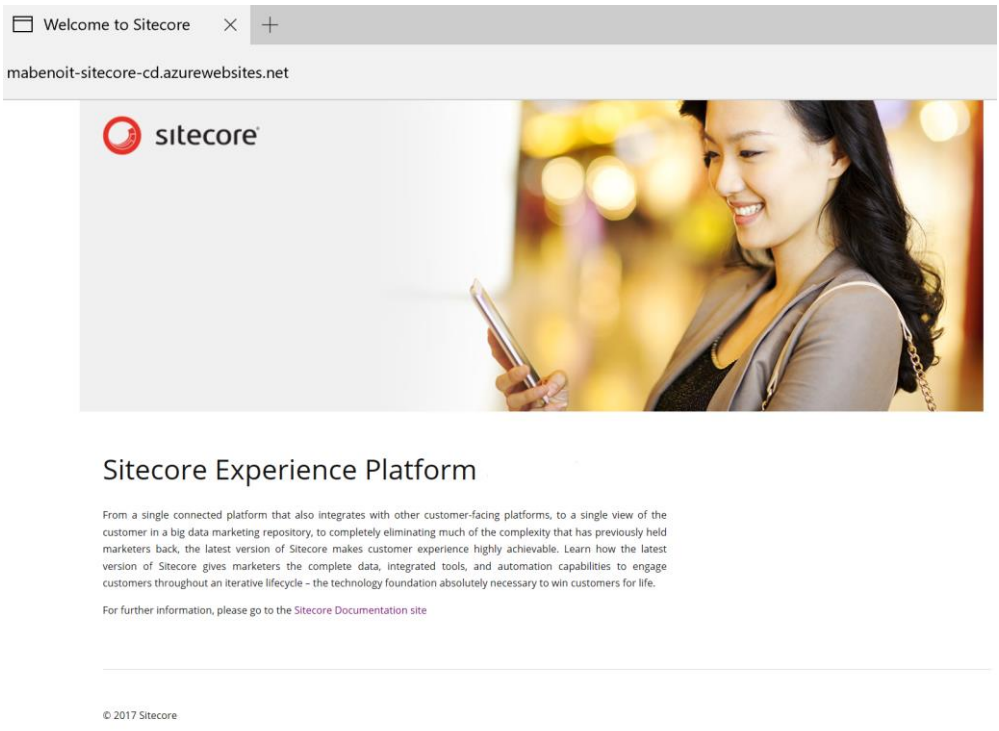
5. If you enter the credentials below, you should be landed on the Sitecore Launchpad page.
    a. **User name**: admin
    b. **Password**: demo@pass12345



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.



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

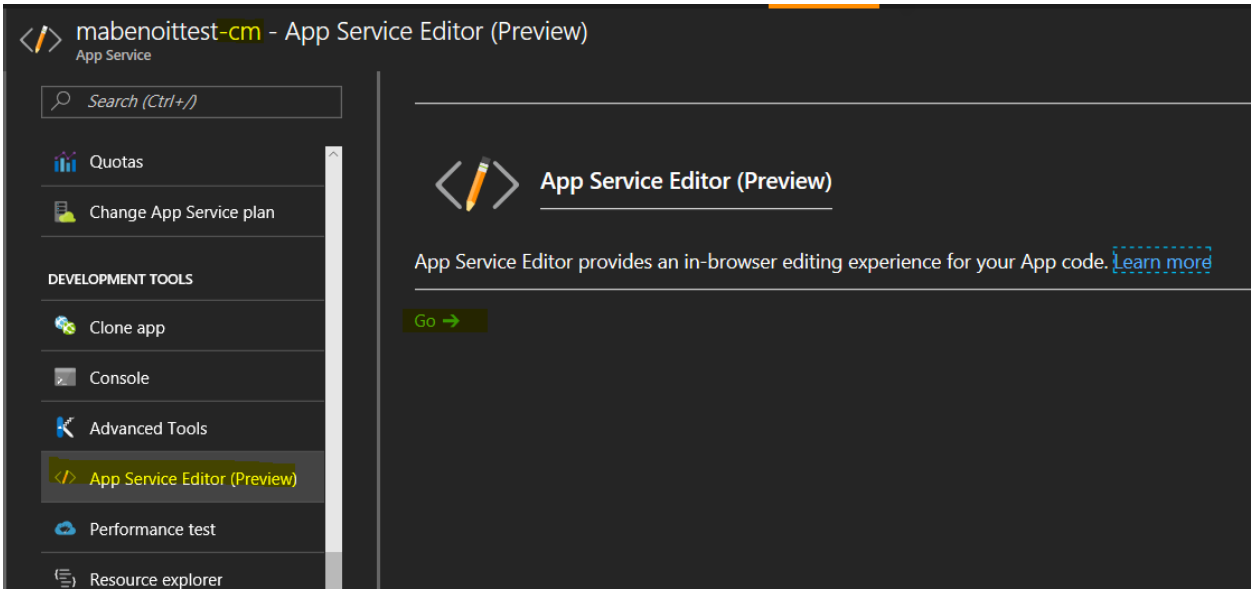


Note: This checkpoint is important for the rest of the lab if you have any issue or difference with the steps below, please ask your lab proctor.

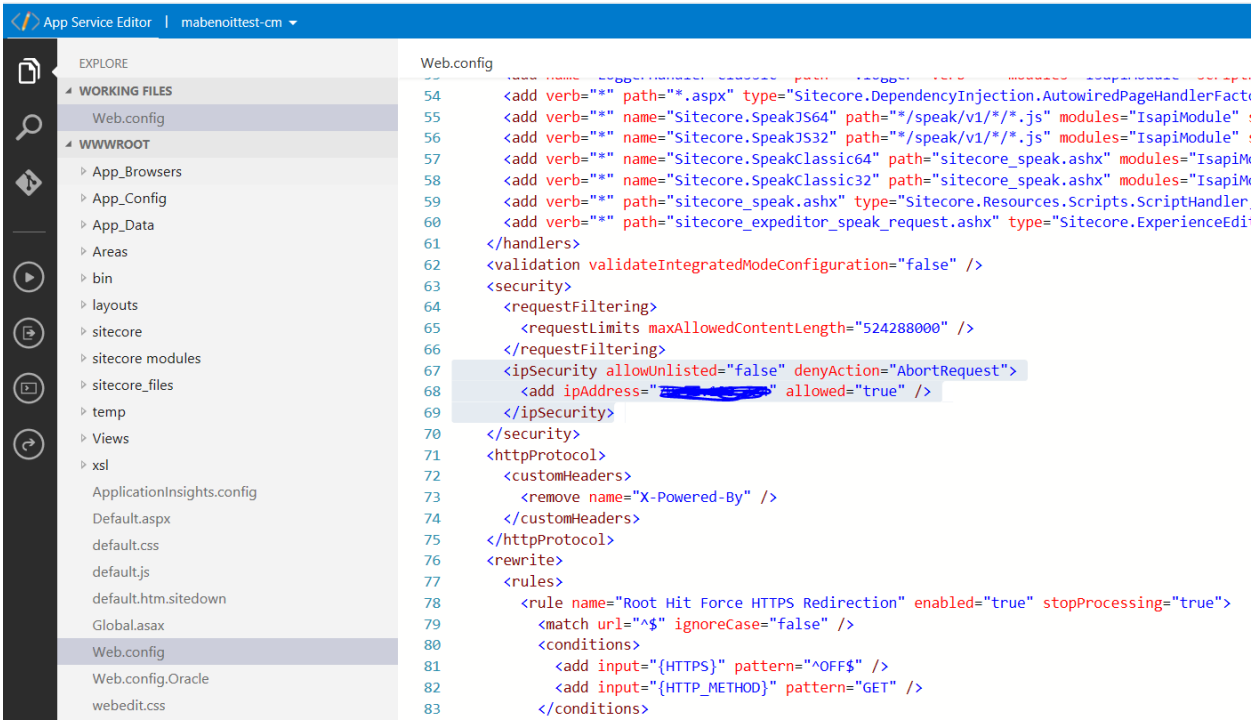
Task 2: Add IP restriction on the CM instance (5 min)

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:



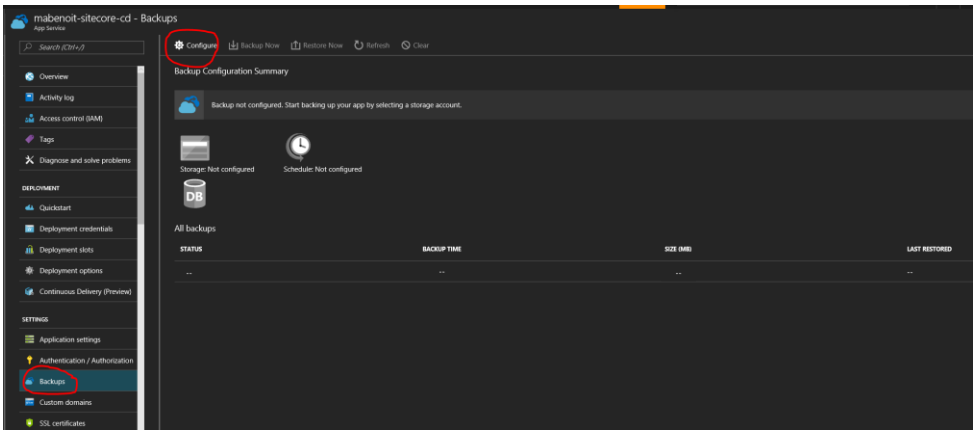
3. Then, browse the CM website by playing/changing the IP address with yours and another to check that works properly.

> ***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.*

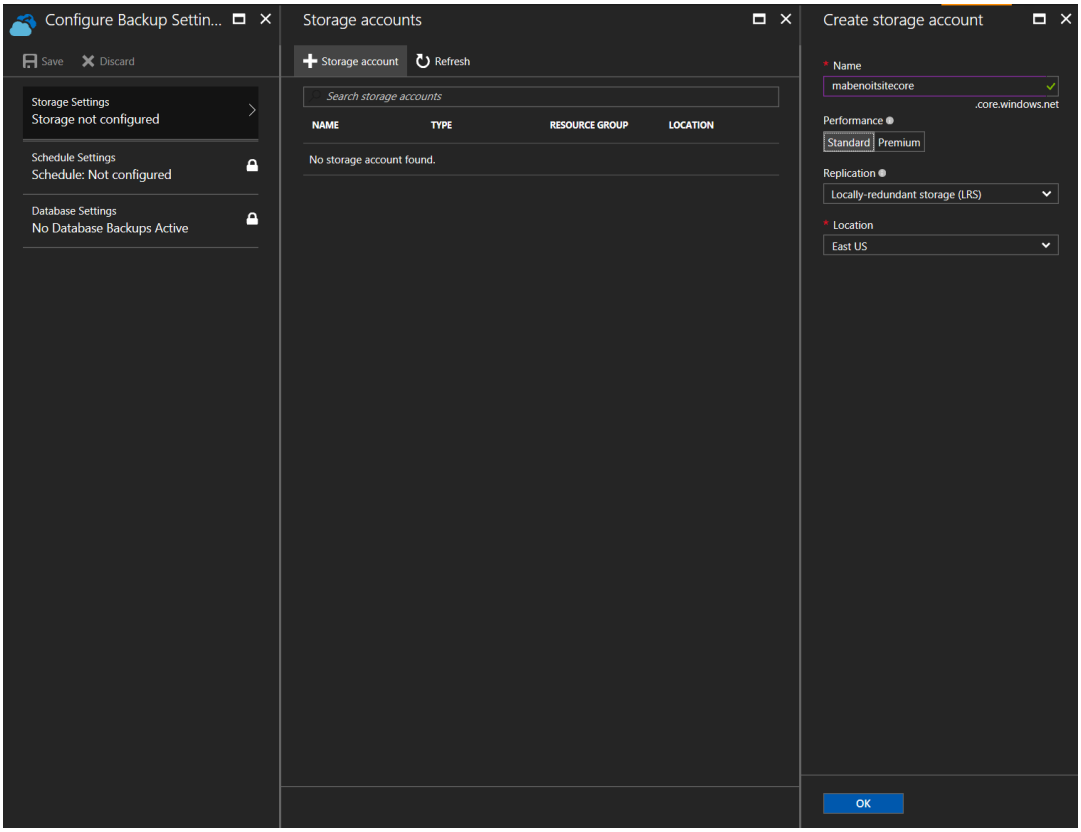
Task 3: Setup Azure Web App Backup (10 min)

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.*

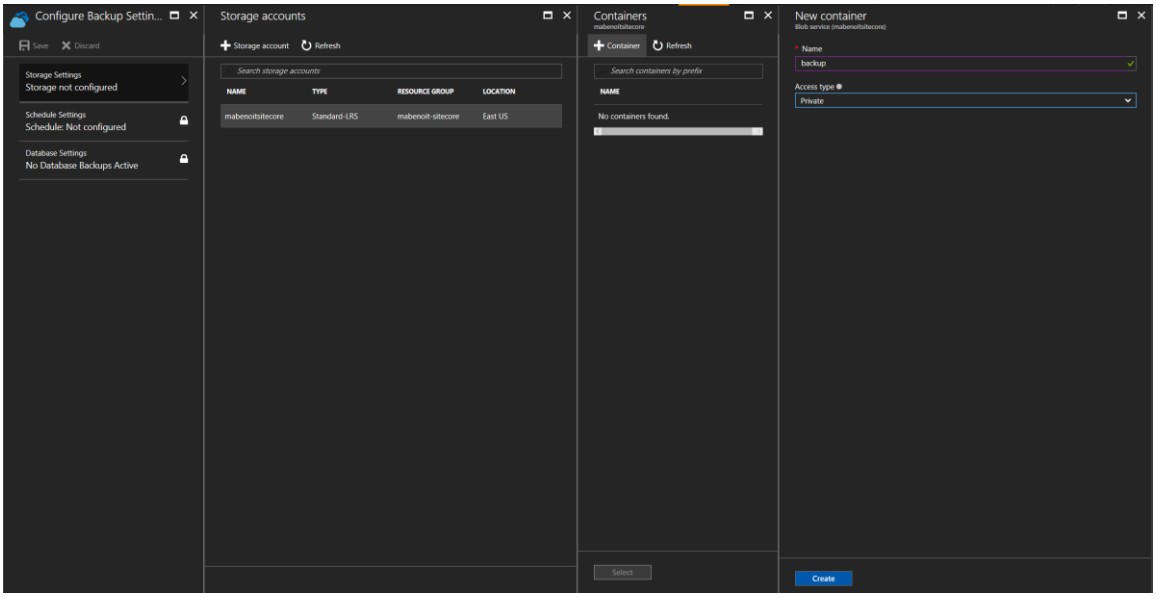
1. On the Azure portal, go back to the "…-cd" Web App Service resource and there go on the "**Backups**" blade. And then click on the "**Configure**" toolbar button.



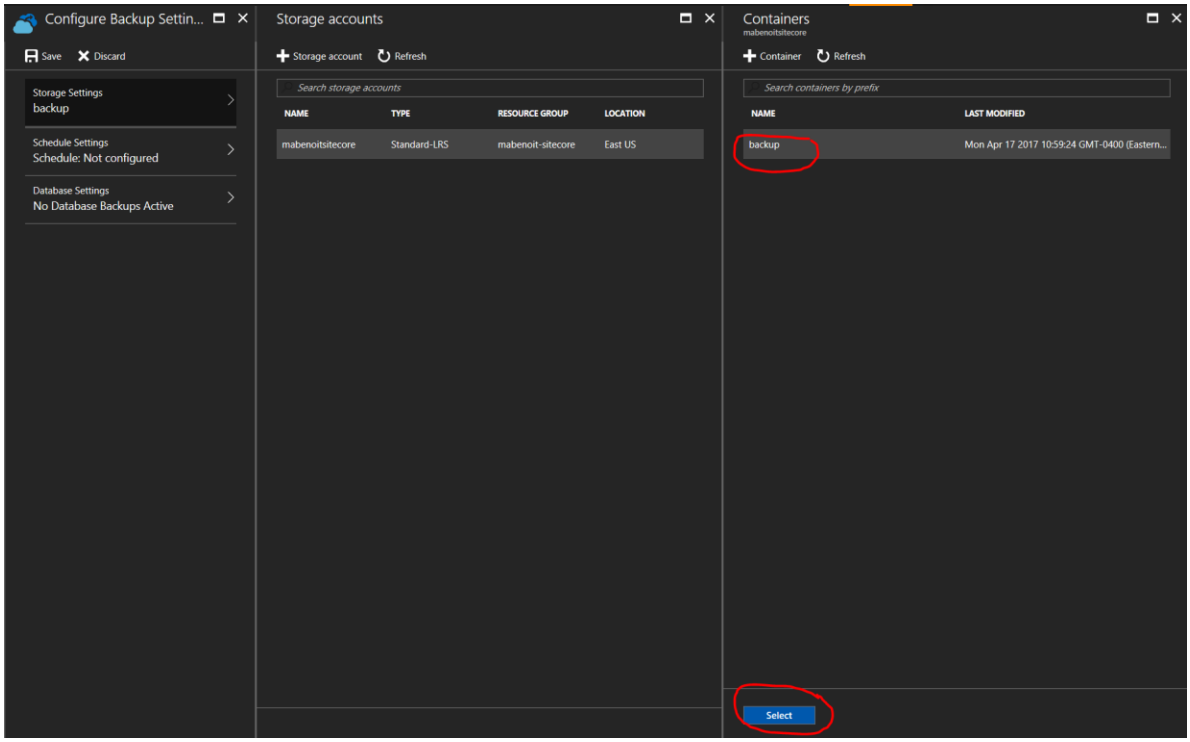
2. From there, you are invited to create a new storage account to store your backup files. You must provide a unique name (ensure the green checkmark appears) for the "**Name**" field and leave the other fields "as-is". Then click on the "**OK**" button to submit the deployment.



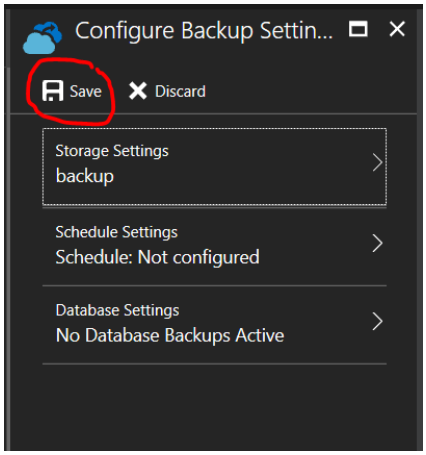
3. Once the Storage account is provisioned it will appear in the list, select it, click on the "**+ Container**" toolbar button and set the "**Name**" field value as "**backup**". Click on the "**Create**" button.



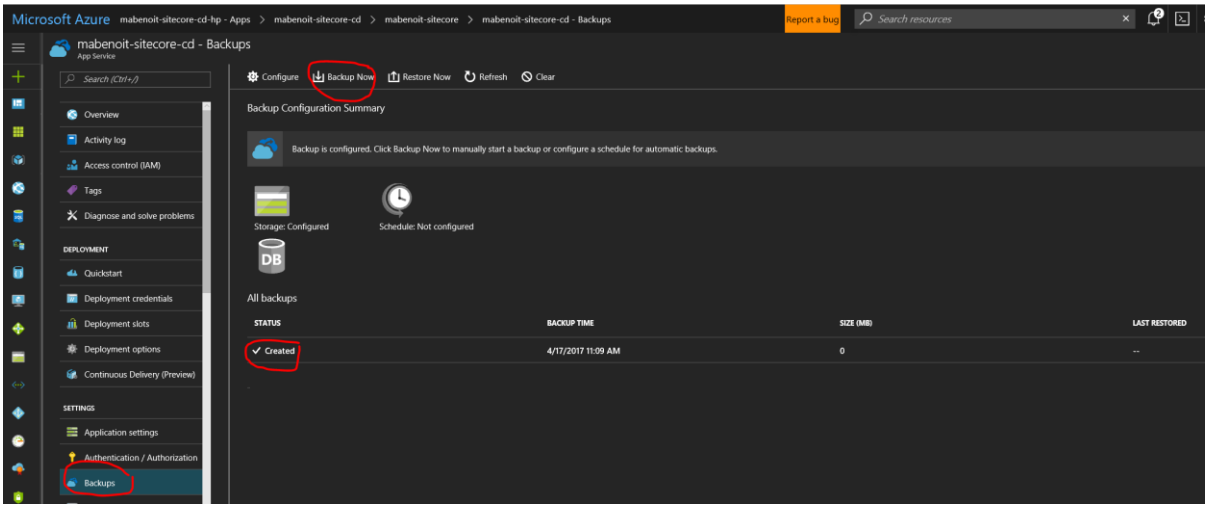
4. Then, select this “**backup**” container and click on the “**Select**” button.



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



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.



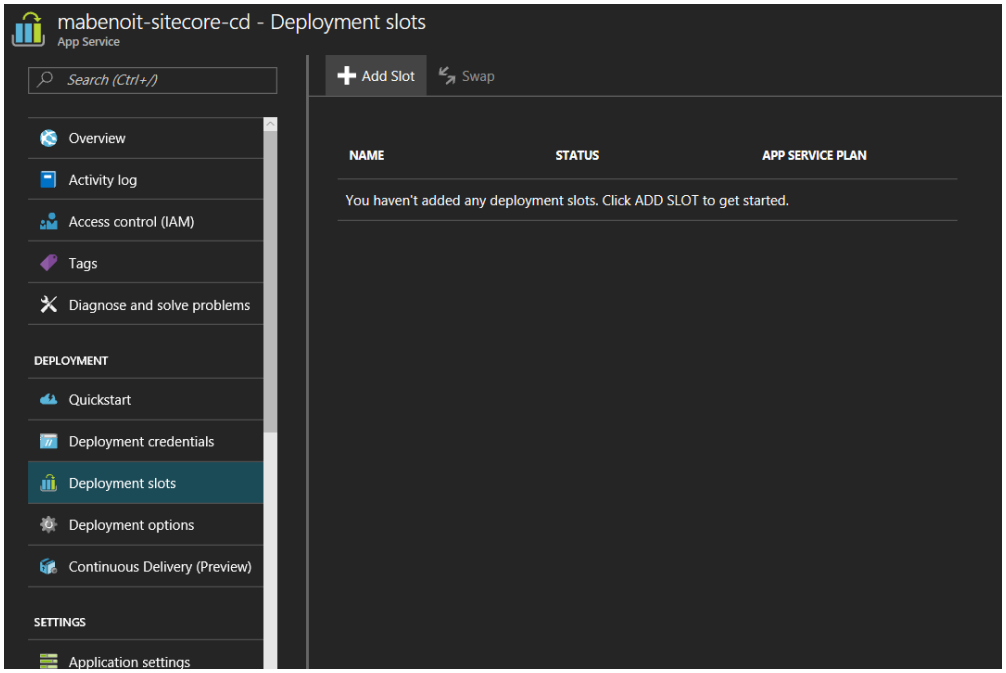
8. A new backup entry will appear with the "**Created**" status. We are not waiting for the end of the backup, let's move forward with the next task below, we will use this backup file generated later on this lab.

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.

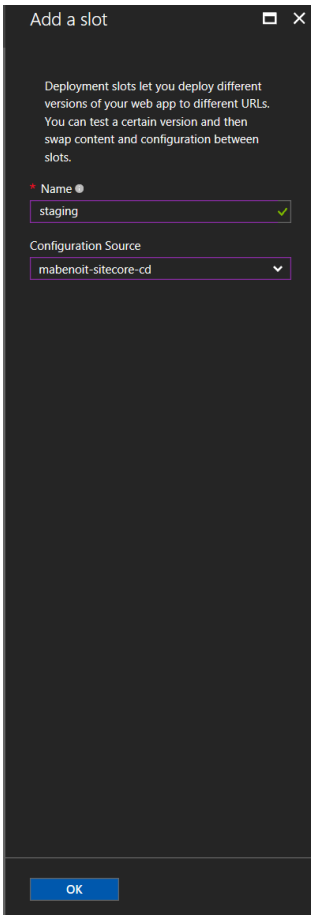
Task 4: Create a Staging Slot for the CD instance (15 min)

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.

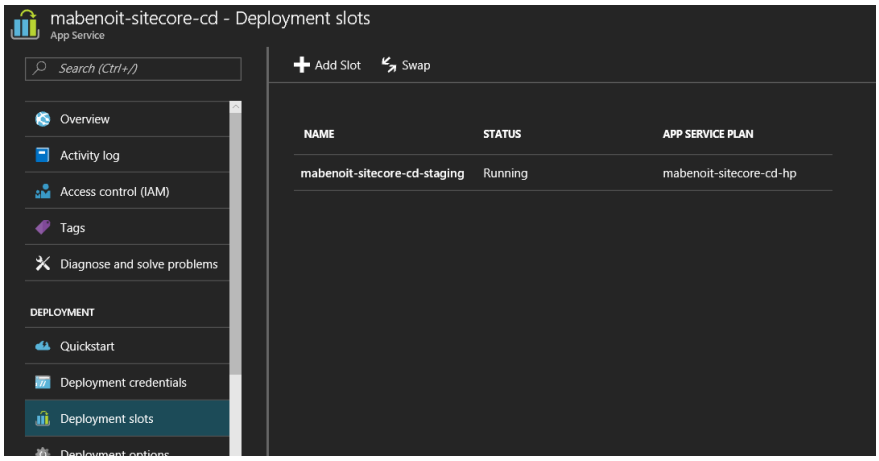
1. On the Azure portal, go back to the "…-cd" Web App Service resource and there go on the "**Deployment slots**" blade.



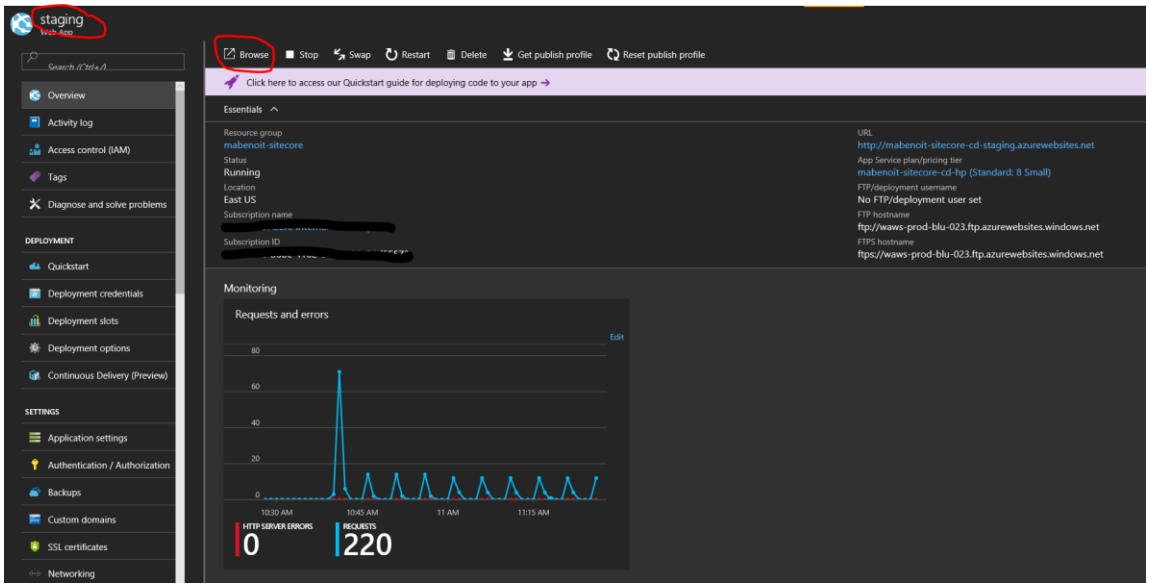
2. Click on the "**Add Slot**" toolbar button. Name the slot "**staging**" and on the "**Configuration Source**" field, select the "…-cd" web app. Click the "**OK**" button then.



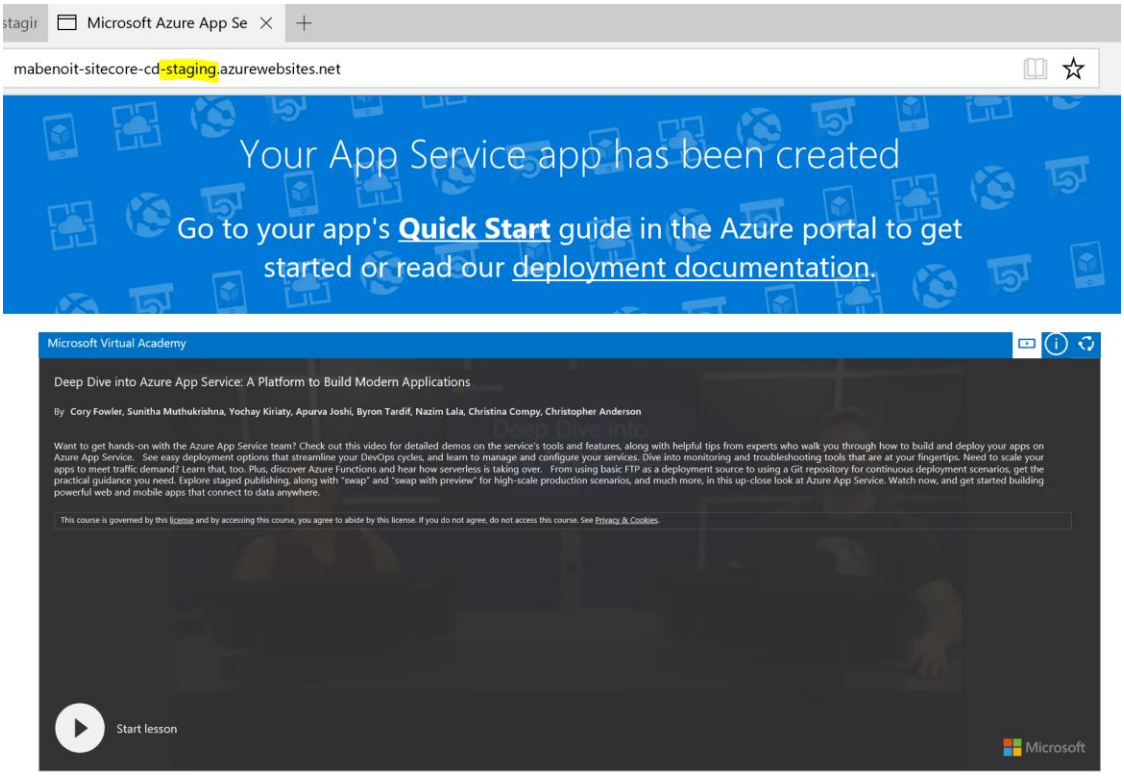
3. Few seconds after, the “**staging**” slot should appear on the “**Deployment slots**” list.



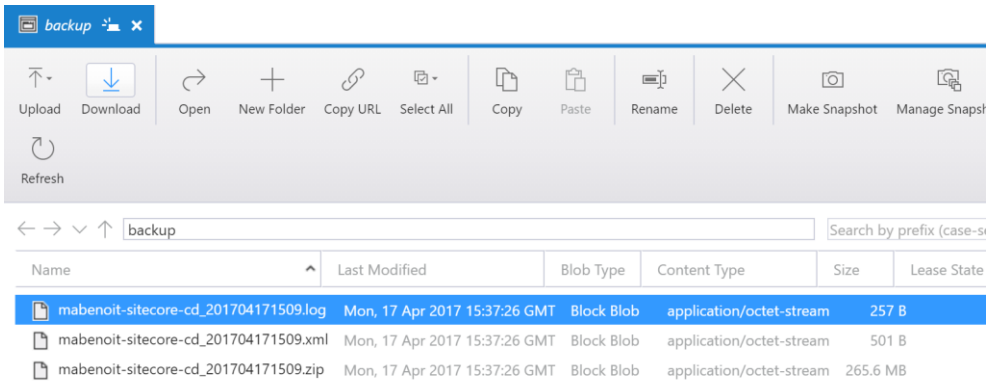
4. Click on it to open its associated resource blade:



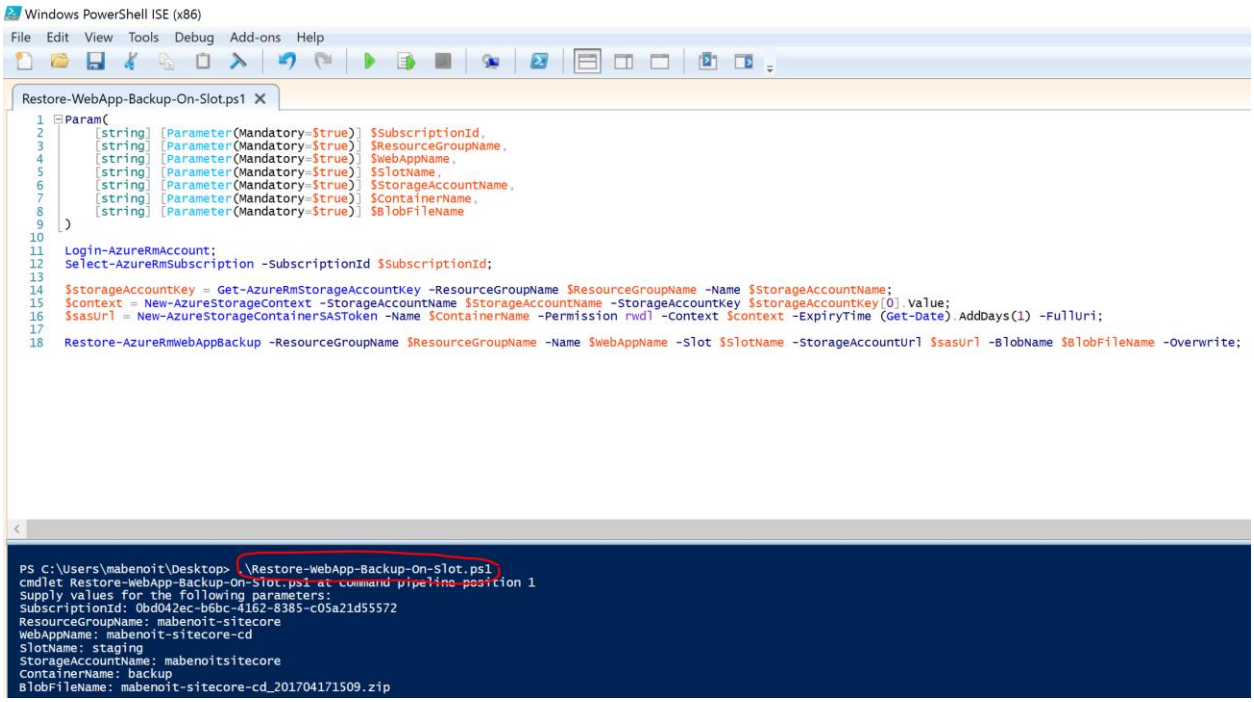
5. Click on the “**Browse**” toolbar button to open the empty website (we have not yet deployed anything on it).



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.



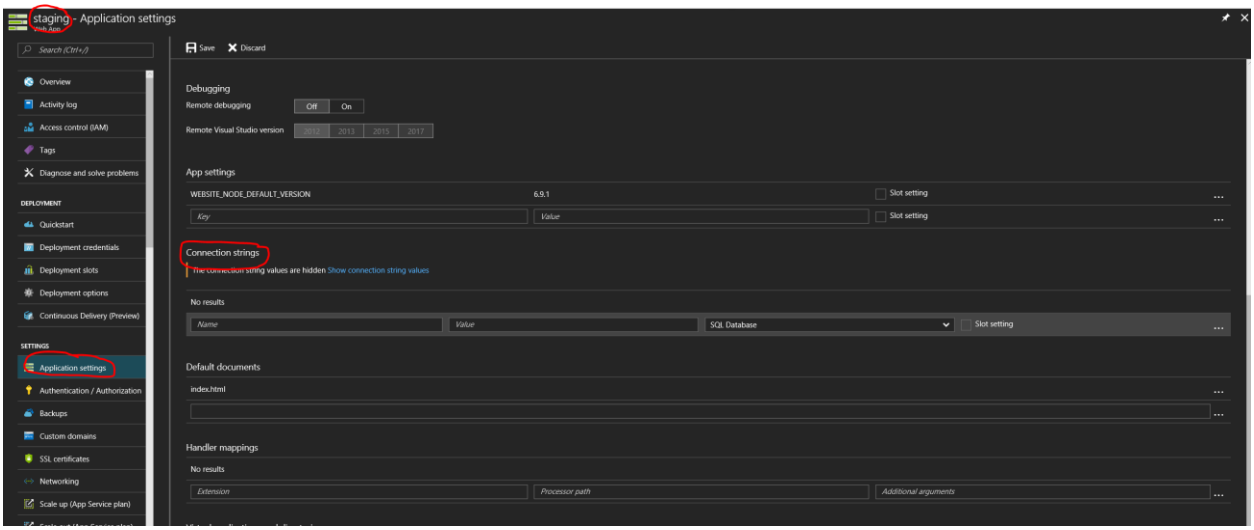
7. Go to https://gist.githubusercontent.com/mathieu-benoit/f3d6b1b2d330c4fe11dbf37b475745ec/raw/0fd2fa961b2fe970918cc2142050788048f230db/Restore-WebApp-Backup-On-Slot.ps1
8. Copy/paste the code as “Restore-WebApp-Backup-On-Slot.ps1” file into “C:\SitecoreLab\PowerShell”.
9. Right click on the “Restore-WebApp-Backup-On-Slot.ps1” file and select “Edit”.
10. The PowerShell ISE tool is now opened.



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

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

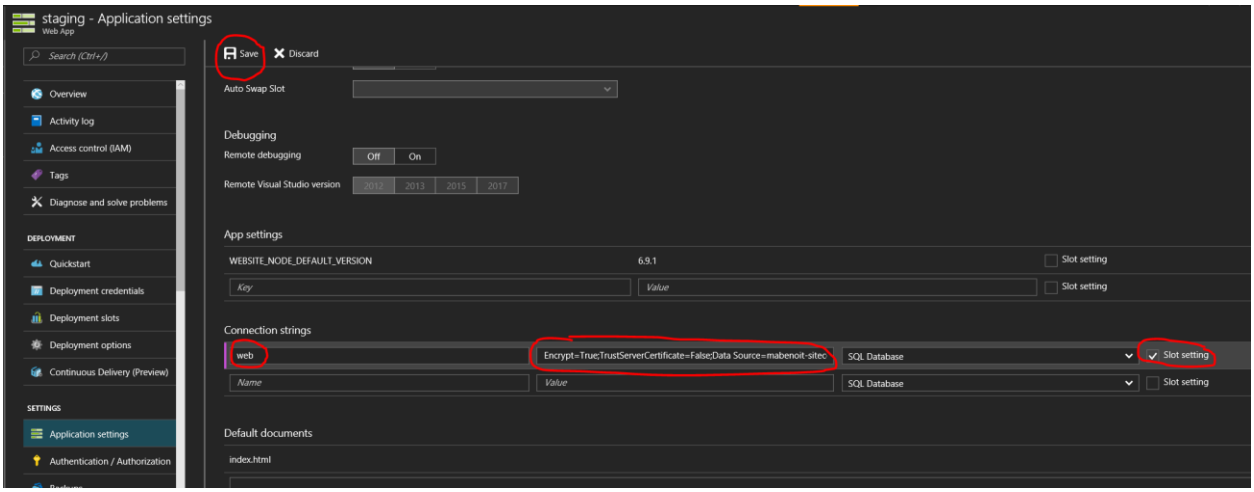
12. You will be prompted to enter the associated values of the following parameters:
    a. **SubscriptionId**: *associated subscription id.*
    b. **ResourceGroupName**: *associated resource group name.*
    c. **WebAppName**: *associated "…-cd" Azure WebApp name.*
    d. **SlotName**: staging
    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/ConnectionStrings.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.



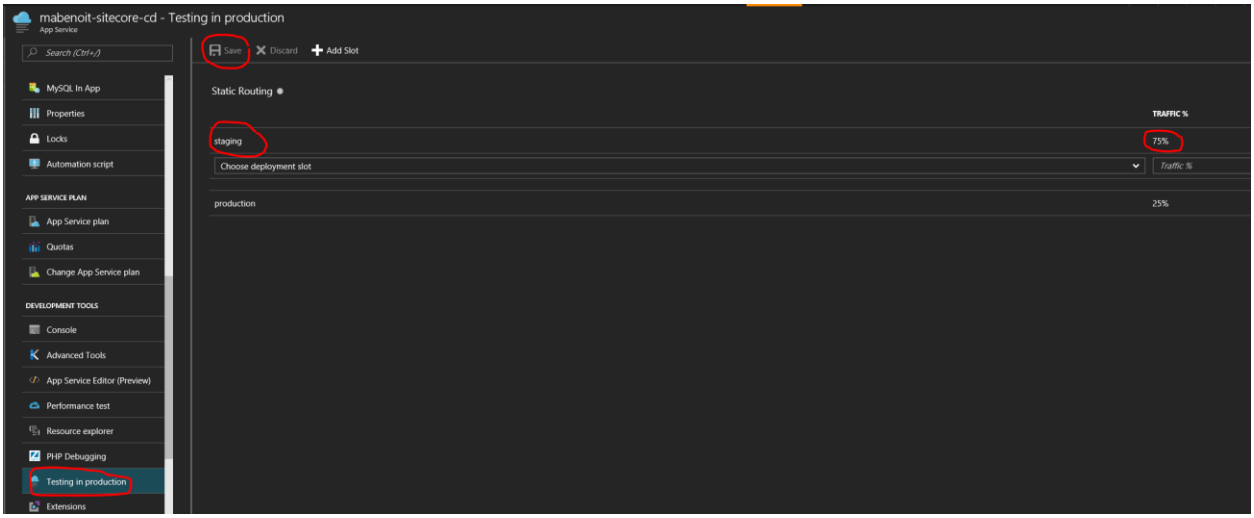
Note: You could perform another PowerShell command which could clone a WebApp into a Slot instead of restoring a WebApp backup in a Slot like we just did. The documentation could be found here. *But there is currently an issue like described here which will be fixed soon. Stay tuned!*

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.

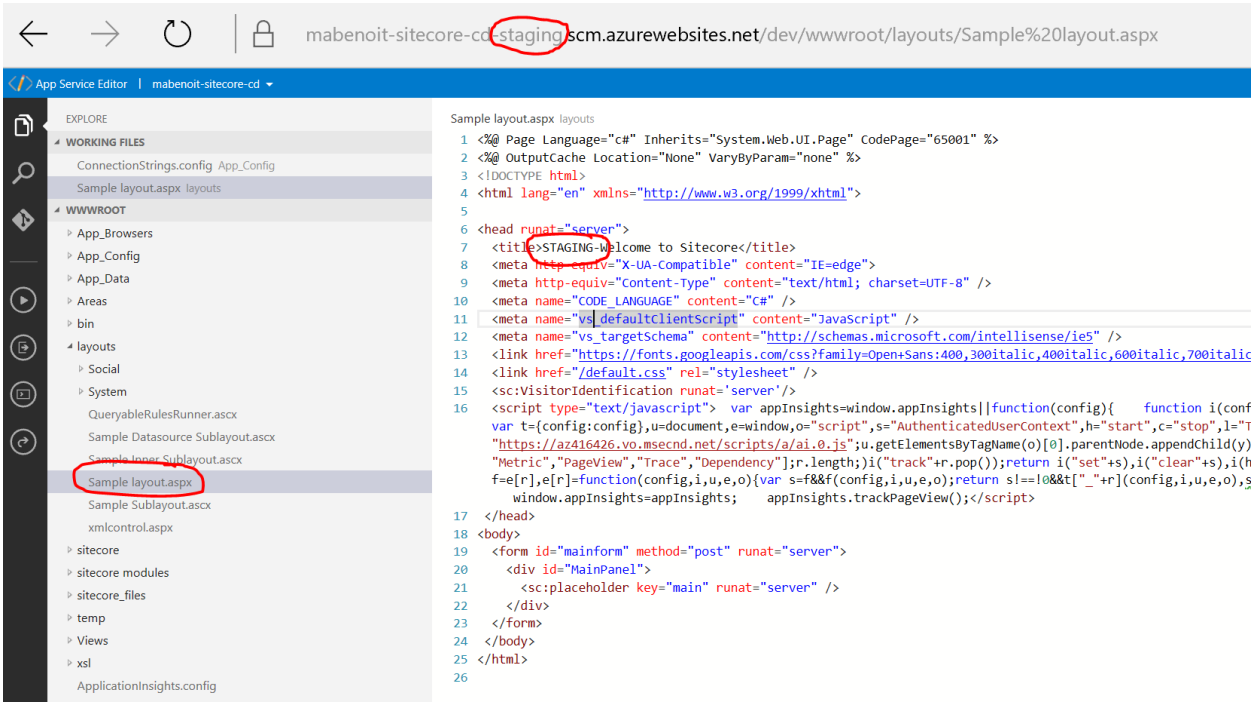
1. Go to the "…-cd" Azure Web App resource blade and click on the "**Testing in Production**" blade.

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

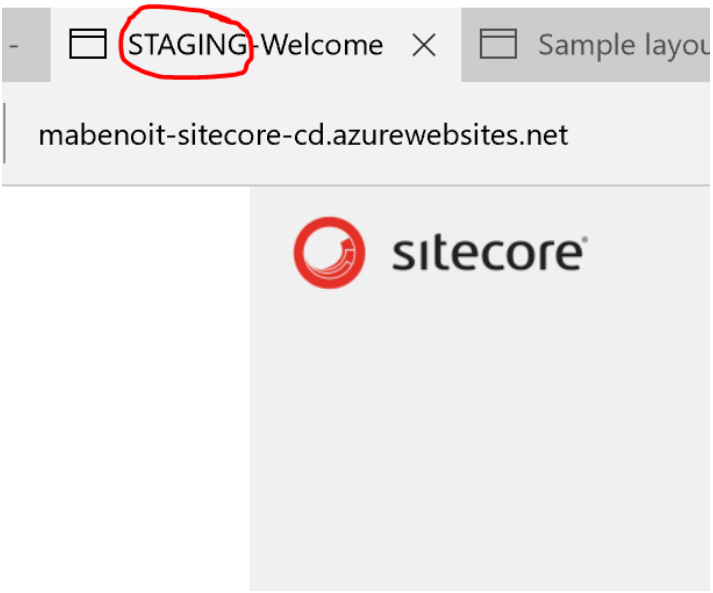


3. If you open a new web browser and navigate to the CD website URL, you will be redirected at 75% to the "**staging**" website (the URL is still the same).
4. You could now invert the value to have more traffic on the "**production**" site than the "**staging**" site. *You will need another web browser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

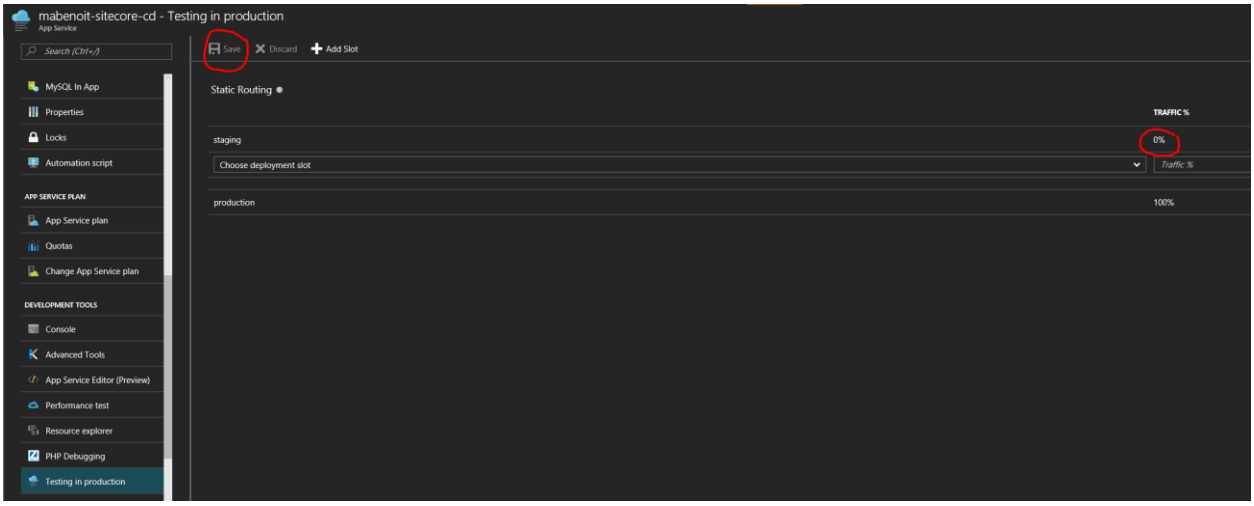
5. To make sure you are on the right site during these tests, you could edit the "**staging**" site by going to the "**App Service Editor (Preview)**" feature and open the "**wwwroot/layouts/Sample layouts.aspx**" file and add the "**STAGING-**" prefix to the <title> tag for example like illustrated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



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



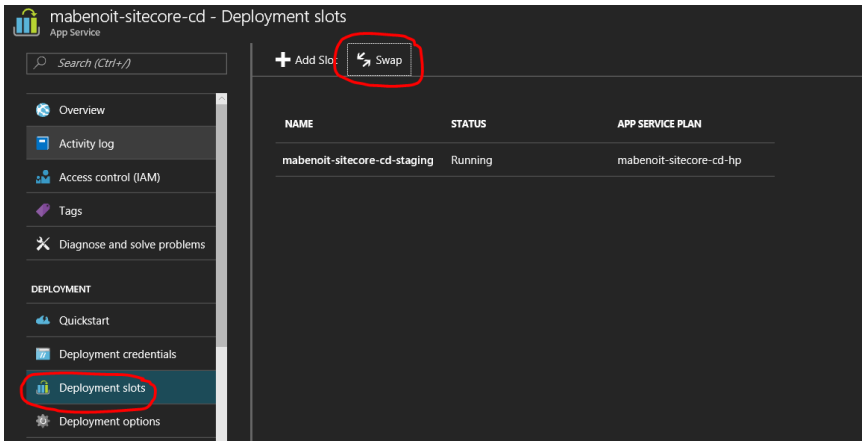
7. After these tests, you could remove the routing rules by setting the "**Traffic %**" of the "**staging**" slot to "0".



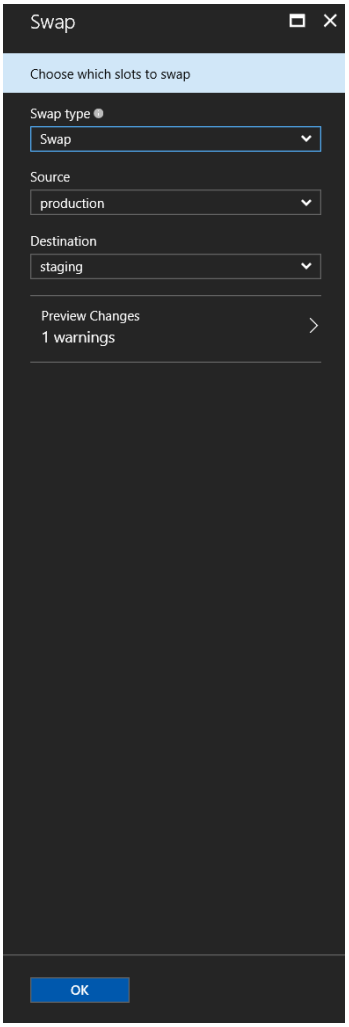
Task 6: Slot Swap (5 min)

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.



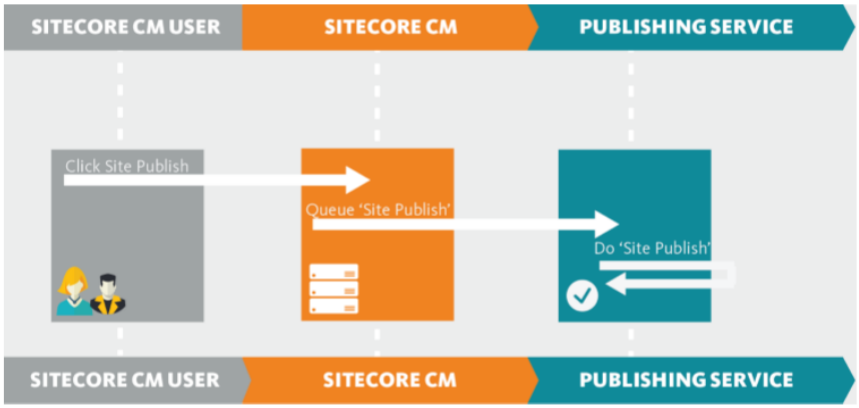
2. On the "**Swap**" blade, you could select your "**Source**" and your "**Destination**"; to actually perform the swap action you could click on the "OK" button. *For the purposes of this lab we are not doing and illustrating that, but be aware of this great feature for your own projects.*



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.

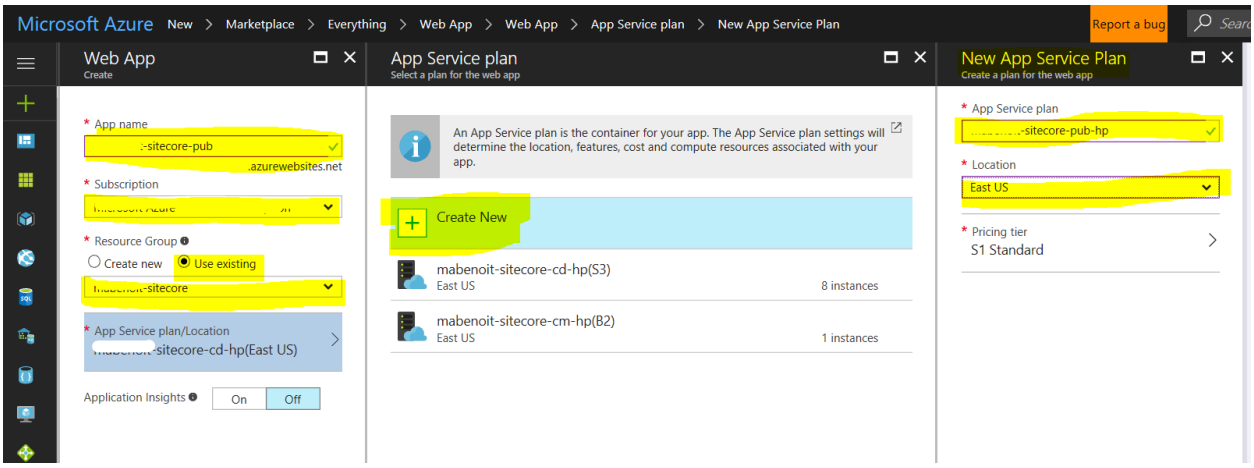
The Publishing Service does not use any of the features, pipelines, and settings in the current publishing system. It is an entirely new way of publishing Sitecore items and media.

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

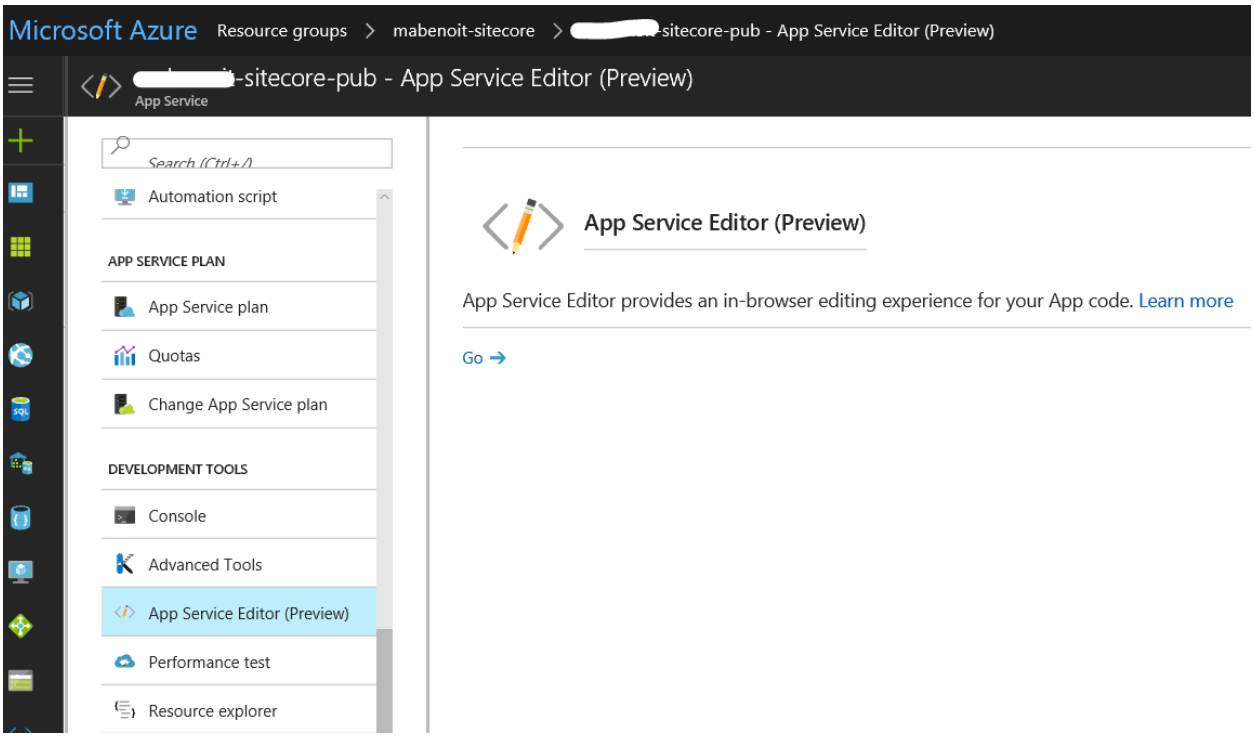
Installation involves:

- 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.

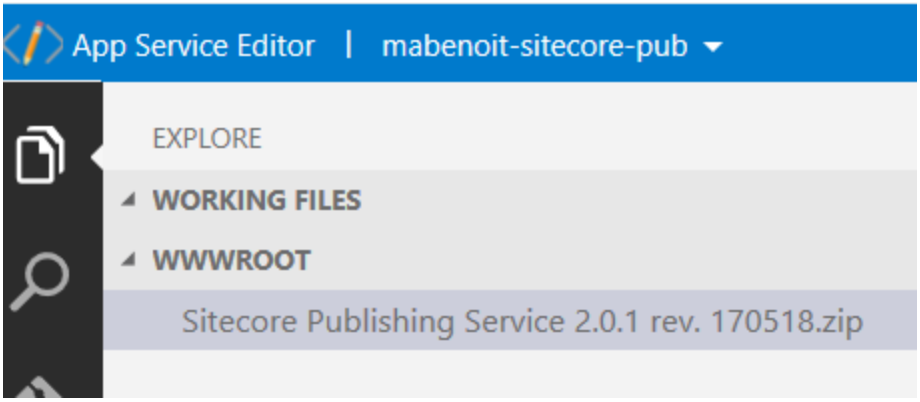
1. Download the Publishing Service and Module from this link: https://dev.sitecore.net/Downloads/Sitecore_Publishing_Service/20/Sitecore_Publishing_Service_20_Update1.aspx
   ➔ *If you don't have a login to download them, please ask your Sitecore partner or rep or your lab proctor*.
2. Go to https://portal.azure.com/#create/Microsoft.WebSite
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:



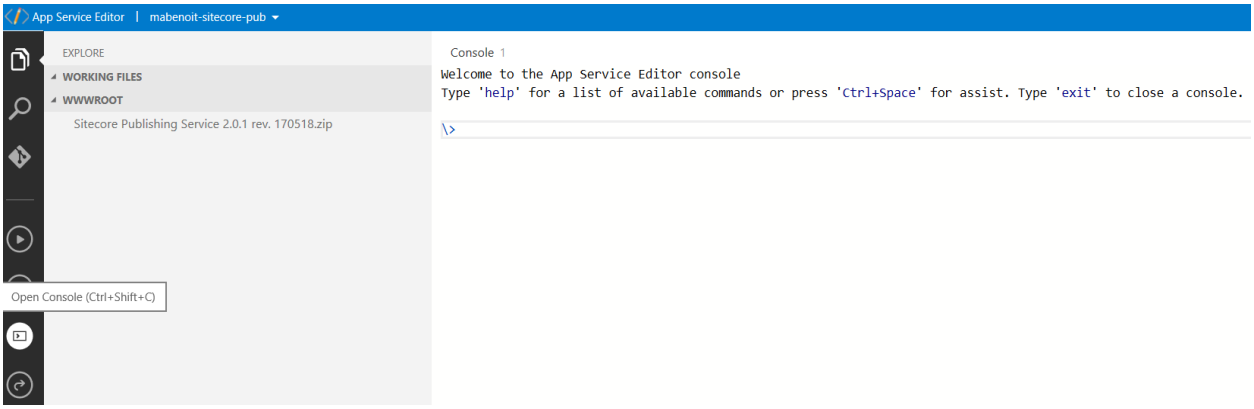
4. Click on the "**Create**" button. The new App Service Plan with the new Web App will be provisioned.
   - ➔ *Remark: having a dedicated App Service Plan for this new server, will allow you to use Scale out and Scale up features without impacting other servers and take advantage of these features.*
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.

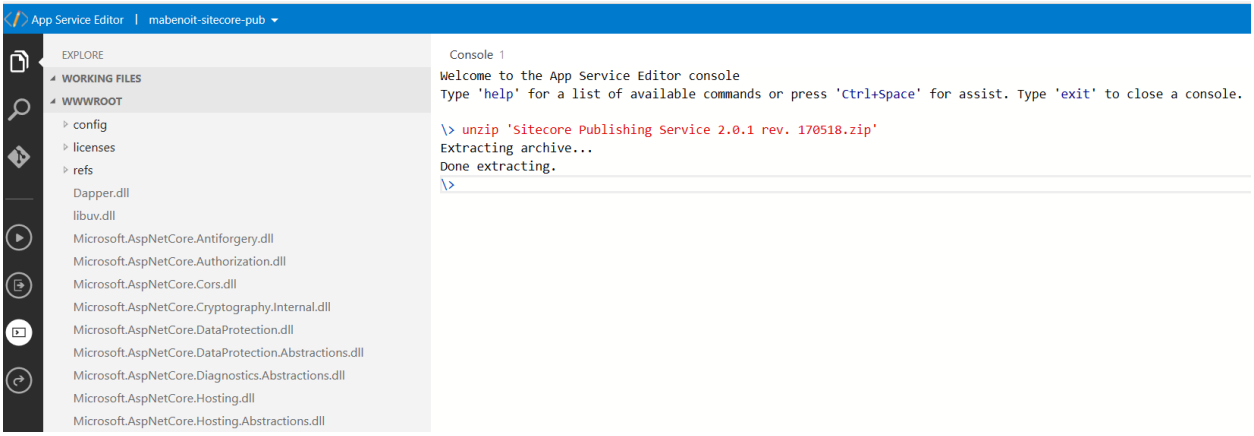


8. Open the associated **Console** like illustrated below:



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

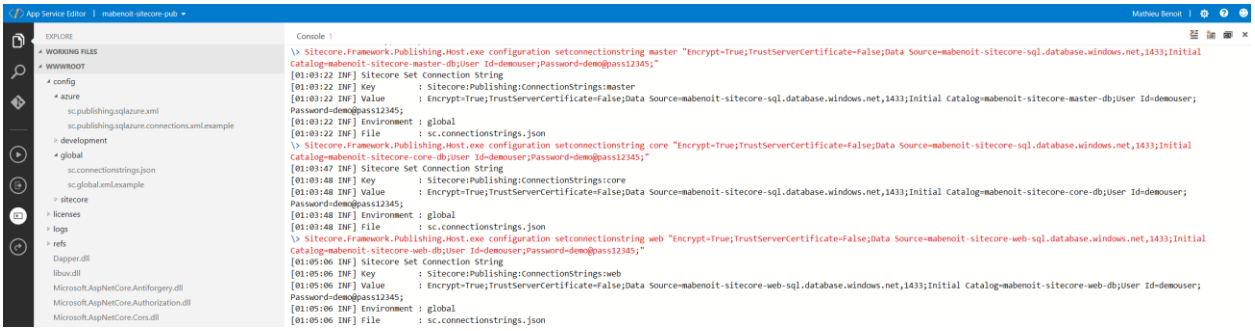
```
unzip 'Sitecore Publishing Service 2.0.1 rev. 170518.zip'
```



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:

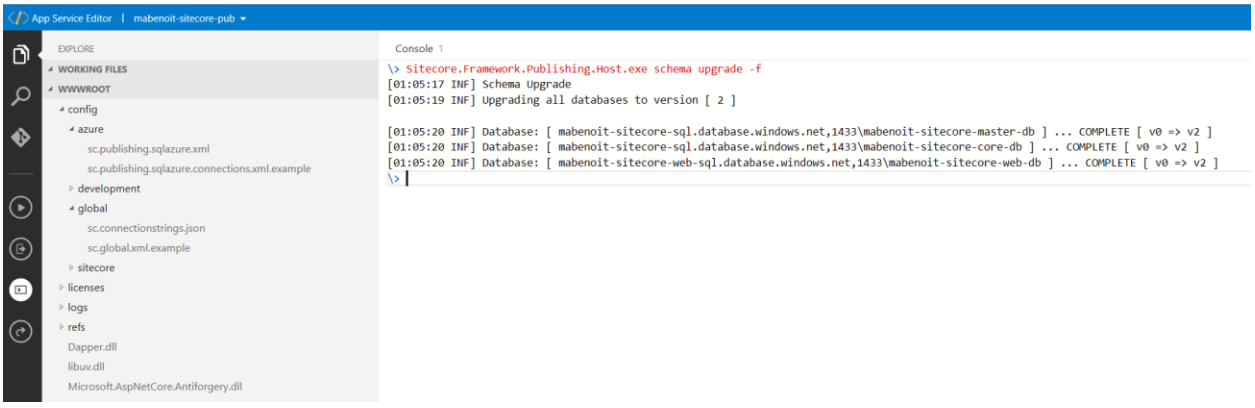
```
Sitecore.Framework.Publishing.Host.exe configuration
setconnectionstring core
"Encrypt=True;TrustServerCertificate=False;Data
```

```
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"
```



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

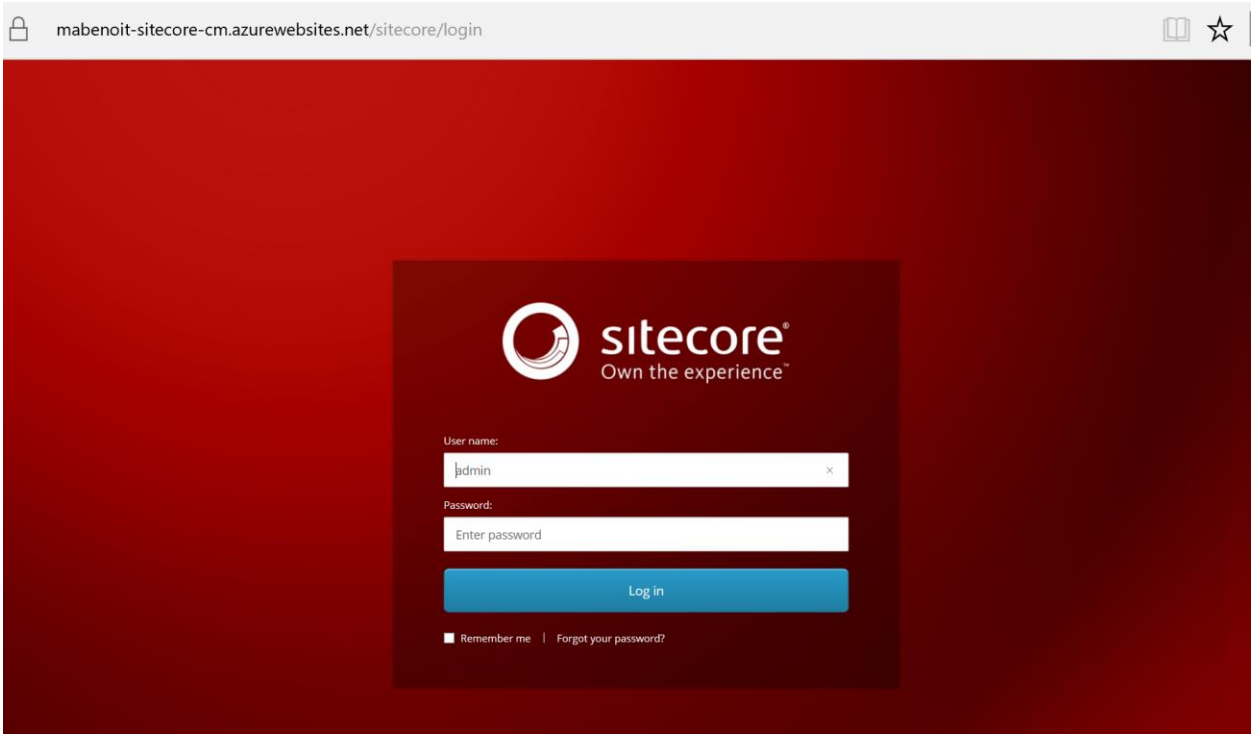
```
Sitecore.Framework.Publishing.Host.exe schema upgrade -f
```



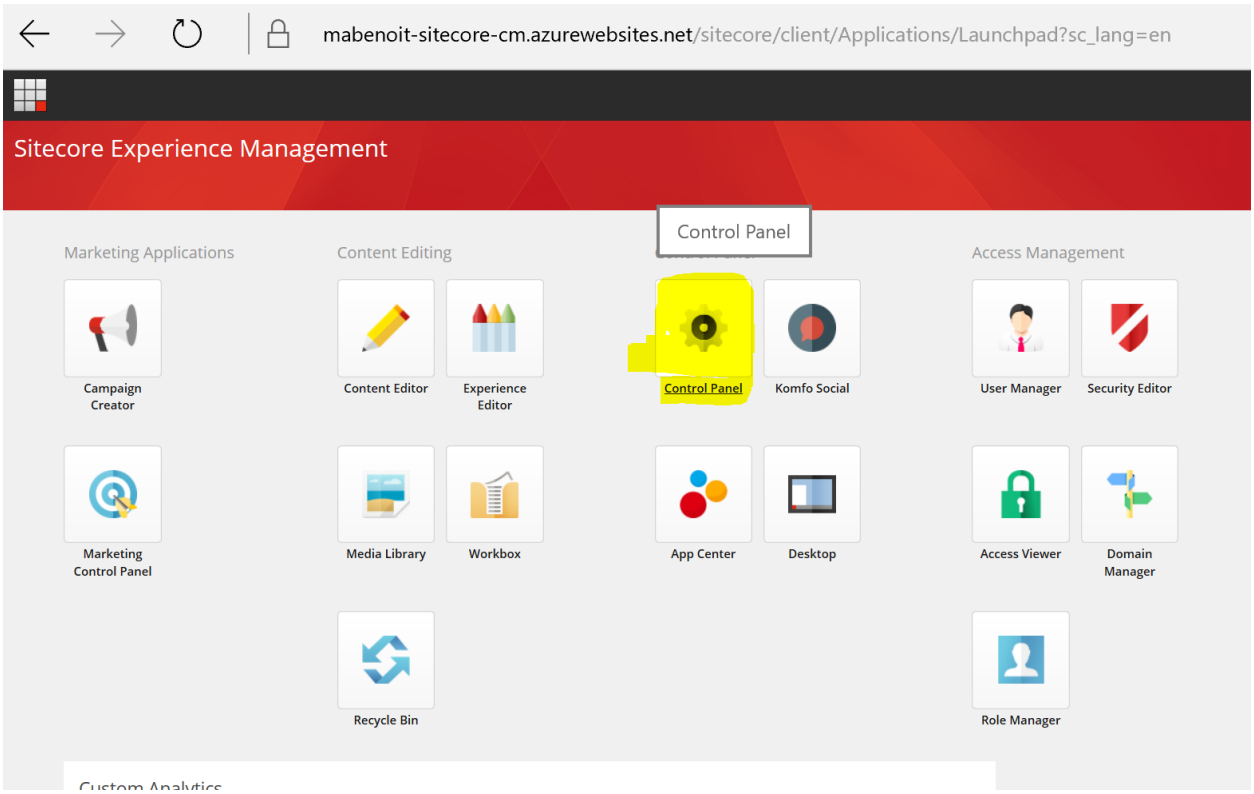
12. Now the service is ready. To test your service, try to hit this url: http://<service host or domain anme>/api/publishing/maintenance/status. The service should return status:0 in json response.

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

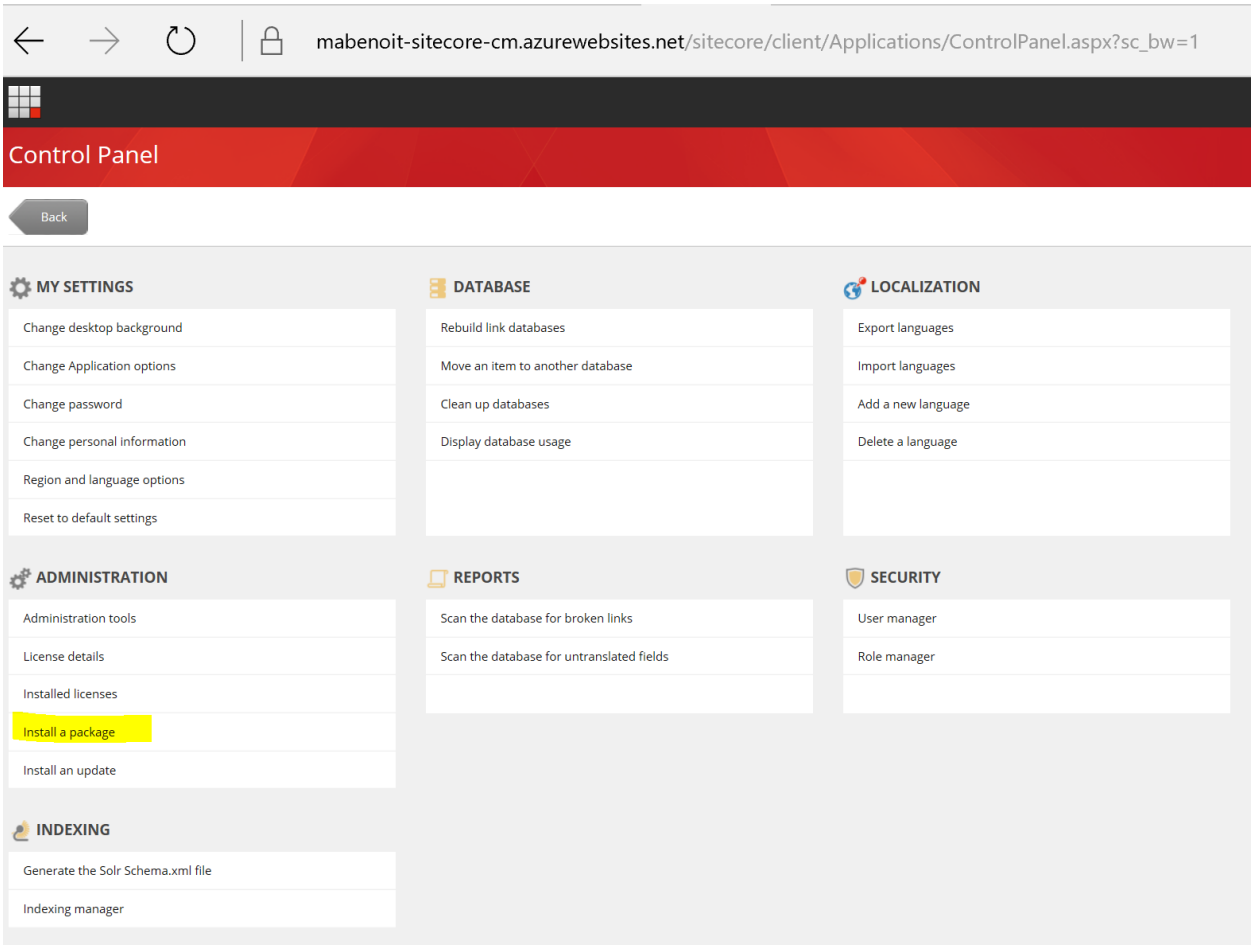
13. Go to your Sitecore CM URL by adding at the end of this URL **"/sitecore**".



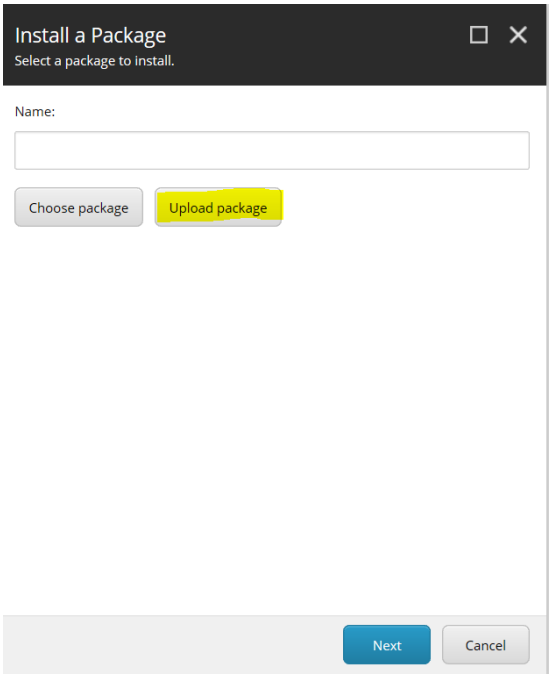
14. Login with these credentials:
    - ➔ Login: admin
    - ➔ Password: demo@pass12345
15. Once logged in, click on Control Panel.



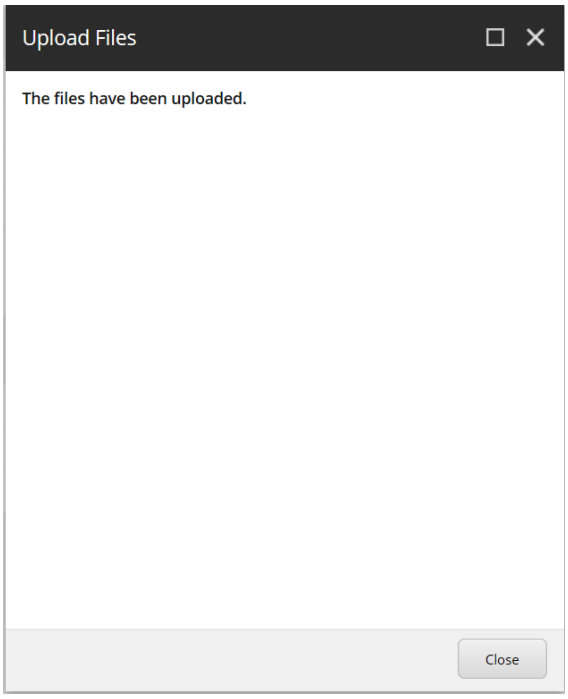
16. Click on “**Install package**”



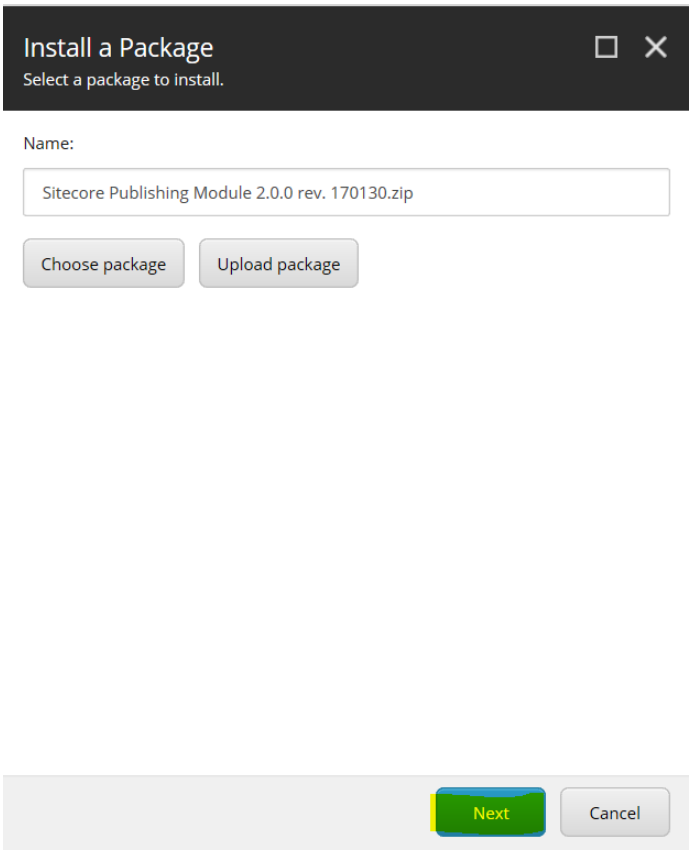
17. On the "**Install a package**" dialog, choose "**Upload package**".



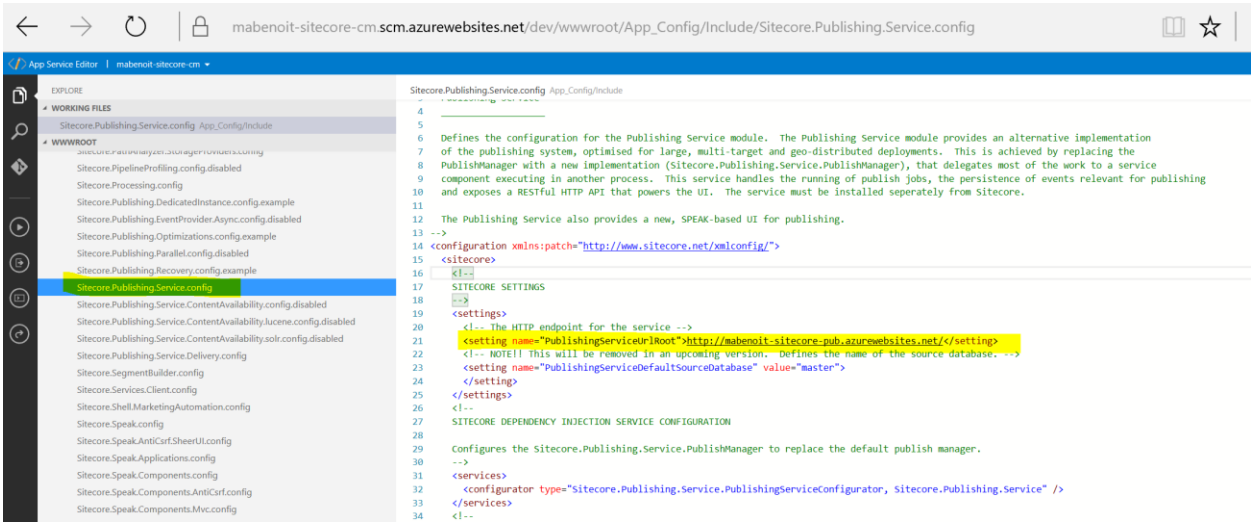
18. Select your local "**Sitecore Publishing Module 2.0.1 rev. 170518.zip**", few seconds after the confirmation dialog should appear:



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 you CM instance - https://{your url}-cm.scm.azurewebsites.net/dev/. And open the wwwroot/App_Config/Include/Sitecore.Publishing.Service.config file.



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.

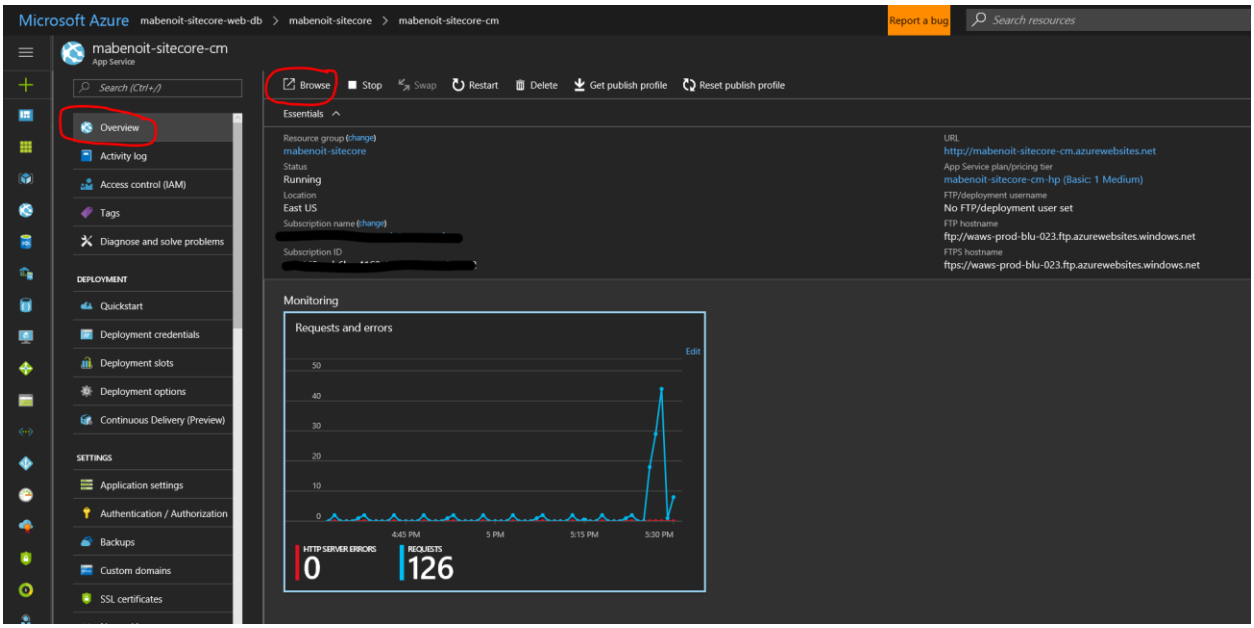
Through this exercise, you will play/use with:

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

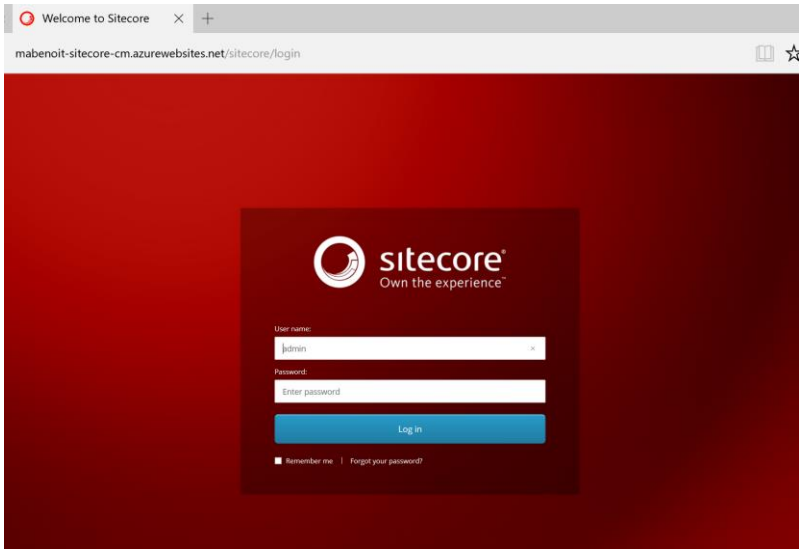
Task 1: Azure Search (15 min)

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.

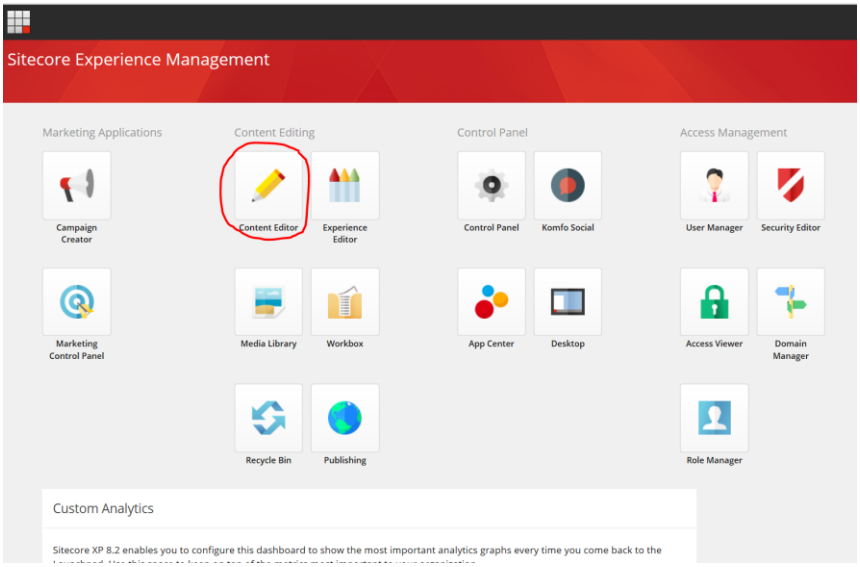
1. Go to https://portal.azure.com/ and select your "…-cm" Azure Web App.



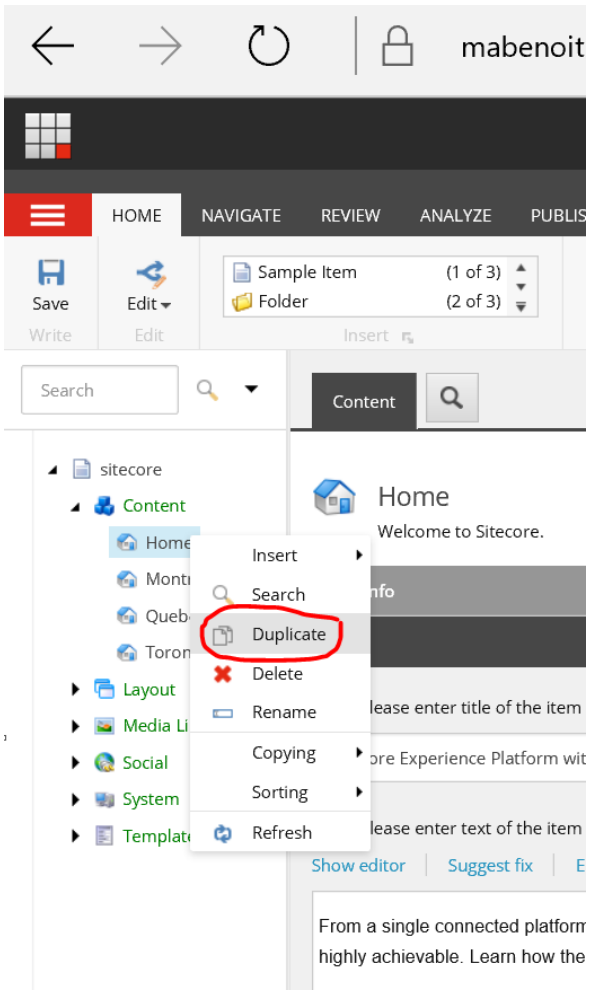
2. On the "**Overview**" blade, click on the "**Browse**" toolbar action.
3. There, login with the following credentials to the "/sitecore" page.
    a. **User name**: admin
    b. **Password**: demo@pass12345

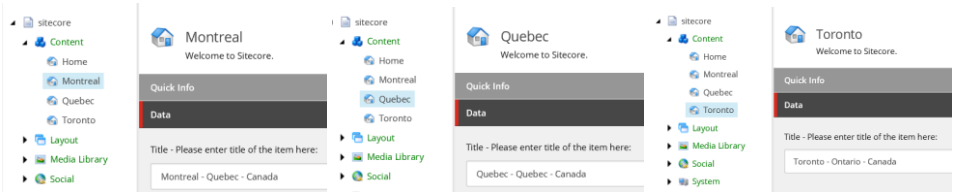


4. On the Sitecore Launchpad page, click on "**Content Editor**".



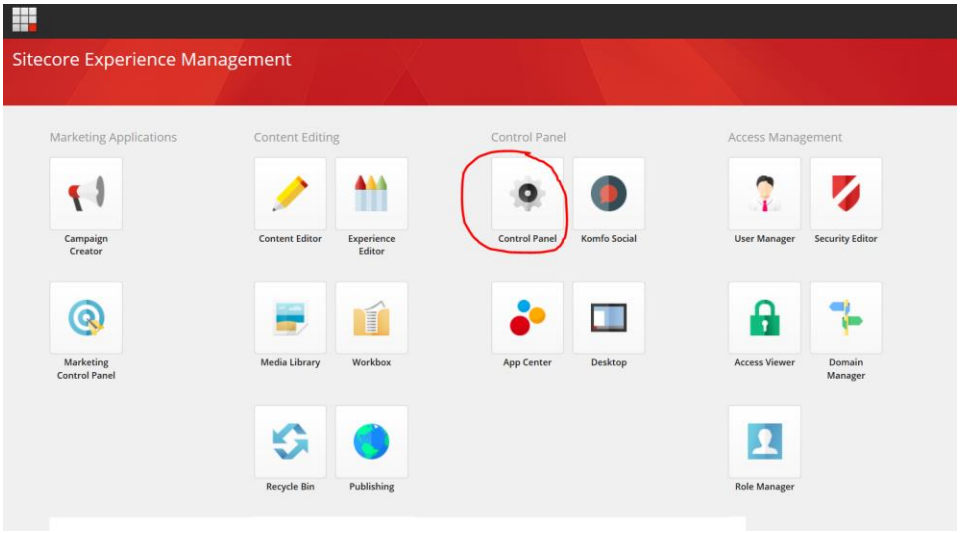
5. By default, the "**Home**" Sitecore item will be selected. Right-click on it and select "**Duplicate**" third times to add these 3 Sitecore items:
    a. Montreal - the field "**Title**" with "Montreal – Quebec – Canada" as value.
    b. Quebec – the field "**Title**" with "Quebec – Quebec – Canada"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



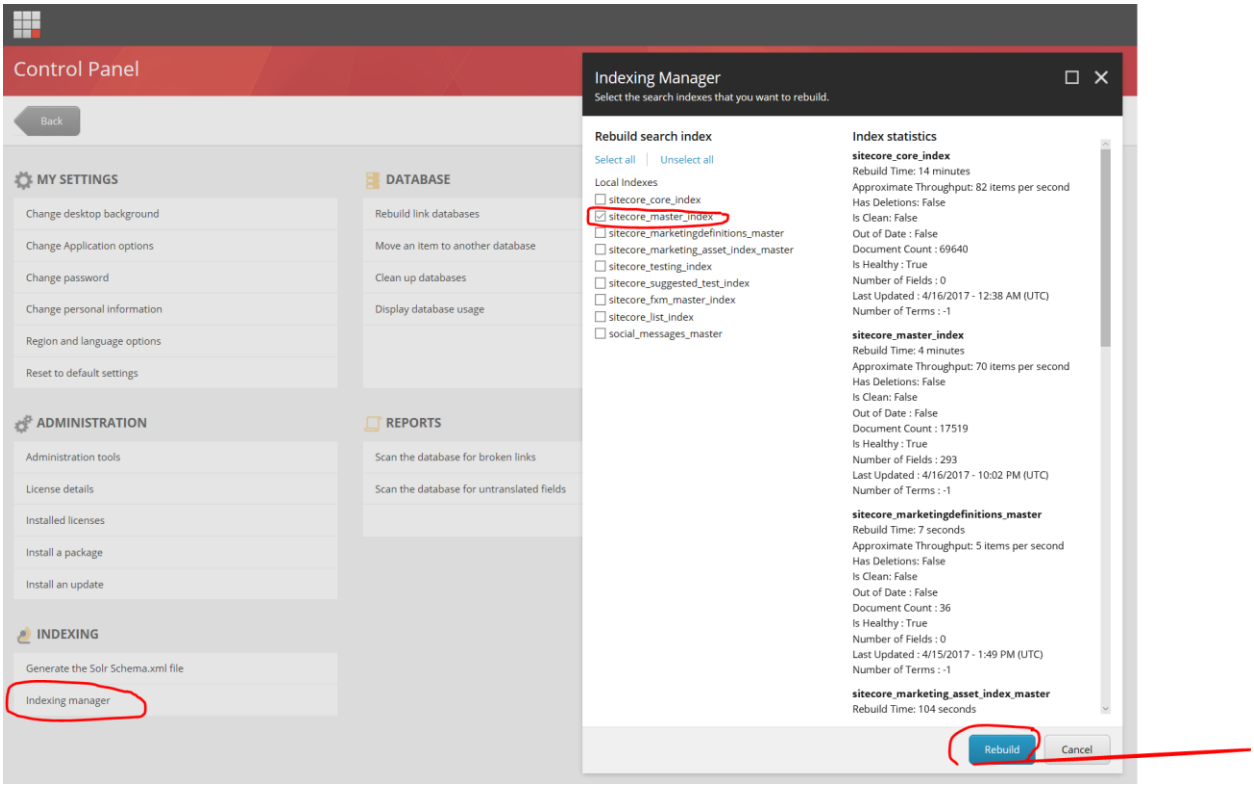


6. Save all these changes and then click on the top left icon to go to the Sitecore Launchpad page.

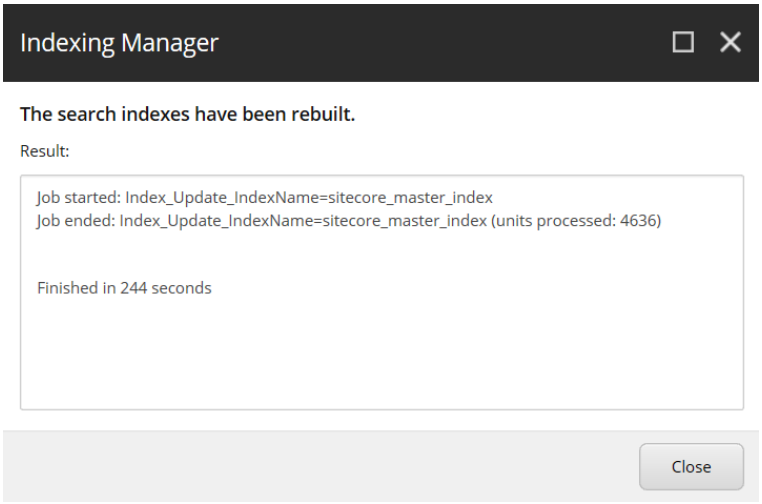
7. There, click on "**Control Panel**".



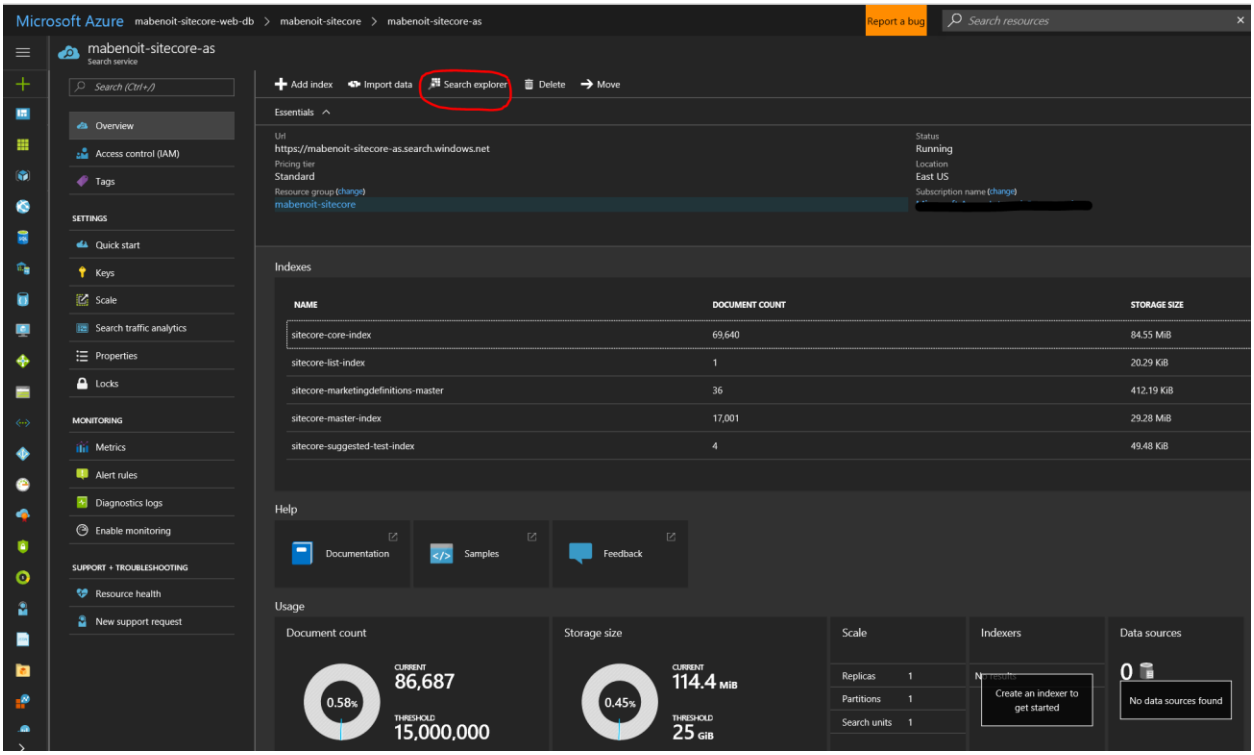
8. From the "**Control Panel**" page, click on "**Indexing Manager**". On the dialog box prompted select the "**sitecore_master_index**" and then click on the "**Rebuild**" button.



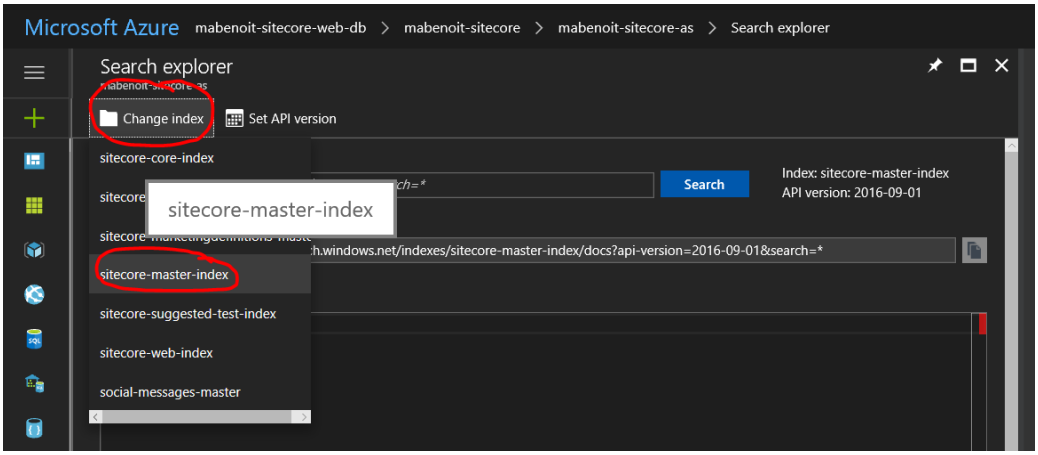
9. After ~4 min the associated index will be rebuilt on the Azure Search service. While the index is rebuilding you could move forward with the following steps in parallel.



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



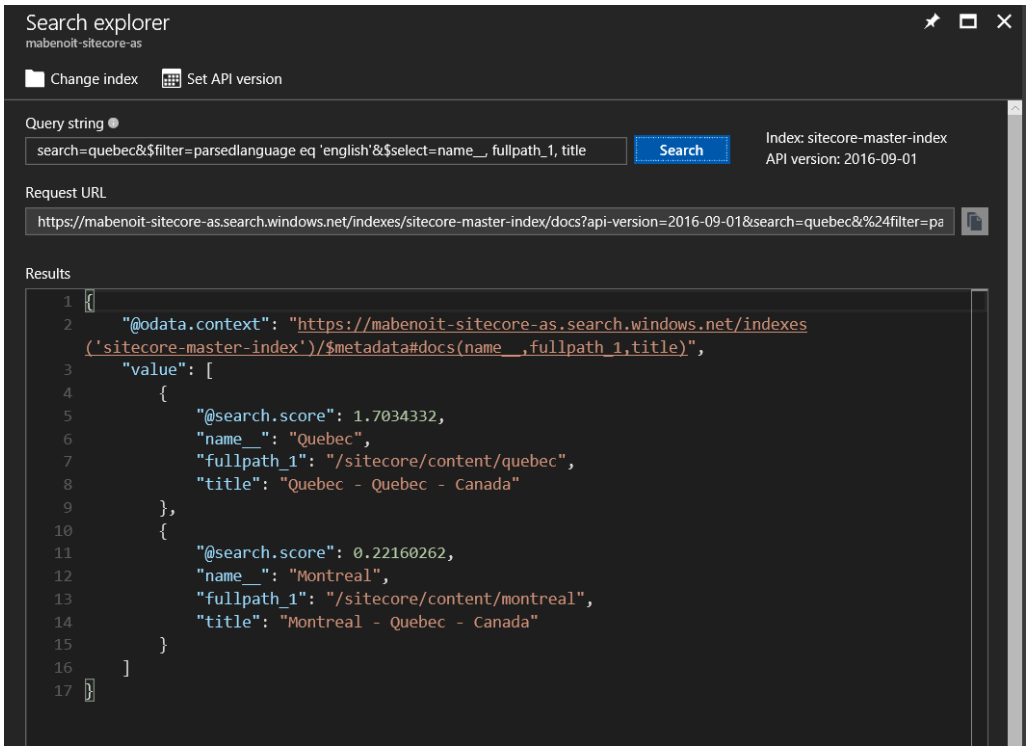
11. On the "**Overview**" blade, click on the "**Search explorer**" toolbar button.
12. On the "**Search explorer**", "**Change index**" to "**sitecore-master-index**".



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

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

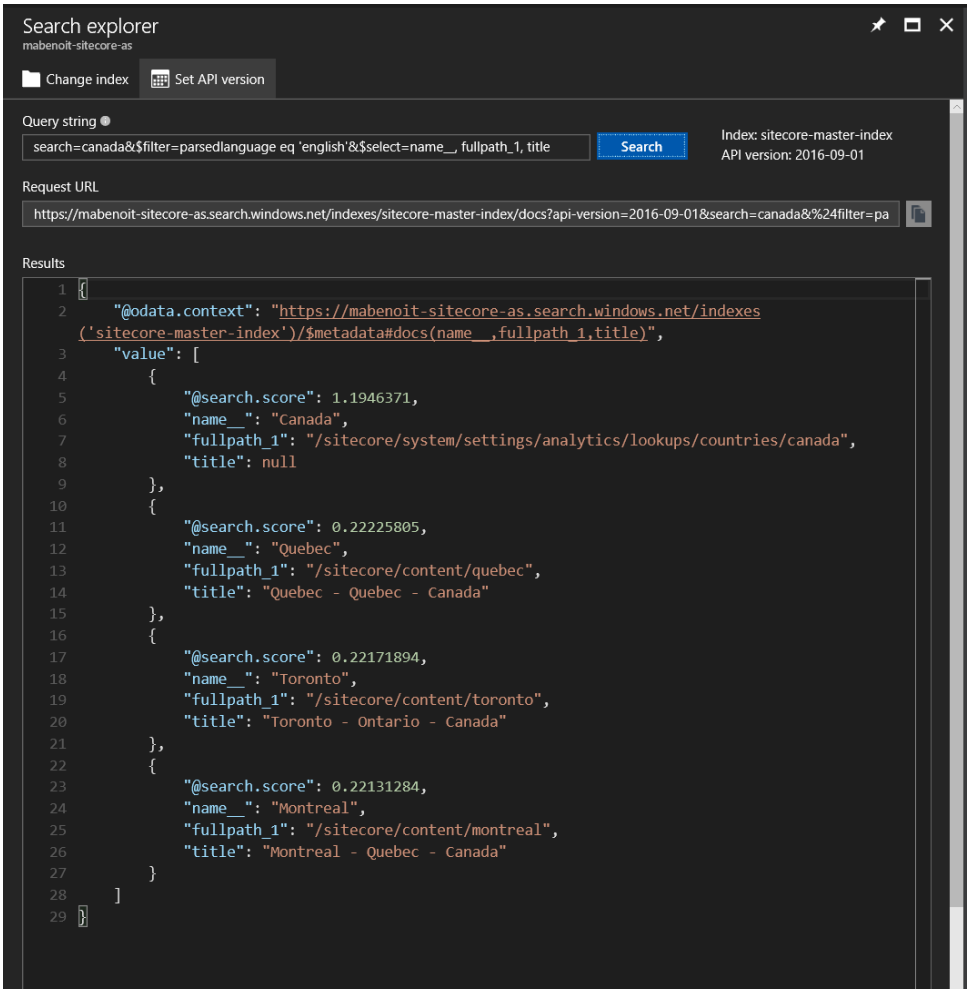
14. By clicking on the "**Search**" button you should get the following result:



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
```

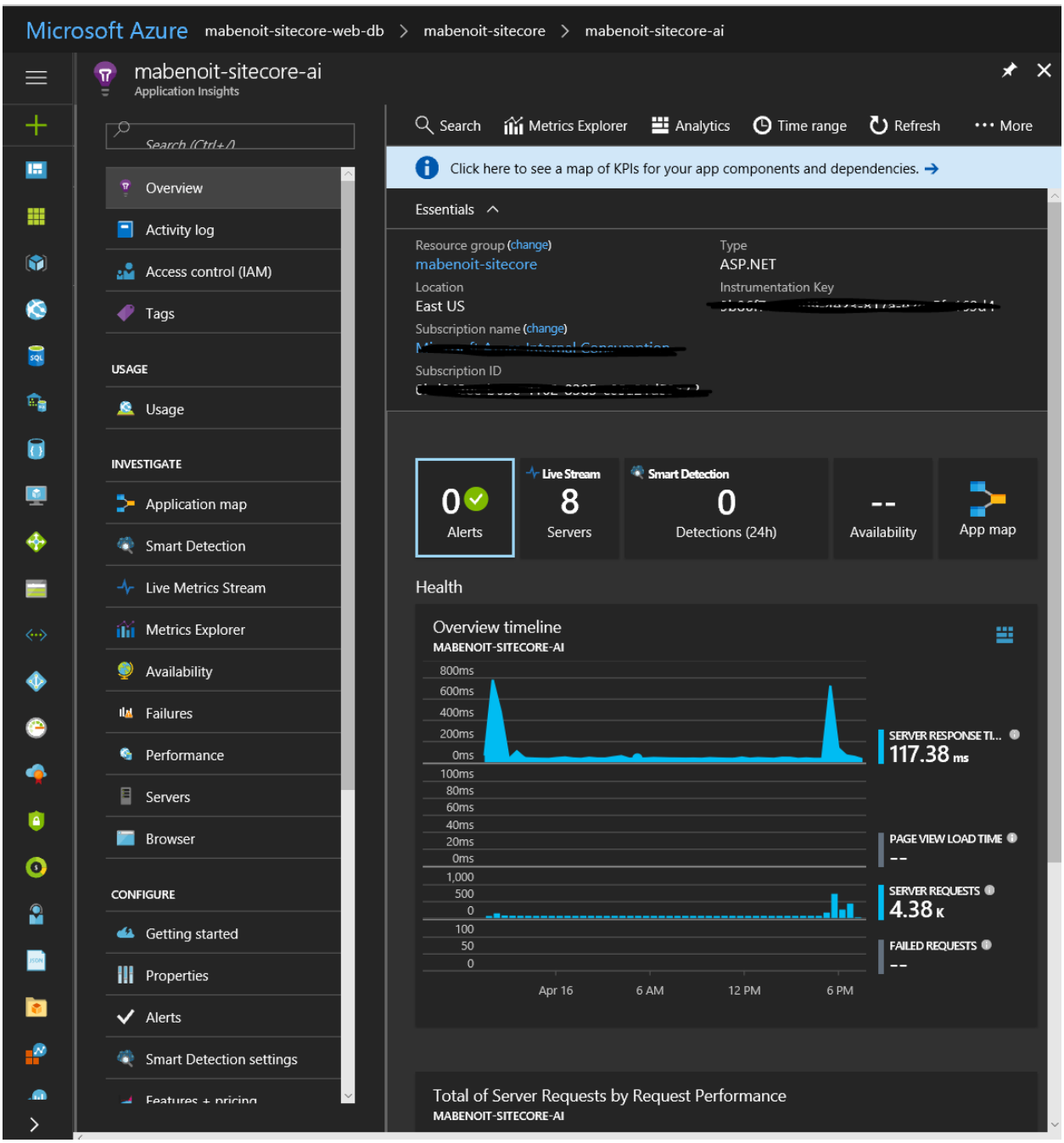
16. By clicking on the "**Search**" button you should get the following result:



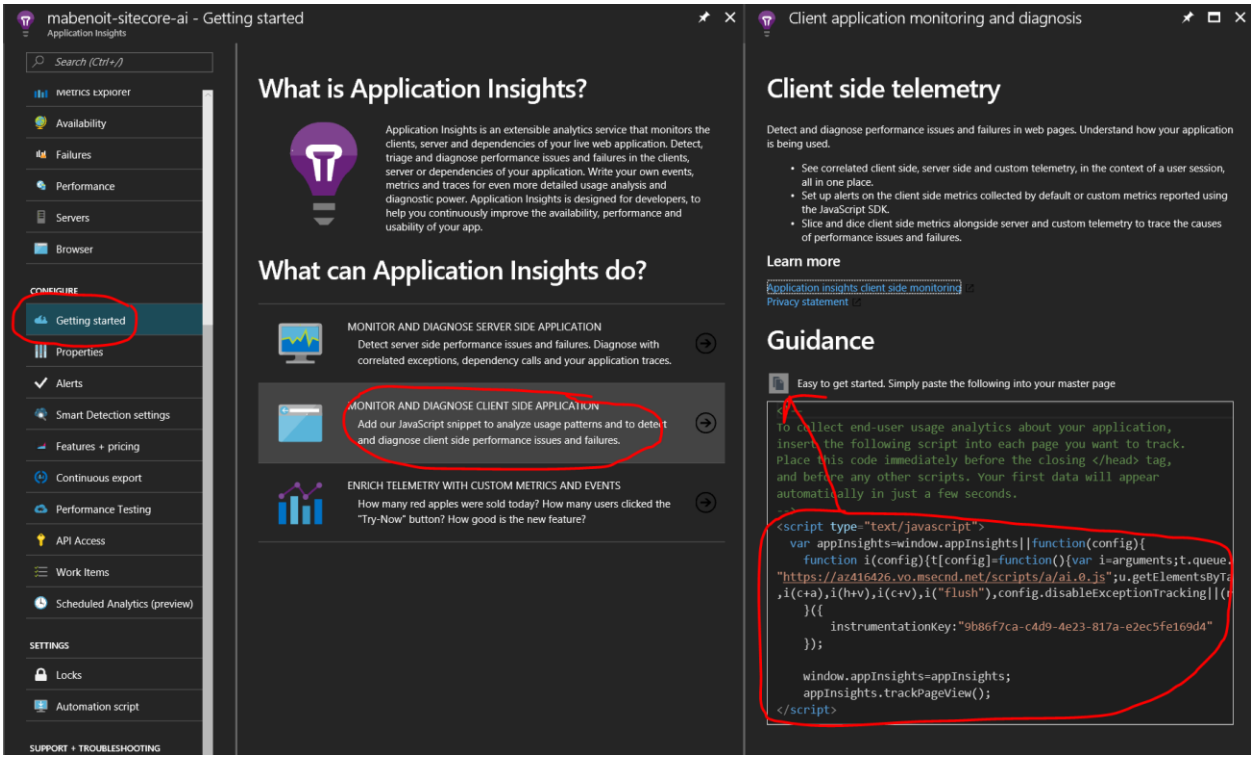
Task 2: Application Insights (20 min)

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.

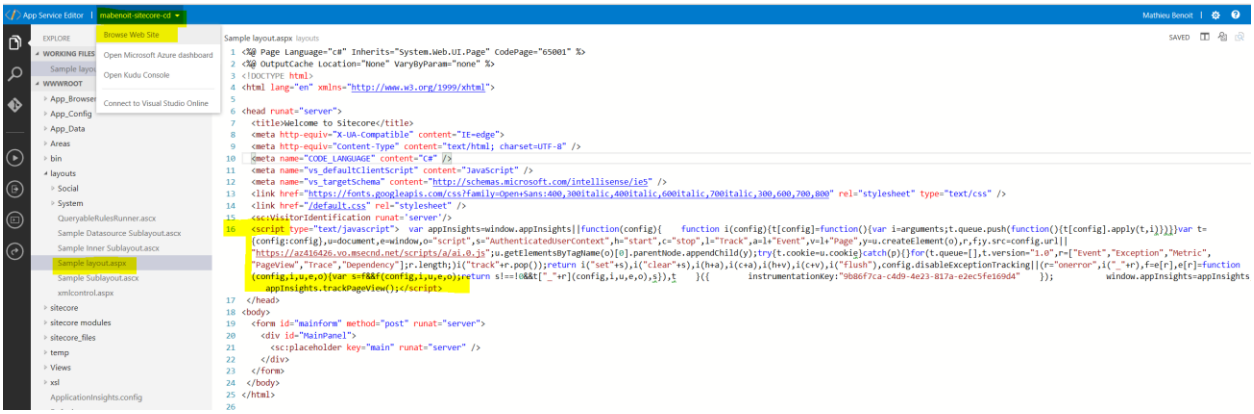
1. Go to https://portal.azure.com/ and select your "…-ai" Application Insights resource.



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



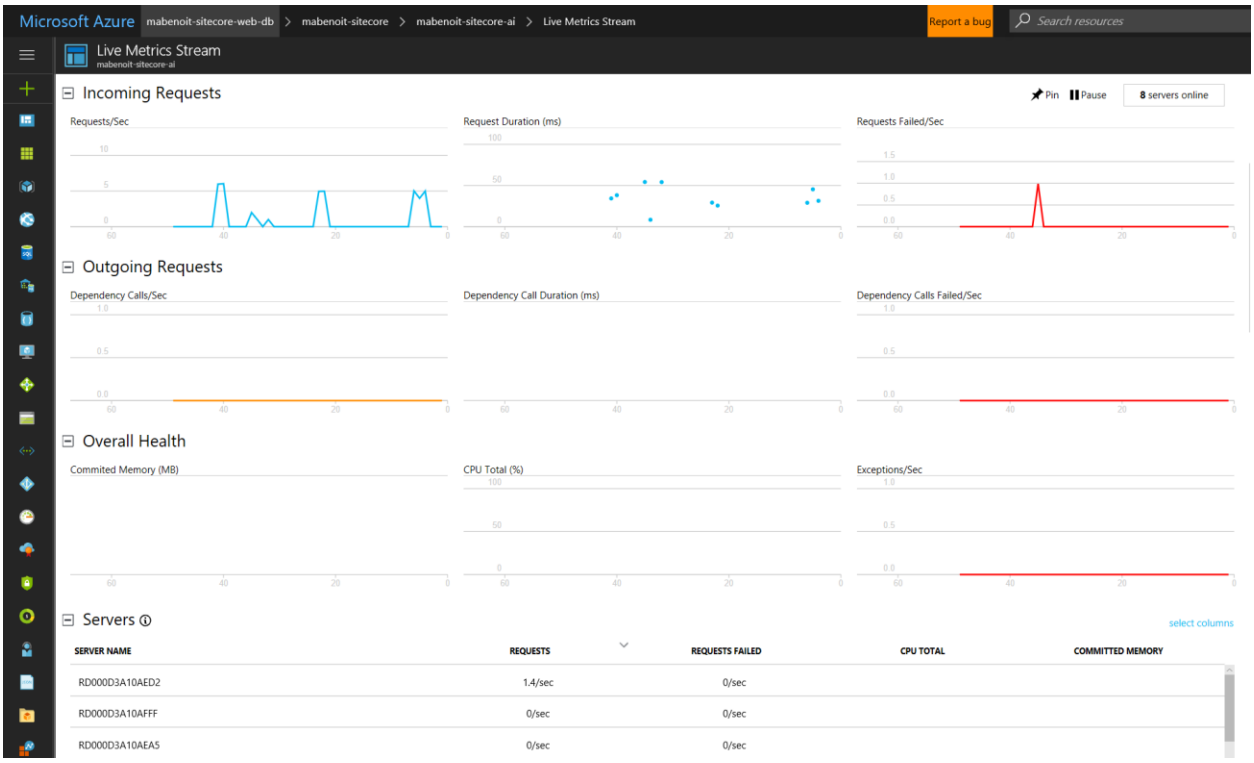
3. Open the "**App Service Editor (Preview)**" blade of you CD Azure Web App, open the "/wwwroot/layouts/Sample layout.aspx" file and edit it like explained above by inserting the Javascript just before the closing </head> tag.



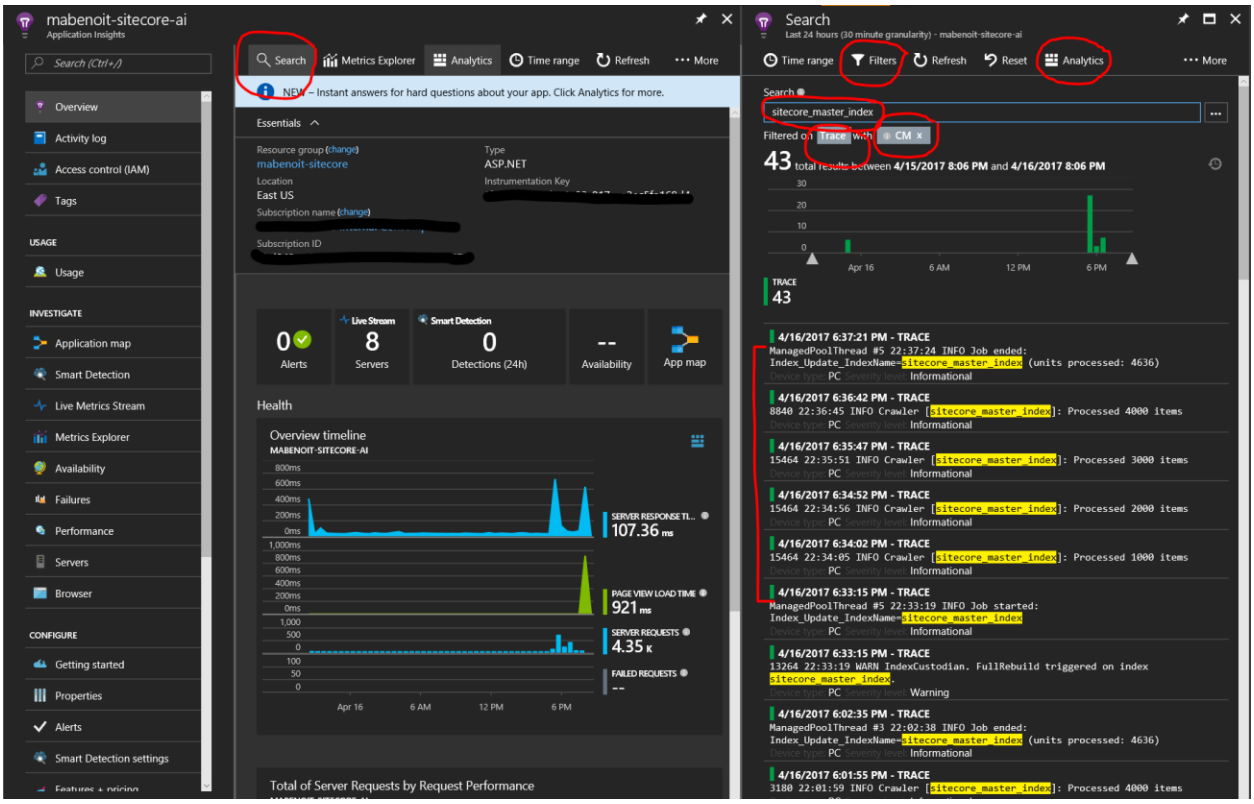
4. Then click on the "**Browse Web Site**" toolbar action to navigate to your CD website. There, do some refresh on the homepage to generate some events. We will use and display them with the following steps above.

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.

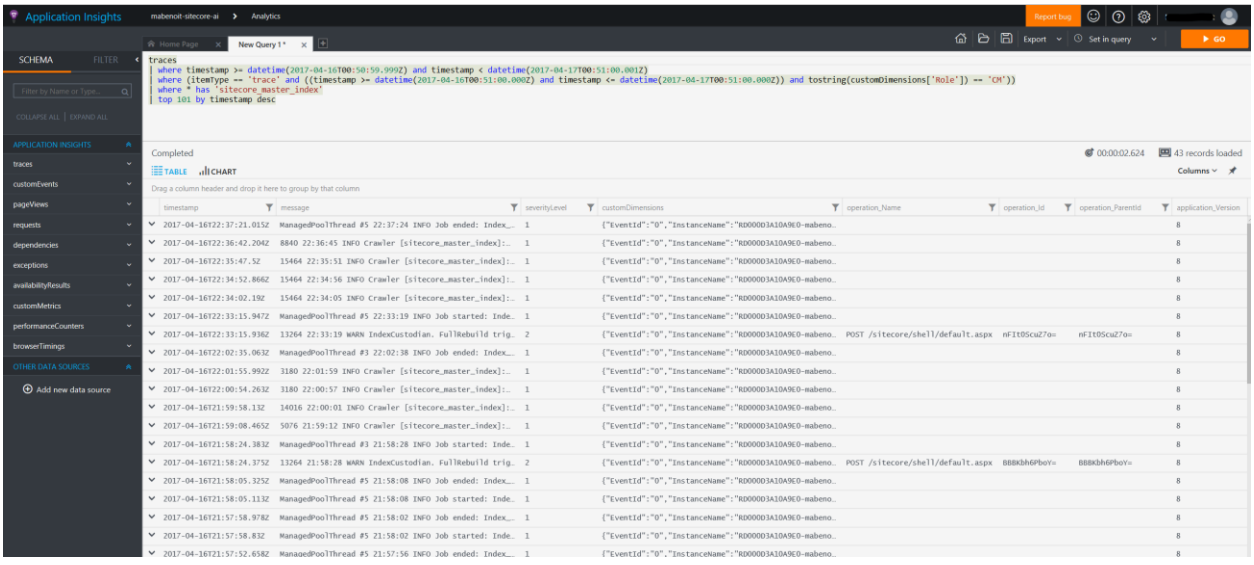
5. Go back on the Azure portal on the Application Insights resource blade, go to the "**Live Metrics Stream**" blade and open in another web browser your CD website URL and do some refresh on the Homepage to generate some server request.
6. On the "**Live Metrics Stream**" blade you could see the generated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



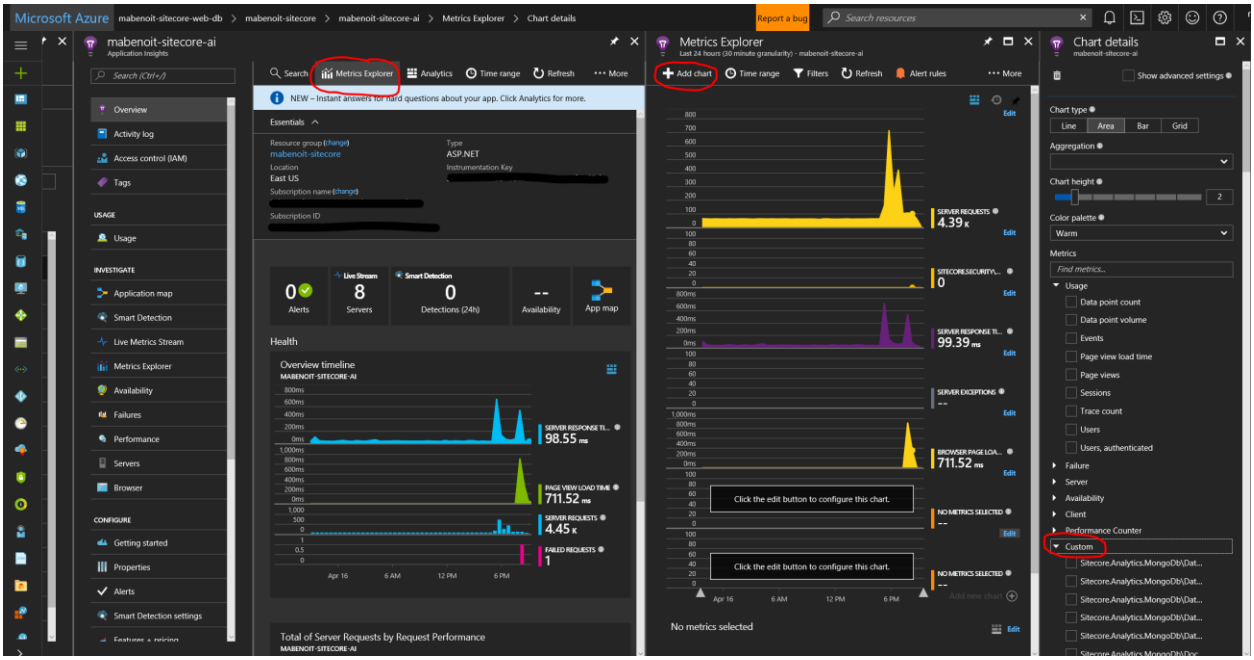
7. Go back on the "**Overview**" resource blade of the Application Insights resource, click on the "**Search**" toolbar action. Like mentioned earlier during this training, by default Sitecore logs all in Application Insights (not anymore as File systems like used to do). On the image below you could 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.



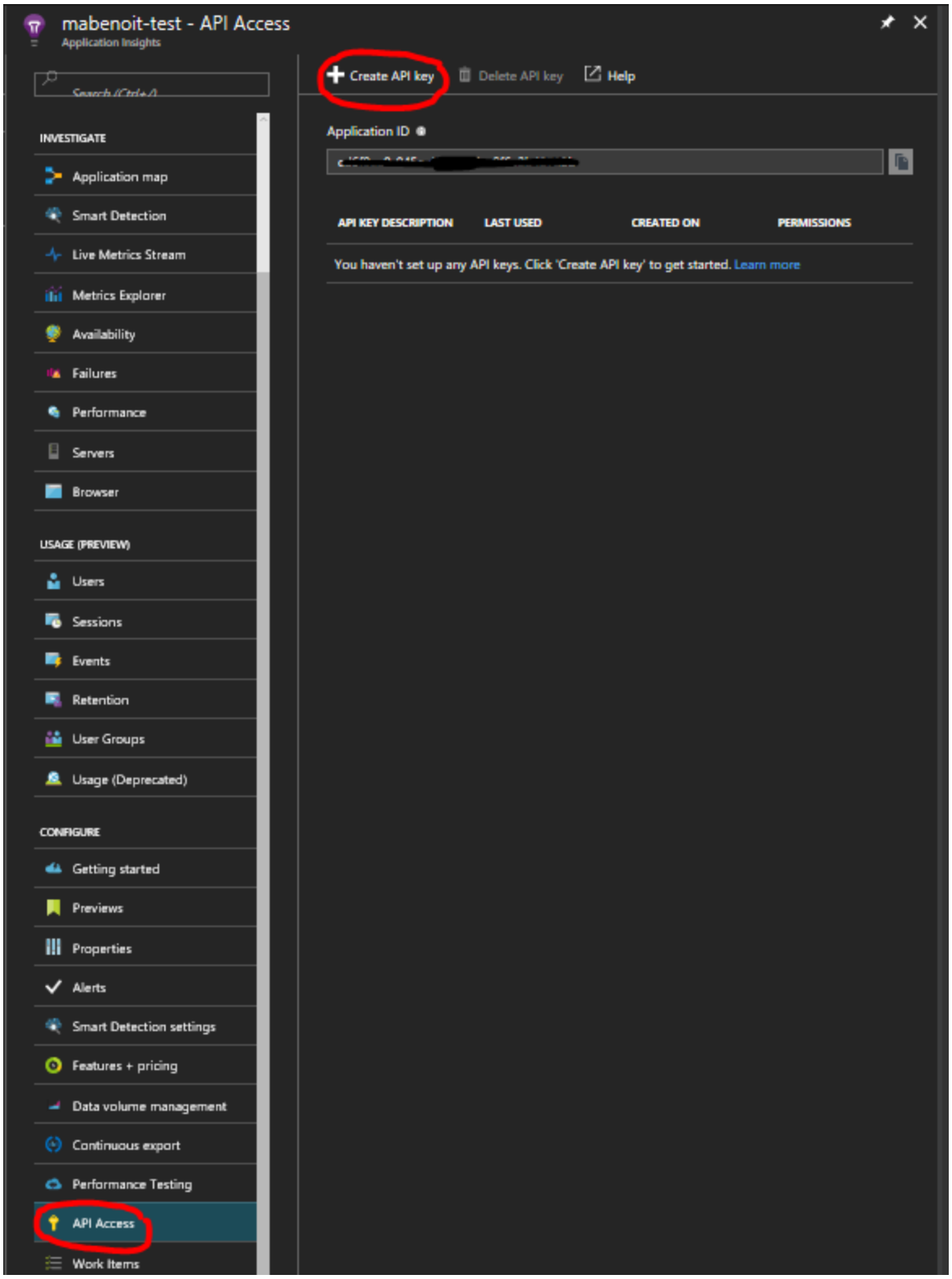
8. From there, you could then click on the "**Analytics**" toolbar button, you will navigate to the Application Insights Analytics tools to perform more advanced queries within a powerful editor.



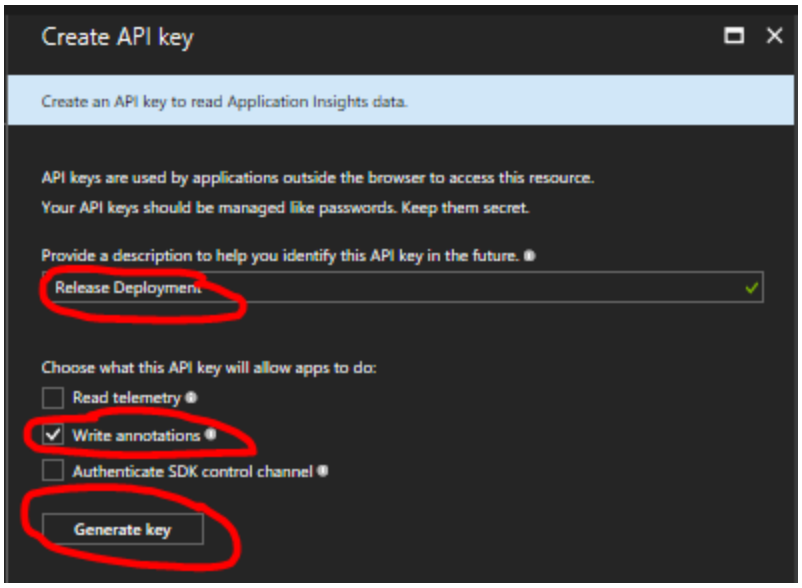
9. Go back on the Azure portal, on the Application Insights resource go to the "**Overview**" blade, click on the "**Metrics Explorer**" toolbar action.



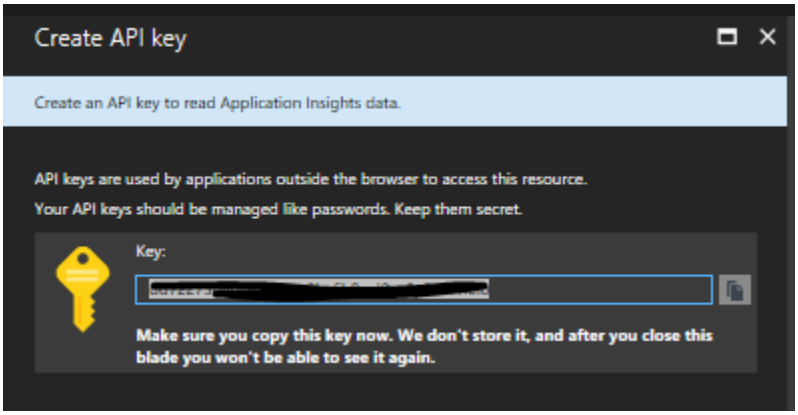
10. From there you could “**Add chart**” (or “**Alert rules**”) based on built-in metrics or Custom like illustrated provided by Sitecore.
11. Now let’s add some annotations within these charts to be able to see when a release has been deployed and the impact it could have with the metrics for example.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



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

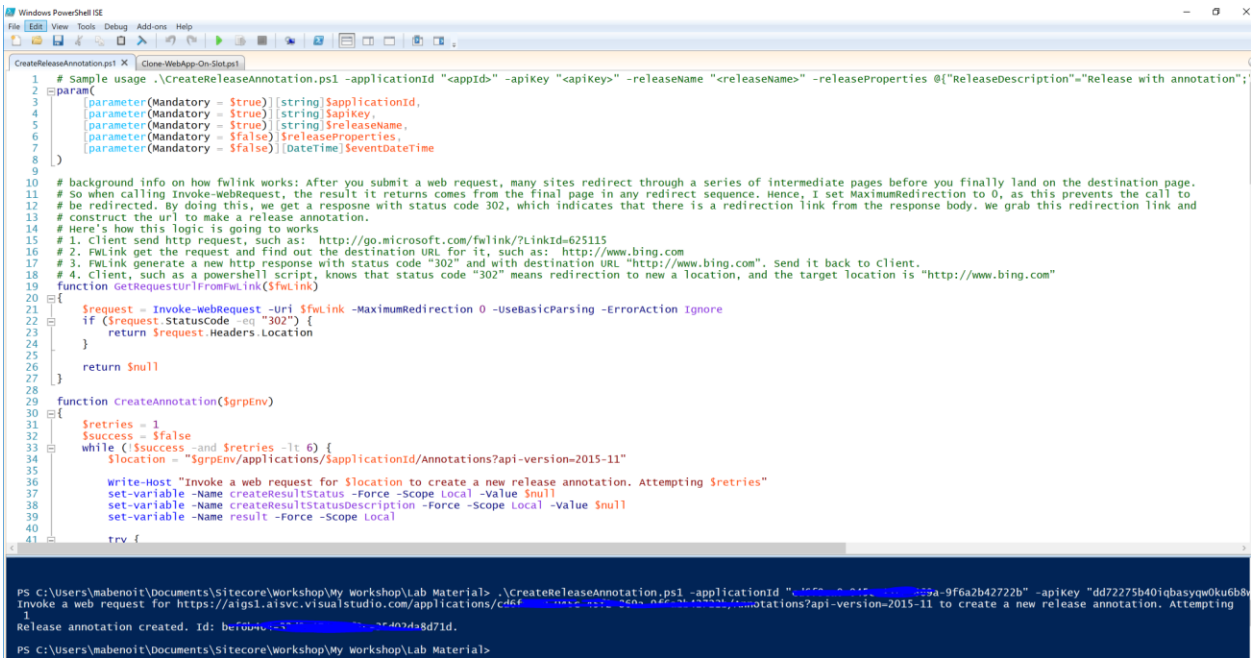


13. Then, make sure you copy this generated key. We will use is soon and like explained “after you close this blade you won’t be able to see it again”.

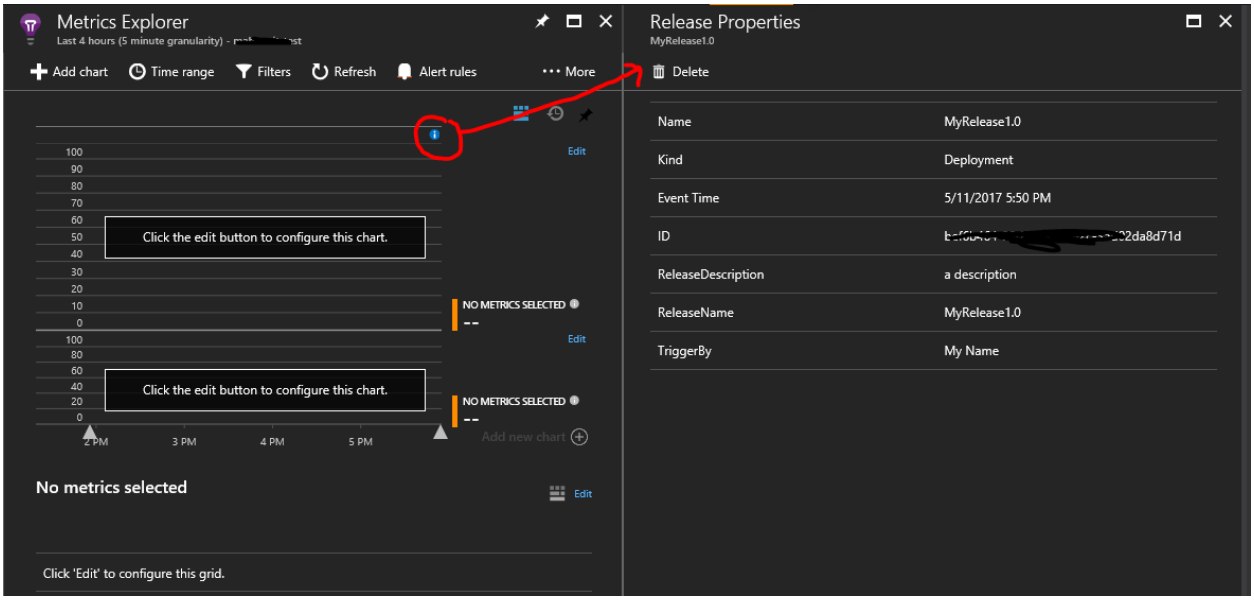


14. Make a local copy of the [Powershell script from GitHub](#).
15. Right-click on it and select “Edit” to open it with PowerShell ISE tool and run the following PowerShell command:

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



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):



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

- 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
- 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_azure_search
- 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 Framwork/Service and see the key features.

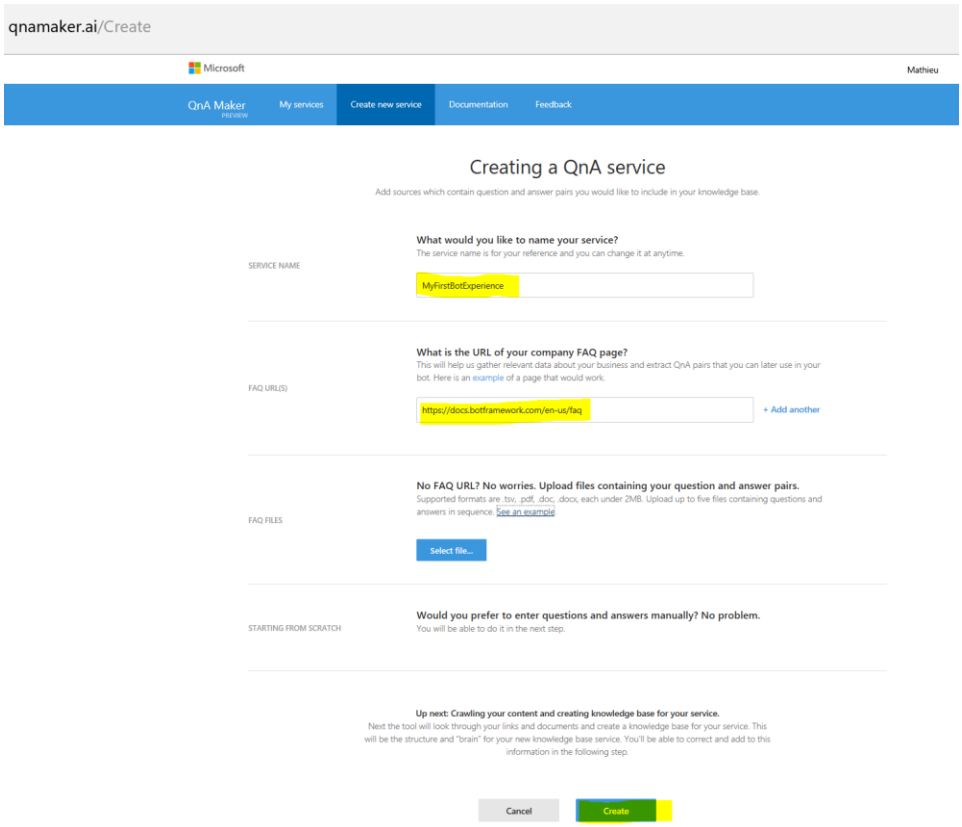
Through this exercise, you will play/use:

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

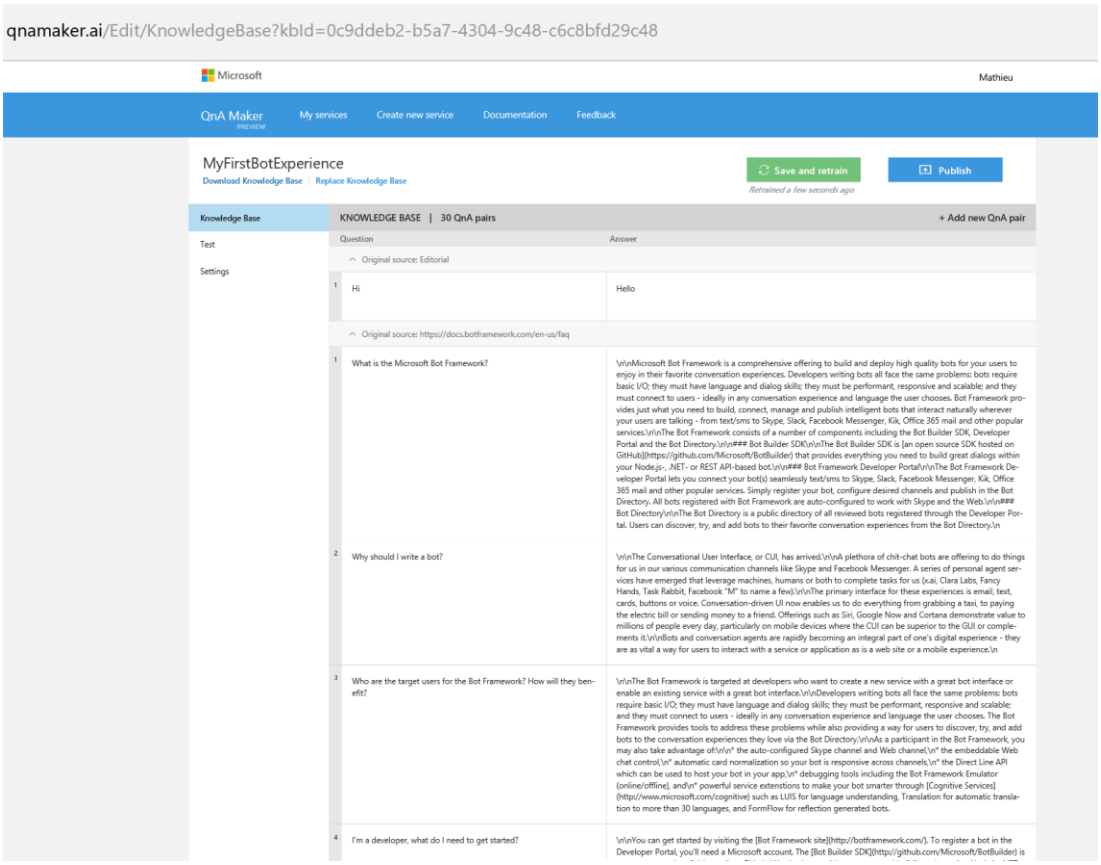
Task 1: QnA Maker (10 min)

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.

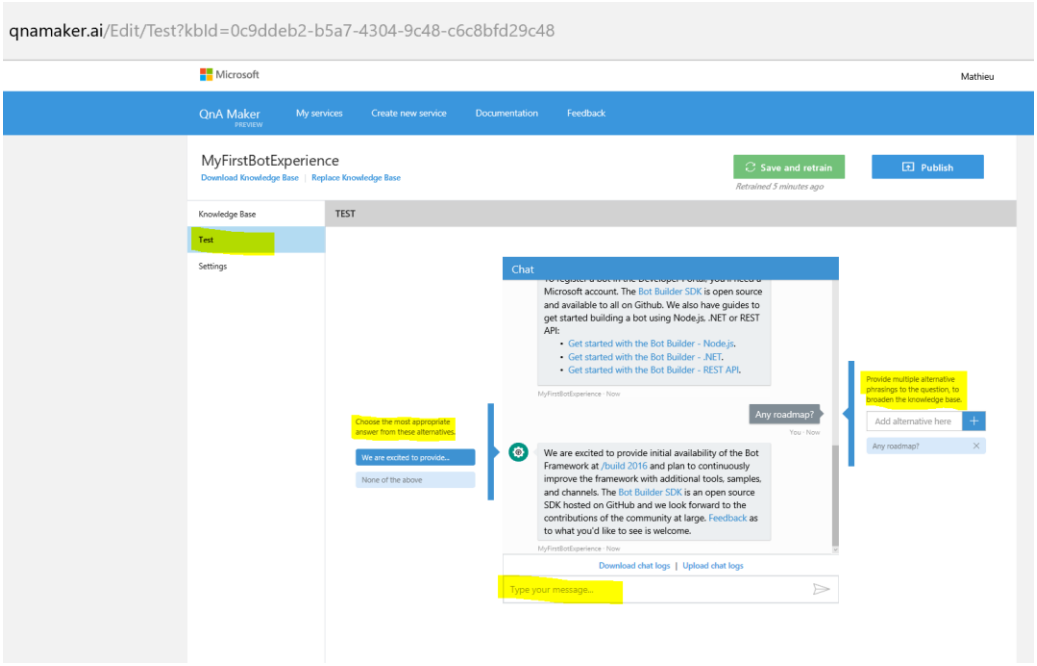
1. Go to https://qnamaker.ai/
2. You will be asked to sign-in.
3. Then go to the "**Create new service**" tab and fill out all the fields below and click on the "**Create**" button:
    - **Service Name**: MyFirstBotExperience
    - **FAQ URL(s)**: https://docs.botframework.com/en-us/faq



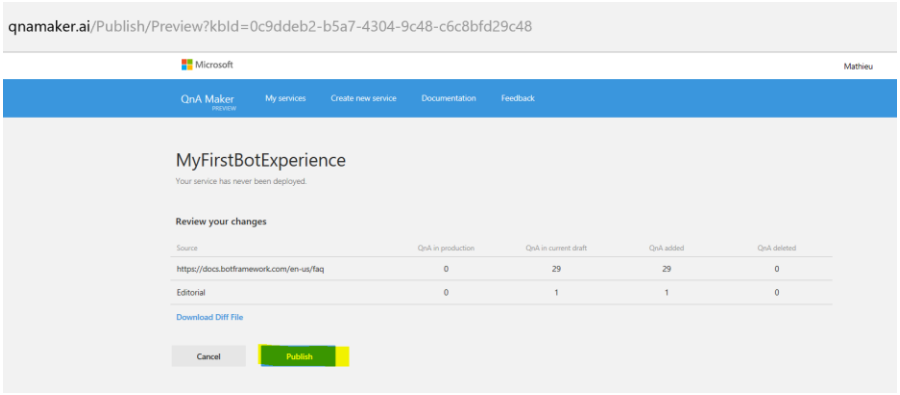
4. After few seconds, your service is provisioned, and you could see the different key/value pairs prepopulated (you landed on the "**Knowledge Base**" tab after the creation):



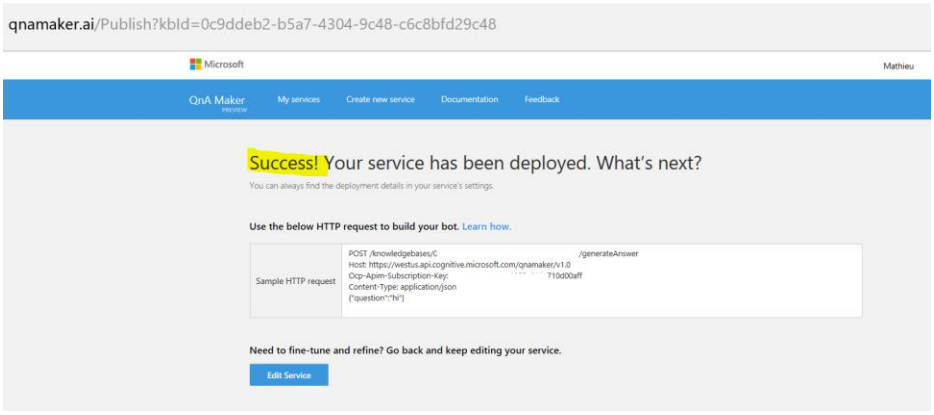
5. Go to the "**Test**" tab and ask the bot with some questions like:
    - Any roadmap?
    - How to get started?
    - What's a bot?



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.



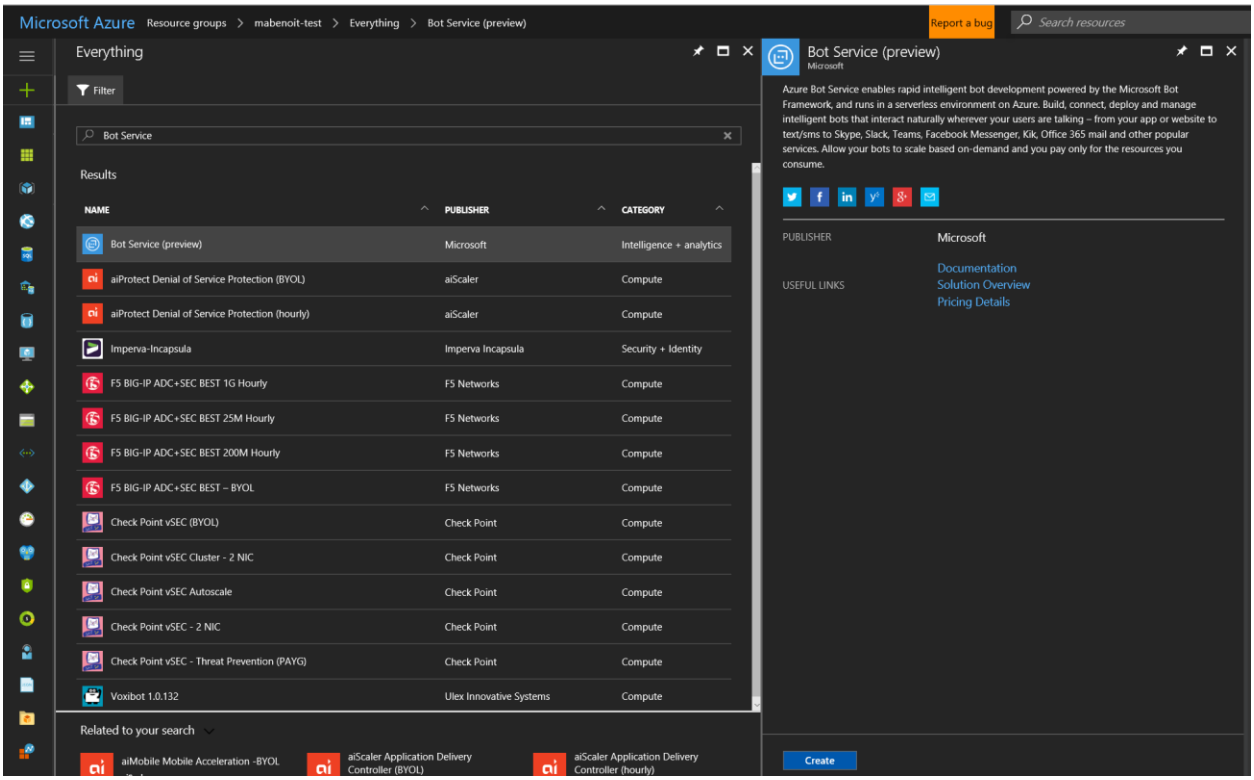
8. Few seconds after, your QnA Maker service is exposed via its URL.



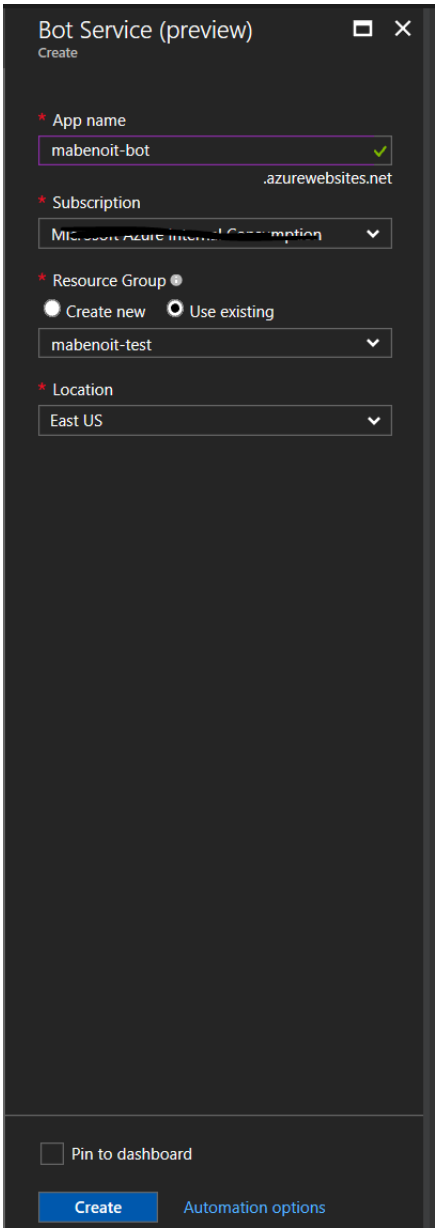
Task 2: Bot Service (10 Min)

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.

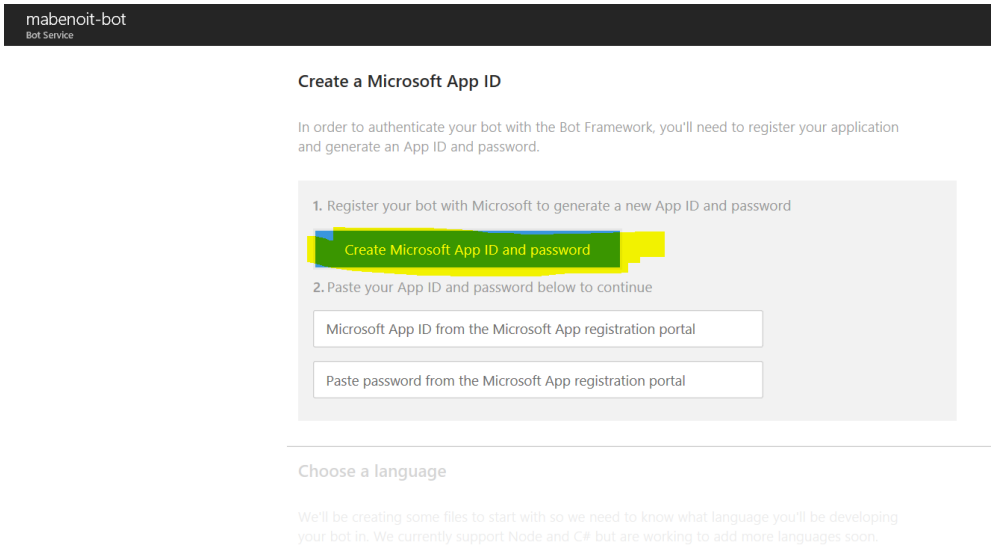
1. Go to the Azure portal at https://portal.azure.com
2. Click on the add button to provision a new service and look for "**Bot Service**", then click on the "**Create**" button.



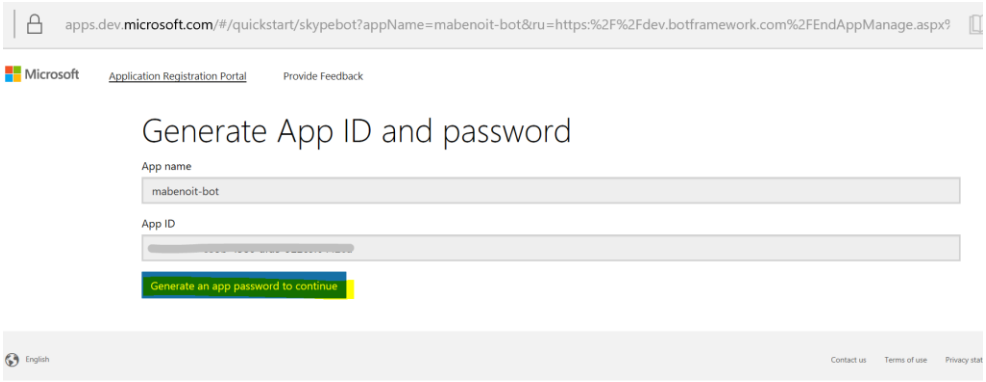
3. On the "**Bot Service (preview)**" creation blade, fill out the fields accordingly and then click on the "**Create**" button:



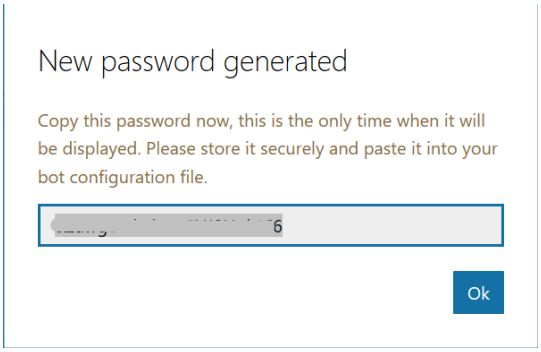
4. Few seconds after the provisioning will be completed, you will land on the default blade of your new Bot Service. Click on the "**Create Microsoft App ID and password**" button.



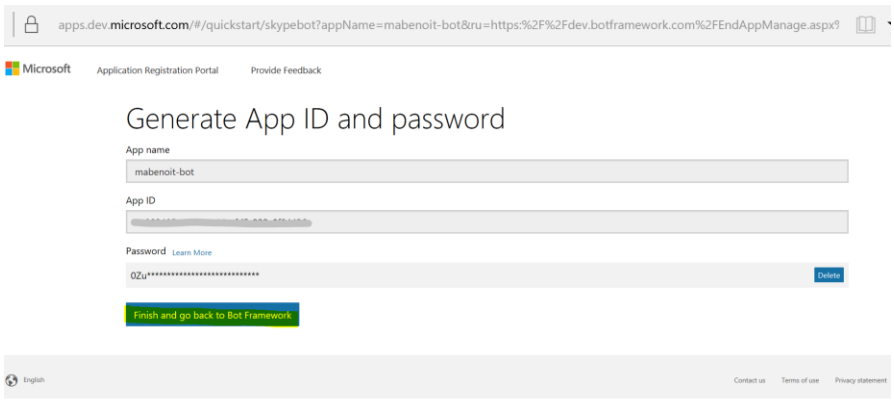
5. You will be redirected through a sign-in process to an associated "**Generate App ID and password**" page. Click on the "**Generate an app password to continue**" button.



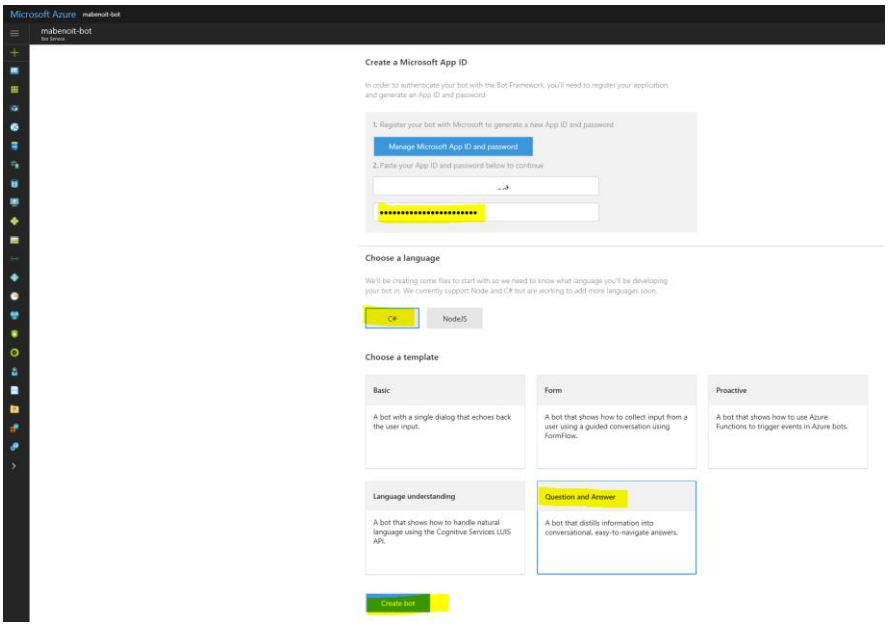
6. Then copy the "only time generated" password.



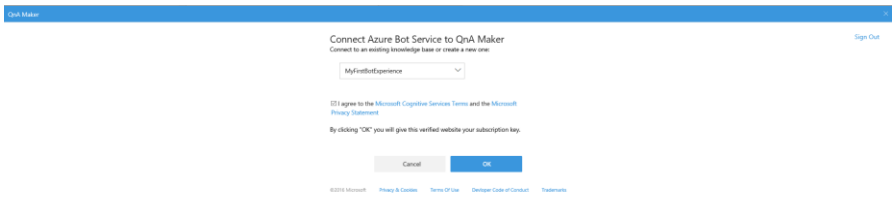
7. Then click on the "Finish and go back to the Bot Framework" button.



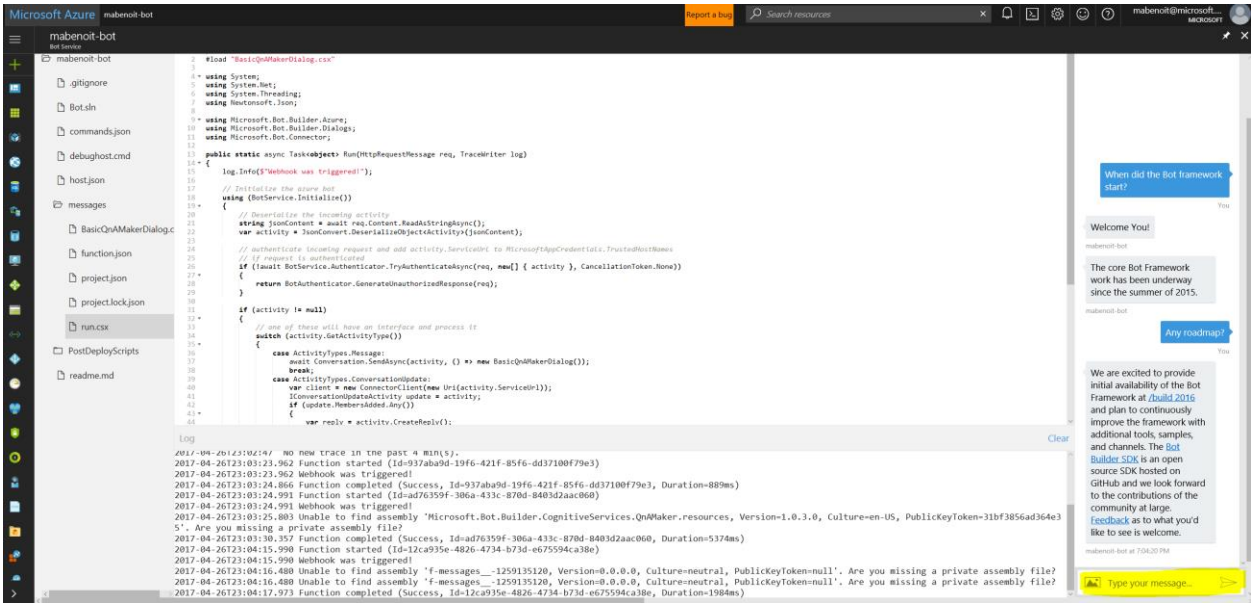
8. You will land on the Azure portal, copy/paste your generated password in the associated field, choose C# as the language and select the "**Question and Answer**" template. Then click on the "**Create bot**" button.



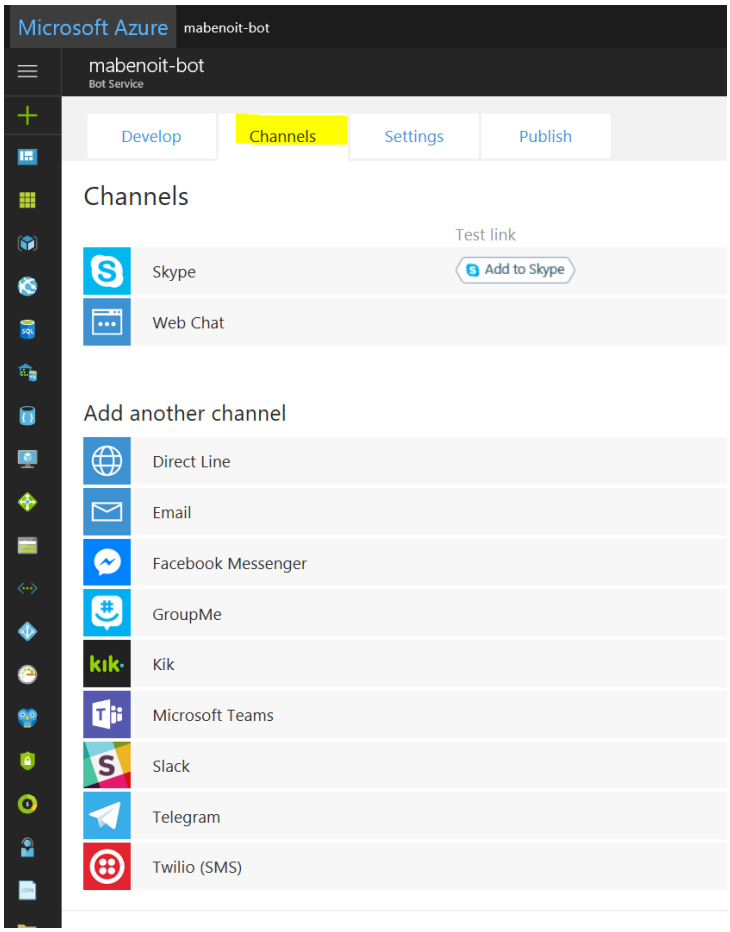
9. Select the associated (previously created) QnA Maker service and check the agreement part and click on the "**OK**" button.



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.



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.



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.

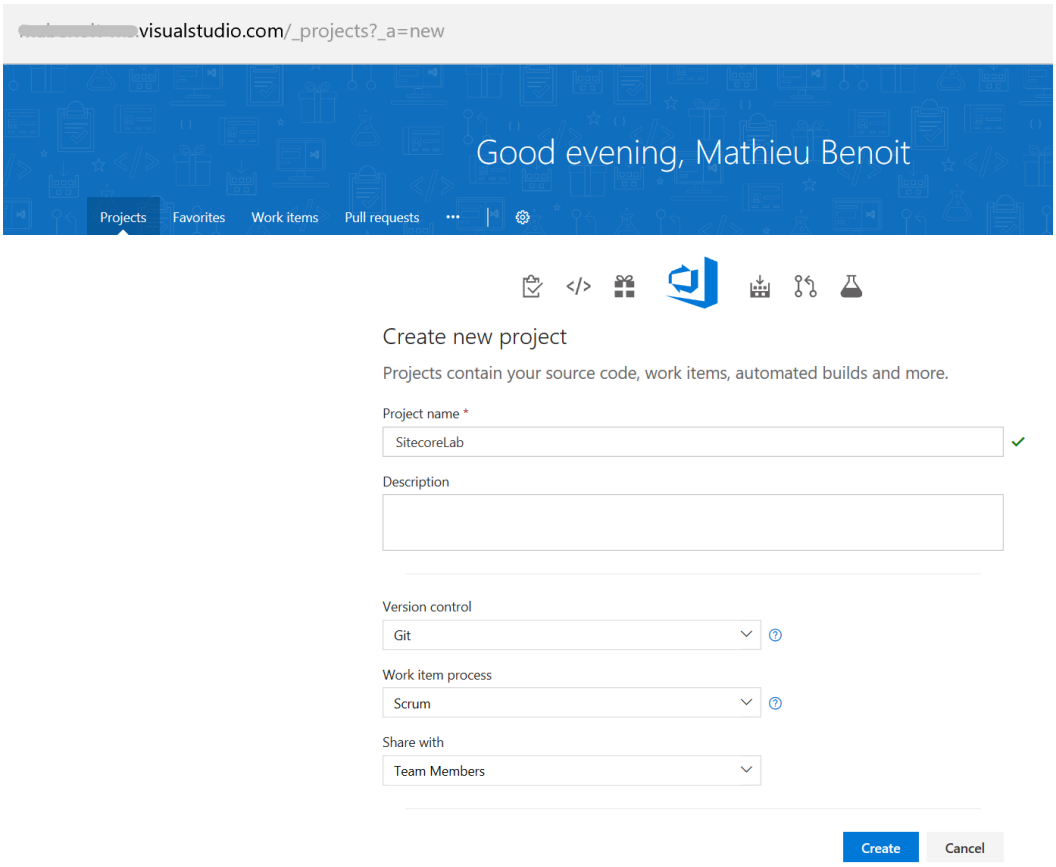
Through this exercise, you will play/use with:

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

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

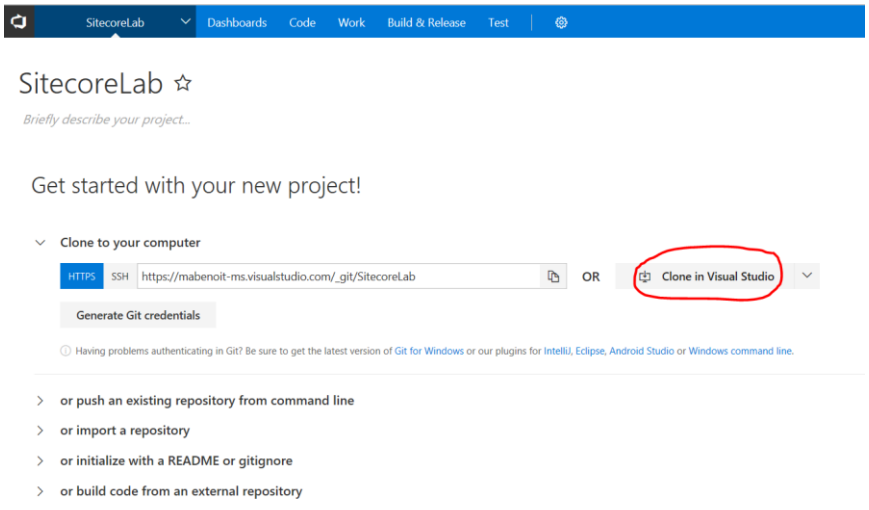
In this section, the attendee will create a new VSTS account and his/her first VSTS project.

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

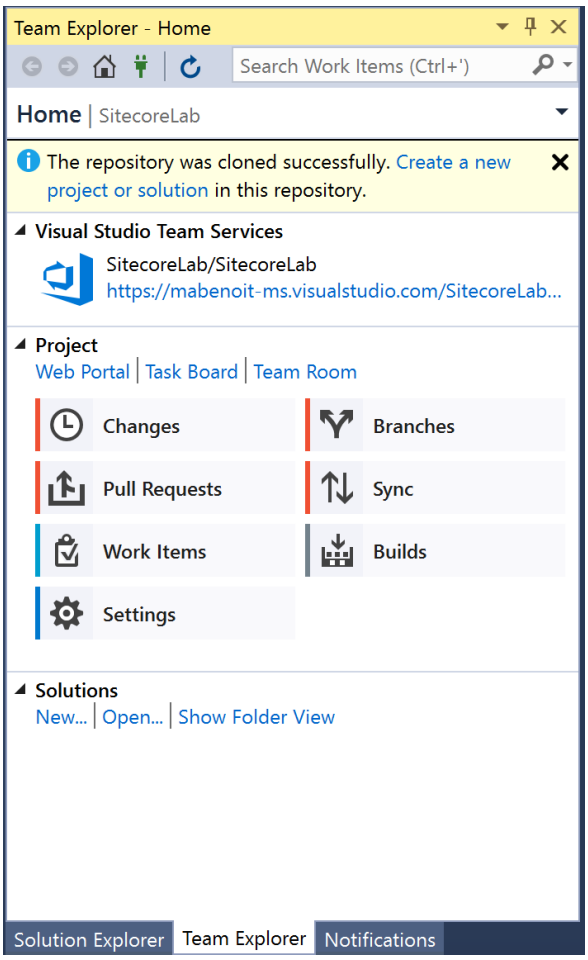


Task 2: Code – Version Controlling with Git (15 min)

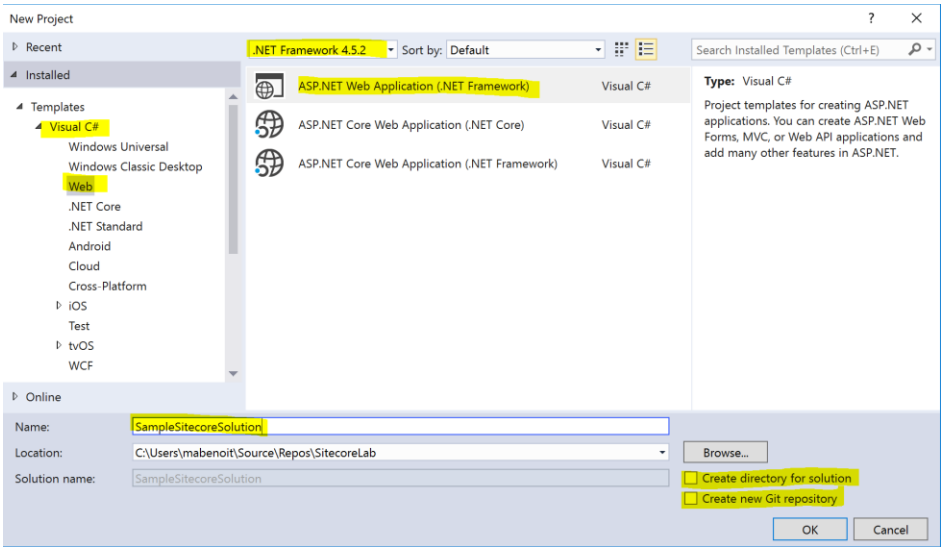
1. Once created, you could click on the "**Clone in Visual Studio**" button.

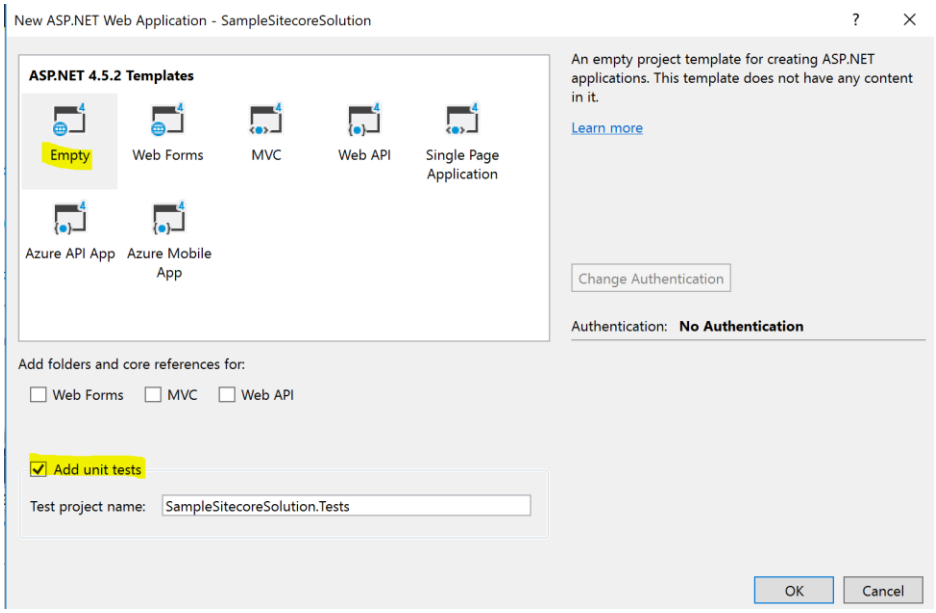


2. Once successfully cloned, you should see this message within the Visual Studio View "Team Explorer":

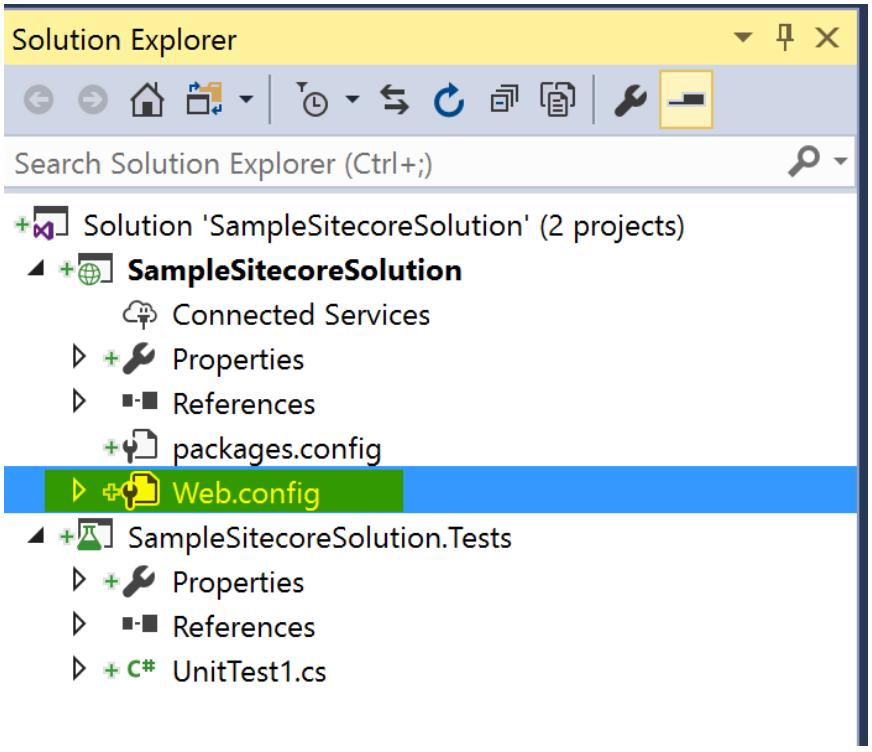


3. You could click on "**Create a new project or solution**".



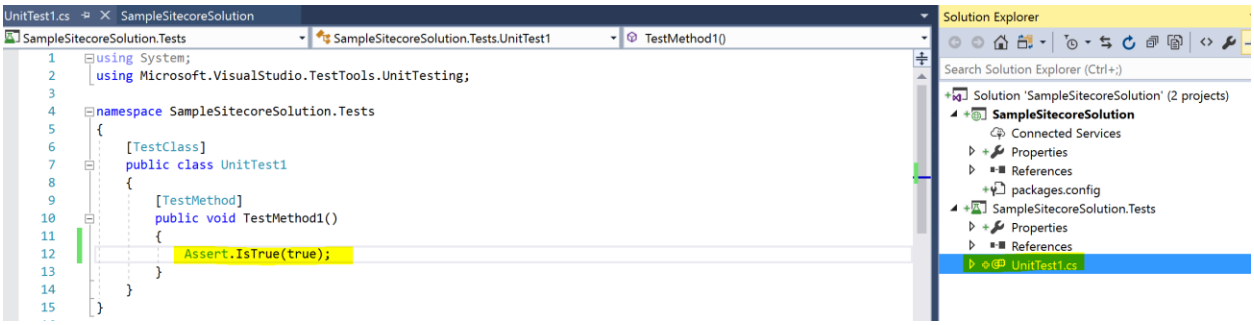


4. Then, the solution with the 2 projects "**SampleSitecoreSolution**" and "**SampleSitecoreSolution.Tests**" will be created.

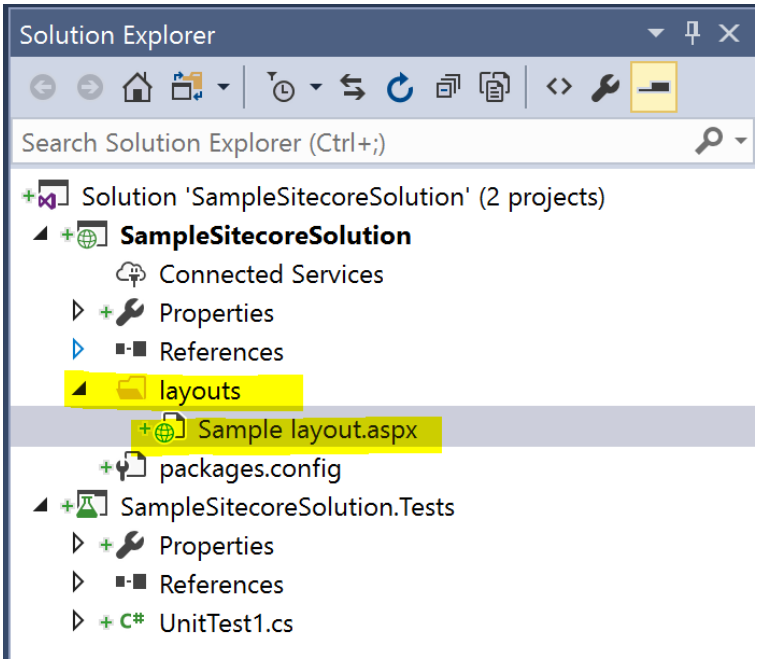


5. In the "**SampleSitecoreSolution**" project, remove the **web.config** file (to avoid overriding the Sitecore one when we will deploy the solution later).
6. In the "**SampleSitecoreSolution.Tests**" project, update the **UnitTest1.cs** file by inserting the code below into the **TestMethod1()** method:

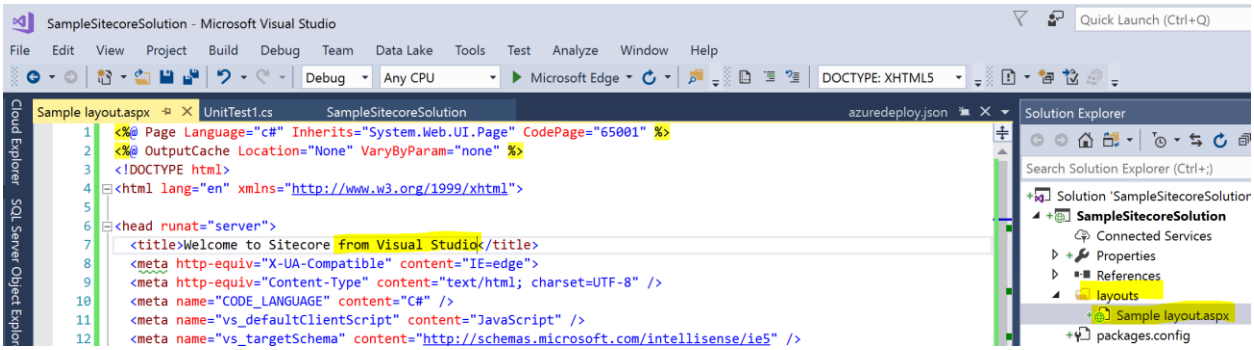
```
Assert.IsTrue(true);
```



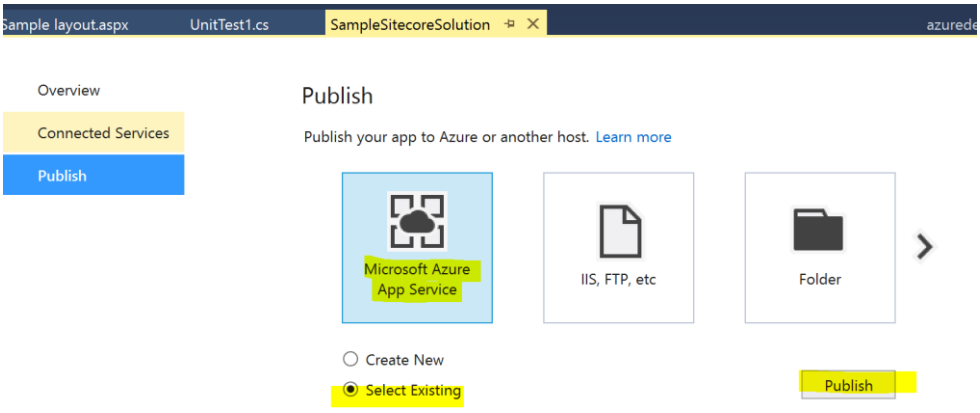
7. In the "**SampleSitecoreSolution**" project, create a "**layouts**" folder and create an empty "**Sample layout.aspx**" file.



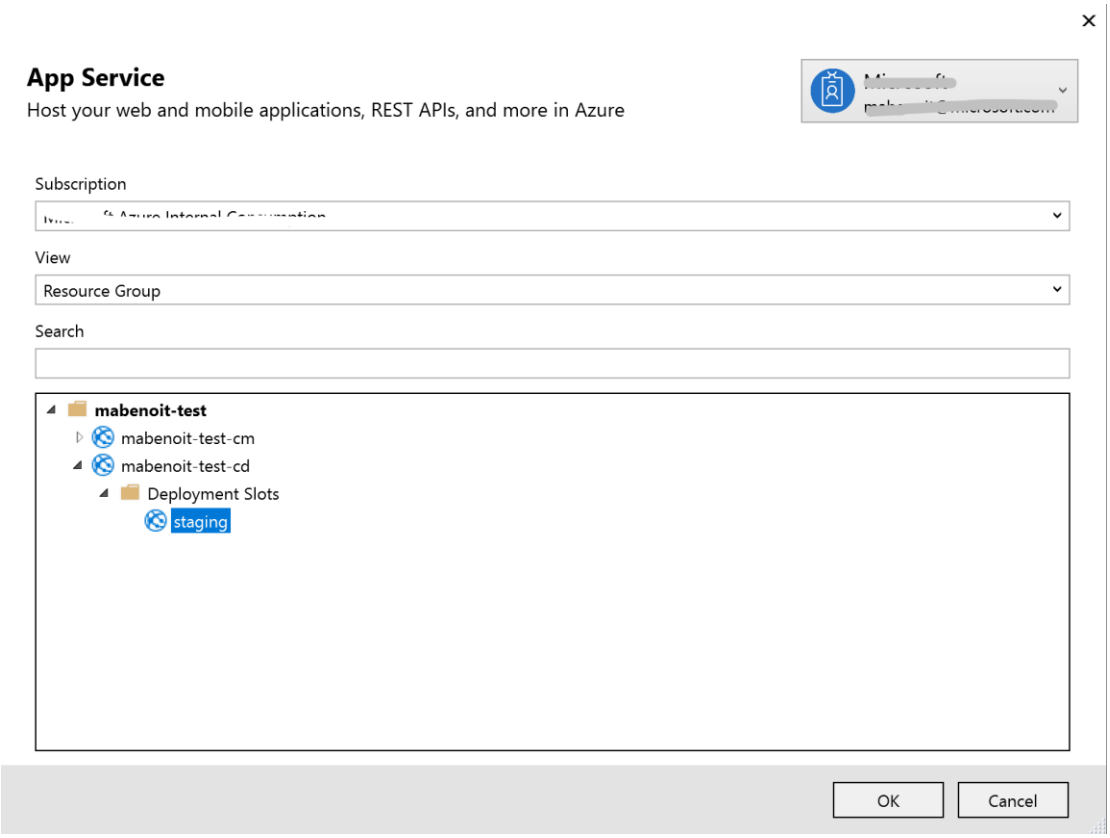
8. Retrieve the content of this file from the "**App Service Editor (Preview)**" feature of the Azure Web App (CD) to populate this file and change the <title> tag by adding the suffix " **from VSTS**" for example.



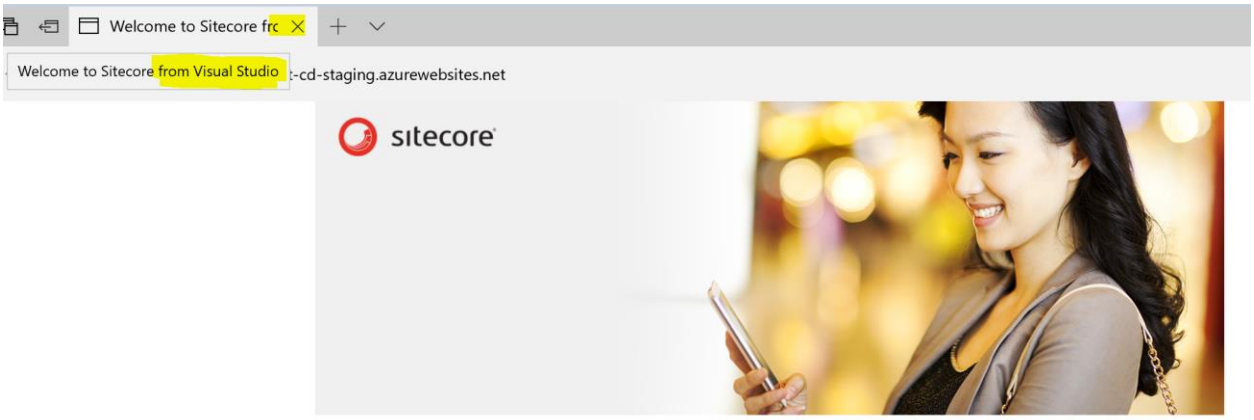
9. Right-click on the "**SampleSitecoreSolution**" project, and choose "**Publish**". Then choose "**Microsoft Azure App Service**" and "**Select existing"** and click on the "**Publish**" button.



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

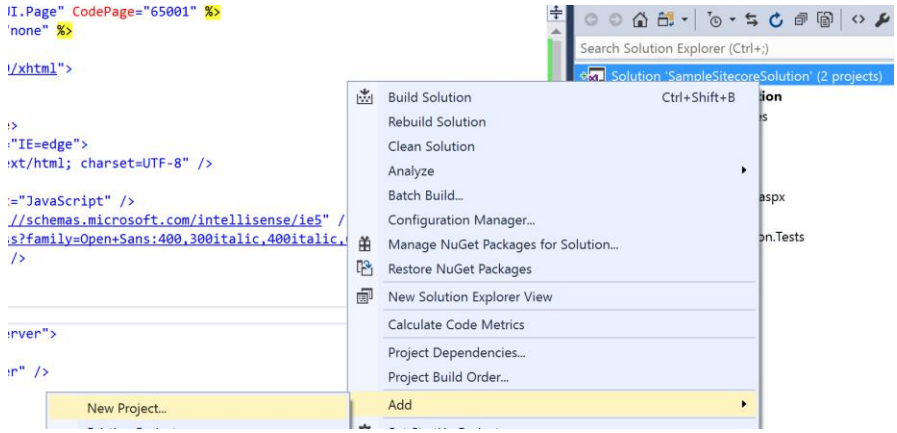


11. After the application is deploying and warming up again, the web browser will automatically be opened, and you could see your new title:

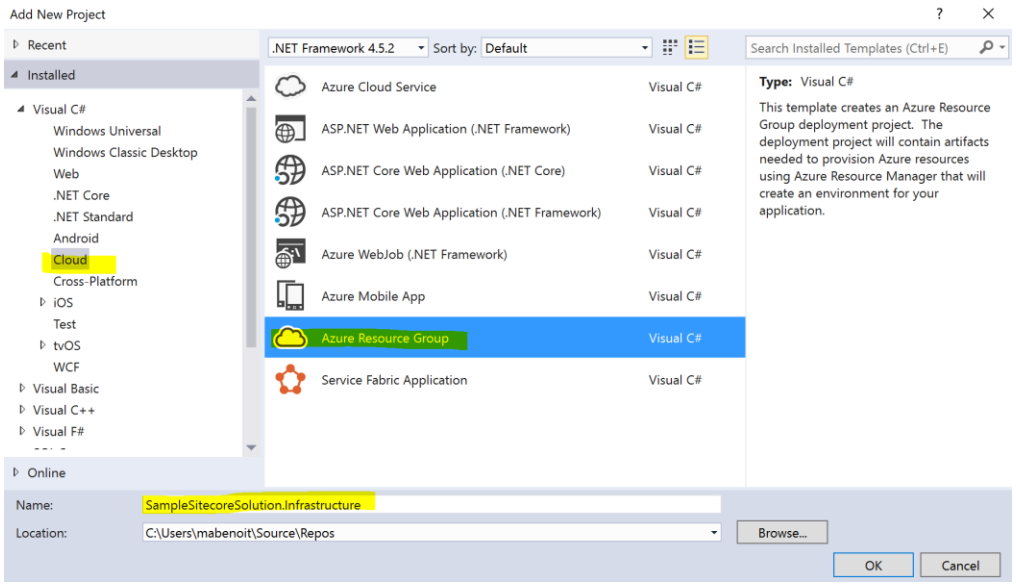


12. In the "**layouts/Sample layout.aspx**" file change the prefix " **from Visual Studio**" by " **from VSTS**" within the <title> tag.

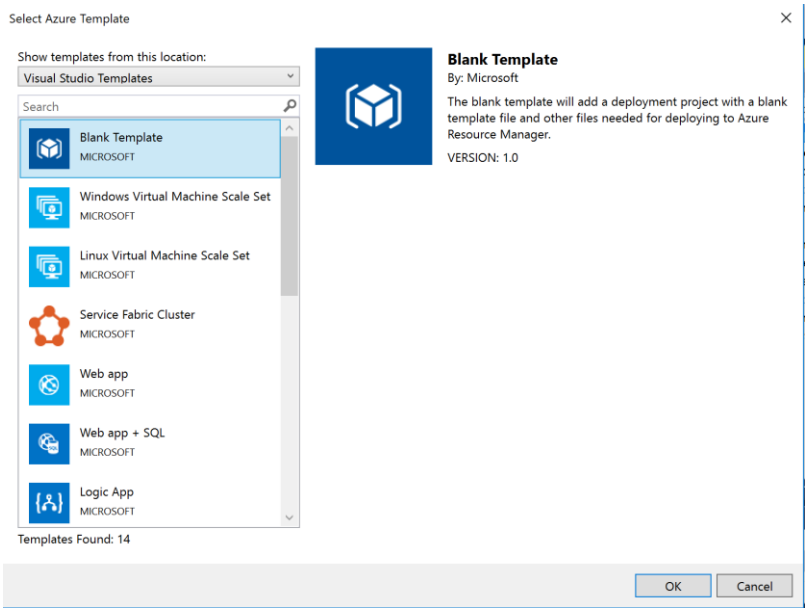
13. Now, let's create a new project within the solution by right-clicking on it and go to "**Add > New Project…**":



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



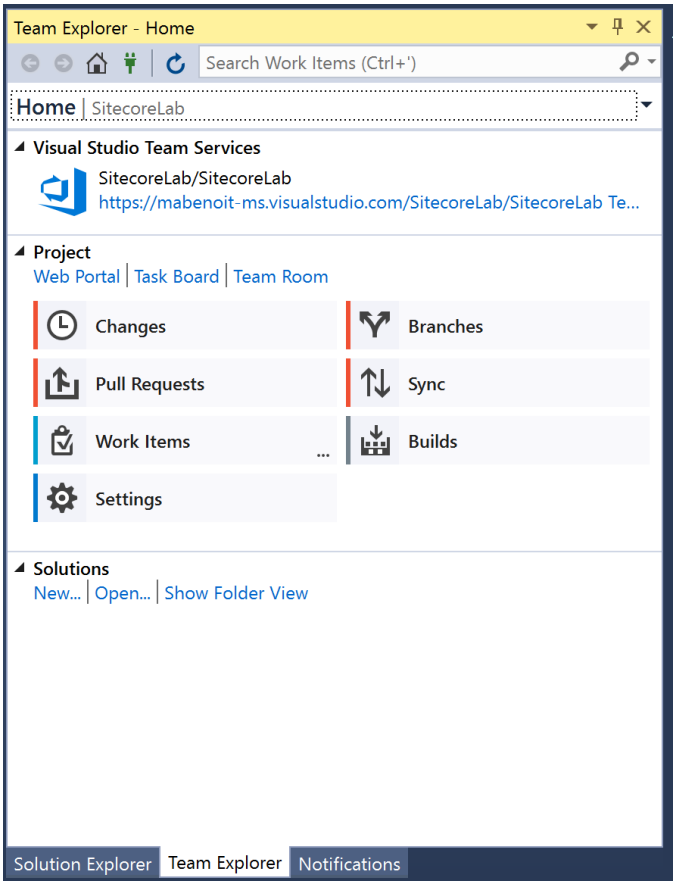
15. Then select the "**Blank Template**":



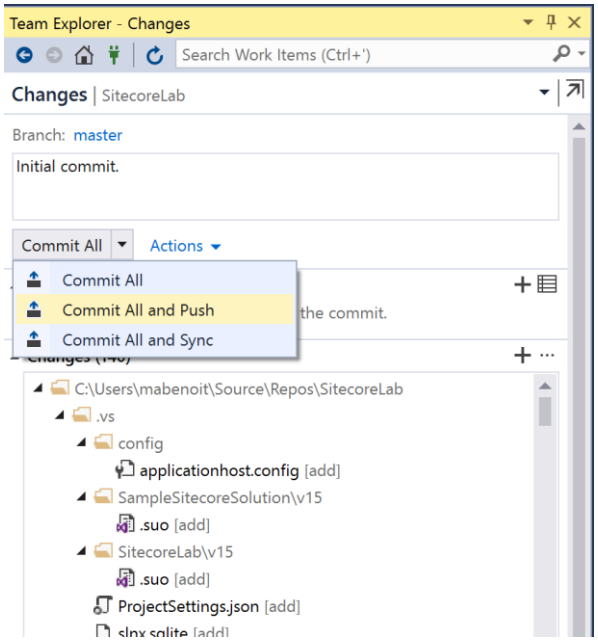
16. Retrieve the content of these 3 files (**.json, .parameters.json and .ps1**) based on what we did earlier in this lab with the Custom ARM Templates section to replace the content of these default files created accordingly.

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.

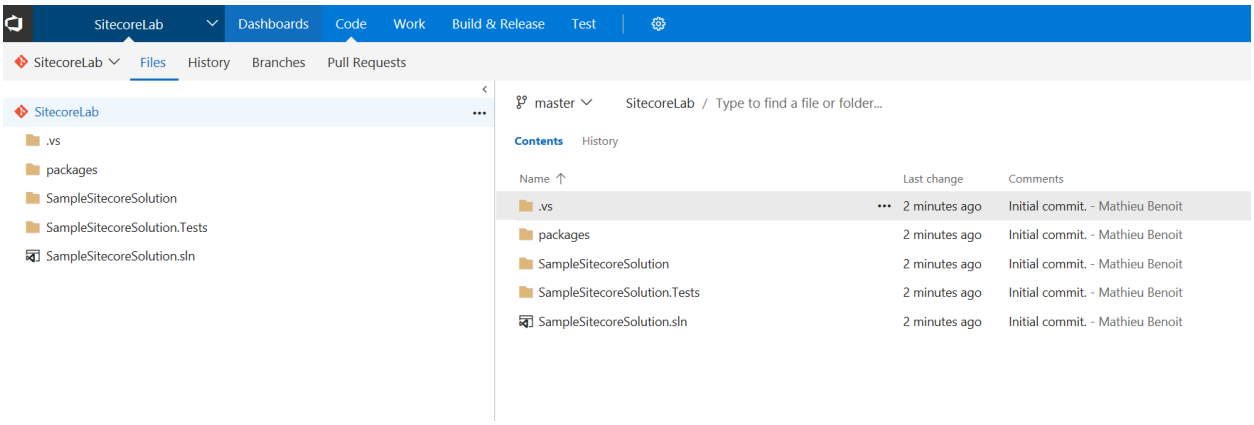
17. Go to the **Team Explorer** view on the right-hand panel and click on "**Changes**".



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.



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.msdn.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/