Azure PCI and HIPAA validated solution

Blueprint, Reference Architecture with ARM templates

# PROJECT OBJECTIVE, SCENARIO, Solution

The objective of this solution is to illustrate how a secure and compliant solution would be deployed end-to-end Azure solution. This solution contains several components.

1. **A BluePrint** – The blue print provides you an understanding on how Contoso achieved it’s compliant state. Included in the solution package is a completed PCI – DSS responsibility Matrix answered for Contoso Health
2. **Reference Architecture** – The Reference Architecture provide the design used for the Contoso Health solution.
3. **ARM templates** – In this deployment the .JSON files provide Microsoft Azure to {auto} deploy the components of the reference architecture once the configuration parameters are provided in the set up.

* **PowerShell Scripts** The scripts provided by [Avyan Consulting solution](https://github.com/AvyanConsultingCorp/pci-paas-webapp-ase-sqldb-appgateway-keyvault-oms) . The scripts provided consist of:
  1. Pre-installation process that establishes Directory user roles, and [EXPLAIN WHAT’S in THE PREINSTALLER]
  2. Post-installation process that deploys an [ARM template, web front end runtime, and SQL backpack](https://github.com/Microsoft/azure-sql-security-sample) built by the Microsoft SQL team, and revised for this scenario by Avyan Consulting. The Contoso Clinic Demo Application builds out the web application, and SQL database to illustrates a before and after data masking [CONTINUE TO EXPLAIN POST INSTALLER]

# USER SCENARIO

This demo is outlined to provide the opportunity to illustrate the following end to end solution (Note this solution requires a **paid subscription on Azure, it will not work with a Trail subscription)**:

A small medical clinic called Contoso Health, is ready to move their patient intake, and payment system to the cloud. They have selected Azure to host the intake process of patients, and be able to allow a clinic manager, or receptionist to collect payment (credit card) from patients when a visit is completed.

The solution can be viewed as a quick deployable POC to understand how Microsoft Azure can be used to collect, store, and retrieve payment card data that meets the stringent PCI DSS requirements, and healthcare data that meets requirements for safe patient health information handling practices governed by HIPAA.

As this is a proof of concept (POC), and installs the required elements to operate a service, it is not a customer ready to go solution, and requires carful understanding of all the regulations, and laws that your data must abide by. *You will be is responsible for conducting appropriate security and compliance reviews of any solution built with this architecture POC, as requirements may vary based on the specifics of your implementation and geography.  PCI DSS requires that you work directly with an accredited Qualified Security Assessor to certify your production ready solution.*

In the solution we will have the following employee functions to explore.

Name: Edna Benson is the receptionist, and business manager. She is responsible to ensure that patient customer information is accurate, and billing is completed.

Username: EdnaB

Password:!Password111!!!

First name: Edna

Last name: Benson

User type: Member

**Permissions:** Create, read patient information, Read DOB

Edna will be able to modify patient information but will not be able to alter patient medical records

Edna can Overwrite (or replace – Credit card number, expiration, and, verification information - cvc)

Edna can replace stored social security number (SSN).

Edna cannot read SSN, or Credit Card information unmasked. Additionally, all her actions are logged.

User: Dr. Chris Aston is the clinics doctor. He is responsible for entering patient history, and treatment information. Chris can update information for patients.

Username: ChrisA

Password: !Password222!!!

First name: Chris

Last name: Aston

User type: Member

**Permissions:** Create, read patient information, Read DOB

Chris can modify patient information – including medical records, Date of birth, and can view masked SSN.

Chris has no access to credit card information

All his actions are logged

In this scenario, you will be able to test the security measures configured for Edna, and Chris to explore their rights to read, create, alter, delete records in the Contoso Health care solution.

# Solution installation

This solution utilized the following Azure services:

* Application Gateway
* Azure Active Directory
* App Service Environment
* OMS Log Analytics
* Azure Key Vault
* Network Security Groups
* Azure SQL DB
* Azure Load Balancer
* Application Insights
* Azure Security Center
* Azure Web App
* Azure Automation
* Azure Automation Runbooks
* Azure DNS
* Azure Virtual Network
* Azure Virtual Machine
* Azure Resource Group and Policies
* Azure Blob Storage
* Azure Active Directory access control (RBAC)

## Procedure to install

1. Install pre-requisites to client (installer)
2. Acquire certificates, and all configuration variables (listed in required variables list)
3. Establish a Azure subscription, and domain service pre-requirements
4. Run pre-installer script
5. Install the ARM templates (build the Azure service elements)
6. Run the post-installer script
7. Validate data in SQL database
8. Enable encryption, and encrypt key service elements (Patient SSN, DOB, Credit card #, Exp date, CVV sames)
9. Validate installation against PCI DSS client responsibilities
10. Review user roles and rights assigned to Edna, and Chris.

## Items you require prior you start

### The following information can be retrieved when you stand up a **new paid subscription on Azure**. Please note many of the features are not available in a Azure trial account.

1. [Ensure that your Local Machine has Azure PowerShell installed, and is configured with the required modules.](#_Local_machine_setup)
2. You will need your Active directory domain name, Subscription ID and create a name for your resource group also called a suffix for this solution.

|  |  |
| --- | --- |
| Parameter Name | Examples |
| $azureADDomainName | **pcidemouseroutlook.onmicrosoft.com** |
| $subscriptionID | **27017c43-3ea4-467a-afa4-7d3d3d9c3572** |
| $suffix | **contosoclinic** |

1. An Active Directory Administrator user is required to run. The local administrator will be required to be in the *.onmicrosoft.com* domain name to run this solution. You can create this user by running CreateGlobalADAdmin.ps1 script that is located in the pre-post-deployment folder of this solution. Or use the Azure portal to create an administrator account called ‘**admin’** with the following permissions.
   1. Must be a Global Admin
   2. Must be part of the Admin group of the subscription you will be using
   3. Must be a “Company Administrator” in your AD directory role
   4. These PowerShell commands help illustrate the user required for this solution

New-MsolUser -UserPrincipalName admin@pcidemouseroutlook.onmicrosoft.com -DisplayName “admin” -FirstName “admin” -LastName “contoso”

Add-MsolRoleMember -RoleName “Company Administrator” –RoleMemberEmailAddress admin@pcidemouseroutlook.onmicrosoft.com

Set-MsolUserPassword -userPrincipalName admin@pcidemouseroutlook.onmicrosoft.com -NewPassword "'CF4!!12sdfStgb'" -ForceChangePassword $false

### Custom domain, and SSL certificate

The installation will require a custom domain, and SSL certificate. It is recommended that a custom domain be purchased with [a SSL package](https://d.docs.live.net/7b2b5032e10686e1/Azure Compliance/PCI DSS quickstart/1.%09https:/docs.microsoft.com/en-us/azure/app-service-web/web-sites-purchase-ssl-web-site). Microsoft offers the ability to create a domain, and request a SSL certificate from a Microsoft partner.

Setting up a [custom domain with a DNS record](https://docs.microsoft.com/en-us/azure/app-service-web/custom-dns-web-site-buydomains-web-app) with a root domain can be configured in the [Azure Portal](https://portal.azure.com/)

The installation of the ARM template will require the domain name, such as contosoclinic.com and the .Pfx from the SSL provider that will be base64 encrypted before uploading to Azure. The following process can be used create the correct file.

1. Review the instructions on [creating a web site ssl certificate](https://docs.microsoft.com/en-us/azure/app-service-web/web-sites-configure-ssl-certificate)
2. Retrieve your private key – This file will have a name structure such as [www.contosoclinic.com\_private\_key.key](http://www.contosoclinic.com_private_key.key)
3. Retrieve your certificate – This file will have a name structure such as [www.contosoclinic.com\_ssl\_certificate.cer](http://www.contosoclinic.com_ssl_certificate.cer)
4. [Create a personal information exchange (pfx) file](https://technet.microsoft.com/en-us/library/dd261744.aspx) protect this file with a password.
5. Convert the pfx file into a string base64 text file. In powershell you can use the following commands for instance.

$fileContentBytes = get-content 'contosoclinic.com\_private\_key.pfx' -Encoding Byte

[System.Convert]::ToBase64String($fileContentBytes) | Out-File 'pfx-bytes.txt'

**Preserve your SSL 64 bit string, and password to be used when you install the ARM template.**

If you cannot use a authority to create a SSL certificate, you can consider using a self-signed certificate generated by services such as [Lets Encrypt](https://letsencrypt.org/). However for PCI compliance self-signed certificates will not pass the requirements, or audit for PCI DSS.

## Local machine setup requirements

It’s recommended that you run all scripts (pre-and post deployment scripts) as local admin or remotely signed credentials to ensure that local permissions to not restrict the installer running correctly.

Client Software requirements

The following client software apps are needed throughout the installation of the script.

1. [Powershell version](https://msdn.microsoft.com/en-us/powershell/scripting/setup/installing-windows-powershell) v5.x or above psversionTable.psversino
2. The following Powershell Modules are required. These must be installed with local Administrative permissions.
   * Open Powershell in Administrator Mode
   * Run the following commands, and accept (or select Yes to user commands)

Set-ExecutionPolicy RemoteSigned;

Install-Module AzureRM;

Install-Module AzureAD -AllowClobber;

Install-Module -Name Enable-AzureRMDiagnostics -Force;

Install-Module -Name AzureDiagnosticsAndLogAnalytics -Force;

Import-Module Sqlps -DisableNameChecking;

|  |
| --- |
|  |

If any of the above commands fail, please refer to the following reference links

|  |  |
| --- | --- |
| **Module** | **Reference Link** |
| AzureRM | <https://docs.microsoft.com/en-us/powershell/azureps-cmdlets-docs/>  **Validate by testing**  $cred = Get-Credential  Login-AzureRmAccount -Credential $cred |
| Azure AD | <https://technet.microsoft.com/en-us/library/dn975125.aspx>  **Validate by testing**  $Credential = Get-Credential  Connect-AzureAD -Credential $Credential |
| Enable AzureRM Diagnostics | <https://www.powershellgallery.com/packages/Enable-AzureRMDiagnostics/1.3/DisplayScript> |
| Azure Diagnostics And LogAnalytics | <https://www.powershellgallery.com/packages/AzureDiagnosticsAndLogAnalytics/0.1> |
| SQL Server Powershell | <https://msdn.microsoft.com/en-us/library/hh231683.aspx?f=255&MSPPError=-2147217396#Installing SQL Server PowerShell Support>  **Validate by testing**  Get-Module -ListAvailable -Name Sqlps; |

## Pre-ARM template deployment

The script called pre-deployment.ps1 provides the setup and configuration of user, and other framework elements. The following steps are required to run the script. Note that that the scripts must complete without errors. Before the ARM template can be deployed successfully.

Using the [Azure portal](https://portal.azure.com/) with an account that is a member of the [Subscription Admins role and co-administrator of the subscription](https://docs.microsoft.com/en-us/azure/active-directory/active-directory-assign-admin-roles" \l "global-administrator).

* 1. Set up a your resource group,
     + In a your PowerShell IDE execute the following example, replacing the name with the same name outlined in section 3.2.1

New-AzureRmResourceGroup -Name Contosoclinic -Location "East US"

* 1. in this example.
  2. *Do not proceed without verifying you Run as service was successful deployed by running the runbook examples in the previous step called ‘azureautomationtutorialscript’ to verify run as service was deployed correctly. Creation of Service Principal has a* ***propensity to fail randomly****. A basic verification whether it was successfully created is* ***mandatory****.*

At this point you will have the following items created:

Name of Automation eg - **Contosoclinic-Automation**

Resource group you added eg – **Contosoclinic**

In your PowerShell IDE execute the following commands (as done in the client side verification):

1. Log into your azure subscription, and retrieve your subscription information

$cred = Get-Credential

Login-AzureRmAccount -Credential $cred

$Credential = Get-Credential

Connect-AzureAD -Credential $Credential

Get-AzureRmSubscription

Record the following information. Refer to the highlighted example.

Environment : AzureCloud

Account : **admin@demouseroutlook.onmicrosoft.com**

TenantId : 21d644f0-12av-4043-b0bb-f5acfde12256

SubscriptionId : **27017c43-3ea4-467a-afa4-7d3d3d9D33232**

SubscriptionName : SubscriptionName

1. Test your connection to your active directory service.

Get-MsolDomain

Record the name of your msol domain Refer to the highlighted example

Name Status Authentication

---- ------ --------------

**pcidemouseroutlook.onmicrosoft.com** Verified Managed



1. Run the predeployment.ps1 script

* If you have not already, download a copy of installation scripts from <https://github.com/AvyanConsultingCorp/pci-paas-webapp-ase-sqldb-appgateway-keyvault-oms> (If you downloaded a .zip file, expand the content of the compressed file to a local directory)
* In your PowerShell IDE change directory to the location of your local repository
* Run the ‘PreDeployment.ps1’ script
  + .\pre-post-deployment\PreDeployment.ps1
  + Select ‘**Run Once**’ to the script warning

|  |  |
| --- | --- |
| Parameter Name | Example from previous steps |
| $azureADDomainName | **demouseroutlook.onmicrosoft.com** |
| $subscriptionID | **27017c43-3ea4-467a-afa4-7d3d3d9D33232** |
| $suffix | **contosoclinic** |



1. Record the information provided by the script. You will need this information to proceed.

Prompt Start copy all the values from below here.

Name of Automation eg - **Contosoclinic-Automation**

Parameters to be used in the registration / configuration.

Azure AD Application Client ID: **952b0b1e-2582-4058-a0a0-0abc42107d70**

Azure AD Application Client Secret: **Password@123**

Azure AD Application Object ID: **e3aa33bb-1cae-4afd-a8ba-9124b2a1838a**

SQL AD Admin Name: **sqladmin@pcidemouseroutlook.onmicrosoft.com**

SQL AD Admin Password:(If user already exists then we have to get password manually) !Password333!!!

TODO - Update permissions for the AD Application ''.Please follow the deployment guide for the specific permissions

-Prompt The following additional users have been created in domain. These users will be used for trying out various scenarios

receptionist\_EdnaB@pcidemouseroutlook.onmicrosoft.com user is created. password is **!Password111!!!**

doctor\_ChrisA@pcidemouseroutlook.onmicrosoft.com user is created. password is **!Password222!!!**

-Prompt End copy all the values from above here.

The script completed execution. Press any key to exit:

## Configuring the Active Directory application

Azure active directory application permissions require to be configured manually, as there are no PowerShell scripts available at this time to manage the settings reliably.

* 1. In [Azure Portal](https://portal.azure.com/) select more services (or press ‘B’ on the right rail) and filter for ‘*Azure Active Directory’*
  2. Select ‘**App Registrations**’
  3. Select your application you created. It will be listed with your selected $suffix with the name “Azure PCI PAAS Sample”
  4. Select “**Required Permissions**”
  5. Select ‘**+Add’**
  6. Select **‘Select an API’**
  7. In this list we will modify, ‘**Windows Azure Active Directory’**, **Microsoft Graph’**, ‘**Windows Azure Service Management API**’, and \*\*‘**Azure Key Vault’.**

**NOTE**: If ‘**Azure Key Vault’** is notIf you don’t see this permission then please follow the [guidance from this blog](https://blogs.technet.microsoft.com/kv/2016/09/17/accessing-key-vault-from-a-native-application/)

The following step through will assist you in configuring each of the permission sets.

1. Select an API - **Windows Azure Active Directory -** Select
   1. Select the following 2 application Permissions

**Read Directory Data**

**Read and Write directory data**

* 1. Select the following 3 Delegated Permissions

**Read all groups**

**Read directory data**

**Access the directory as the signed-in user**

1. Select
2. Done
3. Select an API - **Microsoft Graph -** Select
   1. Select the following 6 application Permissions

**Read files in all site collections**

**Read all groups**

**Read Directory Data**

**Read and Write directory data**

**Read all users’ full profiles**

**Read all identity risk event information**

* 1. Select the following 7 Delegated Permissions

**Sign in and read user profiles**

**Read all users’ basic profiles**

**Read all users’ full profiles**

**Read all groups**

**Read directory data**

**Read and write directory data**

**Access the directory as the signed-in user**

1. Select
2. Done
3. Select an API - **Azure Key Vault -** Select
   1. Select 0 application Permissions
   2. Select the following 1 Delegated Permissions

**Have full access to the Azure Key Vault service**

1. Select
2. Done
3. Select an API - **Windows Azure Service Management API-** Select
   1. Select 0 application Permissions
   2. Select the following 1 Delegated Permissions

**Access Azure Service Management as organization user**

1. Select
2. Done

If the configurations are successful, you will see the following table of permissions

| **API** | **APPLICATION PERMISSIONS** | **DELEGATED PERMISSIONS** |
| --- | --- | --- |

|  |  |  |
| --- | --- | --- |
| Windows Azure Active Directory | 2 | 3 |
| Microsoft Graph | 6 | 7 |
| Azure Key Vault | 0 | 1 |
| Windows Azure Service Management API | 0 | 1 |

## Deploying Azure Resource Template (ARM)

Deploying the ARM template will require the following deployment information you should collect before selecting ‘Deploy to Azure’.

Sample deployment information collected to start deployment of ARM:

**Basics**

Subscription: Contoso Subscription

Resource group: Select Use Existing. In our example we use “Contosoclinic”

Location: Greyed out

**Settings**

\_artifactsLocation: <https://raw.githubusercontent.com/AvyanConsultingCorp/pci-paas-webapp-ase-sqldb-appgateway-keyvault-oms/master>

\_artifactsLocationSasToken: **NULL**

certData: **{The Contoso base 64 SSL string}**

certPassword**:{Password for SSL cert}**

bastionHostAdministratorUserName: **bastionadmin**

bastionHostAdministratorPassword:**Create a password for this user at this time.**

SqlAdministratorLoginUserName: **sqladmin**

sqlAdministratorLoginPassword: **!Password333!!!** This password was provided after pre-installation script completed.

sqlThreatDetectionAlertEmailAddress: [admin@contosoclinic.com](mailto:admin@contosoclinic.com). You can enter a designated recipient for alerts.

automationAccountName: **Contosoclinic-Automation**

customHostName: **azurepcisamples.com**

azureAdApplicationClientId: **952b0b1e-2582-4058-a0a0-0abc42107d70**

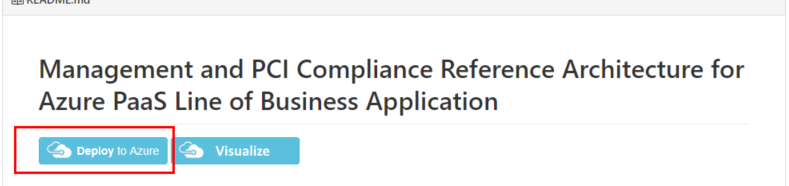
azureAdApplicationClientSecret: **Password@123**

azureAdApplicationObjectId: **e3aa33bb-1cae-4afd-a8ba-9124b2a1838a**

sqlAdAdminUserName: **sqladmin@pcidemouseroutlook.onmicrosoft.com**

sqlAdAdminUserPassword: **CF4!!12sdfStgb**

Once you have collected all the basic, and setting information, you can select ‘Deploy to Azure’ as illustrated.

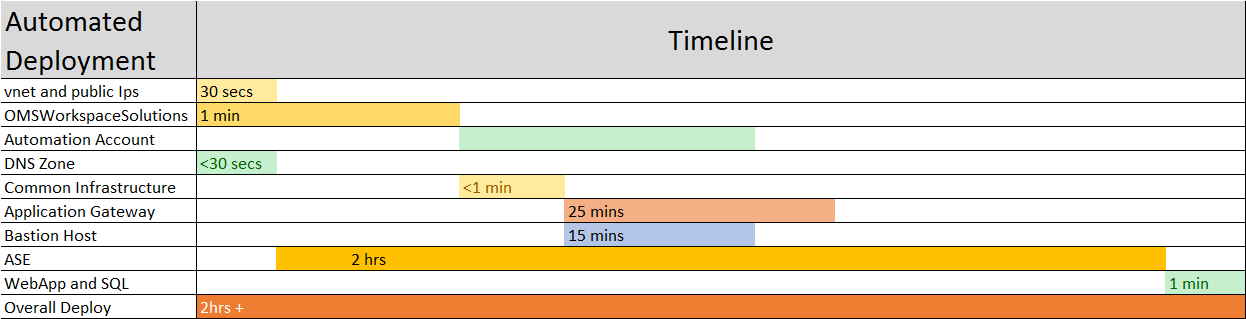


Provide all the values of the form. Select **‘I agree to the Terms and conditions stated above’**

Select ‘Purchase’

## Deployment Timeline

This following graphic displays the estimated time to deploy the components. Overall deployment time is approximately 2.5 hours from the time you select ‘purchase’



# Post Deployment Steps

## Update DNS setting with Application Gateway IP

In the Contoso example the customer’s DNS settings will require the Application Gateway IP address be updated as a DNS record on the hosting site.

1. Collect the application gateway IP address

**Get-AzureRmPublicIpAddress | where {$\_.Name -eq "publicIp-AppGateway"} | select IpAddress**

This PowerShell command will return the IP address. For example.

IpAddress

---------

52.168.26.162

1. Log into your DNS hosting provider, and update you’re A/AAAA and CNAME records to the application Gateway IP
2. Verify you can connect with your site by browsing to you domain, for example [www.contosoclinic.com](http://www.contosoclinic.com)
   1. Note the site will have limited services until the post deployment script is executed.

## Post Deployment Scripts

The post deployment script is designed to run after the successful ARM deployment. The script will set up securities for social security number (SSN) samples, and credit card or payment card information (PCI) protection.

Prior to starting your post deployment, you will need the following information from your installation.

1. Your Azure ‘**admin’** credentials you can use **Add-AzureAccount | select ID**
   1. EG - admin@pcidemouseroutlook.onmicrosoft.com
2. Resource group name - $ResourceGroupName - **Get-AzureRMResourceGroup | select ResourceGroupName** 
   1. EG - Contosoclinic
3. Your SQL server name (SQLServerName), Your AES outbound Address, and Client IP address can be retried in the [Azure Portal](https://portal.azure.com/)
   1. TO retrieve the SQL server name, you will need to login to your Azure Portal.
      1. Select SQL Databases
      2. Select Your database, for our example it will list ‘ContosoClinicDb’
      3. Server name will be listed in the ‘Server name’ Field. In our example:
         1. Server name fully qualified: sqlserver-dzwhejjrwbwdy.database.windows.net
         2. Server name: sqlserver-dzwhejjrwbwdy
   2. Retrieve your Client IP address
      1. Select Overview, and select ‘Set server firewall’ in the banner
      2. Your client IP address will be displayed in the Firewall Settings. In our example:
         * 1. Client IP address is 167.220.1.231
      3. If you are using NAT, or a firewall it’s recommended you also test your IP address with:

Invoke-RestMethod http://ipinfo.io/json | Select-Object -exp ip

ipconfig

* + 1. NOTE- While in this configuration it’s advisable to add your client IP to your firewall setting for the SQL server.
       1. In Rule name add – Rule name, Start IP, and End IP
          1. In our example we create the rule – Client IP, 167.220.1.231, 167.220.1.231
  1. Retrieve your AES outbound IP Address
     1. Select

1. Your SQL username, and password from Azure ARM deployment
   1. In our Example
   2. sqlAdAdminUserName: **sqladmin@pcidemouseroutlook.onmicrosoft.com**
   3. sqlAdAdminUserPassword: **CF4!!12sdfStgb**

You will need to change the permission to the script before you run it. You can set the permissions by running this command.

**Set-ExecutionPolicy Unrestricted**

# 2) Should have successfully deployed pre-Deployment script and Azure ARM deployment

#

# The script does the following things

# 1) Downloads and copies the SQL bacpac file to a new Azure storage account

# 2) Updates SQL DB firewall to allow you (your clientIp) access to manage SQL DB AND Allowing the WebApp deployed on ASE (the ASE outbound virtual IP)

# 3) Data Mask few DB columns (ensuring that only the SQL Admins be able to see the detailed info in the Database) everyone else sees them as masked .. e.g. SSN will show up as XXX-XX-4digitnumber

# 4) Enable Always Encrypt for a few columns (e.g. Credit card)

# 5) Makes an AD User to the the SQL AD Admin [refer command Set-AzureRmSqlServerActiveDirectoryAdministrator]

# 6) Ensures Diagnotics logs are sent to OMS Workspace (script assumes that there's only one WS in the resourcegroup created by the ARM template)

## Run Post Deployment PowerShell Script

--set permissions for script

Set-ExecutionPolicy Unrestricted

Post Deployment PowerShell script used for following configuration

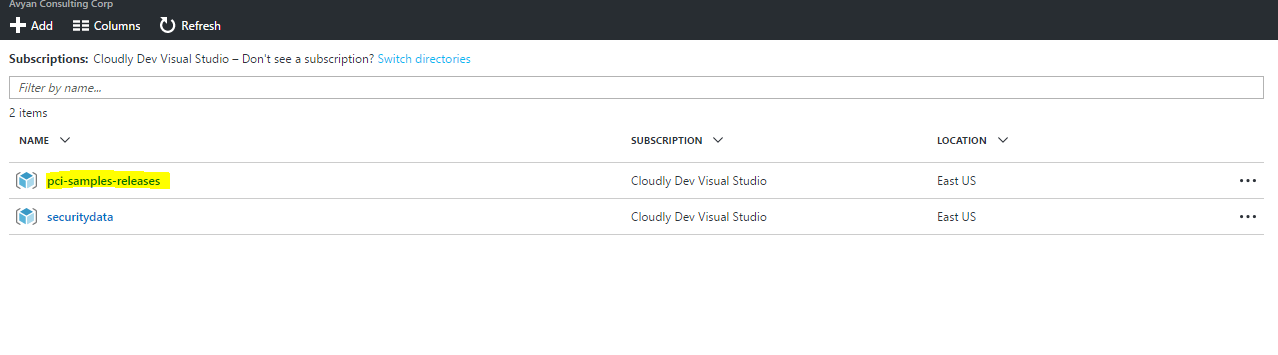
* Set Firewall rules for ASE Outbound and Client IP Address(To run scripts)
* Restore bacpac file into ContosoClinicDB DB
* Configure Dynamic Data Masking in Patients table
* Encrypt Columns using Key vault
* Set AD Authentication Admin
* Enable OMS Diagnostics

**Process to run Post Deployment Script**

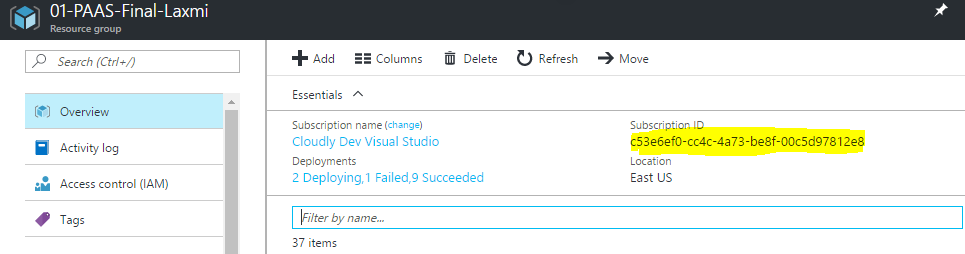
* + - Open Post Deployment PowerShell script (Path: ~/ pre-post-deployment/ PostDeployment.ps1) and Right click and select Run with PowerShell
    - Provide below mandatory values while run script. It will ask Subscription credentials to execute complete script

|  |  |  |
| --- | --- | --- |
| Parameter Name | How to get Value? | Comments |
| $ResourceGroupName | * Login to <https://portal.azure.com> * Open Resource groups and copy resource group created using ARM deployment | Screenshots attached below. Provide Resource Group Name Created through ARM template |
| $SQLServerName | * Login to <https://portal.azure.com> * Copy Sql Server name under resource group created using ARM deployment | Provide Sql Server name (not required full name) Created through ARM template |
| $ClientIPAddress | * Open windows cmd prompt and type ipconfig | Your system IP address |
| $ASEOutboundAddress | * Login to <https://portal.azure.com> * Open ASE under resource group created using ARM deployment * Open Properties and copy Outbound IP address |  |
| $SQLADAdministrator | * Provide SQL AD Administrator name passed while creating ARM template | This value passed as input parameter in ARM Deployment |
| $sqlAdministratorLoginPassword | * Provide SQL Password passed while creating ARM template | This value passed as input parameter in ARM Deployment |
| $subscriptionId | * Login to <https://portal.azure.com> * Open Resource groups and copy subscription id of resource group created using ARM deployment | Screenshots attached below |
| $KeyVaultName | * Login to <https://portal.azure.com> * Copy Key Vault name under resource group created using ARM deployment | Provide Key Vault Name Created through ARM template |
| $azureAdApplicationClientId | * Provide Azure AD Application Id created through Pre deployment script. Same value passed in ARM Deployment as a parameter |  |
| $azureAdApplicationClientSecret | * Provide Azure AD Application Client Secret created through Pre deployment script. Same value passed in ARM Deployment as a parameter |  |

(Screenshot for to Get Resource Name)



(Screenshot for to Get Subscription Id)

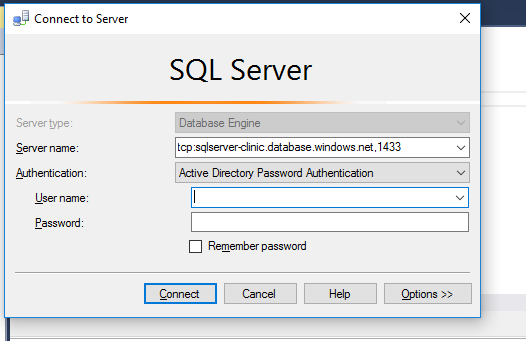


## Sample Values

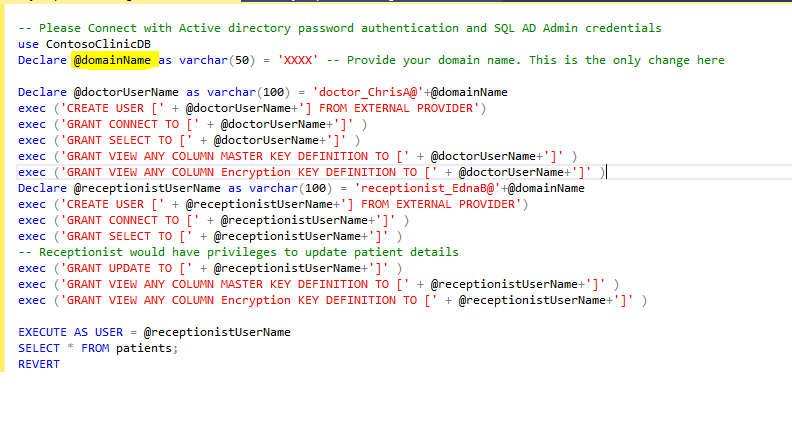


## Run Post Deployment SQL Script

### Open SQL Server Management Studio using Active directory password authentication and SQL AD Admin credentials

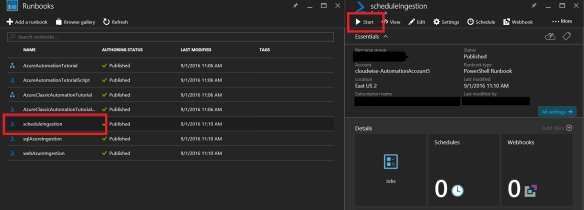


* Open “PostDeploymentSQL.sql” script under the .\pre-post-deployment folder
  + Replace XXXX with your AD domain name.
  + Run the script.



## Schedule Runbooks

* + - Click open the schedule Ingestion runbook and click start to run the runbook. This step will kick start the data ingestion to the OMS workspace specified.

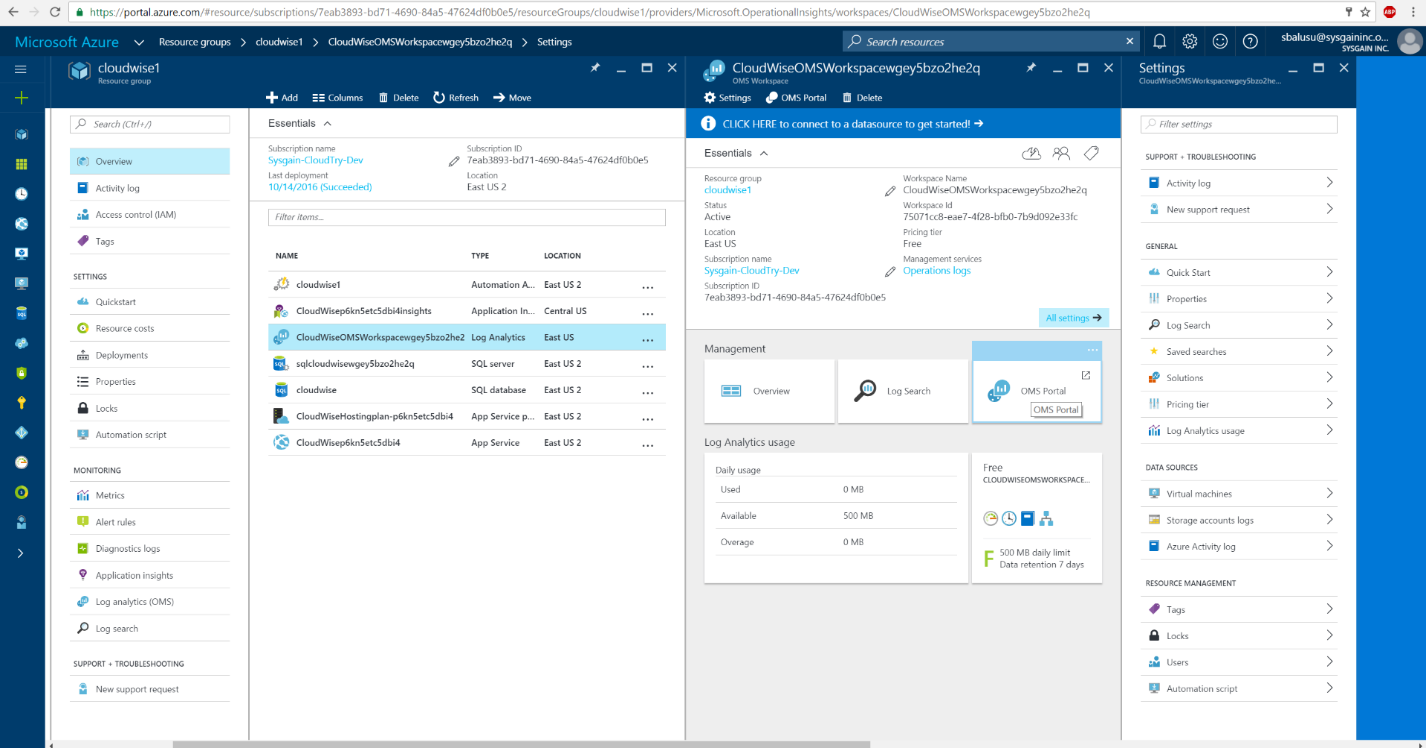


## Install OMS Dashboards Views.

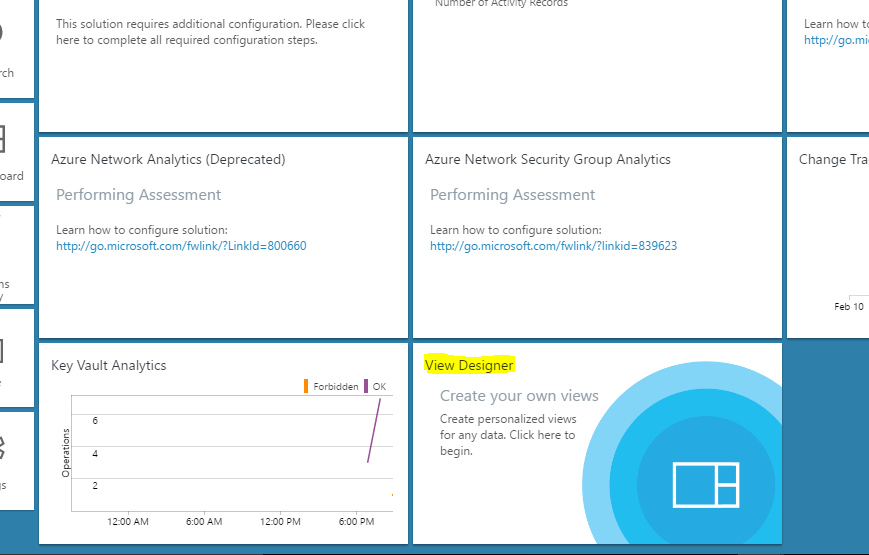
This is currently a manual process as ARM Json deploys do not yet support creation of OMS views.

(By a Service Admin/Contributor role)

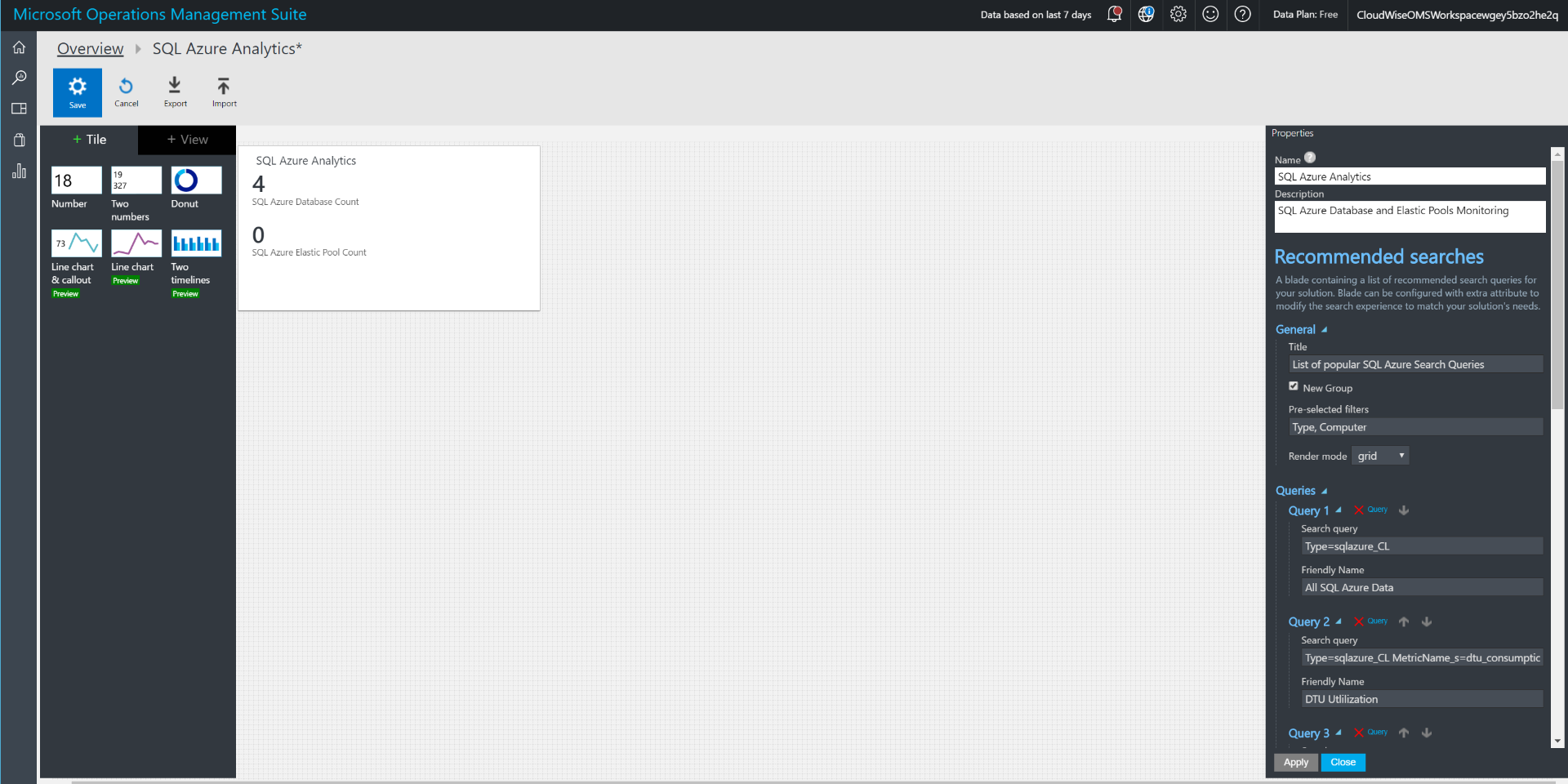
1. Open the resource group and click on the OMS Portal link. This will open the OMS portal in a different window

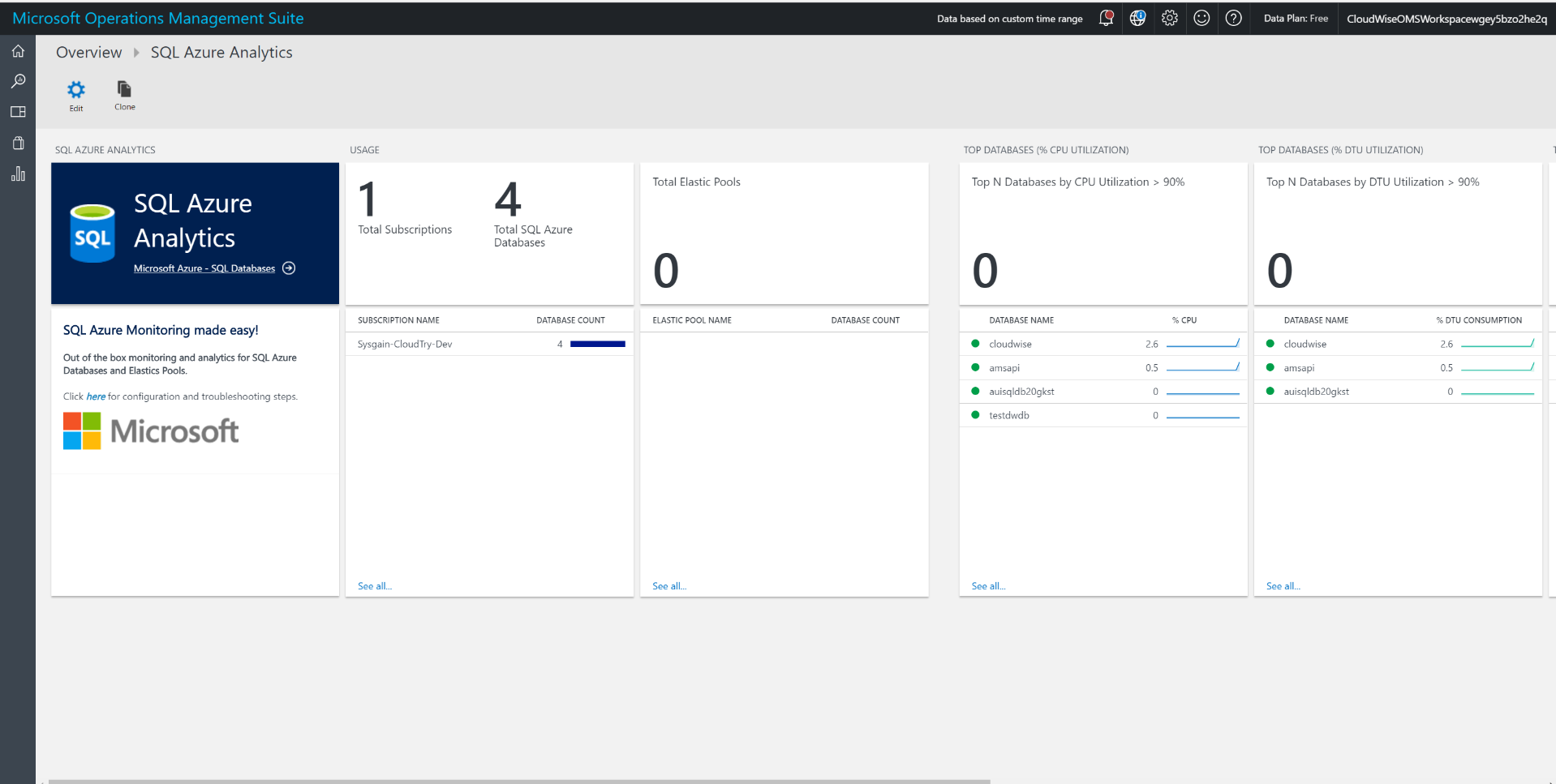


1. Click on the View Designer. You may have to scroll down

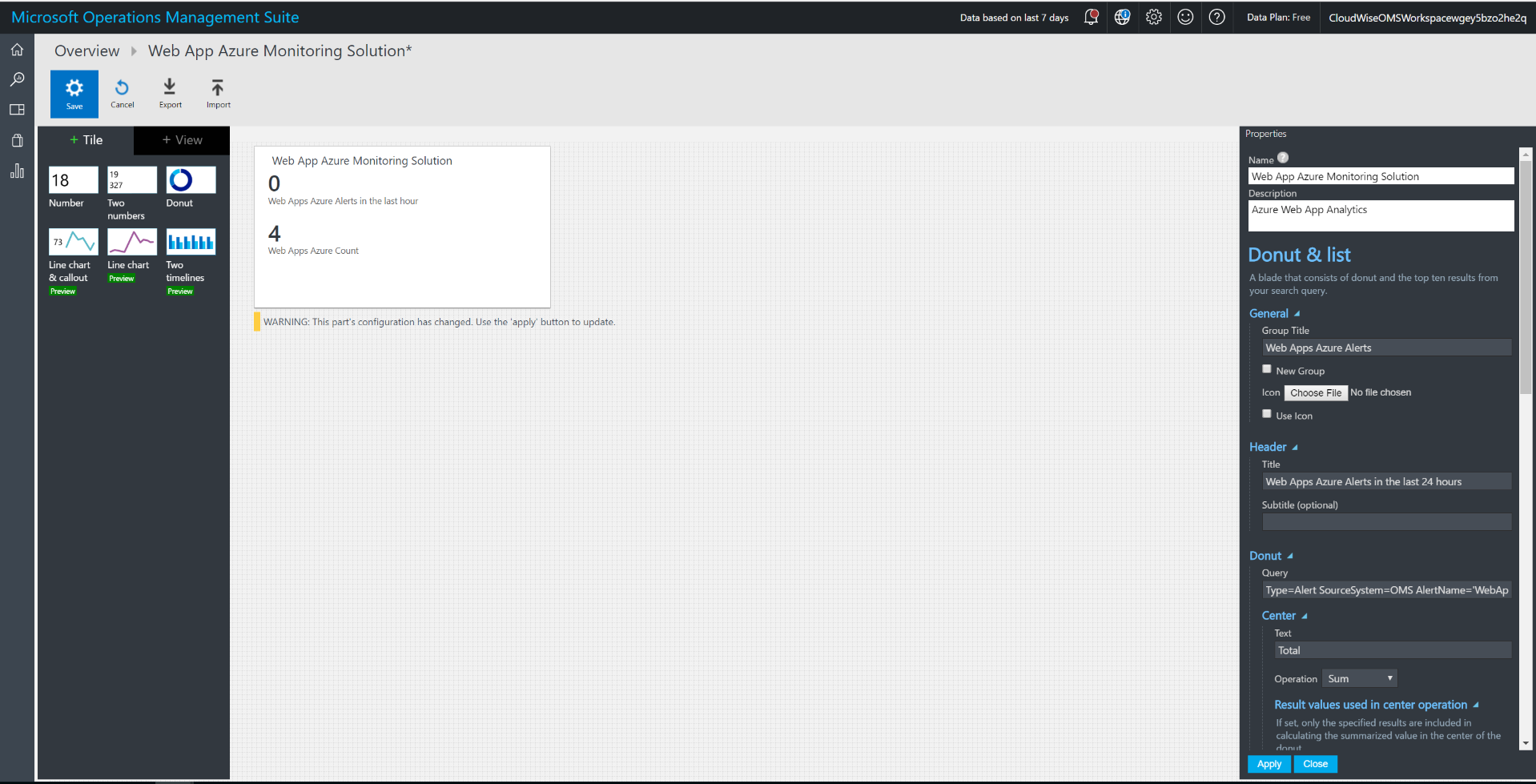


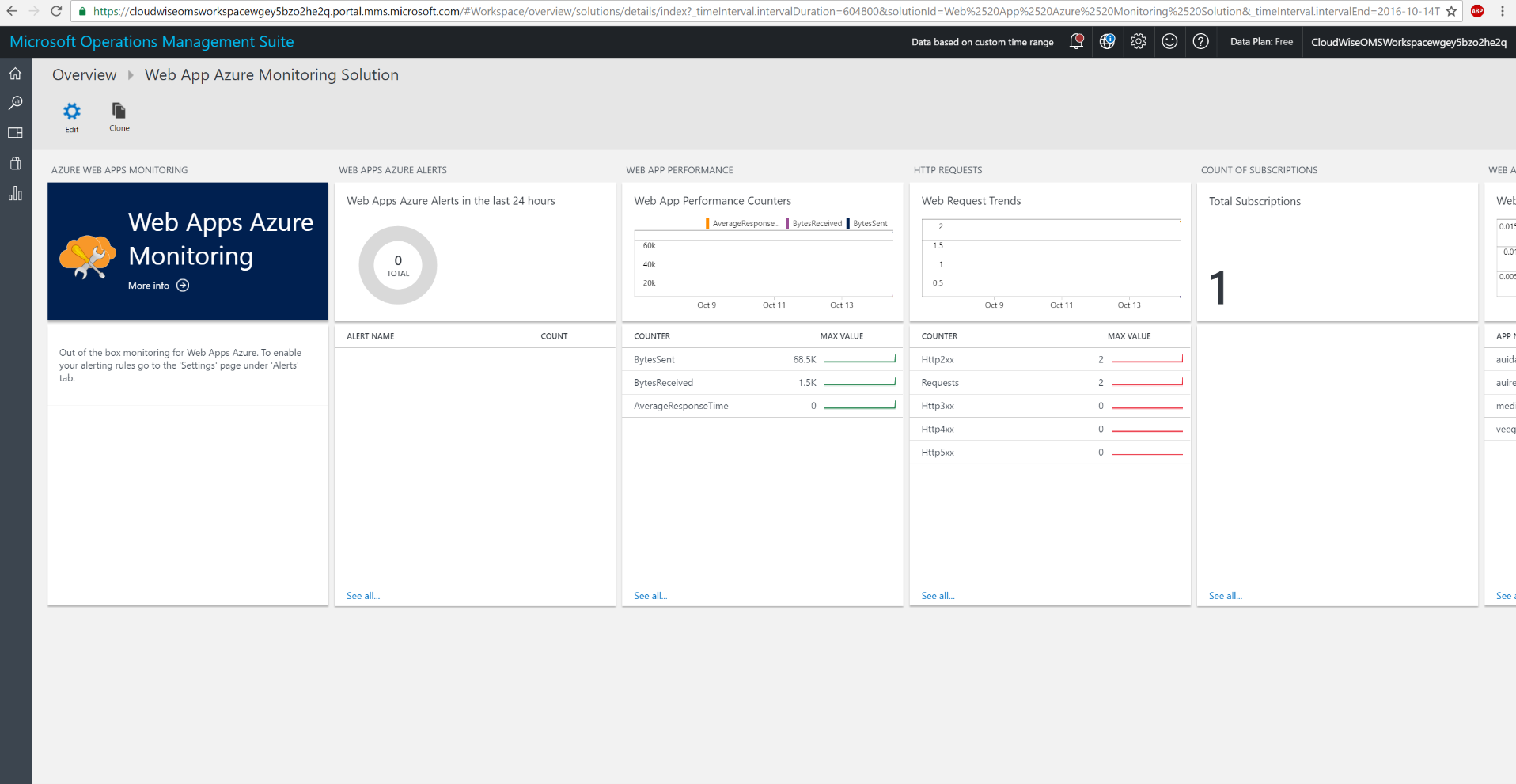
1. Import SQL DB view by clicking on the Import button and browsing to the file (.\omsDashboards\OMSSQLDBAzureMonitoringSolution.omsview)





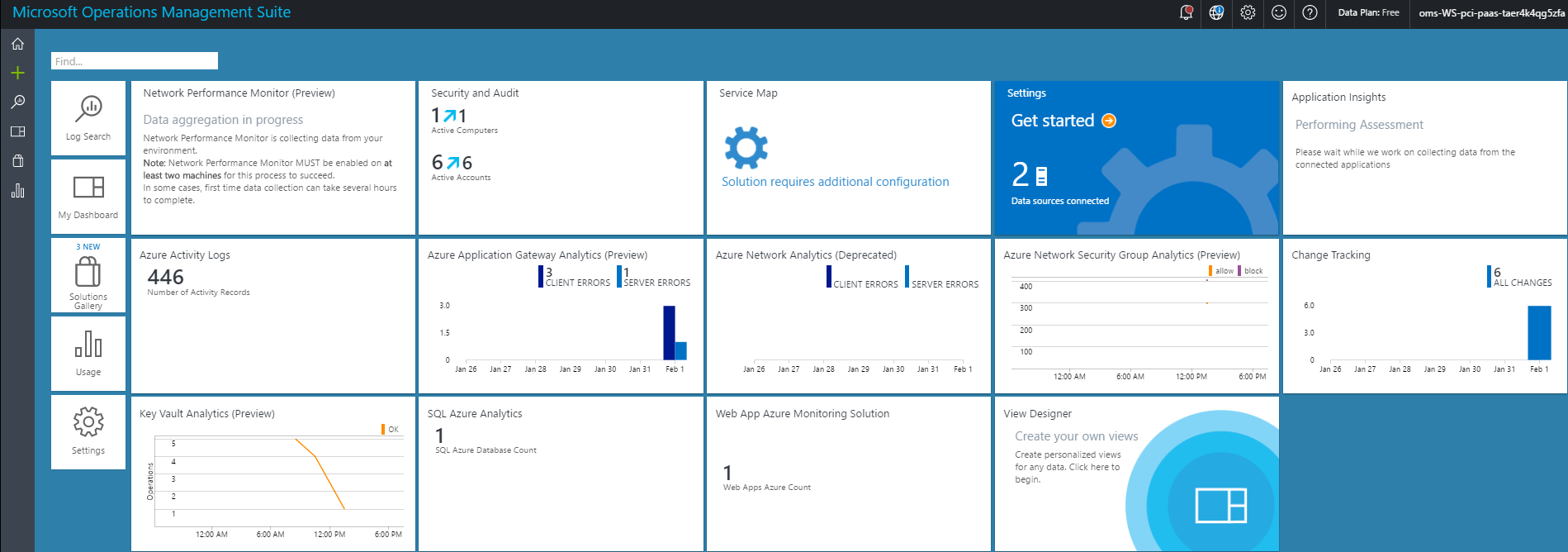
1. Repeat the same step for the Web App Monitoring dashboard. Import view omsDashboards\ OMSWebAppAzureMonitoringSolution.omsview





## Check and verify OMS solutions are collecting data

When you login to the OMS workspace installed in the resource group, you should navigate to the portal and you’ll see something like this. Data is being collected and Solution is being monitored and operationally ready for you to test out.



## Enable Azure Security Center

Follow instructions from here to enable data collections from Azure Security Center <https://docs.microsoft.com/en-us/azure/security-center/security-center-get-started>

To Do

* **Step 1**: Enable Data collection at the Subscription level.
* **Step 2**: Ensure all rules/policies are enabled (they are enabled by default)
* **Step 3**: Run recommendations
* **Step 4**: View the recommendation results. You will see that the solutions passes most rules
  + WAF enabled for public IP address
  + VM disk and data disk encrypted
  + All Azure Storages accounts are encrypted
  + SQL TDE, Auditing enabled
  + And many more…

☛ **Note:**

1. Currently OMS Monitoring agent is automatically installed along with the Bastion Host deployment. We have (on purpose) not installed the security center VM agent, as the ASC team is moving towards using OMS agent for their purposes. Once they completely move to OMS agent, this solution will automatically work well with ASC.

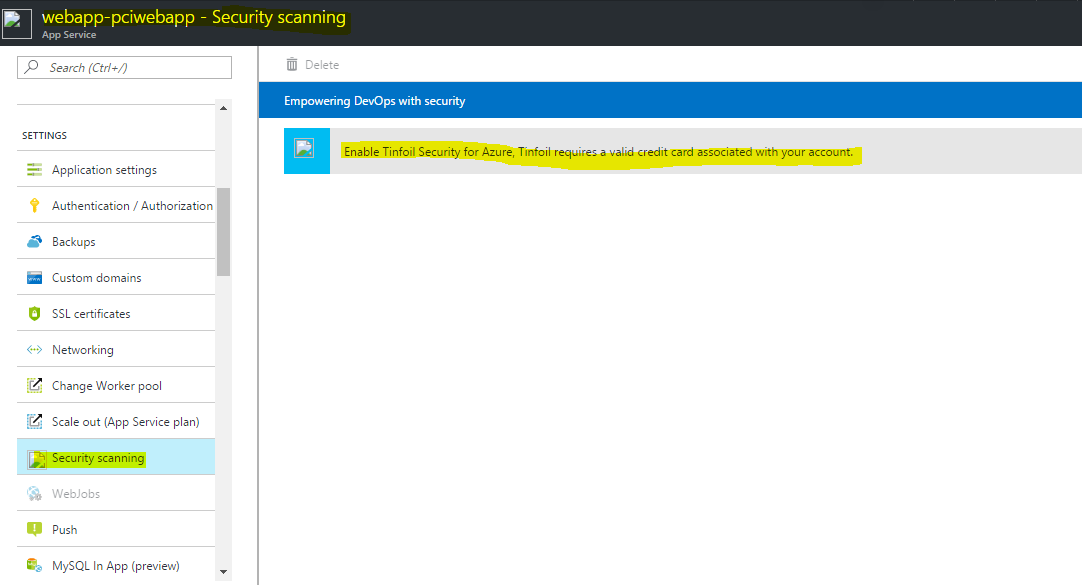
## Enable Tinfoil Security for web app vulnerability assessment

Refer the following documents to understand what is Tinfoil security, it’s an optional component to take in to production

Note: This step involves credit card information, that’s the reason we have not automated it.

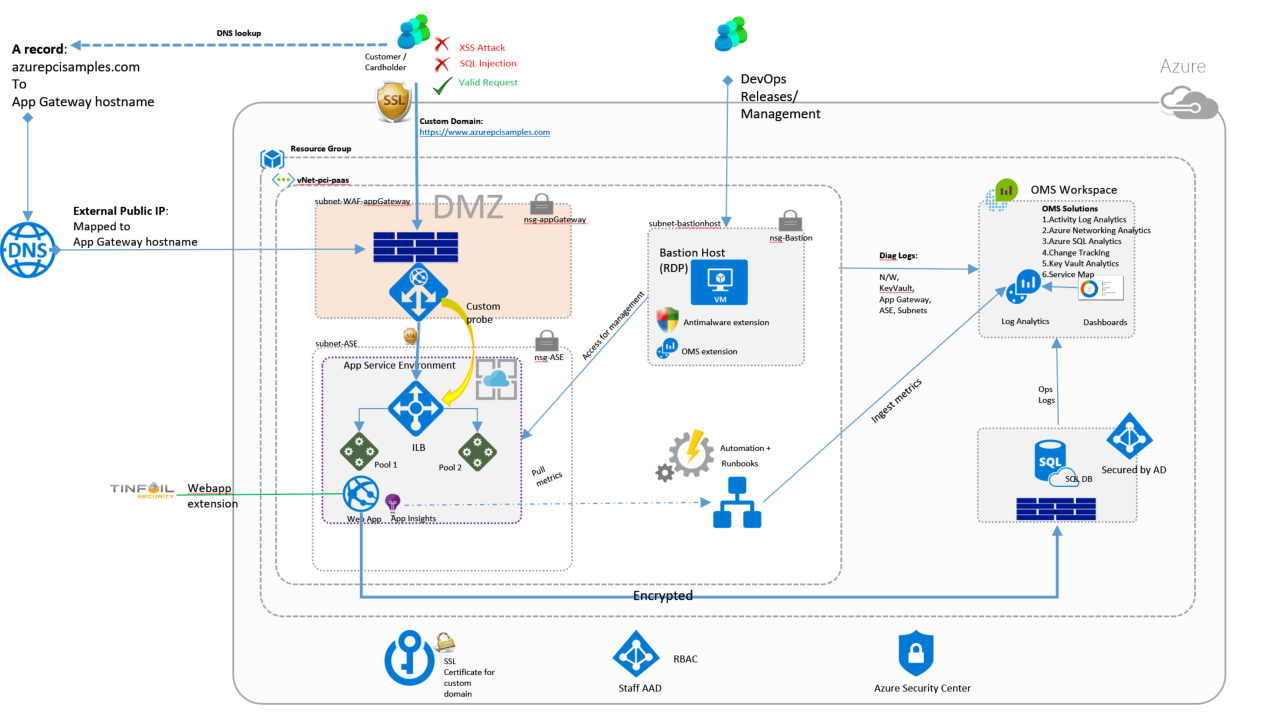
<https://azure.microsoft.com/en-us/blog/web-vulnerability-scanning-for-azure-app-service-powered-by-tinfoil-security/>

<https://www.tinfoilsecurity.com/azure>



# Deployment Architecture

The deployment Reference Architecture consists of the following elements



Following protections security features have been implemented to secure the solution, and payment card data.