# Deployment instructions

1) Connect to Visual Studio Online and load the "Deploying ASE & ASF" solution in Visual Studio. Make sure to get latest code from VSO.

2) Deploy foundational network – this step will deploy 2 peered VNets: one for security devices and the other for LoB resources. Network Security Group rules are also created and attached to appropriate subnets to lock down all unnecessary ports. Deployment duration: < 1 minute.
   a) In Visual Studio Solution Explorer, right-click on **LoBNetworkResourceGroup** project. Hover over **Deploy** and select **New Deployment…**
   b) Select the subscription where you want to deploy to.
   c) Click on the **Resource group** dropdown and select **<Create New…>**. A new window will open:
      i) Enter the resource group name (NetworkRG).
      ii) Choose a location where all the resources belonging to this resource group will be deployed to (West US).
      iii) Click **Create**.
   d) Click on the **Deployment template:** dropdown and choose **network.json**.
   e) Click on the **Template parameters file:** dropdown and choose **network.parameters.json**.
   f) Click on **Edit Parameters** button and fill out the required parameters, which are self-explanatory. You can provide either existing or new resources. ARM deployment will create the resource if it's not already there and reuse if the resource already exists with the same name.
   g) Click **Deploy**
   h) View Visual Studio Output window to make sure deployment was successful. Validate in the portal also.

3) Deploy ADDS/DNS – this an optional step if the network has access to an existing DNS server. Otherwise, this step is required since ASE and ASF will be dependent on DNS for deployment and publishing. The template used in this step will deploy 2 VMs into an existing subnet and then configure those VMs to run ADDS and DNS. Deployment duration: ~ 45 minutes.
   a) In Visual Studio Solution Explorer, right-click on **LoBNetworkResourceGroup** project. Hover over **Deploy** and select the previous deployment name.
   b) Make sure the correct subscription and resource groups are selected from the dropdowns.
   c) Click on the **Deployment template:** dropdown and choose **activedirectory-optional.json**.
   d) Click on the **Template parameters file:** dropdown and choose **activedirectory-optional.parameters.json**.
   e) Click on **Edit Parameters** button and fill out the required parameters. You can provide either existing or new resources. ARM deployment will create the resource if it's not already there and reuse if the resource already exists with the same name. Here are things to note about some of the parameters:
      i) newStorageAccountName – name of new or existing storage account. VHDs for the VMs will be stored here.
      ii) storageAccountType – if new storage account, then specify the type.

iii) virtualNetworkName, virtualNetworkAddressRange – reuse the same values for the security VNet, as specified in the network.parameters.json file because we're deploying to an existing network.

iv) adSubnetName, adSubnetAddressRange – reuse the same values for the "security identity" subnet, as specified in the network.parameters.json file because we're deploying to an existing subnet.

v) adPDCNicIPAddress, adBDCNicIPAddress – provide static internal IP addresses for primary and backup domain controllers.

vi) domainName – this is the local domain name used by domain controllers. It should look something like: mdb.schoolsfirstfcu.org, but change it to whatever fits your needs.

vii) **Keep the default value for assetLocation parameter unless you know what you're doing.**

viii) The rest of the parameters are self-explanatory.

ix) Click **Save** when done filling out all the parameters.

f) Click **Deploy**

g) View Visual Studio Output window to make sure deployment has completed and was successful.

h) ***This template wipes out the rest of the existing subnets when it deploys ADDS/DNS. A temporary workaround is to redeploy the network.json template again.***

4) Manually update DNS servers for VNets. Deployment duration: < 2 minutes.

a) Go to portal.azure.com

b) Navigate to the resource group containing the foundational network resources.

c) Click on the VNet where ASE and ASF will be deployed (SFWUS-NP-VNET-MDB).

d) Click on the **DNS Servers** section.

e) Click on **Custom** radio button and add the IP addresses for the primary and backup domain controllers deployed in step 2 (172.25.255.36, 172.25.255.37).

f) Click **Save**.

g) Repeat steps 3b-3f for the VNet containing the security devices (SFWUS-NP-VNET-SEC).

5) Deploy jump box – this is an optional step if you want to deploy a jump box into the management subnet. This template will create a new Windows VM with an external IP address with RDP port opened. Deployment duration: ~5 minutes.

a) In Visual Studio Solution Explorer, right-click on **LoBNetworkResourceGroup** project. Hover over **Deploy** and select the previous deployment name.

b) Make sure the correct subscription and resource groups are selected from the dropdowns.

c) Choose **jumpbox-optional.json** for the Deployment template.

d) Choose **jumpbox-optional.parameters.json** for the Template parameters file.

e) Click on **Edit Parameters** button and fill out the required parameters. You can provide either existing or new resources. ARM deployment will create the resource if it's not already there and reuse if the resource already exists with the same name. Here are things to note about some of the parameters:

i) storageAccountName, storageAccountType – storage account to store the VHD. Provide the same storage account used earlier in step 3.

ii) existingVnetResourceId – ID of the VNet where you want to deploy the jump box to. You can get this ID from the portal by navigating to the **LoB VNet** then click on **Properties** tab.

Copy the **RESOURCE ID** at the top, which should be in this format: /subscriptions/your-subscription-id-here/resourceGroups/your-resource-group-name-here/providers/Microsoft.Network/virtualNetworks/your-vnet-name-here.

iii) subnetName – name of the subnet where the jump box will reside. It should be the management subnet.

iv) imagePublisher, imageOffer, imageSKU – these values specifies the Windows Server image with Visual Studio 2015 Community edition & Azure SDK. You can change this to whatever fits your needs.

v) All other parameters are self-explanatory.

vi) Click **Save** when done editing the parameters.

f) Click **Deploy**

g) View Visual Studio Output window to make sure deployment was successful.

h) Go to Azure portal to verify VM was successfully provisioned and try to RDP into VM.

i) Optional step – install Remote Server Administration Tools to manage ADDS/DNS.

i) Inside VM, bring up Server Manager.

ii) At the top, click on **Manage**, then choose **Add Roles and Features**

iii) Click on **Next** until you get to the **Features** page

iv) Select the following **Remote Server Administration Tools > Role Administration Tools**

(1) AD DS and AD LDS Tools

(2) DNS Server Tools

v) Click **Next** then **Install**

6) Deploy ASE – This step will deploy Azure App Service Environment into the presentation subnet of the LoB VNet. Along with the ASE, this template also creates an App Service Plan and a Web App. Deployment duration: up to 2 hours.

a) In Visual Studio Solution Explorer, right-click on **LoBResourceGroup** project. Hover over **Deploy** and select **New Deployment…**

b) Select the subscription where you deployed the foundational network to.

c) Click on the **Resource group** dropdown and select **<Create New…>**. A new window will open:

i) Enter the resource group name (LoBAppRG). This is the resource group which will contain ASE, web & mobile apps, and Azure Service Fabric (ASF).

ii) Choose the resource group location. Make sure it's in the same location as the network resource group.

iii) Click **Create**.

d) Click on the **Deployment template:** dropdown and choose **ase-private.json**.

e) Click on the **Template parameters file:** dropdown and choose **ase-private.parameters.json**.

f) Click on **Edit Parameters** button and fill out the required parameters, which are self-explanatory. You can provide either existing or new resources. ARM deployment will create the resource if it's not already there and reuse if the resource already exists with the same name.

i) aseName – name of your ASE, which is a unique name that hasn't already been taken at *.p.azurewebsites.net. Check to make sure it's available first before deploying the template.

ii) aseLocation – make sure this is the same as the rest of the other resources (West US).

iii) ipSslAddressCount – leave it as 0 since we're deploying it into private network.

iv) existingVnetResourceId – ID of the VNet where you'll deploy ASE to. You can get this ID from the portal by navigating to the **LoB VNet** then click on **Properties** tab. Copy the

**RESOURCE ID** at the top, which should be in this format: /subscriptions/your-subscription-id-here/resourceGroups/your-resource-group-name-here/providers/Microsoft.Network/virtualNetworks/your-vnet-name-here.

v) subnetName – name of the subnet where ASE will reside. ASE should reside in its own subnet with nothing else deployed into it.

vi) internalLoadBalancingMode – specifies how to configure the load balancer for ASE. Use mode 1 for now. 0 = public VIP only, 1 = only ports 80/443 are mapped to ILB VIP, 2 = only FTP ports are mapped to ILB VIP, 3 = both ports 80/443 and FTP ports are mapped to an ILB VIP.

vii) dnsSuffix – the domain/subdomain name that ASE will use. This should map to your corporate domain name. For example, if you specify dnsSuffix as contoso.com and you create a web app name helloworld, then the URL for the web app will be helloworld.contoso.com.

viii) frontEndSize, frontEndCount – size and instance count for the ASE frontend. The minimum required by ASE is size=Medium and count=2.

ix) workerPool[x]InstanceSize, workerPool[x]InstanceCount – specifies the VM size and number of instances for different worker pools. You only need to create 1 worker pool and the minimum size is small and count is 2.

x) Click **Save** when done editing the parameters.

g) Click **Deploy**

h) View Visual Studio Output window to make sure deployment was successful.

i) Validate ASE Deployment

i) Navigate to your resource group, you should now see the ASE, the App Service Plan, and the Web App

ii) Validate that the ASE was created in the correct VNet and subnet.

iii) Validate that the Web App is correctly created with the expected URL.

7) Configure DNS for ASE – add DNS records to publish and access Web Apps from inside private network. Deployment duration: < x minutes.

a) Get internal IP address of ASE internal load balancer.

i) In Azure portal, navigate to ASE resource just created. Click on **Properties** tab.

ii) Copy the **VIRTUAL IP ADDRESS** field and save it for later use.

b) RDP into jump box.

c) Inside jump box, start DNS Manager by clicking on the **Start** button and type in **DNS**. Connect to your primary domain controller (SFWUS-NP-VM-PDC).

d) In DNS Manager:

i) Expand the server node, then **Forward Looking Zones**, then select your domain.

ii) Add **New Host** pointing to ASE internal load balancer IP address, the one you copied earlier.

(1) Right click on your domain name and choose **New Host (A or AAAA)…**

(2) **Name** = aseilb, **IP Address** = IP address copied earlier.

(3) Click on **Add Host**, **OK**, then **Done.**

iii) Add new alias records required for publishing to ASE.

(1) Right click on your domain name and choose **New Alias (CNAME)…**

(2) **Alias name** = *.scm, **Fully qualified domain name (FQDN)** = aseilb. <your-domain-name-here>.

(3) Click on **OK.**

(4) Create another alias with **Alias name = publish** and same FQDN as in previous step.

iv) Add new alias for your Web App.

(1) Right click on your domain name and choose **New Alias (CNAME)…**

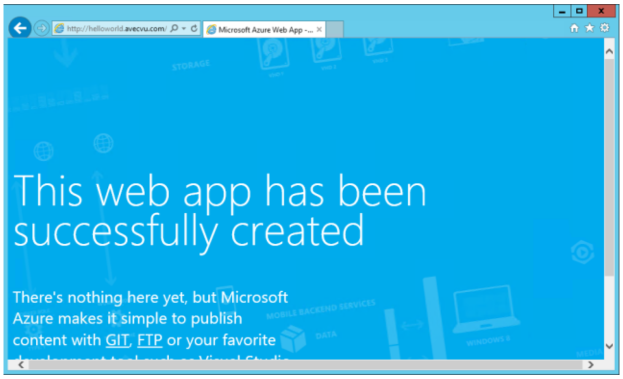
(2) **Alias name** = [your-web-app-name-here], **Fully qualified domain name (FQDN)** = aseilb. [your-domain-name-here].

(3) Click on **OK.**

e) Validate DNS settings

i) Inside jump box, launch a web browser and enter the URL [your-web-app-name-here].[your-domain-name-here]. This is the URL for the Web App you created in step 7c.

ii) You should see something similar to the following:



This web app has been successfully created

There's nothing here yet, but Microsoft Azure makes it simple to publish content with GIT, FTP or your favorite

8) Deploy Application Gateway with Web Application Firewall (WAF). Deployment duration: ~25 minutes.

a) In Visual Studio Solution Explorer, right-click on **LoBNetworkResourceGroup** project. Hover over **Deploy** and select the previous deployment which deployed the network resources (NetworkRG).

b) Make sure the correct subscription and resource groups are selected from the dropdowns.

c) Click on the **Deployment template:** dropdown and choose **applicationgateway.json**.

d) Click on the **Template parameters file:** dropdown and choose **applicationgateway.parameters.json**.

e) Click on **Edit Parameters** button and fill out the required parameters. You can provide either existing or new resources. ARM deployment will create the resource if it's not already there and reuse if the resource already exists with the same name. Here are things to note about some of the parameters:

i) virtualNetworkName – name of Security virtual network

ii) subnetName – name of ingress security subnet

iii) publicIPAddressName, applicationGatewayName – these are name of the respective resources.

iv) applicationGatewaySize – adjust the size of the VMs to fit your situation.

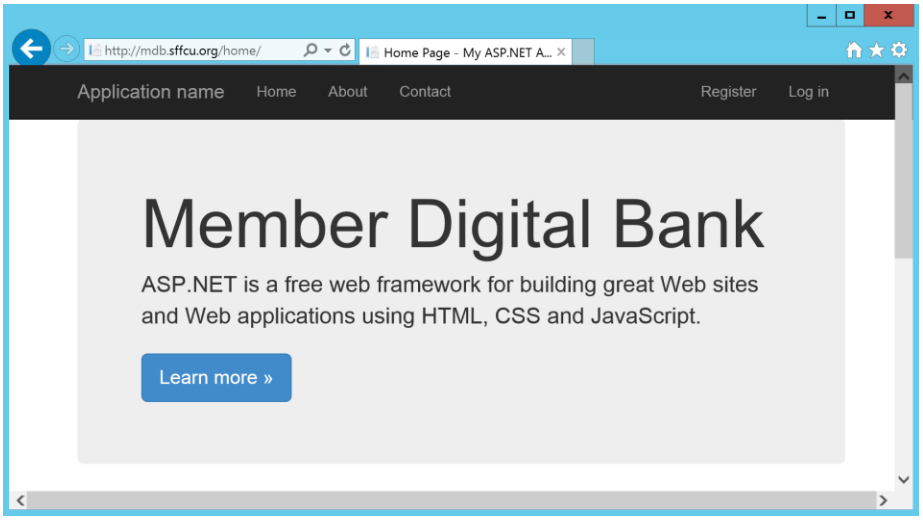
v) applicationGatewayInstanceCount – Keep it at 2.

vi) frontendPort, backendPort – default port 80. If you want to use 443, then update the ARM template to include certificate.

vii) backendIPAddresses – array of IP addresses or URLs to where you'd like to redirect traffic. In our case, we'll use the internal IP address of the ASE (172.25.254.8).
viii) cookieBasedAffinity – Enable/Disable sticky session.
ix) Click **Save** when done editing the parameters.

f) Click **Deploy**
g) View Visual Studio Output window to make sure deployment was successful. Validate in the Portal that the gateway was created.
h) Enable Web Application Firewall feature:
   i) In the Azure Portal, navigate to the Application Gateway resource (SFWUS-NP-AG-WAF1)
   ii) Click on the **Configuration** section
   iii) Click on **WAF Tier**
   iv) Choose **Enabled** for Firewall status
   v) Choose **Prevention** for Firewall mode
   vi) Click **Save** at the top.
i) Enable Diagnostics logs
   i) Click on **Diagnostics logs**
   ii) Click on **Turn on diagnostics** to collect the logs.
   iii) Change **Status** to **On** and then enable **Archive to a storage account**
   iv) In Storage Account section, choose the storage account that was created in created in the earlier steps. If you want to put the logs in a separate storage account, then go ahead and create then put the logs on that new storage account.
   v) Check all the boxes under the **LOG** & **METRIC** sections.
   vi) Click **Save** at the top.

9) Deploy sample web app to ASE. We will follow this blog post to create the new web app. The instructions in that blog post shows how to deploy to the public, multi-tenant App Service. We will augment that post with the below instructions on how to deploy to a private App Service environment. Deployment duration: < 15 minutes.
   a) RDP into the jump box.
   b) Follow the instructions in the blog post to :
      i) Launch & connect Visual Studio to Azure
      ii) Open Cloud Explorer
      iii) Create a new Web Site solution
      iv) Code your web page
   c) Follow below instructions to publish web app to your internal ASE
      i) Back in Solution Explorer, right-click on your WebSite solution and click **Publish Web App**.
      ii) In the Publish window, click on **Microsoft Azure App Service** and a new window opens.
      iii) Select the subscription where ASE was provisioned in.
      iv) In the bottom multiline textbox, you should see your LoBAppRG resource group. Expand it, and you should see the Web App created in step 7c. Select it and click **OK.**
      v) Click **Publish**.
      vi) If everything is configured correctly, then you should see the web page you created in step 10b. Mine looks like this:

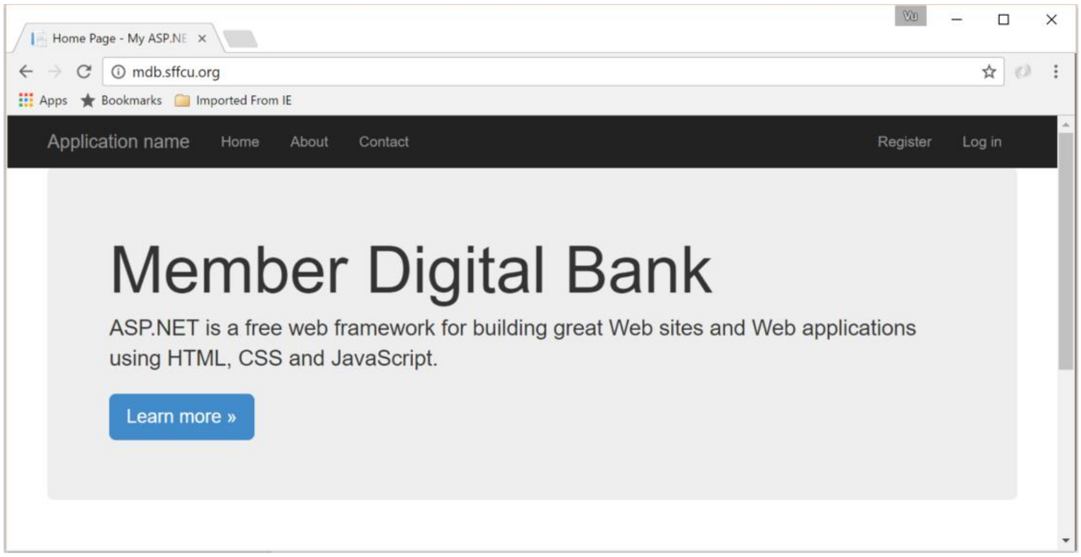


Member Digital Bank

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

10) Deploy Application Gateway health probes. Deployment duration: < 10 minutes.
   a) Run PowerShell script Add-ProbeToExistingGateway.ps1 in the LoBNetworkResourceGroup project, under the Scripts folder. Make sure to run the prerequisites first to log into Azure and select the appropriate subscription.
   b) Wait for the last PowerShell statement to finish. Make sure there are no errors.

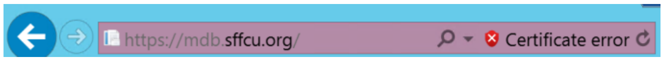
11) Validate Application Gateway deployment. Deployment duration: < 10 minutes.
   a) Get public IP address of Application Gateway
      i) In Azure portal, navigate to your NetworkRG resource group.
      ii) Find and click on the Public IP Address associated with the Application Gateway. The name of the Public IP Address resource was specified in the applicationgateway.parameters.json file.
      iii) Copy the IP Address which should be shown at the top of the Overview section. Save it for use later.
   b) To properly test the App Gateway from outside the corporate network, you'd have to update your external DNS records to point to the new IP address of our new Gateway. For now, we will just update the host file on your computer. Execute the following steps on your laptop:
      i) Copy C:\Windows\System32\drivers\etc\hosts to your desktop
      ii) Add the following line to the bottom of the host file. Replace those between brackets with the appropriate values.
      [public-ip-address-of-gateway] [your-web-app-name-here].[your-domain-name-here]
      iii) Save hosts file.
      iv) Copy hosts file from Desktop back to C:\Windows\System32\drivers\etc. Confirm overwrite of file and provide administer permission when asked.
      v) Bring up a browser and enter this URL: [your-web-app-name-here].[your-domain-name-here]. You should see the same web page you created in step 10b. Mine looks like this:



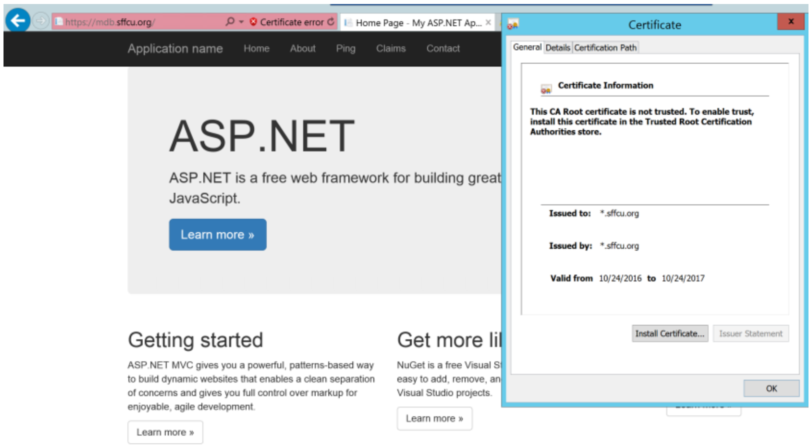
Member Digital Bank

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

12) Deploy Certificates for ASE
   a) Run PowerShell script Create-SelfSignCertificate.ps1 in \LoBResourceGroup\Scripts folder with the following modified variables:
      i) $certDNSName = "*.sffcu.org", or whatever domain name was used to create the ASE. Here, we're creating a wildcard certificate for the sffcu.org domain.
      ii) $certFilePath = "[file path for where to save the pfx file]"
   b) Once you run the PowerShell script, you should find the pfx file in the location you specified earlier.
   c) Upload pfx file to ASE
      i) In Azure Portal, navigate to ASE
      ii) Click on **ILB Certificate**, under the General section.
      iii) Click on **Set ILB Certificate**
      iv) Select the pfx file that was created earlier and provide the password used to create the self-sign certificate
      v) Click **Upload** button
      vi) This step takes a little while to process and the message that a scaling operation is in progress will be shown.
   d) Validate certificate
      i) RDP into Jump Box
      ii) Launch a web browser and navigate to the web app using https – https://mdb.sffcu.org. You should see a warning that the certificate is not from a trusted certificate authority, that's because we're using a self-signed certificate. Click on **Continue to this website**.
      iii) Click on the **Certificate** error link in the browser address bar and then click on **View certificates**



      iv) You should see that the certificate you created and uploaded was used for this web app, similar to what's shown below.



ASP.NET

ASP.NET is a free web framework for building great
JavaScript.

Getting started

ASP.NET MVC gives you a powerful, patterns-based way to build dynamic websites that enables a clean separation of concerns and gives you full control over markup for enjoyable, agile development.

Get more lil

NuGet is a free Visual S
easy to add, remove, an
Visual Studio projects.

Certificate Information

This CA Root certificate is not trusted. To enable trust, install this certificate in the Trusted Root Certification Authorities store.

Issued to: *.sffcu.org

Issued by: *.sffcu.org

Valid from 10/24/2016 to 10/24/2017

13) Deploy Certificate for Application Gateway

a) Run Windows PowerShell ISE as Administrator

b) Copy the script from Configure-AppGatewayCertificate.ps1, located in \LoBNetworkResourceGroup\Scripts folder, to the PowerShell ISE window.

c) Modify the following variables in the PowerShell script to fit your deployment:

i) $gatewayResourceGroup = "MDBNetworkRG" – name of resource group containing the application gateway

ii) $appGatewayName = "SFWUS-NP-AG-WAF1" – name of application gateway

iii) $certName = "sffcu-selfsign" – name of gateway certificate.  Can be anything.

iv) $certDNSName = "*.sffcu.org" – DNS of the certificate.  Only required if recreating the self-sign certificate again.  If reusing existing certificate, then you can leave this variable blank.

v) $certFilePath = "C:\temp\sffcu-selfsign-wildcard.pfx" – where the existing certificate is located or where you'd like to save the new certificate.

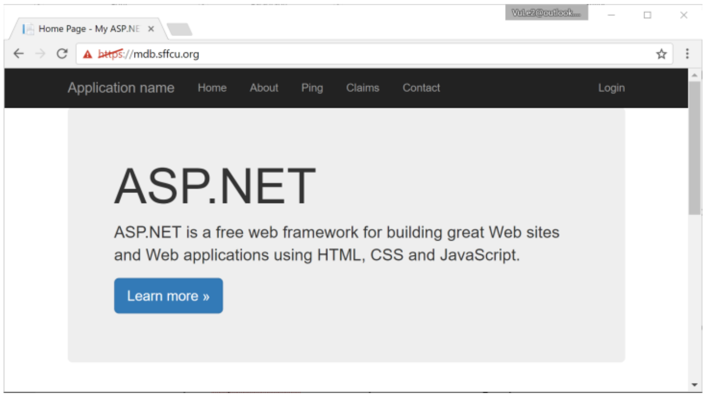
d) In PowerShell ISE, login into Azure and select the default subscription.

e) Run Script (F5)

f) Validate app gateway certificate

i) On your own laptop, bring up a browser and type https://mdb.sffcu.org into the browser address bar.

ii) If everything is correct, then you should see something similar to the following:



ASP.NET

ASP.NET is a free web framework for building great Web sites and Web applications using HTML, CSS and JavaScript.

14) Create Certificates for Azure Service Fabric

a) Click on Start button and run Windows PowerShell ISE as **Administrator**

b) Copy all code from LoBResourceGroup\Scripts\CreateKeyVaultAndCertificate.ps1 to PowerShell ISE

c) Set the following variables to the correct values:

  i) $KeyVaultName – name of your new or existing Key Vault

  ii) $KeyVaultSecretName – Key Vault Secret name

  iii) $ResourceGroupName – resource group that contains the Key Vault

  iv) $Location – location of Service Fabric cluster

  v) $ClusterName – name of Service Fabric cluster

  vi) $Password = certificate Password

d) Click on Run (F5).

e) Once the script finishes, you should see values for the following, which will be used to populate the parameters of our next template.

  i) Source Vault Resource Id

  ii) Certificate URL

  iii) Certificate Thumbprint

15) Deploy Azure Service Fabric. Deployment duration: ~25 minutes.

  a) In Visual Studio Solution Explorer, right-click on **LoBResourceGroup** project. Hover over **Deploy** and select the deployment that contains the ASE.

  b) Select the subscription

  c) Select the **Resource group** where ASE was deployed to

  d) Click on the **Deployment template:** dropdown and choose **asf-private.json**.

  e) Click on the **Template parameters file:** dropdown and choose **asf-private.parameters.json**.

  f) Click on **Edit Parameters** button and fill out the required parameters. Most parameters are self explanatory except for the following:

    i) sourceVaultValue – copy **Source Vault Resource Id** from previous step

    ii) certificateUrlValue – copy **Certificate URL** from previous step

    iii) certificateThumbprint – copy **Certificate Thumbprint** from previous step

    iv) clusterName – globally unique Service Fabric cluster name

    v) adminPassword – update with your own Key Vault secret

    vi) existingVNetResourceGroupName – name of resource group that contains the VNet

    vii) the rest can be kept as default.

    viii) Click **Save** when done editing the parameters.

  g) Click **Deploy**

  h) View Visual Studio Output window to make sure deployment was successful.

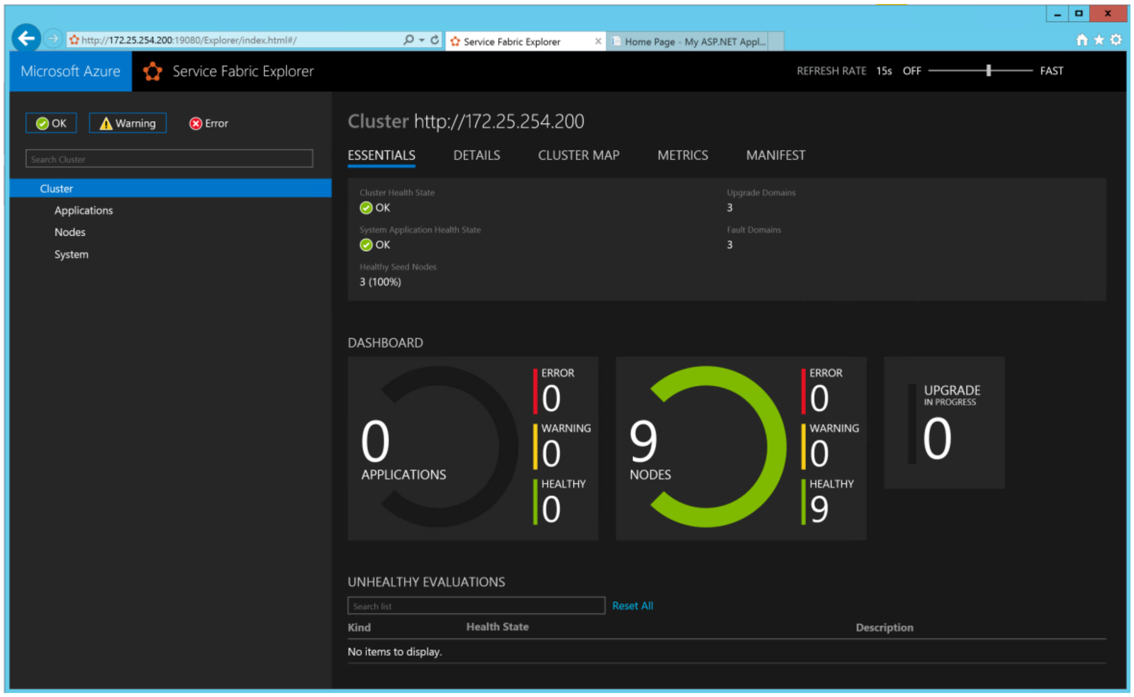
16) Validate Azure Service Fabric deployment

  a) Go to Azure portal and click on the Service Fabric resource that was just created.

  b) In the Overview pane, copy the URL for **Service Fabric Explorer.**

  c) Remote Desktop into jump box

  d) Bring up a browser and paste the Service Fabric Explorer URL into the address bar. After a few minutes, you should see a cluster with all healthy nodes, similar to below:



Cluster http://172.25.254.200

Cluster Health State: OK

Upgrade Domains: 3

System Application Health State: OK

Fault Domains: 3

Healthy Seed Nodes: 3 (100%)

DASHBOARD

0 APPLICATIONS — ERROR 0, WARNING 0, HEALTHY 0

9 NODES — ERROR 0, WARNING 0, HEALTHY 9

UPGRADE IN PROGRESS 0

UNHEALTHY EVALUATIONS

| Kind | Health State | Description |
|---|---|---|
| No items to display. | | |

e) If any of the nodes are no healthy, just find the node in the Service Fabric Explorer, click on the ellipsis (…) and then choose restart.