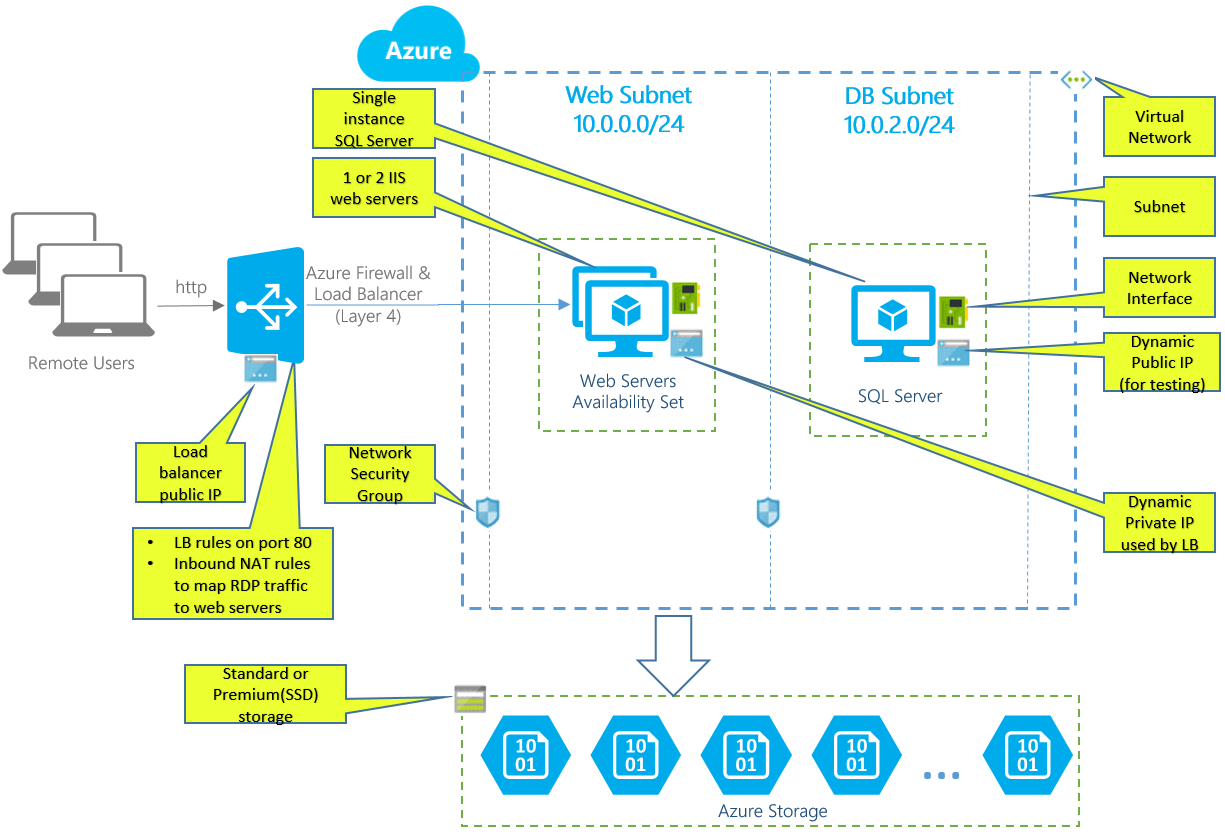
**Deploying app on VM using ARM Templates**

In this task you will learn how to create a virtual machine using ARM Templates.

**Resources and Architecture Diagram:**

The following resources are created by this template:

* 1 or 2 Windows 2012R2 IIS Web Servers.
* 1 SQL Server 2014 running on premium or standard storage.
* 1 virtual network with 2 subnets with NSG rules.
* 1 storage account for the VHD files.
* 1 Availability Set for IIS servers.
* 1 Load balancer with NATing rules.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/create-architecture-diagram-vm.png?raw=true)

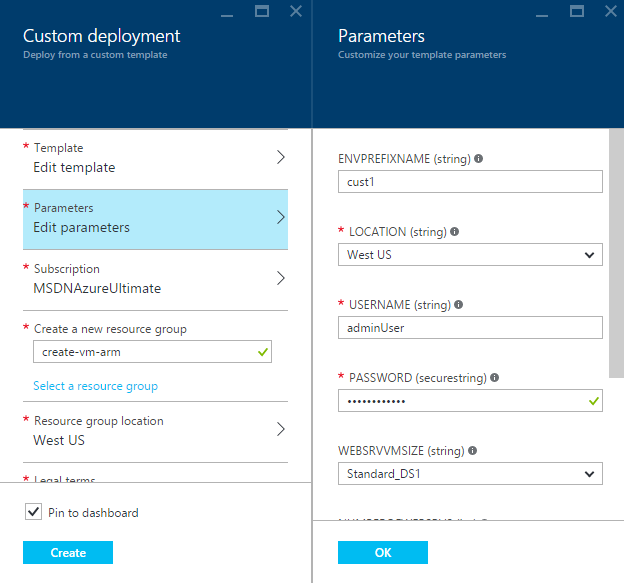
*Architecture diagram for ARM Template*

The below **Deploy to Azure** button embeds an Azure ARM template which creates one or two Windows Server 2012R2 VM(s) with IIS configured using DSC. It also installs one SQL Server 2014 standard edition VM, a VNET with two subnets, NSG, loader balancer, NATing and probing rules.

[https://camo.githubusercontent.com/9285dd3998997a0835869065bb15e5d500475034/687474703a2f2f617a7572656465706c6f792e6e65742f6465706c6f79627574746f6e2e706e67](https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2Fazure%2Fazure-quickstart-templates%2Fmaster%2Fiis-2vm-sql-1vm%2Fazuredeploy.json)

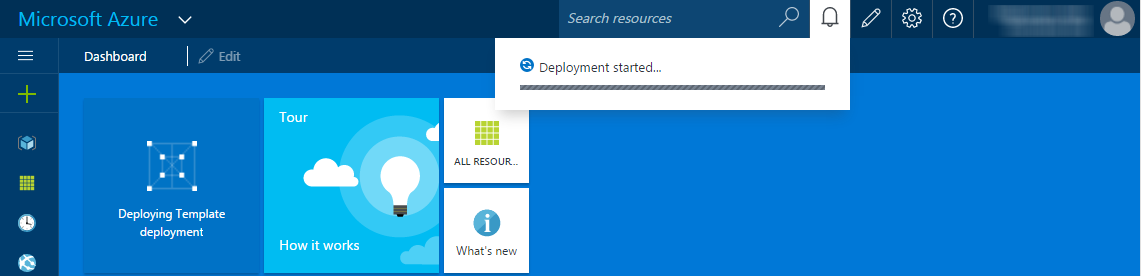
In order to create virtual machine using ARM, perform the following steps:

1. Click on the **Deploy to Azure** button which will navigate you to Azure Portal with Custom Deployment.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/create-vm-arm-parameters.png?raw=true)

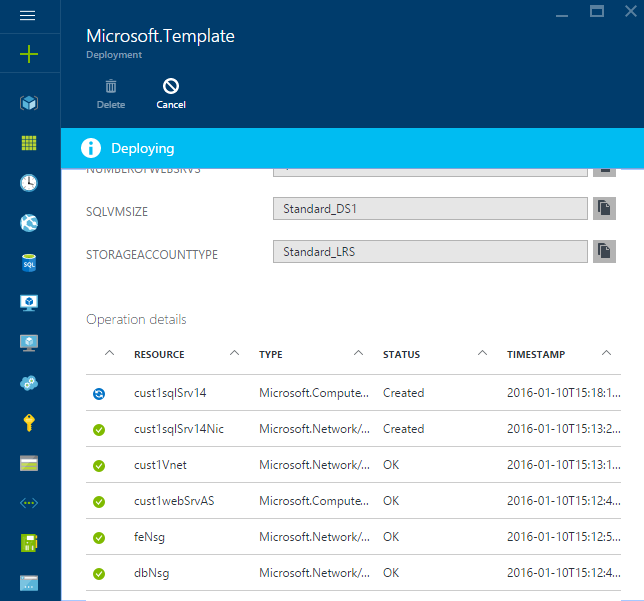
*Custom Deployment using Azure ARM - using Deploy to Azure button*

1. On the Custom Deployment blade that opens, enter the parameters:
   * **ENVPREFIXNAME**: virtual machine name (e.g. azureVM)
   * **Location**: location for the virtual machine (e.g. West US)
   * **User Name**: user name for the administrator account (e.g. adminUser)
   * **Password**: unique password for the administrator account
   * **WEBSRVVMSIZE**: unique password for the administrator account (e.g. Standard\_DS1)
   * **NUMBEROFWEBSRVS**: unique password for the administrator account (e.g. 1)
   * **SQLVMSIZE**: unique password for the administrator account (e.g. Standard\_DS1)
   * **STORAGEACCOUNTTYPE**: unique password for the administrator account (e.g. Standard\_LRS)
2. Review the default settings, such as the **Subscription**(if you have multiple), **Resource Group**(create or select an existing group), and **Resource Group Location** and finally **Legal Terms**, then Click **Create** which will add the resource group on to dashboard.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/create-vm-arm-dashboard.png?raw=true)

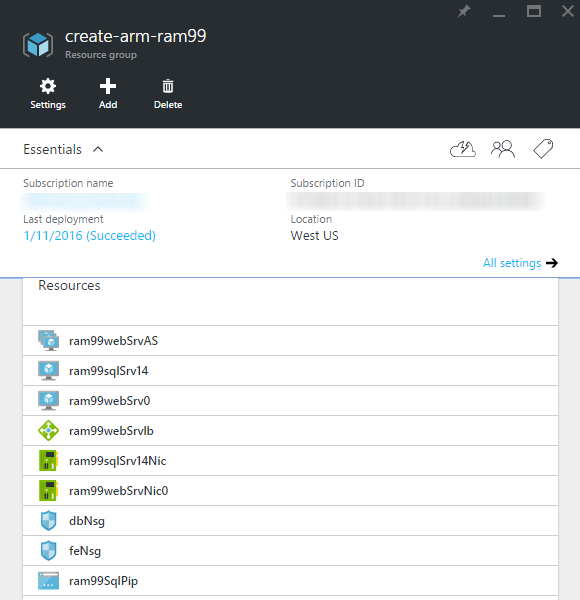
*Creating the architecture using ARM Template*

1. In the meanwhile, you can click on **Deploying Template deployment** which was created on the dashboard.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/deploying-template-deployment-status.png?raw=true)

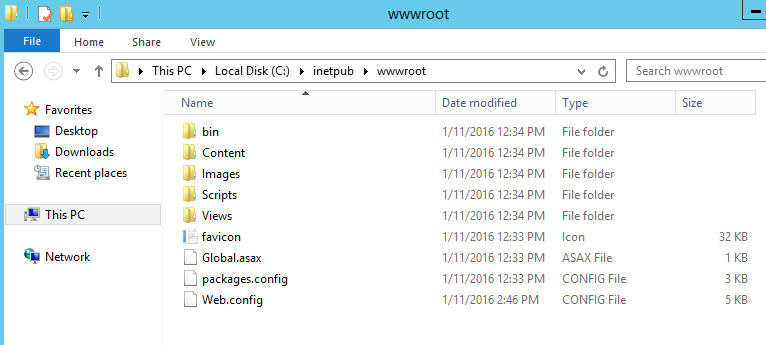
*Azure ARM Template deployment Status*

1. Once the Template Deployment succeeds, you will have WebServer with IIS Installed and SQL Server 2014 Standard deployed on a Virtual Netowrk with 2 subnets with NSG rules and a Load Balancer with NATing rules. Click on the Resource Group Tile pinned on the dashboard and then click on each resource for more details.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/succeeded-arm-template.png?raw=true)

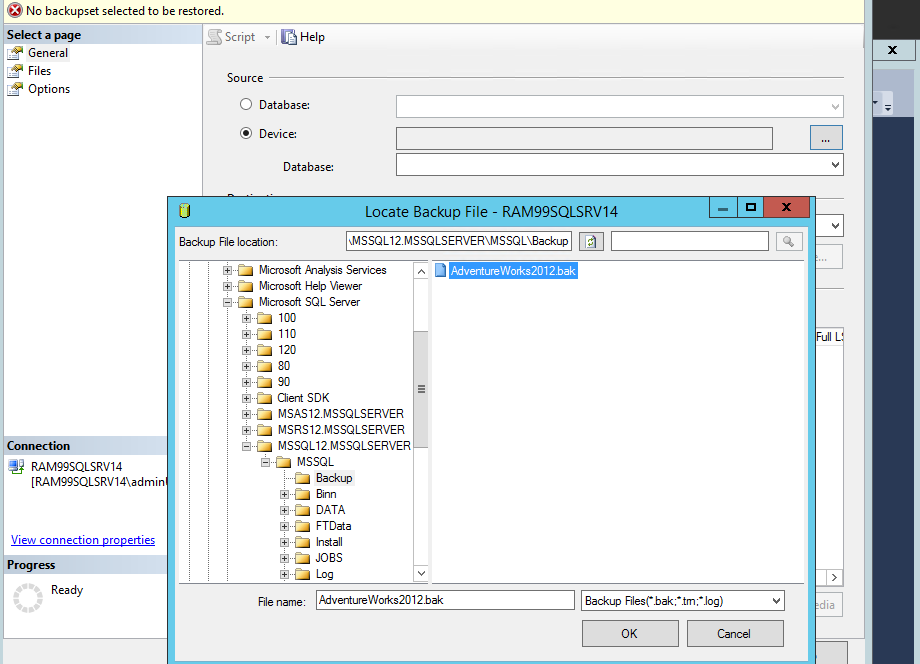
*Azure ARM Template deployment Succeeds*

1. Now, the 2-tier architecture is created with all necessary resources, to check whether all the rules are applied we will deploy an ASP.NET application and a Sample Database(AdventureWorks2012).
   * Click on **dbNsg** which is a Netowork security group and delete the outbound Security rules (Prioirity - 200) - We do this step to download the below AdventureWorks2012 database.
   * Download RDPs for both Database server and Application server and login with the credentails and download the ASP.NET application content into appserver and AdventureWorks2012 database into Database server.
   * ASP.NET Application content can be downloaded here: [application content](http://opsgilitytraining.blob.core.windows.net/armhackathon/cloudshop.zip)
   * Sample Database can be downloaded here : [AdventureWorks2012](http://opsgilitytraining.blob.core.windows.net/public/AdventureWorks2012.bak)
   * Once you download application content onto app server extract the .zip file and copy the content and past in **C:\inetpub\wwwroot**.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/inetpub.png?raw=true)

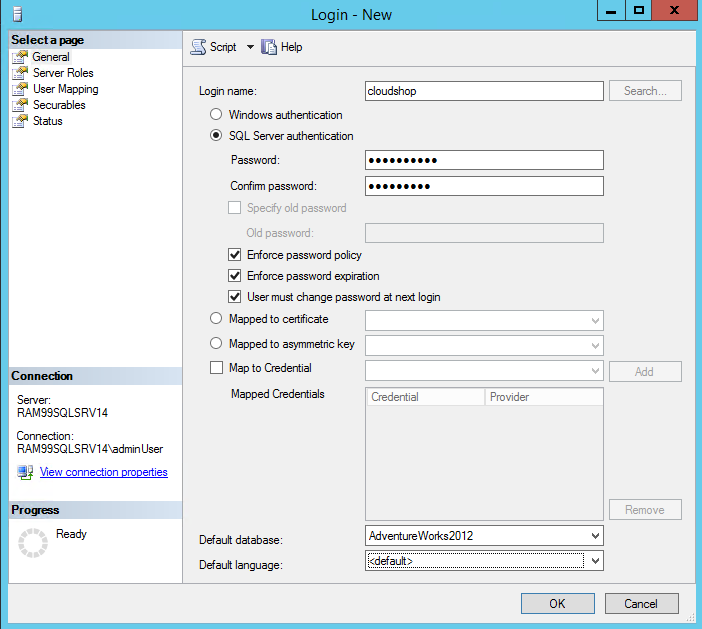
*Copy content in wwwroot*

* + Open Database Server, and Open SQL Server Management Studio 2014 login with Windows Authentication for restoring the AdventureWorks2012 database.
  + Copy the .bak file to the Backup location "C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\Backup" and click **OK**.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/backup-database.png?raw=true)

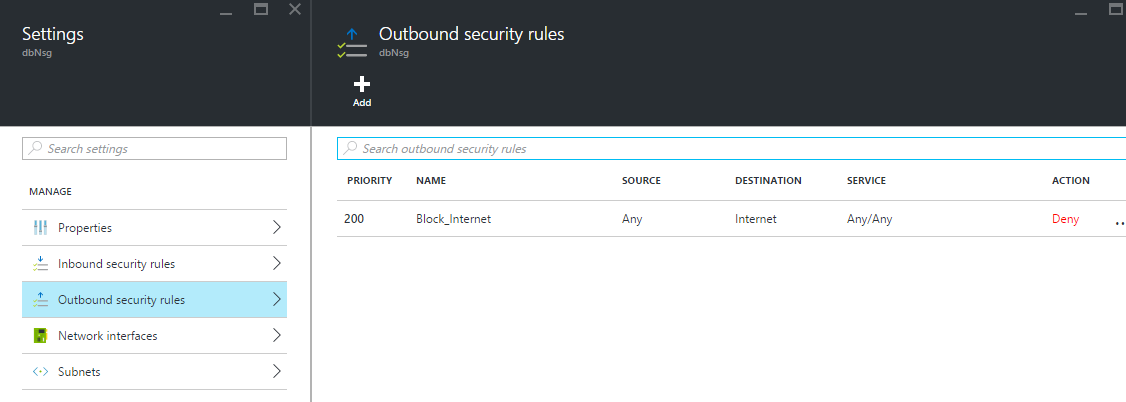
*Restoring Backup database file*

* + In object explorer go to Security section and Login subsection Right Click and New Login and create a user with SQL Server authentication and in the default database select as **AdventureWorks2012**

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/login-user.png?raw=true)

*Create a login user*

* + On Left side you have **Server Roles** -> Select **public** and **sysadmin** and check in **User Mapping** whether **public** is selected or not and click **OK**
  + So till now we have an application content in appserver and database in Database Server. Fianlly we have to setup the NSG Outbound rule which we have deleted earlier to Database Server through portal.
  + In Azure Portal Click on the resource group which we just created and in the resources click on **dbNsg** and click on**All Settings** and Outbound security rules and click on add and fill the details as below:

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/addnsgrule-outbound.png?raw=true)

*Adding a Outbound NSG rule*

* + And Finally login into AppServer and open inetpub\wwwroot and Open **Web.Config** in notepad and replace the**ConnectionString** with the below code:

<add name="DefaultConnection" connectionString="Data Source=tcp:{Destination-Internal-IP},1433;Initial Catalog=AdventureWorks2012;User ID={User-created-DbServer};Password={password};Encrypt=true;Trusted\_Connection=false;TrustServerCertificate=true" providerName="System.Data.SqlClient"/> <add name="AdventureWorksEntities" connectionString="metadata=res://\*/Models.AdventureWorks.csdl|res://\*/Models.AdventureWorks.ssdl|res://\*/Models.AdventureWorks.msl;provider=System.Data.SqlClient;provider connection string=&quot;data source=tcp:{Destination-Internal-IP},1433;initial catalog=AdventureWorks2012;Uid={User-created-DbServer};Password={password};multipleactiveresultsets=True;App=EntityFramework&quot;" providerName="System.Data.EntityClient" />

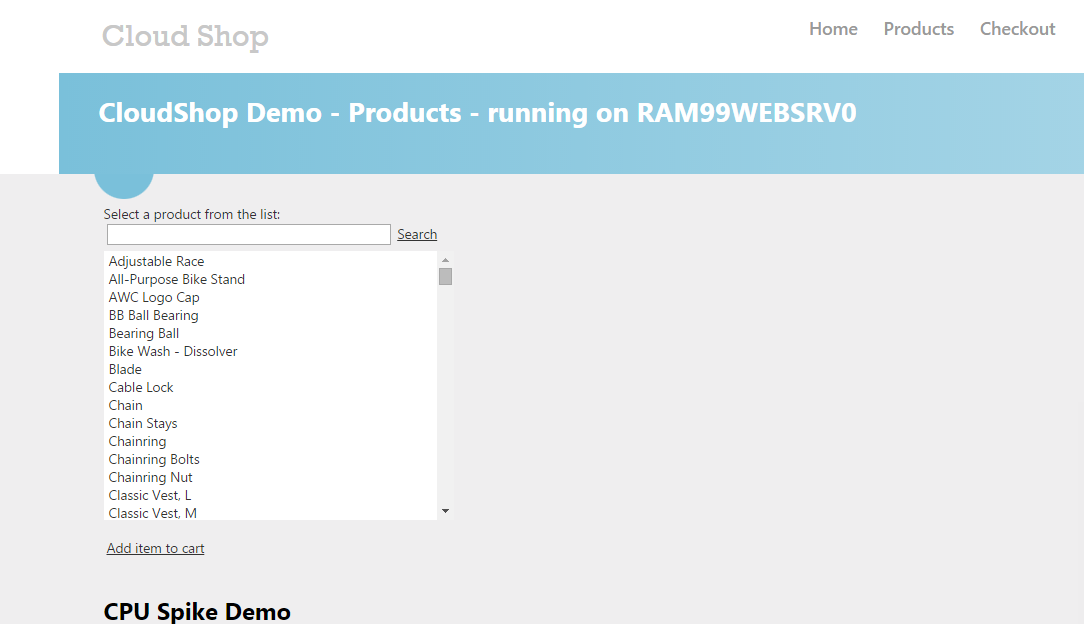
NOTE: Before replacing the connection string, do change the following text with respective values

1) Destination IP address : {Destination-Internal-IP} -> Destination IP

2) User ID: {User-created-DbServer} -> User ID for SQL Authentication

3) Password: {password} -> Password for SQL Authentication

* + Now you can verify by copying the Loadbalancer IP address onto browser and you will see an asp.net application with data populating from the DB Server.

[](https://github.com/opsgility/DevCamp/blob/master/HOL/create-virtual-machine/images/output-demo.png?raw=true)

*Output of the application*

*END OF THE DOCUMENT*