***Foot Locker Assignment***

Provision environment having Azure Application Gateway pointing to two of the backend webservers either using Terraform or ARM template

 VM1

  VM2

VNET

**Application Gateways:**

Azure Application Gateway is a web traffic load balancer and Application Delivery Controller (ADC) that enables you to manage traffic to your web applications. Azure Application Gateway offers a **Layer 7** load-balancing feature for HTTP and HTTPs traffic, and you can route traffic based on the incoming URL.  HTTP Traffic requests are routed in a round-robin method to the backend servers.  In cases where traffic needs to be routed to a specific server such as a shopping cart, cookie-based session affinity is used.  The Azure Application Gateway provides end-to-end SSL encryption, thus offloading the computational tasks for decoding SSL requests. Traditional load balancers operate at the transport layer (OSI layer 4 – TCP and UDP) and route

**Azure Application Gateway Features**

These are some of the features of the application gateway. In the walk-through section, we’ll configure use cases for each of these features.

**SSL Termination**

SSL Offloading and End-to-End SSL encryption allows you to securely transmit sensitive data to the backend encrypted.

**URL-Based Routing (Layer 7)**

URL Path-Based Routing allows you to route traffic to back-end server pools based on URL Paths of the request.

**Redirection**

A common scenario for many web applications is to support automatic HTTP to HTTPS redirection to ensure all communication between the application and its users occurs over an encrypted path.

**Multiple-Site Hosting**

Multiple-site hosting enables you to configure more than one web application on the same application gateway instance. Each website or domain can be directed to its own backend pool.

**Cookie-Based Session Affinity**

The cookie-based session affinity feature is useful when you want to keep a user session on the same server. By using gateway-managed cookies, Azure Application Gateway can direct subsequent traffic from a user session to the same server for processing. traffic based on the source IP address and port, to a destination IP address and port.

**ARM Templates**

With Azure Resource Manager (ARM) you can deploy, delete, or update all resources for your solution in a single and coordinated operation by using templates. The templates are in the form of JSON format, and these templates can be used for deployment in different environments such as staging, testing, and production. The Resource Manager helps in providing auditing, security, and tagging features to help you manage your resources post-deployment.

Although we have the Azure portal for managing the resources but in cases where you need to maintain and deploy multiple servers, in complex configurations - it might get a cumbersome to handle these deployment and configuration activities via the point and click portal. Moreover, remembering every resource category and their respective cost evaluation can become a mammoth task.

As companies look forward to automating deployment, scaling, and operations of cloud applications in the Azure cloud (across clusters of hosts), there is a need for providing automated deployment solutions for the cloud infrastructure. With Azure ARM Templates, you can define your infrastructure environment in simple JSON file template. These file templates can be versioned, validated and checked into your source control systems. This is what we call Infrastructure-as-code (IaC).

***ARM Template Structure***

{

"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

"contentVersion": "1.0.0.0",

"parameters": {

},

"variables": {

"comment": "Azure QuickStarts"

},

"resources": [

],

"outputs": {

}

**$schema**

Location of the JSON schema file that describes the version of the template language.

**Content Version:**

Version of the template. You can provide any value for this element. Use this value to document significant changes in your template. When deploying resources using the template, this value can be used to make sure that the right template is being used.

**Parameters**

When you start a new deployment and you have parameters defined in your resource template then these needs to be entered in before the deployment can start. It are values used for that specific deployment. These parameters are then used in other sections of the template. When we look at a parameter in the parameters section it can be for example a value to select a specific Operating System version when deploying a Virtual Machine:

**Variables**

The variables are values that can be constructed for example from parameters to create a variable and use that in other sections of your deployment template. Or let’s say you want to use a single value across multiple resources. When defined in a variable you can reuse that variable in the resources section of your deployment template so when it needs to change you can change it in your template just in one place, and you don’t need to specify it each time at deployment as a parameter. Here you see an example of the variables section in a resource template:

**Resource Group**

The resources section in the template are the actual resources that you are going to deploy or update. This can be a collection of resources and their configuration settings (properties) you can define directly in the resource or populate them from parameters and/or variables

**Output**

In this section you can define outputs from your template. These values can be for example a connection string from a deployment of a database. This can then be passed into another deployment to use for example as connection string for a website you are going to deploy.

***Deploying ARM templates***

ARM templates can be designed and deployed using by

* Powershell
* Using Azure CLI
* Using VSTS
* Azure portal (Deploying by uploading the file)

**Details of the assignment**

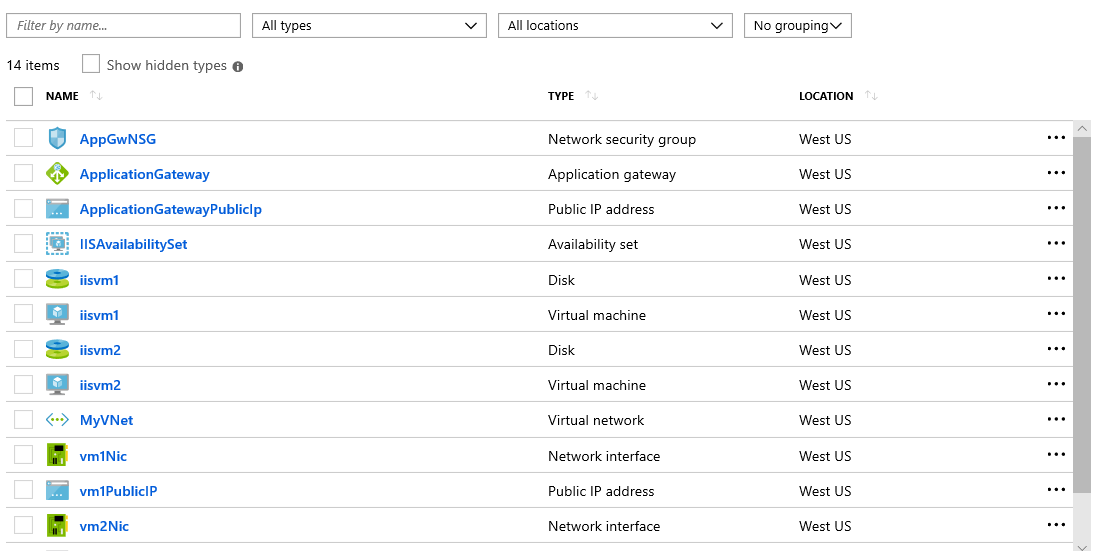
ARM template in the email that deploys application gateways with two backend webservers. I am leveraging the existing template with few of the change and additional parameters to meet the requirements

ARM template parameters include

* **AdminUsername**: <<Provide any user name>>
* **AdminPassword**: << Provide password 12 characters or more>>
* **Windows OS Version**: Windows OS version that you need on the VMs
* **Virtual Machine Size**: Size of the VM’s
* **Application gateway size**: Size of the Gateway
* **Capacity:** number of instances that will be provisioned under the application gateway
* **Waf mode**: default set as prevention mode , a proactive measure in application gateway to block intrusions and attacks detected by its rules.
* **Frontend cert data:** pfx certificate converted to base64 string format. Required to establish a secured connection between application gateway and Virtual Machines
* **Frontend password:** certificate password
* **Backend cert data:** pfx certificate converted to base64 string format. Required to establish a secured connection between application gateway and Virtual Machines
* **Backend password:** certificate password
* **Backend public key**: Base-64 encoded form of the .cer file. This is the public key for the cert on the web servers
* **Backend Cert DNS** name: DNS name of the backend cert
* **Artifacts location**: Location of the artifacts/files that needs to be configured as part of VM
* **Location:** Location to provision the Gateway and the VM’s

Here are the components that will be created once the ARM template deployment is successful

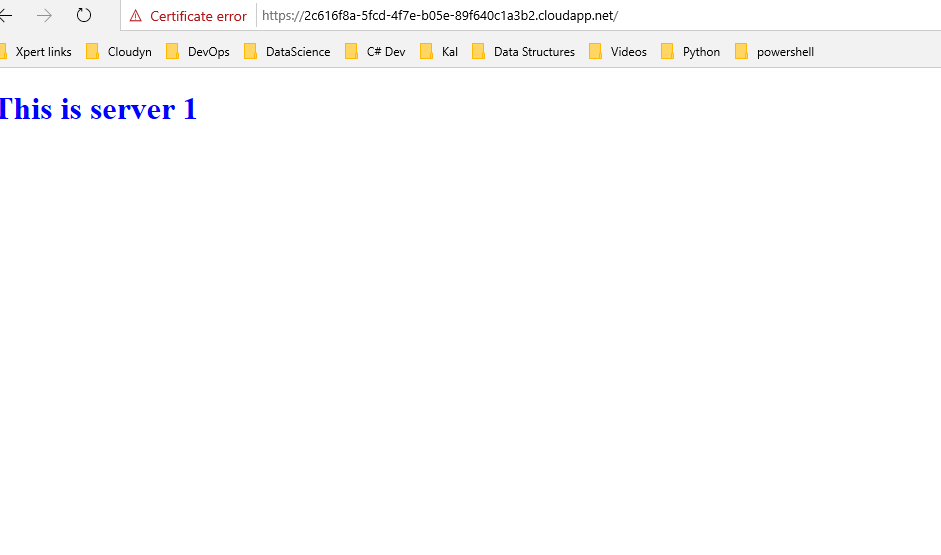
* **Application Gateway:** Application gateway that takes user traffic and communicates to either of the server (VM1 or VM2) based on the round robin mechanism. You can experience it hitting the dns endpoint configured on application gateway
* **Application Gateway NSG**: Network Security Group to allow/deny traffic to two VM’s
* **Application Gateway Public IP**: Public IP address of the application gateway
* **IIS Availability Set**: Availability set will be created as part of the provisioning ensuring each machine is sitting on different fault domain
* **IIS VMs (2):** two Virtual machines will be created as part of the provisioning
* **IIS VMs NIC (2):** One NIC per server will be created
* **Virtual Network**: VNET configured having application gateway and two VM’s part of it
* **VM Public IPs (2):** one dynamic IP on each of the machine will be created as part of provisioning



Here is the application gateway end point

https://2c616f8a-5fcd-4f7e-b05e-89f640c1a3b2.cloudapp.net/

**Server 1**



**Server 2**

