Module 4 - Lab 1: - Azure Application Gateways

Lab Overview

Azure Application Gateways are HTTP/HTTPS load balancing solutions. Compared to Azure Load Balancers which are TCP/UDP load balancing solutions. Similar to Azure Load Balancers, Application Gateways can be configured with internet-facing IP addresses or with internal load balancer endpoints making them inaccessible via the internet. Application Gateways are ideal when you require some of the following features:

- Web-based traffic in any of HTTP, HTTPS, or WebSocket protocols
- TLS/SSL offloading
- Built-in web application firewall
- Cookie affinity for sticky sessions

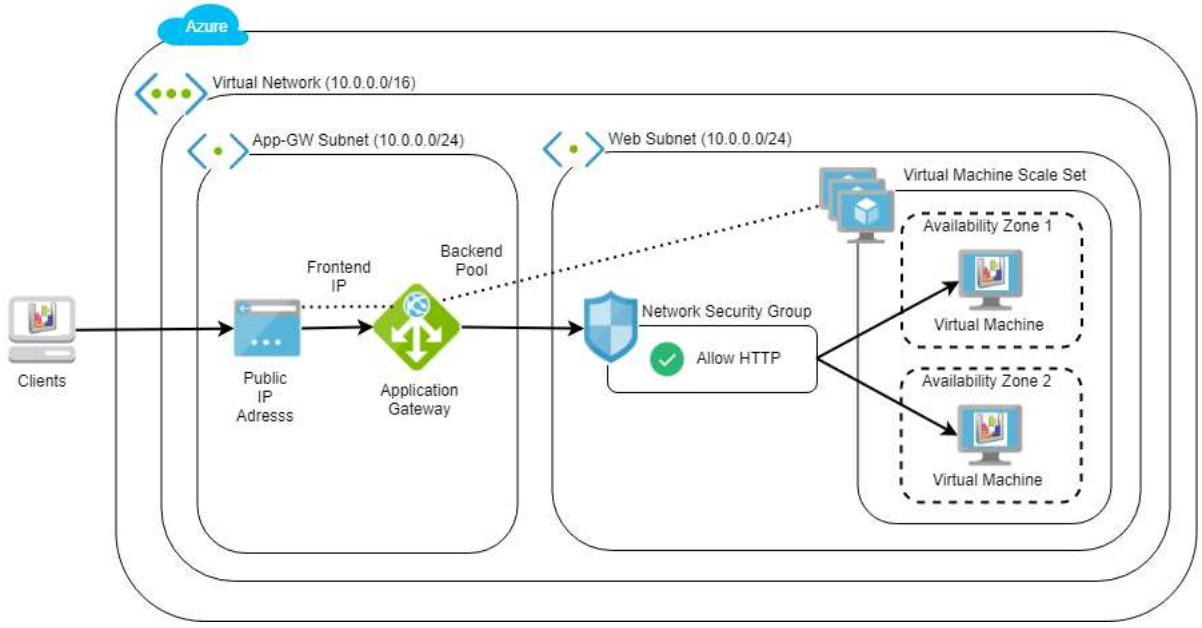
This Lab will take you through a scenario of deploying a web application in Azure, and creating and configuring an Application Gateway to load balance the web application's traffic. The Lab uses the Azure CLI to create and configure resources in the Lab environment.

Lab Objectives

Upon completion of this Lab, you will be able to:

- Use an Application Gateway to load balance application traffic
- Understand the use cases for an Application Gateway and when to use an Azure Load Balancer instead
- Familiarize with the Azure CLI to inspect, create, and update resources in a resource group
- Deploy applications to Virtual Machine Scale Sets (VMSS) using VMSS extensions

The final outcome of the lab is depicted below:



Exercise 1: Understanding Azure Application Gateways and the Lab Scenario

Azure Application Gateways are HTTP/HTTPS load balancing solutions, compared to Azure Load Balancers which are TCP/UDP load balancing solutions. Similar to Azure Load Balancers, Application Gateways can be configured with internet-facing IP addresses or with internal load balancer endpoints making them inaccessible via the internet. Application Gateways are ideal when you require:

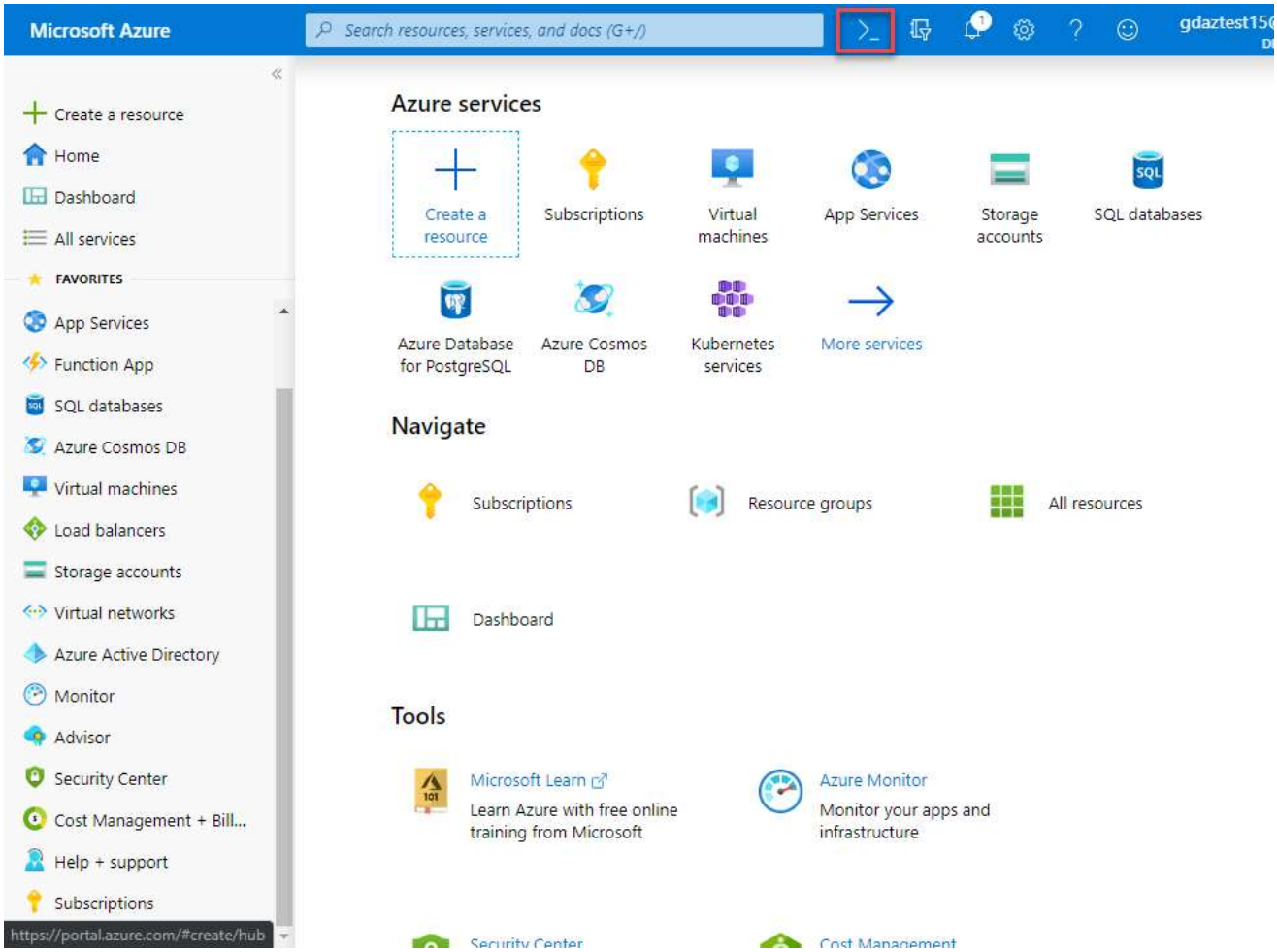
- Web-based traffic in any of HTTP, HTTPS, or WebSocket protocols
- TLS/SSL offloading to improve utilization of backend by offloading encrypt/decrypt operations to the load balancer
- Built-in web application firewall (requires medium Application Gateway type or larger)
- Cookie affinity for sticky sessions

Azure application gateways can be configured using a variety of backends including plain VMs, Azure Web Apps, Azure Kubernetes Service (AKS), and Virtual Machine Scale Sets (VMSS).

In this Lab, you will use the Azure CLI to configure an application gateway to load balance traffic to a VMSS that you will deploy a web application to. VMSSs are sets of identically configured virtual machines (VM) referred to as instances in the VMSS. VMSSs can automatically scale the number of instances based on a variety of metrics. Instances are distributed across availability zones to provide fault tolerance in case of a major failure of a zone within a region.

- [ ] 1. Login to the Azure Portal https://portal.azure.com with the username sheikhnasirKF0PA@gdcs2.com and password w6FuieuIon0xCIKm

- [ ] 2. Select **Cloud Shell** from the Azure Portal tool bar.



- [ ] 3. Select **Bash** on the Welcome screen.

- [ ] 4. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:

    - Leave the **Subscription** drop-down list entry set to its default value.
    - In the **Cloud Shell region:** West US 2.
    - In the **Resource group** section, select the Resource Group that has been created for you.
    - In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
    - In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
    - Click the **Create storage** button.

- [ ] 5. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

- [ ] 6. List the virtual networks and set the default location to WestUS2 where Lab resources are located in the environment by running the following commands:

```
az configure --defaults location=westus2
az network vnet list --output table
```

```
Name    ResourceGroup                Location    NumSubnets    Prefixes     DnsServers    DDOSProtection    VM
otection
------  ---------------------------  ----------  ------------  -----------  ------------  ----------------  --
----------
app     myResourceGroup-CQN79NUGFU   westus2     1             10.0.0.0/16                False             Fa
e
```

There is a single virtual network named **app** that hosts web application resources. There is a single subnet that you will inspect next.

7. List the subnet in the app virtual network by running the following commands:

```
resource_group=$(az group list --query [].name --output tsv)
az network vnet subnet list --resource-group $resource_group --vnet-name app --output table
```

```
AddressPrefix    Name    PrivateEndpointNetworkPolicies    PrivateLinkServiceNetworkPolicies    Provisioning$
te    ResourceGroup
---------------  ------  --------------------------------  -----------------------------------  -------------
----  -------------------------
10.0.0.0/24      web     Enabled                           Enabled                              Succeeded
      myResourceGroup-CQN79NUGFU
```

The **web** Virtual network contains the VMSS that will host the web application.

8. Query the subnet's network security group (NSG) ID:

```
az network vnet subnet list --resource-group $resource_group --vnet-name app --query [].networkSecurityGroup.id --output tsv
```

```
/subscriptions/93fe8ebb-c882-4947-b060-acde1858dc49/resourceGroups/myResourceGroup-CQN79NUGFU/providers/Micr
ft.Network/networkSecurityGroups/website-vmssnsg
```

The output will confirms that an NSG is associated with the subnet and that the NSG is named **website-vmssnsg**.

9. List the rules in the website-vmssnsg NSG by running the following command:

```
az network nsg rule list --resource-group $resource_group --nsg-name website-vmssnsg --output jsonc
```


```
[
  {
    "access": "Allow",
    "description": "Allow web traffic",
    "destinationAddressPrefix": "*",
    "destinationAddressPrefixes": [],
    "destinationApplicationSecurityGroups": null,
    "destinationPortRange": "80",
    "destinationPortRanges": [],
    "direction": "Inbound",
    "etag": "W/\"92b70b88-08ee-4502-8589-f8f37cd53aaf\"",
    "id": "/subscriptions/93fe8ebb-c882-4947-b060-acde1858dc49/resourceGroups/myResourceGroup-CQN79NUGFU/pro
ers/Microsoft.Network/networkSecurityGroups/website-vmssnsg/securityRules/allowHTTP",
    "name": "allowHTTP",
    "priority": 1000,
    "protocol": "Tcp",
    "provisioningState": "Succeeded",
    "resourceGroup": "myResourceGroup-CQN79NUGFU",
    "sourceAddressPrefix": "*",
    "sourceAddressPrefixes": [],
    "sourceApplicationSecurityGroups": null,
    "sourcePortRange": "*",
    "sourcePortRanges": [],
    "type": "Microsoft.Network/networkSecurityGroups/securityRules"
```


The NSG has a single (non-default) rule that allows HTTP (**Tcp** destination port **80**).

10. Page through the output of the VMSS by pressing *spacebar* after entering:

```
az vmss list --output jsonc | more
```

The **website-vmss** has a **capacity** of **2** meaning that two VM instances are in the VMSS. Other properties to note are:

- **"publicIpAddressConfiguration": null** (meaning the instances cannot be reached via the internet)
- **"networkSecurityGroup": null** (meaning there is no additional NSG besides the one associated with the subnet)
- **"offer": "CentOS"** (meaning instances are running the CentOS distribution of Linux)
- **"zones": ["1", "2"]** (meaning the instances are spread across two availability zones in the region)

The instances are not currently running any web application. You will deploy one later.

Summary

In this Exercise, you understood the challenges that an Application Gateway can solve. You also inspected the Lab environment to see the resources that were created when you started the Lab.

Exercise 2: Creating an Application Gateway to Load Balance VMSS Traffic

You will create all of the resources needed to load balance the VMSS traffic using an Application Gateway in this Lab Step. As Microsoft explains, an Application Gateway can be described by five configuration settings:

1. **Backend server pool**: The list of IP addresses of the backend servers. The IP addresses listed should either belong to the virtual network, but in a different subnet for the application gateway, or should be a public IP/VIP.
2. **Backend server pool settings**: Every pool has settings like port, protocol, and cookie-based affinity. These settings are tied to a pool and are applied to all servers within the pool.
3. **Frontend port**: This port is the public port that is opened on the application gateway. Traffic hits this port, and then gets redirected to one of the backend servers.
4. **Listener**: The listener has a frontend port, a protocol (HTTP or HTTPS), and the SSL certificate name (if configuring SSL offload).
5. **Rule**: The rule binds the listener and the backend server pool and defines which backend server pool the traffic should be directed to when it hits a particular listener. The rule can be either basic (round-robin) or path-based

When you first create an Application Gateway it includes an empty default backend pool for you to add addresses to. An Application Gateway must always have at least one backend pool.

- [ ] 1. Create a subnet in the app virtual network by running the following command in the Bash Cloud Shell:

```
az network vnet subnet create --resource-group $resource_group --vnet-name app --name app-gw --address-prefix 10.0.100.0/24
```


```
{
  "addressPrefix": "10.0.100.0/24",
  "addressPrefixes": null,
  "delegations": [],
  "etag": "W/\"8083cb7a-f39e-4ba2-9d28-203c3e577584\"",
  "id": "/subscriptions/93fe8ebb-c882-4947-b060-acde1858d
s/Microsoft.Network/virtualNetworks/app/subnets/app-gw",
  "ipAllocations": null,
  "ipConfigurationProfiles": null,
  "ipConfigurations": null,
  "name": "app-gw",
  "natGateway": null,
  "networkSecurityGroup": null,
  "privateEndpointNetworkPolicies": "Enabled",
  "privateEndpoints": null,
  "privateLinkServiceNetworkPolicies": "Enabled",
  "provisioningState": "Succeeded",
  "purpose": null,
  "resourceGroup": "myResourceGroup-CQN79NUGFU",
```


The output displays the JSON specification of the subnet. The address prefix is chosen to not overlap with the web subnet (10.0.0.0/24).

- [ ] 2. Create a public IP address for the Application Gateway frontend by running the following command:

```
az network public-ip create --resource-group $resource_group --name app-gw-ip --allocation-method Dynamic
```


```
{
  "publicIp": {
    "ddosSettings": null,
    "dnsSettings": null,
    "etag": "W/\"a0de5a3e-a1ea-44a7-88fe-9eb1b2c538d9\"",
    "id": "/subscriptions/93fe8ebb-c882-4947-b060-acde1858dc49/r
ers/Microsoft.Network/publicIPAddresses/app-gw-ip",
    "idleTimeoutInMinutes": 4,
    "ipAddress": null,
    "ipConfiguration": null,
    "ipTags": [],
    "location": "westus2",
    "name": "app-gw-ip",
    "provisioningState": "Succeeded",
    "publicIpAddressVersion": "IPv4",
    "publicIpAllocationMethod": "Dynamic",
    "publicIpPrefix": null,
    "resourceGroup": "myResourceGroup-CQN79NUGFU",
    "resourceGuid": "9b20f746-80fe-4464-a643-0117c78e5b45",
    "sku": {
```


The dynamic allocation method means that an IP address will be assigned when the public IP is associated with a resource. Static allocation can be used to reserve an address that won't change when it is not associated with a resource, for a cost. Currently the public IP is not associated so the **ipAddress** is **null**.

- [ ] 3. Read through the available configuration options for creating an Application Gateway and consider what options are needed to meet the requirements, pressing *spacebar* to advance the output:

```
az network application-gateway create -h
```

```
Command
    az network application-gateway create : Create an application gateway.

Arguments
    --name -n                   [Required] : Name of the application gateway.
    --resource-group -g         [Required] : Name of resource group. You can configure the default
                                             group using `az configure --defaults group=<name>`.
    --custom-error-pages                   : Space-separated list of custom error pages in
                                             `STATUS_CODE=URL` format.
    --location -l                          : Location. Values from: `az account list-locations`. You
                                             can configure the default location using `az configure
                                             --defaults location=<location>`.  Config: westus2.
    --max-capacity                         : Upper bound on the number of application gateway
                                             instances.
    --min-capacity                         : Lower bound on the number of application gateway
                                             instances.
    --no-wait                              : Do not wait for the long-running operation to finish.
    --tags                                 : Space-separated tags: key[=value] [key[=value] ...]. Use
                                             '' to clear existing tags.
```

The **--cert-file** and **--cert-password** options can be used to configure SSL offloading. In the description for the **--sku** option, there are **Standard** and **WAF** SKUs:

The v2 tiers support availability zones. Availability zones are a good idea for production, along with a **--capacity** of at least two to ensure a failure of one Application Gateway instance can be tolerated. For this Lab, a single small instance is sufficient.

4. Create an Application Gateway the satisfies the requirements by running the following command:

```
az network application-gateway create --resource-group $resource_group --vnet-name app --subnet app-gw --name app-gw --capacity 1 --fror
```

The http-settings-port is the port that the Application Gateway uses to communicate with the backend pool. It also automatically sends probes on that port to monitor the health of backend pool instances, removing any that don't provide a healthy response to the probe.

> ⚠ Note that the --servers option is not used. Although you could specify a list of IP addresses it is better to separately configure the VMSS so that when the VMSS scales up or down, the Application Gateway will automatically be updated to include/exclude added/removed VMSS instances. You will do this in the next Lab Step.

> ⚠ The command takes around 17 minutes to complete. To avoid waiting at the command line, the --no-wait option is used. You will use the time to deploy a web application to the VMSS in the next Lab Step before configuring the VMSS as the Application Gateway's backend pool.

> ✓ **Summary**
>
> In this exercise you created several resources necessary to create an Application Gateway. The Application Gateway takes a long time to create.

Exercise 3: Using a VMSS Extension to Deploy a Web Application to the VMSS

> **Introduction**
>
> The VMSS is not running any web application yet. In this Lab Step, you will deploy a web application by running a custom script on the VMSS instances using a VMSS extension. Extensions are small applications that automate tasks on running instances. You will use a custom script extension that allows you to run an arbitrary script on the VMSS instances. The script you will use will install Docker and deploy an application that is packaged in a Docker image.

1. Run the following script to deploy a web application on the VMSS instances using an extension:

```
az vmss extension set --resource-group $resource_group --vmss-name website-vmss --publisher Microsoft.Azure.Extensions --version 2.0 --r
```

Once the command finishes in a few minutes, it outputs the JSON specification of the VMSS. It includes the extension in the **virtualMachineProfile.extensionProfile** property.

The deploy script started a web application on the instances on port 80. The content of the script is below:

```
#!/usr/bin/env bash\
# Install and start Docker\
yum install -y yum-utils device-mapper-persistent-data lvm2\
yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo\
yum install -y docker-ce-18.06.1.ce-3.el7\
systemctl enable docker\
systemctl start docker\
# Run the web application and set the hostname environment variable\
docker run -e hostname=$(hostname) -p 80:80 --restart always -d lrakai/tetris:hostname
```

You don't need to understand the details. Just know that Docker is being installed and a web application is being run using the `docker run` command on the last line.

Summary

In this exercise you used a VMSS extension to deploy a web application to the instances. The instances are not accessible via the internet so you cannot access the application yet. In the next Lab Step, you will configure the VMSS to be in the backend pool of the Application Gateway. You will then be ablel to access the VMSS through the Application Gateway and have traffic load balanced across VMSS instances.

Exercise 4: Configuring the VMSS as the Application Gateway Backend Pool

Introduction

The Application Gateway currently has an empty backend pool. To access the web application through the Application Gateway you need to configure the VMSS to be in the backend pool. You will do that in this Lab Step. The configuration change actually needs to take place on the VMSS rather than the Application Gateway. This makes sense considering the VMSS knows when it scales up or down and can keep the backend pool up to date.

Note: You should wait until the Application Gateway has finished creating before moving on to the instructions of this Lab Step. You can wait until the Application Gateway provisioning completes by issuing

- [ ] 1. Run the following command:

```
az network application-gateway wait --resource-group $resource_group --name app-gw --created
```

```
\ Waiting ..
```

The command prompt will be returned to you when the provisioning completes. While you wait, it could be a good opportunity to explore the metrics and settings of the VMSS and the Application Gateway in the Azure Portal.

- [ ] 2. Inspect the network interface configuration of the VMSS by running the following command:

```
az vmss show --resource-group $resource_group --name website-vmss --query virtualMachineProfile.networkProfile.networkInterfaceConfigura
```


```
[
  {
    "dnsSettings": {
      "dnsServers": []
    },
    "enableAcceleratedNetworking": false,
    "enableIpForwarding": false,
    "id": null,
    "ipConfigurations": [
      {
        "applicationGatewayBackendAddressPools": null,
        "applicationSecurityGroups": null,
        "id": null,
        "loadBalancerBackendAddressPools": null,
        "loadBalancerInboundNatPools": null,
        "name": "website-vmssIpConfig",
        "primary": null,
        "privateIpAddressVersion": "IPv4",
        "publicIpAddressConfiguration": null,
        "subnet": {
          "id": "/subscriptions/93fe8ebb-c882-4947-b060-acde
providers/Microsoft.Network/virtualNetworks/app/subnets/web"
          "resourceGroup": "myResourceGroup-CQN79NUGFU"
        }
```


The main properties to note are **applictionGatewayBackendAddressPools** and **loadBalancerBackendAddressPools** which are used to configure the VMSS in a backend pool for an Application Gateway or an Azure Load Balancer, respectively. As the property names suggest, there can be more than one of each type of backend pool. Because of this, the property values are lists. The elements in the lists are objects containing the ID of the backend pool.

- [ ] 3. Review the properties of the Application Gateway focusing on relevant backend address pool values:

```
az network application-gateway show --resource-group $resource_group --name app-gw | more
```

The **backendAddressPools**[0].**id** is what you need to configure the VMSS backend pool.

- [ ] 4. Extract the backend pool ID and store it in a variable:

```
backend_pool_id=$(az network application-gateway show --resource-group $resource_group --name app-gw --query backendAddressPools[0].id)
```

Now that you have the ID, it's a good time to review the different options available for updating with the Azure CLI. With the Azure CLI, when you need to update a property you can use the --set option with the following pattern --set property=value. The value can be an entire list when the property is a list type. For changes involving lists, you can also use --add and --remove to modify the list elements. If the elements are JSON objects, you specify the JSON string as the parameter to the --add or --remove option. In the case of updating the VMSS backend pool property, you could use either set or add because there is no risk of overwriting other pools when the value is currently null.

- [ ] 5. Update the VMSS backend pool configuration to reference the Application Gateway's default backend pool by ID:

```
az vmss update --resource-group $resource_group --name website-vmss --add virtualMachineProfile.networkProfile.networkInterfaceConfigura
```

The command uses --add to add an element to the (empty) list. The element is a JSON object with only an id property. The pattern for --add and --remove updates is --add/--remove property value.

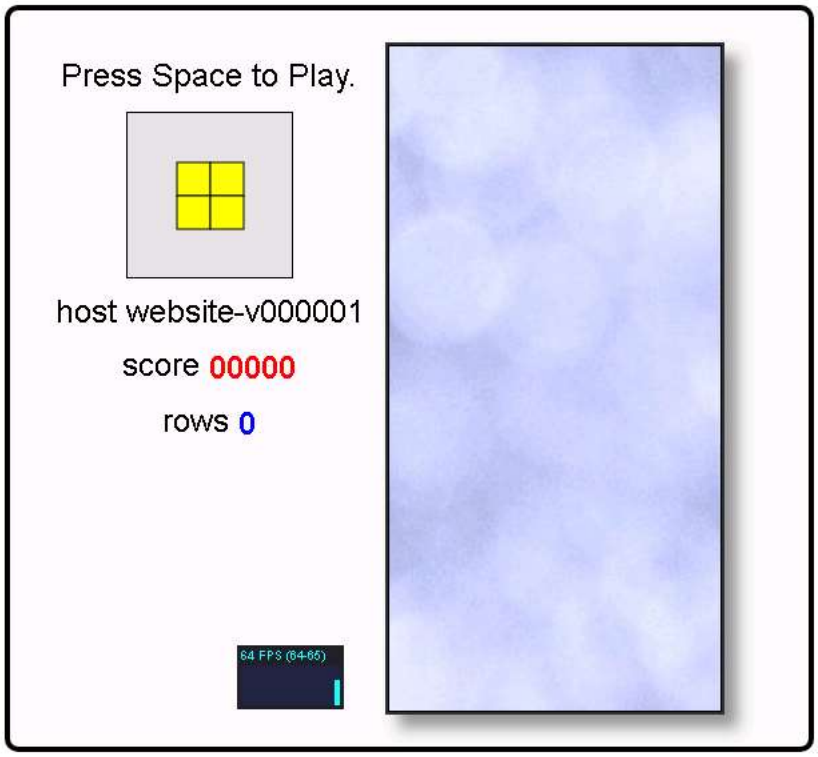
The command outputs the latest JSON specification for the VMSS. You can verify that the **applicationGatewayBackendAddressPools** field is a list with the object you just added.

- [ ] 6. Get the public IP address associated with the Application Gateway frontend:

```
az network public-ip show --resource-group $resource_group --name app-gw-ip --query ipAddress --output tsv
```

The output is the IP address.

- [ ] 7. Open a new web browser tab and navigate to the IP address:



The web application is displayed. Press spacebar if you'd like to test it out. The **host** line gives the name of the VMSS instance that is serving your HTTP request. In the image above the hostname is **website-v000001**.

8. Refresh your browser until you see the **host website-v** line change:

This confirms that the Application Gateway is load balancing traffic to the VMSS instances that are serving the web application.

> ⚠ **Note**: Your browser may cache the application for short periods causing the same host to appear. If you don't observe a change in the **host** after refreshing several times, you can try a different browser or private/incognito mode of your browser and refreshing with a few seconds delay between refreshes.

> ✓ **Summary**
>
> In this Lab Step, you configured the Application Gateway backend pool property of the VMSS to refer to the Application Gateway's default backend pool. This completed the requirements to enable the Application Gateway to load balance application traffic served by the VMSS.

> ⚠ **Note**: It is possible to create a VMSS and Application Gateway in a single step. However, it is instructive to configure the backend pool after the VMSS is created to understand more about how it works and how to use the Azure CLI.