



Microsoft Azure Compute

The Definitive Guide



Avinash Valiramani

Microsoft Azure Compute The Definitive Guide

Avinash Valiramani

USING AZURE POWERSHELL

When using Azure PowerShell to create a scale set, you add the `-Zone` parameter to enable the availability zones for that scale set. For example, the following code shows how to use this parameter in a PowerShell script for VMSS creation. Add the number of zones based on your scale-set design needs:

```
#Define variables
$RG = "VMSSResourceGroup"
$location = "EastUS2"
$VMSSname = "VMSScaleSet01"
$vnet = "VMSS-Vnet"
$subnet = "VMSS-Subnet"
$publicIP = "VMSSPublicIPAddress"
# Create VMSS in availability zone
New-AzVmss `
  -ResourceGroupName $RG""`
  -Location ""$location `
  -VMSScaleSetName ""$VMSSname `
  -VirtualNetworkName ""$vnet `
  -SubnetName ""$subnet `
  -PublicIpAddressName ""$publicip `
  -LoadBalancerName "VMSSLoadBalancer" `
  -UpgradePolicy "Automatic" `
  -Zone "1", "2", "3"
```

USING THE AZURE CLI

When using the Azure CLI to create a scale set, you add the `--zones` parameter to enable the availability zones for that scale set. For example, the following code shows how to use this parameter in a Bash script or prompt for VMSS creation. Add the number of zones based on your scale-set design needs:

```
#Define variables
rg="VMSS-RG01"
name="VMSS-Set01"
adminusername="vmssadminuser"
# Create vmss in availability zones
az vmss create --resource-group $rg\
  --name $name \
  --image UbuntuLTS \
  --upgrade-policy-mode automatic \
  --admin-username $adminusername\
  --generate-ssh-keys \
  --zones 1 2 3
```

Fault domains

In Azure regions with no availability zones, VMSS are created with five fault domains by default. For Azure regions that do support zonal deployments, the fault domain count is one. This implies that the VM instances will be split across as many racks as possible. You must set the number of fault domains when creating a scale set, so you'll want to think about this beforehand to build for maximum resiliency.

TIP It is recommended to set the number of fault domains equivalent to the number of managed disks to prevent loss of quorum if a managed disk fault domain goes down.

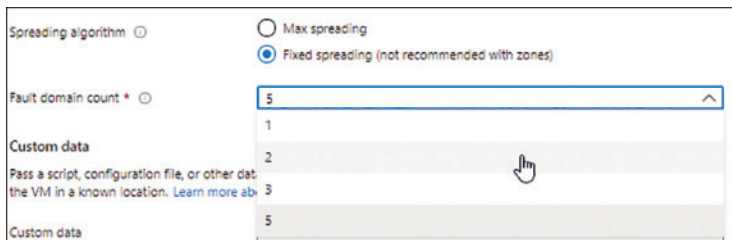
Fault domain walkthrough

The following sections step you through the process of setting up fault domains for a VMSS using the Azure Portal, Azure PowerShell, and the Azure CLI. If you are following along, be sure to select resources and resource names based on your environment, including unique VMSS, vNET, and subnet names for each of your deployments. Also, be sure to delete any unwanted resources after you have completed testing to avoid charges levied by Microsoft for these resources.

USING THE AZURE PORTAL

To enable fault domains using the Azure Portal during VMSS creation, follow these steps:

1. Follow the steps in the "VMSS creation walkthrough" section to start the Create a Virtual Machine Scale Set wizard and create a VMSS.
2. In the **Advanced** tab, next to **Spreading Algorithm**, select one of the following options:
 - **Max Spreading** VMs are spread across as many fault domains as possible in each zone.
 - **Fixed Spreading** VMs are always spread across exactly five fault domains. Remember, if fewer than five fault domains are available, the scale set will fail to deploy.
3. If you chose **Fixed Spreading** in step 2, enter the number of fault domains you want to use in the **Fault Domain Count** drop-down list. (See Figure 2-18.)



The screenshot shows the 'Advanced' tab of the VMSS creation wizard. Under 'Spreading algorithm', 'Fixed spreading (not recommended with zones)' is selected. The 'Fault domain count' dropdown menu is open, showing a list of options: 1, 2, 3, and 5. The option '5' is currently selected and highlighted. Below the dropdown, there is a 'Custom data' section with a text area and a 'Learn more about' link.

FIGURE 2-18 Set the fault domain count.

4. Finish using the VM Scale Set Creation wizard to create the VMSS, as described in the section “VMSS creation walkthrough.”

USING AZURE POWERSHELL

When using Azure PowerShell to create a scale set, you add the `-PlatformFaultDomainCount` parameter to set the number of fault domains to be used for the scale set. For example, the following code shows how to use this in a PowerShell script for VMSS creation. Add the number of fault domains to be used based on your scale-set redundancy needs, selecting between the currently supported values of 1, 2, 3, and 5.

```
#Define variables
$RG = "VMSS-RG01"
$location = "EastUS2"
$VMSSName = "VMScaleSet01"
$vnet = "VMSS-Vnet01"
$subnet = "VMSS-Subnet01"
$publicip = "VMSSPublicIPAddress"
#Create VMSS with fault domain config
New-AzVmss `
  -ResourceGroupName "$RG" `
  -Location "$location" `
  -VMScaleSetName "$VMSSName" `
  -VirtualNetworkName "$vnet" `
  -SubnetName "$subnet" `
  -PublicIpAddressName "$publicip" `
  -LoadBalancerName "VMSSLoadBalancer" `
  -UpgradePolicy "Automatic" `
  -Zone "1", "2", "3"
  -PlatformFaultDomainCount 3
```

USING THE AZURE CLI

When using the Azure CLI to create a scale set, you add the `--platform-fault-domain-count` parameter to set the number of fault domains for the scale set. For example, the following code shows how to use this in a Bash script for VMSS creation. Add the number of fault domains to be used based on your scale-set redundancy needs, selecting between the currently supported values of 1, 2, 3, and 5:

```
#Define variables
rg="VMSS-RG01"
$location = "EastUS2"
$VMSSName = "VMScaleSet01"
$vnet = "VMSS-Vnet01"
#Create VMSS with fault domain config
az vmss create \
```

```
--resource-group $rg \  
--name myScaleSet \  
--image UbuntuLTS \  
--upgrade-policy-mode automatic \  
--admin-username azureuser \  
--platform-fault-domain-count 3\  
--generate-ssh-keys
```

Autoscaling

Now that we've covered availability and resiliency options, and highlighted a number of occasions on how autoscaling relates to scale set availability and resiliency, let's examine how autoscaling works. In this context, autoscaling describes the ability of an Azure VMSS to automatically manage VM requirements based on various rules and resource-utilization patterns. In other words, the scale set can increase or decrease the number of VM instances running the application based on the rules defined in the scale set. As VM instances are added to a scale set, they are automatically added to the load balancer, too, so traffic can be distributed to them.

Autoscaling significantly reduces the overhead associated with managing application workload capacity requirements. Indeed, it completely eliminates the need to constantly check VM utilization and decide to scale in or out. This not only reduces management overhead, but it also makes the entire process more accurate than if it were manually managed.

You define a scale set's autoscaling rules manually. First, though, you must establish what you consider to be acceptable performance, based on application requirements and customer experience testing. After you identify these thresholds, you can put in place autoscaling rules to adjust the scale set's resource capacity accordingly.

A good approach is to base rules on the percentage of VM CPU utilization within a specific period of time. For example, suppose you know from user experience testing that when CPU utilization hits 85%, performance starts to suffer. In that case, you might set an autoscaling rule to increase the number of VM instances if CPU utilization reaches 80% for a period of 15 minutes to avoid performance degradation. Conversely, you might set an autoscaling rule to decrease the number of VM instances if CPU utilization drops below 30% to avoid paying for VM instances you aren't using.

NOTE You might select a higher or lower threshold of CPU usage or higher or lower time thresholds based on the environment and your knowledge of demand requirements.

You might also create autoscaling rules based on scheduled or known triggers or activities that would require a much higher or lower capacity requirement. For example:

- If business activity typically starts at 9 a.m. Monday through Friday, a burst of users will likely log in to the system at that time. In that case, it would be a good idea to create an

autoscaling rule to add VM instances just before that time so they are available before users begin to log in. On the other hand, if business activity tends to drop sharply after 7 p.m. and on weekends, you can create an autoscaling rule to reduce the number of VM instances available during that time.

- If your business employs an enterprise resource planning (ERP) application, it likely sees increased usage at the end of the month or quarter. Knowing this, you could set an autoscaling rule to automatically increase the number of VM instances to accommodate this increased demand.
- If you are planning a marketing promotion, an event, or a large sale, you could schedule a one-time autoscaling rule before the activity or anticipated demand.

NOTE The biggest benefits of autoscaling, apart from reduced management overhead, are cost savings and improved application performance.

Host-based metrics

The easiest metrics to use for autoscaling rules are the built-in host metrics for the various VM instances. These provide details on CPU utilization, memory demand, and disk access for each VM instance. Visibility into these metrics does not require the use of additional agents or configuration. You can configure scaling-in or scaling-out actions based on these metrics when you create a scale set or add them later using the Azure Portal, Azure PowerShell, the Azure CLI, and ARM templates.

Azure Diagnostics Extension for VMs

To obtain more detailed performance metrics, you must deploy an agent on each VM instance. One such agent is the Azure Diagnostic Extension for VMs. Once deployed, this agent provides more in-depth performance metrics, which allow for more-informed decisions regarding autoscaling rules. For example, you can consider more granular metrics like `PercentageIdleTime` for CPU or `AverageReadTime` for disks instead of the basic CPU and disk utilization metrics available using host metrics. This can help in defining more accurate scaling policies that yield higher levels of performance and cost benefits.

Azure Application Insights

You can use Azure Application Insights to monitor app-level performance metrics. To use Application Insights, you must install an instrumentation package within the application to monitor the app and send telemetry data to Azure. Once this is in place, you can access advanced application metrics such as session details, application response times, page loading performance, and so on, which you can use to define autoscaling rules. This level of granularity allows for more insightful and more efficient autoscaling actions, based on actual customer experience rather than baseline parameters.

Advanced autoscaling

In addition to setting autoscaling rules based on performance metric thresholds, a specific schedule, or a combination of the two, you can set up advanced autoscaling rules using email and webhook notifications. This approach requires the creation of multiple scaling profiles that contain a combination of triggers and scaling actions. Based on the email and webhook notifications, different profiles and their associated actions could be triggered.

Best practices and general tips for autoscaling

Following are some best practices and general tips that relate to autoscaling:

- **Set maximum and minimum values appropriately** If the scaling logic is set such that the minimum = 4, the maximum = 4, and the current instance count is 4, no scaling action can occur. Therefore, an appropriate margin is required between the maximum and minimum instance counts, which are inclusive. Autoscale will scale between these two limits.
- **Autoscale resets any manual scaling actions** Manual scaling is temporary unless the autoscaling rules are set or modified in line with the manual scaling actions. If you manually set an instance count value above or below the maximum, the autoscale engine will automatically bring the scale set to the minimum or maximum threshold based on the deviation on its next run. For example, if the range is set to be between 2 and 7, and there is only one instance running, autoscale would automatically scale to two instances the next time it runs. Similarly, if the instance count is manually set to scale to nine instances, on the next autoscale run, it will scale it back to seven instances.
- **Modify autoscaling rules** You can easily modify autoscaling rules over time. So, for example, you might start with utilization rules and then gather performance metrics before deciding on scheduled rules unless you have historical data available.
- **Combine scaling-in and scaling-out rules** While it is possible to create autoscaling rules to take only a single action (scale in or scale out), based on specific performance thresholds in a profile, it is not ideal. The more optimal approach would be to combine rules such that scaling out takes place automatically when usage is high and scaling in occurs automatically when the usage is low.
- **Diagnostics metrics for scaling rules** There are different metrics for scaling, which include total, minimum, maximum, and average. Based on the environment in use, the most appropriate metric would differ. If you're not sure which metric is the right one, it's a good idea to start with the average metric; you can adjust it over time if needed.
- **Threshold management** When setting metric thresholds, account for the environment in which the scale set is running and expect to perform some amount of fine-tuning and adjusting over time based on that environment. Also ensure that metrics do not overlap, clash, or conflict, as this will result in confusion and potentially inaction. Be sure to test a scaling-in and scaling-out operation by performing stress testing on the scale set if possible. This will help validate that the rules work, and that you will only need to adjust thresholds over time based on actual performance.