100 XP

Configure a virtual machine scale set

7 minutes

If you need to handle a steady expansion of work over time, the best approach is to scale horizontally. But if the workload increases in complexity rather than in volume, and this complexity demands more of your resources, you might prefer to scale vertically.

When you scale horizontally, you add instances to your virtual machine scale set. In the shipping company scenario, horizontal scaling is a good way to handle the changing number of requests over time. Horizontal scaling adjusts the number of virtual machines that run the web application as the number of users changes. In this way, the system maintains an even response time, regardless of the current load.

In this unit, you'll learn how to scale a virtual machine scale set. You can scale manually by explicitly setting the number of virtual machine instances in the scale set. Or you can configure autoscaling by defining scale rules that trigger the allocation and deallocation of virtual machines. These scale rules determine when to scale the system by monitoring various performance metrics.

Manually scale virtual machine scale sets

You scale a virtual machine scale set manually by increasing or decreasing the instance count. You can do this task programmatically or in the Azure portal.

The following code uses the Azure CLI to change the number of instances in a virtual machine scale set:

bash Copy

```
az vmss scale \
    --name MyVMScaleSet \
    --resource-group MyResourceGroup \
    --new-capacity 6
```

Autoscale virtual machine scale sets

Manual scaling is useful in some circumstances. But in many situations, autoscaling is better. It lets the system control the number of instances in a scale set.

You can base the autoscale on:

- **Schedule**: Use this approach if you know you'll have an increased workload on a specified date or time period.
- **Metrics**: Adjust scaling by monitoring performance metrics associated with the scale set. When these metrics exceed a specified threshold, the scale set can automatically start new virtual machine instances. When the metrics indicate that the additional resources are no longer required, the scale set can stop any excess instances.

Define autoscale conditions, rules, and limits

Autoscaling is based on a set of scale conditions, rules, and limits. A scale condition combines time and a set of scale rules. If the current time falls within the period defined in the scale condition, the condition's scale rules are evaluated. The results of this evaluation determine whether to add or remove instances in the scale set. The scale condition also defines the limits of scaling for the maximum and minimum number of instances.

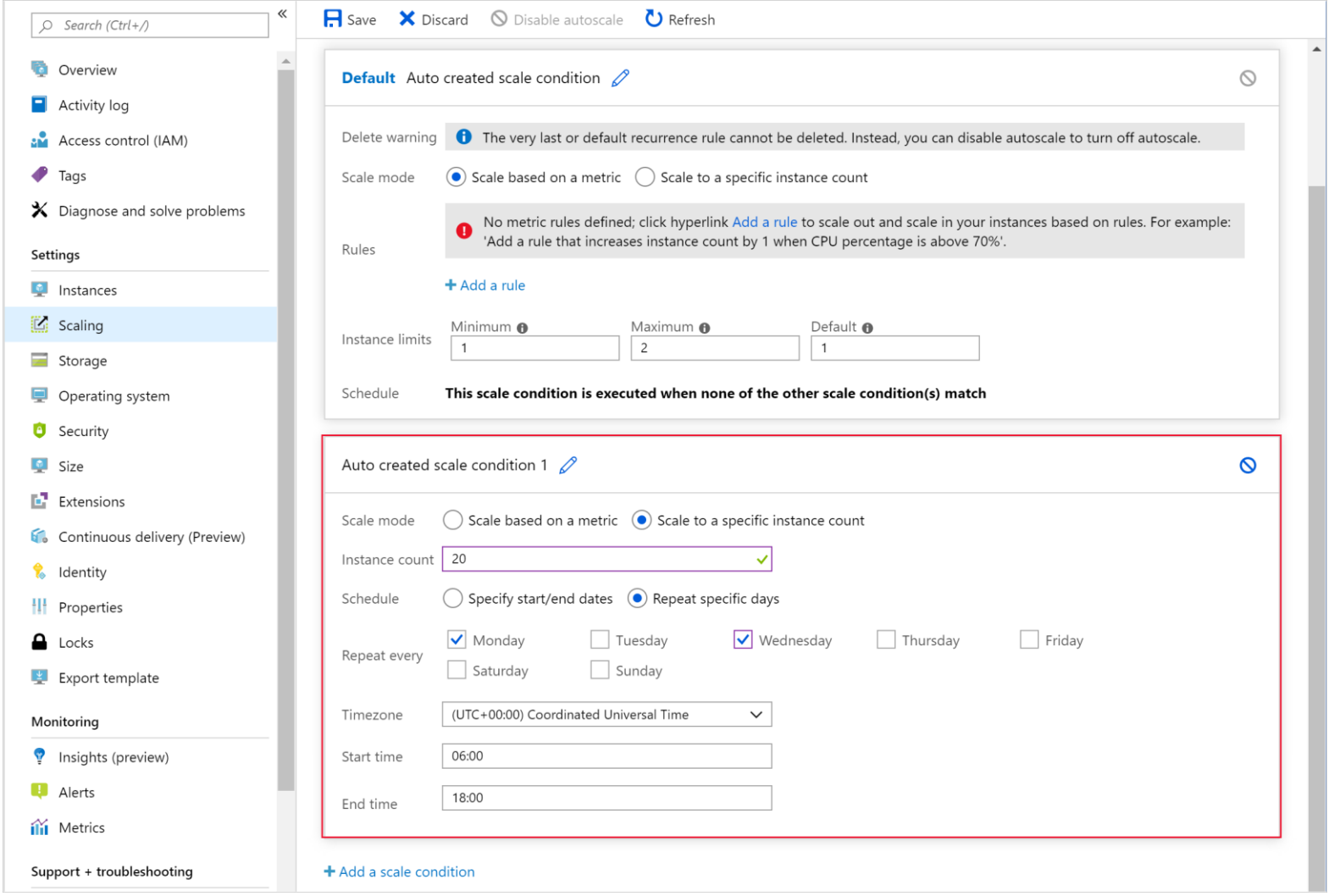
In the shipping company scenario, you can add scale rules that monitor the CPU usage across the scale set. If the CPU usage exceeds the 75 percent threshold, the scale rule can increase the number of virtual machine instances. A second scale rule can also monitor CPU usage but reduce the number of virtual machine instances when usage falls below 50 percent. Because the application is global, these rules should be active all the time rather than just at specific hours.

A virtual machine scale set can contain many scale conditions. Each matching scale condition is acted on. A scale set can also contain a default scale condition that's used if no other scale conditions match the current time and performance metrics. The default scale condition is always active. It contains no scale rules, effectively acting like a *null* scale condition that doesn't scale in or out. However, you can modify the default scale condition to set a default instance count. Or you can add a pair of scale rules that scale out and back in again.

Use schedule-based autoscaling

Schedule-based scaling specifies a start and end time, and the number of instances to add to the scale set. The following screenshot shows an example in the Azure portal. The number of instances is scaled out to 20 between 6 AM and 6 PM each Monday and Wednesday. Outside of these times, if there are no other scale conditions, the default scale condition is applied.

In this case, the default rule scales the system back down to two instances. This value is the **Maximum** in this default scale condition.



Use metrics-based autoscaling

A metrics-based scale rule specifies the resources to monitor, such as CPU usage or response time. This scale rule adds or removes instances from the scale set according to the values of these metrics. You specify limits on the number of instances to prevent a scale set from excessively scaling in or out.

In the example scenario, you want to increase the instance count by one when the average CPU usage exceeds 75 percent. Additionally, you want to limit the scale-out operation to 50 instances. This limit can help to prevent costly runaway scaling caused by an attack. Similarly, you want to scale in when the average CPU usage drops below 50 percent.

These metrics are commonly used to monitor a virtual machine scale set:

- **Percentage CPU** This metric indicates the CPU usage across all instances. A high value shows that instances are becoming CPU-bound, which could delay the processing of client requests.
- **Inbound flows and outbound flows** These metrics show how fast network traffic is flowing into and out of virtual machines in the scale set.
- **Disks read operations/sec and disk write operations/sec** These metrics show the volume of disk I/O across the scale set.
- **Data disk queue depth** This metric shows how many I/O requests to only the data disks on the virtual machines are waiting to be serviced.

A scale rule aggregates the values retrieved for a metric for all instances. It aggregates the values across a period known as the *time grain*. Each metric has an intrinsic time grain, but usually this period is one minute. The aggregated value is known as the *time aggregation*. The time-aggregation options are *average*, *minimum*, *maximum*, *total*, *last*, and *count*.

A one-minute interval is too short to determine whether any change in the metric is long-lasting enough to make autoscaling worthwhile. A scale rule takes a second step, further aggregating the time aggregation's value over a longer, user-specified period. This period is called the *duration*. The minimum duration is five minutes. For example, if the duration is set to 10 minutes, the scale rule aggregates the 10 values calculated for the time grain.

The duration's aggregation calculation can differ from the time grain's aggregation calculation. For example, let's say the time aggregation is *average* and the statistic gathered is *percentage CPU* across a one-minute time grain. So for every minute, the average CPU percentage usage across all instances during that minute will be calculated. If the time-grain statistic is set to *maximum* and the rule's duration is set to 10 minutes, the maximum of the 10 average values for the CPU usage percentage determines whether the rule threshold has been crossed.

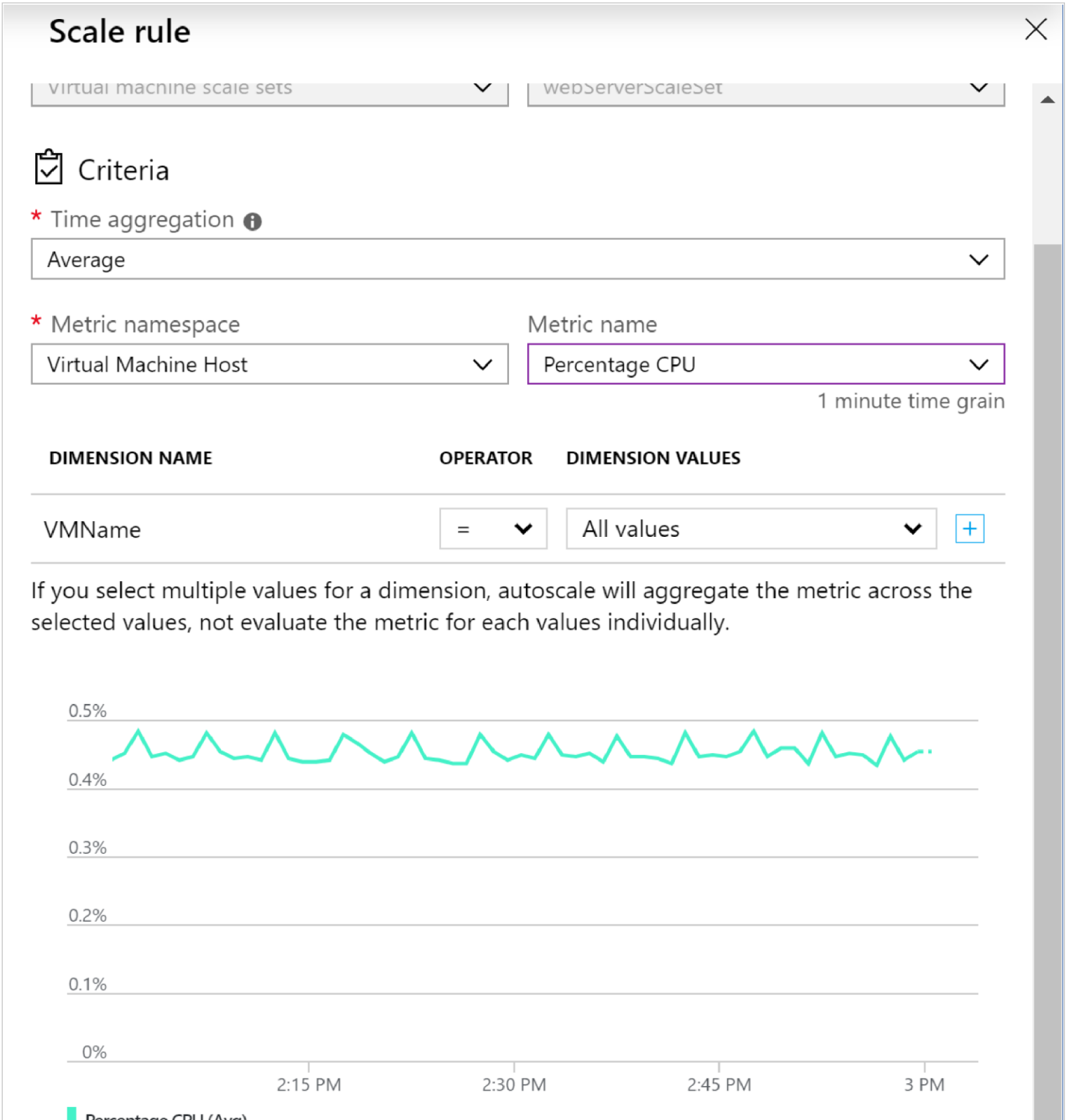
When a scale rule detects that a metric has crossed a threshold, it can do a scale action. A scale action can be *scale-out* or a *scale-in*. A scale-out action increases the number of instances. A scale-in action reduces the instance count.

A scale action uses an operator such as *less than*, *greater than*, or *equal to* to determine how to react to the threshold. Scale-out actions typically use the *greater than* operator to compare the metric value to the threshold. Scale-in actions tend to compare the metric value to the threshold by using the *less than* operator. A scale action also sets the instance count to a specific level rather than increasing or decreasing the number available.

A scale action has a *cool down* period, specified in minutes. During this period, the scale rule isn't triggered again. The cool-down allows the system to stabilize between scale events. Starting or shutting down instances takes time, so any metrics gathered might not show significant changes for several minutes. The minimum cool-down period is five minutes.

Finally, plan for a scale-in when a workload decreases. Consider defining scale rules in pairs in the same scale condition. One scale rule should indicate how to scale the system out when a metric exceeds an upper threshold. The other rule needs to define how to scale the system back in again when the same metric drops below a lower threshold. Don't make both threshold values the same. Otherwise you could trigger a series of oscillating events that scale out and back in again.

The following image shows a scale rule defined in the Azure portal.



Exercise - Configure a virtual machine scale set

10 minutes

Recall from the example scenario that your customers use one of the company's websites to manage and check the status of their shipments. This website is deployed to VMs and hosted on-premises.

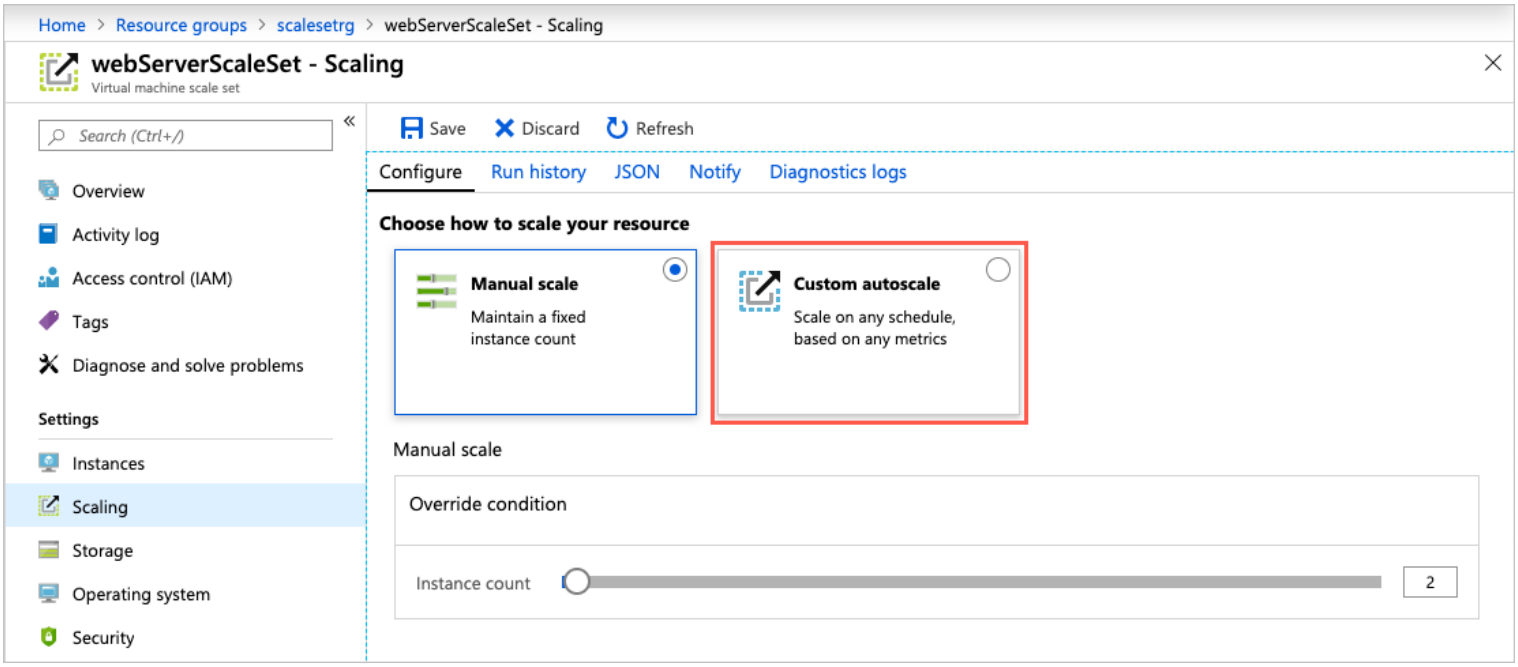
You notice that users of the website have significant delays in response times when the overall CPU usage of the VMs exceeds 75 percent. You need the virtual machine scale set that hosts your web application to scale horizontally when the system hits this threshold. To save costs, you also want to scale back in when demand falls and the overall CPU usage across the scale set drops below 50 percent.

In this exercise, you'll configure autoscaling. You'll define scale rules that scale out and in again, according to the system's CPU usage.

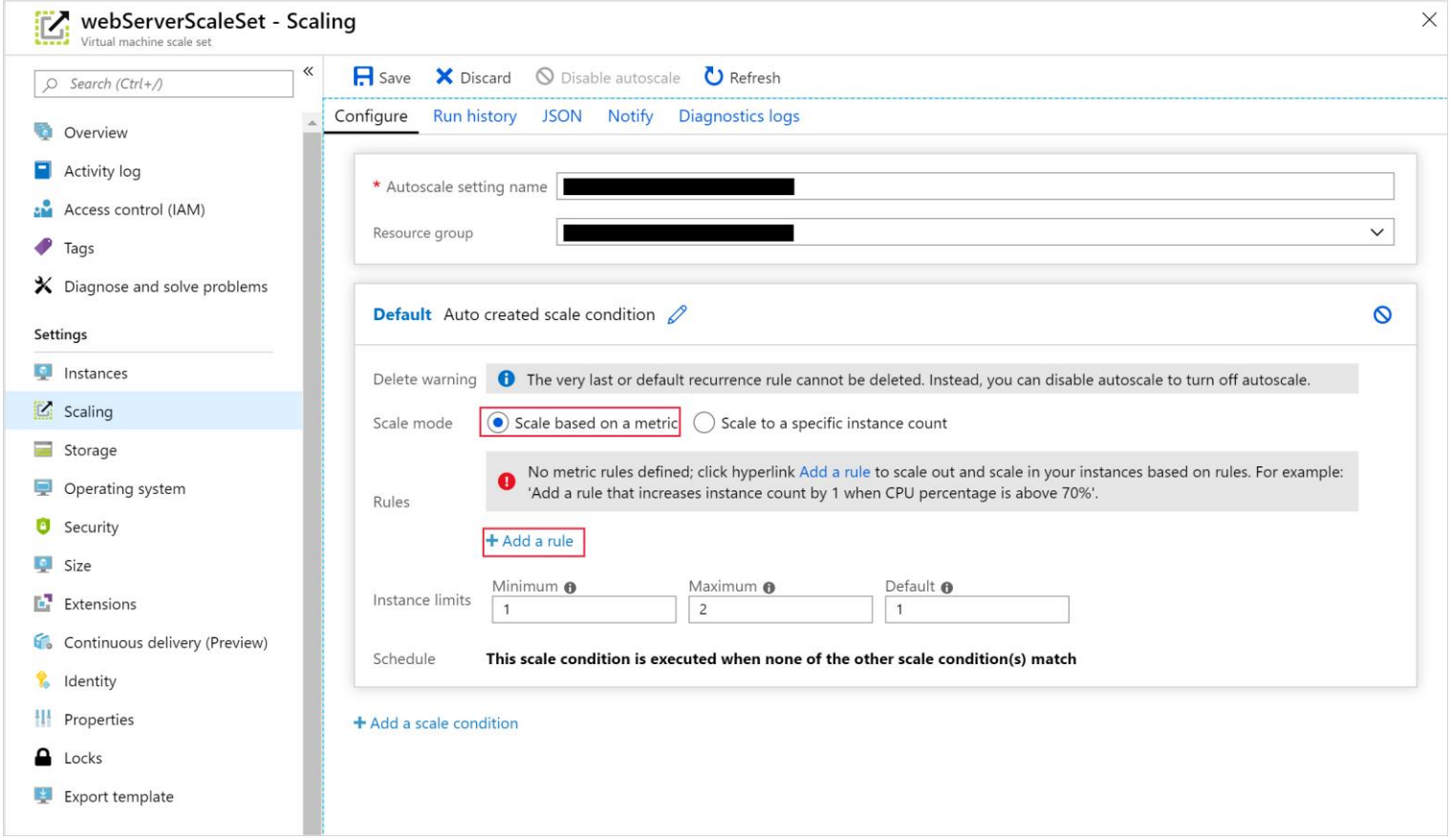
> **Note**
>
> This exercise is optional. If you don't have an Azure account, you can read through the instructions to understand how to use the REST API to retrieve metrics.
>
> If you want to complete this exercise but you don't have an Azure subscription or prefer not to use your account, create a free account before you begin.

Create a scale-out rule

1. In the Azure portal, go to the page for the virtual machine scale set.
2. On the virtual machine scale set page, under **Settings**, select **Scaling**.
3. Select **Custom autoscale**



4. In the **Default** scale rule, ensure that the **Scale mode** is set to **Scale based on a metric** Then select **+ Add a rule**.



5. On the **Scale rule** page, specify the following settings, and then select **Add**:

Property	Value
Metric source	Current resource (webServerScaleSet)
Time aggregation	Average
Metric name	Percentage CPU
Time grain statistic	Average
Operator	Greater than
Threshold	75
Duration	10
Operation	Increase count by
Instance count	1
Cool down (minutes)	5

Create a scale-in rule

1. In the **Default** scale rule, select **+ Add a rule**.

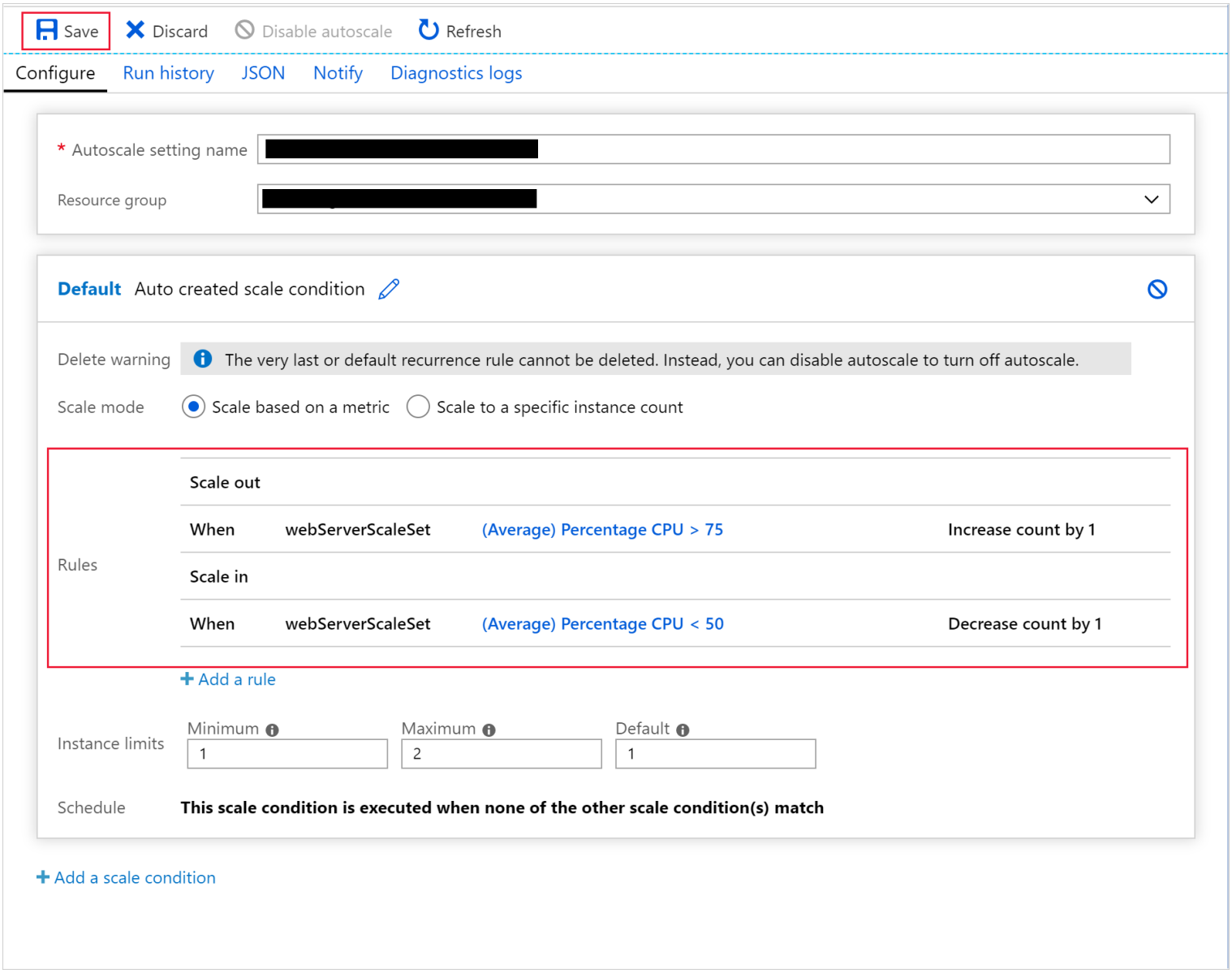
2. On the **Scale rule** page, specify the following settings, and then select **Add**:

Property	Value
Metric source	Current resource (webServerScaleSet)
Time aggregation	Average
Metric name	Percentage CPU

Property	Value
Time grain statistic	Average
Operator	Less than
Threshold	50
Duration	10
Operation	Decrease count by
Instance count	1
Cool down (minutes)	5

3. Select **Save**.

The **Default** scale condition now contains two scale rules. One rule scales the number of instances out. Another rule scales the number of instances back in.



Next unit: Install and update applications in virtual machine scale sets

Continue

100 XP

Exercise - Deploy a scale set in the Azure portal

10 minutes

In the example scenario, you've decided to use a scale set to run the web application for the shipping company. Using a scale set, the shipping company can maintain short response times for users as the workload varies.

Your first task is to create a scale set. You'll configure it to run a web server, in this case **nginx**. When you've configured the scale set correctly, you'll deploy your web application. Then you'll set up a health probe that Azure will use to verify the availability of each VM in the scale set. Finally, you'll test the scale set by sending requests from a web browser.

> **Note**
>
> This exercise is optional. If you don't have an Azure account, you can read through the instructions so you understand how to use the REST API to retrieve metrics.
>
> If you want to complete this exercise but you don't have an Azure subscription or prefer not to use your own account, create a **free account** before you begin.

Deploy a virtual machine scale set

1. Sign in to the Azure portal and open Azure Cloud Shell.
2. In Cloud Shell, start the code editor and create a file named *cloud-init.yaml*

```bash
code cloud-init.yaml
```

3. Add the following text to the file:

```yaml
#cloud-config
package_upgrade: true
packages:
  - nginx
write_files:
  - owner: www-data:www-data
  - path: /var/www/html/index.html
    content: |
        Hello world from Virtual Machine Scale Set !
runcmd:
  - service nginx restart
```

   This file contains configuration information to install nginx on the VMs in the scale set.

4. Press Ctrl+S to save the file. Then press Ctrl+Q to close the code editor.
5. Run the following command to create a new resource group named `scalesetrg` for your scale set:

```azure-cli
az group create \
  --location westus \
  --name scalesetrg
```

6. Run the following command to create the virtual machine scale set:

```azure-cli
az vmss create \
  --resource-group scalesetrg \
  --name webServerScaleSet \
```

```
  --image UbuntuLTS \
  --upgrade-policy-mode automatic \
  --custom-data cloud-init.yaml \
  --admin-username azureuser \
  --generate-ssh-keys
```

By default, the new virtual machine scale set has two instances and a load balancer.

> **Note**
>
> The `custom-data` flag specifies that the VM configuration should use the settings in the *cloud-init.yaml* file after the VM has been created. You can use a cloud-init file to install additional packages, configure security, and write to files when the machine is first installed.
>
> For more information, see **Cloud-init support for VMs in Azure**.

Configure the virtual machine scale set

1. Run the following command to add a health probe to the load balancer:

Azure CLI Copy

```
az network lb probe create \
  --lb-name webServerScaleSetLB \
  --resource-group scalesetrg \
  --name webServerHealth \
  --port 80 \
  --protocol Http \
  --path /
```

The health probe pings the root of the website through port 80. If the website doesn't respond, the server is considered unavailable. The load balancer won't route traffic to the server.

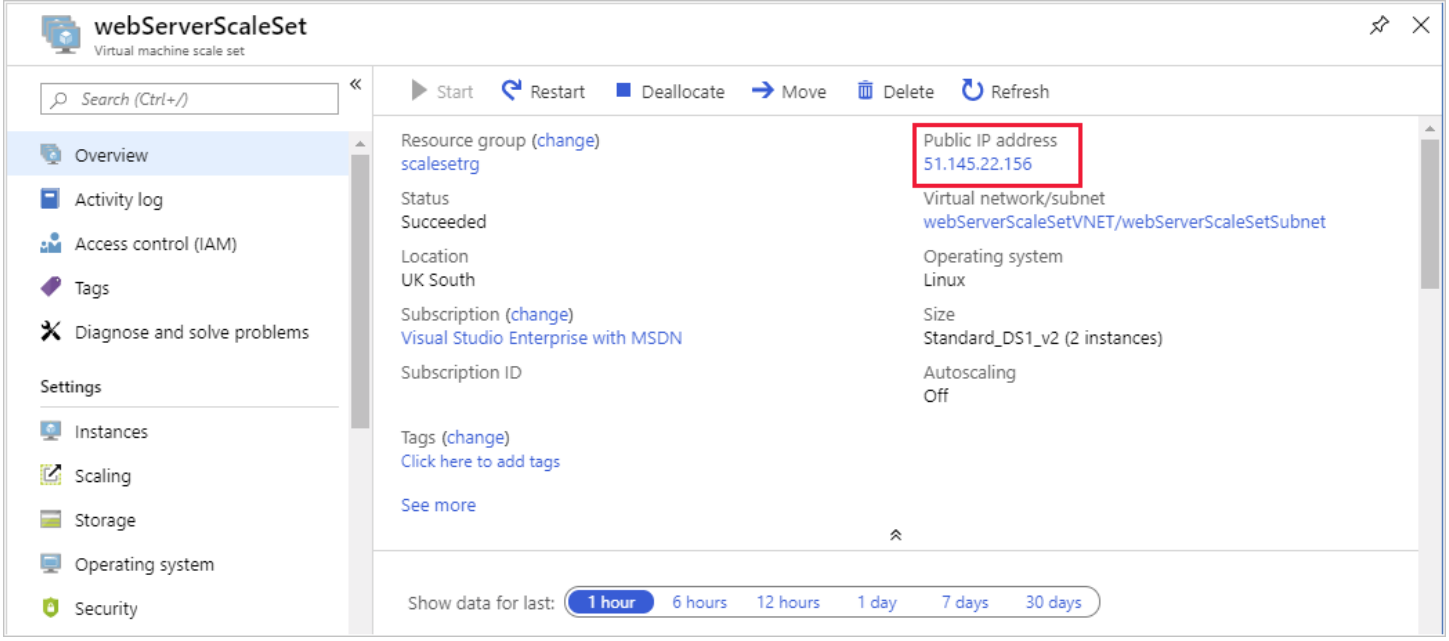
2. Run the following command to configure the load balancer to route HTTP traffic to the instances in the scale set:

Azure CLI Copy

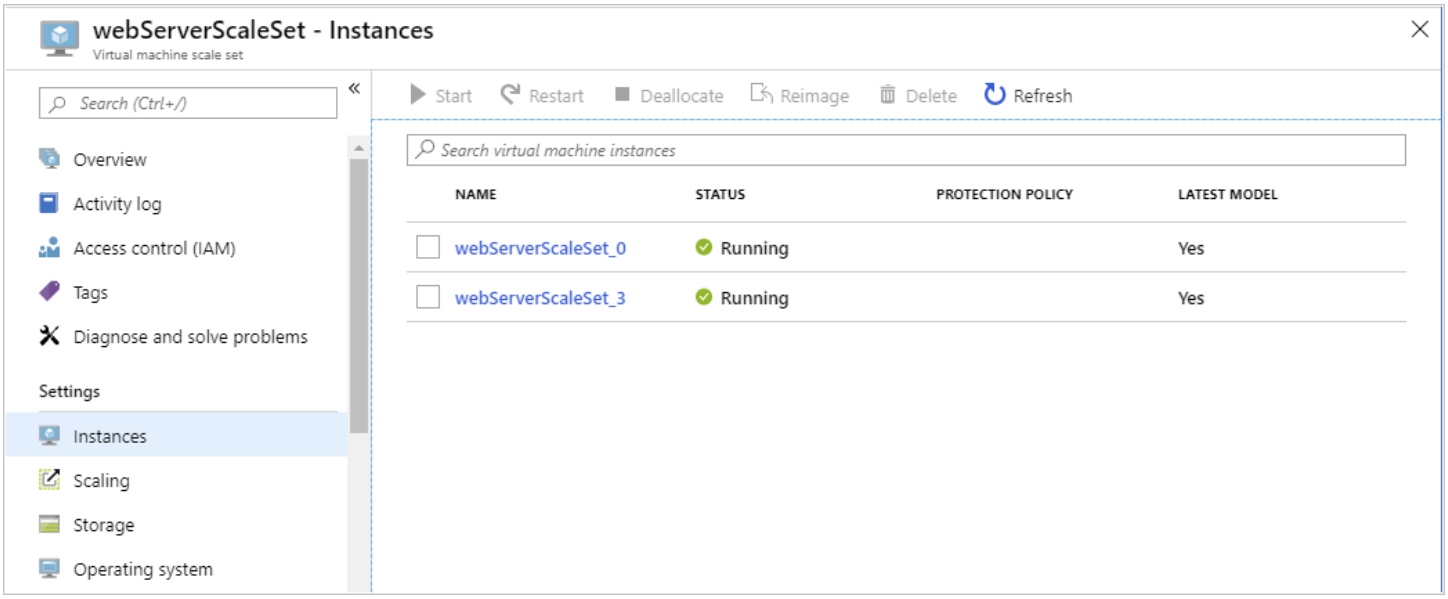
```
az network lb rule create \
  --resource-group scalesetrg \
  --name webServerLoadBalancerRuleWeb \
  --lb-name webServerScaleSetLB \
  --probe-name webServerHealth \
  --backend-pool-name webServerScaleSetLBBEPool \
  --backend-port 80 \
  --frontend-ip-name loadBalancerFrontEnd \
  --frontend-port 80 \
  --protocol tcp
```

Test the virtual machine scale set

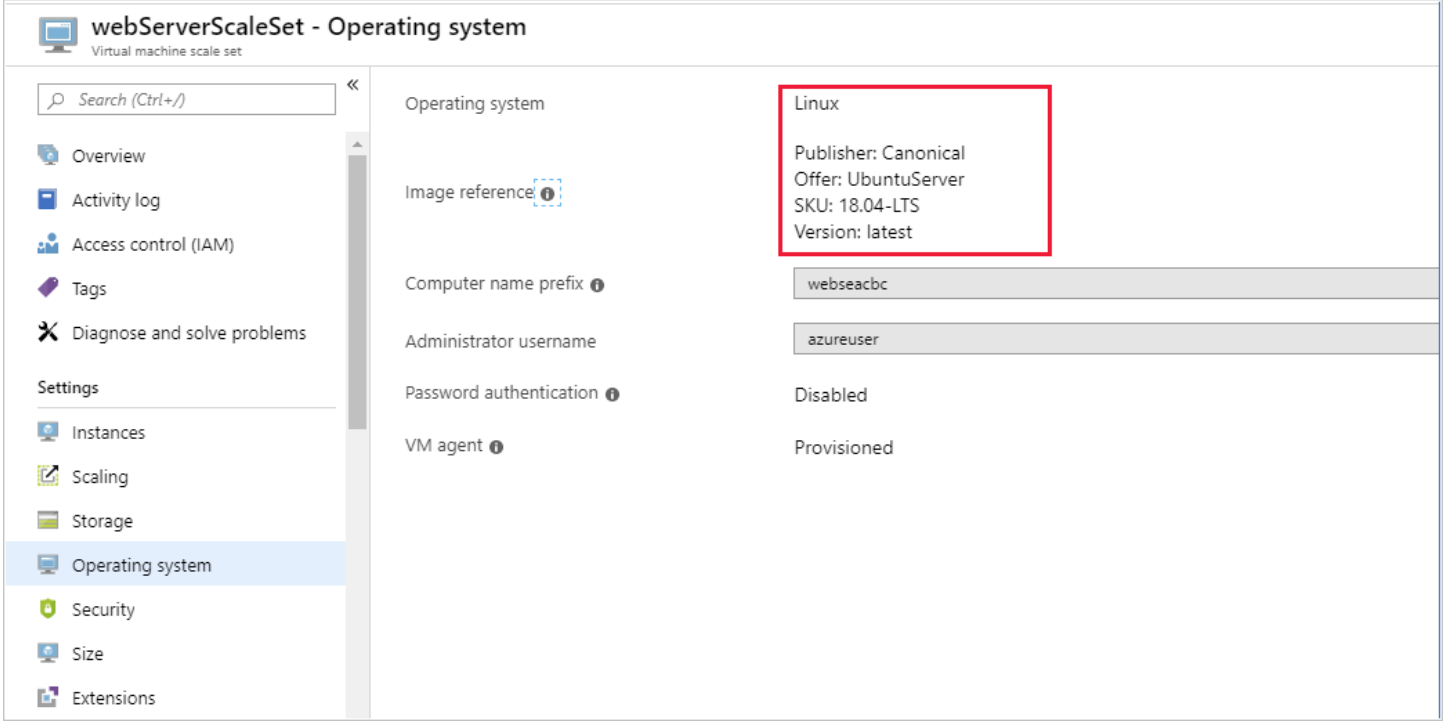
1. In the Azure portal, on the left, select **Resource groups** > **scalesetrg**.
2. Select the **webServerScaleSet** virtual machine scale set.
3. On the **Overview** page, note the public IP address of the virtual machine scale set.



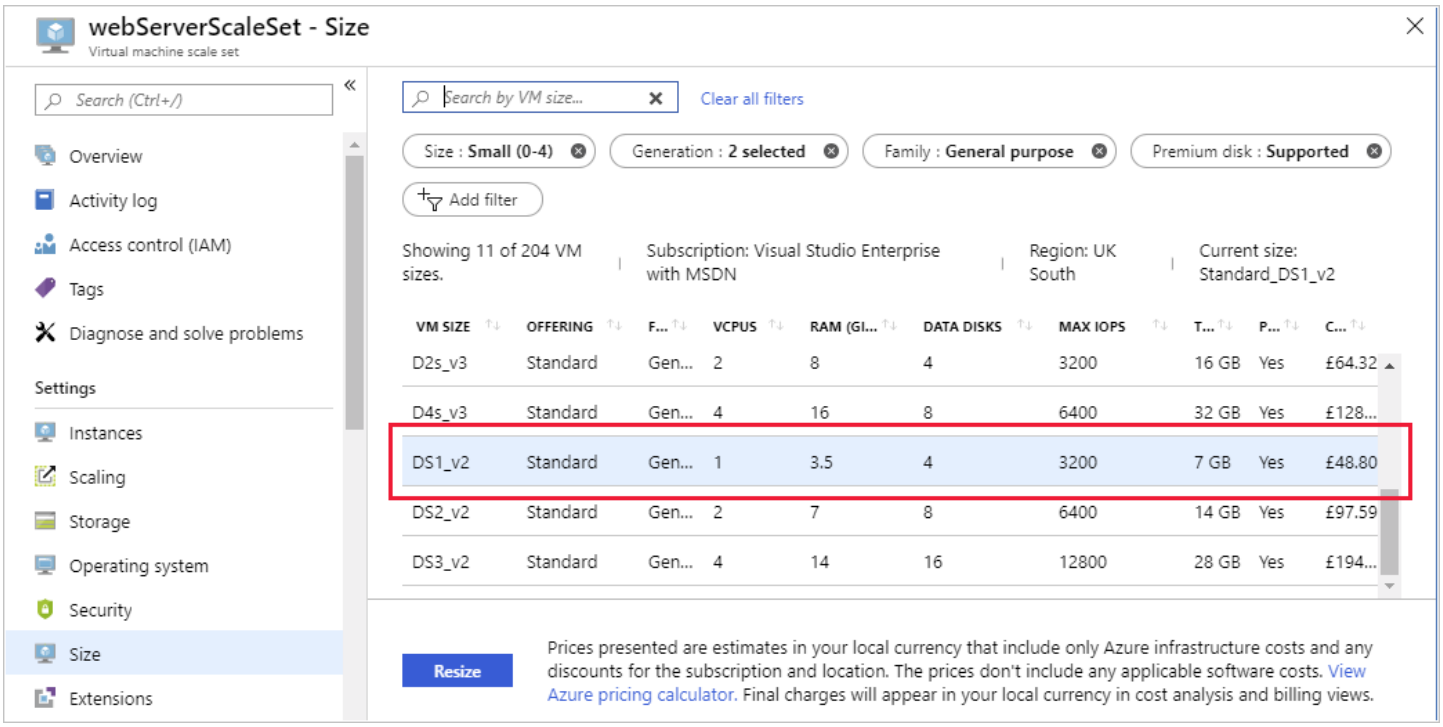
4. Under **Settings**, select **Instances**. Verify that the scale set contains two running VMs.



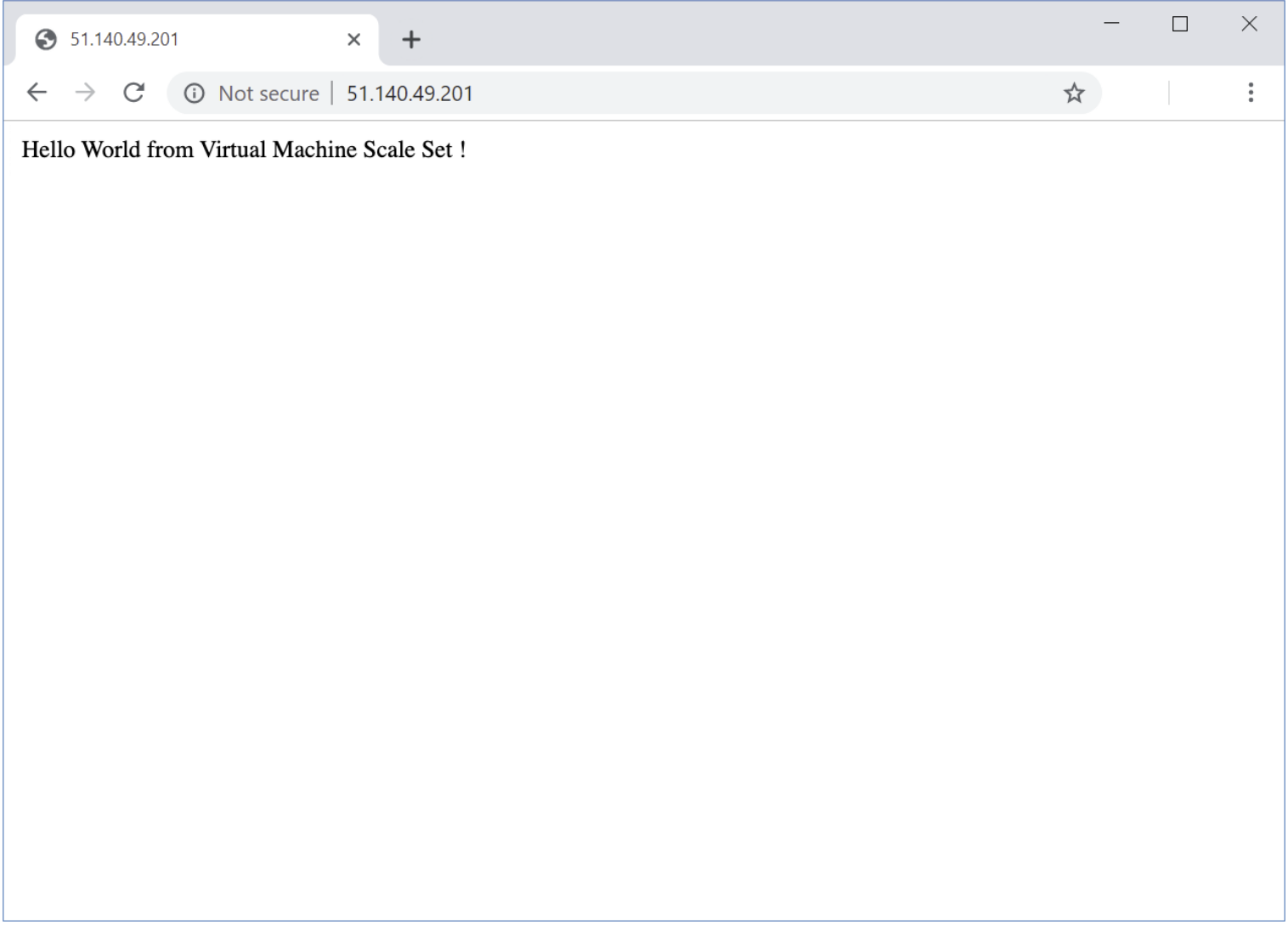
5. Select **Operating system** Verify that the VMs are running Ubuntu Linux.



6. Select **Size**. The VMs should be running on DS1_v2 hardware.



7. In your web browser, go to the public IP address of the scale set. Verify that the message `Hello World from Virtual Machine Scale Set !` appears.



Next unit: Configure a virtual machine scale set

Continue

100 XP

Exercise - Update applications in virtual machine scale sets

10 minutes

In the shipping company scenario, you installed a web application by creating the virtual machine scale set. You now need to update the web app and install a new version across all VMs in the scale set.

You must ensure that the system will remain available during the roll-out. A good way to ensure availability is to use a custom script extension to do the update. Apply this script across the virtual machine scale set. The scale set will apply the update to one VM at a time, leaving the other VMs up and running.

In this exercise, you'll use a custom script extension to roll out a new version of the web app. You'll edit the message that's provided by the nginx server. You can use the same approach for bigger updates.

> **Note**
>
> This exercise is optional. If you don't have an Azure account, you can read through the instructions to understand how to use the REST API to retrieve metrics.
>
> If you want to complete this exercise but you don't have an Azure subscription or prefer not to use your own account, create a free account before you begin.

Deploy the update by using a custom script extension

1. In the Azure portal, run the following command to view the current upgrade policy for the scale set:

```
Azure CLI
az vmss show \
    --name webServerScaleSet \
    --resource-group scalesetrg \
    --query upgradePolicy.mode
```

   Verify that the upgrade policy is set to `Automatic`. You specified this policy when you created the scale set in the first lab. If the policy were `Manual`, you would apply any VM changes by hand. Because the policy is `Automatic`, you can use the custom script extension and allow the scale set to do the update.

2. Run the following command to apply the update script:

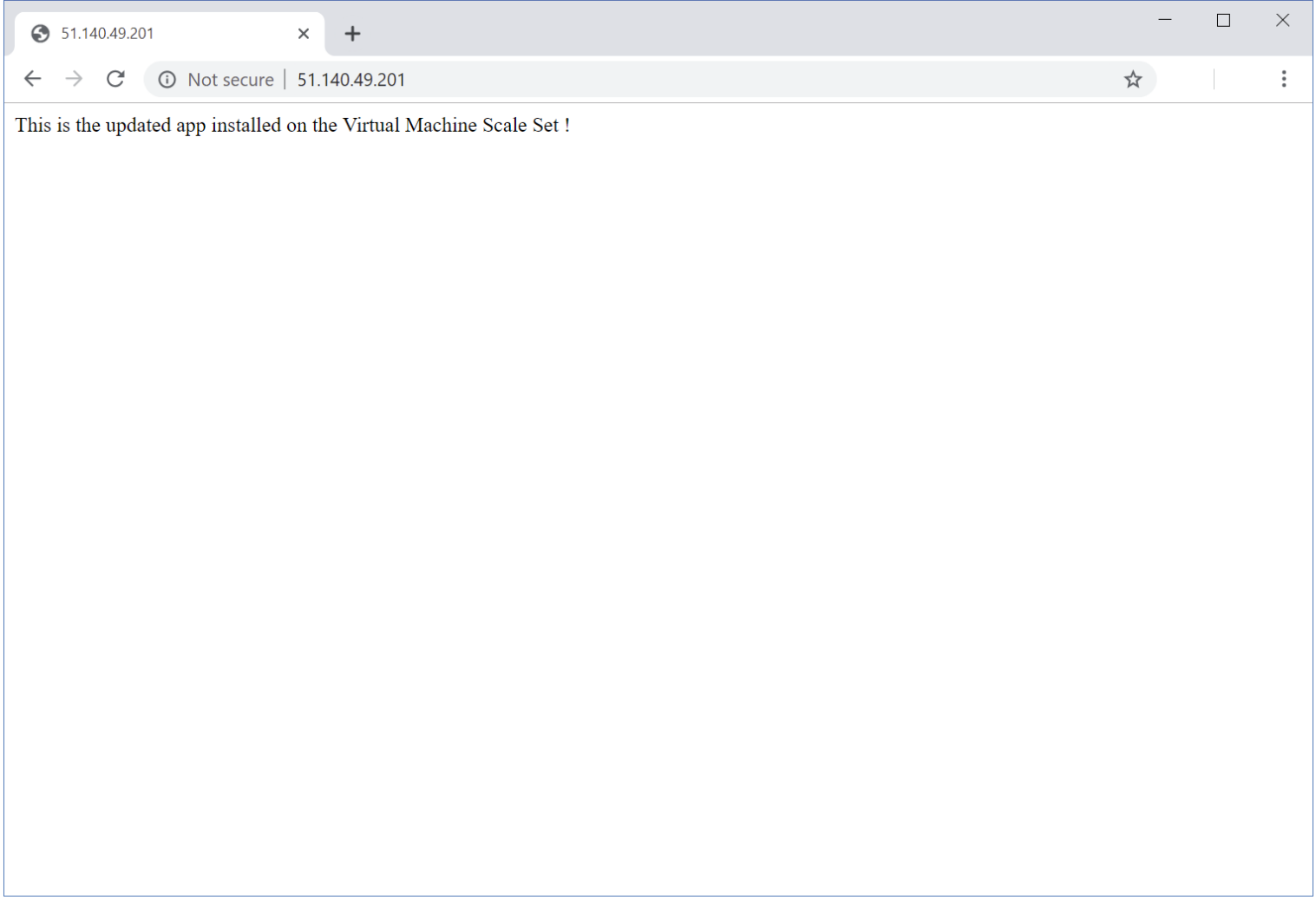
```
Azure CLI
az vmss extension set \
    --publisher Microsoft.Azure.Extensions \
    --version 2.0 \
    --name CustomScript \
    --vmss-name webServerScaleSet \
    --resource-group scalesetrg \
    --settings "{\"commandToExecute\": \"echo This is the updated app installed on the Virtual Machine Scale Set ! > /var/www/html/index.html\"}"
```

Test the updated web application

1. Run the following command to retrieve the IP address of the load balancer for the scale set:

```
Azure CLI
az network public-ip show \
    --name webServerScaleSetLBPublicIP \
    --resource-group scalesetrg \
    --output tsv \
    --query ipAddress
```

2. In your web browser, go to the public address of the scale set load balancer. Verify that you see the message **This is the updated app installed on the Virtual Machine Scale Set !**



Next unit: Summary

Continue

Features and benefits of virtual machine scale sets

7 minutes

Azure virtual machine scale sets provide a scalable way to run applications on a set of virtual machines (VMs). The VMs in this type of scale set all have the same configuration and run the same applications. As demand grows, the number of VMs running in the scale set increases. As demand slackens, excess VMs can be shut down. Virtual machine scale sets are ideal for scenarios that include compute workloads, big-data workloads, and container workloads.

In our example scenario, your customers use one of the company's websites to manage and check the status of their shipments. Because the website is accessed globally, the load is sometimes difficult to predict at any particular time of day. Additionally, loading might vary seasonally, with December being busy because of the holidays at the end of the year. You decide to use a virtual machine scale set to handle the fluctuating load while maintaining a low response time for customer requests.

In this unit, you'll explore the features of virtual machine scale sets. By the end of this unit, you'll be able to describe how a scale set works. You'll understand how a scale set supports scale-out and scale-up scenarios. You'll see how to use autoscaling and schedule-based scaling to adjust the resources available to a scale set.

What is a virtual machine scale set?

Virtual machine scale sets in Azure are designed to allow you to deploy and manage many load-balanced, identical VMs. These machines run with the same configurations. Virtual machine scale sets are intelligent enough to automatically scale up or down the number of VM instances. A scale set can also change the size of VM instances.

The criteria used to activate the upscale or downscale can depend on a customized schedule or actual demand and usage. Scale sets apply the same configuration to a group of VMs simultaneously. They don't require you to manually configure instances individually.

A scale set uses a load balancer to distribute requests across the VM instances. It uses a health probe to determine the availability of each instance. The health probe *pings* the instance. If the instance responds, the scale set knows the instance is still available. If the ping fails or times out, the scale set knows the instance is unavailable and doesn't send requests to it.

Virtual machine scale sets support both Linux and Windows VMs in Azure. However, keep in mind that you're limited to running 1,000 VMs on a single scale set.

If you deal with large workloads whose demand varies and is unpredictable, scale sets are a great solution. Because virtual machine scale sets offer identical VMs scaled and load-balanced in response to demand, they automatically provide a highly available environment.

Scaling options for scale sets

Scale sets are designed for cost-effectiveness. New VM instances are created only when needed. A scale set can scale VMs either horizontally or vertically.

What is horizontal scaling?

Horizontal scaling is the process of adding or removing several VMs in a scale set.

Sometimes you might need to add or remove machines in a scale set, depending on demand. For example, you might not need to run some machines during periods of the week or day when demand is low. You could manually adjust the number of VMs in a scale set by increasing or decreasing the instance count. But in many cases, it's better to automatically add or remove VMs by using rules. The rules are based on metrics. They ensure that the right number of VMs are added, depending on the demand or schedule.

What is vertical scaling?

Vertical scaling is the process of adding resources such as memory, CPU power, or disk space to VMs.

In contrast to horizontal scaling, where new, identically sized VMs are added to or removed from a scale set, vertical scaling focuses on increasing the size of the VMs in the scale set.

For example, you might want to reduce the CPU performance of a group of VMs in a scale set. In this case, you might not necessarily need to remove an entire group of machines. In scale sets, you create rules based on metrics. These rules automatically trigger an increase in the sizes of the VMs.

Vertical scaling typically requires rebooting the affected VMs in the scale set. This process can lead to temporary degraded performance across the scale set while the VMs restart.

Scaling a scale set

Virtual machine scale sets address the need to quickly create and manage VMs for a fluctuating workload. You can configure two types of scaling for a scale set:

- **Scheduled scaling** You can proactively schedule the scale set to deploy one or *N* number of additional instances to accommodate a spike in traffic and then scale back down when the spike ends.
- **Autoscaling** If the workload is variable and can't always be scheduled, you can use metric-based threshold scaling. Autoscaling horizontally scales out based on node usage. It then scales back in when the resources return to a baseline.

Both of these options address the requirement to scale while managing associated costs. The following examples describe scenarios where you might use different types of scaling.

Scheduled scaling

Suppose you're part of the DevOps team for a large food delivery company. Friday night is typically your busiest time. Conversely, 7 AM on Wednesday is generally your quietest time.

Azure charges based on the consumption of resources, so don't run services you don't need. If you need 100 web servers to meet your demand on a Friday night, you're happy to pay for them. But if you need only two servers on a Wednesday morning, you don't want to pay for the 98 idle servers. To manage your costs while fulfilling operational requirements, consider using scheduled scaling.

Autoscaling

Suppose you're on the DevOps team for a popular footwear company. As a product launch approaches, you think you'll see significant demand for your service. However, the demand spike might be unpredictable and hard to quantify. You want your service to meet demand by scaling horizontally as current resources are used.

For this scenario, use metrics-based autoscaling. This type of autoscaling scales out your infrastructure as demand rises. It scales back in when demand declines.

Reducing costs by using low-priority scale sets

A low-priority virtual machine scale set allows you to use Azure compute resources at cost savings of up to 80 percent. The global Azure infrastructure frequently has underused compute resources available. A low-priority scale set provisions VMs through this underused compute capability.

When you use these VMs, keep in mind that they're temporary. Availability depends on size, region, time of day, and so on. These VMs have no SLA.

When Azure needs the computing power again, you'll receive a notification about the VM that will be removed from your scale set. If you need to clean up or gracefully exit code on your VM, you can use Azure Scheduled Events to react to the notification within the VM. You can also make the scale set try to create another VM to replace the one that's being removed. The creation of the new VM is, however, not guaranteed.

In a low-priority scale set, you specify two kinds of removal:

- **Delete**: The entire VM is removed, including all of the underlying disks.
- **Deallocate**: The VM is stopped. The processing and memory resources are deallocated. Disks are left intact and data is kept. You're charged for the disk space while the VM isn't running.

Low-priority scale sets are useful for workloads that run with interruptions or when you need larger VMs at a much-reduced cost. Just keep in mind that you can't control when a VM might be removed.

Next unit: Exercise - Deploy a scale set in the Azure portal

Continue

100 XP

Install and update applications in virtual machine scale sets

7 minutes

When you deploy an application across a scale set, you need a mechanism that updates your application consistently, across all instances in the scale set. You achieve this outcome by using a custom script extension.

In the shipping company scenario, you need a quick way to roll out updates to the application while minimizing disruption to the end users. A custom script extension is an ideal solution.

In this unit, you'll learn how to use a custom script extension to update an application that runs on a scale set.

What is an Azure custom script extension?

An Azure custom script extension downloads and runs a script on an Azure VM. It can automate the same tasks on all the VMs in a scale set.

Store your custom scripts in Azure Storage or in GitHub. To add one to a VM, you can use the Azure portal. To run custom scripts as part of a templated deployment, combine a custom script extension with Azure Resource Manager templates.

Install an application across a scale set by using a custom script extension

To use a custom script extension with the Azure CLI, you create a configuration file that defines the files to get and the commands to run. This file is in JSON format.

The following example shows a custom script configuration that downloads an application from a repository in GitHub and installs it on a host instance by running a script named `custom_application_v1.sh`:

JSON

Copy

```
# yourConfigV 1.json
{
  "fileUris" : ["https://raw.githubusercontent.com/yourrepo/master/custom_application_v1.sh"   ],
  "commandToExecute" : "./custom_application_v1.sh"
}
```

To deploy this configuration on the scale set, you use a custom script extension. The following code shows how to create a custom script extension for a virtual machine scale set by using the Azure CLI. This command installs the new app on the VMs across the scale set:

PowerShell

Copy

```
az vmss extension set \
  --publisher Microsoft.Azure.Extensions \
  --version 2.0 \
  --name CustomScript \
  --resource-group myResourceGroup \
  --vmss-name yourScaleSet \
  --settings @yourConfigV1.json
```

Update an application across a scale set by using a custom script extension

You can use a custom script extension to update an existing app across a virtual machine scale set. You reference an updated deployment script and then reapply the extension to your scale set. For example, the following JSON code shows a configuration that fetches a new version of an application and installs it:

JSON

Copy

```
# yourConfigV 2.json
{
  "fileUris" : ["https://raw.githubusercontent.com/yourrepo/master/custom_application_v2.sh"   ],
  "commandToExecute" : "./custom_application_v2.sh"
}
```

Use the same `az vmss extension set` command shown previously to deploy the updated app. But this time, reference the new configuration file:

Summary

3 minutes

In this module, you looked at the features of virtual machine scale sets. You saw how to manually scale a scale set. You saw how to configure a scale set to autoscale based on a schedule or on performance metrics. You also learned how to use a custom extension script to deploy and update an application across the VMs in a scale set.

In the example scenario, you applied your knowledge to the shipping company's system. You deployed the website by using a scale set, and you configured the scale set to scale out and back in as the CPU usage across the VMs changed. You also rolled out a new version of the web application across the scale set while allowing the VMs to continue running. These actions allow users to access the system and maintain a good response time, even when the application is being updated.

Clean up

In Cloud Shell, run the following command to delete the `scalesetrg` resource group. This action also removes the scale set.

bash Copy

```
az group delete \
  --name scalesetrg \
  --yes
```

Learn more

- Overview of autoscale with Azure virtual machine scale set
- Deploy your application on virtual machine scale set
- Work with large virtual machine scale set
- Use the custom script extension for Windows
- Use the Azure custom script extension version 2 with Linux VMs
- Use the application health extension with virtual machine scale set

---

Module complete:

Unlock achievement