Exam note One of the objectives on the Exam 70-140: Installing and Configuring Windows Server 2012 is *configure Resource Metering*, and so if you are working on your new MCSE for *Server Infrastructure*, you will want to know this material. Of course, if you are running or thinking about running Hyper-V on Windows Server 2012, you will want to know this information as well.

So, the first thing I need to do is to check on the status of Resource Metering on my Hyper-V server. To do this, I use the **Get-VM** cmdlet from the Hyper-V module to return virtual machine objects, and I pipe these to the **Select-Object** cmdlet, and I choose the **name** and the **ResourceMeteringEnabled** properties. This command is shown here along with the associated output from the command.

Note Keep in mind that the **Get-VM** and other cmdlets from the Hyper-V require admin rights to run, and therefore, if you are doing this on Windows 8, you will need to start Windows PowerShell with admin rights.

```
14:34 C:\> get-vm | select name, resourcemeteringenabled

Name                                    ResourceMeteringEnabled

----                                    -----------------------

c1                                      False

C2                                      False

DC1                                     False

sql1                                    False
```

Enabling Resource Metering

Well, the next thing I need to do is to enable Resource Metering. To do this, I use the **Enable-VMResourceMetering** cmdlet. I can enable Resource Metering for a specific machine, such as for SQL1, by specifying the **–VMName** parameter. The command to do this is shown here (keep in mind that nothing returns from this command).

```
Enable-VMResourceMetering -VMName sql1
```

Exam ALERT The default parameter set for the **Enable-VMResourceMetering** is **VMName**, and therefore, you can simply supply the name of the virtual machine **Enable-VMResourceMetering** SQL1. But, there are three parameter sets: **VMName**, **VM**, and **ResourcePoolName**. To make matters worse, **ResourcePoolName** has an alias of **Name**. Keep in mind, **VMName** is the name of the virtual machine, **VM** is a virtual machine object (such as returned by **Get-VM**) and **ResourcePoolName** and **Name** are the names of virtual machine

resource pools. There is no alias for **VMName** or **VM**. The following illustrates available parameter aliases.

```
15:31 C:\> gcm Enable-VMResourceMetering | select -expand parameters | % {$_.values} |

? aliases | select name, aliases

Name                    Aliases

----                    -------

ResourcePoolName        {Name}

Verbose                 {vb}

Debug                   {db}

ErrorAction             {ea}

WarningAction           {wa}

ErrorVariable           {ev}

WarningVariable         {wv}

OutVariable             {ov}

OutBuffer               {ob}
```

If I want to, I can enable Resource Metering on all of the virtual machines by piping the results from **Get-VM** to the **Enable-VMResourceMetering** cmdlet, as shown here.

```
get-vm | Enable-VMResourceMetering
```

Getting the Resource Metering report

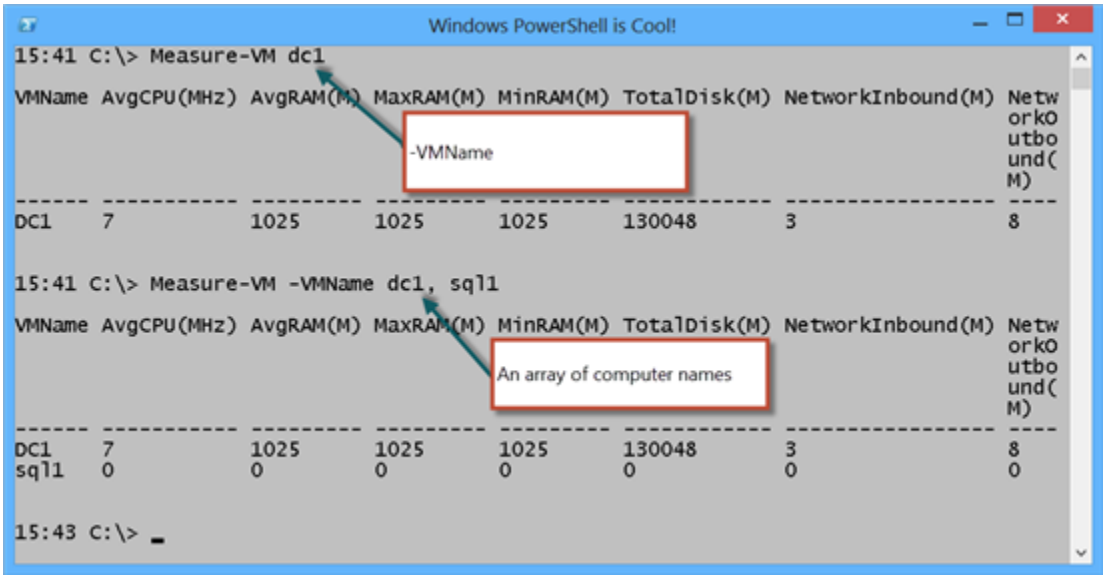
To get the report of metered resources, use the **Measure-VM** cmdlet. The default parameter set is **VMName**, and therefore, the parameter can be left out. The **VMName** parameter accepts an array, and therefore, I can obtain information from more than one virtual machine at a given time.

The following illustrates obtaining Resource Metering reports from one and from two virtual machines. The syntax is shown here.

Measure-VM dc1

Note When a virtual machine is offline, it reports 0.

The following image illustrates using this command, as well as the command to measure two virtual machines.

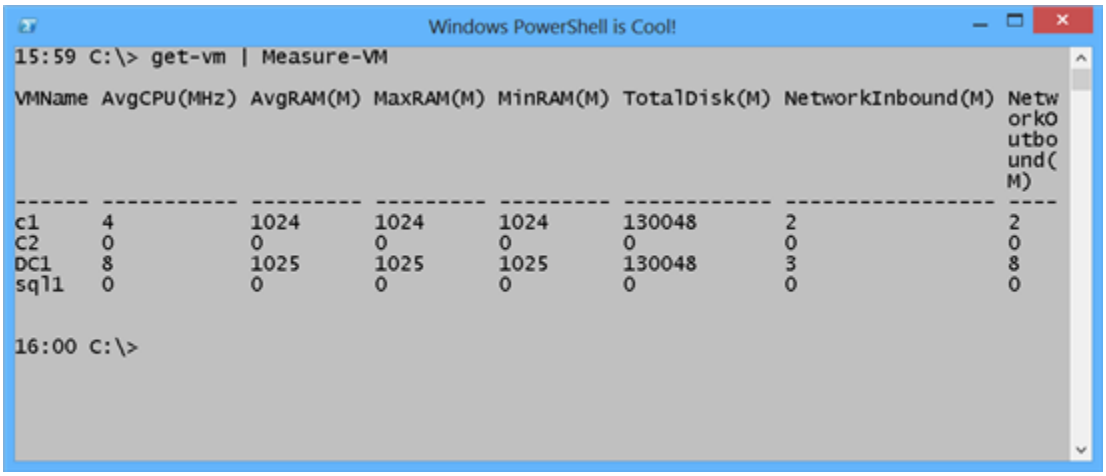


Getting a metering report for all virtual machines

I can use the **Get-VM** cmdlet and pipe the returned **VirtualMachine** objects to the **Measure-VM** cmdlet to obtain a report on all of my virtual machines. The command is shown here.

get-vm | Measure-VM

The command and its output associated used to measure all virtual machines are shown in the image that follows.



But realize this IS Windows PowerShell, and therefore, everything is an object. This means I am not limited to the output appearing above. I use the **Get-Member** cmdlet (**gm** is an alias) to see what my options are. The command and its output are shown here.

```
16:02 C:\> get-vm | Measure-VM | gm -MemberType *property

TypeName: Microsoft.HyperV.PowerShell.VMMeteringReportForVirtualMachine

Name                    MemberType    Definition

----                    ----------    ----------

AvgCPU                  AliasProperty AvgCPU = AverageProcessorUsage

AvgRAM                  AliasProperty AvgRAM = AverageMemoryUsage

MaxRAM                  AliasProperty MaxRAM = MaximumMemoryUsage

MinRAM                  AliasProperty MinRAM = MinimumMemoryUsage

TotalDisk               AliasProperty TotalDisk = TotalDiskAllocation

AverageMemoryUsage      Property      System.Nullable[int] AverageMemoryUsage...

AverageProcessorUsage   Property      System.Nullable[int] AverageProcessorUs...

ComputerName            Property      string ComputerName {get;}

MaximumMemoryUsage      Property      System.Nullable[int] MaximumMemoryUsage...

MeteringDuration        Property      System.Nullable[timespan] MeteringDurat...

MinimumMemoryUsage      Property      System.Nullable[int] MinimumMemoryUsage...

NetworkMeteredTrafficReport Property  Microsoft.HyperV.PowerShell.VMPortAclMe...

TotalDiskAllocation     Property      System.Nullable[int] TotalDiskAllocatio...

VMId                    Property      guid VMId {get;}

VMName                  Property      string VMName {get;}
```

Sorting the output

I am not too impressed with the random order of the output. What I am concerned with is the amount of CPU utilization. So, I sort the output by AvgCPU (I know I can do this because of the

output from the **Get-Member** cmdlet). Here is the command I use to sort the output by average CPU usage.

get-vm | Measure-VM | sort avgCPU -Descending

The command and its associated output are shown here.