

Agenda: Azure Virtual Machine

- Introduction
- About Virtual Machine Workloads
- Create a Windows Virtual Machine using Portal / PowerShell / ARM Templates
- Deploy popular application frameworks by using Azure Resource Manager templates
- Understand and Capture VM Images
- Upload an on-premise VHD to Storage Account
- Deploy a New VM from the Captured Image
- Virtual Machine Scale Sets
- Virtual Machine Disk Types and VM Storage
- Virtual Machine Sizes in Azure
- Importing and Exporting Disks
- Configuring VM Security
- Perform configuration management
 - VM Extensions & VM Agents
 - Custom Script Extensions
 - Desired State Configuration (DSC)
 - Puppet or Chef
- Configure VM monitoring, configure alerts, diagnostic and monitoring storage location.

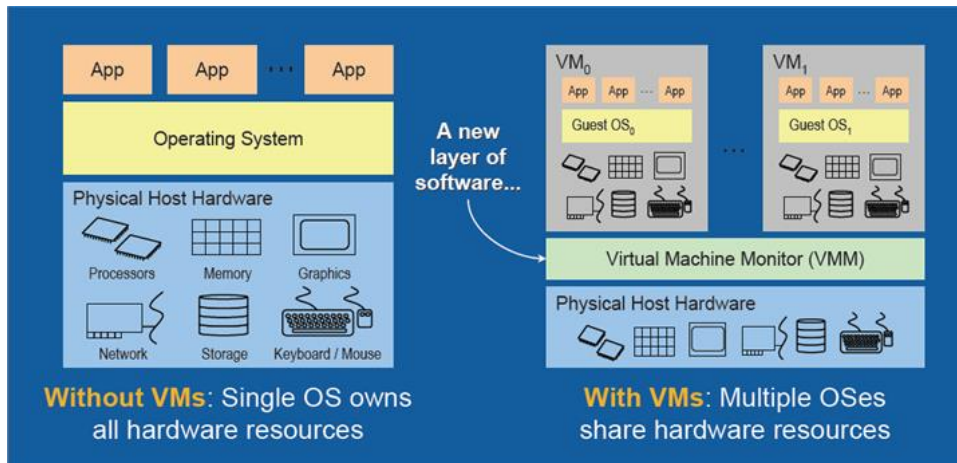
Virtual Machine Introduction

- A virtual machine acts like a **software-based computer** that runs an operating system and applications. The implementation of virtual machines may involve specialized hardware, software, or a combination of both.
- With **server virtualization**, you can create separate virtual machines and run them concurrently on a single server that is running **Microsoft Hyper-V**. These virtual machines are guests, while the computer that is running Hyper-V is the virtualization server or the management operating system.
- You can run multiple virtual machines that are using different operating systems on a virtualization server simultaneously, provided the virtualization server has enough resources.

What is a Virtual Machine

- A virtual machine is a computer file, typically called an image, which behaves like an actual computer – computer within a computer.
- It runs in a window, much like any other program, giving the end user the same experience on a virtual machine as they would have on the host operating system itself.
- Hypervisor is the software required for managing VM, emulates the PC or server's CPU, memory, hard disk, network and other hardware resources completely, enabling virtual machines to share the resources.

- The hypervisor can emulate multiple virtual hardware platforms that are isolated from each other, allowing virtual machines to run Linux and Windows Server operating systems on the same underlying physical host.



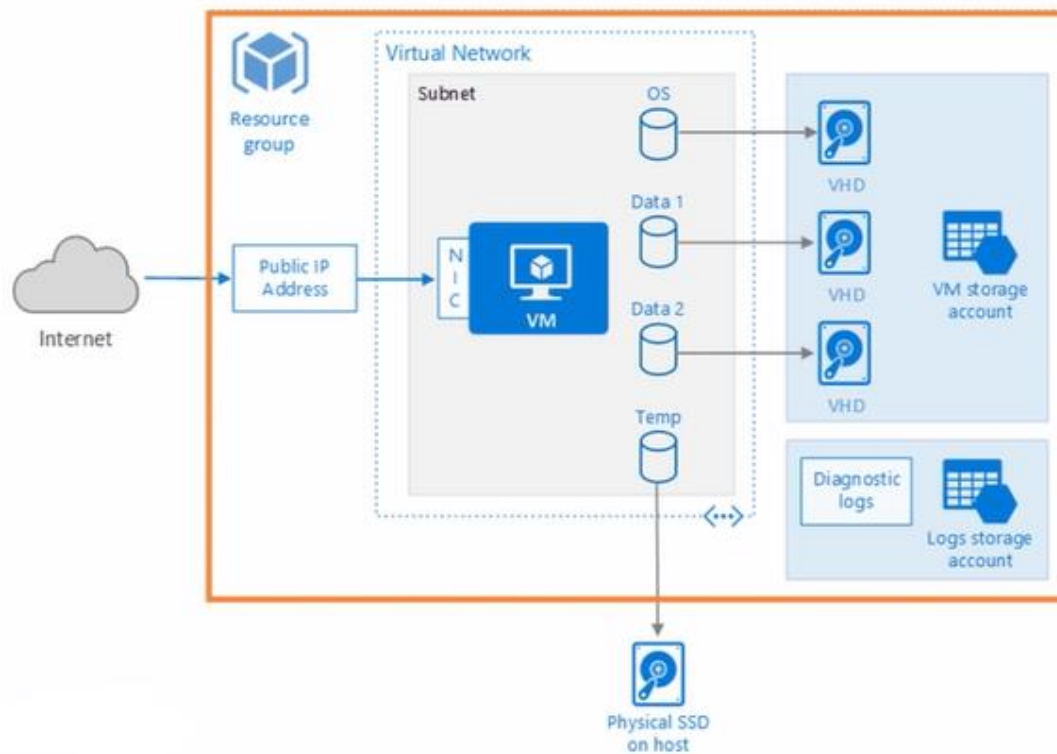
Azure Virtual Machine

- You'll choose a Virtual Machine if you need **more control over the computing** environment than the other choices offered by Azure. You have complete control over the virtual machine at the operating system level.
- We need to maintain the VM - **configuring, patching, and maintaining the operating system** and any other software that runs on the virtual machine.
- You can use an **image** provided by **Azure** or one of its **partners**, or use your **own** to create a VM.
- You must ensure all Microsoft software that you install in the Azure virtual-machine environment have **proper licenses**. By default, Azure Virtual Machines include a **license** that allows you to use **Windows Server** in the Azure environment. Additionally, certain Azure virtual-machine images might include licenses for additional Microsoft software, on a **per-hour or evaluation basis**. You must obtain licenses for other software separately.
- When you **shutdown** the virtual machine from its operating system, it will go into the **Stopped** state, and you will be **charged** for it, even if it is not running. You are not charged when the machine is in **Stopped (Deallocated)** state. The virtual machine will only go into the Stopped (Deallocated) state when you **shut it down from the Azure portal**.

Benefits of VM's:

- Launch Windows Server and Linux** (such as Suse, Ubuntu and CentOS) virtual machines in minutes. Scale from 1 to thousands of virtual machine instances.
- Virtual machines (VMs) for development and test** – We create them, use them and delete them when they are no longer needed.
- Run Applications which have large spikes in demand**. Benefit from built-in virtual networking and load balancing.
- Extend your own datacenter into the public cloud** (virtual network (VNET)). This allows running applications such as SharePoint, SQL Server and others on an Azure VM.

- **Disaster recovery** - If your primary datacenter goes down, you can create VMs running on Azure to run essential applications, then shut them down when they're no longer needed.
- Save money with **per-minute billing**.



About Virtual Machine Workloads

- A workload describes the nature of a solution, whether it is an application that runs on a single machine or it requires a complex topology that prescribes the operating system used, the additional software installed, the performance requirements, and the networking environment.
- You can deploy your existing application workloads to virtual machines in Azure that are running either on the Windows or Linux operating system.
- The Marketplace has a mixture of images provided by Microsoft, by third-party vendors, and by the Open Source community.
- Azure supports the creation of “bare-bones” VMs that provide from pre-built images available in the Marketplace.
 - Various versions of Windows or Linux operating system;
 - Azure supported Linux versions include CentOS by OpenLogic, Core OS, Debian, Oracle Linux, Red Hat Enterprise Linux, and Ubuntu.
 - Database and big-data workloads such as Microsoft SQL Server, IBM DB2.
 - Complete application infrastructure requiring server farms and clusters like SharePoint Biztalk Server, SQL server AlwaysOn and SAP.

- Workloads that provide security and protection such as Antivirus, intrusion detection systems, firewalls, data encryption and key management.
- Workloads that support developer productivity such as Windows 7 and Windows 8 client OS, Visual Studio or JDK.

Two approaches to identifying supported Azure workloads:

1. Determine whether the workload is already explicitly supported and offered through the Marketplace, which provides a large collection of free and for-pay solutions from Microsoft and third parties that deploy to VMs. The Marketplace also offers access to **the VM Depot**, which provides a large collection of community provided and maintained VMs.
2. Compare the requirements of the workload you want to deploy directly to the published capabilities of Azure VMs. The following is a representative, though not exhaustive, list of the requirements you would typically need to take into consideration:
 - a) CPU and RAM memory requirements
 - b) Disk storage capacity requirements, in gigabytes (GB)
 - c) Disk performance requirements, usually in terms of input/output operations per second (IOPS) and data throughput (typically in megabytes per second)
 - d) Operating system compatibility
 - e) Networking requirements
 - f) Availability requirements
 - g) Security and compliance requirements

Suitable Workloads for Microsoft Azure IaaS VMs

There are certain types of workloads that are a better fit for hosting in an Azure IaaS environment than others. Here are some examples:

- Highly available service workloads such as commercial online stores.
- Unpredictable growth workloads like those experienced by small, but rapidly expanding organizations, or short-term increased sales of fad items.
- Spiking workloads, such as those experienced by sites providing news services.
- Steady workload scenarios where organizations simply want to offload their infrastructure to the cloud.
- Periodic workloads such as retail sales spurts during holidays.

Workloads that are NOT SUITABLE for Microsoft Azure IaaS VMs:

When planning virtual machine workloads for Azure IaaS, it is also important to remember that not every application or service is a suitable fit for the cloud. Here are some examples.

- **Low volume or limited growth workloads** where the organization might be able to run the service or application on commodity hardware on-premise less expensively than in the cloud.

- **Regulated environment workloads** where an organization, or even the local government, may regulate the type of data that can be hosted in the cloud. However, these cases might be suitable candidates for a hybrid solution where only some highly available data is hosted in Azure and the more sensitive, regulated data is kept on-premises.

Create a Windows Virtual Machine

Different ways to create a Windows VM with Azure Resource Manager (ARM)

1. Azure Portal
2. Azure PowerShell
3. Azure CLI
4. ARM Template
5. Programming

Creating a VM using Azure portal:

The Management Portal provides many images and scripting tools that help you to create new virtual machines in Azure. The template images that are available in the portal are created and fully supported by either Microsoft or an authorized third-party.

Walkthrough:

1. Azure portal → On the Hub menu, click New → Compute → Windows Server 2016 Datacenter.
Note: To find additional images, click Marketplace and then search or filter for available items.
2. On the Windows Server 2012 R2 Datacenter page, under Select a deployment model = Resource Manager → Create.
3. Create virtual machine blade →
 - Basics → provide values for Name, Username and Password, Resource Group → OK
 - Size → Select an appropriate virtual machine size for your needs. Note that Azure recommends certain sizes automatically depending on the image you choose.
 - Settings to see storage and networking settings for the new virtual machine. For a first virtual machine you can generally accept the default settings.
 - Click Summary to review your configuration choices.
4. Click Create

Creating a VM using PowerShell

#Login to Azure Server.

Login-AzureRmAccount

Set-AzureRmContext -SubscriptionName "Visual Studio Enterprise"

Create the Resource Group

```
$rgName= "PowershellDemoRG"
```

```
$location= "East US"
```

```
New-AzureRmResourceGroup -Name $rgName -Location $location
```

A virtual machine is created based on a VHD image that can be selected from existing Azure prebuilt gallery images, or it can be created from a custom image that was uploaded into a storage account in Azure. Regardless of the selected method, **a storage account is required** to place the VHD(s) for the virtual machine. The storage account needs to be in the same Azure location as the virtual machine is created, so you can use the same variables for the location and the resource group name.

Create a Subnet, Virtual Network (VNet) and NIC Resources

VMs created with the Resource Manager Deployment model require a Resource Manager virtual network. If needed, create a new Resource Manager-based virtual network with at least one subnet for the new virtual machine.

```
$NICName="WebVM1-nic"
```

```
$PublicIPName = "WebVM1-ip"
```

```
$vNetName= "DemoVirtualNetwork"
```

```
$subnetName = "FrontEndSubnet"
```

```
$subnet=New-AzureRmVirtualNetworkSubnetConfig -Name $subnetName -AddressPrefix 192.168.1.0/24
```

```
# Get the reference to the VNet that has the subnet being targeted
```

```
$vNet = New-AzureRmVirtualNetwork -Name $vNetName -ResourceGroupName $rgName -Location $location -  
AddressPrefix "192.168.0.0/16" -Subnet $subnet
```

```
$subnet = Get-AzureRmVirtualNetworkSubnetConfig -Name $subnetName -VirtualNetwork $vnet
```

```
#Add the Backend subnet to the VNet
```

```
Add-AzureRmVirtualNetworkSubnetConfig -Name BackEndSubnet -VirtualNetwork $vnet -AddressPrefix  
192.168.2.0/24
```

```
Set-AzureRmVirtualNetwork -VirtualNetwork $vnet
```

```
# Create a public IP address object that can be assigned to the NIC (Network Interface Card)
```

```
$pubIP = New-AzureRmPublicIpAddress -Name $PublicIPName -ResourceGroupName $rgName -Location  
$location -AllocationMethod Dynamic -DomainNameLabel "dss-webvm1"
```

```
$(change DomainNameLabel when you practice)
```

#Create the NIC attached to a subnet, with a public facing IP, and a private IP

```
$NIC = New-AzureRmNetworkInterface -Name $NICName -ResourceGroupName $rgName -Location $location -  
SubnetId $vNet.Subnets[0].Id -PublicIpAddressId $pubIP.Id -PrivateIpAddress "192.168.1.4"
```

Create VM: Before you can actually create the virtual machine, you must specify the configuration information. In order to do this, you first create the configuration object that will store all the configuration information.

```
$vmName = "WebVM1"
```

```
$vmSize = "Standard_DS1"
```

Create the virtual machine configuration object and save a reference to it

```
$vmConfig = New-AzureRmVMConfig -VMName $vmName -VMSize $vmSize
```

Now that the virtual machine configuration object is created, the configuration information can be assigned to it. This includes defining the operating system, the base gallery image, and the previously created network adapter that you want to assign to the virtual machine. Optionally, you can also specify a name for the operating system VHD. If you do not specify one, Azure will assign a name automatically.

In order to define the gallery image, the publisher, offer, and the SKU for the gallery image is needed. You can refer to the following article to understand how to do determine the available values to use:

<https://azure.microsoft.com/en-us/documentation/articles/resource-groups-vm-searching/#powershell> .

Prompt for credentials that will be used for the local admin password for the VM

```
$cred = Get-Credential -Message "Type the name and password of the local administrator account."
```

OR

To provide fixed username and password.

```
$cred = New-Object System.Management.Automation.PSCredential ("username", $SecurePassword);
```

Assign the operating system to the VM configuration

```
$vmConfig = Set-AzureRmVMOperatingSystem -VM $vmConfig -Windows -ComputerName $vmName -Credential  
$cred -ProvisionVMAgent -EnableAutoUpdate
```

For this example, a Windows Server 2016 R2 Datacenter image is specified in the configuration information.

```
$pubName = "MicrosoftWindowsServer"
```

```
$offerName = "WindowsServer"
```

```
$skuName = "2016-Datacenter"
```

```
$diskName = "WebVM1OSDisk"
```

Assign the gallery image to the VM configuration

```
$vmConfig = Set-AzureRmVMSourceImage -VM $vmConfig -PublisherName $pubName -Offer $offerName -Skus $skuName -Version "latest"
```

Assign the **UNMANAGED OS Disk** name and location to the VM configuration

```
# $vmConfig = Set-AzureRmVMOSDisk -VM $vmConfig DiskSizeInGB 128 -Name $diskName -
```

```
VhdUri="https://<storage account reference of VHD> -CreateOption FromImage -Caching ReadWrite
```

Assign the NIC to the VM configuration

```
$vmConfig = Add-AzureRmVMNetworkInterface -VM $vmConfig -Id $NIC.Id
```

With the virtual machine configuration defined, the actual virtual machine is created using the New-AzureRmVM cmdlet with the configuration information passed as an argument.

```
New-AzureRmVM -ResourceGroupName $rgName -Location $location -VM $vmConfig
```

You can check the status of the provisioning using Get-AzureRmVM, passing it the resource group and VM name parameters. This retrieves the virtual machine configuration information. When the ProvisioningState value shows "Succeeded", then the virtual machine creation completed successfully, and the virtual machine should be in the running state.

```
Get-AzureRmVM -ResourceGroupName $RGPNName -Name $vmName
```

Create a Windows VM with ARM Template

1. Create the following resource file and save it as template.json

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": { "type": "string" },
    "adminPassword": { "type": "securestring" }
  },
  "variables": {
    "vnetID": "[resourceId('Microsoft.Network/virtualNetworks','myvn1')]",
    "subnetRef": "[concat(variables('vnetID'),'/subnets/Subnet1')]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "dssstorageforvhd",
```



```
"apiVersion": "2015-06-15",
"location": "[resourceGroup().location]",
"properties": { "accountType": "Standard_LRS" }
},
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Network/publicIPAddresses",
  "name": "myip1",
  "location": "[resourceGroup().location]",
  "properties": {
    "publicIPAllocationMethod": "Dynamic",
    "dnsSettings": { "domainNameLabel": "dssdomain" }
  }
},
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Network/virtualNetworks",
  "name": "myvn1",
  "location": "[resourceGroup().location]",
  "properties": {
    "addressSpace": { "addressPrefixes": [ "192.168.0.0/16" ] },
    "subnets": [ {
      "name": "Subnet1",
      "properties": { "addressPrefix": "192.168.1.0/24" }
    } ]
  }
},
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Network/networkInterfaces",
  "name": "mync1",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "Microsoft.Network/publicIPAddresses/myip1",
    "Microsoft.Network/virtualNetworks/myvn1"
  ],
  "properties": {
```

```
"ipConfigurations": [ {
  "name": "ipconfig1",
  "properties": {
    "privateIPAllocationMethod": "Dynamic",
    "publicIPAddress": {
      "id": "[resourceId('Microsoft.Network/publicIPAddresses', 'myip1')]"
    },
    "subnet": { "id": "[variables('subnetRef')]" }
  }
}
],
{
  "apiVersion": "2016-03-30",
  "type": "Microsoft.Compute/virtualMachines",
  "name": "myvm1",
  "location": "[resourceGroup().location]",
  "dependsOn": [
    "Microsoft.Network/networkInterfaces/mync1",
    "Microsoft.Storage/storageAccounts/dssstorageforvhd"
  ],
  "properties": {
    "hardwareProfile": { "vmSize": "Standard_A1" },
    "osProfile": {
      "computerName": "myvm1",
      "adminUsername": "[parameters('adminUsername')]",
      "adminPassword": "[parameters('adminPassword')]"
    },
    "storageProfile": {
      "imageReference": {
        "publisher": "MicrosoftWindowsServer",
        "offer": "WindowsServer",
        "sku": "2012-R2-Datacenter",
        "version": "latest"
      },
      "osDisk": {
        "name": "myosdisk1",
```

```

"vhd": {
  "uri": "https://dssstorageforvhd.blob.core.windows.net/vhds/myosdisk1.vhd"
},
" caching": "ReadWrite",
"createOption": "FromImage"
}
},
"networkProfile": {
  "networkInterfaces": [ {
    "id": "[resourceId('Microsoft.Network/networkInterfaces','mync1')]"
  } ]
}
}
}]
}

```

2. Create a parameter file

```

{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUserName": { "value": "dssadmin" },
    "adminPassword": { "value": "Deccansoft@123" }
  }
}

```

3. Create a Resource Group

```

$locName = "East US"
$rgName = "DemoWithARMRG"
New-AzureRmResourceGroup -Name $rgName -Location $locName

```

4. Create the Resources with template and parameters

```

$templateFile = "d:\\VM\\Template.json"
$parameterFile = "d:\\VM\\Parameters.json"
New-AzureRmResourceGroupDeployment -ResourceGroupName $rgName -TemplateFile $templateFile -
TemplateParameterFile $parameterFile

```

Following is copied from Microsoft Documentation:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": {
      "type": "string",
      "metadata": {
        "description": "Username for the Virtual Machine."
      }
    },
    "adminPassword": {
      "type": "securestring",
      "metadata": {
        "description": "Password for the Virtual Machine."
      }
    },
    "dnsLabelPrefix": {
      "type": "string",
      "metadata": {
        "description": "Unique DNS Name for the Public IP used to access the Virtual Machine."
      }
    },
    "windowsOSVersion": {
      "type": "string",
      "defaultValue": "2016-Datacenter",
      "allowedValues": [
        "2008-R2-SP1",
        "2012-Datacenter",
        "2012-R2-Datacenter",
        "2016-Nano-Server",
        "2016-Datacenter-with-Containers",
        "2016-Datacenter",
        "2019-Datacenter"
      ],
      "metadata": {
```

```
"description": "The Windows version for the VM. This will pick a fully patched image of this given Windows version."
```

```
}
```

```
},
```

```
"location": {
```

```
  "type": "string",
```

```
  "defaultValue": "[resourceGroup().location]",
```

```
  "metadata": {
```

```
    "description": "Location for all resources."
```

```
  }
```

```
}
```

```
},
```

```
"variables": {
```

```
  "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'dssvmstorage')]",
```

```
  "nicName": "myVMNic",
```

```
  "addressPrefix": "10.0.0.0/16",
```

```
  "subnetName": "FrontEndSubnet",
```

```
  "subnetPrefix": "10.0.0.0/24",
```

```
  "publicIPAddressName": "SimpleWindows-vm-ip",
```

```
  "vmName": "SimpleWindows-vm",
```

```
  "virtualNetworkName": "MyVNET",
```

```
  "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]"
```

```
},
```

```
"resources": [
```

```
{
```

```
  "type": "Microsoft.Storage/storageAccounts",
```

```
  "apiVersion": "2018-11-01",
```

```
  "name": "[variables('storageAccountName')]",
```

```
  "location": "[parameters('location')]",
```

```
  "sku": {
```

```
    "name": "Standard_LRS"
```

```
  },
```

```
  "kind": "Storage",
```

```
  "properties": {}
```

```
},
```

```
{
```

```
"type": "Microsoft.Network/publicIPAddresses",
"apiVersion": "2018-11-01",
"name": "[variables('publicIPAddressName')]",
"location": "[parameters('location')]",
"properties": {
  "publicIPAllocationMethod": "Dynamic",
  "dnsSettings": {
    "domainNameLabel": "[parameters('dnsLabelPrefix')]"
  }
},
{
  "type": "Microsoft.Network/virtualNetworks",
  "apiVersion": "2018-11-01",
  "name": "[variables('virtualNetworkName')]",
  "location": "[parameters('location')]",
  "properties": {
    "addressSpace": {
      "addressPrefixes": [
        "[variables('addressPrefix')]"
      ]
    },
    "subnets": [
      {
        "name": "[variables('subnetName')]",
        "properties": {
          "addressPrefix": "[variables('subnetPrefix')]"
        }
      ]
    ]
  }
},
{
  "type": "Microsoft.Network/networkInterfaces",
  "apiVersion": "2018-11-01",
  "name": "[variables('nicName')]",
  "location": "[parameters('location')]",
```

```
"dependsOn": [
  "[resourceId('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'))]",
  "[resourceId('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]"
],
"properties": {
  "ipConfigurations": [
    {
      "name": "ipconfig1",
      "properties": {
        "privateIPAllocationMethod": "Dynamic",
        "publicIPAddress": {
          "id": "[resourceId('Microsoft.Network/publicIPAddresses',variables('publicIPAddressName'))]"
        },
        "subnet": {
          "id": "[variables('subnetRef')]"
        }
      }
    }
  ]
},
{
  "type": "Microsoft.Compute/virtualMachines",
  "apiVersion": "2018-10-01",
  "name": "[variables('vmName')]",
  "location": "[parameters('location')]",
  "dependsOn": [
    "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
    "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
  ],
  "properties": {
    "hardwareProfile": {
      "vmSize": "Standard_A2"
    },
    "osProfile": {
      "computerName": "[variables('vmName')]",
      "adminUsername": "[parameters('adminUsername')]",
```

```
"adminPassword": "[parameters('adminPassword')]"
},
"storageProfile": {
  "imageReference": {
    "publisher": "MicrosoftWindowsServer",
    "offer": "WindowsServer",
    "sku": "[parameters('windowsOSVersion')]",
    "version": "latest"
  },
  "osDisk": {
    "createOption": "FromImage"
  },
  "dataDisks": [
    {
      "diskSizeGB": 1023,
      "lun": 0,
      "createOption": "Empty"
    }
  ]
},
"networkProfile": {
  "networkInterfaces": [
    {
      "id": "[resourceId('Microsoft.Network/networkInterfaces',variables('nicName'))]"
    }
  ]
},
"diagnosticsProfile": {
  "bootDiagnostics": {
    "enabled": true,
    "storageUri": "[reference(resourceId('Microsoft.Storage/storageAccounts/',
variables('storageAccountName'))).primaryEndpoints.blob]"
  }
}
},
],
```



```
"outputs": {
  "hostname": {
    "type": "string",
    "value": "[reference(variables('publicIPAddressName')).dnsSettings.fqdn]"
  }
}
```

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"
$adminUsername = Read-Host -Prompt "Enter the administrator username"
$adminPassword = Read-Host -Prompt "Enter the administrator password" -AsSecureString
$dnsLabelPrefix = Read-Host -Prompt "Enter an unique DNS name for the public IP"

New-AzResourceGroup -Name $resourceGroupName -Location "$location"
New-AzResourceGroupDeployment `
  -ResourceGroupName $resourceGroupName `
  -TemplateUri "https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-
windows/azuredeploy.json" `
  -adminUsername $adminUsername `
  -adminPassword $adminPassword `
  -dnsLabelPrefix $dnsLabelPrefix

(Get-AzVm -ResourceGroupName $resourceGroupName).name
```

Deploy popular application frameworks by using Azure Resource Manager templates

Deploying a template by using the Azure portal is easy to do by just sending a URL to it. You need the name of the template file to deploy it. You can find the name by looking at the pages in the template gallery or by looking in the Github repository. Change {template name} in this URL to the name of the template that you want to deploy and then enter it into your browser:

```
https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2FAzure-quickstart-templates%2Fmaster%2F{template name}%2Fazuredeploy.json
```

Example:

```
https://portal.azure.com/#create/Microsoft.Template/uri/https%3A%2F%2Fraw.githubusercontent.com%2FAzure%2FAzure-quickstart-templates%2Fmaster%2Fsharepoint-three-vm%2Fazuredeploy.json
```

Using Azure Portal:

1. Click on the above link
2. For the **Template** pane, click **Save**.
3. Click **Parameters**. On the **Parameters** pane, enter new values, select from allowed values, or accept default values, and then click **OK**.

4. If needed, click **Subscription** and select the correct Azure subscription.
5. Click **Resource group** and select an existing resource group. Alternately, click **Or create new** to create a new one for this deployment.
6. If needed, click **Location** and select the correct Azure location.
7. If needed, click **Legal terms** to review the terms and agreement for using the template.
8. Click **Create**.

A template file can link to another template

<https://github.com/Azure/azure-quickstart-templates/blob/master/sharepoint-three-vm/azuredeploy.json>

links to

<https://github.com/Azure/azure-quickstart-templates/blob/master/sharepoint-three-vm/availabilitySets.json>

This is done using the below JSON in actual Template file.

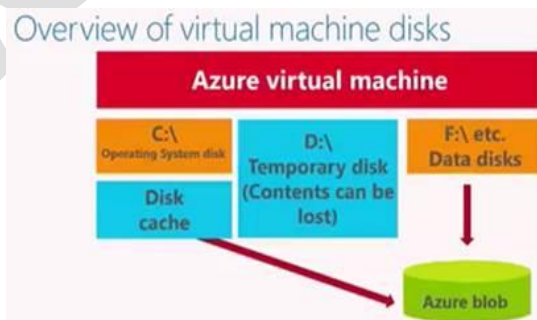
```
"templateLink": {  
  "uri": "[variables('CreatingAvailabilitySetsUrl')]",  
  "contentVersion": "1.0.0.0"  
},
```

Virtual Machine Disk Types

Just like any other computer, virtual machines in Azure uses disks as a place to store an operating system, applications, and data.

All Azure virtual machines have following disks:

1. Windows operating system disk
2. Temporary disk.
3. Optional: One or more data disks.



Operating System Disks

- Every virtual machine has one attached operating system disk.
- It's registered as a SATA drive and labeled as the C: drive by default.
- This disk has a maximum capacity of 32 TB

Temporary Disk

- Every virtual machine has a temporary disk that is **automatically** created for you.
- The size varies depending on tier size used.
- On Windows virtual machines, this disk is labeled as the D: drive by default
- It's used for storing non-persistent storage / caching data (eg: pagefile.sys)

Data Disks (Optional)

- Every virtual machine can have data disks to store application data, or other data you need to keep.
- Maximum size of 4095 GB / 4TB.
- Data disks are registered as SCSI drives and are labeled with a letter that you choose (**F: onwards**) .
- The size of the virtual machine determines how **many data disks** you can attach to it and the type of storage you can use to host the disks.
- Azure creates an operating system disk when you create a virtual machine from an image. If you use an image that includes data disks, Azure also creates the data disks when it creates the virtual machine. Otherwise, you add data disks after you create the virtual machine.

Important Notes:

- Operating system and data disks are implemented as **blob storage** in a storage account. The temporary disk is implemented as **local storage** on the Hyper-V host where the VM is running.
- Premium Storage (SSD's) requires DS, FS or GS-Series Virtual Machines.
- To add a new disk to your VM you must provide Name, Type, Size, Location and Host Caching method.
- All disks are stored as VHDs and the maximum capacity is 4TB.
- If your virtual machine is hosted on Server 2012 or above, you can use **storage pools** to virtualize storage by grouping industry-standard disks into pools, and then create **virtual disks called Storage Spaces** from the available capacity in the storage pools. Storage pools make it easy to grow or shrink volumes depending on your needs (and the capacity of the Azure data disks you have attached).

Types of Disks

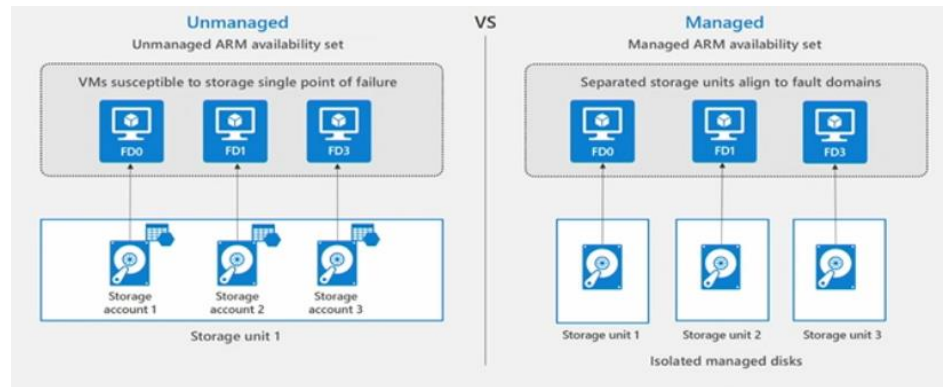
1. Unmanaged Disks:

- We need to create our own storage account.
- VHD files are stored as page blobs in Azure storage accounts.
- We need to ensure that we don't put too many disks in the same storage account because you could exceed the scalability targets of storage account (20,000 IOPS) resulting in VMs being throttled.

2. Managed Disks:

- Azure handles the storage account creation/management in the background for you, and ensures that you **do not have to worry about the scalability limits** of the storage account.
- You simply specify the disk size and the performance tier (Standard/Premium), and Azure creates and manages the disk for you. Even as you add disks or scale the VM up and down, you don't have to worry about the storage being used.
- You don't have to worry about placing the disks in multiple storage accounts to ensure that you stay within scalability limits for your storage accounts.

Enhanced availability – Availability set isolation



Steps to Add a Disk to VM:


1. More Services → Virtual machines → Select the VM created earlier.
2. VM → Settings → Disks → click **Attach new**.
3. On the Attach new blade, review the list of available settings. Fill in the following values and click OK.
 - Name: 01-DATADISK1
 - Type: HDD
 - Size (GiB): 100
 - Location: <Select your storage account>
 - Host caching: None
4. On the menu bar, monitor the alerts for progress as the new virtual disk is created and attached to the virtual machine.

Initialize a new data disk

1. Remotely connect to the VM → execute command **diskmgmt.msc** (Disk Management snap-in).
2. Disk Management will recognize that you have a new, un-initialized disk and the Initialize Disk window will pop up.
3. Make sure the new disk is selected and click **OK** to initialize it.
4. The new disk will now appear as **unallocated**. Right-click anywhere on the disk and select **New simple volume**. The **New Simple Volume Wizard** will start.

5. Go through the wizard, keeping all of the defaults, when you are done select **Finish**.
6. Close Disk Management.
7. You will get a pop-up that you need to format the new disk before you can use it. Click **Format disk**.
8. In the **Format new disk** dialog, check the settings and then click **Start**.
9. You will get a warning that formatting the disks will erase all of the data, click **OK**.
10. When the format is complete, click **OK**.

Detach a data disk using the portal

1. OPTIONAL: In the portal hub, select **Virtual Machines** → click **Stop** to deallocate the VM.
2. In the virtual machine blade, select **Disks**.
3. At the top of the **Disks** blade, select **Edit**.
4. In the **Disks** blade, to the far right of the data disk that you would like to detach, click the  detach button.
5. After the disk has been removed, click Save on the top of the blade.
6. In the virtual machine blade, click **Overview** and then click the **Start** button at the top of the blade to restart the VM.

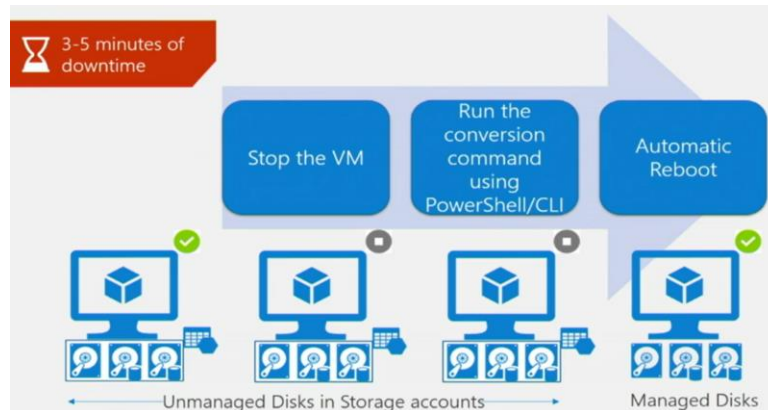
Managed Data Disk:

```
$rgName = 'myResourceGroup'
$vmName = 'myVM'
$location = 'West Central US'
$storageType = 'PremiumLRS'
$dataDiskName = $vmName + '_datadisk1'
$diskConfig = New-AzureRmDiskConfig -AccountType $storageType -Location $location -CreateOption Empty -
DiskSizeGB 128
$dataDisk1 = New-AzureRmDisk -DiskName $dataDiskName -Disk $diskConfig -ResourceGroupName $rgName
$vm = Get-AzureRmVM -Name $vmName -ResourceGroupName $rgName
$vm = Add-AzureRmVMDataDisk -VM $vm -Name $dataDiskName -CreateOption Attach -ManagedDiskId
$dataDisk1.Id -Lun 1
Update-AzureRmVM -VM $vm -ResourceGroupName $rgName
```

Unmanaged data disk:

```
$vm = Get-AzureRmVM -ResourceGroupName $rgName -Name $vmName
Add-AzureRmVMDataDisk -VM $vm -Name "disk-name" -VhdUri
"https://mystore1.blob.core.windows.net/vhds/datadisk1.vhd" -LUN 0 -Caching ReadWrite -DiskSizeinGB 1 -
CreateOption Empty
Update-AzureRmVM -ResourceGroupName $rgName -VM $vm
```

Steps to Convert Unmanaged Disk to Managed Disk



1. Check that the Disk Configuration is **Unmanaged**
`(Get-AzureRmVM DemoRG).StorageProfile.OsDisk`
2. Stop the VM
`Stop-AzureRmVm -ResourceGroupName "DemoRG" -Server DemoVMWithUnmanagedDisk`
3. Run the conversion command
`ConvertTo-AzureRmVMManagedDisk -ResourceGroup "DemoRG" -VMName DemoVMWithUnmanagedDisk`
`Start-AzureRmVM -ResourceGroup "DemoRG" -VMName DemoVMWithUnmanagedDisk`
4. Check Disk Configuration is now **Managed**
`(Get-AzureRmVM DemoRG).StorageProfile.OsDisk`

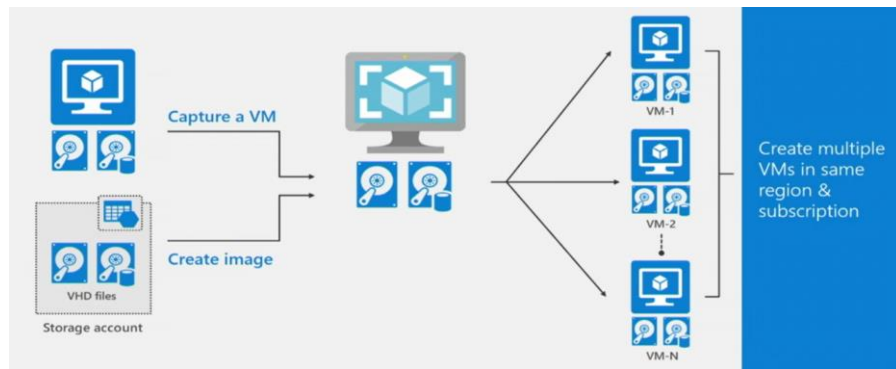
Note: Disk Conversion is supported only within same storage type. i.e. **Standard Unmanaged to Standard Managed and Premium Unmanaged to Premium Managed.**

Generalizing VM Image

Question: Should the OS settings in the VM template be exact copy of VM instance, including things like the machine identifiers and the administrator credentials, or can some of these settings be **generalized or reusable** so that they can be **changed** when the VM instance is provisioned from a template?

Generalized Image: You want to turn the VM you have created into a template from which you can stamp out new cloned instances of the VM, each of them unique with respect to certain settings in the operating system. For example, to scale out a cluster, you add new instances of the template VMs, each of them similarly configured but uniquely identified. In this case, each instance has its own identity and is therefore slightly different. In fact, when you create VMs from images available in the Marketplace, you utilize generalized images.

Sysprep is a process that you could run into a windows installation that will reset the installation of the system and will provide an “out of the box experience” by removing all personal data and resetting several components.



Step 1: Generalizing a Windows VM

1. RDP to the VM
2. Open a Command Prompt window as an administrator
3. Change the directory to %windir%\system32\sysprep, and then run **sysprep.exe**.

Note: Sysprep removes any machine-specific information and personal account information from the VHD.

4. In the **System Preparation Tool** dialog box, do the following:
 - In **System Cleanup Action**, select **Enter System Out-of-Box Experience (OOBE)** and make sure that **Generalize** is checked.
 - In **Shutdown Options**, select **Shutdown**.
 - Click **OK**.

OR Execute the following command at command prompt

Sysprep.exe /oobe /generalize /shutdown

Note: Sysprep shuts down the virtual machine. Its status changes to **Stopped** in the Azure portal.

Capture a Managed Image

Step 2: Capture a MANAGED Image in the Portal

1. Azure Portal → Select the VM → **Capture**
2. Provide the relevant options → **Create**

OR

Step 2: Capture the VM as VM Image using PowerShell:

1. RDP to VM and run Sysprep.exe as explained above
2. Open the Azure PowerShell 1.0.x and login to your Azure account.

[Login-AzureRmAccount](#)

3. Now you will need to **deallocate** the resources used by this virtual machine.

[Stop-AzureRmVM -ResourceGroupName "DemoRG" -Name "WebVM1"](#)

Note: Its status changes to **Stopped (deallocated)** in the Azure portal

- Next you need to set the status of the virtual machine to *Generalized*. Note that you will need to do this because the generalization step above (sysprep) does not do it in a way that Azure can understand.

```
Set-AzureRmVm -ResourceGroupName "DemoRG" -Name "WebVM1" -Generalized
```

- Get the VM

```
$vm = Get-AzureRmVM -Name $vmName -ResourceGroupName "DemoRG"
```

- Create the image configuration.

```
$imageConfig = New-AzureRmImageConfig -Location $location -SourceVirtualMachineId $vm.ID
```

- Create the image

```
New-AzureRmImage -Image $imageConfig -ImageName "NewImage" -ResourceGroupName "DemoRG"
```

Step 3: Use the Managed Image to Create a New VM

- Select the New Image Created → Settings → Create VM

Upload an on-premise VHD to Storage Account and Attach to VM as Data Disk

You can migrate on-premises workloads to Azure by uploading the .VHD file to Azure and attaching it to an Azure virtual machine.

You must upload a .VHD file to Azure Storage before you can attach it to the virtual machine, and consider several factors, including that:

- VHD files must be from Hyper-V virtual machines. VHDX files are not supported as Azure virtual machine disks.
- You must generalize the on-premises virtual machine by using sysprep.exe.
- The maximum size allowed for the VHD is 1,023 GB.
- The .VHD file must be a fixed-size virtual disk.

Summary of Steps for uploading .VHD files from local machine:

- Use Sysprep and Generalize the local virtual machine in Hyper-V.
- Make the VHD as fixed size file.
- Using **Convert-vhd** we can convert a .VHDX to .VHD file.
- Upload the VHD to Azure Storage Account using **Add-AzureRmVhd**
- Attach the VHD as data disk to VM

Step1: Generalize the VM using sysprep

Step2 & 3: Steps to Convert the virtual disk to VHD and fixed size disk

- Hyper-V Manager → select Local Computer → Menu Action → Edit Disk
- Select the Virtual Disk to be converted
- On Choose Action → **Convert** → Next
- Select **Fixed size** → Next → Provide Path of New VHD → Finish.

OR PowerShell Command to Convert VHDX to VHD

Convert-VHD –Path c:\test\MY-VM.vhdx –DestinationPath c:\test\MY-NEW-VM.vhd -VHDType Fixed

Step4: Upload the VHD to Azure Storage Account

Add-AzureRmVhd –Destination <http://<storageaccount>.blob.core.windows.net/vhd/Uploaded.vhd> -LocalFile 'c:\...\Local.vhd' –ResourceGroupName DemoRG – Verbose –NumberOfUploaderThreads 10

Note: The Storage account URL now can be used as Image for create a New VM.

Step5: Attach the disk as data disk to VM

1. Go to VM → Disk → Create Disk
2. Source type = Storage blob,
Storage blob = <http://<storageaccount>.blob.core.windows.net/vhd/Uploaded.vhd>,
OS type = None (data disk)
3. Create → Save

Verify: RDP to VM and check that the new Drive letter is attached to it.

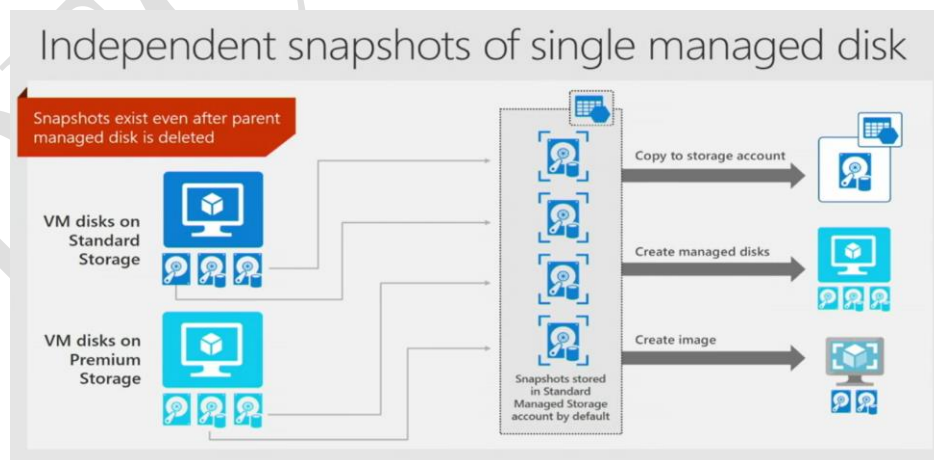
Working with Disk Snapshot

Snapshot can be created from Managed Disk and after creating the snapshot we can delete the managed disk and that is they are called as **Independent Snapshot**

Benefits of Snapshot

Snapshot can be copied onto another storage account in a different region.

Using Snapshot, we can restore our VM to a Point-In-Time by creating a Managed Disk from snapshot.

**Create Snapshot - Copy a disk**

1. Azure Portal → All Resources → Select the existing disk
2. Overview page → +Create snapshot

3. Create

Create a New Managed Disk from the snapshot

4. +Create a resource → Managed Disks → Create

Use the Managed Disk to create a new VM.

5. As steps provided above.

Realtime Usage Example:

Moving VM From One Subscription to Another Subscription in a different Azure Account

1. Stop the VM.
2. Take Snapshot of OS Disk of VM
3. Generate the **URL** for Snapshot with SAS Token
4. In Target Account Create StorageAccount and Blob Container
5. Execute following PowerShell commands

```
$destContext = New-AzureStorageContext -StorageAccountName "<targetstoragename>" -  
StorageAccountKey "<Target Storage Key>"  
Start-AzureStorageBlobCopy -AbsoluteUri <URL> -DestContainer "<TargetContainer>" -DestContext  
$destContext -DestBlob "VMDisk.vhd"
```
6. Create Managed Disk from the Storage Account BLOB
7. Create VM from the Managed Disk. (This works only if the Managed Disk is OS Disk and not data disk)

Note: Follow same steps as in 1, 2, 3, 4 to move Data disk. Attach the Data disk to VM

Virtual Machine Disk Types

Disk Types:

1. **Premium SSD:** Azure Premium Storage delivers **high-performance, low-latency** disk support for virtual machines (VMs) with **input/output (I/O)-intensive** workloads. Best suitable to run performance-intensive workloads in applications like Dynamics CRM, Exchange Server, SAP, SharePoint, SQL Server, Oracle, Redis, which require **consistent** high performance and low latency.
2. **Standard SSD:** A **cost-effective** storage option optimized for workloads that need **consistent performance** at **lower IOPS** levels. Standard SSDs deliver **better availability, consistency, reliability** and **latency** compared to HDD Disks, and are suitable for Web servers, low IOPS application servers, lightly used enterprise applications, and Dev/Test workloads.
3. **Standard HDD:** For development and testing purpose.

IOPS is number of requests that your application is sending to the storage disks in one second. An input/output operation could be read or write, sequential or random.

Throughput or Bandwidth is the amount of data that your application is sending to the storage disks in a specified interval.

There is a relation between Throughput and IOPS as shown in the formula below.

$$\boxed{\text{IOPS}} \times \boxed{\text{IO Size}} = \boxed{\text{Throughput}}$$

Latency is the time it takes an application to receive a single request, send it to the storage disks and send the response to the client

About Premium SSD

- In Azure, you can attach several premium storage disks to a VM. Using multiple disks gives your applications up to **256 TB** of storage per VM.
- With Premium Storage, your applications can achieve 80,000 I/O operations per second (IOPS) per VM, and a disk throughput of up to 2,000 megabytes per second (MB/s) per VM. Read operations give you very low latencies.
- Any object placed in a premium storage account will be a page blob. The page blob snaps to one of the supported provisioned sizes. This is why a premium storage account is not intended to be used to store tiny blobs.

Premium Storage disk limits: When you provision a premium storage disk, **the size of the disk determines** the maximum IOPS and throughput (bandwidth)

Premium Disks Type	P4	P6	P10	P15	P20	P30	P40	P50
Disk size	32 GiB	64 GiB	128 GiB	256 GiB	512 GiB	1024 GiB (1 TiB)	2048 GiB (2 TiB)	4095 GiB (4 TiB)
IOPS per disk	120	240	500	1100	2300	5000	7500	7500
Throughput per disk	25 MB per second	50 MB per second	100 MB per second	125 MB per second	150 MB per second	200 MB per second	250 MB per second	250 MB per second

Note: Maximum throughput and IOPS are based on the VM limits (based on Size), not on the disk limits described in the preceding table.

Disk size: Azure maps the disk size (rounded up) to the nearest premium storage disk option, as specified in the table in the preceding section. For example, a disk size of 100 GB is classified as a P10 option. It can perform up to 500 IOPS, with up to 100-MB/s throughput.

Standard SSD Features

1. Standard SSDs are only available as Managed Disks. Unmanaged Disks and Page Blobs are not supported on Standard SSD.
2. Standard SSDs can be used with all Azure VMs.
3. Standard SSDs are built on the same Azure Disks platform, which has consistently delivered high availability and durability for disks. Azure Disks are designed for 99.999 percent availability.

The following table contains disk sizes, which are currently offered for Standard SSD.

Standard SSD Disk Type	Disk Size	IOPS per Disk	Throughput per disk
E4	32 GiB	Up to 120	Up to 25 MiB per second
E6	64 GiB	Up to 240	Up to 50 MiB per second
E10	128 GiB	Up to 500	Up to 60 MiB per second
E15	256 GiB	Up to 500	Up to 60 MiB per second
E20	512 GiB	Up to 500	Up to 60 MiB per second
E30	1,024 GiB	Up to 500	Up to 60 MiB per second
E40	2,048 GiB	Up to 500	Up to 60 MiB per second
E50	4,095 GiB	Up to 500	Up to 60 MiB per second

Standard SSDs are designed to provide single-digit millisecond latencies for most IO operations, and to deliver the IOPS and throughput up to the limits described in the above table 99% of the time. Actual IOPS and Throughput may vary sometimes depending on the traffic patterns.

If development machine is moved to Production, we can move from Standard to Premium.

Steps to Convert from Standard **Unmanaged Disk to Premium Unmanaged Disk**

1. Stop the VM
2. Create a **Premium** Storage Account.
3. Copy all VM Disks (All VHD files) to the New Storage Account (**problem: can take time**)
4. Create a **New VM** using the copied VM Disks.

Problem: We have to reconfigure the new VM and we might forget few configuration steps from old VM.

Steps to Convert from Standard **Managed Disk to Premium Managed Disk**

1. Stop the VM
2. Update to Premium capable VM Size.
3. Update Storage Type to Premium.
4. Reboot

PowerShell Script:

Note: We can downgrade the Premium Storage to Standard Storage also.

Virtual Machine Sizes in Azure

- If you resize the VM in the portal, the only content that will be "lost" is the data on the temporary disk (typically the D:\ drive). All data on the OS disk and any persistent data disks will be retained.
- If your VM(s) are deployed using the *Resource Manager (ARM) deployment model* you can resize VMs by first **stopping** your VM, selecting a new VM size and then restarting the VM.

Azure offers several virtual-machine size groups that offer different levels of compute resources

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2, Fs, F	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2, Ls	High disk throughput and IO. Ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NV, NVv2, NC, NCv2, NCv3, ND,	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

For Pricing:

<https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/>

When determining sizing for your Azure Virtual Machines, you should consider the following:

- There are two tiers of Azure Storage for storing your virtual machine's virtual disks: **Standard and Premium**. Premium offers higher I/O throughput, but at a higher pricing level. SSD disks are supported in Premium only.
- The size of the virtual machine affects the pricing, and the tier affects some capabilities.
- A1 Standard is the smallest size that is recommended for production workloads.
- When deploying a virtual machine for SQL Server Enterprise Edition, select a virtual machine with at least four CPU cores.

- You can also use the online [Pricing Calculator](#) tool which enables you to cost out different workloads and services in Microsoft Azure.

Configuring VM Disk Encryption

Full Disk Encryption is the term used to indicate a technology that encrypts your entire hard drive. Your sensitive data, operating system, software programs, temp files, etc., are all encrypted. By encrypting the entire drive, in the event of the drive (your computer) being lost, stolen, or improperly decommissioned, the data that resides on the drive is inaccessible.

Using Azure Disk Encryption:

- **Azure Disk Encryption** is a capability built into the Azure platform that allows you to encrypt file system volumes residing on Windows and Linux virtual machine disks.
- Azure Disk Encryption leverages existing file system-based encryption technologies already available in the guest operating system (**BitLocker** in Windows and **DM-Crypt** in Linux) to provide encryption of volumes hosting the operating system and data disks.
- The solution integrates with **Key Vault** to securely store volume encryption keys.

Azure Disk Encryption is not supported for:

1. Basic tier virtual machines.
2. DS and GS series virtual machines (due to their support for Premium Storage disks).
3. Integration with on-premises Key Management Service.
4. Linux virtual machines running Red Hat Enterprise Linux.
5. Content of Azure Files (Azure file share), Network file system (NFS), dynamic volumes, and software based RAID configurations.

Steps:

<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/encrypt-disks>

Encrypt the Azure virtual machine:

1. \$vmName = <'your_vm_name'>
2. Use the values of parameters from Step 6 and execute the following command
Set-AzureRmVMDiskEncryptionExtension -ResourceGroupName \$resourceGroupName -VMName \$vmName -AadClientId \$aadClientId -AadClientSecret \$aadClientSecret -DiskEncryptionKeyVaultUrl \$diskEncryptionKeyVaultUrl -DiskEncryptionKeyVaultId \$keyVaultResourceId
3. Go to VM → Disks and you will see that **Encryption** is **Enabled**.

Perform Configuration Management using VM Extensions

Microsoft offers a number of different methods that simplify and enhance management of both Windows and Linux operating systems hosted on Azure virtual machines.

In general, you can categorize management options for Azure VMs depending on the operating system support they provide.

Windows Management Options

- RDP (Remote Desktop)

Linux Management Option

- SSH (Secure Shell)

Cross Platform management options

- Azure PowerShell
- Azure CLI
- **VM Agent and VM Agent Extensions**

VM Extensions:

- Azure virtual machine extensions are **small applications** that provide **post-deployment configuration** and automation tasks on Azure virtual machines. For example, if a virtual machine requires software installation, anti-virus protection, or Docker configuration, a VM extension can be used to complete these tasks.
- Extensions can be bundled with a new virtual machine deployment or run against any existing system.
- Azure VM extensions can be run by using
 - Azure portal.
 - PowerShell
 - Azure Resource Manager templates
 - Azure CLI
- There are many Available VM extensions and some which are popular are:
 - Custom Script Extension
 - Desired State Configuration
 - Chef
 - Puppet

VM Agents:

- The VM Agent is a set of lightweight software components running within the operating system of an Azure VM. Their primary purpose is to load additional programs and services known as **VM Agent Extensions**.
- The Azure VM agent is **preinstalled** on Azure Marketplace images and can be installed on supported operating systems.

- If the agent is not installed at provisioning time, or if you have migrated a virtual hard disk from on-premises, you can manually install the agent on these virtual machines by downloading and installing the agent from Microsoft at <http://go.microsoft.com/fwlink/?LinkID=394789&clcid=0x409>

To see a list of all VM extensions, run the following PowerShell commands.

```
get-command Set-AzureRM*Extension* -Module AzureRM.Compute
```

Custom Script Extension:

- The Custom Script Extension **downloads and executes** scripts on Azure virtual machines.
- The most common use of Custom Script extension involves applying **custom configuration settings** during VM provisioning.
- This extension is also useful for **stopping a VM or software installation**, or any other configuration / management task.
- Scripts can be downloaded from either **Azure Storage or GitHub**, or provided to the Azure portal at extension run time.
- The Custom Script Extension for Windows requires that the target virtual machine is connected to the internet.
- Following can be used to deploy custom script VM extension
 - Azure portal,
 - PowerShell,
 - Azure Resource Manager templates,
 - Azure CLI,
 - Azure Virtual Machine REST API.

Use **Set-AzureRmVMExtension** to install the Custom Script Extension.

Example: The extension runs *powershell Add-WindowsFeature Web-Server* to install the IIS webserver and then updates the *Default.htm* page to show the hostname of the VM:

```
Set-AzureRmVMExtension -ResourceGroupName $rgName `
  -ExtensionType CustomScriptExtension `
  -ExtensionName IIS `
  -VMName $vmName `
  -Publisher Microsoft.Compute `
  -TypeHandlerVersion 1.4 `
  -SettingString '{"commandToExecute":"powershell Add-WindowsFeature Web-Server; powershell Add-Content
-Path "C:\\inetpub\\wwwroot\\Default.htm" -Value $($env:computename)}' `
  -Location $location
```


The **Set-AzureRmVMCustomScriptExtension** cmdlet adds a custom script Virtual Machine Extension to a virtual machine. This extension lets you run your own scripts on the virtual machine

Example: Add a custom script extension using GitHub:

```
Set-AzureRmVMCustomScriptExtension -ResourceGroupName $rgName -VMName $vmName -Name  
"myCustomScript" -FileUri "https://raw.githubusercontent.com/neilpeterson/nepeters-azure-  
templates/master/windows-custom-script-simple/support-scripts/Create-File.ps1" -Run "Create-File.ps1" -  
Location "South India"
```

Note:

- In the above cmdlet change Location and set the value **same as VM location**.
- On completion, it creates a **c:\temp\AzureLog.txt** in VM.

To see the deployment state of extensions for a given VM, run the following command:

```
PS C:\> Get-AzureRmVMExtension -ResourceGroupName "DemoRG" -VMName "DemoVM" -Name  
"myCustomScript"
```

Example of Custom Script Extension using Storage Account:

```
PS C:\> Set-AzureRmVMCustomScriptExtension -ResourceGroupName "ResourceGroup11" -Location "Central US" -  
VMName "DemoVM" -Name "myCustomScript" -TypeHandlerVersion "1.1" -StorageAccountName  
"DssDemoStorage" -StorageAccountKey "XXXXXXX" -ContainerName "MyStorageBlobContainer" -FileName  
"Create-File.ps1"
```

Azure RM Template for Creating a Virtual Machine along with Custom Script Extension:

<https://github.com/Azure/azure-quickstart-templates/blob/master/201-vm-custom-script-windows/azuredeploy.json>

Following Section must be placed after "Properties" section of a VM Section in the ARM Template used above for creating a VM (Page 9 of this handout)

```
"resources": [  
  ...  
  {  
    "type": "extensions",  
    "name": "myCustomScript",  
    "apiVersion": "2016-04-30-preview",  
    "location": "eastus",  
    "properties": {  
      "publisher": "Microsoft.Compute",
```

```

"type": "CustomScriptExtension",
"typeHandlerVersion": "1.4",
"autoUpgradeMinorVersion": true,
"settings": {
  "fileUris": [
    "https://raw.githubusercontent.com/neilpeterson/nepeters-azure-templates/master/windows-custom-script-simple/support-scripts/Create-File.ps1"
  ],
  "commandToExecute": "powershell -ExecutionPolicy Unrestricted -file Create-File.ps1 "
},
"protectedSettings": {}
},
"dependsOn": [
  "[concat('Microsoft.Compute/virtualMachines/', 'myvm1')]"
]
}
]

```

Command to download and execute a PowerShell script that **installs IIS** and configures a basic home page.

```

az vm extension set \
--resource-group 58aa17ce-beb9-4ddd-8a56-9bc690c17358 \
--vm-name myVM \
--name CustomScriptExtension \
--publisher Microsoft.Compute \
--settings '{"fileUris":["https://raw.githubusercontent.com/MicrosoftDocs/mslearn-welcome-to-azure/master/configure-iis.ps1"]}' \
--protected-settings '{"commandToExecute": "powershell -ExecutionPolicy Unrestricted -File configure-iis.ps1"}'

```

Desired State Configuration (DSC) Extension

- It allows you to **declaratively configure** the **state** of the virtual machine. Using built-in resource providers or custom providers with a DSC script enables you to declaratively configure settings such as **roles and features**, registry settings, files and directories, firewall rules, and most settings available to Windows.
- Due to its declarative nature, it bears some resemblance to Azure Resource Manager, however, while Azure Resource Manager templates deploy Azure resources such as VMs, **DSC targets operating systems running within these VMs**.

- DSC supports ARM templates, Azure PowerShell, and CLI.
- One of the compelling features of DSC is that, instead of writing logic to detect and correct the state of the machine, the providers do that work for you and make the system state as defined in the script. Every resource has a property named **Ensure** that can be set to **Present** or **Absent**. In the example below, the WindowsFeature resource will verify whether the Web-Server role is present on the target machine and if it is not, the resource will install it.

A DSC script consists of the following:

- The **Configuration** block. This is the outermost script block. You define it by using the Configuration keyword and providing a name. In this case, the name of the configuration is "IISInstall".
- One or more **Node** blocks. These define the nodes (computers or VMs) that you are configuring. In the above configuration, there is one Node block that targets a computer named "localhost".
- One or more **resource** blocks. This is where the configuration sets the properties for the resources that it is configuring. In this case, there are two resource blocks, each of which call the WindowsFeature resource.

For example, the following DSC script declares that the Web-Server role should be installed, along with the Web-Asp-Net45 feature.

Create a file **D:\FeatureInstall.ps1** (extension must be ps1 only)

```
configuration IISConfig
{
    Node ("localhost") {
        WindowsFeature IIS {
            Ensure = "Present"
            Name = "Web-Server"
        }
        WindowsFeature AspNet45 {
            Ensure = "Present"
            Name = "Web-Asp-Net45"
        }
    }
}
```

Deploying DSC:

Execute the following PowerShell script after creating a Storage Account: **dssdemostorage**

```
$resourceGroup = "DemoRG"
$location = "South India"
```

```
$vmName = "DemoVM"
$storageName = "dssdemostorage"

#Publish the configuration script into user storage
Publish-AzureRmVMDscConfiguration -ConfigurationPath d:\featureInstall.ps1 -ResourceGroupName $resourceGroup -StorageAccountName $storageName -Force
# Note: Published file will have .zip extension

#Set the VM to run the DSC configuration
Set-AzureRmVmDscExtension -Version 2.21 -ResourceGroupName $resourceGroup -VMName $vmName -ArchiveStorageAccountName $storageName -ArchiveBlobName featureInstall.ps1.zip -AutoUpdate:$true -ConfigurationName "IISConfig"
```

Note: **Get-AzureVMDscExtension** retrieves the DSC extension status of a particular VM.

DSC Using Template:

```
"resources": [
  {
    "name": "Microsoft.Powershell.DSC",
    "type": "extensions",
    "location": "[resourceGroup().location]",
    "apiVersion": "2015-06-15",
    "dependsOn": [
      "[concat('Microsoft.Compute/virtualMachines/', variables('vmName'))]"
    ],
    "tags": {
      "displayName": "dscExtension"
    },
    "properties": {
      "publisher": "Microsoft.Powershell",
      "type": "DSC",
      "typeHandlerVersion": "2.20",
      "autoUpgradeMinorVersion": false,
      "forceUpdateTag": "[parameters('dscExtensionUpdateTagVersion')]",
      "settings": {
        "configuration": {
          "url": "[concat(parameters('_artifactsLocation'), '/', variables('dscExtensionArchiveFolder'), '/', variables('dscExtensionArchiveFileName'))]"
        }
      }
    }
  }
]
```

```

    "script": "FeatureInstall.ps1",
    "function": "Main"
  },
  "configurationArguments": {
    "nodeName": "[variables('vmName')]"
  }
},
"protectedSettings": {
  "configurationUrlSasToken": "[parameters('_artifactsLocationSasToken')]"
}
}
}
]

```

A **virtual machine scale set** node has a **"properties"** section with the "VirtualMachineProfile", "extensionProfile" attribute. DSC is added under "extensions"

```

"extensionProfile": {
  "extensions": [
    {
      "name": "Microsoft.Powershell.DSC",
      "properties": {
        "publisher": "Microsoft.Powershell",
        "type": "DSC",
        "typeHandlerVersion": "2.20",
        "autoUpgradeMinorVersion": false,
        "forceUpdateTag": "[parameters('DscExtensionUpdateTagVersion')]",
        "settings": {
          "configuration": {
            "url": "[concat(parameters('_artifactsLocation'), '/', variables('DscExtensionArchiveFolder'), '/', variables('DscExtensionArchiveFileName'))]",
            "script": "DscExtension.ps1",
            "function": "Main"
          },
          "configurationArguments": {
            "nodeName": "localhost"
          }
        }
      },

```

```
    "protectedSettings": {  
      "configurationUrlSasToken": "[parameters('_artifactsLocationSasToken')]"  
    }  
  }  
}  
]  
}
```

VM Access Extension:

- Currently, the extension can only be enabled using the **Set-AzureVMAccessExtension** cmdlet.
- This cmdlet can reset the local administrator name, password, and also enable Remote Desktop access if it is accidentally disabled.
- This extension does not work against Active Directory domain accounts or on domain controllers.

```
Set-AzureRmVMAccessExtension -ResourceGroupName "DemoRG" -Location "Central US" -VMName "DemoVM" -  
Name "DemoAE" -TypeHandlerVersion "2.0" -UserName "dssadmin" -Password "Password@123"
```

Virtual Machine Scale Sets

- A VM scale set consists of a **group** of automatically provisioned Windows or Linux virtual machines that share **identical configurations** and deliver the same functionality to support a service or application.
- With a VM scale set, it is possible to have the number of virtual machines **increase or decrease**, adjusting dynamically to changes in demand for the service or application. To implement on demand autoscaling, you combine VM Scale Sets with Azure Insights **Autoscale**.
- With scale sets, all VM instances are created from the **same base OS image and configuration**. This approach lets you easily manage hundreds of VMs without additional configuration tasks or network management.
- It's easier to **build large-scale services** targeting big compute, big data, and containerized workloads.

General guidance

- You can create both Linux and Windows VM Scale Sets from the Azure Portal. These scale sets are automatically created with **load balancer NAT rules** to enable SSH or RDP connections.
- With un-managed disk, Max VM = 100 and with Managed Disk it can go upto 1000 VMs
- If you create and upload your own custom VM images, the limit is 300 VM instances.
- When you increase the number of virtual machines in a scale set, VMs are balanced across update and fault domains to ensure, maximum availability. Similarly, when you scale in, VMs are removed with maximum availability in mind.
- You can set the maximum, minimum and default number of VMs, and define triggers – action rules based on resource consumption.

To Create a Virtual Machine Scale Set

2. More Services → Virtual machine scale sets → +Add
3. Basic → Name = DemoScaleSet, single placement group = True . . . → Resource group, Create new = DemoRG → OK
4. Virtual Machine scaleset service settings: Provide Public IP, Domain name label, Operating system disk image = 2016-Datacenter, Auto Scaling=True → OK

Remote Desktop Login to VM of a Scale Set

5. Select **Load balancer** used by Scale set → Settings → Inbound NAT rules → Note the **IP address and Port number** of each instance.
6. Local Windows → Search Remote Desktop → Computer = Provide **IP:PortNo** → Connect
7. Provide the username and password which was provided when scale set was created → OK

To create a VMSS

```
New-AzureRmVmss `
-ResourceGroupName "myResourceGroup" `
-Location "EastUS" `
-VMScaleSetName "myScaleSet" `
-VirtualNetworkName "myVnet" `
-SubnetName "mySubnet" `
-PublicIpAddressName "myPublicIpAddress" `
-LoadBalancerName "myLoadBalancer" `
-UpgradePolicyMode "Automatic" `
-DataDiskSizeInGb 64,128
```

Note: This creates VM ScaleSet with two data disks. The first disk is 64 GB in size, and the second disk is 128 GB.

Create and use a custom image for virtual machine scale sets with Azure PowerShell

1. Create a New VM
2. RDP and install IIS and all other required softwares into it.
3. Generalize the VM: C:\Windows\system32\sysprep\sysprep.exe /oobe /generalize /shutdown
4. Create a custom VM Image from the source VM (**Capture** option in VM Overview blade)
5. Create a scale set from the custom VM Image

```
New-AzureRmVmss `
-ResourceGroupName "myResourceGroup" `
-Location "EastUS" `
-VMScaleSetName "myScaleSet" `
```

```
-VirtualNetworkName "myVnet" `
-SubnetName "mySubnet" `
-PublicIpAddressName "myPublicIpAddress" `
-LoadBalancerName "myLoadBalancer" `
-UpgradePolicyMode "Automatic" `
-ImageName "myImage"
```

Preparing the Disk using Custom Script Extension

The disks that are created and attached to your scale set VM instances are raw disks. Before you can use them with your data and applications, the disks must be prepared. To prepare the disks, you create a partition, create a filesystem, and mount them.

The following example executes a script from a GitHub sample repo on each VM instance with [Add-AzureRmVmssExtension](#) that prepares all the raw attached data disks:

```
# Define the script for your Custom Script Extension to run
$customConfig = @{
    "fileUri" = (,"https://raw.githubusercontent.com/Azure-Samples/compute-automation-
configurations/master/prepare_vm_disks.ps1");
    "commandToExecute" = "powershell -ExecutionPolicy Unrestricted -File prepare_vm_disks.ps1"
}
```

To confirm that the disks have been prepared correctly, RDP to one of the VM instances.

Install applications in virtual machine scale sets with Azure PowerShell

```
# Get information about the scale set
$vmss = Get-AzureRmVmss `
    -ResourceGroupName "myResourceGroup" `
    -VMScaleSetName "myScaleSet"

# Add the Custom Script Extension to install IIS and configure basic website
$vmss = Add-AzureRmVmssExtension `
    -VirtualMachineScaleSet $vmss `
    -Name "customScript" `
    -Publisher "Microsoft.Compute" `
    -Type "CustomScriptExtension" `
    -TypeHandlerVersion 1.8 `
    -Setting $customConfig

# Update the scale set and apply the Custom Script Extension to the VM instances
```



```
Update-AzureRmVmss `
  -ResourceGroupName "myResourceGroup" `
  -Name "myScaleSet" `
  -VirtualMachineScaleSet $vmss
```

To Update app deployment:

Apply the Custom Script Extension configuration to the VM instances in your scale set again with **Add-AzureRmVmssExtension** followed by **Update-AzureRmVmss**

Attach a disk to existing scale set

```
# Get scale set object
$vmss = Get-AzureRmVmss `
  -ResourceGroupName "myResourceGroup" `
  -VMScaleSetName "myScaleSet"

# Attach a 128 GB data disk to LUN 2
Add-AzureRmVmssDataDisk `
  -VirtualMachineScaleSet $vmss `
  -CreateOption Empty `
  -Lun 2 `
  -DiskSizeGB 128

# Update the scale set to apply the change
Update-AzureRmVmss `
  -ResourceGroupName "myResourceGroup" `
  -Name "myScaleSet" `
  -VirtualMachineScaleSet $vmss
```

What is Azure Resource Explorer?

- Download from <http://resources.azure.com>
- It's a great tool to view and modify resources you have created in your subscription. The tool is web-based and uses your Azure portal login credentials.
- This tool is particularly useful in viewing Azure scale sets. With the tool you can see the individual virtual machines and their properties.

Configure VM monitoring, configure alerts, diagnostic and monitoring storage location

- Azure IaaS monitoring involves collecting and tracking metrics, analyzing log files, defining custom metrics and logging generated by specific applications or workloads running in Virtual Machines.
- Monitoring will also involve triggering alarms when certain conditions are met, and providing diagnostic data to help in troubleshooting and root cause analysis.
- Monitoring can help you gain visibility into your running deployments, resource utilization, application performance, operational health and application diagnostics.

Following Items can be logged once Diagnostic Settings is enabled:

1. Performance counters
 - a. Configure the performance counters to collect, and how often they should be sampled
 - i. CPU
 - ii. Memory
 - iii. Disk
 - iv. Network
 - v. ASP.NET
 - vi. SQL Server
2. **Logs:** Collecting data for these logs: Application, Security and System.
3. **Crash Dumps:** Collect memory dumps when a process crashes
4. Sinks
5. Send your diagnostic data to other services for more insights
6. Diagnostics Agent
 - a. Configure additional options for the Azure Diagnostics agent.
7. Boot Diagnostics
 - a. Periodically capture screenshots of the virtual machine running on a host to help diagnose startup issues.
 - b. Screenshots can be viewed from Boot diagnostics blade.

To configure: Azure VM → Settings → Diagnostic Settings

To View: Azure VM → Settings → Diagnose and solve problems.