Day#9+10

Azure administrations

How to create and manage resources in Azure

Manage resources with Azure **PowerShell.**

Manage Resources with Power Shell

- Creating administration **scripts** is a powerful way to optimize your **workflow**. You can automate common, repetitive tasks. Once a script has been verified, it will run consistently, likely reducing errors.
- How to Choose an administrative tool:
  - **Automation**: Do you need to automate a set of **complex** or **repetitive** tasks? Azure PowerShell and the Azure CLI support this, while **Azure portal** does not.
  - **Learning curve**: Do you need to **complete a task quickly** without learning new commands or syntax? The Azure portal does not require you to learn syntax or memorize commands. In Azure PowerShell and the Azure CLI, you must know the detailed syntax for each command you use.
  - **Team skillset:** Does your team have existing expertise? For example, your team may have used PowerShell to administer Windows. If so, they will quickly become comfortable using Azure PowerShell.

What is a PowerShell command (cmdlet)?

- Commands for PowerShell are known as cmdlets (pronounced command-lets). In addition to cmdlets, PowerShell allows you to run any command available on your system.

- **What is a cmdlet?**
  - Cmdlets are native PowerShell commands, not stand-alone executables. Cmdlets are collected into PowerShell modules that can be loaded on demand. Cmdlets can be written in any compiled .NET language or in the PowerShell scripting language itself.

- **Cmdlet names**
  - PowerShell uses a Verb-Noun name pair to name cmdlets. For example, the Get-Command cmdlet included in PowerShell is used to get all the cmdlets that are registered in the command shell. The verb identifies the action that the cmdlet performs, and the noun identifies the resource on which the cmdlet performs its action.

How to install Power Shell?

Linux

On Linux, the package manager will change based on the Linux distribution you choose.

Distribution(s)	Package manager
Ubuntu, Debian	`apt-get`
Red Hat, CentOS	`yum`
OpenSUSE	`zypper`
Fedora	`dnf`

Mac

On macOS, you'll use `Homebrew` to install PowerShell.

How to install Power Shell? - Linux

- Installing PowerShell for Linux involves using a package manager. Here come for Ubuntu 18.04 for our example, but check other versions here

Install PowerShell on Ubuntu Linux using the Advanced Packaging Tool (**apt**) and the Bash command line.

1. Import the encryption key for the Microsoft Ubuntu repository. This key enables the package manager to verify that the PowerShell package you install comes from Microsoft.

```
Bash
curl https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -
```

2. Register the **Microsoft Ubuntu repository** so the package manager can locate the PowerShell package.

```
Bash
sudo curl -o /etc/apt/sources.list.d/microsoft.list https://packages.microsoft.com/config/ubuntu/18.04/pr
sudo curl -o /etc/apt/sources.list.d/microsoft.listhttps://packages.microsoft.com/config/ubuntu/18.04/prod.list
```

3. Update the list of packages.

```
Bash
sudo apt-get update
```

4. Install PowerShell.

```
Bash
sudo apt-get install -y powershell
```

5. Start PowerShell to verify that it installed successfully.

```
Bash
pwsh
```

How to install Power Shell? - Windows

- Installing power shell here
- Install DotNet SDK here

1. In the **System tray search box**, type **PowerShell**. You may have multiple shortcut links:

   - PowerShell 7 (x64) - The 64-bit version. Generally, this is the shortcut you should choose.
   - Windows PowerShell - The 64-bit version included with Windows.
   - Windows PowerShell (x86) - A 32-bit version installed on 64-bit Windows.
   - Windows PowerShell ISE - The Integrated Scripting Environment (ISE) is used for writing scripts in Windows PowerShell.
   - Windows PowerShell ISE (x86) - A 32-bit version of the ISE on Windows.

2. Select the best match PowerShell icon.

3. Type the following command to determine the version of PowerShell installed.

```
PowerShell                                                    Copy
$PSVersionTable.PSVersion
```

or

```
PowerShell                                                    Copy
pwsh -ver
```

If the major version number is lower than 7, follow the instructions to upgrade existing Windows PowerShell. It is important to install the SDK to support .NET tools, as well.

You need the .NET SDK installed to run this command.

```
PowerShell                                                    Copy
dotnet tool install --global PowerShell
```

After the .NET tool is installed, run the PowerShell version command again to verify your installation.

What is Power Shell CmdLets? ?

A PowerShell command is called a **cmdlet** (pronounced "command-let"). A cmdlet is a command that manipulates a single feature. The term **cmdlet** is intended to imply "small command". By convention, cmdlet authors are encouraged to keep cmdlets simple and single-purpose.

The base PowerShell product ships with cmdlets that work with features such as sessions and background jobs. You can add modules to your PowerShell installation to get cmdlets that manipulate other features. For example, there are third-party modules to work with ftp, administer your operating system, access the file system, and so on.

Cmdlets follow a verb-noun naming convention; for example, `Get-Process`, `Format-Table`, and `Start-Service`. There is also a convention for verb choice: "get" to retrieve data, "set" to insert or update data, "format" to format data, "out" to direct output to a destination, and so on.

Cmdlet authors are encouraged to include a help file for each cmdlet. The `Get-Help` cmdlet displays the help file for any cmdlet. For example, to get help on the `Get-ChildItem` cmdlet, enter the following statement in a Windows PowerShell session:

PowerShell Copy

```
Get-Help -Name Get-ChildItem -Detailed
```

What is a PowerShell module?

Cmdlets are shipped in *modules*. A PowerShell Module is a DLL that includes the code to process each available cmdlet. You'll load cmdlets into PowerShell by loading the module in which they're contained. You can get a list of loaded modules using the `Get-Module` command:

PowerShell Copy

```
Get-Module
```

This will output something like:

Output Copy

```
ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Manifest   3.1.0.0    Microsoft.PowerShell.Management     {Add-Computer, Add-Content, Checkpoint-Computer, Clea
Manifest   3.1.0.0    Microsoft.PowerShell.Utility        {Add-Member, Add-Type, Clear-Variable, Compare-Object
Binary     1.0.0.1    PackageManagement                   {Find-Package, Find-PackageProvider, Get-Package, Get
Script     1.0.0.1    PowerShellGet                       {Find-Command, Find-DscResource, Find-Module, Find-Ro
Script     2.0.0      PSReadline                          {Get-PSReadLineKeyHandler, Get-PSReadLineOption, Remo
```

What is the Az PowerShell module?

- Az is the formal name for the Azure PowerShell module, which contains cmdlets to work with Azure features. It contains hundreds of cmdlets that let you control nearly every aspect of every Azure resource. You can work with resource groups, storage, virtual machines, Azure Active Directory, containers, machine learning, and so on. This module is an open-source component available on GitHub.

How to Install the Az PowerShell module?

After you Open the **Start** menu and enter **PowerShell**.
And Select the **PowerShell** icon.

1. Use the `SetExecutionPolicy` cmdlet to change the policy to "RemoteSigned":

PowerShell Copy

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
```

This will prompt you for permission:

Output Copy

```
The execution policy helps protect you from scripts that you do not trust. Changing the execution policy
you to the security risks described in the about_Execution_Policies help topic at
https:/go.microsoft.com/fwlink/?LinkID=135170. Do you want to change the execution policy?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"): Y
```

2. Enter **Y** or **A**, then press `Enter`.

How to Install the Az PowerShell module? - ctd

3. Enter the following command, and then press `Enter`.

PowerShell Copy

```
Install-Module -Name Az -Scope CurrentUser -Repository PSGallery
```

This installs the module for your current user (controlled by the `Scope` parameter).

The command relies on NuGet to retrieve components, so depending on the version you have installed, you might be prompted to download and install the latest version of NuGet.

Output Copy

```
NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories.
 provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or
'C:\Users\<username>\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provi
'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install a
 the NuGet provider now?
 [Y] Yes  [N] No  [S] Suspend  [?] Help (default is "Y"):
```

Enter **Y** and press `Enter`.

By default, the PowerShell Gallery isn't configured as a trusted repository for PowerShellGet. Each time you perform an installation from an untrusted repository, you'll be prompted to confirm you want to install the module with following output:

Output Copy

```
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules
'PSGallery'?
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is "N"):
```

Enter **Y** or **A**, then press `Enter`.

How to update Az PowerShell module?

If you get a warning or error message indicating that a version of the Azure PowerShell module is already installed, you can update to the latest version by issuing the following command:

PowerShell Copy

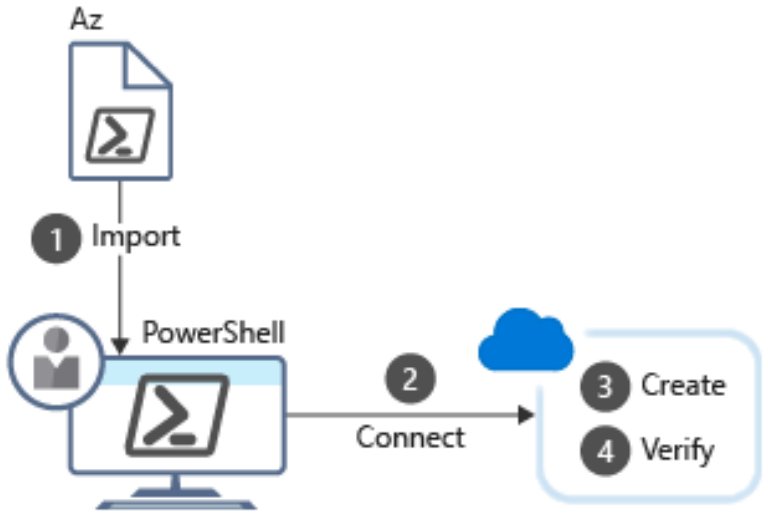
```
Update-Module -Name Az
```

As with the `Install-Module` cmdlet, answer **Yes** or **Yes to All** when prompted to trust the module. You can also use the `Update-Module` command to reinstall a module if you're having trouble with it.

How to Create Resource group With Azure power Shell?

There are four steps you need to perform:

1.Import the Azure cmdlets.
2.Connect to your Azure subscription.
3.Create the resource group.
4.Verify that creation was successful.



- Each step corresponds to a different cmdlet.

How to Create Resource group With Azure power Shell?

Connect

When you're working with a local install of Azure PowerShell, you'll need to authenticate before you can execute Azure commands. The `Connect-AzAccount` cmdlet prompts for your Azure credentials, then connects to your Azure subscription. It has many optional parameters, but if all you need is an interactive prompt, you don't need any parameters:

Azure PowerShell Copy

```
Connect-AzAccount
```

Work with subscriptions

If you're new to Azure, you probably only have a single subscription. But if you've been using Azure for a while, you might have created multiple Azure subscriptions. You can configure Azure PowerShell to execute commands against a particular subscription.

You can only be in one subscription at a time. Use the `Get-AzContext` cmdlet to determine which subscription is active. If it's not the correct one, you can change subscriptions using another cmdlet.

1. Get a list of all subscription names in your account with the `Get-AzSubscription` command.

2. Change the subscription by passing the name of the one to select.

Azure PowerShell Copy

```
Set-AzContext -Subscription '00000000-0000-0000-0000-000000000000'
```

If you need to look up the Subscription ID, open Azure and select **Subscriptions** on the home page.

How to Create Resource group With Azure power Shell?

Get a list of all resource groups

You can retrieve a list of all Resource Groups in the active subscription.

Azure PowerShell Copy

```
Get-AzResourceGroup
```

To get a more concise view, you can send the output from the `Get-AzResourceGroup` to the `Format-Table` cmdlet using a pipe '|'.

Azure PowerShell Copy

```
Get-AzResourceGroup | Format-Table
```

The output will look something like this:

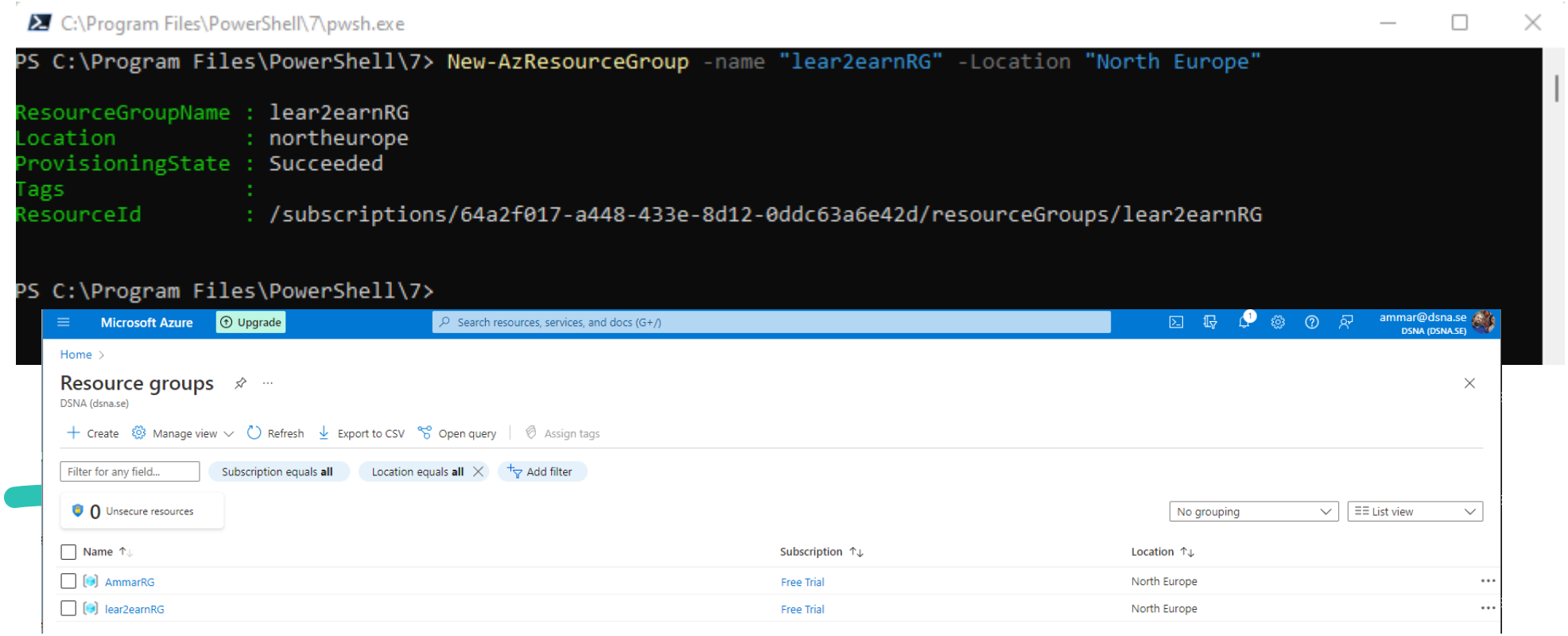
Output Copy

```
ResourceGroupName                 Location       ProvisioningState Tags TagsTable ResourceId
-----------------                 --------       ----------------- ---- --------- ----------
cloud-shell-storage-southcentralus southcentralus Succeeded                       /subscriptions/00000000-000
ExerciseResources                 eastus         Succeeded                       /subscriptions/00000000-000
```

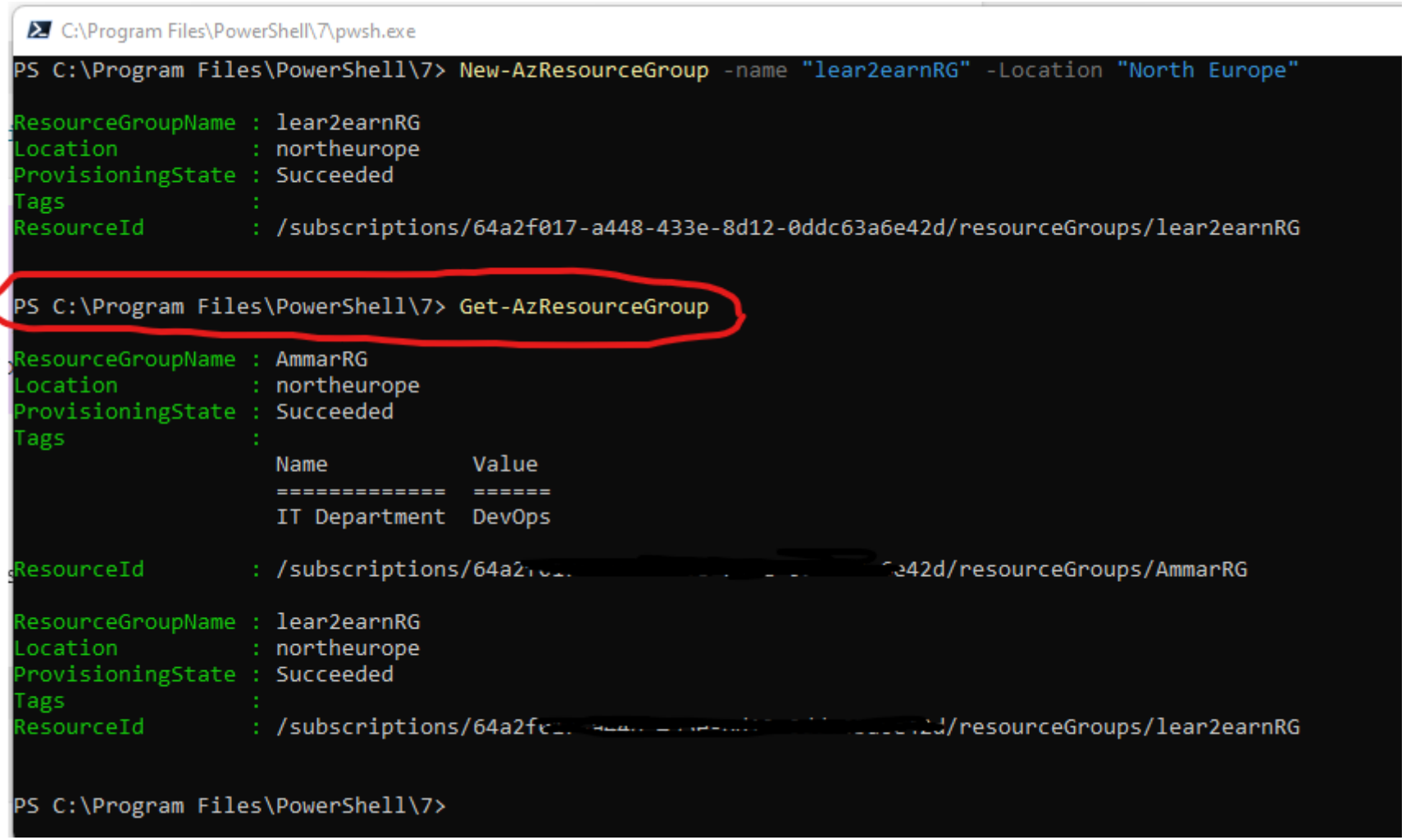
How to Create Resource group With Azure power Shell?

- You can create resource groups by using the **New-AzResourceGroup** cmdlet. You must specify a name and location. The name must be unique within your subscription. The location determines where the metadata for your resource group will be stored (which may be important to you for compliance reasons). You use strings like "West US", "North Europe", or "West India" to specify the location. As with most of the Azure cmdlets,
- New-AzResourceGroup has many optional parameters. However, the core syntax is:

```
New-AzResourceGroup -Name <name> -Location <location>
```



How to Create Resource group With Azure power Shell?



How to Create VM With Azure power Shell?

Azure PowerShell provides the `New-AzVm` cmdlet to create a virtual machine. The cmdlet has many parameters to let it handle the large number of VM configuration settings. Most of the parameters have reasonable default values, so we only need to specify five things:

- **ResourceGroupName**: The resource group into which the new VM will be placed.
- **Name**: The name of the VM in Azure.
- **Location**: Geographic location where the VM will be provisioned.
- **Credential**: An object containing the username and password for the VM admin account. We'll use the `Get-Credential` cmdlet. This cmdlet will prompt for a username and password and package it into a credential object.
- **Image**: The operating system image to use for the VM, which is typically a Linux distribution or Windows Server.

PowerShell Copy

```
New-AzVm
    -ResourceGroupName <resource group name>
    -Name <machine name>
    -Credential <credentials object>
    -Location <location>
    -Image <image name>
```

You can supply these parameters directly to the cmdlet as shown above. Alternatively, you can use other cmdlets to configure the virtual machine, such as `Set-AzVMOperatingSystem`, `Set-AzVMSourceImage`, `Add-AzVMNetworkInterface`, and `Set-AzVMOSDisk`.

How to Create VM With Azure power Shell?


```
PS C:\Program Files\PowerShell\7> Get-AzResource | Format-Table

Name                      ResourceGroupName ResourceType                          Location
----                      ----------------- ------------                          --------
AmmarVM                   AmmarRG           Microsoft.Network/networkInterfaces     northeurope
AmmarVM                   AmmarRG           Microsoft.Network/networkSecurityGroups northeurope
AmmarVM                   AmmarRG           Microsoft.Network/virtualNetworks       northeurope
NetworkWatcher_northeurope NetworkWatcherRG  Microsoft.Network/networkWatchers       northeurope

PS C:\Program Files\PowerShell\7>
```

How to Create Linux VM With Azure power Shell?

1. Use the New-AzVm cmdlet to create a VM.
   - Give the VM a name
   - Select a location close to you
   - Use "UbuntuLTS" for the image
   - Use the Get-Credential cmdlet and feed the results into the Credential parameter.
   - Add the -OpenPorts parameter and pass "22" as the port. This port will let us SSH into the machine.
   - Create a public IP address name. You'll use this name to create and find your static IP address to sign in to the machine.-PublicIpAddressName ”xxxxxxxx”
2. Create a username and password, then press Enter. PowerShell will start creating your VM.
3. The VM creation takes a few minutes to complete. After completion, you can query it and assign the VM object to a variable ($vm).
   $vm = (Get-AzVM -Name "testvm-eus-01" -ResourceGroupName …….)
4. Query the value to dump out the information about the VM. $vm.

How to Create Linux VM With Azure power Shell?

4. Query the value to dump out the information about the VM. $vm.

PowerShell Copy

```
$vm
```

You should see something like the following output:

PowerShell Copy

```
ResourceGroupName : [sandbox resource group name]
Id                : /subscriptions/00000000-0000-0000-0000-000000000000/resourceGroups/[sa
VmId              : 00000000-0000-0000-0000-000000000000
Name              : testvm-eus-01
Type              : Microsoft.Compute/virtualMachines
Location          : eastus
Tags              : {}
HardwareProfile   : {VmSize}
NetworkProfile    : {NetworkInterfaces}
OSProfile         : {ComputerName, AdminUsername, LinuxConfiguration, Secrets}
ProvisioningState : Succeeded
StorageProfile    : {ImageReference, OsDisk, DataDisks}
```

5. You can reach into complex objects through a dot (".") notation. For example, to see the properties in the `VMSize` object associated with the HardwareProfile section, run the following command:

PowerShell Copy

```
$vm.HardwareProfile
```

How to Create Linux VM With Azure power Shell?

6. Get your Public IP Address as :
   Get-AzPublicIpAddress -ResourceGroupName “your resource group” -Name ”xxxxxxx”
7. With IP Adress you can run SSH to connect to VM – till Ex:
   ssh bob@205.22.16.5

How to Delete VM With Azure power Shell?

To try out some more commands, let's delete the VM. We'll shut it down first:

PowerShell Copy

```
Stop-AzVM -Name $vm.Name -ResourceGroupName $vm.ResourceGroupName
```

Now, let's delete the VM by running the `Remove-AzVM` cmdlet:

PowerShell Copy

```
Remove-AzVM -Name $vm.Name -ResourceGroupName $vm.ResourceGroupName
```

Run this command to list all the resources in your resource group:

PowerShell Copy

```
Get-AzResource -ResourceGroupName $vm.ResourceGroupName | Format-Table
```

You should see a bunch of resources (disks, virtual networks, and so on) that all still exist.

Output Copy

```
Microsoft.Compute/disks
Microsoft.Network/networkInterfaces
Microsoft.Network/networkSecurityGroups
Microsoft.Network/publicIPAddresses
Microsoft.Network/virtualNetworks
```

The `Remove-AzVM` command *just deletes the VM*. It doesn't clean up any of the other resources. At this point, we'd likely just delete the resource group itself and be done with it. However, let's run through the exercise to clean it up manually. You should see a pattern in the commands.

1. Delete the network interface:

PowerShell Copy

```
$vm | Remove-AzNetworkInterface -Force
```

2. Delete the managed OS disks and storage account:

PowerShell Copy

```
Get-AzDisk -ResourceGroupName $vm.ResourceGroupName -DiskName $vm.StorageProfile.OSDisk.Na
```

3. Next, delete the virtual network:

PowerShell Copy

```
Get-AzVirtualNetwork -ResourceGroupName $vm.ResourceGroupName | Remove-AzVirtualNetwork -
```

4. Delete the network security group:

PowerShell Copy

```
Get-AzNetworkSecurityGroup -ResourceGroupName $vm.ResourceGroupName | Remove-AzNetworkSecu
```

5. And finally, delete the public IP address:

PowerShell Copy

```
Get-AzPublicIpAddress -ResourceGroupName $vm.ResourceGroupName | Remove-AzPublicIpAddress
```

PowerShell techniques

Variables

As you saw in the last unit, PowerShell supports variables. Use $ to declare a variable and = to assign a value. For example:

PowerShell Copy

```
$loc = "East US"
$iterations = 3
```

Variables can hold objects. For example, the following definition sets the **adminCredential** variable to the object returned by the **Get-Credential** cmdlet.

PowerShell Copy

```
$adminCredential = Get-Credential
```

To obtain the value stored in a variable, use the $ prefix and its name, as in the following:

PowerShell Copy

```
$loc = "East US"
New-AzResourceGroup -Name "MyResourceGroup" -Location $loc
```

Loops

PowerShell has several loops: **For**, **Do...While**, **For...Each**, and so on. The **For** loop is the best match for our needs, because we will execute a cmdlet a fixed number of times.

The core syntax is shown below; the example runs for two iterations and prints the value of **i** each time. The comparison operators are written **-lt** for "less than", **-le** for "less than or equal", **-eq** for "equal", **-ne** for "not equal", etc.

PowerShell Copy

```
For ($i = 1; $i -lt 3; $i++)
{
    $i
}
```

Parameters

When you execute a script, you can pass arguments on the command line. You can provide names for each parameter to help the script extract the values. For example:

PowerShell Copy

```
.\setupEnvironment.ps1 -size 5 -location "East US"
```

Inside the script, you'll capture the values into variables. In this example, the parameters are matched by name:

PowerShell Copy

```
param([string]$location, [int]$size)
```

You can omit the names from the command line. For example:

PowerShell Copy

```
.\setupEnvironment.ps1 5 "East US"
```

Inside the script, you'll rely on position for matching when the parameters are unnamed:

PowerShell Copy

```
param([int]$size, [string]$location)
```

Azure administrations

How to create and manage resources in Azure

Manage resources with Azure

ARM

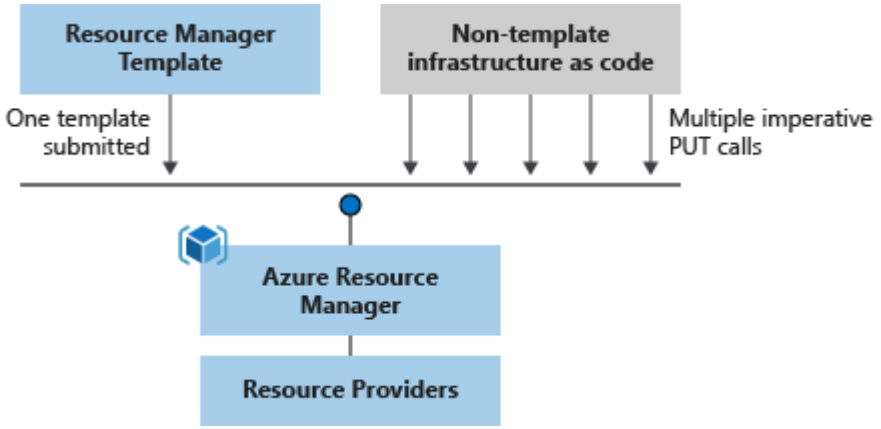
What is infrastructure as code?

- ***Infrastructure as code*** enables you to describe, through code, the infrastructure that you need for your application.
- With infrastructure as code, you can maintain both your application code and everything you need to deploy your application in a central code repository. The advantages to infrastructure as code are:
  - Consistent configurations
  - Improved scalability
  - Faster deployments
  - Better traceability

Let's Watch this Webinaruim about Infra structur as code here from June 2022

What is infrastructure as code?

- ***Infrastructure as code*** enables you to describe, through code, the infrastructure that you need for your application.
- **What is an ARM template?**
  - ARM templates are **JavaScript Object Notation (JSON) files** that **define the infrastructure and configuration for your deployment**. The template uses a **declarative syntax.** The declarative syntax is a way of building the structure and elements that outline what resources will look like without describing its control flow. Declarative syntax is different than **imperative syntax**, which uses commands for the computer to perform. Imperative scripting focuses on **specifying each step** in deploying the resources.



Manage resources with Azure Resource Manager template

- **Azure Resource Manager template** precisely defines all the Resource Manager resources in a deployment. You can deploy a Resource Manager template into a resource group as a single operation.
- **Template benefits:**
  - Templates improve **consistency**. Resource Manager templates provide a common language for you and others to describe your deployments. Regardless of the tool or SDK that you use to deploy the template, the structure, format, and expressions inside the template remain the same.
  - Templates help **express complex deployments**. Templates enable you to deploy multiple resources in the correct order. For example, you wouldn't want to deploy a virtual machine prior to creating an operating system (OS) disk or network interface. Resource Manager maps out each resource and its dependent resources, and creates dependent resources first. Dependency mapping helps ensure that the deployment is carried out in the correct order.
  - Templates **reduce manual, error-prone tasks**. Manually creating and connecting resources can be time consuming, and it's easy to make mistakes. Resource Manager ensures that the deployment happens the same way every time.

Manage resources with Azure Resource Manager template

- **Template benefits:**
  - **Templates are code**. Templates express your requirements through code. Think of a template as a type of Infrastructure as Code that can be shared, tested, and versioned similar to any other piece of software. Also, because templates are code, you can create a "paper trail" that you can follow. The template code documents the deployment. Most users maintain their templates under some kind of revision control, such as GIT. When you change the template, its revision history also documents how the template (and your deployment) has evolved over time.
  - Templates **promote reuse.** Your template can contain parameters that are filled in when the template runs. A parameter can define a username or password, a domain name, and so on. Template parameters enable you to create multiple versions of your infrastructure, such as staging and production, while still using the exact same template.
  - Templates are **linkable.** You can link Resource Manager templates together to make the templates themselves modular. You can write small templates that each define a piece of a solution, and then combine them to create a complete system.
  - Templates **simplify orchestration.** You only need to deploy the template to deploy all of your resources. Normally this would take multiple operations.

Manage resources with Azure Resource Manager template

- Azure Resource Manager templates are written in JSON, which allows you to express data stored as an object (such as a virtual machine) in text. A JSON document is essentially a collection of key-value pairs. Each key is a string, whose value can be:
    - A string
    - A number
    - A Boolean expression
    - A list of values
    - An object (which is a collection of other key-value pairs)

A Resource Manager template can contain sections that are expressed using JSON notation, but are not related to the JSON language itself:

JSON Copy

```
{
    "$schema": "http://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "",
    "parameters": {},
    "variables": {},
    "functions": [],
    "resources": [],
    "outputs": {}
}
```

What is Azure Resource Manager file structure?

Element	Description
schema	A required section that defines the location of the JSON schema file that describes the structure of JSON data. The version number you use depends on the scope of the deployment and your JSON editor.
contentVersion	A required section that defines the version of your template (such as 1.0.0.0). You can use this value to document significant changes in your template to ensure you're deploying the right template.
apiProfile	An optional section that defines a collection of API versions for resource types. You can use this value to avoid having to specify API versions for each resource in the template.
parameters	An optional section where you define values that are provided during deployment. These values can be provided by a parameter file, by command-line parameters, or in the Azure portal.
variables	An optional section where you define values that are used to simplify template language expressions.
functions	An optional section where you can define user-defined functions that are available within the template. User-defined functions can simplify your template when complicated expressions are used repeatedly in your template.
resources	A required section that defines the actual items you want to deploy or update in a resource group or a subscription.
output	An optional section where you specify the values that will be returned at the end of the deployment.

What are Azure Resource Manager template parameters?

JSON Copy

```
"parameters": {
    "<parameter-name>" : {
        "type" : "<type-of-parameter-value>",
        "defaultValue": "<default-value-of-parameter>",
        "allowedValues": [ "<array-of-allowed-values>" ],
        "minValue": <minimum-value-for-int>,
        "maxValue": <maximum-value-for-int>,
        "minLength": <minimum-length-for-string-or-array>,
        "maxLength": <maximum-length-for-string-or-array-parameters>,
        "metadata": {
        "description": "<description-of-the parameter>"
        }
    }
}
```

What are Azure Resource Manager template parameters?

Here's an **example** that illustrates **two parameters**: one for a virtual machine's **username**, and one for its **password**:

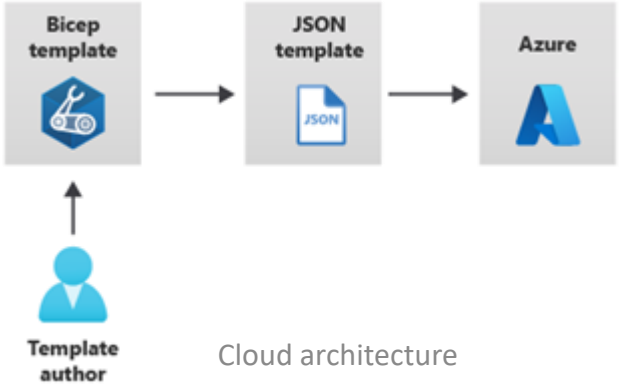
JSON Copy

```
"parameters": {
  "adminUsername": {
    "type": "string",
    "metadata": {
      "description": "Username for the Virtual Machine."
    }
  },
  "adminPassword": {
    "type": "securestring",
    "metadata": {
      "description": "Password for the Virtual Machine."
    }
  }
}
```

You're limited to **256 parameters** in a template. You can reduce the number of parameters by using objects that contain multiple properties.

What is Bicep templates?

- Azure Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. It provides concise syntax, reliable type safety, and support for code reuse.
- You can use **Bicep instead of JSON** to develop your Azure Resource Manager templates (ARM templates). The JSON syntax to create an ARM template can be verbose and require complicated expressions. Bicep syntax reduces that complexity and improves the development experience. Bicep is a transparent abstraction over ARM template JSON and doesn't lose any of the JSON template capabilities.



What is Bicep templates?

- Azure Bicep provides many improvements over JSON for template authoring, including:
  - **Simpler syntax:** Bicep provides a simpler syntax for writing templates. You can reference parameters and variables directly, without using complicated functions. String interpolation is used in place of concatenation to combine values for names and other items. You can reference the properties of a resource directly by using its symbolic name instead of complex reference statements. These syntax improvements help both with authoring and reading Bicep templates.
  - **Modules:** You can break down complex template deployments into smaller module files and reference them in a main template. These modules provide easier management and greater reusability.
  - **Automatic dependency management:** In most situations, Bicep automatically detects dependencies between your resources. This process removes some of the work involved in template authoring.
  - **Type validation and IntelliSense:** The Bicep extension for Visual Studio Code features rich validation and IntelliSense for all Azure resource type API definitions. This feature helps provide an easier authoring experience.
- Let's Check Azure Bicep !

How to Deploy an ARM template to Azure?

- You can deploy an ARM template to Azure in one of the following ways:
  - Deploy a **local** template.
  - Deploy a **linked** template.
  - Deploy in a **continuous deployment CD pipeline**.
- We foucs on deploy local templete using power shell

1 First, sign in to Azure by using the Azure CLI or Azure PowerShell.

Azure CLI | PowerShell

Azure PowerShell — Copy

```
Connect-AzAccount
```

2 Next, define your resource group. You can use an already-defined resource group or create a new one with the following command. You can obtain available location values from: `az account list-locations` (CLI) or `Get-AzLocation` (PowerShell). You can configure the default location using `az configure --defaults location=<location>`.

Azure CLI | PowerShell

Azure PowerShell — Copy

```
New-AzResourceGroup `
  -Name {name of your resource group} `
  -Location "{location}"
```

How to Deploy an ARM template to Azure?

- We foucs on deploy local templete using power shell
- Documentation path :
  - New-AzResourceGroupDeployment

How to add resources to ARM template to Azure?

- You'll need to know the **resource provider and its types of resources**.
- The syntax for this combination is in the form of *{resource-provider}/{resource-type}*.
- For example, to add a storage account resource to your template, you'll need the *Microsoft.Storage* resource provider.
- One of the types for this provider is *storageAccount*.
- So your resource type will be displayed as *Microsoft.Storage/storageAccounts*. You can use a list of resource providers for Azure services to find the providers you need.
- After you've defined the provider and resource type, you need to **understand the properties** for each resource type you want to use. For details, see Define resources in Azure Resource Manager templates.
- View the list in the left column to find the resource. Notice that the properties are sorted by API version.

- Mini Labb : Find Bicep or Jason resource properties for :
  - **Microsoft.Storage storageAccounts/blobServices 2021-09-01 , and**
  - **Microsoft.Compute virtualMachines 2022-03-01**

Storage account as ex. ARM template to Azure?

JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.1",
  "apiProfile": "",
  "parameters": {},
  "variables": {},
  "functions": [],
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "name": "learntemplatestorage123",
      "location": "westus",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "StorageV2",
      "properties": {
        "supportsHttpsTrafficOnly": true
      }
    }
  ],
  "outputs": {}
}
```

Microsoft.Storage/storageAccounts object

Name	Type	Required	Value
name	string	Yes	The name of the storage account within the specified resource group. Storage account names must be between 3 and 24 characters in length and use numbers and lower-case letters only.
type	enum	Yes	Microsoft.Storage/storageAccounts
apiVersion	enum	Yes	2019-06-01
sku	object	Yes	Required. Gets or sets the SKU name. - Sku object
kind	enum	Yes	Required. Indicates the type of storage account. - Storage, StorageV2, BlobStorage, FileStorage, BlockBlobStorage
location	string	Yes	Required. Gets or sets the location of the resource. This will be one of the supported and registered Azure Geo Regions (e.g. West US, East US, Southeast Asia, etc.). The geo region of a resource cannot be changed once it is created, but if an identical geo region is specified on update, the request will succeed.
tags	object	No	Gets or sets a list of key value pairs that describe the resource. These tags can be used for viewing and grouping this resource (across resource groups). A maximum of 15 tags can be provided for a resource. Each tag must have a key with a length no greater than 128 characters and a value with a length no greater than 256 characters.
identity	object	No	The identity of the resource. - Identity object
properties	object	Yes	The parameters used to create the storage account. - StorageAccountPropertiesCreateParameters object
resources	array	No	encryptionScopes privateEndpointConnections managementPolicies

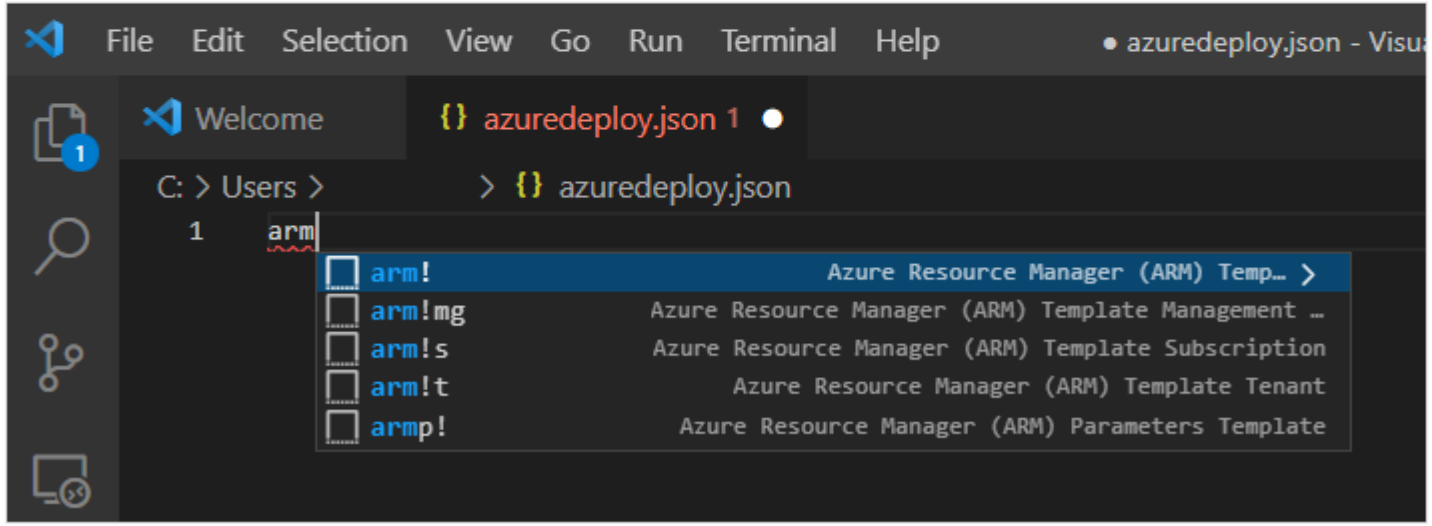
Lab#13 – Deploy Azure Resource manager

1. Add Azure Resource manager (ARM) Tools to your Visual studio code, see how from here.
2. Create an ARM template :

    1. Open Visual Studio Code, and create a new file called *azuredeploy.json*.

    2. The Visual Studio Code ARM template extension comes configured with snippets to help you develop templates. Let's start by adding a blank template. On the first line of the file, enter *arm*.

    3. The VS Code automatically displays several potential choices that start with **arm!**. Select the **Azure Resource Manager (ARM) template**. VS Code automatically processes the schemas and languages for your template.



Lab#13 – Deploy Azure Resource manager-ctn

2. Create an ARM template :

Your file now looks like this:

JSON Copy

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "functions": [],
  "variables": {},
  "resources": [],
  "outputs": {}
}
```

Notice that this file has all of the sections of an ARM template that we described in the previous unit.

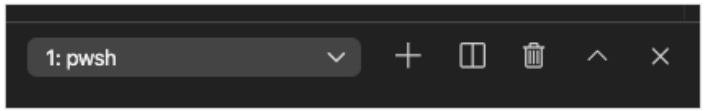
4. Save the changes to the file by pressing `Ctrl+S`.

Lab#13 – Deploy Azure Resource manager-ctn

3. Deploy the ARM template to Azure:

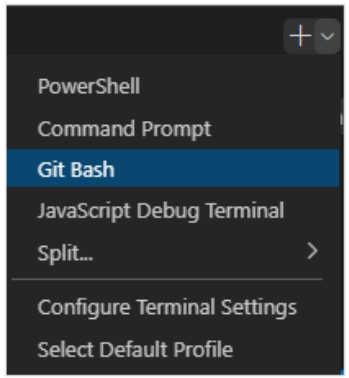
To deploy this template to Azure, you need to sign in to your Azure account from the Visual Studio Code terminal. Be sure you have installed Azure PowerShell Tools from the VS Code Extensions, and sign in.

1. In the command bar, select **Terminal** > **New Terminal** to open a PowerShell window.
2. If the command bar of the terminal window shows **PowerShell**, you have the right shell to work from, and you can skip to the next section.



   a. If not, select the down arrow and in the dropdown list select PowerShell. If that option is missing, then select **Select Default Profile**.

   b. In the input field, scroll down and select **PowerShell**.



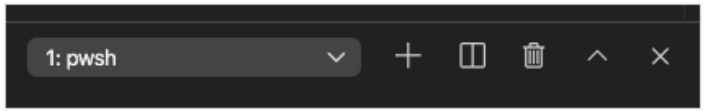
   c. Select **Terminal** > **New Terminal** to open a PowerShell terminal window.

Lab#13 – Deploy Azure Resource manager-ctn

3. Deploy the ARM template to Azure:

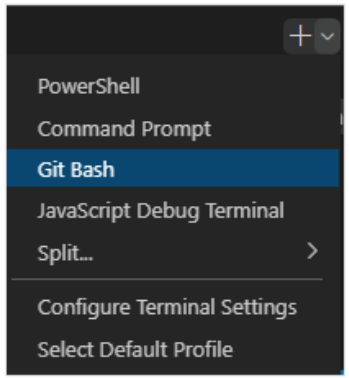
To deploy this template to Azure, you need to sign in to your Azure account from the Visual Studio Code terminal. Be sure you have installed Azure PowerShell Tools from the VS Code Extensions, and sign in.

1. In the command bar, select **Terminal** > **New Terminal** to open a PowerShell window.
2. If the command bar of the terminal window shows **PowerShell**, you have the right shell to work from, and you can skip to the next section.



   a. If not, select the down arrow and in the dropdown list select PowerShell. If that option is missing, then select **Select Default Profile**.

   b. In the input field, scroll down and select **PowerShell**.



   c. Select **Terminal** > **New Terminal** to open a PowerShell terminal window.

Lab#13 – Deploy Azure Resource manager-ctn

4. Sign in to Azure by power shell :

    1. From the terminal in Visual Studio Code, run the following command to sign in to Azure. A browser opens so you can sign in to your account.

```
Azure PowerShell                                                    Copy
Connect-AzAccount
```

5. Set the default subscription for all PowerShell commands in this session:

    1. Run the following command to obtain your subscription(s) and their ID(s).

```
Azure PowerShell                                                    Copy
Get-AzSubscription
```

    2. Run the following command, replacing *{Your subscription ID}* with the one you copied in the previous step to change your active subscription to the Concierge Subscription.

```
Azure PowerShell                                                    Copy
$context = Get-AzSubscription -SubscriptionId {Your subscription ID}
Set-AzContext $context
```

Lab#13 – Deploy Azure Resource manager-ctn

5. Set the default subscription for all PowerShell commands in this session:

3. Run the following command to let the default resource group be the resource group created for you in the sandbox environment. This action lets you omit that parameter from the rest of the Azure PowerShell commands in this exercise.

Azure PowerShell Copy

```
Set-AzDefault -ResourceGroupName [sandbox resource group name]
```

6. Deploy the template to Azure:

Azure PowerShell Copy

```
$templateFile="azuredeploy.json"
$today=Get-Date -Format "MM-dd-yyyy"
$deploymentName="blanktemplate-"+"$today"
New-AzResourceGroupDeployment `
  -Name $deploymentName `
  -TemplateFile $templateFile
```

The top section of the preceding code sets Azure PowerShell variables, which includes the path to the deployment path and the name of the deployment. Then, the `New-AzResourceGroupDeployment` command deploys the template to Azure. Notice that the deployment name is `blanktemplate` with the date as a suffix.

Lab#13 – Deploy Azure Resource manager-ctn

7. Add a resource to the ARM template :
   In the previous task, we created a **blank template and deploy it**. Now, you're ready to deploy an **actual resource**.

1. In the *azuredeploy.json* file in Visual Studio Code, place your cursor inside the brackets in the resources block `"resources":[],`.

2. Enter *storage* inside the brackets. A list of related snippets appears. Select **arm-storage**.


```
  "resources": [
    {
      "name": "storageaccount1",
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "tags": {
        "displayName": "storageaccount1"
      },
      "location": "[resourceGroup().location]",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

Lab#13 – Deploy Azure Resource manager-ctn

7. Add a resource to the ARM template :
   In the previous task, we created a **blank template and deploy it**. Now, you're ready to deploy an **actual resource**.

JSON Copy

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {},
  "functions": [],
  "variables": {},
  "resources": [
    {
      "name": "storageaccount1",
      "type": "Microsoft.Storage/storageAccounts",
      "apiVersion": "2019-06-01",
      "tags": {
        "displayName": "storageaccount1"
      },
      "location": "[resourceGroup().location]",
      "kind": "StorageV2",
      "sku": {
        "name": "Premium_LRS",
        "tier": "Premium"
      }
    }
  ],
  "outputs": {}
}
```

Lab#13 – Deploy Azure Resource manager-ctn

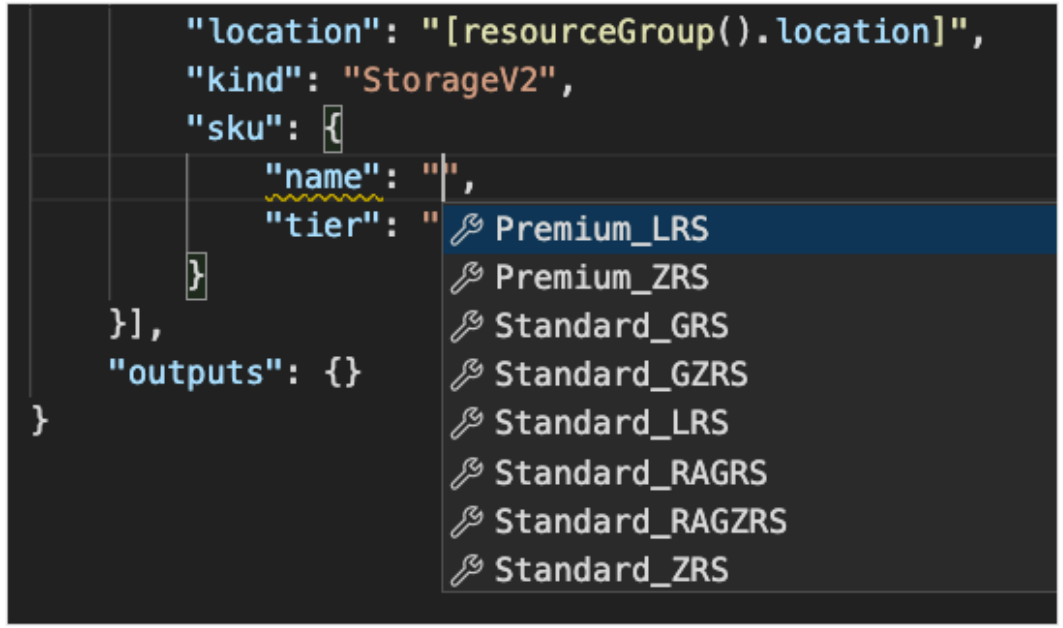
7. Add a resource to the ARM template :
   In the previous task, we created a **blank template and deploy it**.
   Now, you're ready to deploy an **actual resource**.

Values that you should edit are highlighted in the new section of your file and can be navigated by pressing the `Tab` key.

Notice the `tags` and `location` attributes are filled in. The `location` attribute uses a function to set the location of the resource to the location of the resource group.

3. Change the values of the resource *name* and *displayName* to something unique, (for example, **learnexercise12321**). This name must be unique across all of Azure, so choose something unique to you.

4. Change the value of the sku *name* from **Premium_LRS** to **Standard_LRS**. Change the value of *tier* to **Standard**. Notice that Visual Studio Code gives you the proper choices for your attribute values in IntelliSense. Delete the default value including the quotation marks, and enter quotation marks to see this work.



5. The location of the resource is set to the location of the resource group where it will be deployed. Leave the default here.

6. Save the file.

Lab#13 – Deploy Azure Resource manager-ctn

8. Deploy the updated ARM template

Run the following Azure PowerShell commands in the terminal. This snippet is the same code you used previously, but the name of the deployment is changed.

Azure PowerShell Copy

```
$templateFile="azuredeploy.json"
$today=Get-Date -Format "MM-dd-yyyy"
$deploymentName="addstorage-"+"$today"
New-AzResourceGroupDeployment `
  -Name $deploymentName `
  -TemplateFile $templateFile
```

Lab#13 – Deploy Azure Resource manager-ctn

8. More to do and to know

ARM Series #1: Introduction

#2: Creating Your First Template

#3: Parameters

#4: Template Functions

#5: Variables

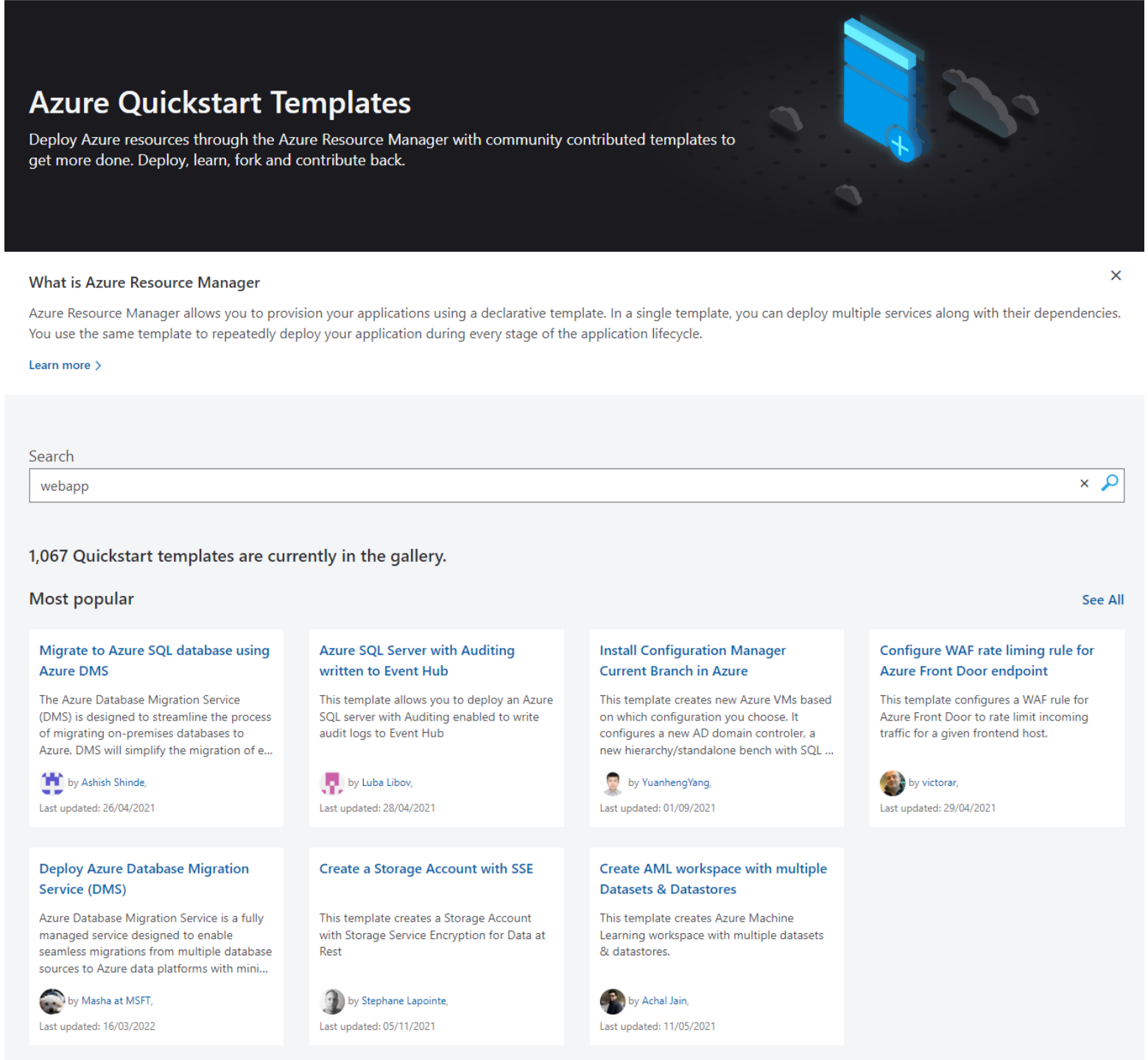
#6: Template Output

#7: Controlling Deployment

#8: Linked and Nested Templates

What are QuickStart templates?

- Microsoft Azure as 1000 + ready templates for all needs to deploy!
- You can find them into [Azure Quickstart Templates]



Solutions KPI och ROI

What is KPI and ROI?

Solution's KPI

As architects, nearly everything we do is associated with a project. As a project-centric business, understanding which key performance indicators (KPIs) are meaningful is only half the battle.

What is KPI ?

Imagine you're driving in your car and you look down at the fuel gauge on the dashboard. In most newer cars, not only would you see that you are low on fuel, but that you have only 170 KMleft. **That's really helpful**! You now know that you ***need to find a gas station within 170KM or you will be stranded on the road***. Your fuel **KPI is critical and real-time**.

Now, imagine if the fuel level was only *updated weekly*. How comfortable would you be taking a road trip if you didn't have a totally accurate reading of the fuel in the tank (or charge in the battery for our electric vehicle readers)?

Even if your car had a fuel gauge that was only updated weekly, the car has one advantage over your projects. When it runs out of gas, it stops dead in the road. You can't go another inch. There is no denying you have run out of gas. *The same can't be said for your Company or customer*. Once we have used up the fees or budget for a project, our employees don't stop working and sit motionless at their desk waiting for a tow truck to arrive.

This example demonstrates that knowing the KPIs is only as useful as your ability to believe **that the data is accurate and real-time**. Otherwise, you will ***be making decisions on conditions that existed a week or more in the past.*** In situations where the data is showing a problem, that means you are already a week or more late in your ability to address it.

Solution's KPI

As architects who are expected to create **sustainable architecture** for your clients, you owe it to yourself to create a sustainable firm **first by implementing a system that gives you real-time KPIs**. Considering every firm in the world is now operating as a distributed workforce, it is essential that you invest in a native-cloud system

8 Essential KPIs for Architects to Measure Firm Growth

Once you have a great cloud-based system in place, then we can look at the KPIs the system should provide to ensure that your firm is performing as planned.

#1: OVERHEAD MULTIPLIER

The overhead multiplier **is the cost of non-project-related expenditures** (such as indirect expenses including indirect labor) expressed as a percentage of total direct labor.

This is one of the most ***vital KPIs because it's required to accurately determine a firm's profitability***. A lower overhead multiplier means a higher profit margin and vice versa. Some firms choose to calculate overhead multipliers every *quarter*, while others do it *annually*. To calculate the overhead multiplier, follow the formula below:

Overhead Multiplier = Total Indirect Expenses / Total Direct Labor

If you want to reduce the overhead, you'll need to manage indirect expenses more carefully. Aim to keep your overhead rate at or below 175% of total direct labor (also expressed as 1.75). If your overhead multiplier is greater than 1.75, you should take immediate action to resolve it.

Solution's KPI -8 Essential KPIs for Architects to Measure Firm Growth

#2: UTILIZATION RATE "utnyttjandegrad"

The utilization rate is the **percentage of hours spent on billable projects vs. the total number of hours worked**.

Utilization is important for firms that charge their time to clients and need to maximize the productive time of their employees because it helps determine the overall productive use of an individual or firm.

The utilization rate **illustrates the efficiency and overall performance** of an employee by comparing an employee's billable and non-billable hours.

For example, if an employee logs 40 hours a week and 30 of those hours are billable, their utilization rate is 75%. To calculate utilization rate, follow the formula below:

Utilization Rate = Billable Hours Worked / Total Available Work Hours

If your rate is too high, you likely need to add more resources. When your rate is too low, it could mean you are not bringing in enough work. With CORE Architect, your firm is able to look directly into an employee's record and instantly see their utilization for any time period.

Solution's KPI -8 Essential KPIs for Architects to Measure Firm Growth

#3: EFFECTIVE COST RATE "EFFEKTIV KOSTNADSPRIS"

An effective cost rate is the **actual cost of each person's employment based on the work they do**.

This metric is important because it gives you a true picture of what an employee costs on an hourly basis, and it can help you determine what you should bill for each employee's services to stay profitable.

Costs vary from job to job and project to project. Looking at the individual costs can give you valuable information about that job, but what about your employee's overall performance? The effective cost rate shows you the average cost of an employee.

It is not enough to simply calculate the effective cost rate by combining your firm's overhead multiplier with the employee's hourly salary. For example, many firms would simply take an employee with an hourly salary of $60, where the firm has an overhead multiplier or 1.75 and think that the effective cost rate for the employee is $105 (60 x 1.75 = 105).

We take this a step further by looking at the costs of the employee based on their utilized time. This is hugely important to think about: for every hour they perform utilized work, what do they actually cost your firm.

Effective Cost Rate = Cost Amount / Total Actual Hours Billed where ,

Cost Amount = (Actual Hours x Cost Rate) of billed time

If your effective cost rate is high, it's a sign you're paying too much or your employee is spending too much time on non-billable work.

Solution's KPI -8 Essential KPIs for Architects to Measure Firm Growth

#3: EFFECTIVE COST RATE "EFFEKTIV KOSTNADSPRIS"

An effective cost rate is the **actual cost of each person's employment based on the work they do**.

This metric is important because it gives you a true picture of what an employee costs on an hourly basis, and it can help you determine what you should bill for each employee's services to stay profitable.

Costs vary from job to job and project to project. Looking at the individual costs can give you valuable information about that job, but what about your employee's overall performance? The effective cost rate shows you the average cost of an employee.

It is not enough to simply calculate the effective cost rate by combining your firm's overhead multiplier with the employee's hourly salary. For example, many firms would simply take an employee with an hourly salary of $60, where the firm has an overhead multiplier or 1.75 and think that the effective cost rate for the employee is $105 (60 x 1.75 = 105).

We take this a step further by looking at the costs of the employee based on their utilized time. This is hugely important to think about: for every hour they perform utilized work, what do they actually cost your firm.

Effective Cost Rate = Cost Amount / Total Actual Hours Billed where ,

Cost Amount = (Actual Hours x Cost Rate) of billed time

If your effective cost rate is high, it's a sign you're paying too much or your employee is spending too much time on non-billable work.

Solution's KPI -8 Essential KPIs for Architects to Measure Firm Growth

#4: NET MULTIPLIER "NETTOMULTIPLIERARE"

The net multiplier is the **ratio of net operating revenue to total direct labor**.

If you think of direct labor as an investment, the net multiplier is a measure of your return on this investment.

It shows you how many dollars of revenue you generate for every dollar you spend on direct labor and measures your actual performance. To calculate net multiplier, follow the equation below:

Net Multiplier = Net Operating Revenue / Total Direct Labor

The net multiplier is a good gauge of your firm's financial health. Your net operating revenue should be greater than total direct labor. If total direct labor is greater than net operating revenue, you should investigate why your costs are so high.

#5: WORK IN PROGRESS (WIP) "PÅGÅENDE ARBETE"

Work in progress (WIP) **refers to the value of the billable hours and expenses that a firm hasn't billed yet**.

WIP is valuable in that it can help you forecast expected revenue. It can also help *managers understand how far along work is, and firm owners can use WIP, along with their accounts receivables (AR),* as a booster when applying for a commercial line-of-credit. Some firms track WIP as an asset on the balance sheet, and once invoiced, it shows up as revenue on the income statement.

You want to make sure you have a consistent cash flow. If one week's WIP looks significantly lower than others, that could be a signal that not everyone is making the most of their time.

Some firms will require a more sophisticated calculation to determine WIP since not all projects bill hourly. It is quite normal for firms to have contracts that are based on a fixed fee or some other form of a stipulated sum (i.e. a percentage of the cost of construction). In situations like this, projects that are subject to this form of contract should look at their earned value (EV).

Solution's KPI -8 Essential KPIs for Architects to Measure Firm Growth

#6: AGED ACCOUNTS RECEIVABLE "ÅLDRADE KUNDFORDRINGAR"

Accounts receivable aging **shows unpaid customer invoices and unused credit memos by date ranges.** This is a valuable tool when trying to determine which invoices are overdue for payment. Calculating your average aged accounts receivable will show you the average number of days it takes to get paid from the invoice date. To calculate your aged accounts receivable, use the following formula:

Aged Accounts Receivable = Annual Average Accounts Receivable / (Net Operating Revenue / 365 Days

Architecture firms should do their best to collect all outstanding invoices within 30 days of the invoice date. If your average aged accounts receivable is greater than 30 days, it might be a signal that it's time to reexamine your invoicing process and develop a better way of collecting what's owed.

#7: PROFIT-TO-EARNINGS RATIO "RESULTATFÖRHÅLLANDEN"

The profit-to-earnings **ratio indicates a firm's effectiveness in completing projects profitably.**

The profit-to-earnings ratio is determined by dividing the profit (after expenses and salaries have been accounted for but before non-salary distributions and taxes) by the net operating revenue. To calculate your profit-to-earnings ratio, use the following formula:

Profit-to-Earnings Ratio = Profit Before Distributions and Taxes / Net Operating Revenue

The higher the number, the more profitable your firm is. If your profit-to-earnings ratio is low, you could be spending too much internally.

Solution's KPI -8 Essential KPIs for Architects to Measure Firm Growth

#8: NET REVENUE PER EMPLOYEE

The net revenue per employee **shows, on average, how much revenue each of your employees generates.**
This metric is a meaningful analytical tool because it measures how effectively your firm utilizes its employees. It can also help you forecast a more realistic range for future annual net operating revenue. To calculate your net revenue per employee, use the following formula:

Net Revenue Per Employee = Company's Net Revenue / Current Number of Employees

Compare your net revenue per employee against that of other companies in the same industry, or use it to look at changes within your company. If your average goes down, this means you're generating less revenue per employee.
For example, imagine you had 10 employees and made $1,000,000 in revenue last year. Then this year, you have 20 employees, but you only made $1,500,000 in revenue. You have twice the workforce, but your revenue increased by only 50%. This is clearly a problem.

ROI?

Return on investment (ROI) is a financial metric that is widely used to measure the **probability of gaining a return from an investment." sannolikheten att få avkastning på en investering."**
It is a ratio that compares the gain or loss from an investment relative to its cost. It is as useful in evaluating the potential return from a stand-alone investment as it is in comparing returns from several investments.

KEY TAKEAWAYS

- Return on investment (ROI) is an **approximate measure of an investment's profitability**.
- ROI has a wide range of applications; it can be used to measure the profitability of a stock investment, when deciding whether or not to invest in the purchase of a business or evaluate the results of a real estate transaction.
- ROI is relatively easy to calculate and understand, and its simplicity means that it is a standardized, universal measure of profitability.
- One disadvantage of ROI is that it *doesn't account for how long an investment is held*; so, a profitability measure that incorporates the holding period may be more useful for an investor that wants to compare potential investments.

How to calculate ROI ?

First method:

$$\text{ROI} = \frac{\text{Net Return on Investment}}{\text{Cost of Investment}} \times 100\%$$

Second method:

$$\text{ROI} = \frac{\text{FVI} - \text{IVI}}{\text{Cost of Investment}} \times 100\%$$

where:

$\text{FVI} = \text{Final value of investment}$

$\text{IVI} = \text{Initial value of investment}$

Return on Investment (ROI) Example

Assume an investor bought 1,000 shares of the hypothetical company Worldwide Wickets Co. at $10 per share. One year later, the investor sold the shares for $12.50. The investor earned dividends of $500 over the one-year holding period. The investor also spent a total of $125 on trading commissions in order to buy and sell the shares.

The ROI for this investor can be calculated as follows:

$$\text{ROI} = \frac{(\$12.50 - \$10) \times 1000 + \$500 - \$125}{\$10 \times 1000} \times 100$$

$$= 28.75\%$$

When ROI calculations yield a **positive figure**, it means that net returns are in the **black** (because total returns exceed total costs). Alternatively, **when ROI calculations yield a negative figure, it means that net returns are in the red because total costs exceed total returns**. (In other words, this investment produces a loss.)

Finally, to calculate ROI with the highest degree of accuracy, total returns and total costs should be considered. For an apples-to-apples comparison between competing investments, annualized ROI should be considered.

Application Insights overview

What are Application Insights?

Application Insights is a feature of Azure Monitor that **provides** extensible **application performance management (APM)** and **monitoring for live web apps**.

- ***Developers and DevOps professionals*** can use Application Insights to:
  - Automatically detect performance anomalies.
  - Help diagnose issues by using powerful analytics tools.
  - See what users do with apps.
  - Help continuously improve app performance and usability.

Application Insights:

- Supports a wide variety of platforms, including .NET, Node.js, Java, and Python.
- Works for apps hosted on-premises, hybrid, or on any public cloud.
- Integrates with DevOps processes.
- Has connection points to many development tools.
- Can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center.

How Application Insights works?

To use Application Insights, you either:

- install a small instrumentation package (SDK) in your app,
- or enable Application Insights by using the Application Insights agent.

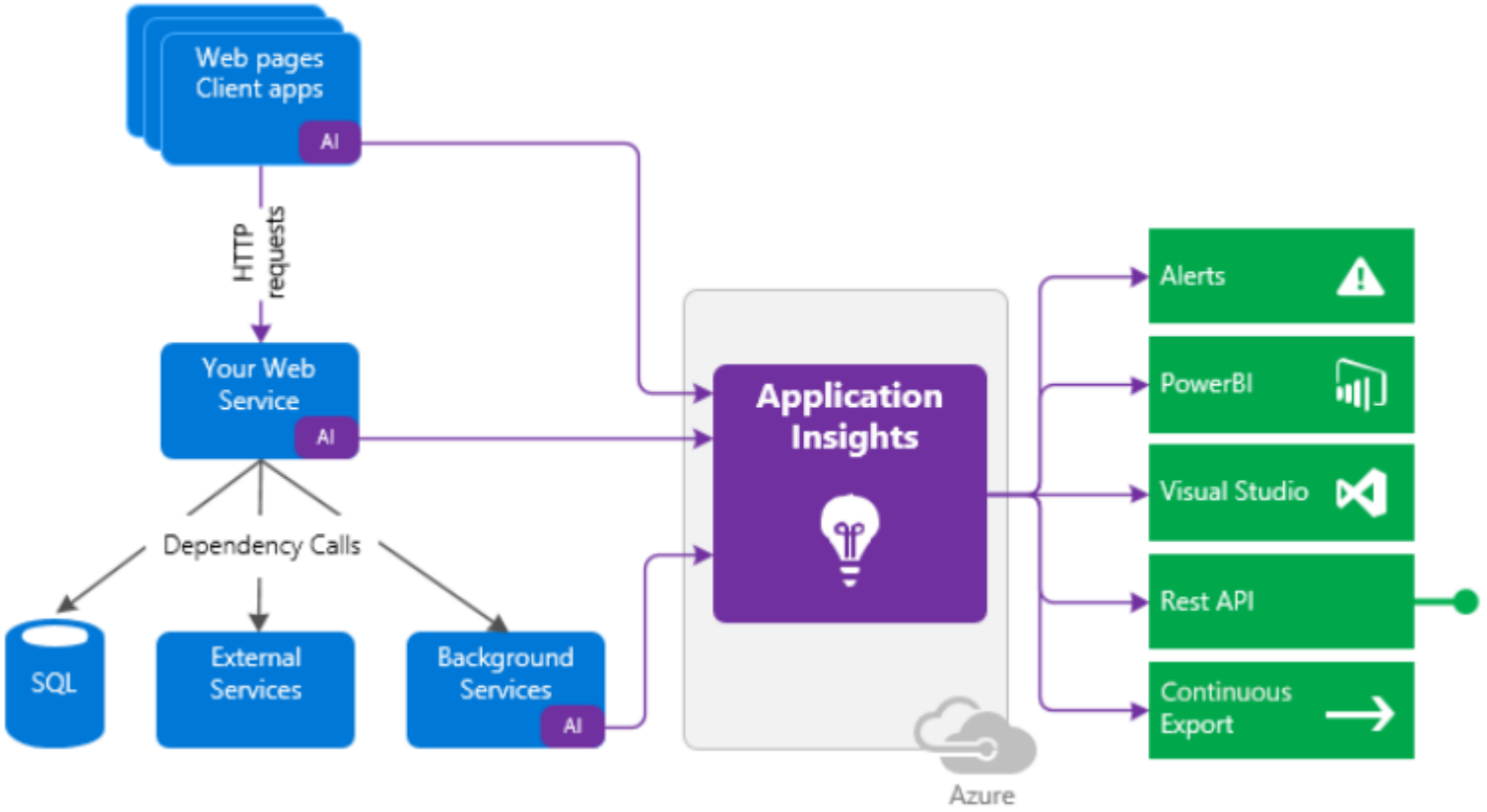
For languages and platforms that support the Application Insights agent, see Supported languages.

The app and its components **don't have to be hosted in Azure**.

The instrumentation **monitors** your app and **directs the telemetry data to an Application Insights** resource by using a **unique instrumentation key**. The impact on your app's performance is small. Tracking calls are non-blocking, and are batched and sent in a separate thread.

You can **pull in telemetry** like performance counters, Azure diagnostics, or Docker logs from host environments. You can also **set up web tests** that periodically send synthetic requests to your web service. All these telemetry streams are integrated into Azure Monitor. In the Azure portal, you can apply powerful analytics and search tools to the raw data.

How Application Insights works?



What Application Insights monitors

Application Insights helps development teams understand app **performance** and **usage**.
Application Insights monitors:

- Request rates, response times, and failure rates. It finds out which pages are most popular, at what times of day, and where users are. See which pages perform best. If response times and failure rates are high when there are more requests, there might be a resourcing problem.
- Dependency rates, response times, and failure rates, to show whether external services are slowing down performance
- Exceptions
  Analyze the aggregated statistics or pick specific instances and drill into the stack trace and related requests. Application Insights reports both server and browser exceptions.
- Page views and load performance reported by users' browsers
- AJAX calls from web pages, including rates, response times, and failure rates
- User and session counts
- Performance counters from Windows or Linux server machines, such as CPU, memory, and network usage
- Host diagnostics from Docker or Azure
- Diagnostic trace logs from apps, so you can correlate trace events with requests
- Custom events and metrics in client or server code that track business events, like items sold

What Application Insights monitors

Application Insights helps development teams understand app **performance** and **usage**.
Application Insights monitors:

- Request rates, response times, and failure rates. It finds out which pages are most popular, at what times of day, and where users are. See which pages perform best. If response times and failure rates are high when there are more requests, there might be a resourcing problem.
- Dependency rates, response times, and failure rates, to show whether external services are slowing down performance
- Exceptions
  Analyze the aggregated statistics or pick specific instances and drill into the stack trace and related requests. Application Insights reports both server and browser exceptions.
- Page views and load performance reported by users' browsers
- AJAX calls from web pages, including rates, response times, and failure rates
- User and session counts
- Performance counters from Windows or Linux server machines, such as CPU, memory, and network usage
- Host diagnostics from Docker or Azure
- Diagnostic trace logs from apps, so you can correlate trace events with requests
- Custom events and metrics in client or server code that track business events, like items sold

What Application Insights monitors

There are many ways to explore Application Insights telemetry. For more information, see the following articles:

- Smart detection in Application Insights
- Create, view, and manage log alerts using Azure Monitor.
- Application Map: Triage distributed applications
- Profile live Azure App Service apps with Application Insights
- Usage analysis with Application Insights
- Use Search in Application Insights
- Advanced features of the Azure metrics explorer
- Application Insights overview dashboard
- Live Metrics Stream: Monitor and diagnose with one-second latency
- Log queries in Azure Monitor
- Debug your applications with Application Insights in Visual Studio
- Debug snapshots on exceptions in .NET apps
- Feed Power BI from Application Insights
- Use the Application Insights REST API to build custom solutions
- Export telemetry from Application Insights