1) TYPES OF CMDLETS?

A **cmdlet** (pronounced "command-let") is a lightweight Windows PowerShell script that performs a single function. ... Although Windows PowerShell includes more than two hundred basic core **cmdlets**, administrators can also write their own **cmdlets** and share them.

Az module features

- Az is a replacement for AzureRM and AzureRM.Netcore.
- Az runs on PowerShell 5.1 and PowerShell Core.
- Az is always up to date with the latest tooling for Azure services.
- Az ships in Cloud Shell.
- Az shortens and normalizes cmdlet names. All cmdlets use "Az" as their noun prefix.
- Az will simplify and normalize module names. Data plane and management plane cmdlets for each service will use the same Az module.
- Az ships with new cmdlets to enable script compatibility with AzureRM (Enable/Disable-AzureRmAlias).

Supported platforms

- PowerShell 5.1 – Windows 7 or greater with .Net Framework 4.7.2 or greater
- installed

2) POWERSHELL EXECUTION POLICIES?

The PowerShell execution policies are as follows:

AllSigned

- Scripts can run.
- Requires that all scripts and configuration files be signed by a trusted publisher, including scripts that you write on the local computer.
- Prompts you before running scripts from publishers that you haven't yet classified as trusted or untrusted.
- Risks running signed, but malicious, scripts.

Bypass

- Nothing is blocked and there are no warnings or prompts.
- This execution policy is designed for configurations in which a PowerShell script is built in to a larger application or for configurations in which PowerShell is the foundation for a program that has its own security model.

Default

- Sets the default execution policy.
- **Restricted** for Windows clients.
- **RemoteSigned** for Windows servers.

RemoteSigned

- The default execution policy for Windows server computers.
- Scripts can run.

- Requires a digital signature from a trusted publisher on scripts and configuration files that are downloaded from the internet which includes email and instant messaging programs.
- Doesn't require digital signatures on scripts that are written on the local computer and not downloaded from the internet.
- Runs scripts that are downloaded from the internet and not signed, if the scripts are unblocked, such as by using the Unblock-File cmdlet.
- Risks running unsigned scripts from sources other than the internet and signed, but malicious, scripts.

Restricted

- The default execution policy for Windows client computers.
- Permits individual commands, but will not run scripts.
- Prevents running of all script files, including formatting and configuration files (.ps1xml), module script files (.psm1), and PowerShell profiles (.ps1).

Undefined

- There is no execution policy set in the current scope.
- If the execution policy in all scopes is **Undefined**, the effective execution policy is **Restricted**, which is the default execution policy.

Unrestricted

- The default execution policy for non-Windows computers and cannot be changed.
- Unsigned scripts can run. There is a risk of running malicious scripts.
- Warns the user before running scripts and configuration files that are downloaded from the internet.

3) POWERSHELL TYPES OF BRACKETS?

Powershell supports three types of brackets.

- **Parenthesis brackets. – ()**
- **Braces brackets. – {}**
- **Square brackets. – []**

Parenthesis brackets

This type of brackets is used to

- pass arguments
- enclose multiple set of instructions
- resolve ambiguity
- create array

Example

```
> $array = @("item1", "item2", "item3")

> foreach ($element in $array) { $element }
item1
item2
```

item3

Braces brackets

This type of brackets is used to

- enclose statements
- block commands

4) WRITE-OUTPUT VS WRITE-HOST

Write-Output sends the output to the pipeline. From there it can be piped to another cmdlet or assigned to a variable. Write-Host sends it directly to the console.

```
$a = 'Testing Write-OutPut'  | Write-Output
$b = 'Testing Write-Host' | Write-Host

Get-Variable a,b
```

Outputs:

```
Testing Write-Host

Name                 Value
----                 -----
a                    Testing Write-OutPut
b
```

In Windows PowerShell 2.0, a new parameter attribute is introduced. This parameter attribute uses the **Parameter** keyword, and sets the **Mandatory** attribute to **$true**. The syntax to define a particular parameter as mandatory is shown here:

```
[Parameter(Mandatory=$true)]
```

A revision to the **Get-DiskInformation** function so that it uses the new **Mandatory** parameter attribute allows us to remove the **if…throw** construction as shown here:

```
if(-not($drive)) { Throw “You must supply a value for -drive” }
```

In addition to removing the **if…throw** construction, the only other change involves adding **[Parameter(Mandatory=$true)]** to the line immediately above the **[string]$drive** line. The revised **Get-DiskInformation** function is shown here:

```
Function Get-DiskInformation
{
 Param(
  [Parameter(Mandatory=$true)]
  [string]$drive,
  [string]$computerName = $env:computerName
) #end param
```

```
Get-WmiObject -class Win32_volume -computername $computername -filter “DriveLetter = ‘$drive’”
} #end function Get-DiskInformation
```

```
$str1 = "test"
$str2 = "test"

IF (Compare-Object $str1 $str2)

{
   Write-Host "strings are different"
}
ELSE
{
   Write-Host "strings are equal"
}
```

```
IF ($str1 -eq $str2) {...
```

```
$string1 = “Hello World”
    $string2 = “hello world”
$string1 -eq $string2 # Returns $true
    $string1 -ceq $string2 # Returns $false
```

PowerShell can be a little inconsistent and you have to remember which commands are case sensitive and which aren’t. Sometimes it’s obvious as in the case above but often you’ll just use the –eq rather than the –ceq operator. You could just force both stings to the same case using .ToLower or .ToUpper, e.g. $string1.ToLower() –eq $string2.ToLower() but using the –ceq makes more sense I think.

5) COMPARE 2 STRINGS IN POWERSHELL AND HOW WE WILL SPLIT THE VALUES?

$text = "Windows PowerShell - Hello world!" $split = $text.split("-") echo $split[0] echo $split[1]

First, a string is created named "$text" which holds the text to be split. The string is then split using the PowerShell "Split" function, passing in the character the string is to be split by. In the code above, the string is split at the location of the hyphen. This creates an array in "$split" with two elements. The first element contains all the text up to the hyphen and the second element contains all the text after the hyphen. The contents of the two elements are then displayed to verify the command has worked.

Step

Run the script in PowerShell by typing ".\split.ps1" and the display will show the output below,

6) WHAT IS THE USE OF PIPE SYMBOL?

Get-ChildItem -Path C:\WINDOWS\System32 | Out-Host –Paging

1 get-process | sort -Property "VirtualMemorySize" –Descending

Pipeline Fundamentals

In **PowerShell**, the vertical bar ( ) **is the pipe** symbol. This tells **PowerShell** that you want to take the output of one command and pass it as the input (or **pipe** it) to the next command. ... **PowerShell** has a command called Sort-Object that **will** sort any type of object, typically on a given property

7) THE $_ SYMBOL USE?

The $_ symbol seems to be causing confusion from some recent forum questions I've seen.

$_ represents the current object on the pipeline – if you want to know why $_ was chosen you'll have to read PowerShell in Action!

You can use $_ in a number of situations – in commands that perform an action on every object (or selected objects) on the pipeline. Here's some of the commoner usages:

In the early versions of PowerShell you used it in the filter script of Where-Object

```
Get-Process | Where-Object -FilterScript {$_.CPU -gt 50}
```

This is more usually written as

```
Get-Process | Where {$_.CPU -gt 50}
```

As of PowerShell v3 if you are filtering on ONE property you can simplify the syntax

```
Get-Process | Where CPU -gt 50
```

which is a truncated version of

```
Get-Process | Where -Property CPU -gt -Value 50
```

When you write it like this its obvious what is happening. Get in the habit of thinking of the syntax in this manner even if you write in the shortened form

If you need to filter on TWO or more properties you have to use the old style syntax

```
Get-Process | Where {$_.CPU -gt 50 -AND $_.Handles -gt 1000}
```

In all of these cases you're comparing a property of the current pipeline object against a value. If the comparison is true the object is passed onto the next step of the pipeline. If its false the object is discarded.

You can use $psitem in place of $_ if you prefer

$PSVersionTable

8) Execution of azure resources using powershell?

Creating an Azure Virtual Machine

If you need to create a new Azure virtual machine for production or development use, you can use the New-AzureRMVM PowerShell cmdlet. You can specify parameters such as location of VHD, memory, processors, etc. To quickly create a virtual machine in Azure and with

default configuration, just execute the following PowerShell command:

```
New-AzureRMVM -ResourceGroupName ResourceGroupHere -Name VMNameHere
```

While this command creates a virtual machine quickly in the Resource Group specified in the "-ResourceGroupName" parameter, the next command will create a virtual machine in Azure from a managed image.

```
New-AzureRMVM -ResourceGroupName ResourceGroupHere -Name VMNameHere -ImageName "YourImage" -Location "West-US"
```

Reporting Azure Virtual Machines

For reporting purposes, you might find the need to check how many Azure virtual machines have been deployed and are currently running. Microsoft provides the Get-AzureRMVM PowerShell cmdlet, which can be used to report virtual machines from an Azure subscription and/or from an Azure resource group. The next set of PowerShell commands provides some examples of using Get-AzureRMVM:

To get all the virtual machines from an Azure Resource Group, execute this PowerShell command:

```
Get-AzureRMVM -ResourceGroupName "ResourceGroupNameHere"
```

To export output to a CSV file, add the Export-CSV cmdlet as shown in below:

```
Get-AzureRMVM -ResourceGroupName "ResourceGroupNameHere" | Export-CSV C:\Temp\AllAzureVMs.CSV -NoTypeInfo
```

In Get-AzureRMVM you can also specify the properties that you would like to be reported when querying virtual machines in Azure. For example, the following PowerShell command only collects the type of Operating System running on virtual machines:

```
Get-AzureRMVM | Select Name,@{Name="OSType"; Expression={$_.StorageProfile.OSDisk.OSType}}
```

By using the method shown in the command above, you can access nested properties of Azure virtual machines.

If you wish to list all the virtual machines in an Azure subscription, simply execute the "Get-AzureRMVM" command in an elevated PowerShell window. If you would like to list virtual machines from Resource Groups listed in a text file, using the following PowerShell script would work:

```
$ResGroups = "C:\Temp\ResGroups.TXT"
$ReportFile = "C:\Temp\AllVMsInAzure.CSV"
$STR = "VM Name, Resource Group, Location, Status Code"
Add-Content $ReportFile $STR
Foreach ($ResGroup in Get-Content $ResGroups)
```

```
    {
    $allVMsInReg = Get-AzureRMVM -ResourceGroupName "$ResGroup"
    ForEach ($AllVMs in $AllVMsInRes)
    {
    $VMLocation = $AllVMs.Location
    $VMName = $AllVMs.Name
    $VMStatus = $AllVMs.StatusCode

    $STR = $VMName+","+$ResGroup+","+$VMLocation+","+$VMStatus
    Add-Content $ReportFile $STR
    }
    }
```

When you have finished executing the above script, a report file will be generated that contains the name of the virtual machine, the resource group name of the virtual machine, the virtual machine Azure location and the status code indicating the overall status of the virtual machine.

Staring, Stopping, and Restarting Azure Virtual Machines

From time to time you may need to stop and start Azure virtual machines in order to reduce billing costs. For example, you may not want to run development virtual machines if they are not in use. Similarly, if you have assigned temporary virtual machines to a third-party vendor, you may want to remove them on your end as quickly as possible.

To stop an Azure virtual machine, you can use the Stop-AzureRMVM PowerShell cmdlet as shown in the following PowerShell command:

```
Stop-AzureRMVM -ResourceGroupName "ResourceGroupName" -Name VMNameToStopHere
```

And the next PowerShell script can be used to stop specific virtual machines specified in a text file. Let's assume the text file contains the list of virtual machines to be stopped. Once you have created the text file that contains the virtual machines to be stopped, run the following PowerShell script:

```
$VMFile = "C:\Temp\AllVMsToStop.TXT"
    Foreach ($ThisVM in Get-Content $VMFile)
    {
    Stop-AzureRMVM -Name $ThisVM
    }
```

Starting an Azure virtual machine is similar to stopping an Azure virtual machine. However, you will of course need to use the Start-AzureRMVM PowerShell cmdlet in place of the Stop-AzureRMVM cmdlet as shown in the following command:

```
Start-AzureRMVM -ResourceGroupName "ResourceGroupName" -Name VMNameToStopHere
```

If you run into any issues with the Azure virtual machine and are not able to take control via the Azure management portal or if you have any issues accessing the virtual machine, what you can do is restart the virtual machine using the PowerShell command below:

```
Restart-AzureRMVM -ResourceGroupName "ResourceGroupName" -Name
VMNameToStopHere
```

9) How will u catch the exception in powershell?

```
Try
{
  $AuthorizedUsers = Get-Content \\ FileServer\HRShare\UserList.txt -ErrorAction Stop
}
Catch [System.OutOfMemoryException]
{
  Restart-Computer localhost
}
Catch
{
  $ErrorMessage = $_.Exception.Message
  $FailedItem = $_.Exception.ItemName
  Send-MailMessage -From ExpensesBot@MyCompany.Com -To
   WinAdmin@MyCompany.Com -Subject "HR File Read Failed!" -SmtpServer
   EXCH01.AD.MyCompany.Com -Body "We failed to read file $FailedItem. The error message
   was $ErrorMessage"
  Break
}
Finally
{
  $Time=Get-Date
  "This script made a read attempt at $Time" | out-file c:\logs\ExpensesScript.log -append
}
```

The last part of Try Catch Finally is the Finally block. This must be defined immediately after the Catch block and runs every time, regardless of whether there was an error or not. In this way you can perform actions that need to be made regardless of whether an operation succeeds or fails. In our example we are going to log that a file read was attempted. Our Get-Content line now looks like:

10) Gated check-in?

Use a gated check-in build process to validate changes

When a developer checks in changes that break the build, the result can be a significant hassle for small teams. The cost to larger teams can be even more expensive when measured by

lost productivity and schedule delays. You can guard some or all of your code base against these problems by creating a gated check-in build definition.

Note

Gated check-in builds are available only in TFVC team projects. They are not available in Git team projects.

What do you want to do?

- Understand how gated check-in builds affect your team
- Define a gated check-in build process
- Guidelines to improve build process function and performance
- Avoid blocking your team
- Manually run gated check-in builds and private builds

Scenarios for gates

Some scenarios and use cases for gates are:

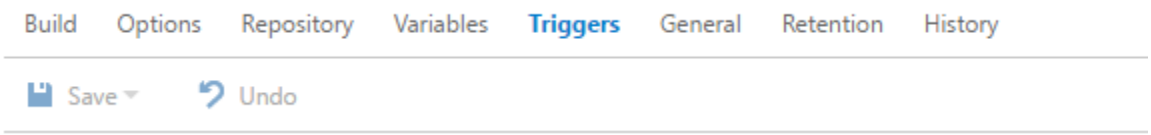
- **Incident and issues management**. Ensure the required status for work items, incidents, and issues. For example, ensure deployment occurs only if no priority zero bugs exist, and validation that there are no active incidents takes place after deployment.
- **Seek approvals outside Azure Pipelines**. Notify non-Azure Pipelines users such as legal approval departments, auditors, or IT managers about a deployment by integrating with approval collaboration systems such as Microsoft Teams or Slack, and waiting for the approval to complete.
- **Quality validation**. Query metrics from tests on the build artifacts such as pass rate or code coverage and deploy only if they are within required thresholds.
- **Security scan on artifacts**. Ensure security scans such as anti-virus checking, code signing, and policy checking for build artifacts have completed. A gate might initiate the scan and wait for it to complete, or just check for completion.
- **User experience relative to baseline**. Using product telemetry, ensure the user experience hasn't regressed from the baseline state. The experience level before the deployment could be considered a baseline.
- **Change management**. Wait for change management procedures in a system such as ServiceNow complete before the deployment occurs.
- **Infrastructure health**. Execute monitoring and validate the infrastructure against compliance rules after deployment, or wait for healthy resource utilization and a positive security report.

Introduction

Visual Studio Team Services (VSTS) is the online team foundation server hosted in cloud. It helps to manage, build and deploy code in the cloud. Using VSTS build definition, we can automate the process of creating a build. A build definition contains all the steps that a build is supposed to do. It also specifies when a build should be triggered. You can trigger the build whenever a check-in is performed by a developer or you can schedule the build to get triggered at a specific time. There is another way to trigger a build which is called "Gated Check-in". Gated check-in helps to restrict developers from checking in a broken code into source control system and thus helps to avoid blocking your team. With gated check-in when a check-in is initiated by a developer, it will build the project and will check-in the code only if the build is successful. Gated check-in is suitable for projects whose overall build time is less than few minutes.

Enable Gated Check-in Using VSTS Build Definition

To create a build definition, navigate to "Build & Release" tab of VSTS team project and click on "Builds". Under Build Definitions, click on New button, it will display a wizard to create a new build definition. Select the required template and configure the required repository settings. The build definition has different tabs to configure the build settings. To enable gated check-in, navigate to Triggers tab as shown in the below screenshot.



Under Gated Check-in section, click on the check box to enable gated check in as shown in the below screenshot.

Save the build definition. Now when a developer tries to check-in code in Visual Studio, gated check-in build is triggered automatically as shown in the below screenshot.

The build server creates a shelveset, merges with the latest code and then compiles the code. If the build is configured to run unit tests, then the unit tests are also executed. Once the build is successful, the code is committed to source control.

There is an option to Bypass validation build and check in changes directly. This checkbox is enabled only when the user has "Override Check-in Validation by Build" permission set for the selected build definition.

If "Preserve my pending changes locally" option is checked, then the changes being submitted will remain in the workspace.

How to Set Permission for Build Definition

To enable "Override Check-in Validation by Build" permission, select the build definition in VSTS and click on security as shown in the below screenshot.

Set the required option for "Override check-in validation by build" field as shown in the below screenshot:

11) SonarQube use?

Configure

The first thing to do is to declare your SonarQube server as a service endpoint in your Azure DevOps project settings.

1. Open the Connections page in your Azure DevOps project: **Project Settings > Pipelines > Service Connections**.
2. Click on **New service connection** and choose **SonarQube**.
3. Specify a **Connection name**, the **Server URL** of your SonarQube Server (including the port if required) and the Authentication Token to use.

Each extension provides three tasks you will use in your build definitions to analyze your projects:

- **Prepare Analysis Configuration** task, to configure all the required settings before executing the build.
  - This task is mandatory.
  - In case of .NET solutions or Java projects, it helps to integrate seamlessly with MSBuild, Maven and Gradle tasks.
- **Run Code Analysis** task, to actually execute the analysis of the source code.
  - This task is not required for Maven or Gradle projects, because scanner will be run as part of the Maven/Gradle build.
- **Publish Quality Gate Result** task, to display the Quality Gate status in the build summary and give you a sense of whether the application is ready for production "quality-wise".
  - This tasks is optional.
  - It can significantly increase the overall build time because it will poll SonarQube until the analysis is complete. Omitting this task will not affect the analysis results on SonarQube - it simply means the Azure DevOps Build Summary page will not show the status of the analysis or a link to the project dashboard on SonarQube.

When creating a build definition you can filter the list of available tasks by typing "Sonar" to display only the relevant tasks.

Analyzing a .NET solution

1. In your build definition, add:
   - At least **Prepare Analysis Configuration** task and **Run Code Analysis** task
   - Optionally **Publish Quality Gate Result** task
2. Reorder the tasks to respect the following order:
   - **Prepare Analysis Configuration** task before any **MSBuild** or **Visual Studio Build** tasks.
   - **Run Code Analysis** task after the **Visual Studio Test task**.
   - **Publish Quality Gate Result** task after the **Run Code Analysis** task
3. Click on the **Prepare Analysis Configuration** build step to configure it:
   - You must specify the service connection (i.e. SonarQube) to use. You can:
     - select an existing endpoint from the drop down list
     - add a new endpoint
     - manage existing endpoints
   - Keep **Integrate with MSBuild** checked and specify at least the project key

- **Project Key** - the unique project key in SonarQube
- **Project Name** - the name of the project in SonarQube
- **Project Version** - the version of the project in SonarQube

4. Click the **Visual Studio Test** task and check the **Code Coverage Enabled** checkbox to process the code coverage and have it imported into SonarQube. (Optional but recommended)

Once all this is done, you can trigger a build.

Analyzing a Java project with Maven or Gradle

1. In your build definition, add:
   - At least **Prepare Analysis Configuration** task
   - Optionally **Publish Quality Gate Result** task
2. Reorder the tasks to respect the following order:
   - **Prepare Analysis Configuration** task before the **Maven** or **Gradle** task.
   - **Publish Quality Gate Result** task after the **Maven** or **Gradle** task.
3. Click on the **Prepare Analysis Configuration** task to configure it:
   - Select the **SonarQube Server**
   - Select **Integrate with Maven or Gradle**
4. On the Maven or Gradle task, in **Code Analysis**, check **Run SonarQube or SonarCloud Analysis**

Once all this is done, you can trigger a build.

SonarQube (formerly Sonar) is an open-source platform developed by SonarSource for continuous inspection of code quality to perform automatic reviews with static analysis of code to detect bugs, code smells, and security vulnerabilities on 20+ programming languages

Sonar covers the 7 sections of code quality

1. Architecture and Design
2. Unit tests
3. Duplicated code
4. Potential bugs
5. Complex code
6. Coding standards
7. Comments

Sustainability:

SonarQube platform significantly increases the lifetime of applications by reducing complexities, duplications and potential bugs in the code, by keeping neat and clean code architecture and increased unit tests.

SonarQube increases maintainability of the software. It also has the ability to handle changes.

Productivity:

SonarQube increases productivity by enabling development teams to detect and muzzle duplication and redundancy of code.

SonarQube facilitates the team members to reduce the size of application, code complexity, maintenance time and cost and make code easy to read and understand.

SonarQube decreases the risk of extra cost and time when changing the application code.

Raise Quality:

SonarQube can perform as a multi-dimensional analyst and can inform on seven sections of code quality. For the better quality, it avoids duplicate code, keeps code complexity low and increases coverage by units.

It can determine violation of code standards and helps software development team to abolish bugs. It maintains high-quality architecture, enforces coding standards and document APIs.

This platform also facilitates developers to create a customizable dashboard and filters to focus on key areas. These tools help in monitoring the code quality and also keep track issues. It helps in on time delivery of the quality product.

Increase Developer Skills:

The development teams as a part of their development process can adopt it quickly, because SonarQube provides enormous value to the development teams.

The development teams receive regular feedbacks on quality issues and it helps them increase their programming skills.

SonarQube helps developers to understand the quality of their software and ensures the transparency of code.

Scale With Business Needs:

SonarQube is designed to scale with business needs. There has been no limit discovered to its scalability yet.

SonarQube has been tested in environments. It performs daily analysis on more than five thousand projects with more than four million lines of code and twenty developers.

Enable Continuous Code Quality Management:

With SonarQube, analysis of code becomes more easy and developers receive valuable insights to ensure that this is broadly adopted.

Code quality becomes a part of development process and development teams. By enabling continuous code quality management, the software quality is raised and decreases the cost and risk of software management.

Define And Increment Requirements Efficiently:

SonarQube has a set of some predefined standards that enable developers and software managers to get immediate insight into application quality. To adapt the organization or team specific requirements, it can be configured easily.

Foster Innovation:

As more companies migrate to SonarQube platform, they increase in size as well as in diversity. This platform enables these companies to customize and extend its functionality. Companies can get an increasing number of plugins and an extensive developers network.

Reduce Risk with Vendor Support and Services:

To enable customers to get maximum value from their investment, SonarQube provides additional value and professional support. Services including development, technical support, consulting and training are designed to help companies get long term benefits.

In Vizteck Solutions, we used peer-based code reviews. However, after integration SonarQube with our Continuous Integration Process with Jenkin. SonarQube facilitates our developers by identifying most of the things and reducing time required by the developers

12) hosted agents vs self hosted agents?

Microsoft-hosted agents

If your pipelines are in Azure Pipelines, then you've got a convenient option to run your jobs using a **Microsoft-hosted agent**. With Microsoft-hosted agents, maintenance and upgrades are taken care of for you. Each time you run a pipeline, you get a fresh virtual machine. The virtual machine is discarded after one use. Like self-hosted agents, Microsoft-hosted agents can run jobs directly on the VM or in a container.

For many teams this is the simplest way to run your jobs. You can try it first and see if it works for your build or deployment. If not, you can use a self-hosted agent.

Tip

You can try a Microsoft-hosted agent for no charge. If your job fails, the issues will be reported in the logs.

Learn more about Microsoft-hosted agents.

Self-hosted agents

An agent that you set up and manage on your own to run jobs is a **self-hosted agent**. You can use self-hosted agents in Azure Pipelines or Team Foundation Server (TFS). Self-hosted agents give you more control to install dependent software needed for your builds and deployments. Also, machine-level caches and configuration persist from run to run, which can boost speed.

Tip

Before you install a self-hosted agent you might want to see if a Microsoft-hosted agent pool will work for you. In many cases this is the simplest way to get going. **Give it a try**.

You can install the agent on Linux, macOS, or Windows machines. You can also install an agent on a Docker container. See the following topics for additional information on installing a self-hosted agent:

- macOS agent
- Linux agent (x64, ARM, RHEL6)
- Windows agent (x64, x86)
- Docker agent

After you've installed the agent on a machine, you can install any other software on that machine as required by your jobs.

13) Use of **NuGet Package?**

NuGet is a **Package** management system for Visual Studio. It makes it easy to add, update and remove external libraries in our **application**. Using **NuGet**, we can create our own **packages** easily and make it available for others. **NuGet** is a **Package** management system for Visual Studio

NuGet can be **used** to find and install **packages**, that is, software pieces and assemblies and things that you want to **use** in your project. **NuGet** is not a tool that is specific to ASP.NET **MVC** projects.

14) Use of **MSBuild** ?

MSBuild is a **build** tool that helps automate the process of **creating** a software product, including compiling the source code, packaging, testing, deployment and **creating** documentations. With **MSBuild**, it is possible to **build** Visual Studio projects and solutions without the Visual Studio IDE installed

MSBuild is the build platform that enables all build activity in the Visual Studio world.

A better, more practical example would be to state that

1. The .csproj files (every C# project) are msbuild files
2. When you hit F5, you basically (oversimplifying) call msbuild.exe and passing in your .csproj file.

MSBuild empowers all the things that make *hitting F5* work. From creating the "debug" or "release" folder, to dropping references into the bin\ directory, to invoking CSC ... and everything in between ... MSBuild "powers" all that.

If all you will ever need from a build is the output that F5 gives you, then you know about all you probably need to know about MSBuild.

In most commercial/practical development scenarios, however, there will come a time where there is a need to customize the build process. The most common approach is automating the build process (using either TeamBuild or some homegrown system). You may also need to

- create a "packaged" deployment
- link to another library outside of your project that is also actively being developed
- publish your build to an FTP and send an email to a customer notifying them of its availability.

The use of a unified and extensible build platform (ie MSBuild) is what makes all these these possible, while still being part of the build process ... keeping the "build" part of the development pipeline simple and contained.

15) How we will use start and the values in powershell?

Start-Process
Module:
Microsoft.PowerShell.Management
Starts one or more processes on the local computer.
Syntax
PowerShellCopy
Start-Process
[-FilePath] <String>
[[-ArgumentList] <String[]>]
[-Credential <PSCredential>]
[-WorkingDirectory <String>]
[-LoadUserProfile]
[-NoNewWindow]
[-PassThru]
[-RedirectStandardError <String>]
[-RedirectStandardInput <String>]
[-RedirectStandardOutput <String>]
[-WindowStyle <ProcessWindowStyle>]
[-Wait]
[-UseNewEnvironment]
[-WhatIf]
[-Confirm]
[<CommonParameters>]
PowerShellCopy
Start-Process
[-FilePath] <String>
[[-ArgumentList] <String[]>]
[-WorkingDirectory <String>]
[-PassThru]
[-Verb <String>]
[-WindowStyle <ProcessWindowStyle>]
[-Wait]
[-WhatIf]

[-Confirm]
[<CommonParameters>]

Description

The Start-Process cmdlet starts one or more processes on the local computer. To specify the program that runs in the process, enter an executable file or script file, or a file that can be opened by using a program on the computer. If you specify a non-executable file, Start-Process starts the program that is associated with the file, similar to the Invoke-Item cmdlet.

You can use the parameters of Start-Process to specify options, such as loading a user profile, starting the process in a new window, or using alternate credentials.

Examples

Example 1: Start a process that uses default values

This example starts a process that uses the **Sort.exe** file in the current folder. The command uses all of the default values, including the default window style, working folder, and credentials.

PowerShellCopy

```
Start-Process -FilePath "sort.exe"
```

Example 2: Print a text file

This example starts a process that prints the C:\PS-Test\MyFile.txt file.

PowerShellCopy

```
Start-Process -FilePath "myfile.txt" -WorkingDirectory "C:\PS-Test" -Verb Print
```

Example 3: Start a process to sort items to a new file

This example starts a process that sorts items in the Testsort.txt file and returns the sorted items in the Sorted.txt files. Any errors are written to the SortError.txt file.

PowerShellCopy

```
Start-Process -FilePath "Sort.exe" -RedirectStandardInput "Testsort.txt" -RedirectStandardOutput "Sorted.txt" -RedirectStandardError "SortError.txt" -UseNewEnvironment
```

The **UseNewEnvironment** parameter specifies that the process runs with its own environment variables.

Example 4: Start a process in a maximized window

This example starts the **Notepad.exe** process. It maximizes the window and retains the window until the process completes.

PowerShellCopy

```
Start-Process -FilePath "notepad" -Wait -WindowStyle Maximized
```

Example 5: Start PowerShell as an administrator

This example starts PowerShell by using the "Run as administrator" option.

PowerShellCopy

```
Start-Process -FilePath "powershell" -Verb RunAs
```

Example 6: Using different verbs to start a process

This example shows how to find the verbs that can be used when starting a process. The available verbs are determined by the filename extension of the file that runs in the process.

PowerShellCopy

```
$startExe = New-Object System.Diagnostics.ProcessStartInfo -Args PowerShell.exe
$startExe.verbs
```

```
open
runas
```

The example uses New-Object to create a **System.Diagnostics.ProcessStartInfo** object for **PowerShell.exe**, the file that runs in the PowerShell process. The **Verbs** property of the **ProcessStartInfo** object shows that you can use the **Open** and **RunAs** verbs with **PowerShell.exe**, or with any process that runs a .exe file.

16) How we will execute function in poweshell?

How to Create a Function in PowerShell

If you have worked with other programming languages, you have may used functions for code reusability. You can also create functions in PowerShell.

PowerShell function example

Below is syntax of simple function

```
function hello-world

{

write-host "hello world"

}
```

You can also pass parameters in a function using param keyword

```
function Get-TimesResult {

Param ([int]$a,[int]$b)

$c = $a * $b

Write-Output $c
```

}

You can call a function using its name like

hello-world

and if a function is taking some parameters type

Get-TimesResult -a 5 -b 10

You can also return a value from function by using return keyword. The function when called return a value

```
function Get-TimesResult {

Param ([int]$a,[int]$b)

$c = $a * $b

return $c

}
```

You call this function and store it's returned value in some variable

```
$r= Get-TimesResult -a 5 -b 5
```

2nd example

Functions are a common occurrence in PowerShell and to truly understand the PowerShell language, it's important that you know how functions work. In this article, we're going to show how a PowerShell function evolves from basically nothing all the way to having parameters using validation attributes.

To demonstrate this, let's first start off with a function that can't get any simpler.

```
function Write-Log {
    [CmdletBinding()]
    param()
}
```

When run inside of a PowerShell session, this function will execute but will do nothing because there's no code to execute. PowerShell simply executes the function itself and returns.

Learn how to automate IT tasks with PowerShell. **Download this eBook.**

For the function to actually do something, we need to add some code. Code is added in between the param block and the last curly brace. You can see below that I'm making my **Write-Log** function return the string "I did something" to the console.

```
function Write-Log {
    [CmdletBinding()]
    param()

    'I did something'
}

PS> Write-Log
I did something
```

Our function is called **Write-Log** so I'm assuming that this will be a function that will eventually write some kind of text to a log file. Because we won't want to write the same thing to a log file every time, we need to provide some way to change the message when the function runs. To do that, we will add a parameter. Parameters allow the user to pass different values into the function at run-time. This allows the function to gather dynamic input at runtime.

To add a parameter, I'll add a variable. In this case, the variable is called **Message**. I'll add it inside of the **param** block as shown below. You can then see that I can reference that parameter inside of the function itself. When run, the function will return whatever value I pass to the **Message** parameter.

```
function Write-Log {
    [CmdletBinding()]
    param($Message)

    $Message
}

PS> Write-Log -Message 'I did something'
I did something
```

You don't have to stop at just one parameter. We can add as many as we want here. Below, I'm adding a **Severity** parameter, providing a value when the function is run and you can see that it exhibits the same behavior.

```
function Write-Log {
    [CmdletBinding()]
    param($Message, $Severity)

    "$Message - Severity: $Severity"
}
```

```
Write-Log -Message 'I did something' -Severity 1
```

Now that we know how to handle parameters, we can get into parameter types. A parameter has a type just like anything else in PowerShell. Above, it was using a type but it accepted any kind of object imaginable. The parameter wasn't explicitly typed. It's good practice to do this on all parameters to ensure only the values you expect are passed.

Enhance your IT career by learning how to automate with Python. **Get started with this free Python guide.**

To define a type, we can add the type in square brackets right before the parameter is declared.

```
function Write-Log {
    [CmdletBinding()]
    param([System.ServiceProcess.ServiceController]$Message)

    $Message
}
```

Once an explicit type is assigned to the parameter, PowerShell will only accept input of that type *or* an object that it can convert. In this example below, I'm passing the boolean value **$false** to the Message parameter. You can see that PowerShell won't allow it. This is because it can't convert a boolean type to a **ServiceController** type.

```
PS C:\> Write-Log -Message $false
Unable to find type [System.ServiceProcess.ServiceController].
At line:3 char:8
+     param([System.ServiceProcess.ServiceController]$Message)
+           ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (System.ServiceProcess
    + FullyQualifiedErrorId : TypeNotFound
```

However, if we pass a **ServiceController** object that **Get-Service** returns, it works just fine.

```
PS> Write-Log -Message $service

Status Name DisplayName
------ ---- -----------
Stopped AdtAgent Microsoft Monitoring Agent Audit Fo...
```

We can also use parameter attributes. Parameter attributes allow us to define different characteristics of each parameter which determines how it works. In the example below, I'm using the **Mandatory** parameter. These attributes forces the user to pass a value to the **Message** parameter else the function will not run.

I'm also setting a default value on the **Severity** parameter. This allows me to force **Severity** to always be 1 *unless* it is overridden at run time by passing a value to that parameter.

```
function Write-Log {
    [CmdletBinding()]
    param(
```

```
                                                    [Parameter(Mandatory)]
                                                    [string]$Message,

                                                    [Parameter()]
                                [int]$Severity      =       1
                                                    )

                    "$Message       -       Severity:       $Severity"

    }
```

Finally, we can use parameter validation attributes. Parameter validation attributes are a great way to restrict what values are passed to a parameter. In the example below, I'm using the **ValidateRange()** validation attribute. This attribute is used on parameters of type integer to define a range of allowed numbers. In this example, I am only allowing values of **Severity** to be 1-5. Any other value not in that range will fail.

```
function        Write-Log       {
                                    [CmdletBinding()]
                                    param(
                                    [Parameter(Mandatory)]
                                    [pscustomobject]$Message,

                                    [Parameter()]
                    [ValidateRange(1,       5)]
                                    [int]$Severity
                                    )

                    "$Message       -       Severity:       $Severity"
    }
```

17) how to print variable value in a text powershell?

Write-Output

Module:

Microsoft.PowerShell.Utility

Sends the specified objects to the next command in the pipeline. If the command is the last command in the pipeline, the objects are displayed in the console.

Syntax

```
Write-Output
   [-InputObject] <PSObject[]>
   [-NoEnumerate]
   [<CommonParameters>]
```

Description

The **Write-Output** cmdlet sends the specified object down the pipeline to the next command. If the command is the last command in the pipeline, the object is displayed in the console.

Write-Output sends objects down the primary pipeline, also known as the "output stream" or the "success pipeline." To send error objects down the error pipeline, use Write-Error.

This cmdlet is typically used in scripts to display strings and other objects on the console. However, because the default behavior is to display the objects at the end of a pipeline, it is generally not necessary to use the cmdlet. For instance, Get-Process | Write-Output is equivalent to Get-Process.

Examples

Example 1: Get objects and write them to the console

```
PS C:\> $P = Get-Process
PS C:\> Write-Output $P
PS C:\> $P
```

The first command gets processes running on the computer and stores them in the $P variable.

The second and third commands display the process objects in $P on the console.

Example 2: Pass output to another cmdlet

```
PS C:\> Write-Output "test output" | Get-Member
```

This command pipes the "test output" string to the Get-Member cmdlet, which displays the members of the **System.String** class, demonstrating that the string was passed along the pipeline.

Example 3: Suppress enumeration in output

```
PS C:\> Write-Output @(1,2,3) | measure

Count    : 3
...

PS C:\> Write-Output @(1,2,3) -NoEnumerate | measure

Count    : 1
```

This command adds the *NoEnumerate* parameter to treat a collection or array as a single object through the pipeline.

Parameters

-InputObject

Specifies the objects to send down the pipeline. Enter a variable that contains the objects, or type a command or expression that gets the objects.

Type:	PSObject[]

Position:	0
Default value:	None
Accept pipeline input:	True (ByValue)
Accept wildcard characters:	False

-NoEnumerate

By default, the **Write-Output** cmdlet always enumerates its output. The *NoEnumerate* parameter suppresses the default behavior, and prevents **Write-Output** from enumerating output. The *NoEnumerate* parameter has no effect on collections that were created by wrapping commands in parentheses, because the parentheses force enumeration.

Note

This switch only works correctly with PowerShell Core 6.2 and newer. On older versions of PowerShell Core, the collection is still enumerated even with use of this switch. The behavior in PowerShell Core 6.2 is consistent with Windows PowerShell.

Type:	SwitchParameter
Position:	Named
Default value:	None
Accept pipeline input:	False
Accept wildcard characters:	False

Inputs

System.Management.Automation.PSObject

You can pipe objects to **Write-Output**.

Outputs

System.Management.Automation.PSObject

Write-Output returns the objects that are submitted as input.

18) flow of poweshell scripts?

Run the scripts

Once the modules are installed, follow these instructions to run the scripts provided below. If you receive a security warning, you may need to unblock running the downloaded script, see this article for more details.

1. Download the desired script.
2. Run PowerShell as an administrator and make sure you're in the same directory as the script.
3. Run the script by typing out the name
4. .\findFlowsWithHttpAction.ps1
5. Each of these scripts have optional parameters to specify behavior, such as the Environment (EnvironmentName) or the output file path name (Path). Some of the scripts have mandatory parameters. More details on each parameter is provided in the subfolder's documentation.
6. .\findFlowsWithHttpAction.ps1 -EnvironmentName 820d6103-3f73-4107-a1b2-3449a98f5049 -Path ./myFlowsWithHttp.csv

Sections

Based on the task, there are subfolders that hold multiple scripts to programmatically access the PowerApps and Flow APIs.

Reporting

Use the reporting scripts to help discover a filtered list of PowerApps or Flows based on some features they leverage.

Administration

Automated administrative tasks, such as updating permissions or cleaning up unauthorized resources.

19) if i want to select single subscription in azure powershell what is cmdlets?

Select-AzureSubscription

Module:

Azure

Changes the current and default Azure subscriptions.

Syntax

PowerShellCopy

```
Select-AzureSubscription
    -SubscriptionName <String>
```

```
    [-Account <String>]
    [-Current]
    [-PassThru]
    [-Profile <AzureSMProfile>]
    [<CommonParameters>]
PowerShellCopy
Select-AzureSubscription
    -SubscriptionName <String>
    [-Account <String>]
    [-Default]
    [-PassThru]
    [-Profile <AzureSMProfile>]
    [<CommonParameters>]
PowerShellCopy
Select-AzureSubscription
    -SubscriptionId <String>
    [-Account <String>]
    [-Current]
    [-PassThru]
    [-Profile <AzureSMProfile>]
    [<CommonParameters>]
PowerShellCopy
Select-AzureSubscription
    -SubscriptionId <String>
    [-Account <String>]
    [-Default]
    [-PassThru]
    [-Profile <AzureSMProfile>]
    [<CommonParameters>]
PowerShellCopy
Select-AzureSubscription
    [-Account <String>]
    [-NoCurrent]
    [-PassThru]
    [-Profile <AzureSMProfile>]
    [<CommonParameters>]
PowerShellCopy
Select-AzureSubscription
    [-Account <String>]
    [-NoDefault]
```

[-PassThru]
[-Profile <AzureSMProfile>]
[<CommonParameters>]

Description

The **Select-AzureSubscription** cmdlet sets and clears the current and default Azure subscriptions.

The "current subscription" is the subscription that is used by default in the current Windows PowerShell session. The "default subscription" is used by default in all Windows PowerShell sessions. The "current subscription" label lets you specify a different subscription to be used by default for the current session without changing the "default subscription" for all other sessions.

The "default" subscription designation is saved in your subscription data file. The session-specific "current" designation is not saved.

This topic describes the cmdlet in the 0.8.10 version of the Microsoft Azure PowerShell module. To get the version of the module you're using, in the Azure PowerShell console, type (Get-Module -Name Azure).Version.

Examples

Example 1: Set the current subscription

PowerShellCopy

C:\PS> Select-AzureSubscription -Current -SubscriptionName ContosoEngineering

This command makes "ContosoEngineering" the current subscription.

Example 2: Set the default description

PowerShellCopy

C:\PS> Select-AzureSubscription -Default -SubscriptionName ContosoFinance -SubscriptionDataFile "C:\subs\MySubscriptions.xml"

This command changes the default subscription to "ContosoFinance." It saves the setting in the Subscriptions.xml subscription data file, instead of the default subscription data file.

To set the Subscription in PowerShell so you can run cmdlets against those features, perform the following:

Login via PowerShell

Set the current subscription context

Once you complete the above, you will have set the current azure subscription context

Login via PowerShell

I like to use the Windows PowerShell ISE console app because I find it more friendly, but choose you own approach. Enter the following command to login your PowerShell to Azure:

Login-AzureRmAccount

This will open a challenge/response window where you can enter your userid and password, use the same credentials that you used when you created the PowerShellResourceGroup in the previous section. Once successfully logged in, you will see some output similarly to that shown in Figure 1.

image_thumb[4]

Figure 1, Login to the Azure Portal using PowerShell

Confirm that the SubscriptionId shown in the output is the expected SubscriptionId, if not then continue to the next section

Set the current subscription context

To set the current context to a different subscription, execute this command (using your subscription):

Set-AzureRmContext -SubscriptionId "25ec5bae-****-****-****-**********"

20) Deploy resources with Resource Manager templates and Azure PowerShell

Deploy resources with Resource Manager templates and Azure PowerShell

08/21/2019

7 minutes to read

+4

Learn how to use Azure PowerShell with Resource Manager templates to deploy your resources to Azure. For more information about the concepts of deploying and managing your Azure solutions, see Azure Resource Manager overview.

Note

This article has been updated to use the new Azure PowerShell Az module. You can still use the AzureRM module, which will continue to receive bug fixes until at least December 2020. To learn more about the new Az module and AzureRM compatibility, see Introducing the new Azure PowerShell Az module. For Az module installation instructions, see Install Azure PowerShell.

Deployment scope

You can target your deployment to either an Azure subscription or a resource group within a subscription. In most cases, you'll target deployment to a resource group. Use subscription

deployments to apply policies and role assignments across the subscription. You also use subscription deployments to create a resource group and deploy resources to it. Depending on the scope of the deployment, you use different commands.

To deploy to a resource group, use New-AzResourceGroupDeployment:

Azure PowerShell

Copy

```
New-AzResourceGroupDeployment -ResourceGroupName <resource-group-name> -TemplateFile <path-to-template>
```

To deploy to a subscription, use New-AzDeployment:

Azure PowerShell

Copy

```
New-AzDeployment -Location <location> -TemplateFile <path-to-template>
```

Currently, management group deployments are only supported through the REST API. See Deploy resources with Resource Manager templates and Resource Manager REST API.

The examples in this article use resource group deployments. For more information about subscription deployments, see Create resource groups and resources at the subscription level.

Prerequisites

You need a template to deploy. If you don't already have one, download and save an example template from the Azure Quickstart templates repo. The local file name used in this article is c:\MyTemplates\azuredeploy.json.

Unless you use the Azure Cloud shell to deploy templates, you need to install Azure PowerShell and connect to Azure:

Install Azure PowerShell cmdlets on your local computer. For more information, see Get started with Azure PowerShell.

Connect to Azure by using Connect-AZAccount. If you have multiple Azure subscriptions, you might also need to run Set-AzContext. For more information, see Use multiple Azure subscriptions.

Deploy local template

The following example creates a resource group, and deploys a template from your local machine. The name of the resource group can only include alphanumeric characters,

periods, underscores, hyphens, and parenthesis. It can be up to 90 characters. It can't end in a period.

Azure PowerShell

Copy

```
$resourceGroupName = Read-Host -Prompt "Enter the Resource Group name"
$location = Read-Host -Prompt "Enter the location (i.e. centralus)"

New-AzResourceGroup -Name $resourceGroupName -Location $location
New-AzResourceGroupDeployment -ResourceGroupName $resourceGroupName `
  -TemplateFile c:\MyTemplates\azuredeploy.json
```

The deployment can take a few minutes to complete.

21) Validate Azure ARM templates with PowerShell

Validate Azure ARM templates with PowerShell
18th March 2019 no comments in DevOps

If you are writing and using ARM templates as as a base of your infrastructure as code (IaC) principle, then the next few lines might be interesting to you. If you are new to IaC read this or if you are new to ARM templates read here.

We are using ARM templates with Azure DevOps to deploy resource groups of Virtual machines, databases and Application services. Templates tend to get large and even with a Visual Studio Intellisense which is good by the way, you can make mistakes. What we need is a tool that can validate the whole script before you commit/push to the source control. I was thinking, “is there a way to do this?” and indeed there is.

Microsoft documentation states that there is a tool (PowerShell) that can check your ARM templates for compatibility with Azure Stack in your region of choice. It also checks the JSON syntax and some other goodies (examples below). Let’s get started.

1. Download and install Azure Stack tools.

2. Unpack it to the folder of your choice and “cd” to the Azure Stack Tools folder.

3. Import Cloud capabilities module with the command below. The instructions in the documentation are not very clear, regarding the “<your location>” parameter. But I made some digging and that is of course Azure location (run Get-AzureRMLocation in PowerShell to get your location name). We use westeurope for a location parameter.

```
//import cloud capabilities module
Import-Module .\CloudCapabilities\AzureRM.CloudCapabilities.psm1

//Generate capabilities json file
Get-AzureRMCloudCapability -Location "westeurope" -Verbose -IncludeComputeCapabilities -
    IncludeStorageCapabilities
1
2
3
4
5
```

```
//import cloud capabilities module
Import-Module .\CloudCapabilities\AzureRM.CloudCapabilities.psm1

//Generate capabilities json file
Get-AzureRMCloudCapability -Location "westeurope" -Verbose -IncludeComputeCapabilities -
    IncludeStorageCapabilities
```

It takes some time 30 min or so, to generate capabilities JSON file, which will be then used by our next script Template Validator. This JSON file contains all the capabilities (all resources) in the selected region. It's probably good to update it once in a while.

4. Then import Template validator module and finally run a validation of your ARM templates. You need to specify the -TemplatePath and -CapabilitiesPath parameters:

```
//Import validate template module
Import-Module .\TemplateValidator\AzureRM.TemplateValidator.psm1

//Validate ARM template
Test-AzureRMTemplate  -TemplatePath  "c:\Users\MyPC\source\repos\MyARMTemplates\"  -
    CapabilitiesPath ".\CloudCapabilities\AzureCloudCapabilities.Json" -Verbose
1
2
3
4
5
```

```
//Import validate template module
Import-Module .\TemplateValidator\AzureRM.TemplateValidator.psm1

//Validate ARM template
```

Test-AzureRMTemplate -TemplatePath "c:\Users\MyPC\source\repos\MyARMTemplates\" -CapabilitiesPath ".\CloudCapabilities\AzureCloudCapabilities.Json" -Verbose

Because I set parameter -Verbose in the command above, I get a lot of feedback to the command line.

Verbose mode when running Test-AzureRMTemplate

Figure 1: Verbose mode when running Test-AzureRMTemplate.

Below are some examples of validation warnings and errors, that a tool can detect. Tool returns warnings like hardcoded URLs or wrong API version or errors like invalid JSON syntax or misspelled Azure resource type.

Warning message because of the hard coded value for Azure Storage URL.

Figure 2: Warning message because of the hard coded value for Azure Storage URL.

Invalid JSON syntax error message.

Figure 3: Invalid JSON syntax error message.

Misspelled Azure resource type error.

Figure 4: Misspelled Azure resource type error.

It successfully detects wrong API versioning values.

Figure 5: It successfully detects wrong API versioning values.

Not everything is validated at the moment (like Azure virtual machine type or offer) and this might be an improvement for the future.

There are some things I would like to try/see in the future regarding the ARM templates validation:

A way to run validation easily inside Visual Studio or Visual Studio Code (if you know how to do it let me know).

Integrate ARM template validation to Azure DevOps build pipeline.

Add validation for Azure resource properties like virtual machine types, offers etc. I don't see any reason why this could not be compiled through Cloud capabilities JSON file.

22) How to execute a .ps1 from another .ps1 file?

In order to find the location of a script, use `Split-Path $MyInvocation.MyCommand.Path` (make sure you use this in the script context).

The reason you should use that and not anything else can be illustrated with this example script.

```
## ScriptTest.ps1
```

```
Write-Host "InvocationName:" $MyInvocation.InvocationName
Write-Host "Path:" $MyInvocation.MyCommand.Path
```

Here are some results.

```
PS C:\Users\JasonAr> .\ScriptTest.ps1
InvocationName: .\ScriptTest.ps1
Path: C:\Users\JasonAr\ScriptTest.ps1

PS C:\Users\JasonAr> . .\ScriptTest.ps1
InvocationName: .
Path: C:\Users\JasonAr\ScriptTest.ps1

PS C:\Users\JasonAr> & ".\ScriptTest.ps1"
InvocationName: &
Path: C:\Users\JasonAr\ScriptTest.ps1
```

In **PowerShell 3.0** and later you can use the automatic variable $PSScriptRoot:

```
## ScriptTest.ps1
Write-Host "Script:" $PSCommandPath
Write-Host "Path:" $PSScriptRoot
PS C:\Users\jarcher> .\ScriptTest.ps1
Script: C:\Users\jarcher\ScriptTest.ps1
Path: C:\Users\jarcher
```

2nd method

I am calling myScript1.ps1 from myScript2.ps1 .

Assuming both of the script are at the same location, first get the location of the script by using this command :

```
$PSScriptRoot
```

And, then, append the script name you want to call like this :

```
& "$PSScriptRoot\myScript1.ps1"
```

This should work.

23) how to create container in azure storage using powershell?

24) The **New-AzureStorageContainer** cmdlet creates an Azure storage container.

25) **Examples**

26) **Example 1: Create an Azure storage container**

27) PowerShellCopyTry It

28) PS C:\>New-AzureStorageContainer -Name "ContainerName" -Permission Off

29) This command creates a storage container.

30) **Example 2: Create multiple Azure storage containers**

31) PowerShellCopyTry It

32) PS C:\>"container1 container2 container3".split() | New-AzureStorageContainer -Permission Container

33) This example creates multiple storage containers. It uses the **Split** method of the .NET **String** class and then passes the names on the pipeline.

Prerequisites

You should have Powershell ISE installed on your workstation.
You should be an authenticated user on Azure Portal.
You should possess a basic understanding of Azure.
Note
In Windows 10, PowerShell ISE comes by default, as shown below.

Create Azure Storage Account and Container using PowerShell
As soon as you open this, you will have the following window.

Create Azure Storage Account and Container using PowerShell
Now, it's time to execute the following command in the PowerShell Editor. You can execute all of them in a single go or can execute one by one.
NOTE : Use azurerm in azure command place
Step 1

Execute the command - Add-AzureAccount
It will prompt the following window. Put your credentials like username and password as depicted in the below windows.

Create Azure Storage Account and Container using PowerShell

Create Azure Storage Account and Container using PowerShell

Create Azure Storage Account and Container using PowerShell
Step 2

Execute the following command in order to create a Storage Account using the Run Selection (F8). Kindly refer to the image below for the command line.
New-AzureStorageAccount -StorageAccountName "dotnetpiperstorage" -Description "dotnetpiperstorage" -Location "East US" -Type Standard_LRS

Create Azure Storage Account and Container using PowerShell
Step 3

Execute the following command.

Get-AzureStorageAccount
The above command confirms and returns the details of a recent storage account.

Create Azure Storage Account and Container using PowerShell
Step 4

Execute the following command-
Get-AzureStorageKey -StorageAccountName "dotnetpiperstorage"

Create Azure Storage Account and Container using PowerShell
Step 4

Execute the following commands -
$storagekey = (Get-AzureStorageKey -StorageAccountName "dotnetpiperstorage").Primary
$storagecontext = New-AzureStorageContext -StorageAccountName "dotnetpiperstorageContainer" -StorageAccountKey $storagekey
$container = New-AzureStorageContainer -Name "dotnetpiperstorageContainer" -Permission Container -Context $storagecontext
Note
Any keyword created along with “$” sign is considered local variable and retains in the same session.

Step5

After creating the context and container sucessfuly, you can verify the follwoing cmdlet.

Create Azure Storage Account and Container using PowerShell
Get - AzureStorageContainer - Context $storagecontext
Add - AzureAccount
New - AzureStorageAccount - StorageAccountName "dotnetpiperstorage" - Description "dotnetpiperstorage" - Location "East US" - Type Standard_LRS
Get - AzureStorageAccount
Get - AzureStorageKey - StorageAccountName "dotnetpiperstorage"
New - AzureStorageKey - KeyType Primary - StorageAccountName "dotnetpiperstorage"#
to create Storage Container

$storagekey = (Get - AzureStorageKey - StorageAccountName "dotnetpiperstorage").Primary
$storagecontext = New - AzureStorageContext - StorageAccountName "dotnetpiperstorage" - StorageAccountKey $storagekey
$container = New - AzureStorageContainer - Name "dotnetpiperstoragecontainer" - Permission Container - Context $storagecontext
Get - AzureStorageContainer - Context $storagecontext
Now, you can also jump into Azure Portal to see the stuff you have created in the last few steps.

Open - https://portal.azure.com -> All Services ->Storage Account

Create Azure Storage Account and Container using PowerShell
Click on Storage Account. It will take you on another screen, which has potential information about all the storage accounts available within subscription as depicted below.

Create Azure Storage Account and Container using PowerShell
Click on dotnetpiperstorage. It will take you the following screen which has detailed information about Storage Account.

Create Azure Storage Account and Container using PowerShell
Click on the blobs as shown on the above window to see the container which we've created recently using the above step.

Create Azure Storage Account and Container using PowerShell
That's it.

Note
Azure Storage forces you to keep the name of the storage account in the lower font as well as for the container. Rather than jumping into ARM portal, you can easily create Azure Storage Account with these few commands.

Clean up resources
If you created a new resource group and a storage account for this exercise, yous can remove all of the assets you created by removing the resource group. This also deletes all resources contained within the group. In this case, it removes the storage account created and the resource group itself.
PowerShellCopy
Remove-AzurermResourceGroup -Name $resourceGroup

24) What is an azure SPN? Create spn using powershell?

Azure SPNs (Service Principal Names) – PowerShell. ... The **SPN** is created on the tenant (Directory) which can essentially have access to one or many **Azure** subscriptions when used

What is a 'service principal'?

An Azure service principal is a security identity used by user-created apps, services, and automation tools to access specific Azure resources. Think of it as a 'user identity' (username and password or certificate) with a specific role, and tightly controlled permissions. A service principal should only need to do specific things, unlike a general user identity. It improves security if you only grant it the minimum permissions level needed to perform its management tasks.

Verify your own permission level

First, you must have sufficient permissions in both your Azure Active Directory and your Azure subscription. You must be able to create an app in the Active Directory and assign a role to the service principal.

Create a service principal for your app

Once signed in to your Azure account, you can create the service principal. You must have one of the following ways to identify your deployed app:

- The unique name of your deployed app, such as "MyDemoWebApp" in the following examples, or
- the Application ID, the unique GUID associated with your deployed app, service, or object

Get information about your application

The Get-AzureRmADApplication cmdlet can be used to get information about your application.

Azure PowerShellCopyTry It

```
Get-AzureRmADApplication -DisplayNameStartWith MyDemoWebApp
```

outputCopy

```
DisplayName          : MyDemoWebApp
ObjectId             : 775f64cd-0ec8-4b9b-b69a-8b8946022d9f
IdentifierUris       : {http://MyDemoWebApp}
HomePage             : http://www.contoso.com
Type                 : Application
ApplicationId        : 00c01aaa-1603-49fc-b6df-b78c4e5138b4
AvailableToOtherTenants : False
AppPermissions       :
ReplyUrls            : {}
```

Create a service principal for your application

The New-AzureRmADServicePrincipal cmdlet is used to create the service principal.

Azure PowerShellCopyTry It

```
$servicePrincipal = New-AzureRmADServicePrincipal -ApplicationId 00c01aaa-1603-49fc-b6df-b78c4e5138b4
```

outputCopy

```
Secret                : System.Security.SecureString
ServicePrincipalNames : {00c01aaa-1603-49fc-b6df-b78c4e5138b4, http://MyDemoWebApp}
ApplicationId         : 00c01aaa-1603-49fc-b6df-b78c4e5138b4
DisplayName           : MyDemoWebApp
Id                    : 698138e7-d7b6-4738-a866-b4e3081a69e4
AdfsId                :
Type                  : ServicePrincipal
```

From here, you can either directly use the $servicePrincipal.Secret property in Connect-AzureRmAccount (see "Sign in using the service principal" below), or you can convert this SecureString to a plain text string for later usage:

Azure PowerShellCopyTry It

```
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($servicePrincipal.Secret)
$password = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
[Runtime.InteropServices.Marshal]::ZeroFreeBSTR($BSTR)
```

Sign in using the service principal

You can now sign in as the new service principal for your app using the *appId* you provided and *password* that was automatically generated. You also need the Tenant ID for the service principal. Your Tenant ID is displayed when you sign into Azure with your personal credentials. To sign in with a service principal, use the following commands:

Azure PowerShellCopyTry It

```
$cred = New-Object System.Management.Automation.PSCredential ("00c01aaa-1603-49fc-b6df-b78c4e5138b4", $servicePrincipal.Secret)
Connect-AzureRmAccount -Credential $cred -ServicePrincipal -TenantId XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
```

After a successful sign-in you see output like:

outputCopy

```
Environment           : AzureCloud
Account               : 00c01aaa-1603-49fc-b6df-b78c4e5138b4
TenantId              : XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX
SubscriptionId        :
SubscriptionName      :
CurrentStorageAccount :
```

Congratulations! You can use these credentials to run your app. Next, you need to adjust the permissions of the service principal.

Managing roles

Note

Azure Role-Based Access Control (RBAC) is a model for defining and managing roles for user and service principals. Roles have sets of permissions associated with them, which determine the resources a principal can read, access, write, or manage. For more information on RBAC and roles, see **RBAC: Built-in roles**.

Azure PowerShell provides the following cmdlets to manage role assignments:

- Get-AzureRmRoleAssignment
- New-AzureRmRoleAssignment
- Remove-AzureRmRoleAssignment

The default role for a service principal is **Contributor**. It may not be the best choice depending on the scope of your app's interactions with Azure services, given its broad permissions. The **Reader** role is more restrictive and can be a good choice for read-only apps. You can view details on role-specific permissions or create custom ones through the Azure portal.

In this example, we add the **Reader** role to our prior example, and delete the **Contributor** one:

Azure PowerShellCopyTry It

```
New-AzureRmRoleAssignment -ResourceGroupName myRG -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4 -RoleDefinitionName Reader
```

outputCopy

```
RoleAssignmentId   : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myRG/providers/Microsoft.Authorization/roleAssignments/818892f2-d075-46a1-a3a2-3a4e1a12fcd5
Scope              : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myRG
DisplayName        : MyDemoWebApp
SignInName         :
RoleDefinitionName : Reader
RoleDefinitionId   : b24988ac-6180-42a0-ab88-20f7382dd24c
ObjectId           : 698138e7-d7b6-4738-a866-b4e3081a69e4
ObjectType         : ServicePrincipal
```

Azure PowerShellCopyTry It

```
Remove-AzureRmRoleAssignment -ResourceGroupName myRG -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4 -RoleDefinitionName Contributor
```

To view the current roles assigned:

Azure PowerShellCopyTry It

```
Get-AzureRmRoleAssignment -ResourceGroupName myRG -ObjectId 698138e7-d7b6-4738-a866-b4e3081a69e4
```

outputCopy

```
RoleAssignmentId   : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myRG/providers/Microsoft.Authorization/roleAssignments/0906bbd8-9982-4c03-8dae-aeaae8b13f9e
```

Scope : /subscriptions/XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX/resourceGroups/myRG
DisplayName : MyDemoWebApp
SignInName :
RoleDefinitionName : Reader
RoleDefinitionId : acdd72a7-3385-48ef-bd42-f606fba81ae7
ObjectId : 698138e7-d7b6-4738-a866-b4e3081a69e4
ObjectType : ServicePrincipal

2nd method

Developing with Azure Resource Manager - Part 1 - Creating a Service Principal for your AAD using PowerShellTOBIAS ZIMMERGREN / FEBRUARY 20, 2016
This article is part of a series. Here's a list of all available parts.

Part 0: Introduction to the article series
Part 1: Create an AzureRm Active Directory (AAD) Application using PowerShell
Part 2: Getting started with the AzureRm PowerShell cmdlets
Part 3: Build an application using C# which is using the Azure Resource Manager API's
Part 4: Tip: Azure Resource Explorer Tool
Part 5: Tip: Get all available api-version alternatives for the ARM endpoints
Part 6: Move Azure Resources from one Resource Group to another
Part 7: Download an Azure Publishing Profile (xml) programmatically using REST
Part 8: Programmatically export Resource Group template using the REST API
Introduction
Welcome to Part 1 in the Developing with Azure series. In this post I will walk you through how to use PowerShell in order to create an Azure Active Directory (AAD) Application and then also create a new Service Principal which we'll use to authenticate and authorize requests to the Azure Resource Manager.

When we talk about various auth alternatives, there's a lot of options. We'll focus on using an account/service principal for our AAD Application but in another post we'll investigate how we can set up the authentication using a Certificate.

Background: Why use a Service Principal?
On of the considerations I've faced during the work I've done with the Azure Resource Manager is how I would authenticate my requests. There's really three options that comes to mind, and those are:

Use your Username and Password (Your own account)
Use a Certificate
Use a Service Principal
I've tried all fo the above methods, and find that using a Service Principal is the easiest way to manage and control the permissions in Azure. Since Azure supports RBAC (Role-Based Access Control), you can easily assign specific permissions or limitations on what the service principal or account should be allowed to do.

Benefits of RBAC with a Service Principal:

Fine-grained permission configuration
An account limited to a single purpose using role-based access control
Remove any unecessary permissions and privileges
Create a unique service principal with limited permissions per service
etc..
Scenarios when this type of account comes in handy:

Continuous Integration (CI) from a Build Server using PowerShell
Release Manager automation with PowerShell for automatic deployments etc
Custom API's, Applications or Services you build that need to authenticate and work with the Azure Resource Manager API's
Etc..
Authenticate to Azure Resource Management from PowerShell
First of all, make sure you're running Azure PowerShell 1.0 or later. Previous editions are targeting older versions with deprecated cmdlets for Azure Resource Management.

Next, we also need to have the Azure Resource Manager Cmdlets (please visit the link and install..).

In order to get started, we need to make sure that the powershel session is authenticated. We'll do this by using the following command:

Login-AzureRmAccount
It should bring up an Azure Sign-in dialogue. Use it to sign in with your account (or whichever account has the required permissions to manage your subscription):

Login-AzureRmAccount cmdlet in PowerShell

The output of the command should be some data about your session, like this:

```
PS C:\Users\tobia> Login-AzureRmAccount


Environment          : AzureCloud
Account              : YourEmail@yourdomain.com
TenantId             : YourGuid
SubscriptionId       : YourGuid
CurrentStorageAccount :
```

Creating an AzureRM AD Application

Great, we're using PowerShell to authenticate ourselves to our subscription with the aforementioned command. Now it's time to create a new AAD Application (Azure Active Directory). This will be an Azure Resource Manager application, and we'll use the new AzureRm cmdlets. As I mentioned in the pre-reqs, make sure you've got at least Azure PowerShell 1.0 installed.

The cmdlet for creating a new AAD Application is:

```
New-AzureRmADApplication
```

You should run it like this:

```
$myAADApp = New-AzureRmADApplication
    -DisplayName "ZimmerAADApp1"
    -HomePage "https://zimmergren.net/aadapp1"
    -IdentifierUris "https://zimmergren.net/aadapp1" -Password "My Pass"
```

After running this command, you need to check that the application was successfully created. Do this by simply checking the value of the $myAADApp variable (or whatever you named it):

```
PS C:\Users\tobia> $myAADApp


DisplayName          : ZimmerAADApp1
Type                 : Application
ApplicationId        : YourGuid
ApplicationObjectId  : YourGuid
AvailableToOtherTenants : False
AppPermissions       : {}
IdentifierUris       : {https://zimmergren.net/aadapp1}
ReplyUrls            : {}
```

NOTE: You should copy your ApplicationId from the output and save for later.

Creating a Service Principal

We have created our AzureRm AD Application and we're ready to create an account which can get access to this application in order to later work with the APIs. We're doing this with something called a Service Principal, which essentially is a type of service account.

New-AzureRmADServicePrincipal -ApplicationId <your application id>

*NOTE: The **ApplicationId *you should use is the one you got from the previous query and saved.

Next step is to assign a Role Assignment to the Application. We'll do this using the New-AzureRmRoleAssignment command. I'm giving Contributor access, but you can choose whatever access level you want:

New-AzureRmRoleAssignment
-RoleDefinitionName Contributor
-ServicePrincipalName "<application id>"

The result of this should be something like this:

RoleAssignmentId : /subscriptions/A GUID/providers/Microsoft.Authorization/roleAssignments/<GUID>
Scope : /subscriptions/A GUID
DisplayName : ZimmerAADApp1
SignInName :
RoleDefinitionName : Contributor
RoleDefinitionId : A GUID
ObjectId : A GUID
ObjectType : ServicePrincipal

In order to move on to the next step, we first need to get the subscription id. The easiest thing to do in achieving this is:

$mySubscription = Get-AzureRmSubscription

This will result in something like this:

C:\Users\tobias> $mySubscription

SubscriptionName : Production
SubscriptionId : <a guid>
TenantId : <a guid>

State : Enabled

From here, make a note of the SubscriptionId and the TenantId. Both are required for the next commands to execute.

Authenticate using your new Service Principal to verify

In order to try our new Service Principal out, we need to try and authenticate the requests through that account.

To do this, create a new credential object like so:

```
$svcPrincipalCredentials = Get-Credential
```

It should yield a login image like this:

Azure Resource Manager Authenticating the service principal

Enter the ApplicationId as username and the Password that you configured in the beginning of this article.

Awesome - you have fetched the Credentials for the new Service Principal. That means we can now start logging in to the Azure Resource Manager with any approach we want. In this post we'll just verify that we can access it using PowerShell:

```
Login-AzureRmAccount
    -Credential $svcPrincipalCredentials
    -ServicePrincipal
    -TenantId "<your tenant id guid>"
```

It should yield an output like this, and verify that you are signed in using the new service principal account (the account guid..):

```
Environment          : AzureCloud
Account              : <your guid>
TenantId             : <your guid>
SubscriptionId       : <your guid>
CurrentStorageAccount :
```

Run the Azure Resource Manager PowerShell commands

Once you've hooked up everything as described above, you're ready to run some PowerShell cmdlets (and currently also have a session which is authenticated).

There's some Azure PowerShell cmdlets for the Resource Manager which are described in more detail available here.

As an example, you can now list all your resource groups by running this command:

Get-AzureRmResourceGroup | ft ResourceGroupName,Location
This should yield a result like this:

ARM results

Summary
Bingo! We have achieve the following result:

Sign in and authenticate with Azure using PowerShell
Create a new Azure Resource Manager AAD Application
Create a Service Principal (essentially a "service account")
Assign the Service Principal with (in my case) Contributor permissions
Verified that we could login using the new service principal through PowerShell with the Login-AzureRmAccount cmdlet
Run the Azure Resource Manager cmdlets successfully to fetch amazing magic from our Azure subscription
This means we're good to go, and can now start building any type of application which can authenticate (using the Service Principal's ID and Password) and work with the Azure Resource Manager API's. Which in turn means that if we want to continue and evolve what we've just done, we can start building C# applications, web api's, backend services or whatever you prefer.

26) What is artifacts in Azure DevOps?

A release is a collection of **artifacts** in your **DevOps** CI/CD processes. An **artifact** is a deployable component of your application. **Azure** Pipelines can deploy **artifacts** that are produced by a wide range of **artifact** sources, and stored in different types of **artifact** repositories

27)
Publish Maven artifacts using Azure DevOps Services and TFS
Azure DevOps Services | **TFS 2018**
Publish Maven artifacts to a feed in **Azure Artifacts** to share them with your team and organization.
To publish a Maven artifact, you'll need to have a Maven artifact to publish on your local machine. If you don't have one, you can generate one by running the following command:

CommandCopy

```
mvn -B archetype:generate -DarchetypeGroupId="org.apache.maven.archetypes" -DgroupId="MyGroup" -DartifactId="myFirstApp"
```

1. Set up the Maven client with your feed.
2. Navigate to the directory containing your Maven artifact's **pom.xml** file. If you've just created an artifact, the **pom.xml** file will be in the *myFirstApp* folder.
3. From the **Connect to feed** dialog in Azure DevOps Services, copy the <repository> information. Paste it into your **pom.xml** file twice (see the sample file above):

- Between the <repositories> tags.
- Between the <distributionManagement> tags.

From the directory containing your **pom.xml** file, run the command mvn deploy. The Maven artifact should appear in your feed.

Sample pom.xml file:

XMLCopy

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd">
 <modelVersion>4.0.0</modelVersion>
 <groupId>MyGroup</groupId>
 <artifactId>myFirstApp</artifactId>
 <packaging>jar</packaging>
 <version>1.0-SNAPSHOT</version>
 <name>myFirstApp</name>
 <url>http://maven.apache.org</url>
 <dependencies>
   <dependency>
     <groupId>junit</groupId>
     <artifactId>junit</artifactId>
     <version>3.8.1</version>
     <scope>test</scope>
   </dependency>
 </dependencies>
 <repositories>
  <!-- Copy this section from the Maven section of the "Connect to Feed" dialog -->
   <repository>
     <id>dev.azure.com-mseng-zcalvinmaven</id>
     <url>https://pkgs.dev.azure.com/mseng/_packaging/zCalvinMaven2/maven/v1</url>
```

```
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </repository>
  </repositories>
  <distributionManagement>
   <!-- Copy this section from the Maven section of the "Connect to Feed" dialog -->
    <repository>
      <id>dev.azure.com-mseng-zcalvinmaven</id>
      <url>https://pkgs.dev.azure.com/mseng/_packaging/zCalvinMaven2/maven/v1</url>
      <releases>
        <enabled>true</enabled>
      </releases>
      <snapshots>
        <enabled>true</enabled>
      </snapshots>
    </repository>
  </distributionManagement>
</project>
```

Important

Th

28) steps to install self hosted agents in azure devops?

Self-hosted Windows agents

- 08/15/2019
- 16 minutes to read
-
  -
  -
  -

-
- +3

Azure Pipelines | TFS 2018 | TFS 2017 | TFS 2015 | Previous versions (XAML builds)

To build and deploy Windows, Azure, and other Visual Studio solutions you'll need at least one Windows agent. Windows agents can also build Java and Android apps.

Before you begin:

- If your code is in Azure Pipelines and a Microsoft-hosted agent meets your needs, you can skip setting up a self-hosted Windows agent.
- If your code is in an on-premises Team Foundation Server (TFS) 2015 server, see Deploy an agent on Windows for on-premises TFS 2015.
- Otherwise, you've come to the right place to set up an agent on Windows. Continue to the next section.

Learn about agents

If you already know what an agent is and how it works, feel free to jump right in to the following sections. But if you'd like some more background about what they do and how they work, see Azure Pipelines agents.

Check prerequisites

Make sure your machine is prepared with our Windows system prerequisites.

If you're building from a Subversion repo, you must install the Subversion client on the machine.

You should run agent setup manually the first time. After you get a feel for how agents work, or if you want to automate setting up many agents, consider using unattended config.

Hardware specs

The hardware specs for your agents will vary with your needs, team size, etc. It's not possible to make a general recommendation that will apply to everyone. As a point of reference, the Azure DevOps team builds its hosted agents using the hosted agents. On the other hand, the bulk of the Azure DevOps code is built by 24-core server class machines running 4 agents apiece.

Prepare permissions

Decide which user you'll use

As a one-time step, you must register the agent. Someone with permission to administer the agent queue must complete these steps. The agent will not use this person's credentials in everyday operation, but they're required to complete registration. Learn more about how agents communicate.

Authenticate with a personal access token (PAT)

1. Sign in with the user account you plan to use in either your Azure DevOps organization (https://dev.azure.com/{your_organization}) or your Team Foundation Server web portal (https://{your-server}:8080/tfs/).

2. From your home page, open your profile. Go to your security details.

3. Create a personal access token.

4. For the scope select **Agent Pools (read, manage)** and make sure all the other boxes are cleared. If it's a deployment group agent, for the scope select **Deployment group (read, manage)** and make sure all the other boxes are cleared.
5. Copy the token. You'll use this token when you configure the agent.

Confirm the user has permission

Make sure the user account that you're going to use has permission to register the agent.

Is the user an Azure DevOps organization owner or TFS server administrator? **Stop here**, you have permission.

Otherwise:

1. Open a browser and navigate to the **Agent pools** tab for your Azure Pipelines organization or TFS server:

a. Choose **Azure DevOps**, **Organization settings**.

b. Choose **Agent pools**.

Click the pool on the left side of the page and then click **Roles**.

If the user account you're going to use is not shown, then get an administrator to add it. The administrator can be an agent pool administrator, an Azure DevOps organization owner, or a TFS server administrator.

If it's a deployment group agent, the administrator can be an deployment group administrator, an Azure DevOps organization owner, or a TFS server administrator.

You can add a user to the deployment group administrator role in the **Security** tab on the **Deployment Groups** page in **Azure Pipelines**.

Note

If you see a message like this: **Sorry, we couldn't add the identity. Please try a different identity.**, you probably followed the above steps for an organization owner or TFS server administrator. You don't need to do anything; you already have permission to administer the agent queue.

Download and configure the agent

Azure Pipelines

1. Log on to the machine using the account for which you've prepared permissions as explained above.
2. In your web browser, sign in to Azure Pipelines, and navigate to the **Agent pools** tab:

a. Choose **Azure DevOps**, **Organization settings**.

b. Choose **Agent pools**.

Select the **Default** pool, select the **Agents** tab, and choose **New agent**.

On the **Get the agent** dialog box, choose **Windows**.

On the left pane, select the processor architecture of the installed Windows OS version on your machine. The x64 agent version is intended for 64-bit Windows, whereas the x86 version is intended for 32-bit Windows. If you aren't sure which version of Windows is installed, follow these instructions to find out.

On the right pane, click the **Download** button.

Follow the instructions on the page to download the agent.

Unpack the agent into the directory of your choice. Then run config.cmd. This will ask you a series of questions to configure the agent.

Note

We strongly recommend you configure the agent from an elevated PowerShell window. If you want to configure as a service, this is **required**.

Server URL and authentication

When setup asks for your server URL, for Azure DevOps Services, answer https://dev.azure.com/{your-organization}.

When setup asks for your authentication type, choose **PAT**. Then paste the PAT token you created into the command prompt window.

Note

When using PAT as the authentication method, the PAT token is only used during the initial configuration of the agent. Later, if the PAT expires or needs to be renewed, no further changes are required by the agent.

Choose interactive or service mode

For guidance on whether to run the agent in interactive mode or as a service, see Agents: Interactive vs. service.

If you choose to run as a service (which we recommend), the username you run as should be 20 characters or less.

Run the agent

If you configured the agent to run interactively, to run it:

psCopy

.\run.cmd

If you configured the agent to run as a service, it starts automatically. You can view and control the agent running status from the services snap-in. Run services.msc and look for one of:

- "Azure Pipelines Agent (*name of your agent*)".
- "VSTS Agent (*name of your agent*)".
- "vstsagent.(*organization name*).(*name of your agent*)".

Note

If you need to change the agent's logon account, don't do it from the Services snap-in. Instead, see the information below to re-configure the agent.

To use your agent, run a job using the agent's pool. If you didn't choose a different pool, your agent will be in the **Default** pool.

Replace an agent

To replace an agent, follow the *Download and configure the agent* steps again.

When you configure an agent using the same name as an agent that already exists, you're asked if you want to replace the existing agent. If you answer Y, then make sure you remove the agent (see below) that you're replacing. Otherwise, after a few minutes of conflicts, one of the agents will shut down.

Remove and re-configure an agent

To remove the agent:

psCopy

.\config remove

After you've removed the agent, you can configure it again.

Unattended config

The agent can be set up from a script with no human intervention. You must pass --unattended and the answers to all questions.

To configure an agent, it must know the URL to your organization or collection and credentials of someone authorized to set up agents. All other responses are optional. Any command-line parameter can be specified using an environment variable instead: put its name in upper case and prepend VSTS_AGENT_INPUT_. For example, VSTS_AGENT_INPUT_PASSWORD instead of specifying --password.

Required options

- --unattended - agent setup will not prompt for information, and all settings must be provided on the command line
- --url <url> - URL of the server. For example: https://dev.azure.com/myorganization or http://my-azure-devops-server:8080/tfs
- --auth <type> - authentication type. Valid values are:
  - pat (Personal access token)
  - negotiate (Kerberos or NTLM)
  - alt (Basic authentication)
  - integrated (Windows default credentials)

Authentication options

- If you chose --auth pat:
  - --token <token> - specifies your personal access token
- If you chose --auth negotiate or --auth alt:
  - --userName <userName> - specifies a Windows username in the format domain\userName or userName@domain.com
  - --password <password> - specifies a password

Pool and agent names

- --pool <pool> - pool name for the agent to join
- --agent <agent> - agent name
- --replace - replace the agent in a pool. If another agent is listening by the same name, it will start failing with a conflict

Agent setup

- --work <workDirectory> - work directory where job data is stored. Defaults to _work under the root of the agent directory. The work directory is owned by a given agent and should not share between multiple agents.
- --acceptTeeEula - accept the Team Explorer Everywhere End User License Agreement (macOS and Linux only)
- --once - accept only one job and then spin down gracefully (useful for running on a service like Azure Container Instances)

Windows-only startup

- --runAsService - configure the agent to run as a Windows service (requires administrator permission)
- --runAsAutoLogon - configure auto-logon and run the agent on startup (requires administrator permission)
- --windowsLogonAccount <account> - used with --runAsService or --runAsAutoLogon to specify the Windows user name in the format domain\userName or userName@domain.com
- --windowsLogonPassword <password> - used with --runAsService or --runAsAutoLogon to specify Windows logon password
- --overwriteAutoLogon - used with --runAsAutoLogon to overwrite the existing auto logon on the machine
- --noRestart - used with --runAsAutoLogon to stop the host from restarting after agent configuration completes

Deployment group only

- --deploymentGroup - configure the agent as a deployment group agent
- --deploymentGroupName <name> - used with --deploymentGroup to specify the deployment group for the agent to join
- --projectName <name> - used with --deploymentGroup to set the project name
- --addDeploymentGroupTags - used with --deploymentGroup to indicate that deployment group tags should be added
- --deploymentGroupTags <tags> - used with --addDeploymentGroupTags to specify the comma separated list of tags for the deployment group agent - for example "web, db"

.\config --help always lists the latest required and optional responses.

30) what is **Infrastructure** as **code** (IaC)

Infrastructure as **code** (IaC) is the process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware

configuration or interactive configuration tools. ... IaC supports IaaS, but should not be confused with it.

An ARM template is a JSON file used to configure and deploy various **Azure** resources like VMs, AKS clusters, web apps, VNets, functions, and more to the **Azure** cloud. The basic idea behind **Infrastructure**-as-**Code** (IAC) is to provide the **infrastructure** through automation rather than using manual processes

31) different types of tasks in azure release pipeline?

Key concepts for new Azure Pipelines users

- 07/24/2019
- 3 minutes to read
-
  -
  -
  -
  -
  -
  - +3

Azure Pipelines

Learn about the key concepts and components that are used in Azure Pipelines. Understanding the basic terms and parts of Azure Pipelines helps you further explore how it can help you deliver better code more efficiently and reliably.

Agent

When your build or deployment runs, the system begins one or more jobs. An agent is installable software that runs one job at a time.

For more in-depth information about the different types of agents and how to use them, see Build and release agents.

Approvals

Approvals define a set of validations required before a deployment can be performed. Manual approval is a common check performed to control deployments to production environments. When checks are configured on an environment, pipelines will stop before

starting a stage that deploys to the environment until all the checks are completed successfully.

Artifact

An artifact is a collection of files or packages published by a run. Artifacts are made available to subsequent tasks, such as distribution or deployment. For more information, see Artifacts in Azure Pipelines.

Continuous delivery

Continuous delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production stages. Deploying and testing in multiple stages helps drive quality. Continuous integration systems produce deployable artifacts, which includes infrastructure and apps. Automated release pipelines consume these artifacts to release new versions and fixes to existing systems. Monitoring and alerting systems run constantly to drive visibility into the entire CD process. This process ensures that errors are caught often and early.

Continuous integration

Continuous integration (CI) is the practice used by development teams to simplify the testing and building of code. CI helps to catch bugs or problems early in the development cycle, which makes them easier and faster to fix. Automated tests and builds are run as part of the CI process. The process can run on a set schedule, whenever code is pushed, or both. Items known as artifacts are produced from CI systems. They're used by the continuous delivery release pipelines to drive automatic deployments.

Environment

An environment is a collection of resources, where you deploy your application. It can contain one or more virtual machines, containers, web apps, or any service that's used to host the application being developed. A pipeline might deploy the app to one or more environments after build is completed and tests are run.

Job

A stage contains one or more jobs. Each job runs on an agent. A job represents an execution boundary of a set of steps. All of the steps run together on the same agent. For example, you might build two configurations - x86 and x64. In this case, you have one build stage and two jobs.

Pipeline

A pipeline defines the continuous integration and deployment process for your app. It's made up of one or more stages. It can be thought of as a workflow that defines how your test, build, and deployment steps are run.

Run

A run represents one execution of a pipeline. It collects the logs associated with running the steps and the results of running tests.

Stage

A stage is a logical boundary in the pipeline. It can be used to mark separation of concerns (e.g., Build, QA, and production). Each stage contains one or more jobs.

Step

A step is the smallest building block of a pipeline. For example, a pipeline might consist of build and test steps. A step can either be a script or a task. A task is simply a pre-created script offered as a convenience to you. To view the available tasks, see the Build and release tasks reference. For information on creating custom tasks, see Create a custom task.

Trigger

A trigger is something that's set up to tell the pipeline when to run. You can configure a pipeline to run upon a push to a repository, at scheduled times, or upon the completion of another build. All of these actions are known as triggers. For more information, see build triggers and release triggers.

32) complete cicd pipeline

What's covered in this lab?

In this lab, you will

- Create an ASP.NET sample DevOps project using **Azure DevOps Project** feature in Azure
- Examine the CI/CD pipelines configured by **Azure DevOps Project**
- Commit the code changes and execute CI/CD
- Configure Azure Application Insights monitoring

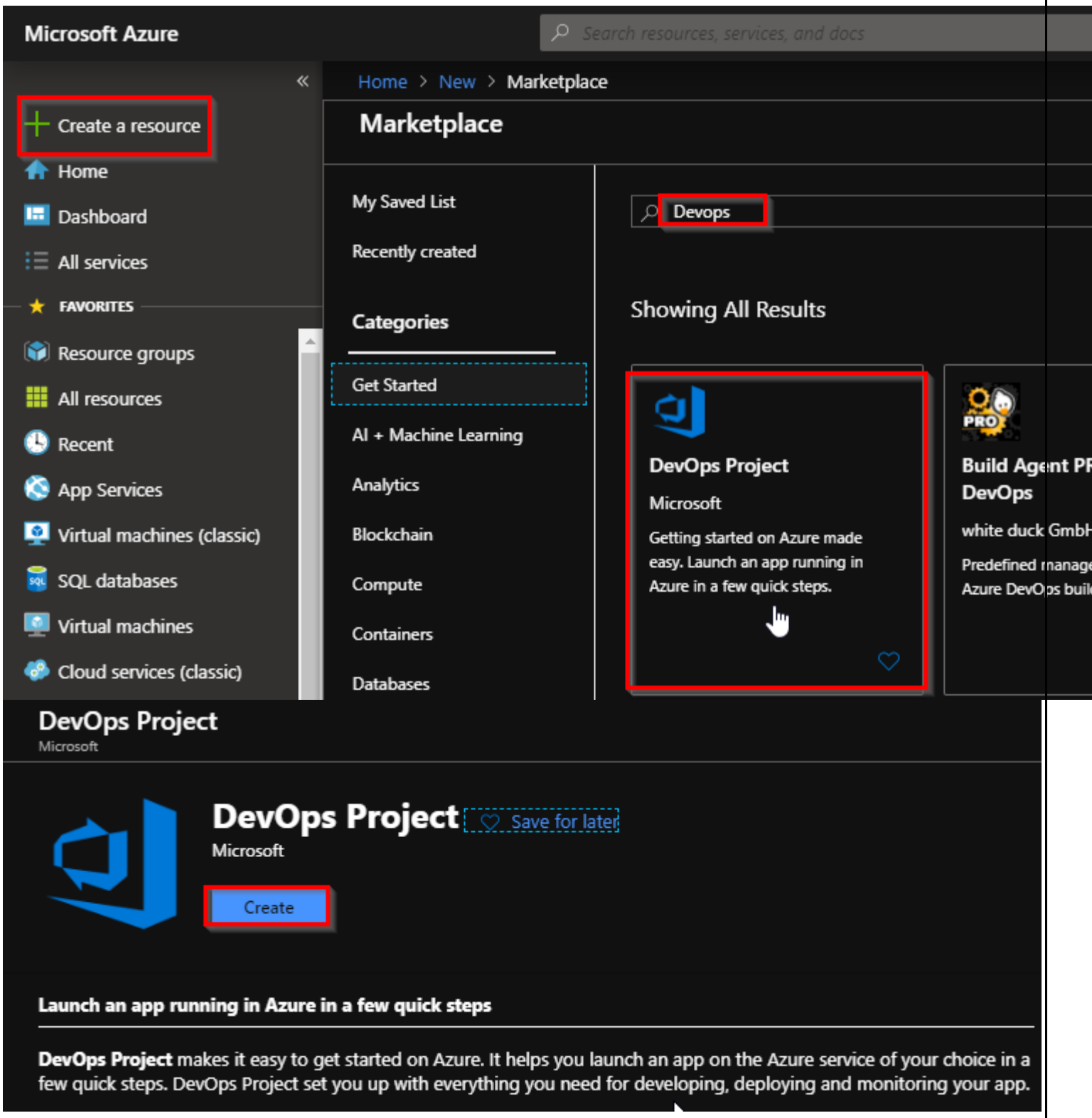
Pre-requisites for the lab

1. **Microsoft Azure Account**: You will need a valid and active Azure account for the Azure labs. If you do not have one, you can sign up for a free trial
- If you are a Visual Studio Active Subscriber, you are entitled for a $50-$150 credit per month. You can refer to this link to find out more including how to activate and start using your monthly Azure credit.
- If you are not a Visual Studio Subscriber, you can sign up for the FREE Visual Studio Dev Essentialsprogram to create **Azure free account** (includes 1 year of free services, $200 for 1st month).

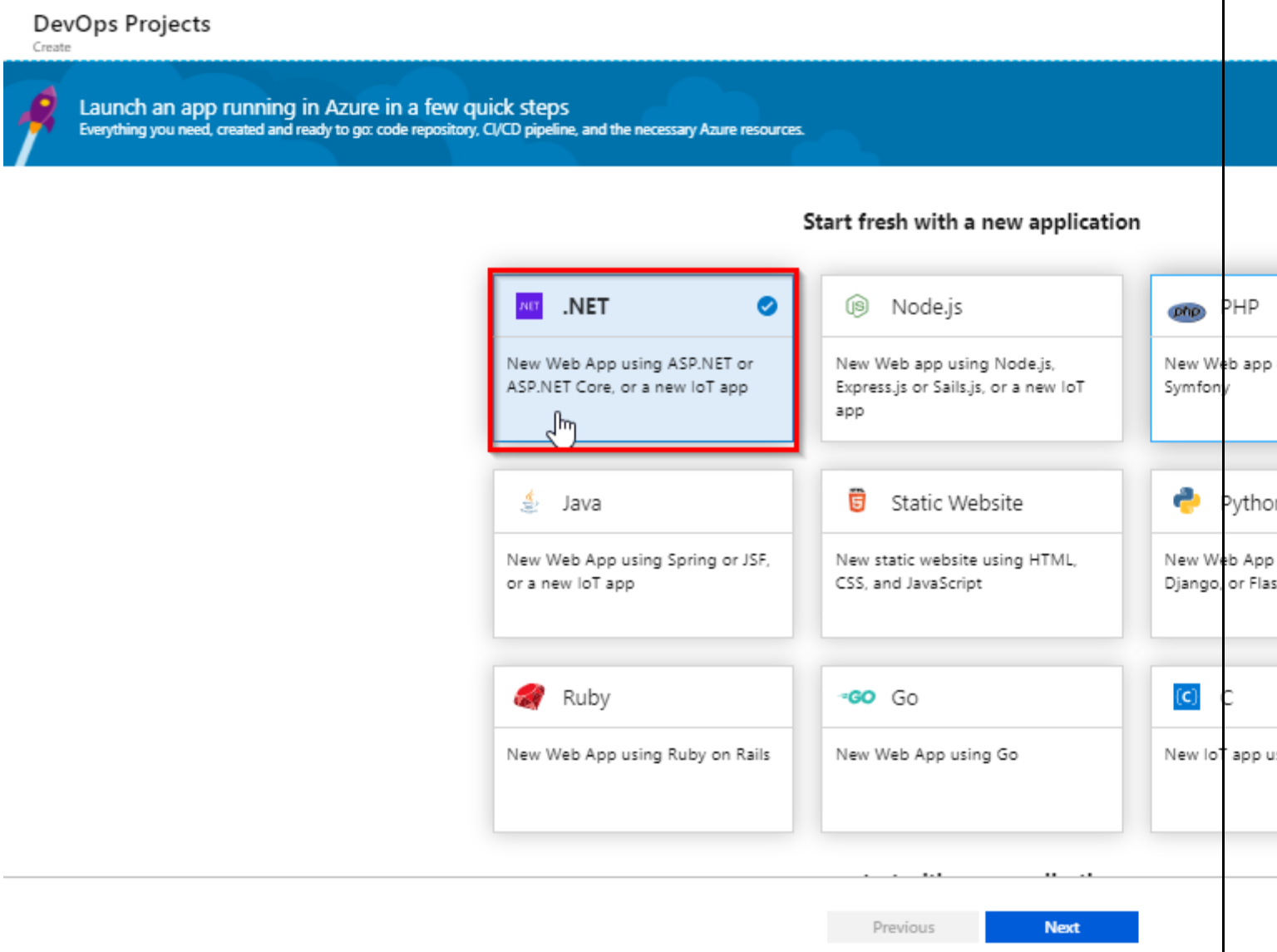
  You will need an Azure DevOps account. If you do not have one, you can sign up for free here.

Exercise 1: Setting up a sample ASP.NET project using Azure DevOps Project

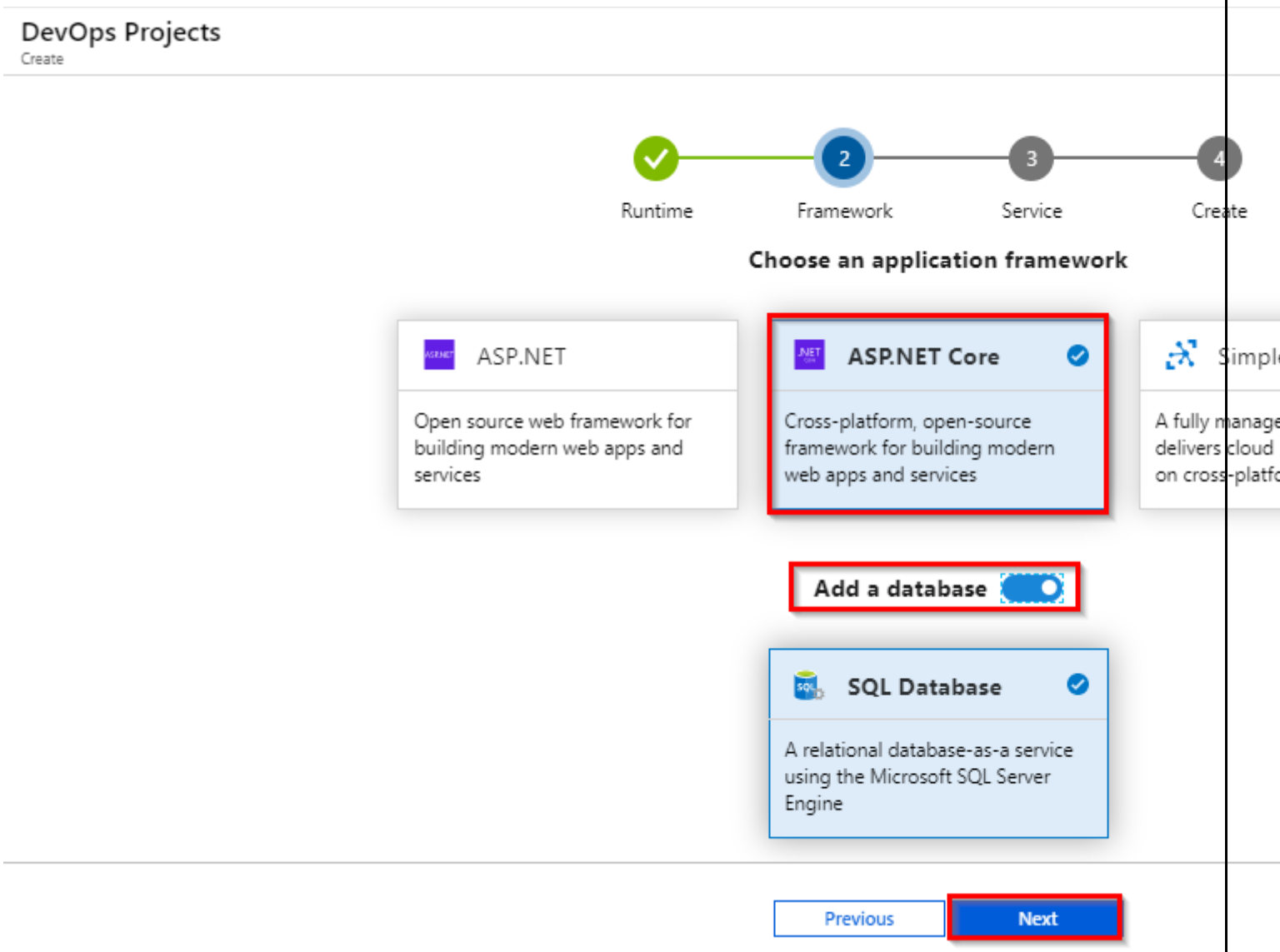
1. Sign into the Microsoft Azure portal.
2. Choose the **+ Create a resource** icon in the left navigation bar, then search for **DevOps project**. Then choose **DevOps Project** in the list. Select **Create**.



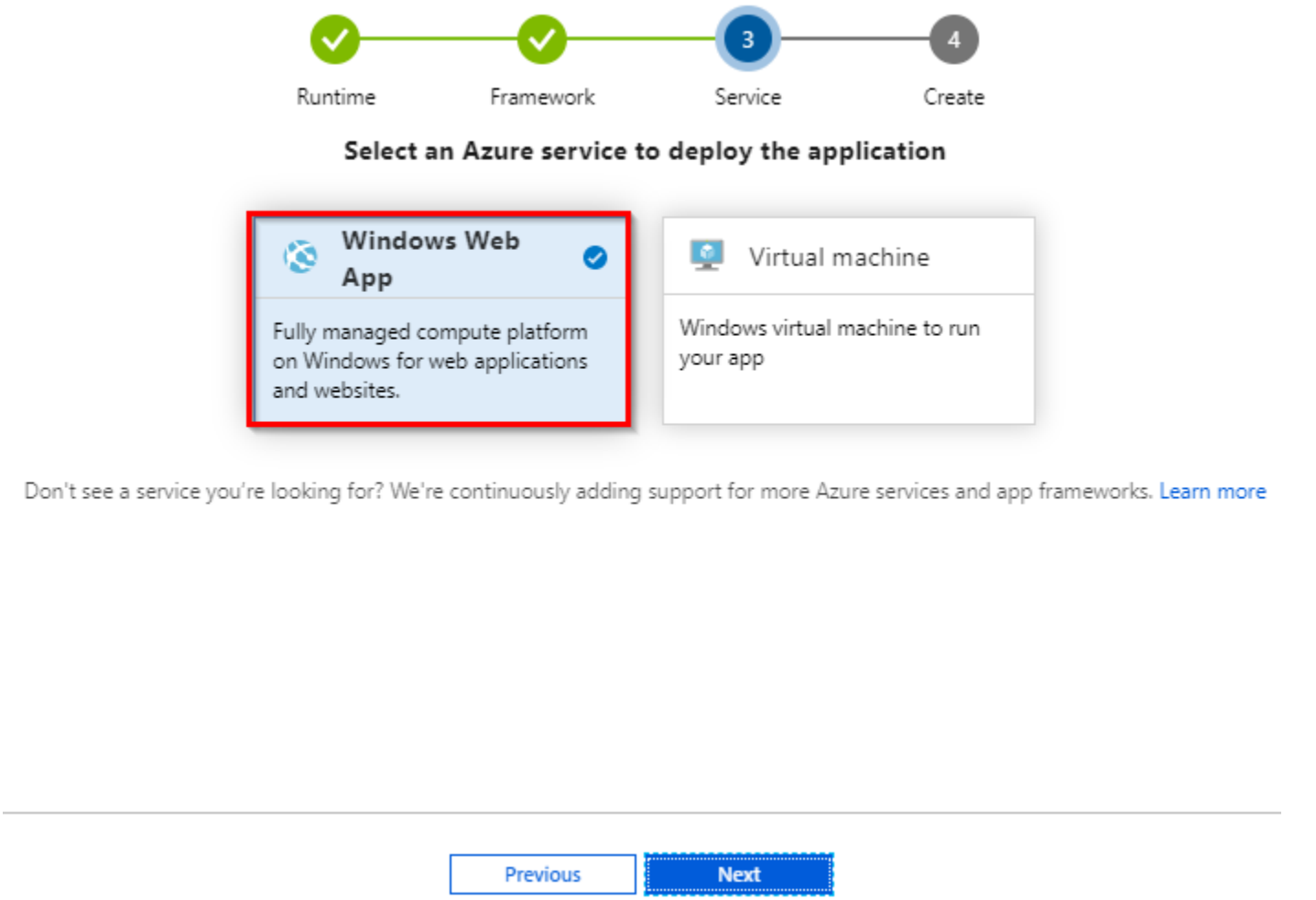
3. Select the **.NET** sample application and click **Next**.



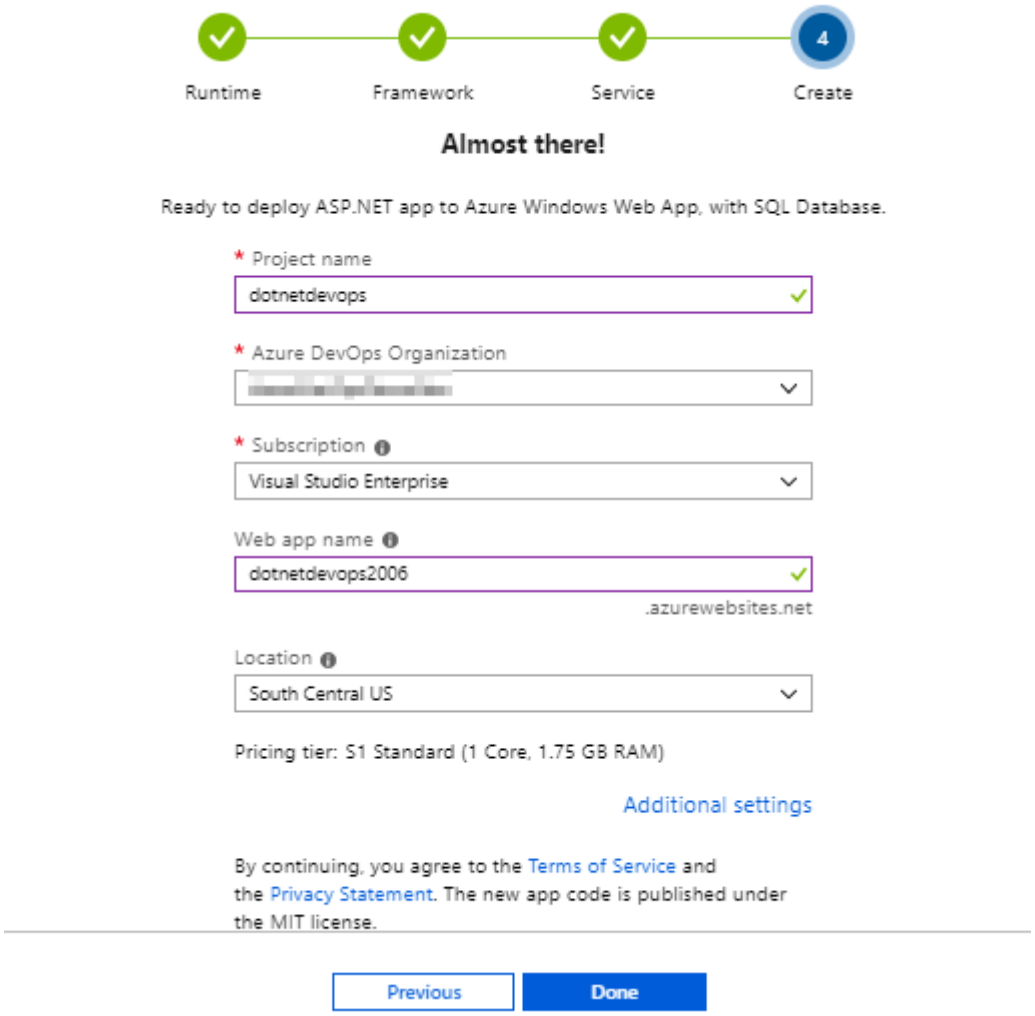
4. The .NET samples include a choice of either the open source ASP.NET framework or the cross-platform .NET Core framework. Select the .NET Core application framework. This sample is an ASP.NET Core MVC application. And also enable **Add a database** toggle to add the database to the application. When you're done, choose **Next**.



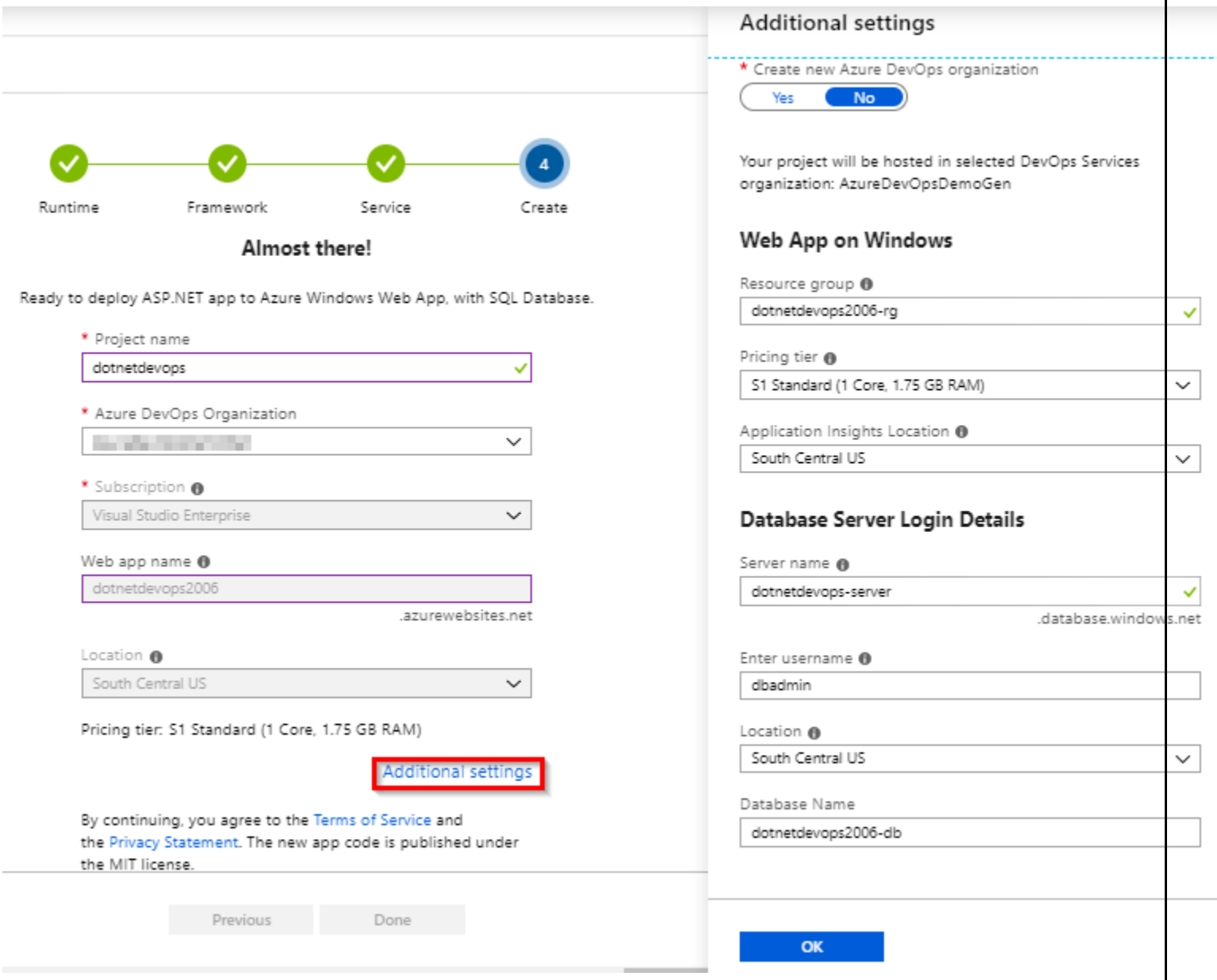
5. Web App on Windows is the default deployment target. You can optionally choose Virtual Machine also. When you're done, choose **Next**.



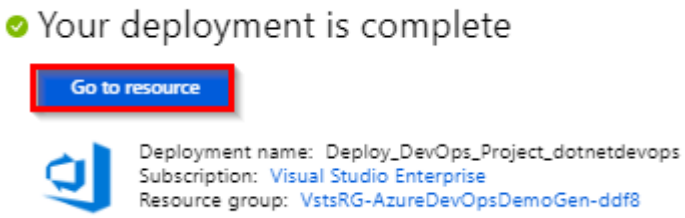
6. Select your Azure DevOps organization and choose a **name** for your project and Web app. When you're done, choose **Done**.



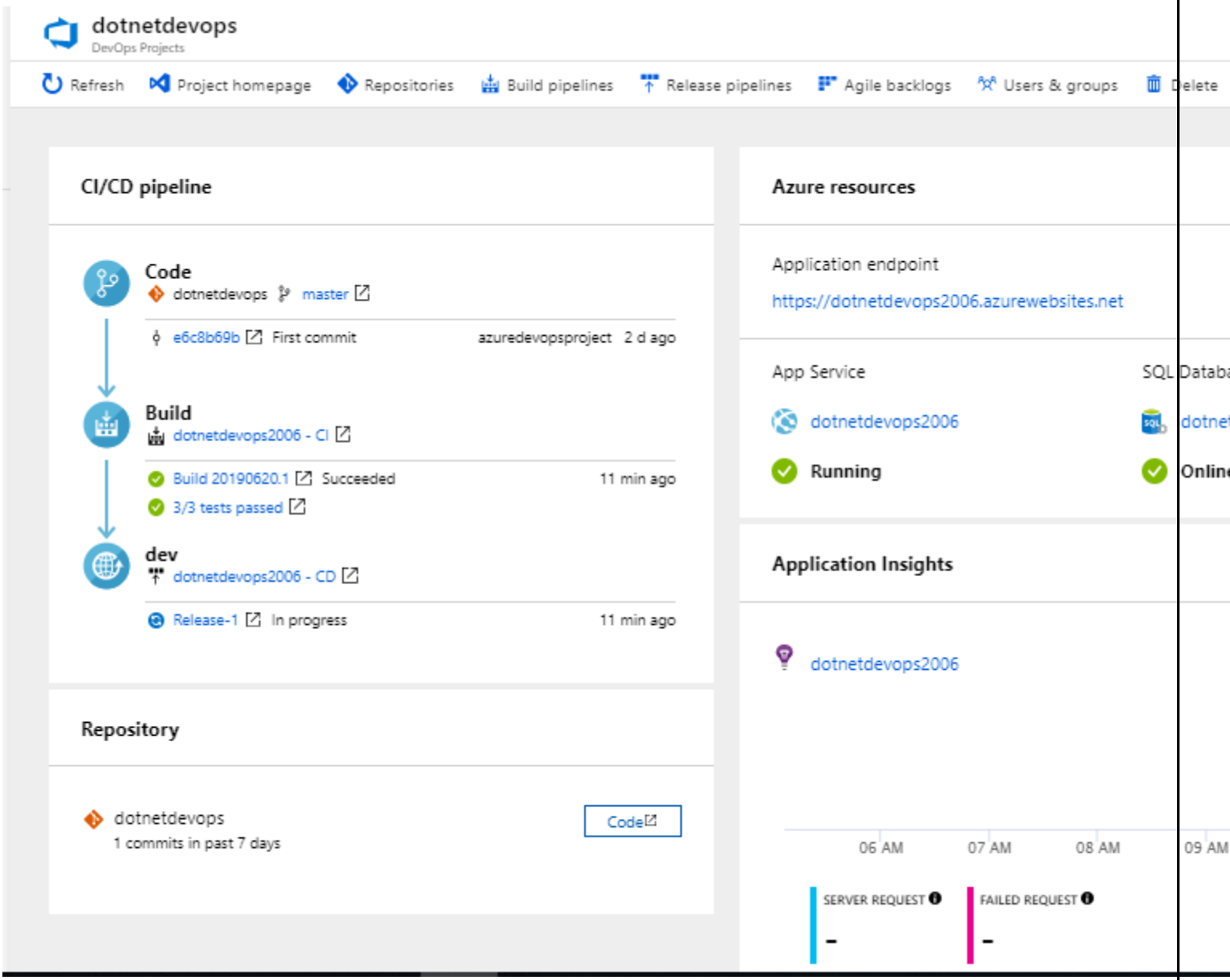
You can click on **Additional Settings** if you would like to edit web app and database parameters



7. Once the deployment completes, click **Go to resource**.



8. DevOps project dashboard loads as shown in below image.

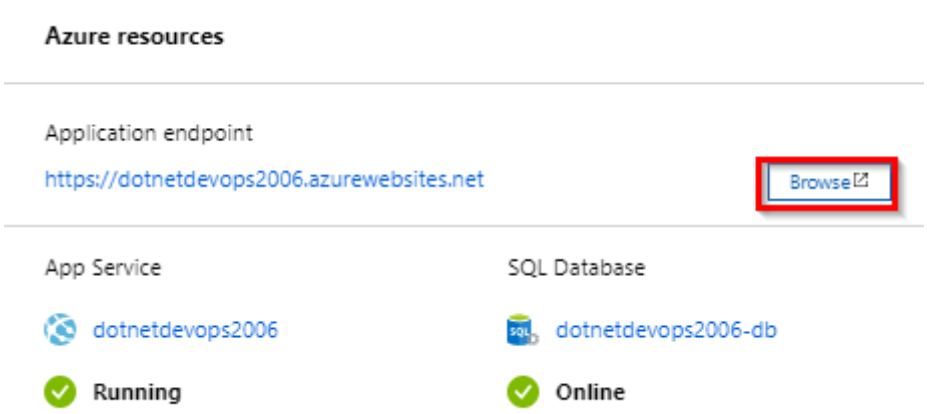


DevOps project

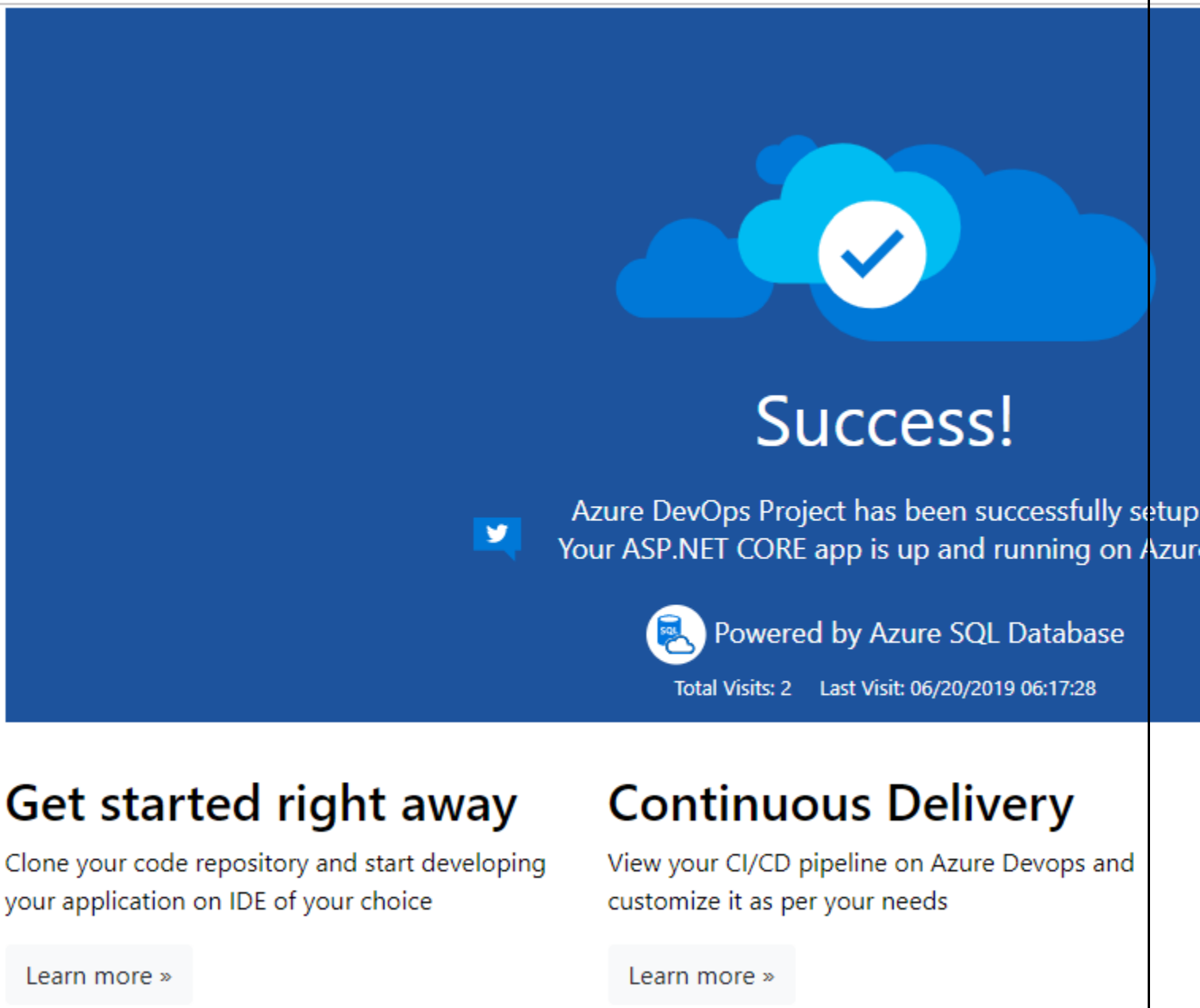
- Created a team project with sample .NET code repository
- Created Azure Web App and Azure SQL database in Azure
- Created a build and release pipelines to compile, test and deploy the application

You're now ready to collaborate with a team on an ASP.NET Core app with a CI/CD process that automatically deploys your latest work to your web site.

On the right side of the dashboard, select **Browse** to view your running application.



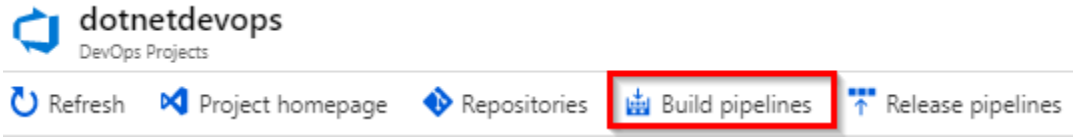
The web app looks like as shown in the below figure



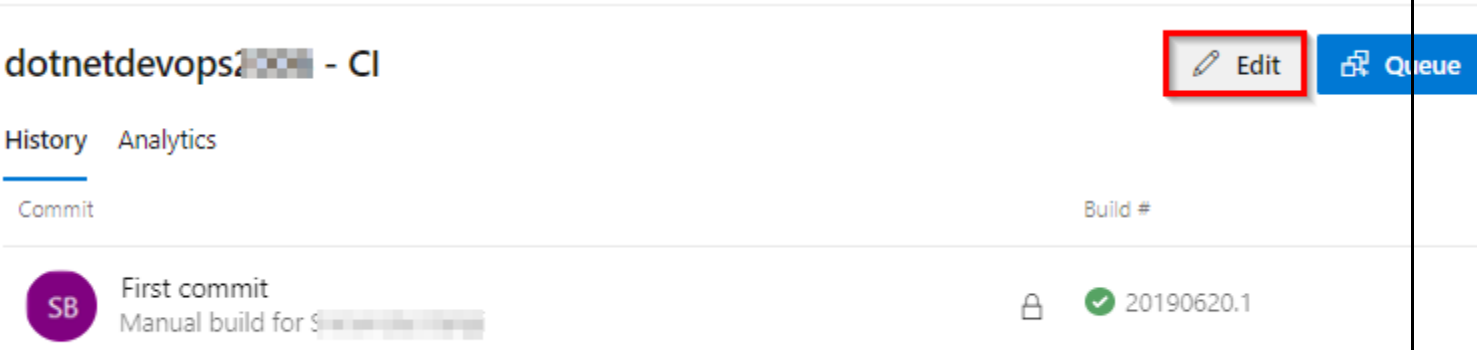
Exercise 2: Examine the CI/CD pipelines configured by Azure DevOps Project

The Azure DevOps project automatically configured a full CI/CD pipeline in your Azure DevOps organization. You can explore and customize the pipeline as needed. Follow the steps below to familiarize yourself with the Azure DevOps build and release pipelines.

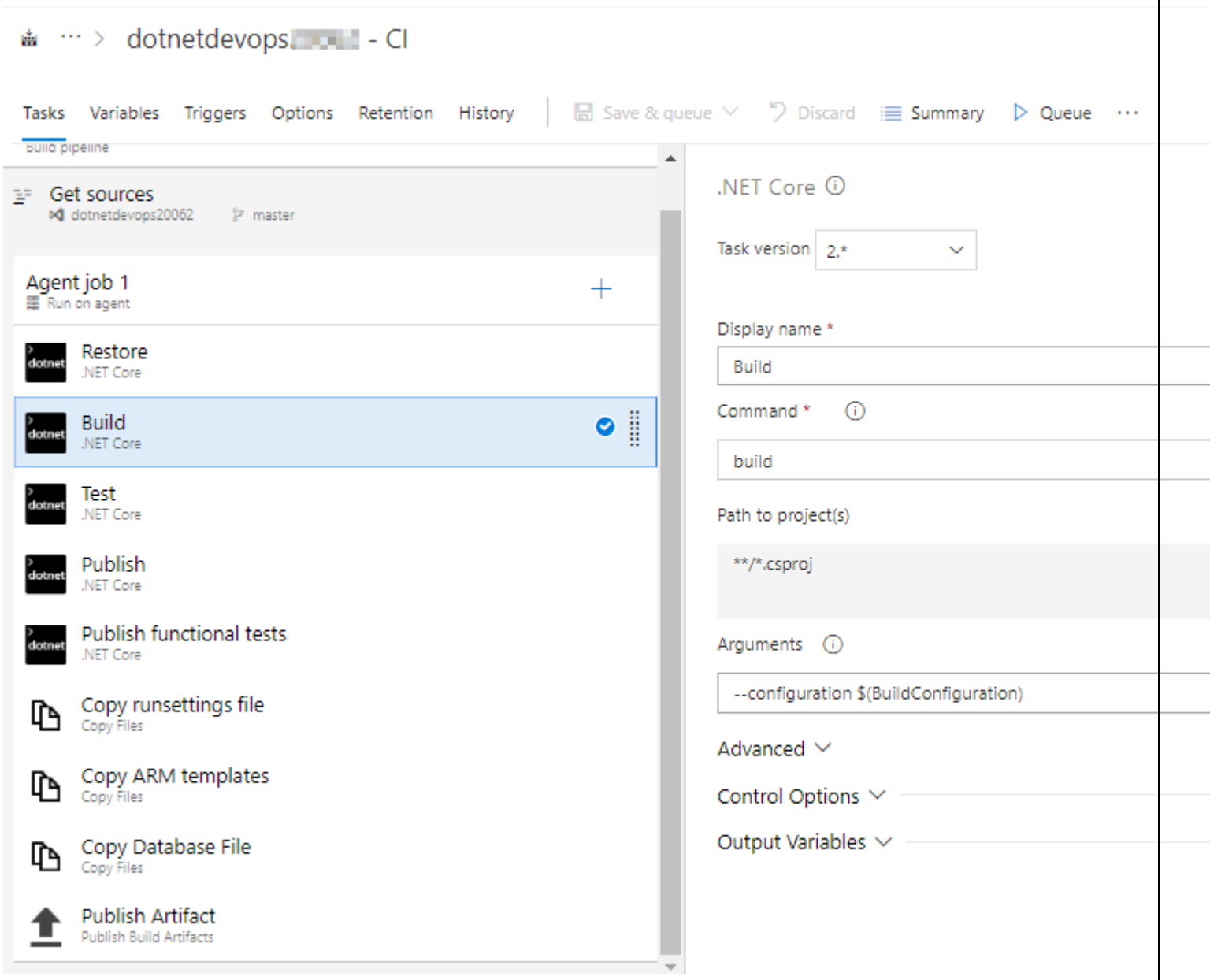
1. Select **Build Pipelines** from the top of the Azure DevOps project dashboard. This link opens a browser tab and the Azure DevOps build pipeline for your new project.



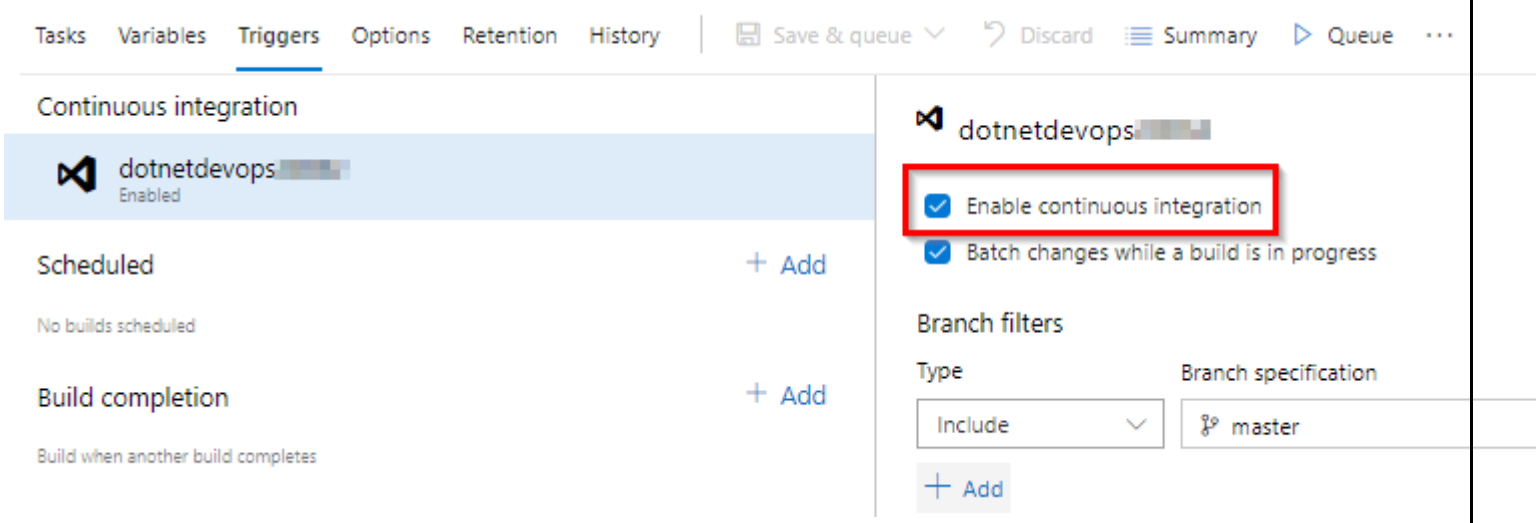
2. Select **Edit**.



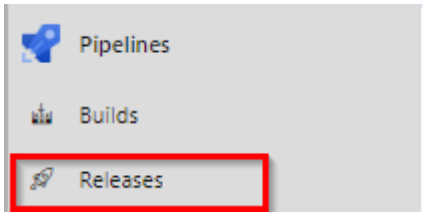
3. In this pane, you can examine the various tasks for your build pipeline. This build pipeline performs various tasks such as fetching sources from the Git repository, restoring dependencies, compile the application, run tests and publishing outputs used for deployments.



4. Under your build pipeline name, select **History**. You see an audit trail of your recent changes for the build. Azure DevOps keeps track of any changes made to the build definition and allows you to compare versions.
5. Select **Triggers**. The Azure DevOps project automatically created a CI trigger, and every commit to the repository initiates a new build. You can optionally choose to include or exclude branches from the CI process.

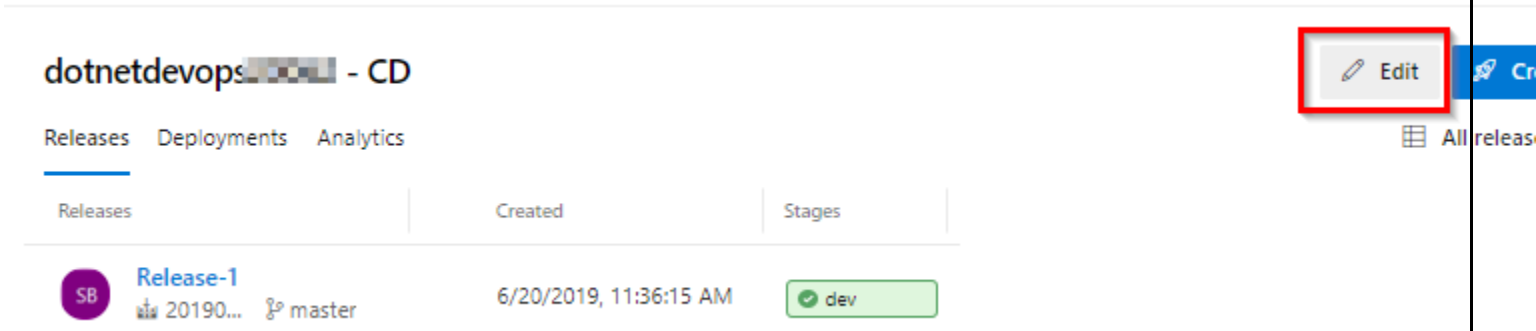


6. Select **Retention**. Depending on your scenario, you can specify policies to keep or remove a certain number of builds.
7. Select **Releases** under **Pipelines** section.

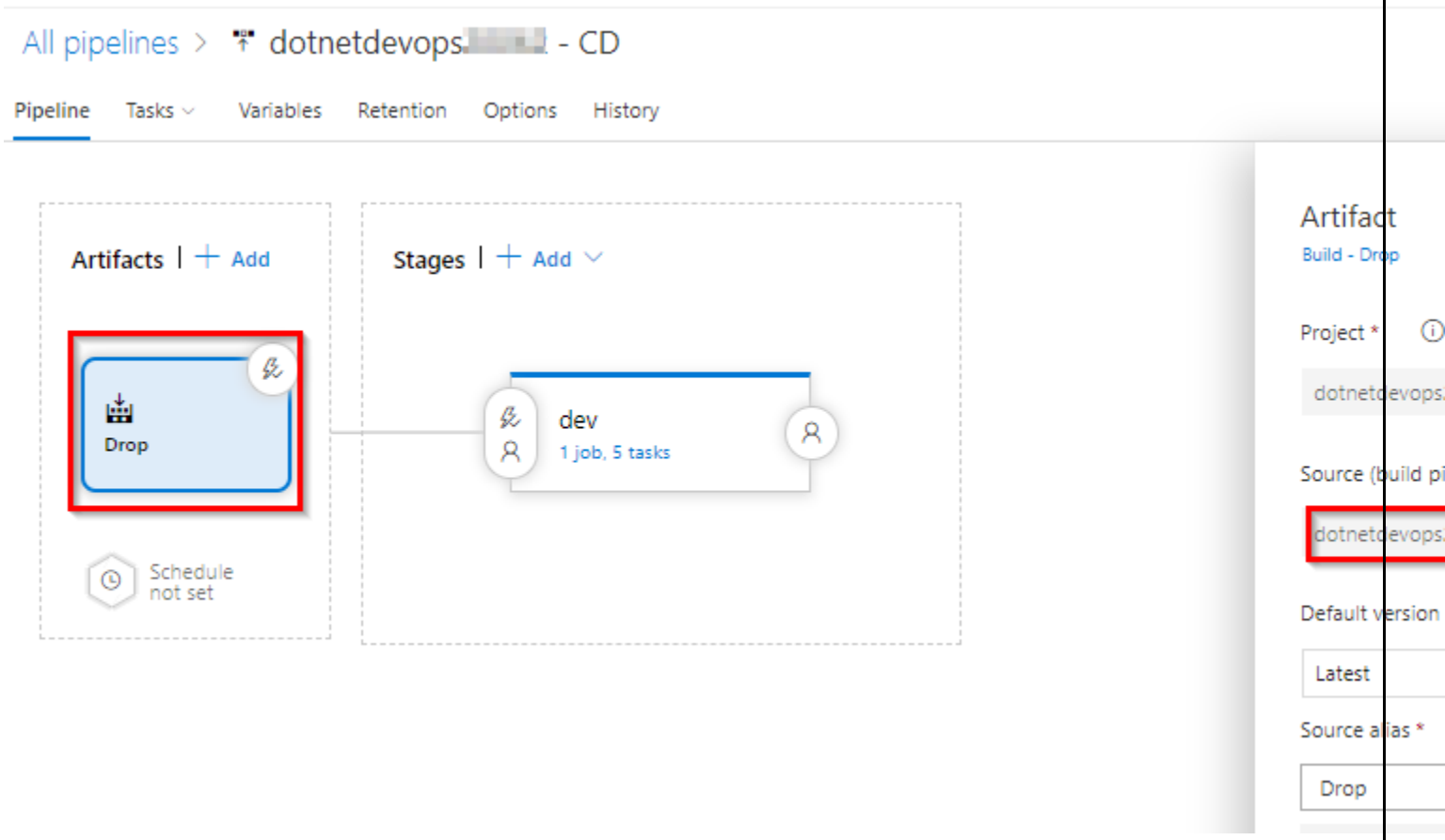


The Azure DevOps project created a release pipeline to manage deployments to Azure.

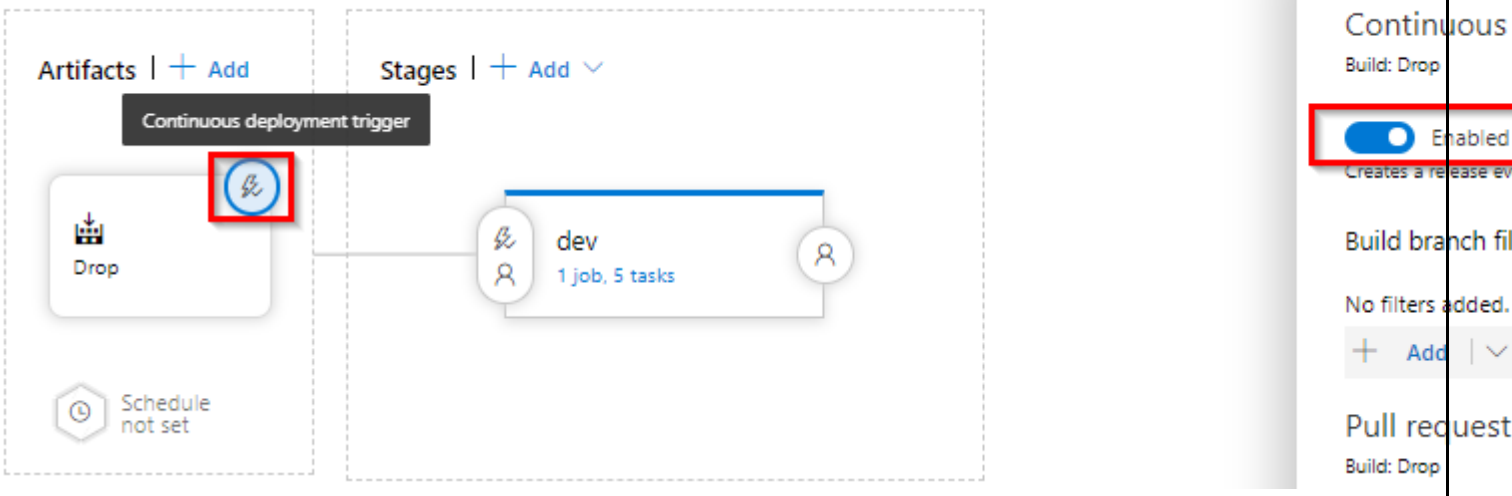
8. Select the release pipeline, then choose **Edit**.



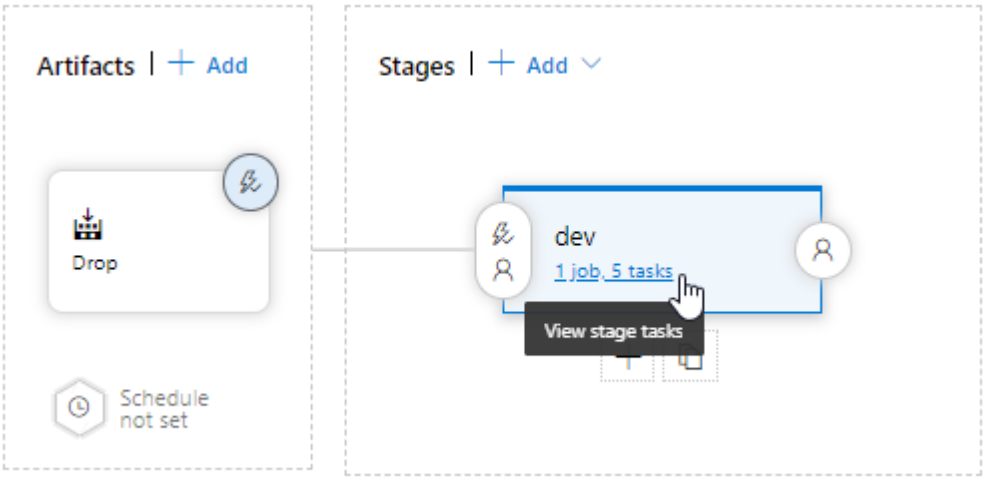
9. Under **Artifacts**, select **Drop**. The build pipeline you examined in the previous steps produces the output used for the artifact.

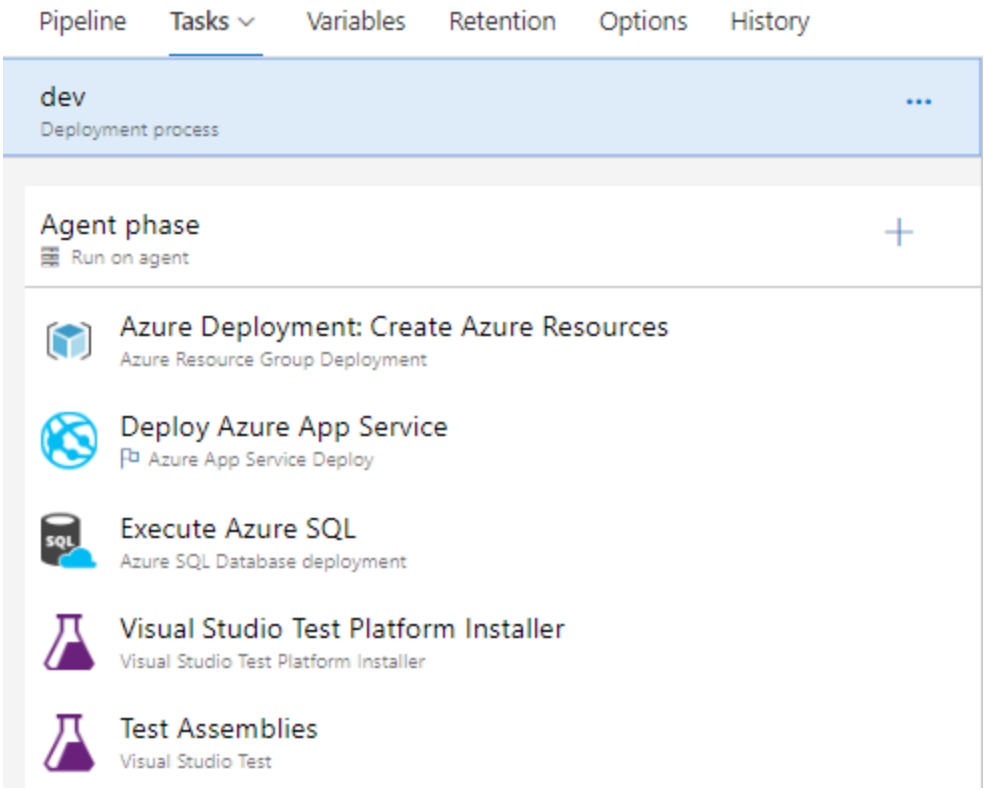


10. To the right-hand side of the **Drop** icon, select the **Continuous deployment trigger**. This release pipeline has an enabled CD trigger, which executes a deployment every time there is a new build artifact available. Optionally, you can disable the trigger, when your deployments require manual execution.



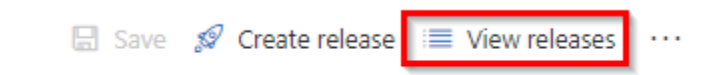
11. Select **Tasks**. The tasks are the activities your deployment process performs. In this example, you have five tasks.

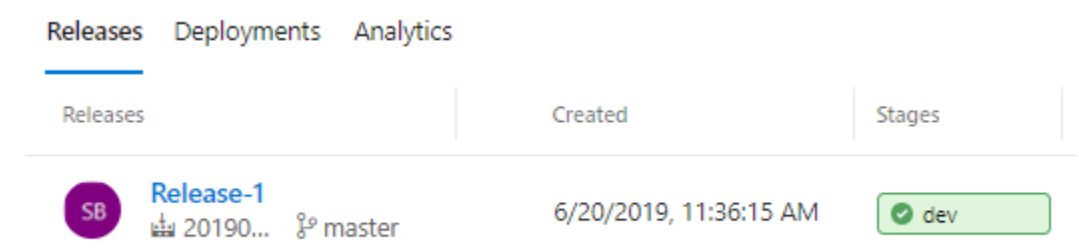




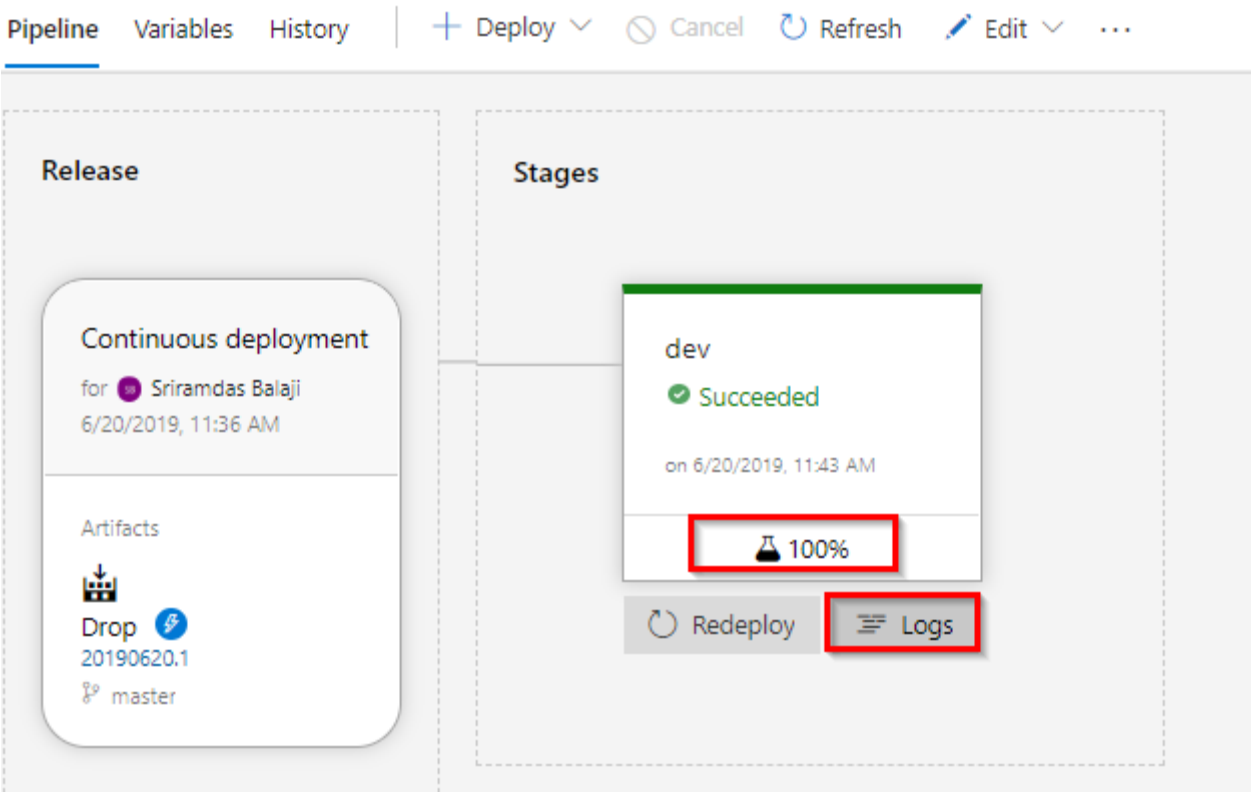
- **Azure Resource Group Deployment** task deploy the required Azure resources, Azure Web app and Azure SQL database for the application to use.
- **Azure App Service Deploy** task deploy the application package to the web site
- **Azure SQL Database deployment** task deploy SQL changes to the database.
- **Visual Studio Test** tasks run functional tests after the successful deployment of the application

On the right-hand side of the browser, select **View releases**. This view shows a history of releases.





Click on the release number to view the release summary. There are several menus to explore from this view such as a release summary, associated work items, and tests.

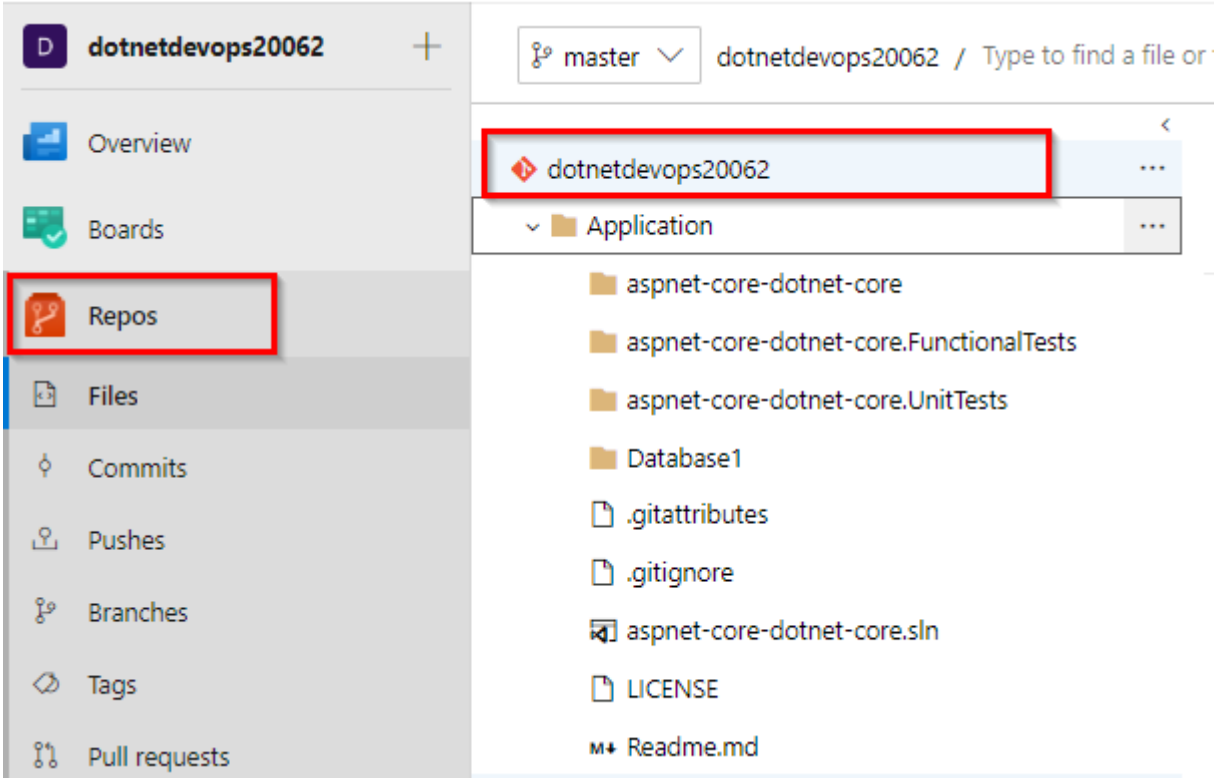


Select **Logs**. The logs contain useful information about the deployment process. They can be viewed both during and after deployments.

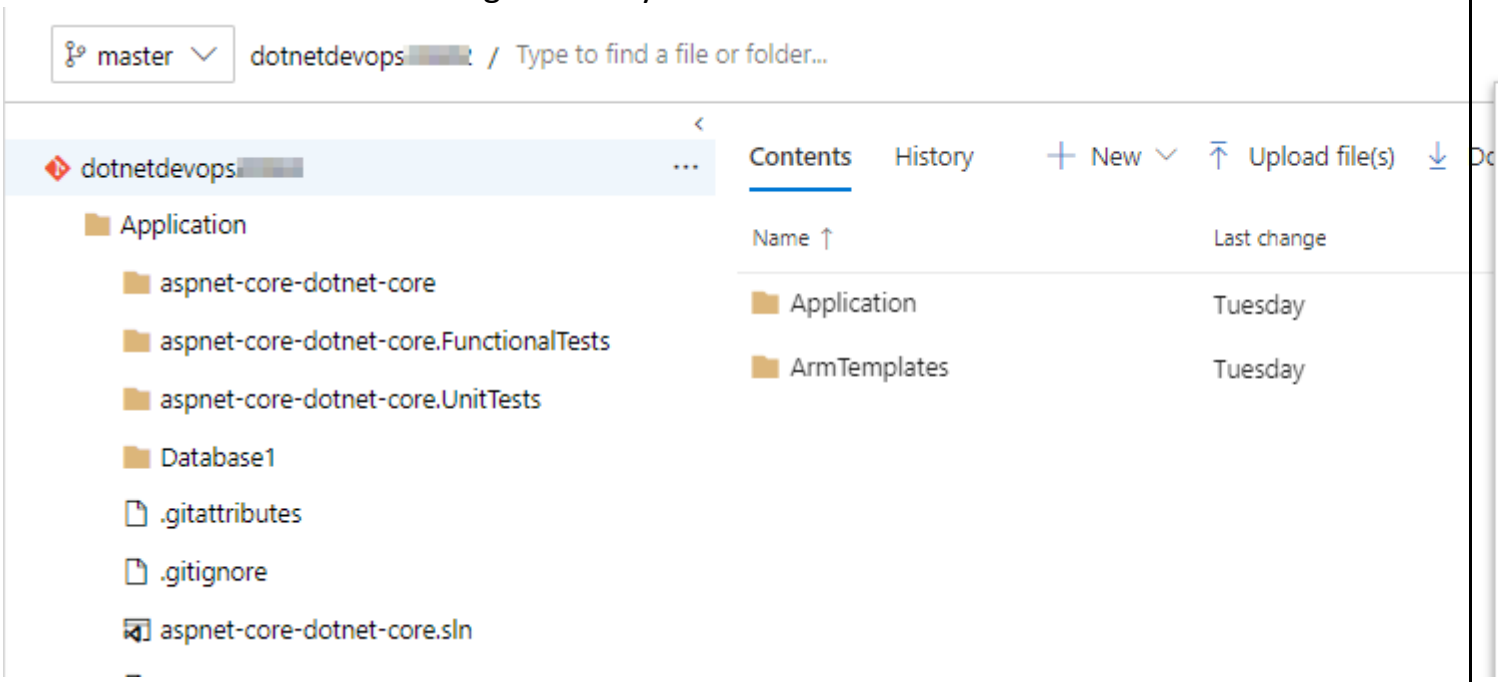
Exercise 3: Commit the code changes and execute CI/CD

The Azure DevOps project created a Git repository in your Azure DevOps organization. Follow the steps below to view the repository and make code changes to your application.

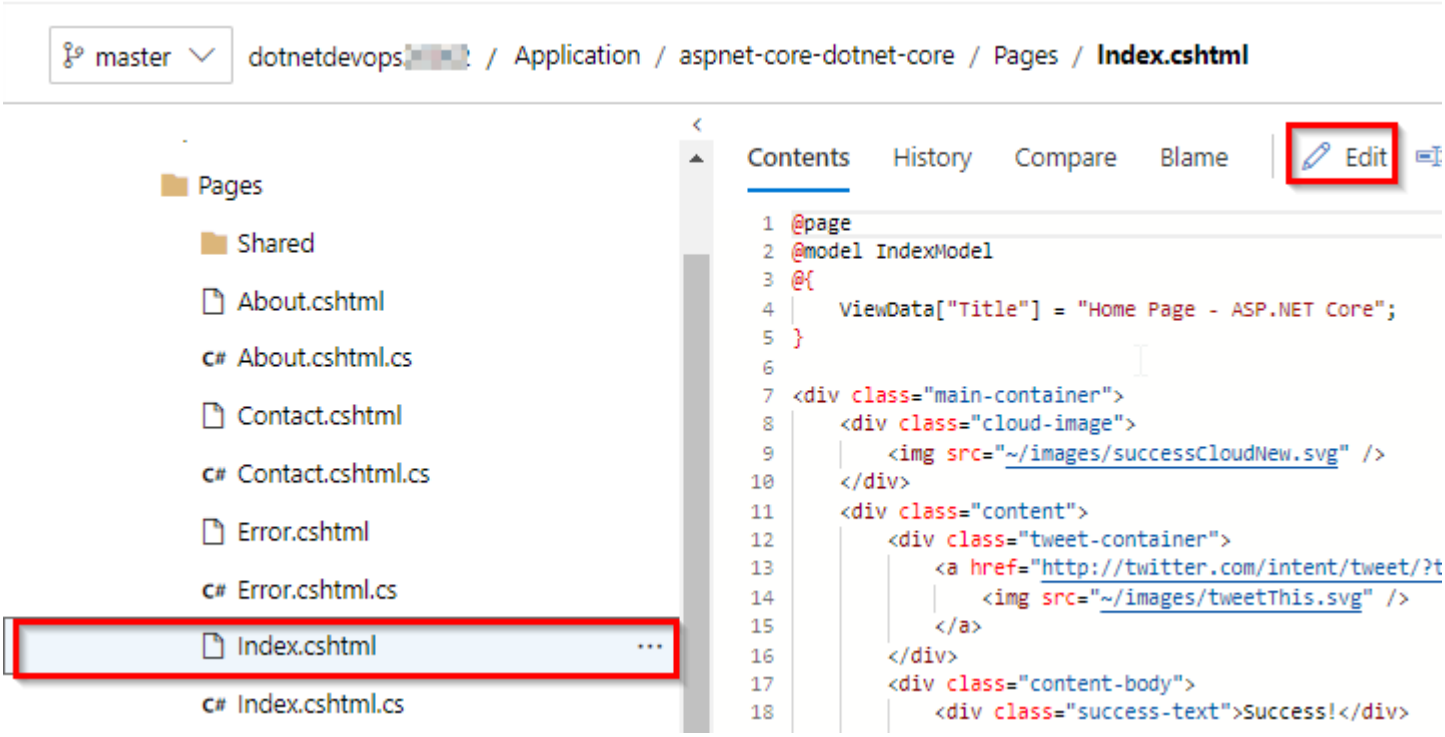
1. Select **Repos** to view the created Git repository by Azure DevOps project.



2. To view the repository clone URL, select **Clone** from the top right of the browser. You can clone your Git repository in your favourite IDE. In this lab, you can use the web browser to make and commit code changes directly to the master branch.

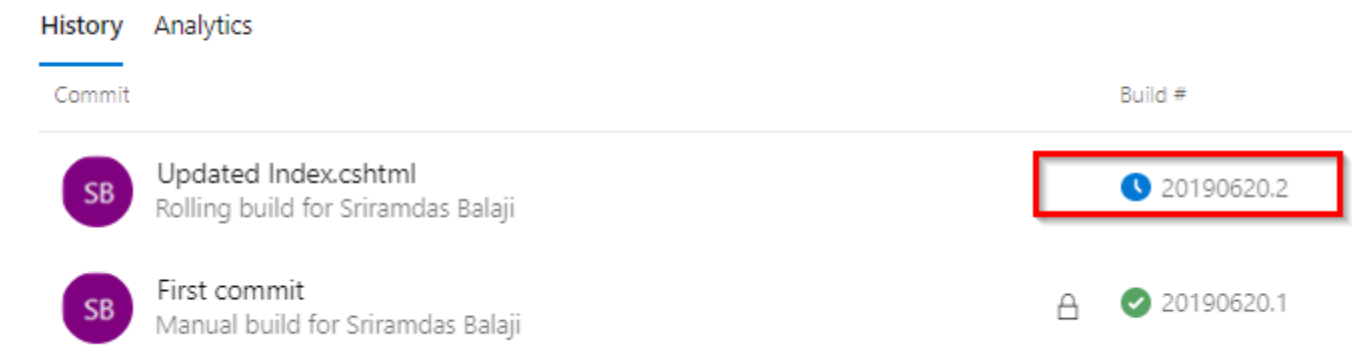


3. On the left-hand side of the browser, navigate to the **Application/aspnet-core-dotnet-core/Pages/Index.cshtml** file. Select **Edit**, and make a change.

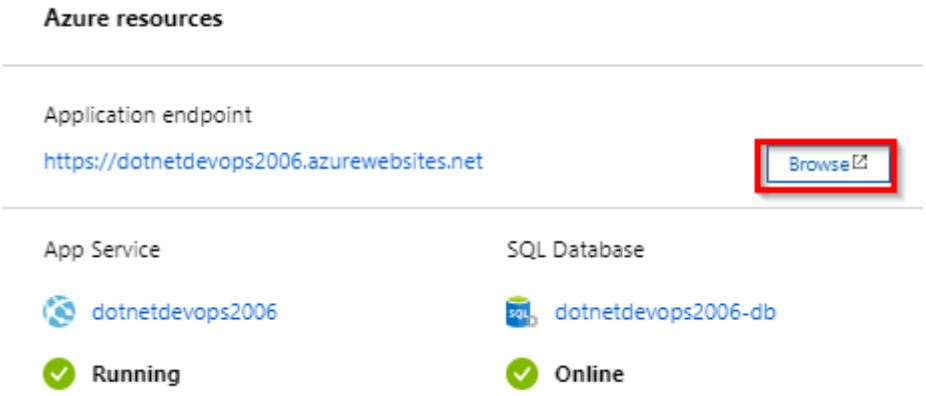


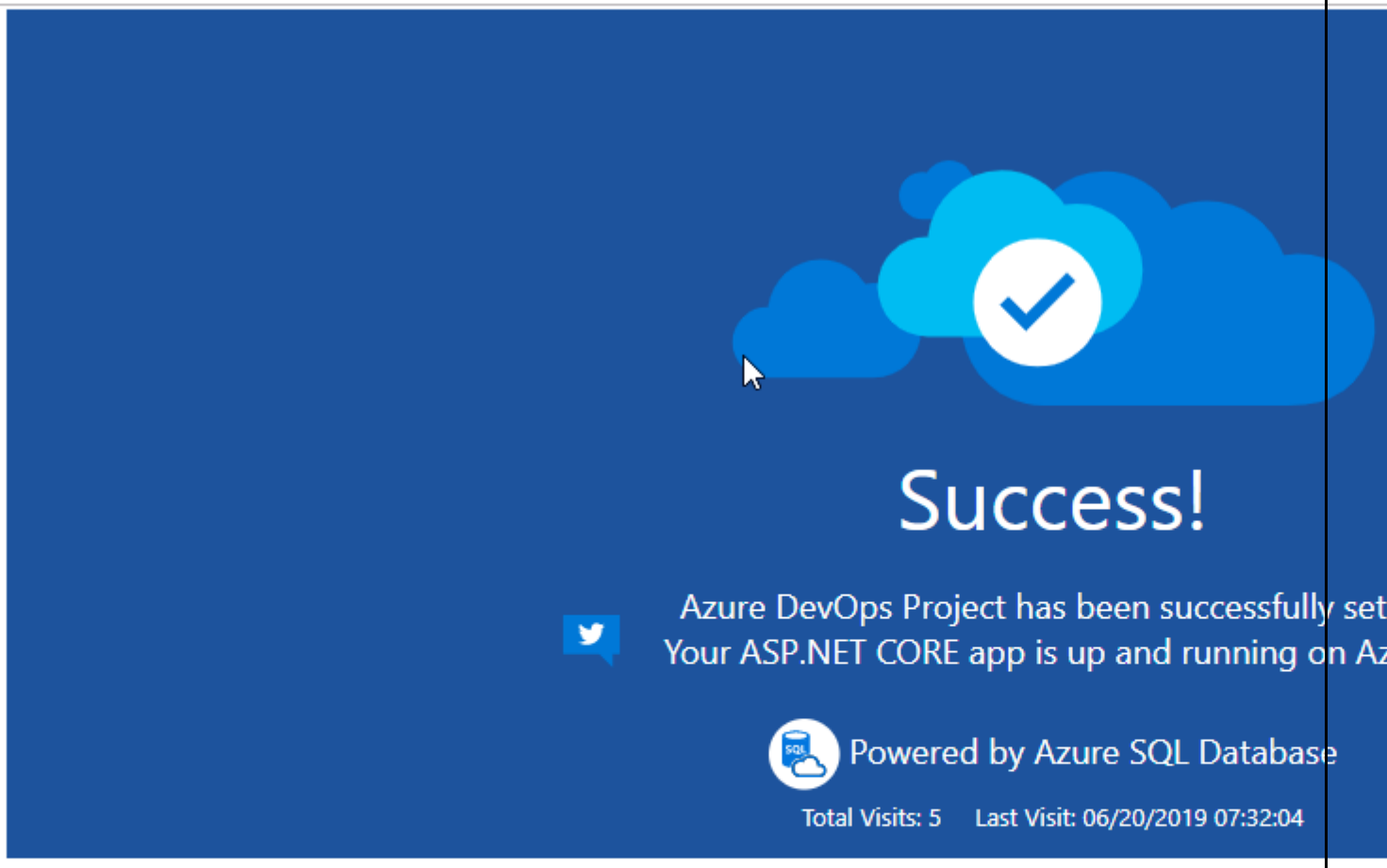
Make a change to the h2 heading. For example, type **Get started right away with the Azure DevOps Projects** or make some other change. Choose **Commit**, to save and check-in your changes.

4. In your browser, navigate to the **Pipelines | Builds**. You should now see a build is in progress. The changes you just made are automatically built and deployed via Azure DevOps CI/CD pipelines.



5. Once the Build and Release are completed in your browser, navigate to the **Azure DevOps project dashboard**. On the right side of the dashboard, select **Browse** to view your updated running application. You will see the updated header in the web app.





Get started right away with the Azure DevOps Projects

Clone your code repository and start developing your application on IDE of your choice

Continuous Delivery

View your CI/CD pipeline on Azure Devops and customize it as per your needs

Learn more »

2nd method

Build Your First CI/CD Pipeline using Azure DevOps

Continuous integration and continuous delivery (CI/CD) are considered by most to be the backbone of DevOps. Things start to get really interesting when you combine these practices with programmable infrastructure and a suite of services that allow you to automate the entire lifecycle of an application.

The goal with this guide is to give you a practical example of what that all looks like when you're building, testing, and deploying applications with Azure DevOps Services. I'll walk you through the end-to-end process of building a fully automated build and release pipeline for a Node and Express application. We'll use Azure DevOps Services to create the CI/CD pipeline and Azure App Service for deploying to development/staging and production.

To follow along, you'll need a GitHub account and Azure Subscription. The demo application is open source, so the Azure DevOps pipeline infrastructure we build will be covered under the free tier.

Create Your Azure DevOps Organization

The first step is to navigate to dev.azure.com and sign in to Azure DevOps. If you've never done this before, you'll need to create a new organization.

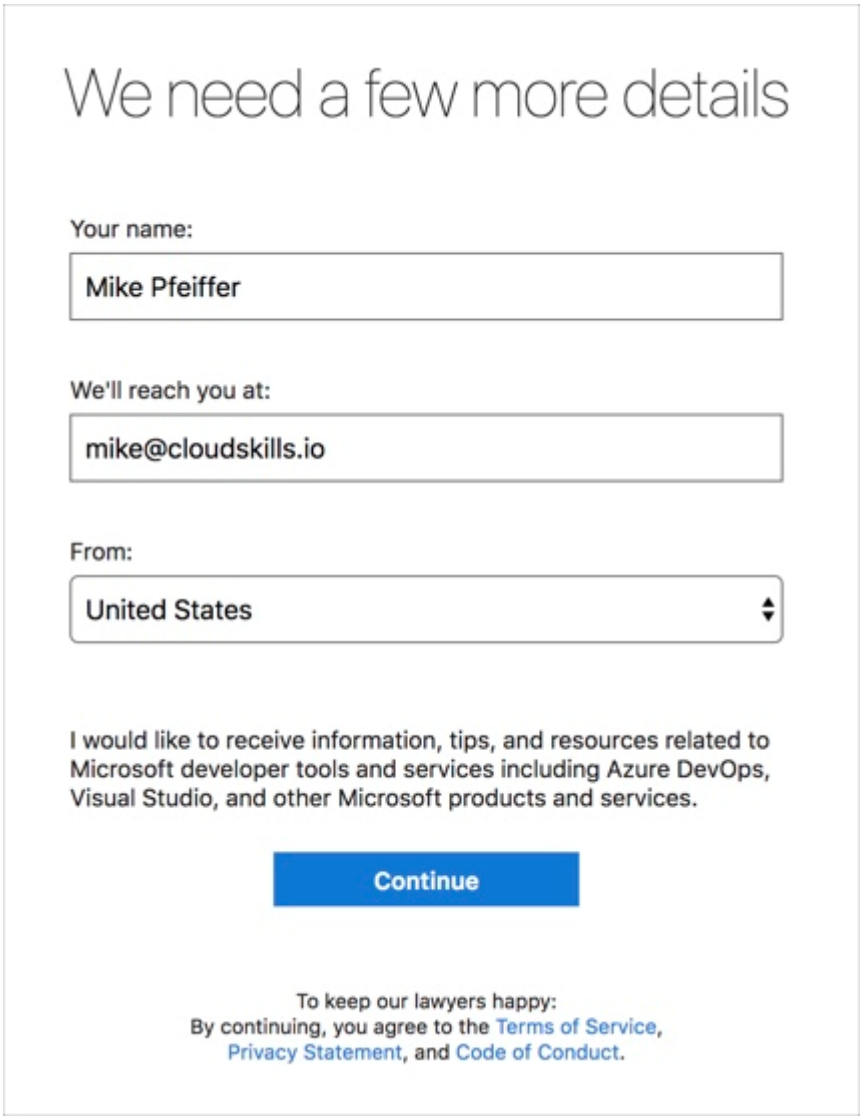


Figure 1. ***Creating an Azure Organization.***

You need to have at least one organization, which is used to store your projects, structure your repositories, set up your teams, and manage access to data. The guidance from Microsoft is to keep things simple and start with a single organization. For more advanced scenarios, take a look at plan your organization structure in Microsoft's documentation.

After clicking on continue, you may end up with an organization name that was generated at random. You can change this as shown in **Figure 2**.

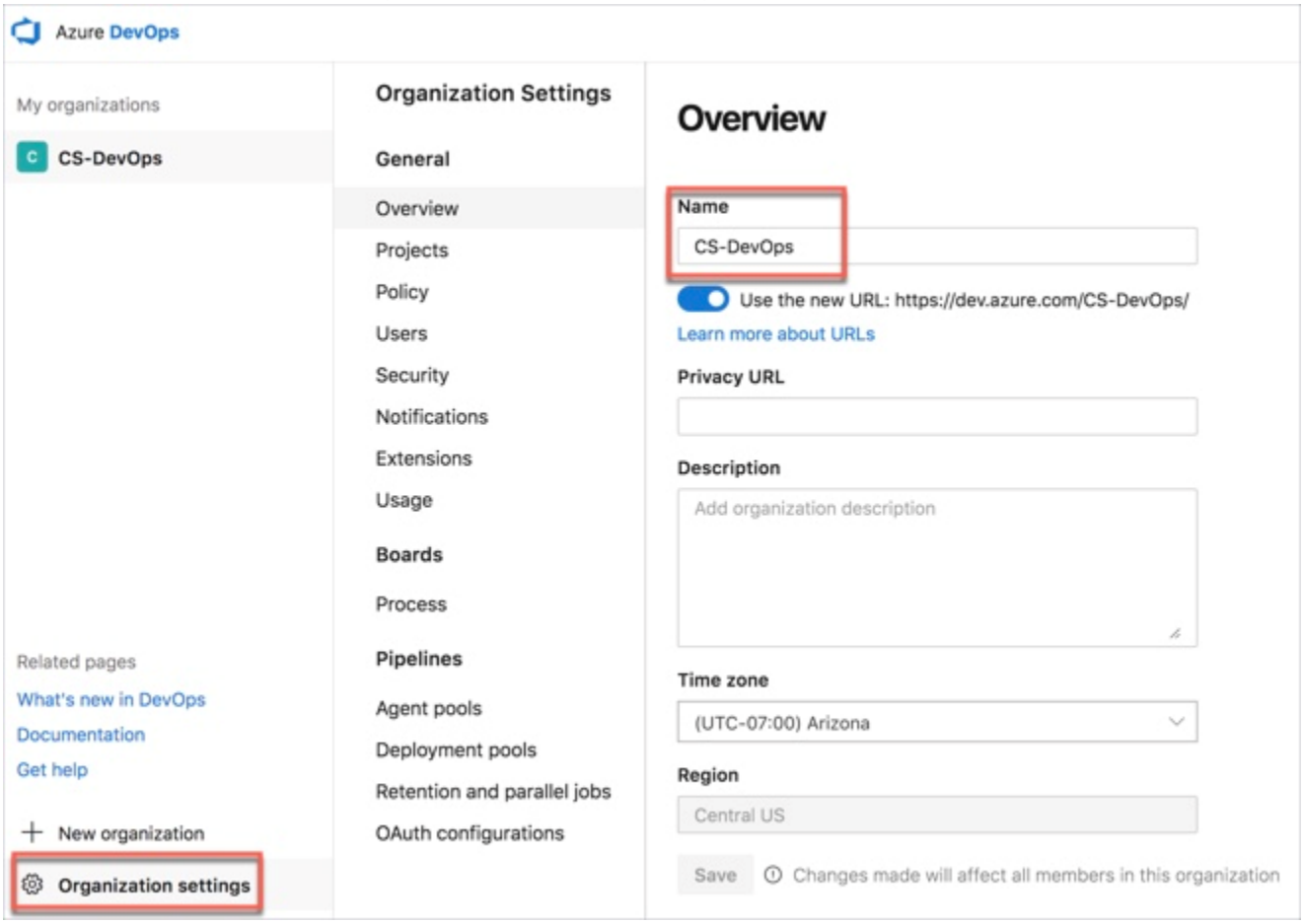


Figure 2. ***Updating Your Azure Organization Name.***

Simply navigate to **Organization Settings** > **Overview** and update the name.

Fork the Node & Express Demo App Repository

I wanted to demonstrate an application that was somewhat realistic but not overly complex for this walkthrough. The Node and Express app is a simple website for a fictitious company. This app uses Express and Handlebars to serve up a few common pages you'd see on any company website. Also included are some unit tests that ensure those routes are working and serving up the right content.

You can head over to my GitHub account to fork this repository.

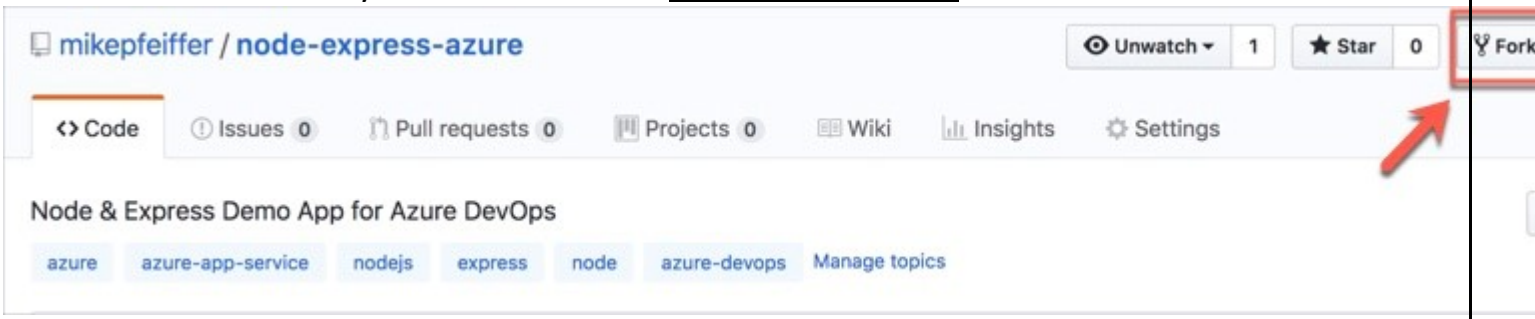


Figure 3. ***Fork the*** *node-express-azure* ***Repository.***

Next, we can move on to deploying the infrastructure to support both development and production deployment slots using Azure App Service.

Deploy the App Service Infrastructure

We're going to use an Azure Web App for Linux resource to power our Node and Express application. We'll set things up so our CI/CD pipeline can build and deploy the app into a development/staging slot. Then we'll set up up a manual approval into the production slot.

We'll use an Azure Resource Manager (ARM) template to build the App Service infrastructure.

Navigate to the **node-express-azure** repository you forked in the previous step. You'll see a "Deploy to Azure" button about halfway down the screen.

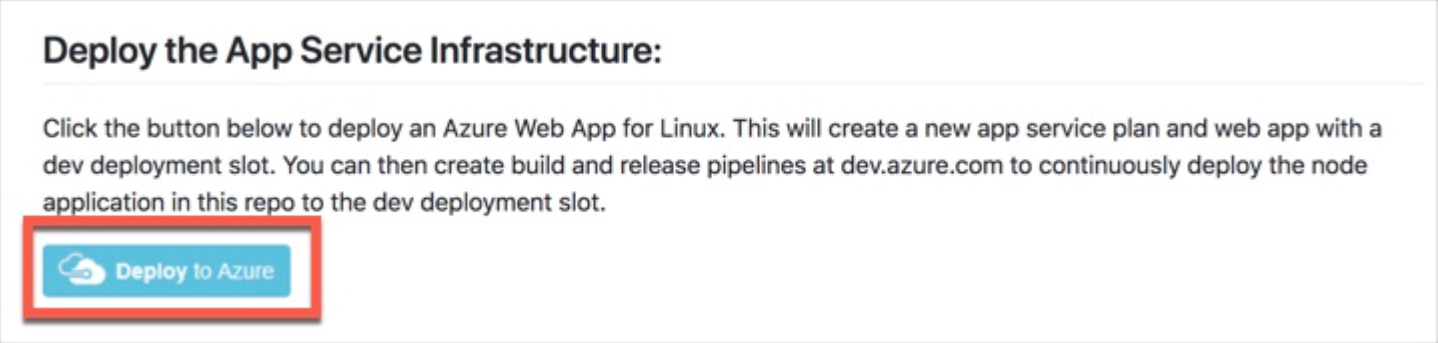


Figure 4. ***Deploying the ARM Template.***

Clicking the "Deploy to Azure" button will redirect you to the Azure portal as shown in **Figure 5**.

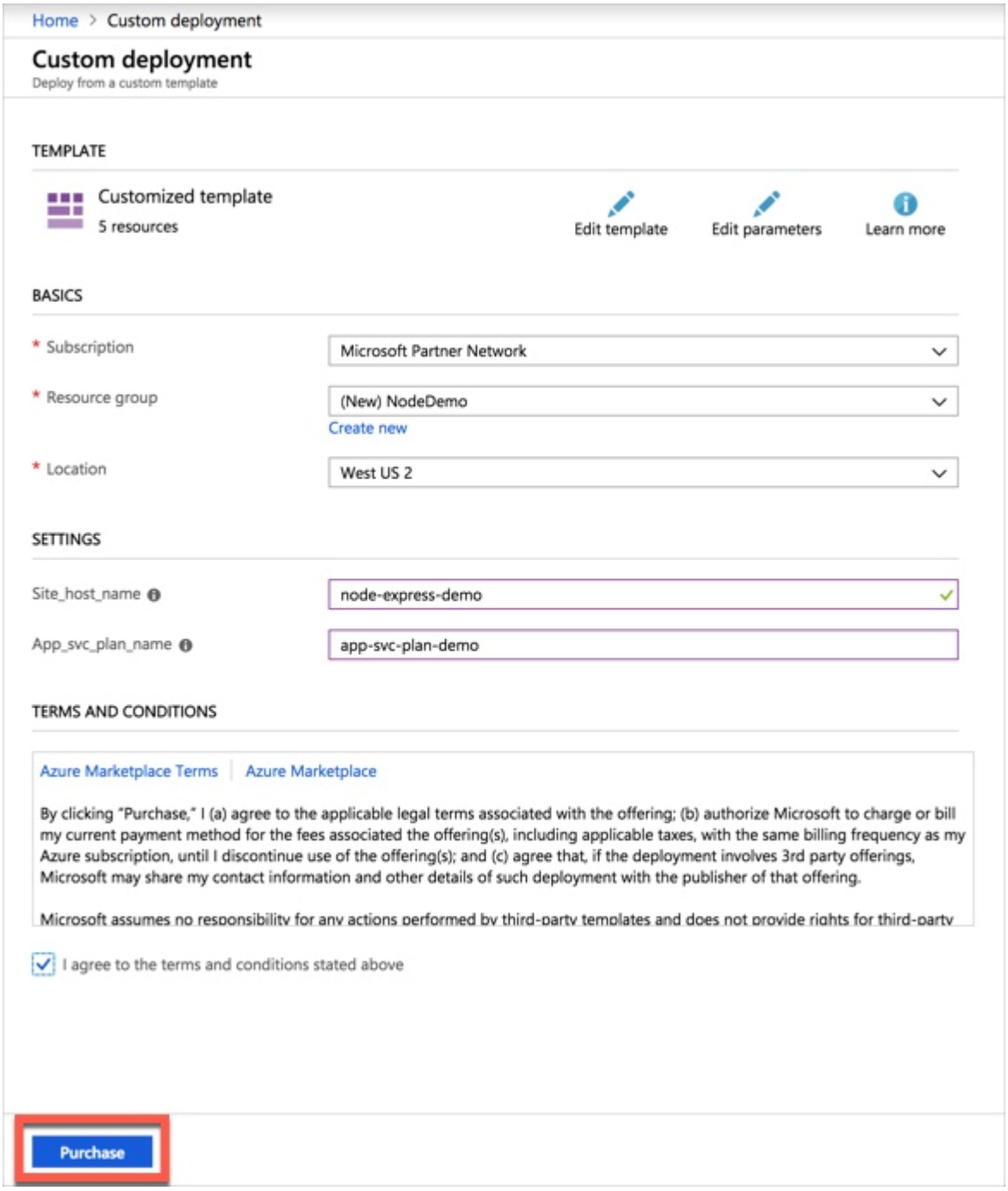


Figure 5. ***Launching the ARM Template.***

Notice that you'll need to set a globally unique hostname for your web application, along with a name for the new app service plan. I'd recommend deploying these resources into a new resource group. That way when you're done with this walkthrough, you can clean up the Azure resources easily by deleting the resource group.

Click "Purchase" to launch the template to agree that you'll have to pay for the App Service resources that this template deploys on your behalf.

After you launch the template you should see a successful deployment message, and you should have a new resource group similar to the one shown in **Figure 6**. Notice that there is an App Service Plan, a web app that represents the production deployment slot, and a slot for development called "dev".

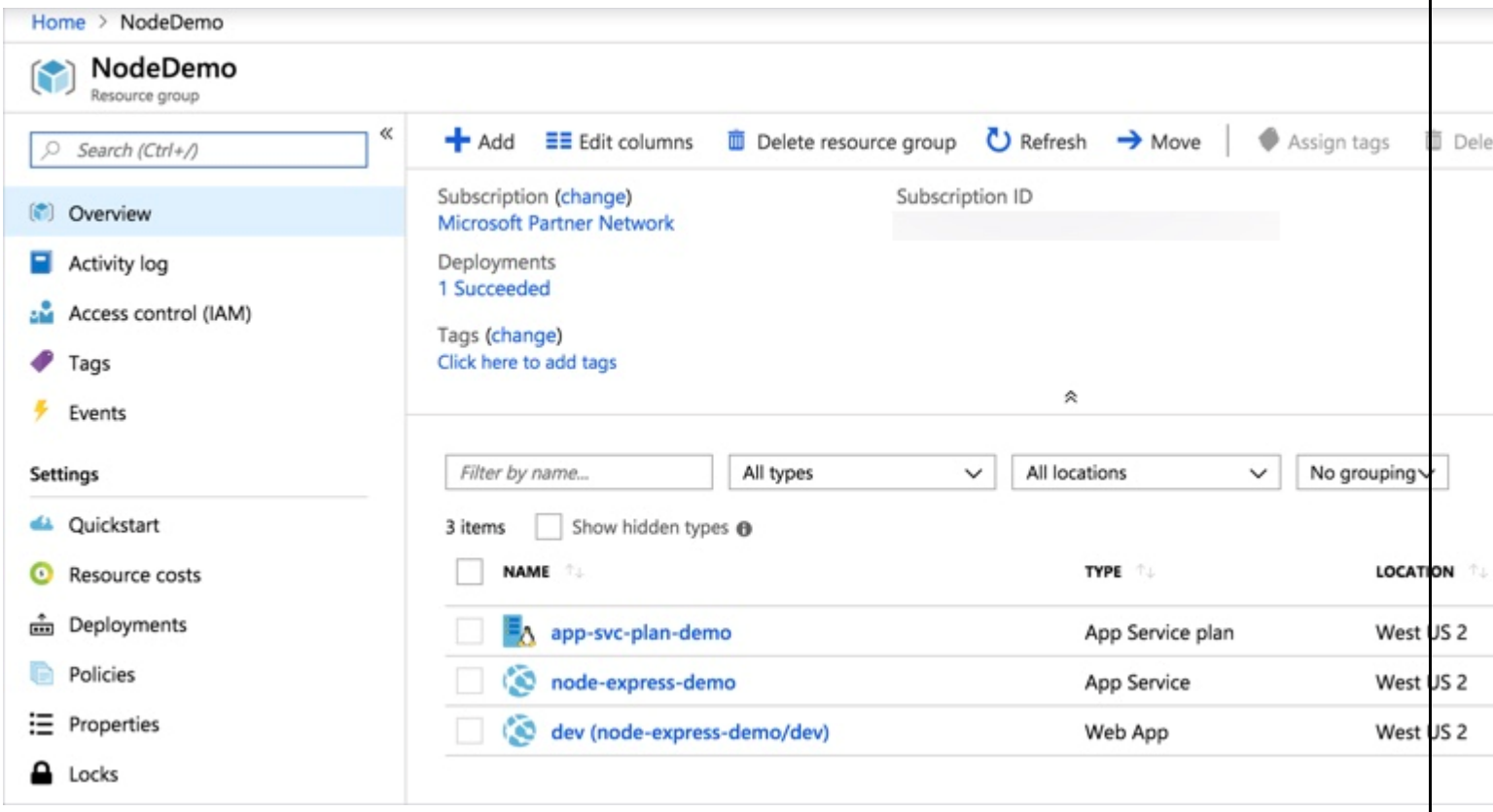


Figure 6. ***Reviewing the Resources.***

Quick side note about the ARM template: the *Deploy to Azure* button references the **azuredeploy.json** ARM template in *my GitHub repository*. If you want to update the template, update the version in your own repo, and don't forget to change the target of the button in the source of your **README.md** file.

Create a Build Pipeline

We're ready to move on and set up a build pipeline in Azure DevOps. Head back to dev.azure.com and create a new project inside your organization. Use the settings shown in **Figure 7**.

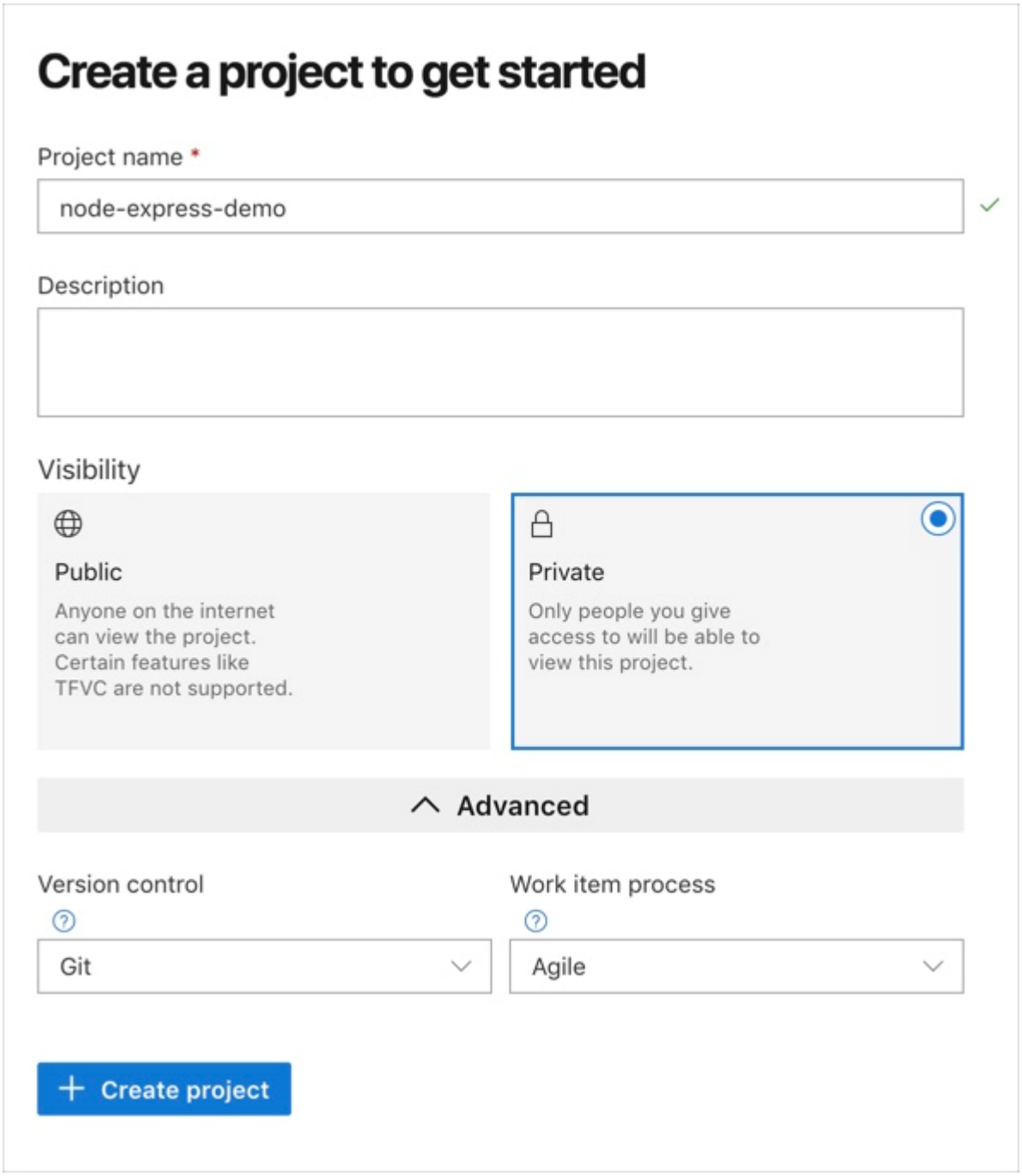


Figure 7. ***The New Project Settings.***

After clicking on the "Create project" button, you'll see a summary page for the project. Navigate to Pipelines and click on Builds as shown in **Figure 8**.

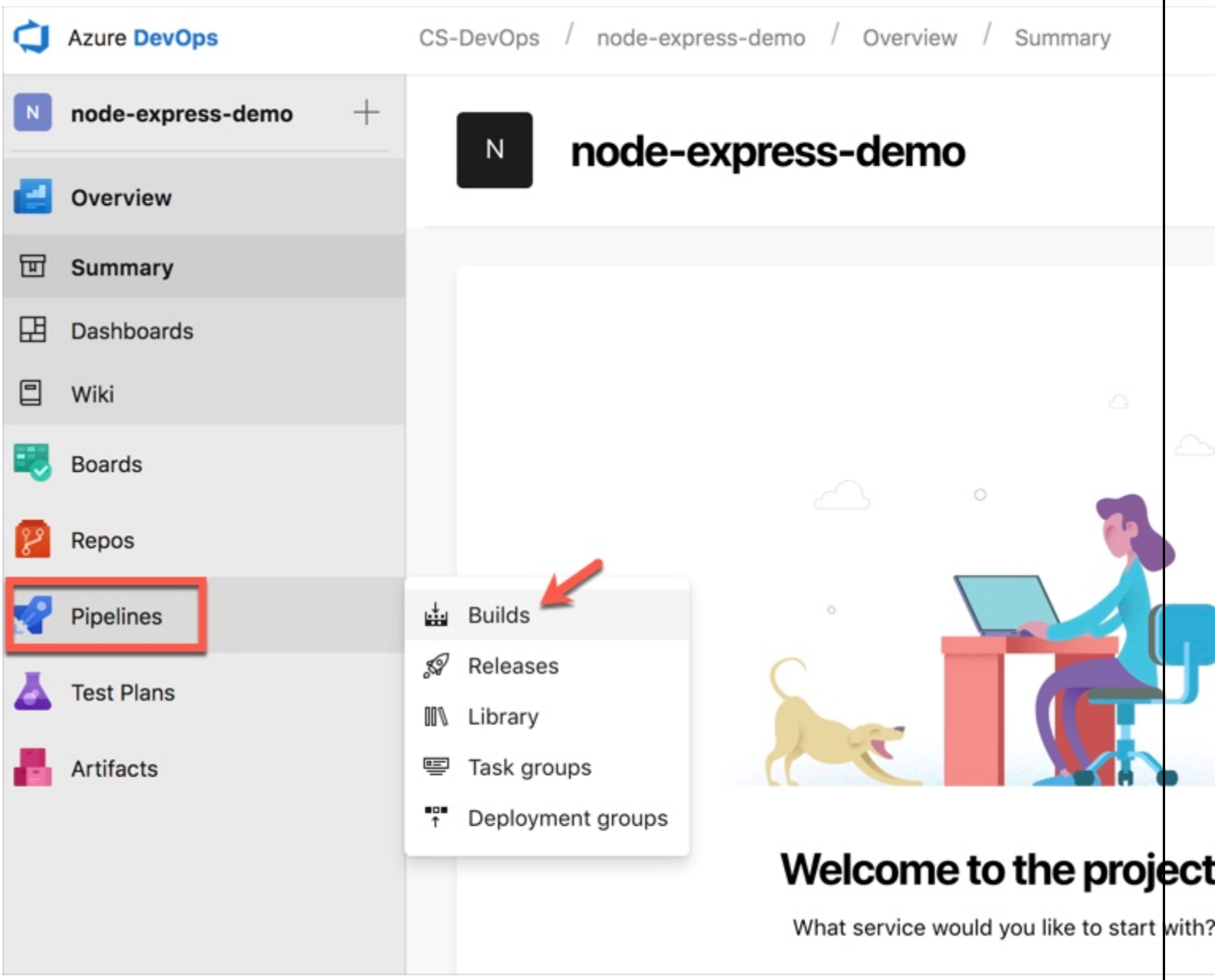


Figure 8. ***Creating a New Build Pipeline.***

Next, click the button to create a new build pipeline. You'll be prompted to choose a repository. Select **GitHub**. You'll see a screen like the one in **Figure 9** where you'll need to authorize the Azure DevOps service to connect to your GitHub account on your behalf. Click **Authorize**.

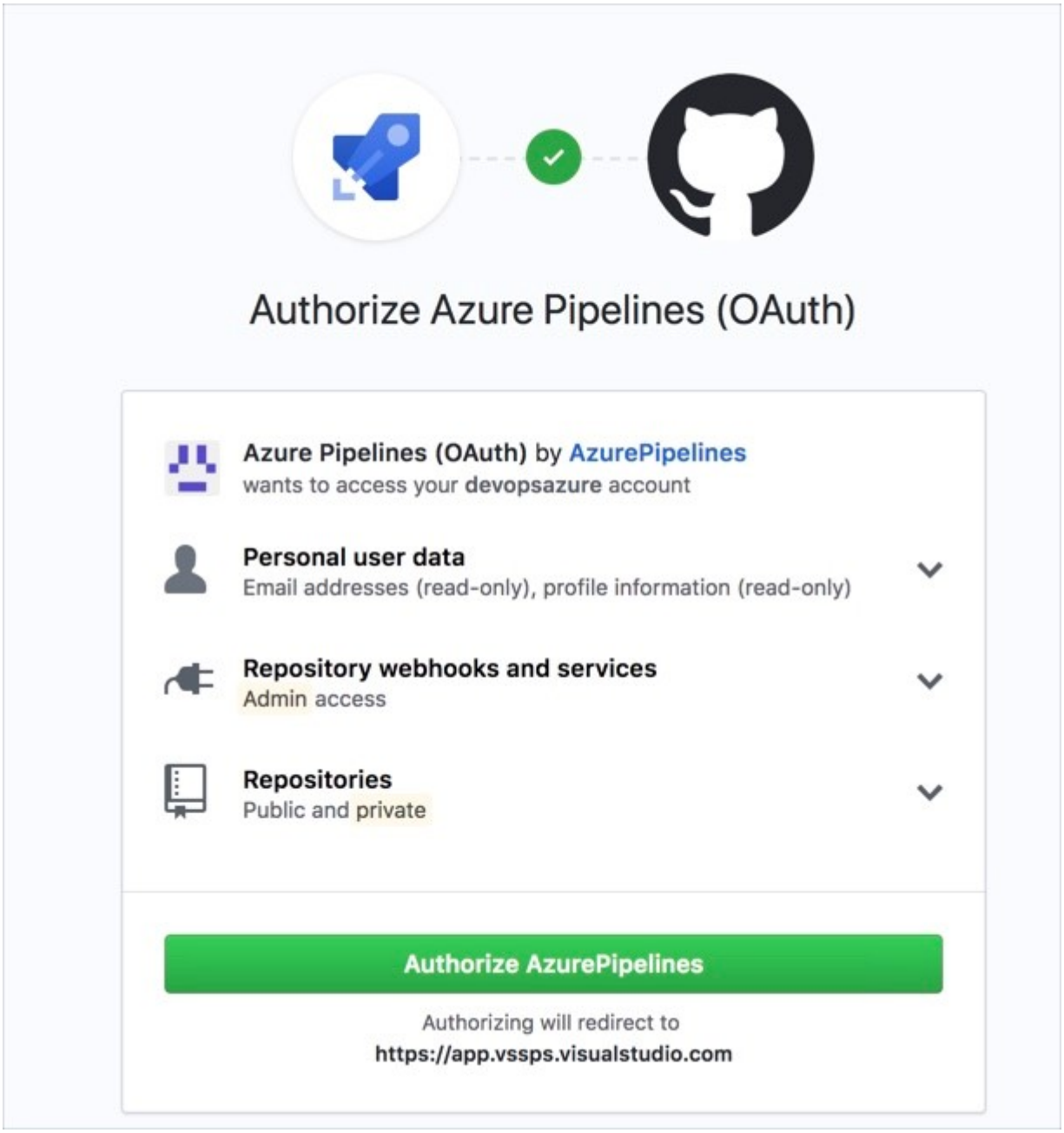


Figure 9. ***Authorizing the GitHub Connection.***

After your connection to GitHub has been authorized select the **node-express-azure** repo that you forked in the first step. You should end up seeing a “New pipeline” screen like the one shown in **Figure 10**.

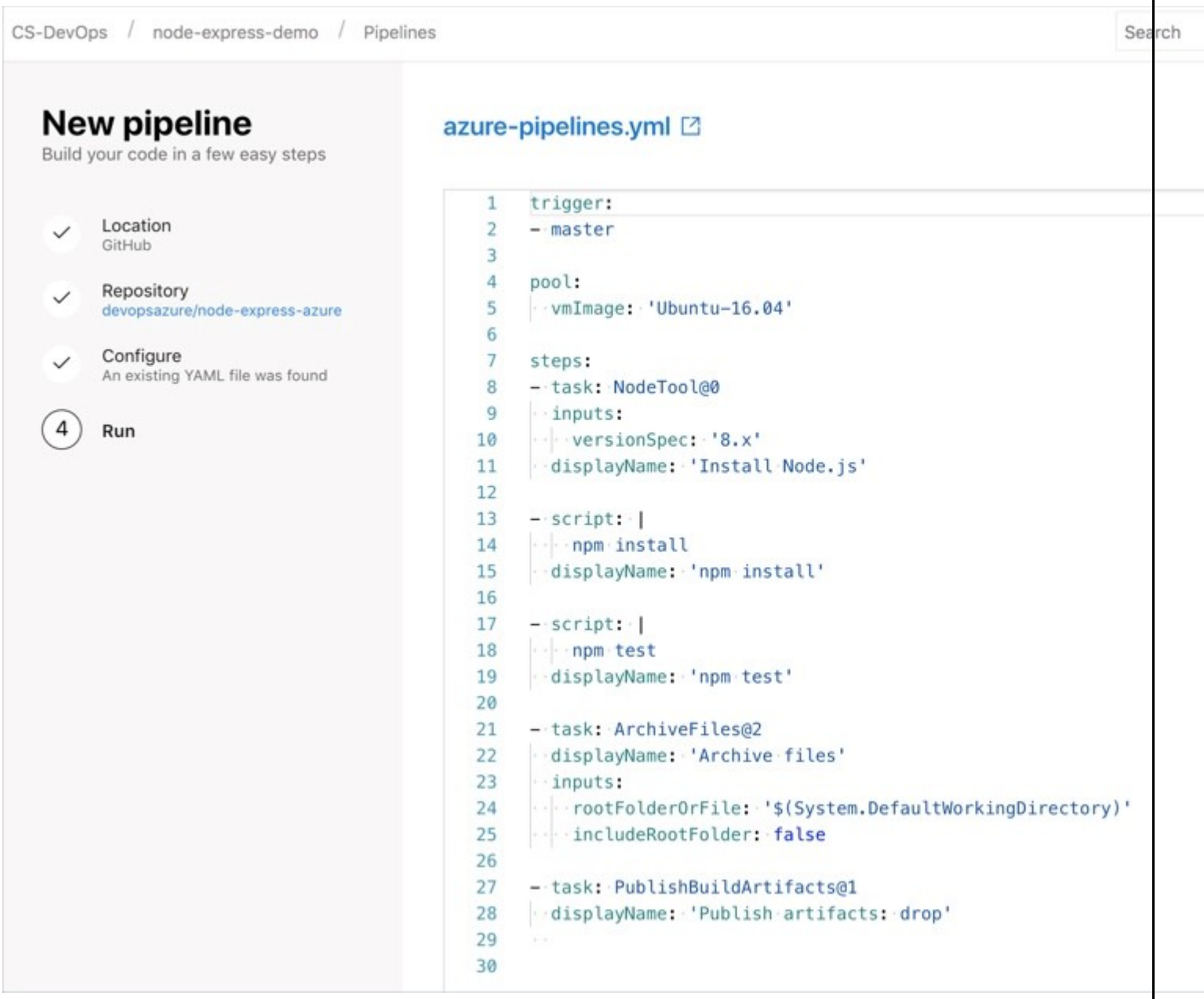


Figure 10. ***Creating the New Build Pipeline.***

The new pipeline wizard should recognize that we already have an **azure-pipelines.yml** in the repository. This file contains all of the settings that the build service should use to build and test our application, as well as generate the output artifacts that will be used to deploy the app later in our release pipeline.

After you click “Run” to kick off your first build, you should see a screen like the one shown in **Figure 11**.

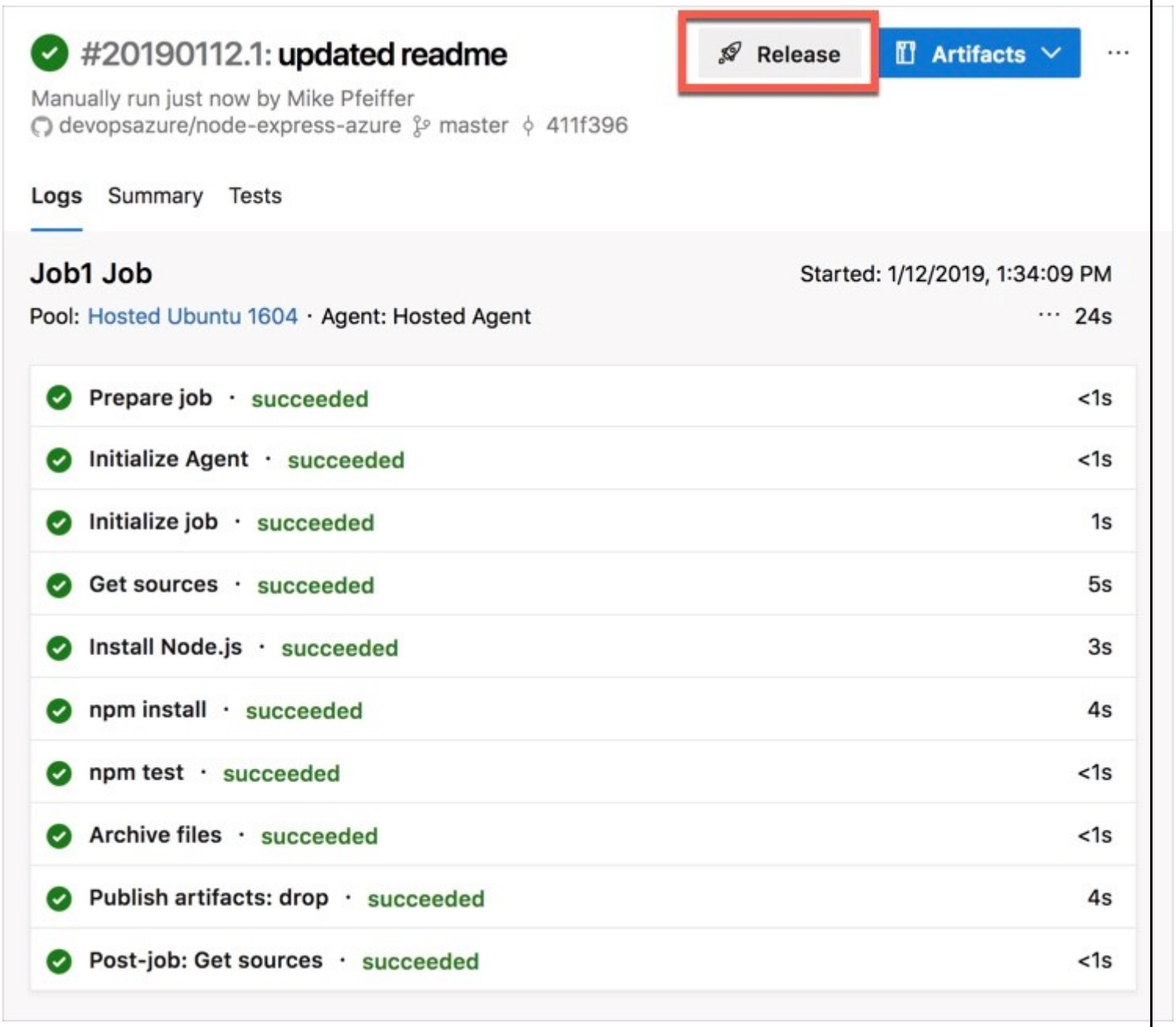


Figure 11. ***Reviewing the Build Status.***

Notice that a lot went on with the build. The service used an Ubuntu 16.04 build agent to grab the code from GitHub, installed our development dependencies, and then ran our unit tests to validate the application. Finally, the code was bundled into an output artifact and published so we can use it as an input artifact for our upcoming release pipeline.

Click on the release button at the top of this screen to create a new release pipeline.

Create a Release Pipeline

When you get into the release pipeline screen, you'll need to select a template. For this scenario, we are going to choose "App Service deployment with slot".

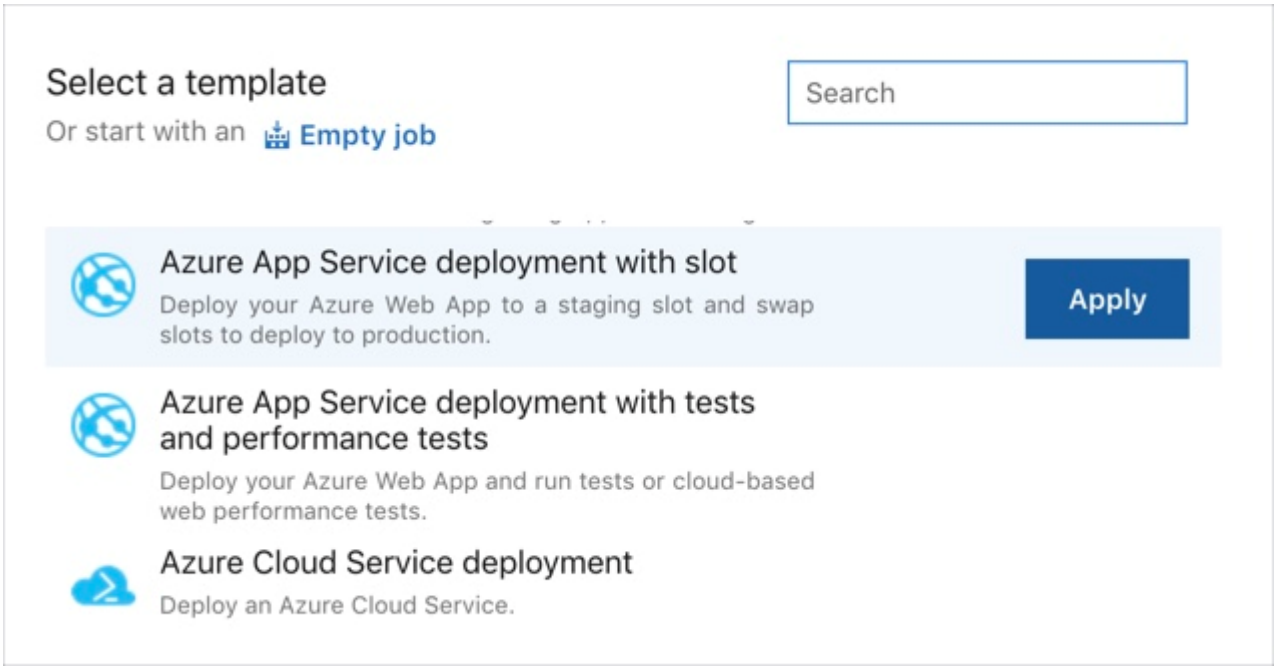


Figure 12. ***Selecting the Deployment Template.***

Click on the apply button to create the new deployment stage within the release pipeline. On the next screen, you'll be able to configure this stage. Change the name to "development" as shown in **Figure 13**.

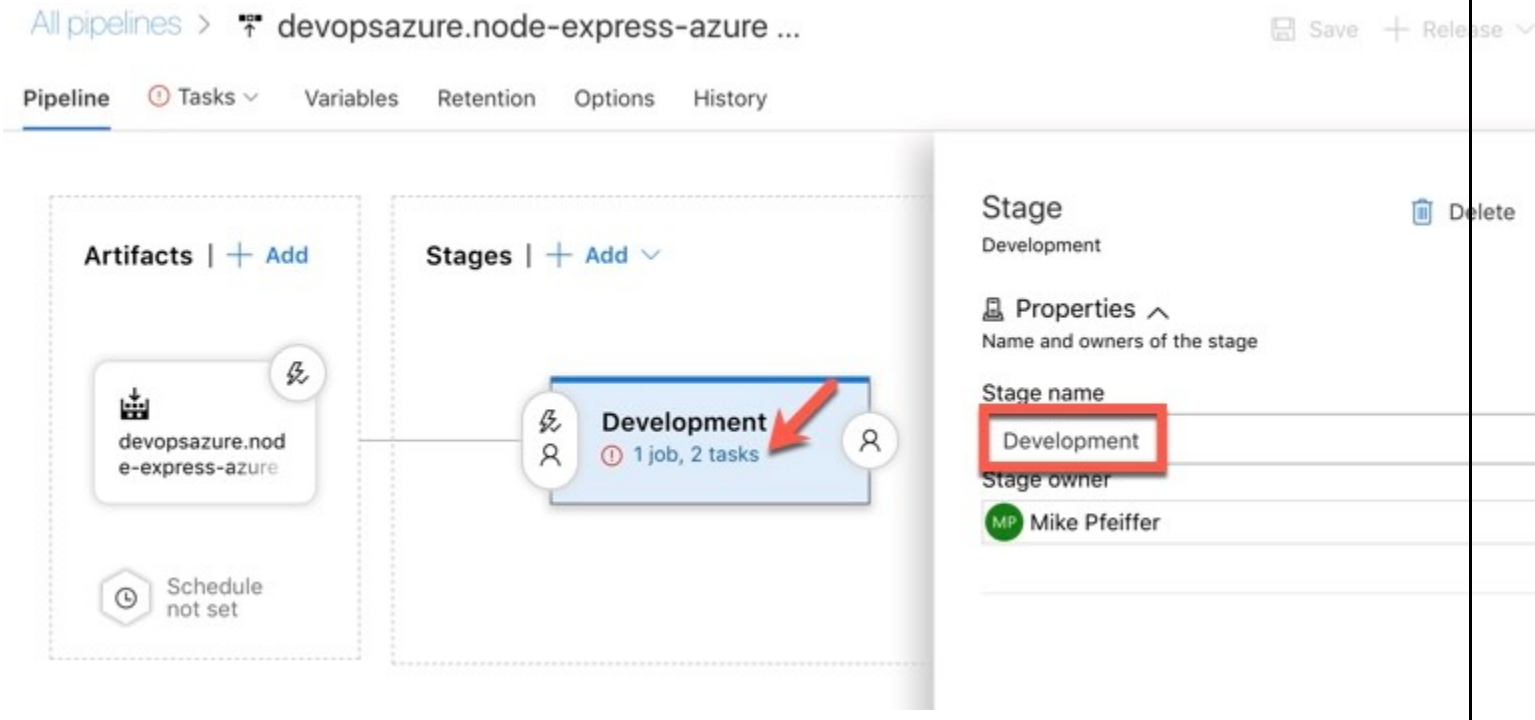


Figure 13. ***Updating the Development Stage.***

While on this screen, click on the link that says "2 tasks" inside your development stage. This will take you to a screen where you can configure the deployment task. Make sure you fill out all the fields as shown in **Figure 14**.

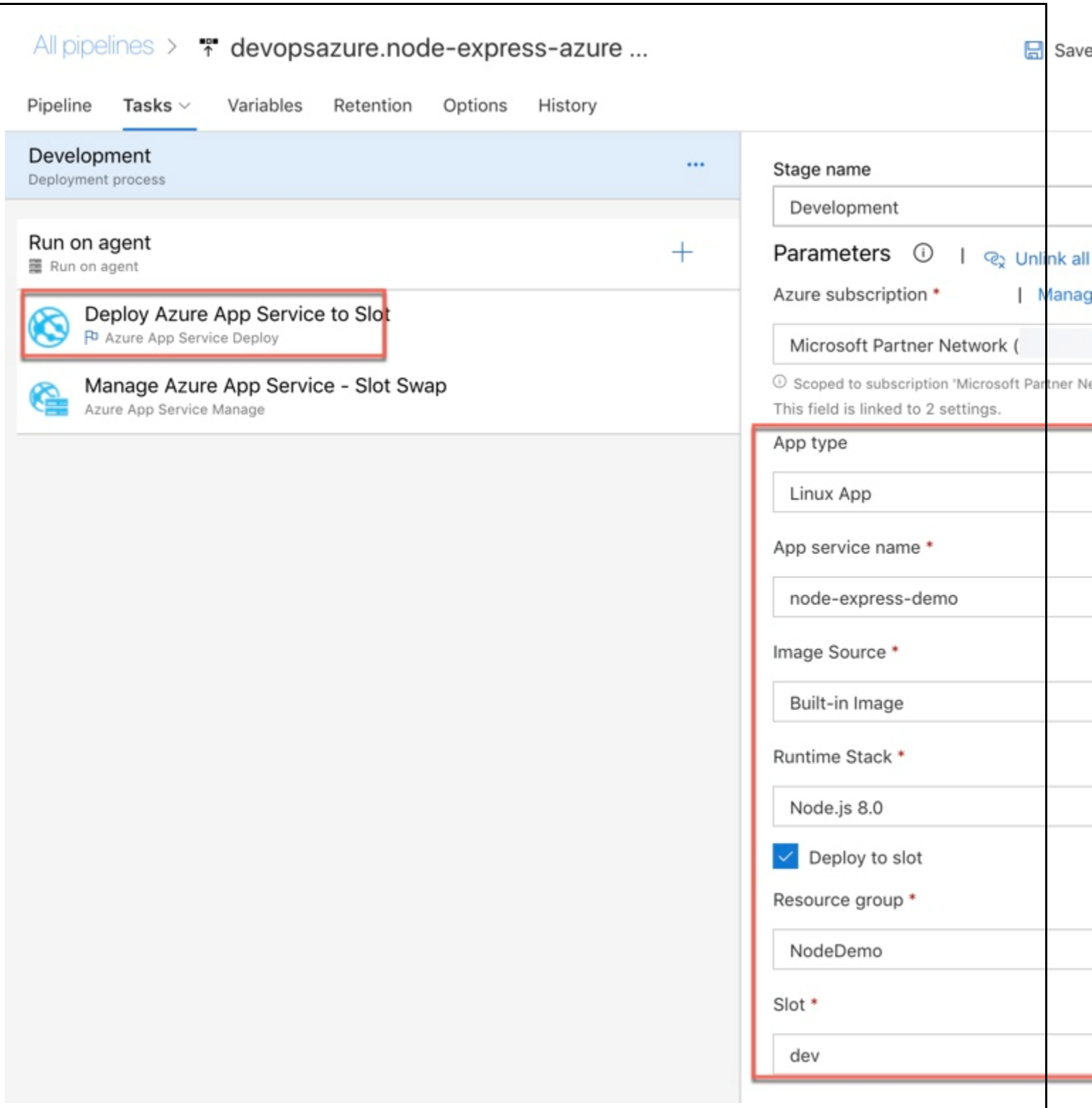


Figure 14. ***Configuring the Deployment Task.***

Next, highlight and remove the second deployment task for swapping the slots.

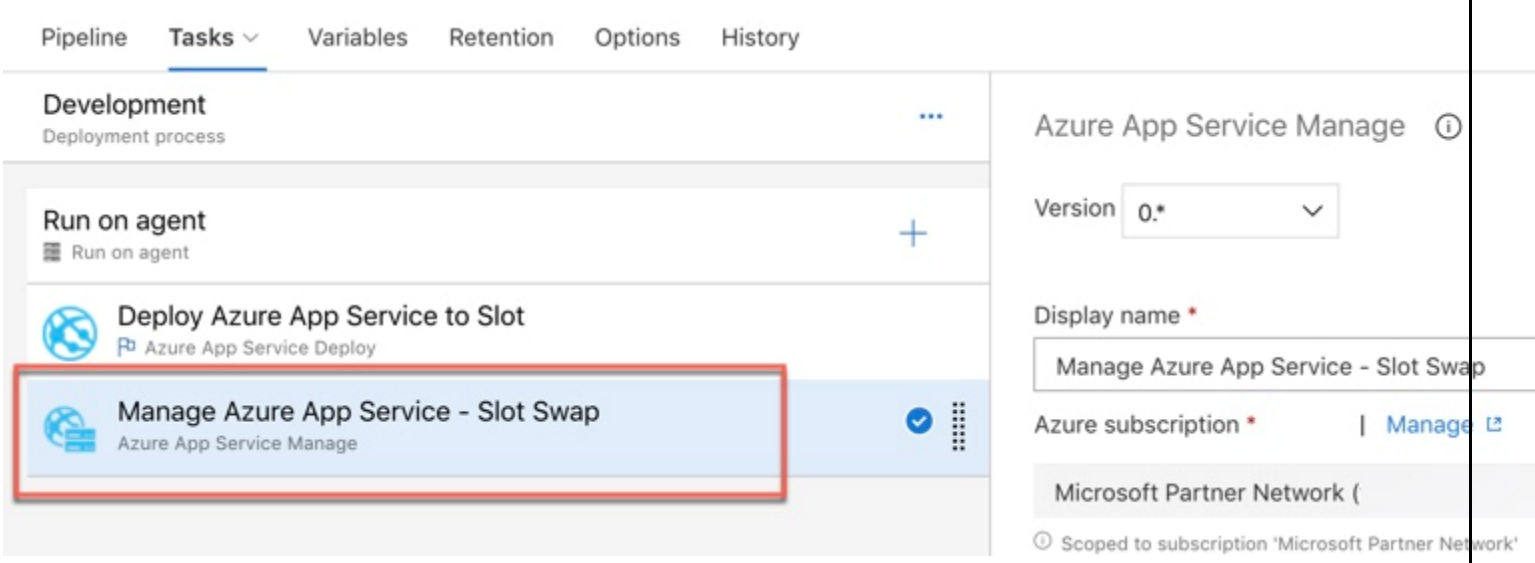


Figure 15. ***Removing the Slot Swap Task.***

Finally, click Save.

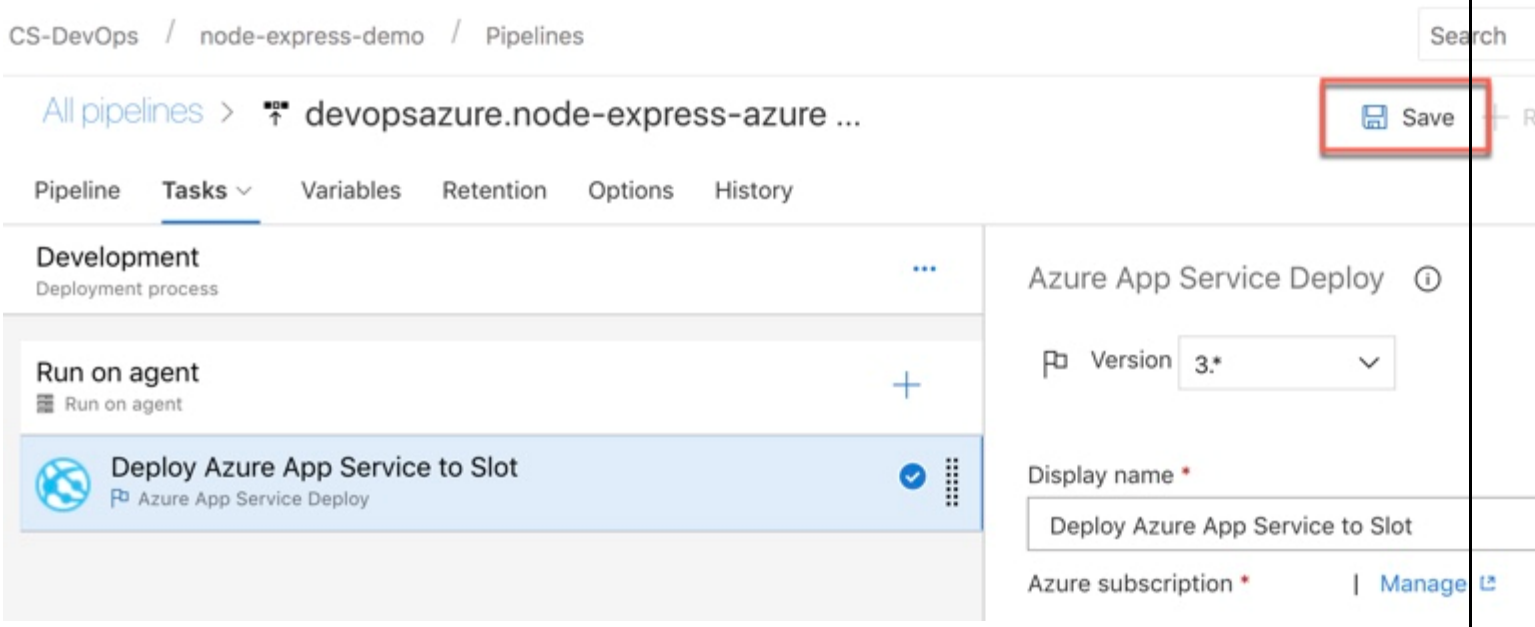


Figure 16. ***Save the Changes to the Development Stage.***

Head back over to the "Pipeline" tab at the top left of the screen. Inspect the deployment triggers for the artifacts as shown in **Figure 17**.

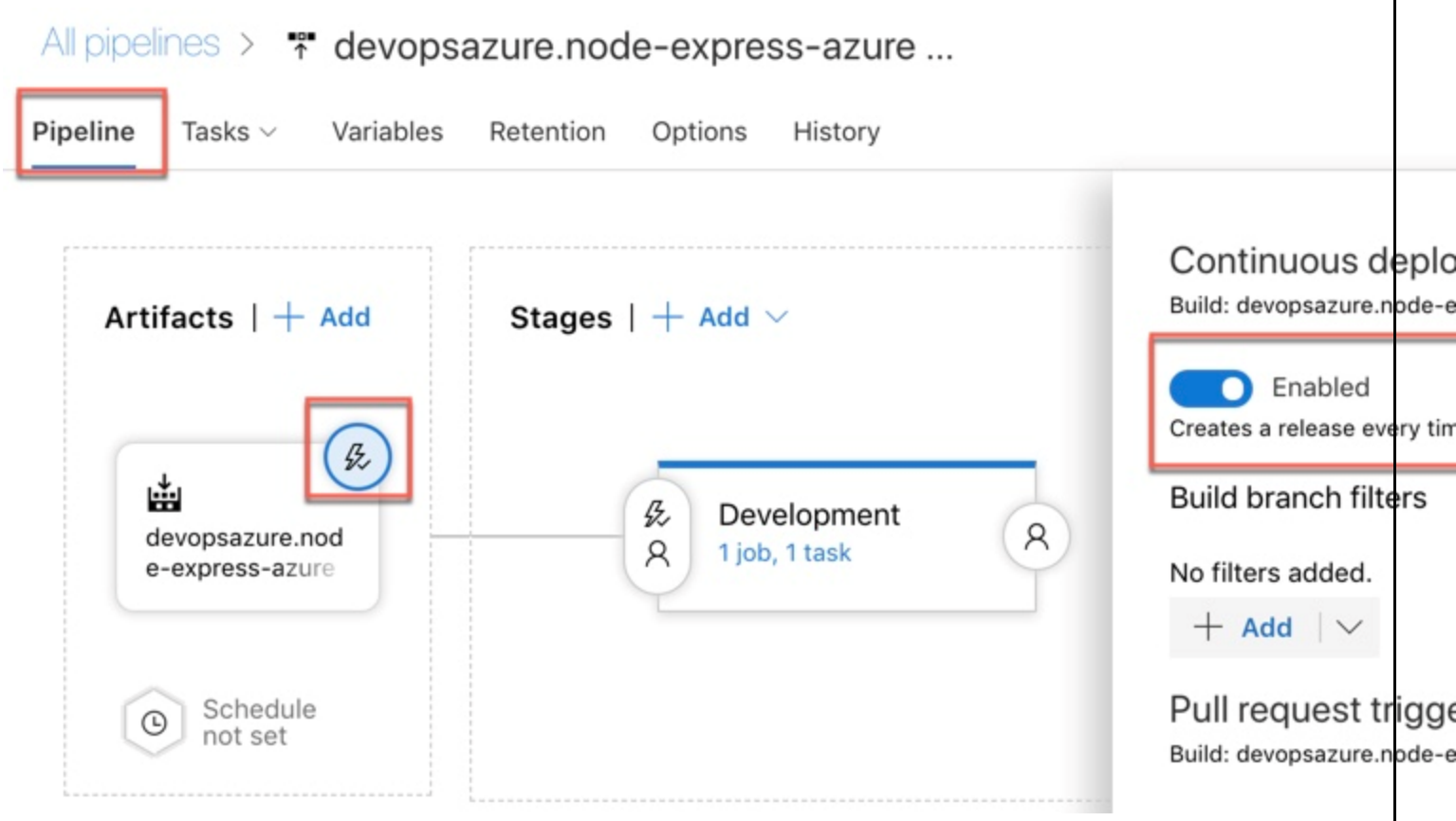


Figure 17. ***Reviewing the Deployment Trigger.***

Notice that continuous deployment is enabled by default. Going forward, each new build will trigger a deployment to our development slot in Azure App Service.

First, let's trigger a manual release.

Create Your First Release to Development

Click the "Release" button on the top right of the release pipeline screen and create a new release. Use the settings as shown in **Figure 18**.

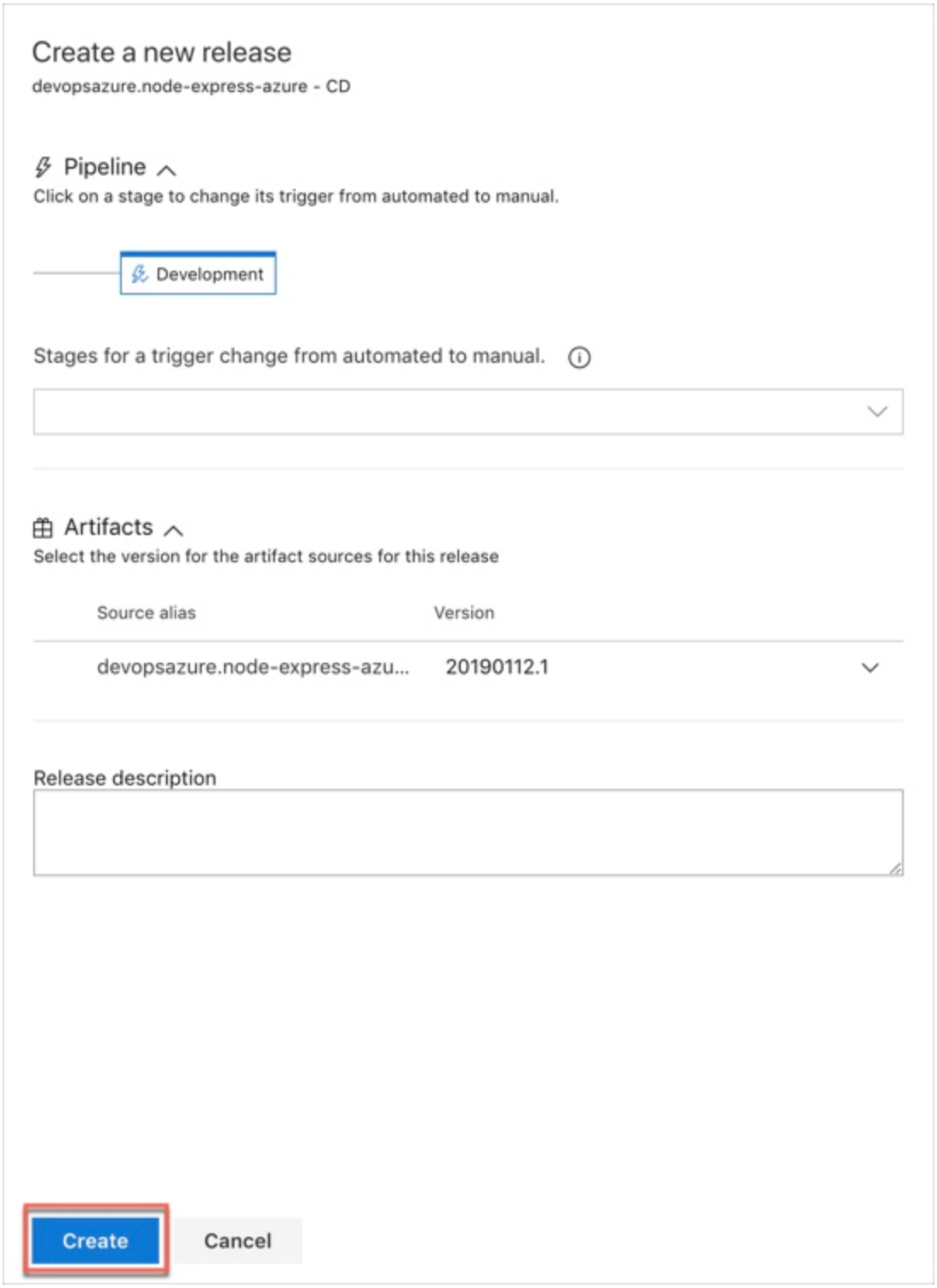


Figure 18. ***Triggering a Release.***

Click on the "Create" button to deploy the application to the development deployment slot. You should see a successful status in the properties of the release.

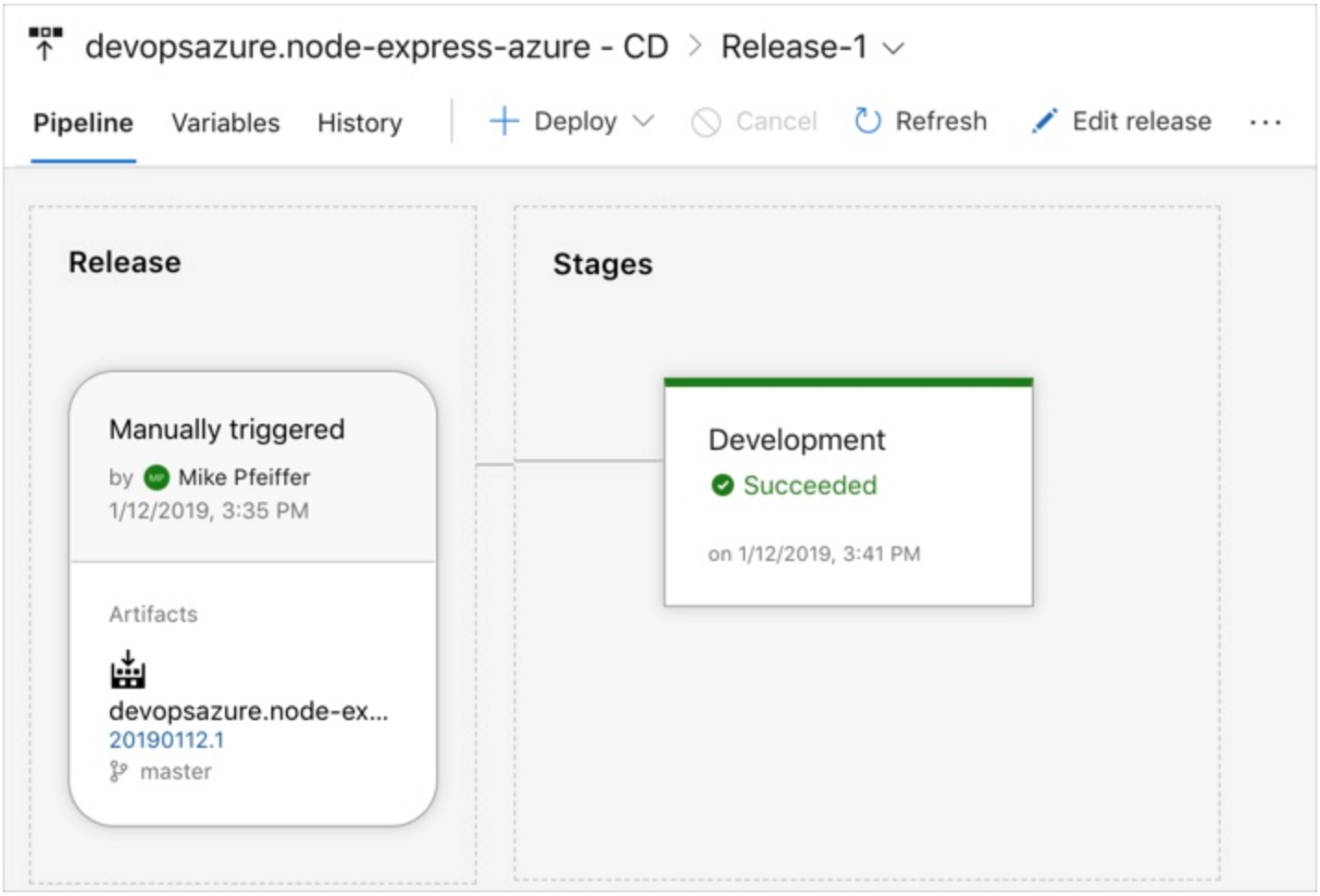


Figure 19. ***Reviewing the Release Status.***

Navigate to the public URL of the "dev" deployment slot in your web browser. The hostname will have "-dev" appended to it. For example, my web app is named "node-express-demo" and the "dev" deployment slot URL is https://node-express-demo-dev.azurewebsites.net.

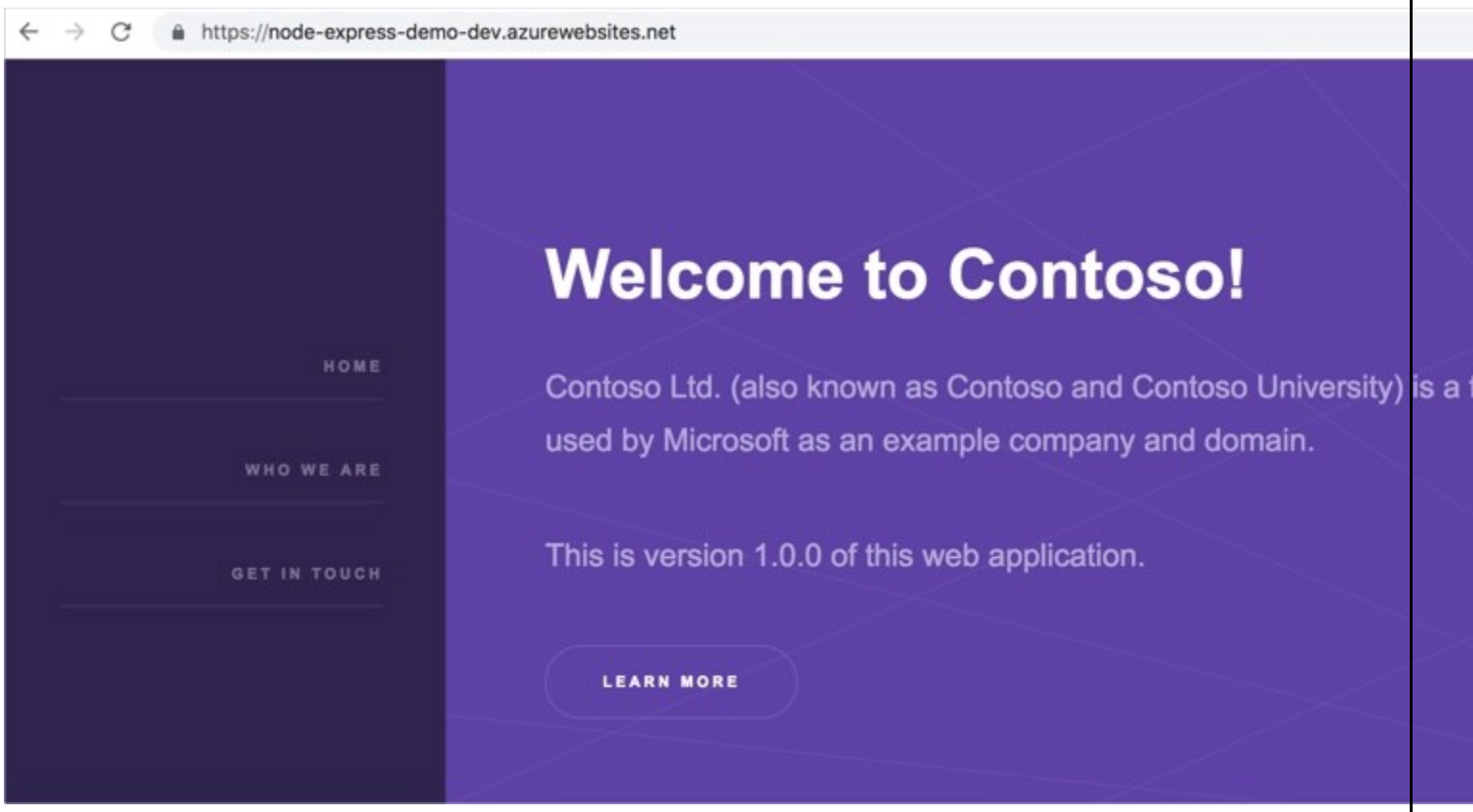


Figure 20. ***The Demo Node App Running in the "dev" Deployment Slot.***

You should see the sample web application when you visit the "dev" slot URL. The production slot will show the default Azure App Service splash page since it is virtually untouched at this point. Let's change that in the next step.

Create a Release Stage for Production

Head back over to the Azure DevOps portal and go to **Pipelines** > **Releases**. Click on the "Edit" button to modify the pipeline. Highlight the **Development** stage and click the dropdown to clone the stage.

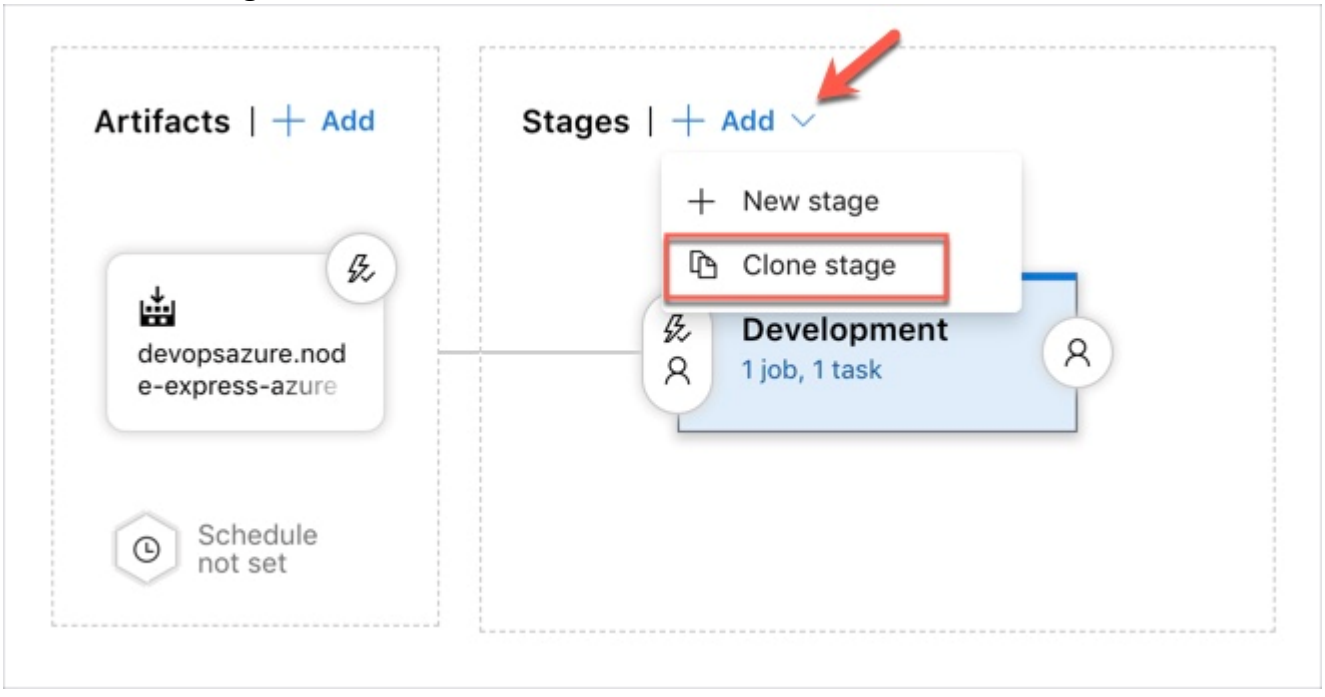


Figure 21. ***Cloning the Development Stage.***

Rename the stage to "Production".

Next, click the pre-deployment conditions button for the **Production** stage. Enable pre-deployment approvals and add yourself as an approver.

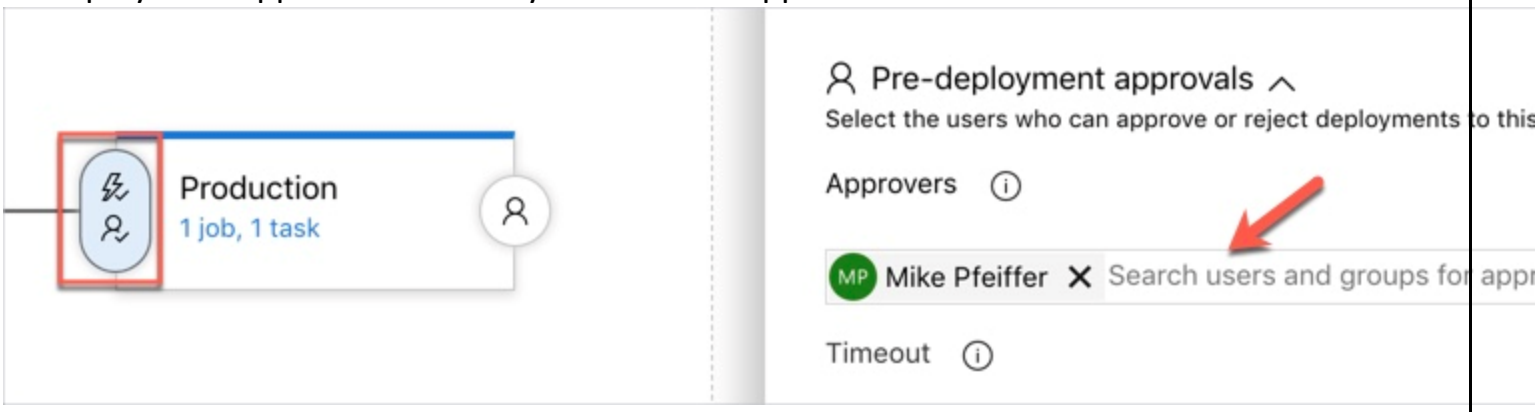


Figure 22. ***Requiring Approval for Production Deployment.***

We're doing this because we don't want automated deployments going straight into production. We're not building a continuous deployment pipeline for production. We're building continuous delivery pipeline.

Continuous delivery is a process that ensures our application is production ready. When we are doing a scheduled deployment we can do so with confidence since we know the application has been through a pipeline of tests before-hand.

Next, click on the "task" link on the **Production** stage. We need to modify this task so that it does not deploy our code into the development slot.

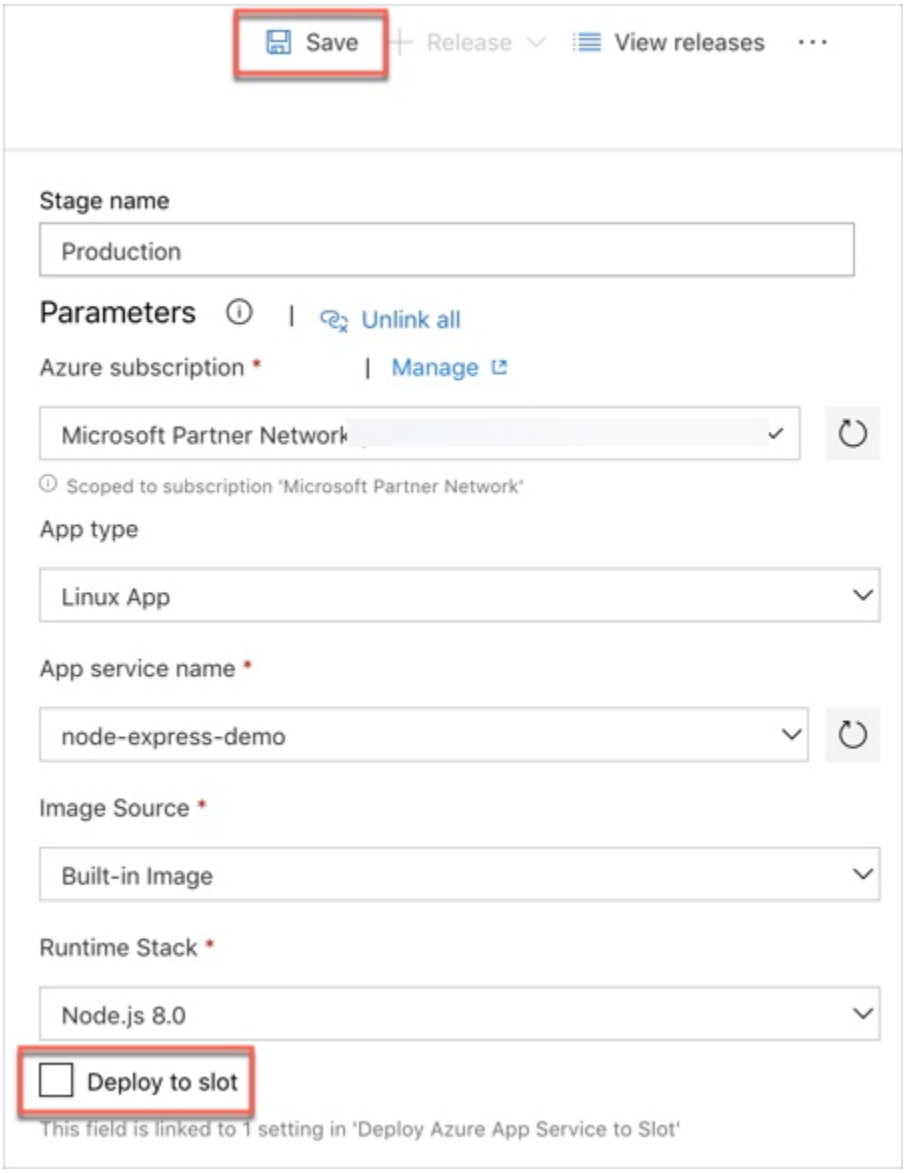


Figure 23. ***Updating the Deployment Task.***

Simply uncheck "slot" and this will infer that the production slot of the web app should be used during the deployment. Click save when complete.

Validating the Pipeline

Navigate to your GitHub account and into the views folder of the demo application. Edit the index.handlebars file to update the app to version 2.0.0.

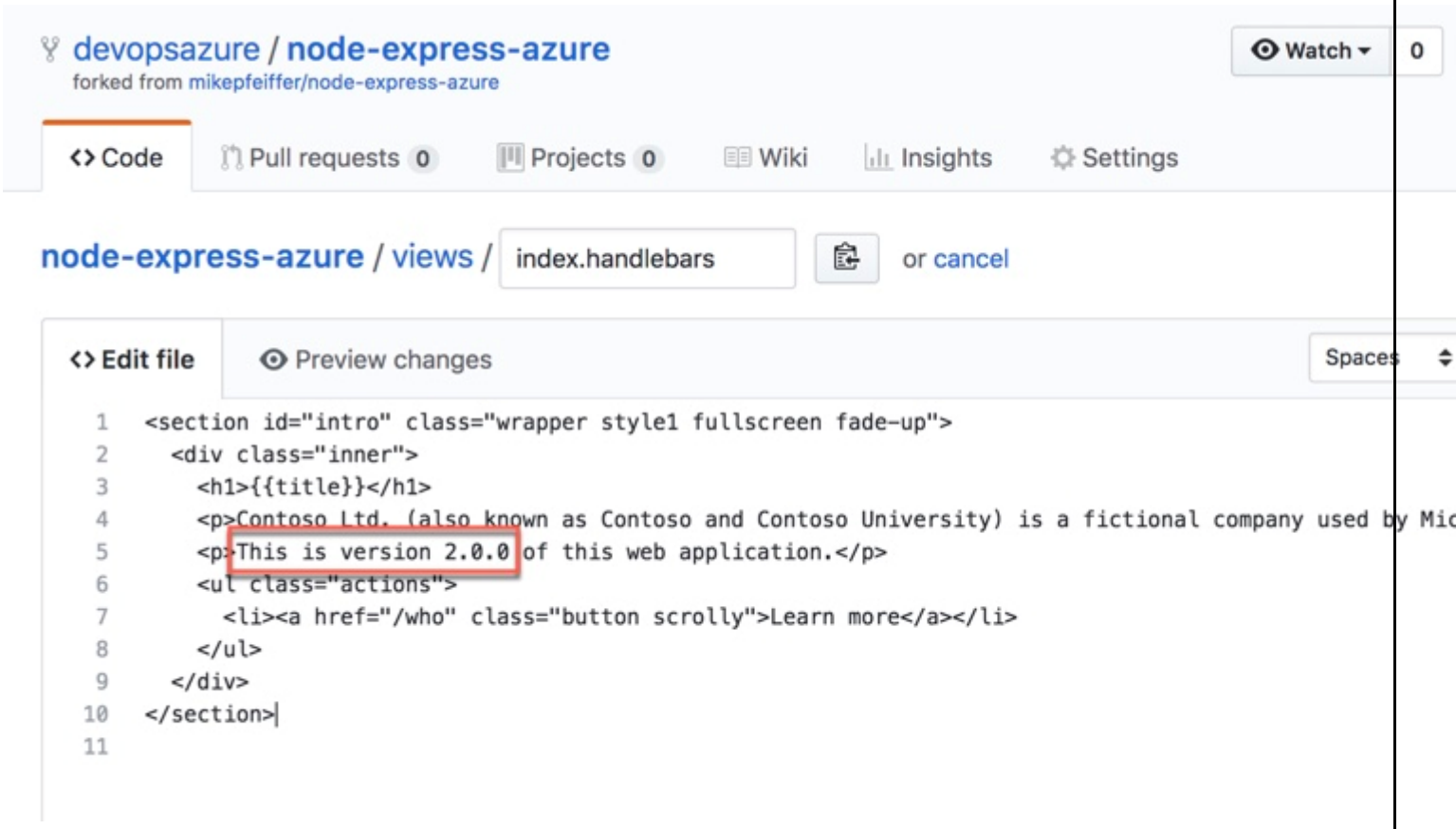


Figure 24. ***Radically Modifying the Code Base for the App.***

Committing the change in this repo should automatically trigger a build, perform our tests, and publish a deployment package. We can confirm this by reviewing the build status.

After the build, you should see a new release. The development stage should be green indicating that the deployment succeeded. The production stage should be blue and show that it's pending approval.

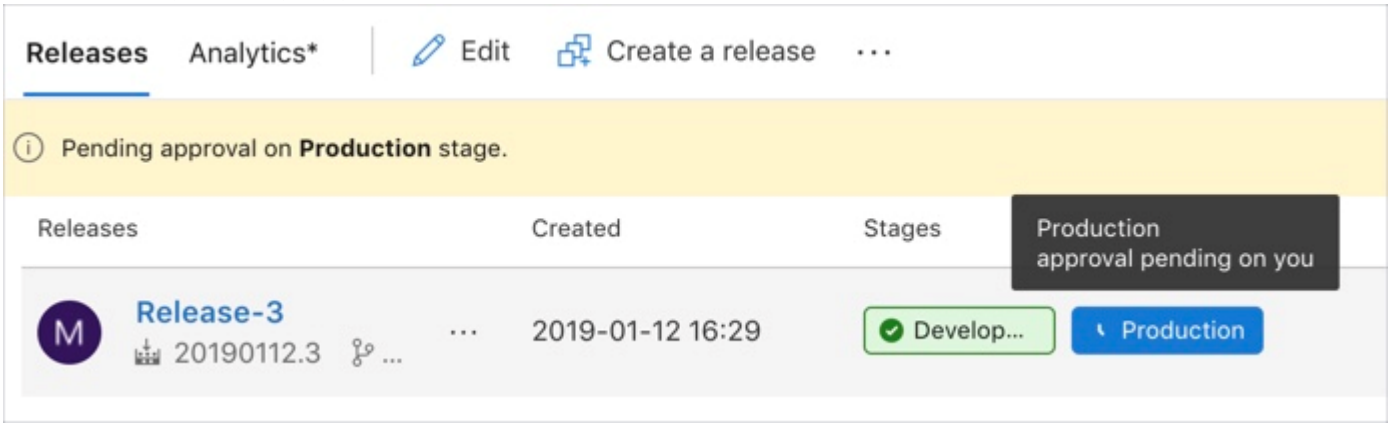


Figure 25. ***Reviewing the Release Status.***

Click approve to kick-off the production deployment.

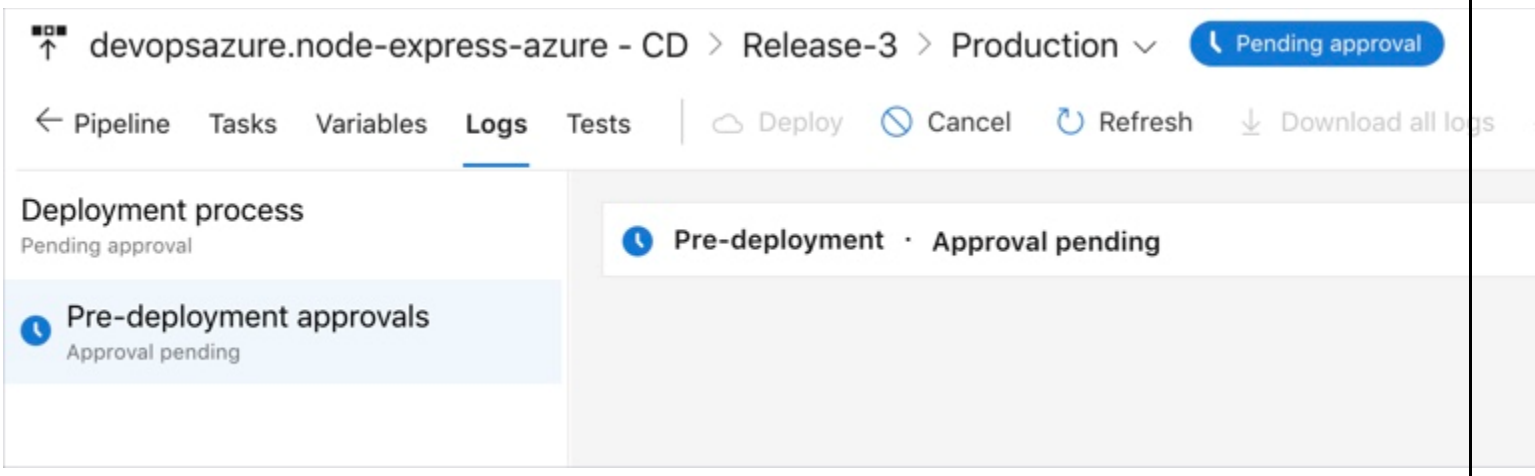


Figure 26. ***Deploying to Production. Never do this on a Friday!***

Go back to your pipeline view and you should see the deployment to production succeeded.

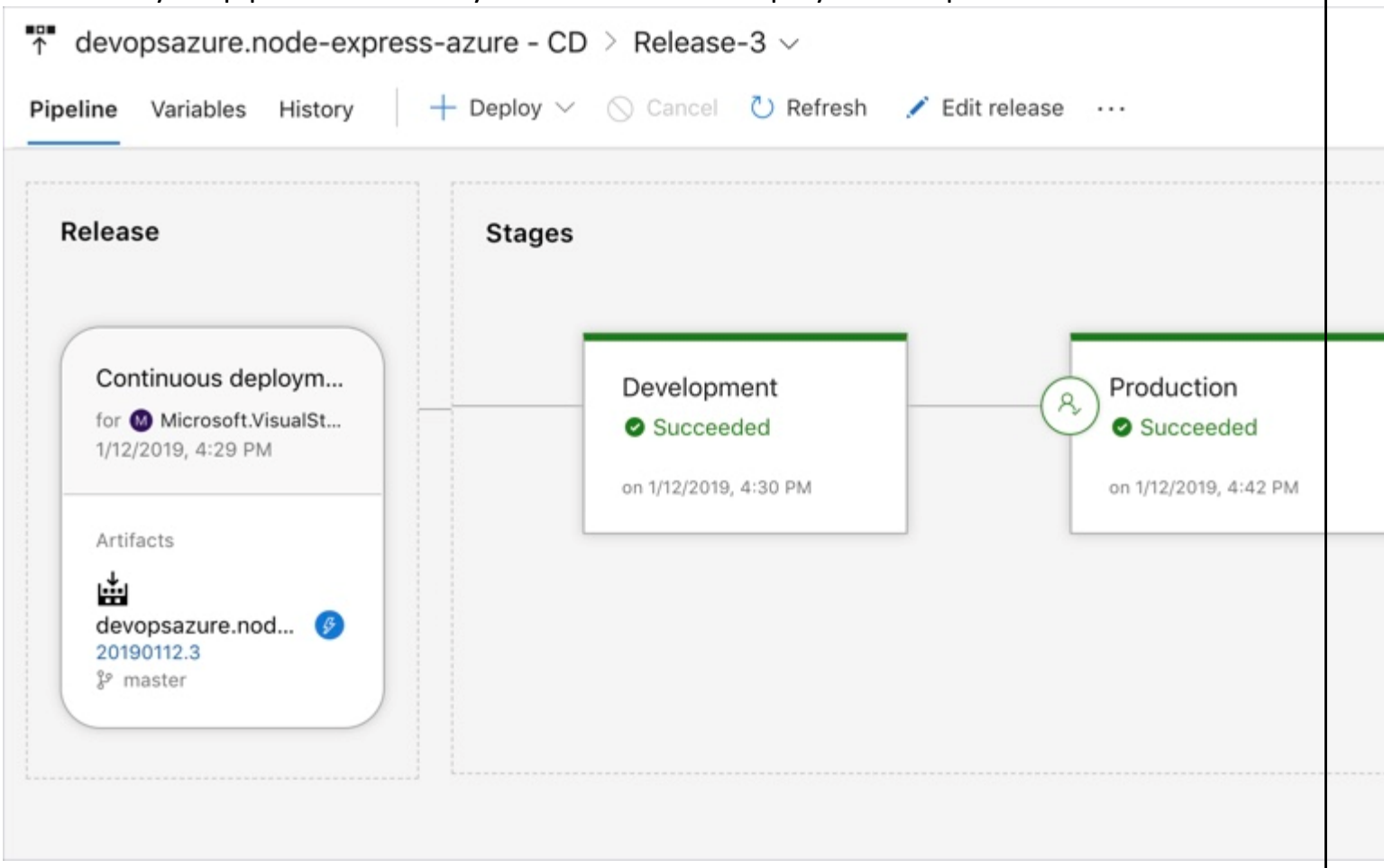


Figure 27. ***Validating the Status.***

Finally, head over to the web app URL for the production slot to confirm the correct version is running.

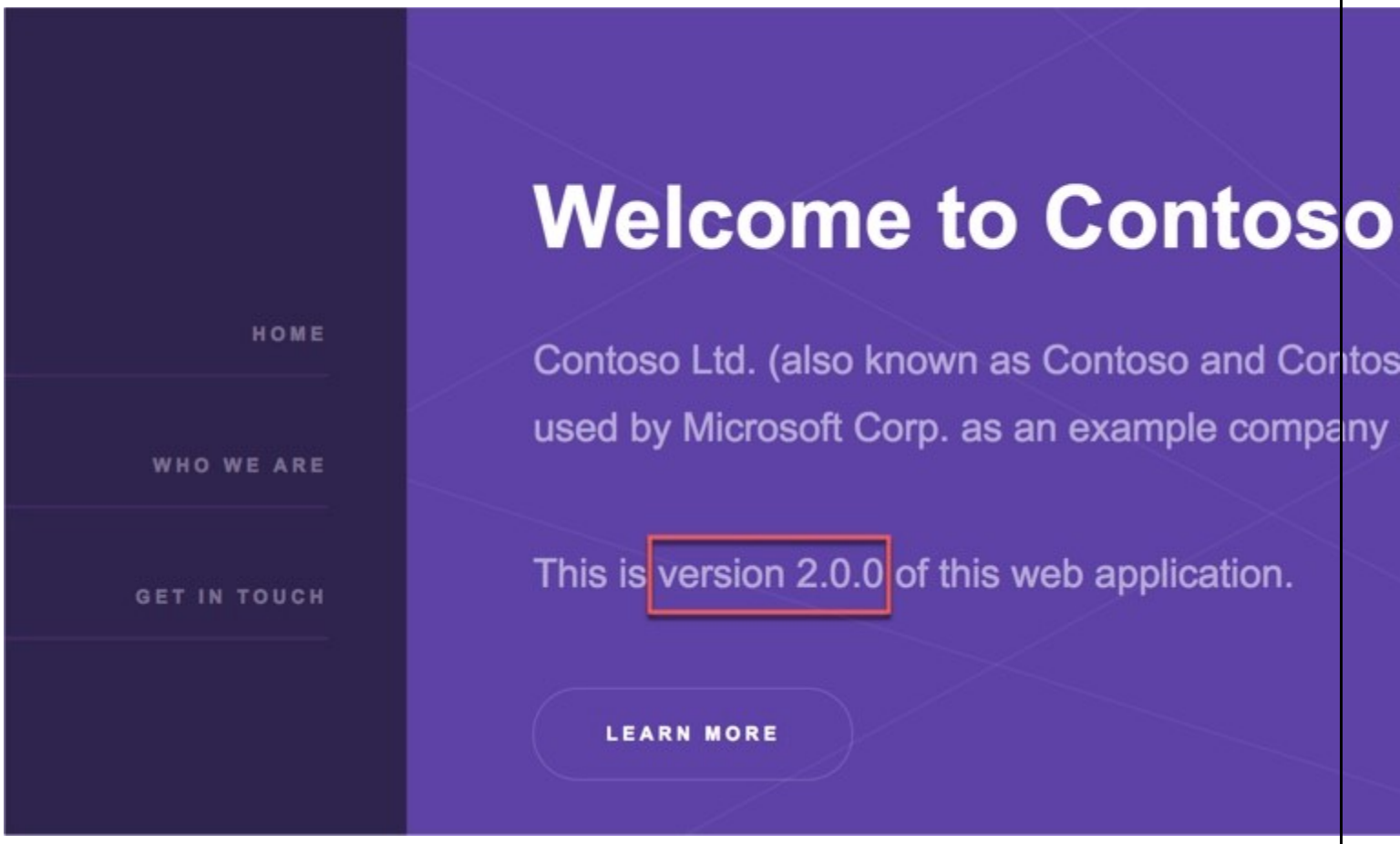


Figure 28. ***Reviewing the Version.***

You should see version 2.0.0 on the homepage.

Set up a Build Badge for Your Project

Have you ever seen those build pass/fail badges when browsing projects on GitHub? They're really cool because you can tell at a glance if the code is still working or if it's old and busted.

Let's setup a badge for this project.

Go back to your Builds section and click the status badge button.

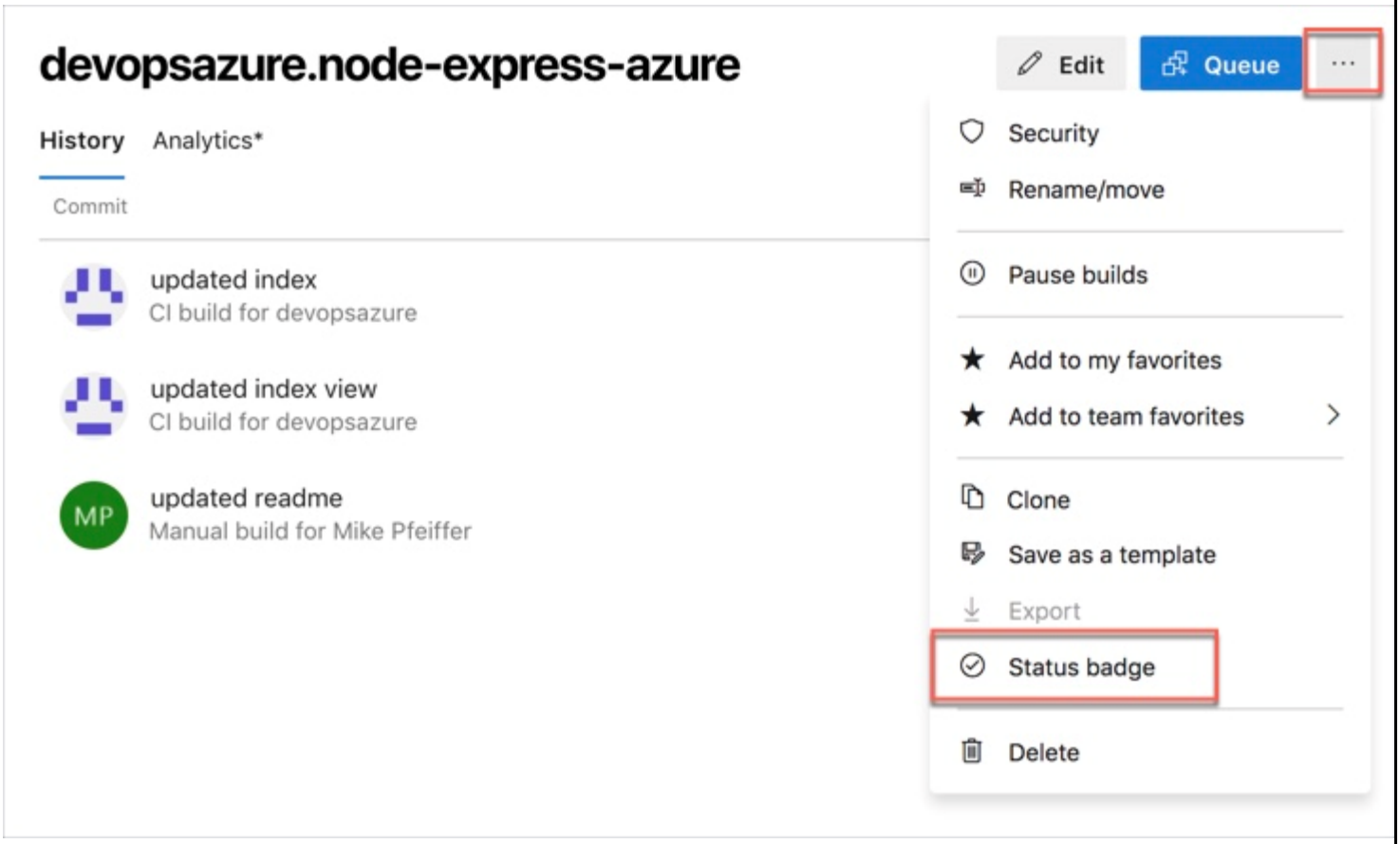


Figure 29. ***Locating the Status Badge Button.***

Copy the markdown code for the status badge.

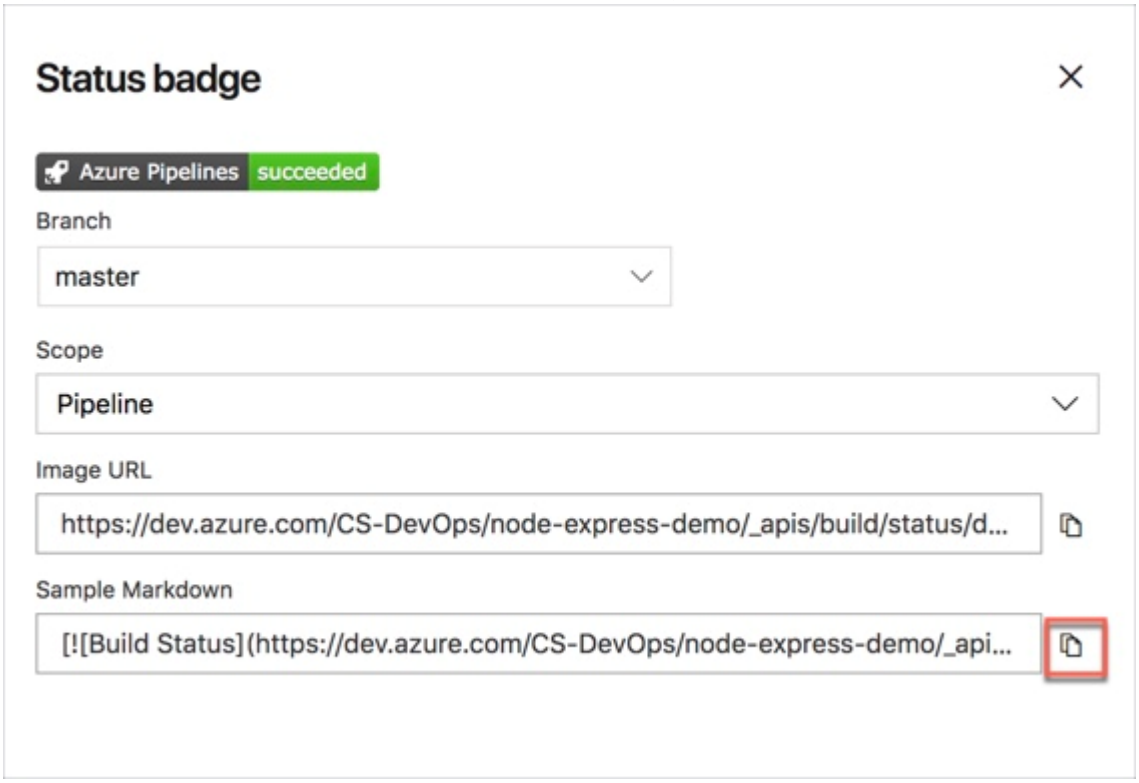


Figure 30. ***Copying the Markdown.***

Now, go back to GitHub and modify the README.md file in your **node-express-azure** repo. Paste the markdown you copied from the status badge page.

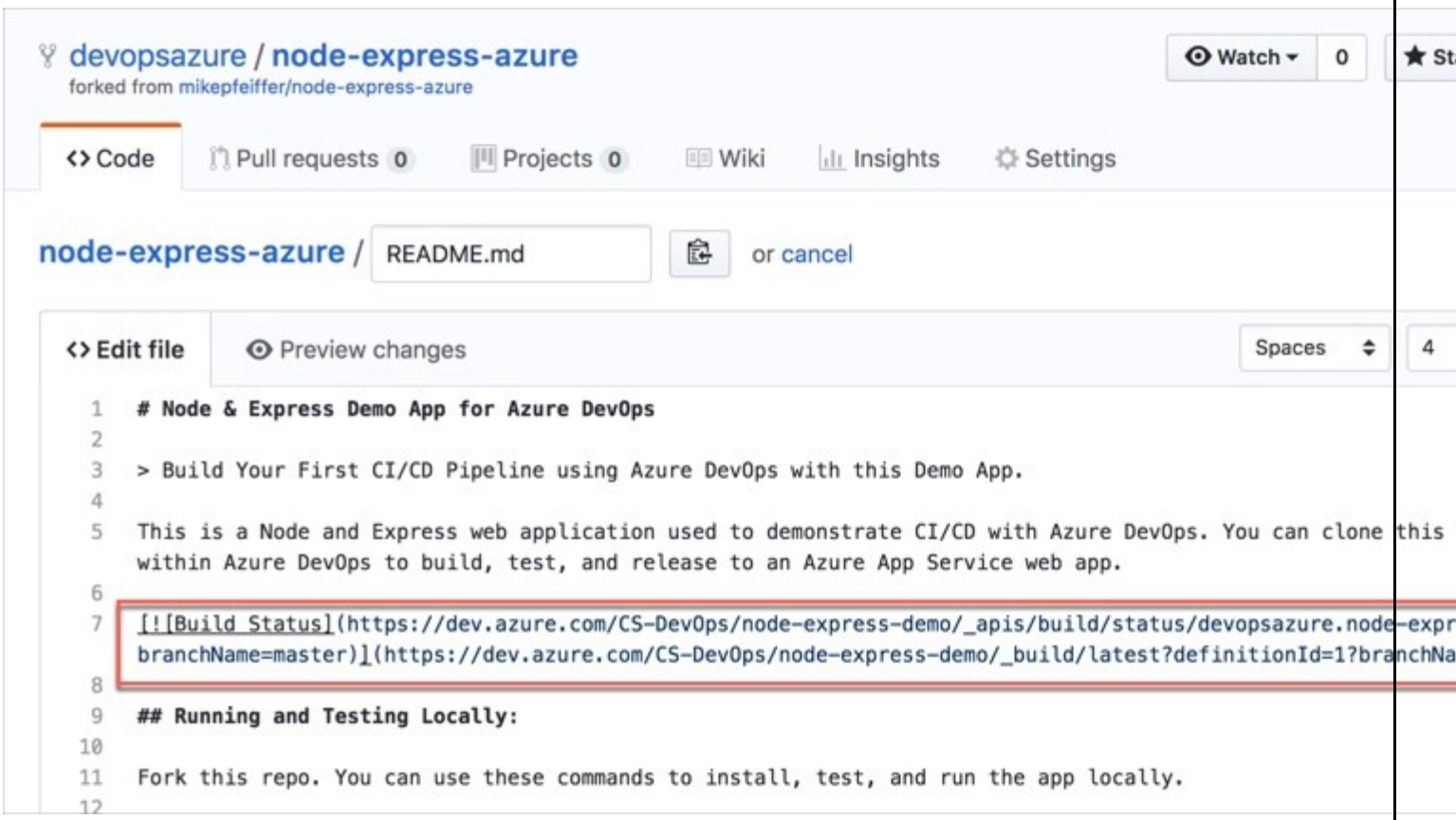


Figure 31. ***Updating README.md.***

Commit the change and view the README. You should see a build passing status icon.

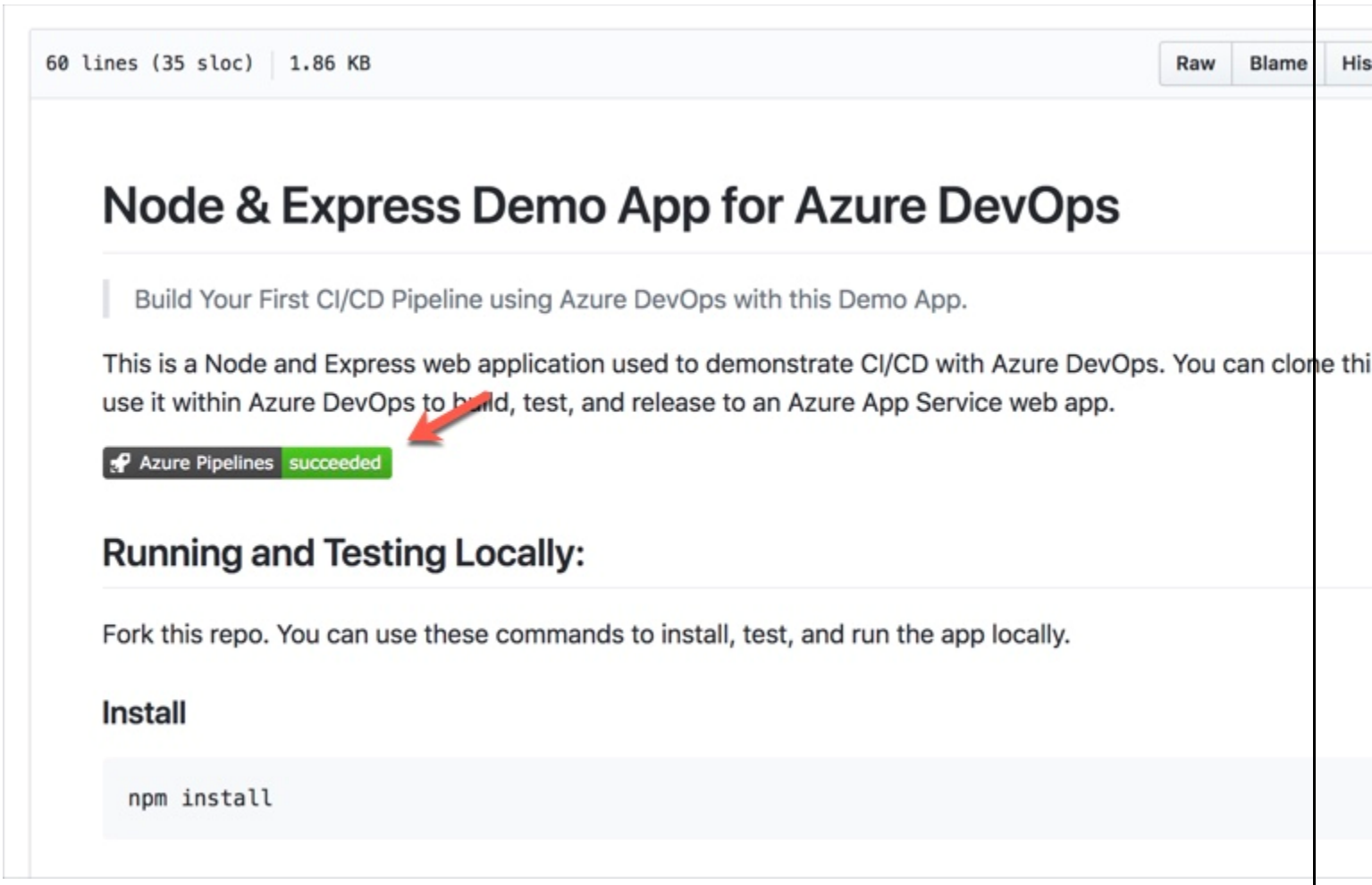


Figure 32. ***Badge Status Icon in Action.***

If you're still reading after all this time, respect! You now know who to build a CI/CD pipeline on Azure.

You can simply delete all the resources to clean things up. Delete the resource group you created for this project, delete the Node demo project in the Azure DevOps portal, and delete the GitHub repo that you forked from my account (unless you want to keep a copy).

34) how to change build format in azure devops?

PowerShell .NET Core Azure DevOps

- DevOps
- C# and .NET

People familiar with the. NET framework know that we can let the compiler self-increase the version number by specifying AssemblyVersion as 10.0.*. But. NET core and. NET Standard are not. Even with open source projects like MSBump, there are certain flaws. Typically, such requirement happens on a CI/CD server. Let's take a look at how to easily handle it with Azure DevOps.

About Versioning in .NET Core Applications

I have wrote a post on the. NET Core Application Version number: https://edi.wang/post/2018/9/27/get-app-version-net-core
We're going to control **Version** field this time.

Use **dotnet build** to compile your application into dll files, and this version will be showed in the property window.
But actually, the **build** command can have a parameter that sets a specific version number, like this:

```
dotnet build /p:Version=10.0.8888.1234
```

Based on this, we can control the build number for a .NET Core application on build servers.

Why Not Use MSBump

You may heard of a project: Msbump, which can also be used to change the version number at compile time. However, it modifies the **csproj** file at compile time and is a code change. The traditional. NET FX compilation system does not change the code. This is unacceptable to me because it is a kind of uncontrollable factor. It's not a good practice to check in the csproj file that just changed the version number. And instead of generating the version number based on the timestamp, MSBump by default, generates the version by the current version string in the project file. So in a development team, everyone could have inconsistent version of the same csproj file. So I decided to give up MSBump and try to use Microsoft's own technology to solve it's own problem.

Automatically Generate Version Numbers

In a fully automated CI environment, it is not possible for us to manually specify a version number each time. I need a rule and a method to generate it automatically.
The rules I personally use is: **[Major].[Minor].[Number of days from January 1, 2000].[Lucky Number]**
I use PowerShell to calculate the number of days from 1/1/2000.

```
$baseDate = [datetime]"01/01/2000"
```

```
$currentDate = $(Get-Date)
```

```
$interval = NEW-TIMESPAN –Start $baseDate –End $currentDate
```

```
$days = $interval.Days
```

Configure Azure DevOps

We basically need to tell Azure DevOps to calculate the version number, and use **/p:Version** parameter to build our project.

Environment variables

Add a variable named **buildNumber** in build definition.

PowerShell Task

Add a **PowerShell Task** and put it as the **first** step in the build pipeline. This PowerShell Task need to calculate the version number and assign it to **buildNumber** variable

To set value to a variable in Azure DevOps pipeline, use **Write-Host** as:

##vso[task.setvariable variable=variableName]variableValue

Then, our script will be

```
Write-Host "Generating Build Number"
$baseDate = [datetime]"01/01/2000"
$currentDate = $(Get-Date)
$interval = NEW-TIMESPAN –Start $baseDate –End $currentDate
$days = $interval.Days

Write-Host "##vso[task.setvariable variable=buildNumber]10.0.$days.1024"
```

Choose **Inline** and copy this script to the textarea.

Modify .NET Core Task Parameter

Append this to the **Arguments** filed of both **Build** and **Publish** steps.

```
/p:Version=$(buildNumber)
```

Notice: there should be a space [ ] before **/p**

Compile and Done

Trigger a new build, you can see the version number is being set in the build log
and the application will have an updated version after deployment

Azure Pipelines | Azure DevOps Server 2019 | TFS 2018 | TFS 2017 | TFS 2015 | Visual Studio 2017 | Visual Studio 2015

Performing user interface (UI) testing as part of the release pipeline is a great way of detecting unexpected changes, and need not be difficult. This topic describes using Selenium to test your website during a continuous deployment release and test automation. Special considerations that apply when running UI tests are discussed in UI testing considerations.

Typically you will run unit tests in your build workflow, and functional (UI) tests in your release workflow after your app is deployed (usually to a QA environment).

For more information about Selenium browser automation, see:

- Selenium HQ
- Selenium documentation

Create your test project

As there is no template for Selenium testing, the easiest way to get started is to use the Unit Test template. This automatically adds the test framework references and enables you run and view the results from Visual Studio Test Explorer.

1. In Visual Studio, open the **File** menu and choose **New Project**, then choose **Test** and select **Unit Test Project**. Alternatively, open the shortcut menu for the solution and choose **Add** then **New Project** and then **Unit Test Project**.
2. After the project is created, add the Selenium and browser driver references used by the browser to execute the tests. Open the shortcut menu for the Unit Test project and choose **Manage NuGet Packages**. Add the following packages to your project:

- Selenium.WebDriver
- Selenium.Firefox.WebDriver
- Selenium.WebDriver.ChromeDriver
- Selenium.WebDriver.IEDriver

Create your tests. For example, the following code creates a default class named **MySeleniumTests** that performs a simple test on the Bing.com website. Replace the contents of the **TheBingSearchTest** function with the Selenium code required to test your web app or website. Change the **browser** assignment in the **SetupTest** function to the browser you want to use for the test.

```
C#Copy
using System;
using System.Text;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.IE;

namespace SeleniumBingTests
{
  /// <summary>
  /// Summary description for MySeleniumTests
  /// </summary>
  [TestClass]
  public class MySeleniumTests
  {
    private TestContext testContextInstance;
    private IWebDriver driver;
    private string appURL;

    public MySeleniumTests()
    {
    }

    [TestMethod]
    [TestCategory("Chrome")]
    public void TheBingSearchTest()
    {
      driver.Navigate().GoToUrl(appURL + "/");
      driver.FindElement(By.Id("sb_form_q")).SendKeys("Azure Pipelines");
      driver.FindElement(By.Id("sb_form_go")).Click();
      driver.FindElement(By.XPath("//ol[@id='b_results']/li/h2/a/strong[3]")).Click();
      Assert.IsTrue(driver.Title.Contains("Azure Pipelines"), "Verified title of the page");
    }

    /// <summary>
    ///Gets or sets the test context which provides
    ///information about and functionality for the current test run.
    ///</summary>
```

```
public TestContext TestContext
{
 get
 {
  return testContextInstance;
 }
 set
 {
  testContextInstance = value;
 }
}

[TestInitialize()]
public void SetupTest()
{
 appURL = "http://www.bing.com/";

 string browser = "Chrome";
 switch(browser)
 {
  case "Chrome":
   driver = new ChromeDriver();
   break;
  case "Firefox":
   driver = new FirefoxDriver();
   break;
  case "IE":
   driver = new InternetExplorerDriver();
   break;
  default:
   driver = new ChromeDriver();
   break;
 }

}

[TestCleanup()]
public void MyTestCleanup()
{
 driver.Quit();
```

```
    }
  }
}
```

Run the Selenium test locally using Test Explorer and check that it works.

Define your build pipeline

You'll need a continuous integration (CI) build pipeline that builds your Selenium tests. For more details, see Build your .NET desktop app for Windows.

Create your web app

You'll need a web app to test. You can use an existing app, or deploy one in your continuous deployment (CD) release pipeline. The example code above runs tests against Bing.com. For details of how to set up your own release pipeline to deploy a web app, see Deploy to Azure Web Apps.

Decide how you will deploy and test your app

You can deploy and test your app using either the Microsoft-hosted agent in Azure, or a self-hosted agent that you install on the target servers.

- When using the **Microsoft-hosted agent**, you should use the Selenium web drivers that are pre-installed on the Windows agents (agents named **Hosted VS 20xx**) because they are compatible with the browser versions installed on the Microsoft-hosted agent images. The paths to the folders containing these drivers can be obtained from the environment variables named IEWebDriver (Internet Explorer), ChromeWebDriver (Google Chrome), and GeckoWebDriver (Firefox). The drivers are **not** pre-installed on other agents such as Linux, Ubuntu, and macOS agents. Also see UI testing considerations.
- When using a **self-hosted agent** that you deploy on your target servers, agents must be configured to run interactively with auto-logon enabled. See Build and release agents and UI testing considerations.

Include the test in a release

1. If you don't have an existing release pipeline that deploys your web app:

- Open the **Releases** page in the **Azure Pipelines** section in Azure DevOps or the **Build & Release** hub in TFS (see Web portal navigation) and choose the **+** icon, then choose **Create release pipeline**.

- Select the **Azure App Service Deployment** template and choose **Apply**.
- In the **Artifacts** section of the **Pipeline** tab, choose **+ Add**. Select your build artifacts and choose **Add**.

- Choose the **Continuous deployment trigger** icon in the **Artifacts** section of the **Pipeline** tab. In the Continuous deployment trigger pane, enable the trigger so that a new release is created from every build. Add a filter for the default branch.

- Open the **Tasks** tab, select the **Stage 1** section, and enter your subscription information and the name of the web app where you want to deploy the app and tests. These settings are applied to the **Deploy Azure App Service** task.

  If you are deploying your app and tests to environments where the target machines that host the agents do not have Visual Studio installed:
- In the **Tasks** tab of the release pipeline, choose the **+** icon in the **Run on agent** section. Select the **Visual Studio Test Platform Installer** task and choose **Add**. Leave all the settings at the default values.

You can find a task more easily by using the search textbox.

In the **Tasks** tab of the release pipeline, choose the **+** icon in the **Run on agent** section. Select the **Visual Studio Test** task and choose **Add**.

If you added the **Visual Studio Test Platform Installer** task to your pipeline, change the **Test platform version** setting in the **Execution options** section of the **Visual Studio Test** task to **Installed by Tools Installer**.

How do I pass parameters to my test code from a build pipeline?

Save the release pipeline and start a new release. You can do this by queuing a new CI build, or by choosing **Create release** from the **Release** drop-down list in the release pipeline.

To view the test results, open the release summary from the **Releases** page and choose the **Tests** link.

35) multiple storage account using single arm template?

36) Nested ARM template?

Azure ARM allows you to deploy your solution either as a single template or by distributing the responsibilities into multiple templates. The latter is considered to be a good coding practice. But why is it needed and recommended as a good coding practice? Our cloud applications, software architecture and environments are getting more and more complex with new technologies, apps and devices constantly flooding the market. We have to think about questions like what if there are sub environments? How can you manage data better? How do you ensure maximum reusability with less effort?

Splitting your template over multiple files can help manage data better and help with readability and organization. You can have a single main Azure ARM template that drives multiple Azure ARM nested templates. Nested Azure ARM templates are very common for more advance scenarios like deployment of more than one simplistic VM. There may be a bunch of databases, VMs, queues and so on for a complex environment. You can break down a solution into targeted components using nested Azure ARM templates. Another advantage is the modularization and reusability of the nested templates and the template code.

Nested Templates best practices:

- You should have one main Azure ARM template that is used for all types of input parameters.
- You can use a shared Azure resource that you can use to deploy all of the Azure resources which are mandatory for your environment deployment.
- You can have an optional Azure resource template which will actually deploy those Azure resources which are optional in nature. It means that the value that you get from the parameters in the Azure ARM template can be parameterized whether or not you want to have a particular Azure resource deployed in your Azure environment.

Along with optional Azure resource template, you can also have member resource templates and script. Some examples of the member templates could be VM creation template, VNet creation template, etc.

Here I have a parent template opened up with the name Demo1ParentTemplate.json. This will be the primary operating Azure ARM template. This parent Azure ARM template uses a containerSasToken to load and execute another ARM template. A shared access signature (SAS) provides you with a way to grant limited access to objects in your storage account to other clients without exposing your account key. An SAS provides delegated access to resources in your storage account. With SAS, you can grant clients access to resources in

your storage account without sharing your account keys. This is the key point in using shared access signatures in your applications.

Along with the parent template, I have HelloWorld.json which is the child template that is loaded or referenced from the parent Azure ARM template.

In the parameter section, there is the containerSasToken and you can generate this containerSasToken using Azure PowerShell scripts.

In the resources section, there is the apiVersion and the name of the Azure resource. It is very critical that every Azure resource must have a name and type. So, the name of this resource is **linkedTemplate.** The type of this resource is Microsoft.Resources/Deployment. This tells that the resource define in this template is actually another nested Azure ARM template. Before we move ahead, we need to make a storage account in Azure because this Azure ARM template would be hosted in a storage account.

In the Azure portal, click on Create a resource, go to Storage and select Storage Account.

Select a subscription and for this demo purpose, create a new resource group. Give a name to the storage account. Select the location. Select Standard performance. For the account kind, choose StorageV2 and for the replication strategy, choose LRS. The access tier would be Hot. Click Review + Create.

Once the validation gets passed, click Create.

Now talking about the child template, the helloworld.json file. You can see that this template has no resources but it does have the output section. Once these templates are deployed, this nested template will get an output which will say that this is the output from the linked template. It does nothing but verifies that the parent Azure ARM template has executed the HelloWorld template as the child template.

Once the storage account gets created, select Blobs.

Right now, we don’t have any container in the blob. So, click on + to create a new container.

Give name test to the container and for the public access level, choose Container.

Open the container, click Upload and select the HelloWorld.json file. Click Upload to upload the file to the container.

It would get uploaded right away and you would be able to see the file inside the container.

Along with that, also upload the Demo1ParentTemplate.json file.

Now, we are going to execute the Azure ARM templates using Microsoft Azure PowerShell. Microsoft Azure PowerShell is an extension of Windows PowerShell. It lets Windows PowerShell users control Azure's robust functionality. From the command line, Azure PowerShell programmers use preset scripts called cmdlets to perform complex tasks. Azure PowerShell can also work programmatically to automate tasks.

Open PowerShell ISE, log in to Azure account and set your subscription in which your storage account has been created.

Run this cmdlet to set the current storage account to the one that you have created.

Run the second cmdlet to fetch the URL of the Demo1ParentTemplate.json that we have uploaded in the blob container. You can also verify the URL by running the command $url.

In this scenario, we are not using an SAS token. Rather, we are using a generally available public container. You can also use an SAS token and then pass the token value which is highly recommended for production scenarios. But to make things easier here, we are simply using a publicly accessible container.

Then, execute the third cmdlt which will create a new resource group where we are going to deploy the nested templates.

Run the cmdlet again to fetch the URL for the Helloworld.json template and copy the url.

Back in VS Code, paste the url and modify the code as shown in the image. Remove the SAS token parameters as well. The final Demo1ParentTemplate.json file should look like this. So this url will call the Helloworld.json once this template gets deployed and then the child template i.e. Helloworld.json will be called and then it will get deployed.

And this is the child template that will produce this output once it gets deployed. If you execute or trigger the parent template and inspect the output of the child template, that would prove that our parent template has successfully fired our child template.

Back in PowerShell, run the second cmdlet again to store the value of the parent template's url again in $url.

Finally, run the final cmdlet that fires our parent template. Once it gets deployed, the final output would be something like this that says that ‘this is the output from linked template’. Hence, we can say that the nested deployment was successful.

Back in the portal when go and check the recently created resource group, you will find 2 deployments that have succeeded.

And if you go to those deployments, you can see both the templates have got deployed.

So, this is how easy it is to deploy nested Azure ARM templates. You can use this concept when you have a huge single template and want to divide that bulky template into multiple templates.

37) call powershell script from arm template

No, Azure ARM could not execute scripts directly. Executing scripts need host, Azure template does not provide such host.

One solution, you could select Azure Custom Script Extension.

The Custom Script Extension downloads and executes scripts on Azure virtual machines.

38)

Save output from a PowerShell pipeline to a variable

John Savill | Jul 21, 2015

Q. How can I save the output of a pipeline of PowerShell to a variable?

A. Normally to save the output of a PowerShell command to a variable you can use:

$variable = <command>

If however you have a sequence of PowerShell commands and you wish to save the final output to a variable it may not seem obvious how to save it, i.e.

<command1> | <command2>

The output object of command2 will be saved to $variable. You may also use -outvariable with command2 however make sure you don't add $ to the variable and the output is also still sent to screen, i.e.

<command1> | <command2> -outvariable variable

Description

The **Set-Variable** cmdlet assigns a value to a specified variable or changes the current value. If the variable does not exist, the cmdlet creates it.

Examples

Example 1: Set a variable and get its value

PowerShellCopy

```
PS C:\> Set-Variable -Name "desc" -Value "A description"
PS C:\> Get-Variable -Name "desc"
```

These commands set the value of the desc variable to A description, and then gets the value of the variable.

Example 2: Set a global, read-only variable

PowerShellCopy

```
PS C:\> Set-Variable -Name "processes" -Value (Get-Process) -Option constant -Scope global -Description "All processes" -PassThru | Format-List -Property *
```

This command creates a global, read-only variable that contains all processes on the system, and then it displays all properties of the variable.

The command uses the **Set-Variable** cmdlet to create the variable. It uses the *PassThru* parameter to create an object representing the new variable, and it uses the pipeline operator (|) to pass the object to the Format-List cmdlet. It uses the *Property* parameter of Format-List with a value of all (*) to display all properties of the newly created variable.

The value, "(Get-Process)", is enclosed in parentheses to ensure that it is executed before being stored in the variable. Otherwise, the variable contains the words "Get-Process".

Example 3: Understand public vs. private variables

PowerShellCopy

```
PS C:\> New-Variable -Name "counter" -Visibility Public -Value 26
PS C:\> $Counter
26
PS C:\> Get-Variable c*

Name                           Value
----                           -----
Culture                        en-US
ConsoleFileName
ConfirmPreference              High
CommandLineParameters          {}
Counter                        26

PS C:\> Set-Variable -Name "counter" -Visibility Private
PS C:\> Get-Variable c*
```

```
Name                 Value
----                 -----
Culture              en-US
ConsoleFileName
ConfirmPreference    High
CommandLineParameters {}

 PS C:\> $counter
"Cannot access the variable '$counter' because it is a private variable"

PS C:\> .\use-counter.ps1
#Commands completed successfully.
```

This command shows how to change the visibility of a variable to Private. This variable can be read and changed by scripts with the required permissions, but it is not visible to the user.

The sample output shows the difference in the behavior of public and private variables

About Variables

- 09/03/2019
- 9 minutes to read
-
  -
  -
  -
  -
  -
  - +1

Short description

Describes how variables store values that can be used in PowerShell.

Long description

You can store all types of values in PowerShell variables. For example, store the results of commands, and store elements that are used in commands and expressions, such as names, paths, settings, and values.

A variable is a unit of memory in which values are stored. In PowerShell, variables are represented by text strings that begin with a dollar sign ($), such as $a, $process, or $my_var.

Variable names aren't case-sensitive, and can include spaces and special characters. But, variable names that include special characters and spaces are difficult to use and should be avoided. For more information, see Variable names that include special characters.

There are several different types of variables in PowerShell.

- User-created variables: User-created variables are created and maintained by the user. By default, the variables that you create at the PowerShell command line exist only while the PowerShell window is open. When the PowerShell windows is closed, the variables are deleted. To save a variable, add it to your PowerShell profile. You can also create variables in scripts with global, script, or local scope.
- Automatic variables: Automatic variables store the state of PowerShell. These variables are created by PowerShell, and PowerShell changes their values as required to maintain their accuracy. Users can't change the value of these variables. For example, the $PSHOME variable stores the path to the PowerShell installation directory.

For more information, a list, and a description of the automatic variables, see about_Automatic_Variables.

- Preference variables: Preference variables store user preferences for PowerShell. These variables are created by PowerShell and are populated with default values. Users can change the values of these variables. For example, the $MaximumHistoryCount variable determines the maximum number of entries in the session history.

For more information, a list, and a description of the preference variables, see about_Preference_Variables.

Working with variables

To create a new variable, use an assignment statement to assign a value to the variable. You don't have to declare the variable before using it. The default value of all variables is $null.

To get a list of all the variables in your PowerShell session, type Get-Variable. The variable names are displayed without the preceding dollar ($) sign that is used to reference variables.

For example:

PowerShellCopy

```
$MyVariable = 1, 2, 3

$Path = "C:\Windows\System32"
```

Variables are useful for storing the results of commands.

For example:

PowerShellCopy

```
$Processes = Get-Process

$Today = (Get-Date).DateTime
```

To display the value of a variable, type the variable name, preceded by a dollar sign ($).
For example:

```
PowerShellCopy
$MyVariable
OutputCopy
1
2
3
PowerShellCopy
$Today
OutputCopy
Tuesday, September 3, 2019 09:46:46
```

To change the value of a variable, assign a new value to the variable.
The following examples display the value of the $MyVariable variable, changes the value of the variable, and then displays the new value.

```
PowerShellCopy
$MyVariable = 1, 2, 3
$MyVariable
OutputCopy
1
2
3
PowerShellCopy
$MyVariable = "The green cat."
$MyVariable
OutputCopy
The green cat.
```

To delete the value of a variable, use the Clear-Variable cmdlet or change the value to $null.

```
PowerShellCopy
Clear-Variable -Name MyVariable
PowerShellCopy
$MyVariable = $null
```

To delete the variable, use Remove-Variable or Remove-Item.

```
PowerShellCopy
Remove-Variable -Name MyVariable
PowerShellCopy
Remove-Item -Path Variable:\MyVariable
```

Types of variables

You can store any type of object in a variable, including integers, strings, arrays, and hash tables. And, objects that represent processes, services, event logs, and computers.

PowerShell variables are loosely typed, which means that they aren't limited to a particular type of object. A single variable can even contain a collection, or array, of different types of objects at the same time.

The data type of a variable is determined by the .NET types of the values of the variable. To view a variable's object type, use Get-Member.

For example:

PowerShellCopy

```
$a = 12                    # System.Int32
$a = "Word"                # System.String
$a = 12, "Word"            # array of System.Int32, System.String
$a = Get-ChildItem C:\Windows   # FileInfo and DirectoryInfo types
```

You can use a type attribute and cast notation to ensure that a variable can contain only specific object types or objects that can be converted to that type. If you try to assign a value of another type, PowerShell tries to convert the value to its type. If the type can't be converted, the assignment statement fails.

To use cast notation, enter a type name, enclosed in brackets, before the variable name (on the left side of the assignment statement). The following example creates a $number variable that can contain only integers, a $words variable that can contain only strings, and a $dates variable that can contain only **DateTime** objects.

PowerShellCopy

```
[int]$number = 8
$number = "12345"  # The string is converted to an integer.
$number = "Hello"
```

OutputCopy

```
Cannot convert value "Hello" to type "System.Int32". Error: "Input string was
 not in a correct format."
At line:1 char:1
+ $number = "Hello"
+ ~~~~~~~~~~~~~~~~
+ CategoryInfo          : MetadataError: (:) [],
   ArgumentTransformationMetadataException
+ FullyQualifiedErrorId : RuntimeException
```

PowerShellCopy

```
[string]$words = "Hello"
$words = 2      # The integer is converted to a string.
$words += 10    # The plus (+) sign concatenates the strings.
$words
```

OutputCopy
210
PowerShellCopy
[datetime] $dates = "09/12/91" # The string is converted to a DateTime object.
$dates
OutputCopy
Thursday, September 12, 1991 00:00:00
PowerShellCopy
$dates = 10   # The integer is converted to a DateTime object.
$dates
OutputCopy
Monday, January 1, 0001 00:00:00

Next 40 questions:

1) storage account types in azure?

Types of storage accounts

Azure Storage offers several types of storage accounts. Each type supports different features and has its own pricing model. Consider these differences before you create a storage account to determine the type of account that is best for your applications. The types of storage accounts are:

- **General-purpose v2 accounts**: Basic storage account type for blobs, files, queues, and tables. Recommended for most scenarios using Azure Storage.
- **General-purpose v1 accounts**: Legacy account type for blobs, files, queues, and tables. Use general-purpose v2 accounts instead when possible.
- **Block blob storage accounts**: Blob-only storage accounts with premium performance characteristics. Recommended for scenarios with high transactions rates, using smaller objects, or requiring consistently low storage latency.
- **FileStorage storage accounts**: Files-only storage accounts with premium performance characteristics. Recommended for enterprise or high performance scale applications.
- **Blob storage accounts**: Blob-only storage accounts. Use general-purpose v2 accounts instead when possible.

The following table describes the types of storage accounts and their capabilities:

Storage account type	Supported services	Supported performance tiers	Supported access tiers	Replication options	Deployment model [1]	En [2]

Storage account type	Supported services	Supported performance tiers	Supported access tiers	Replication options	Deployment model [1]	En [2]
General-purpose V2	Blob, File, Queue, Table, and Disk	Standard, Premium [5]	Hot, Cool, Archive [3]	LRS, GRS, RA-GRS, ZRS, ZGRS (preview), RA-ZGRS (preview) [4]	Resource Manager	En
General-purpose V1	Blob, File, Queue, Table, and Disk	Standard, Premium [5]	N/A	LRS, GRS, RA-GRS	Resource Manager, Classic	En
Block blob storage	Blob (block blobs and append blobs only)	Premium	N/A	LRS	Resource Manager	En
FileStorage	Files only	Premium	N/A	LRS	Resource Manager	En
Blob storage	Blob (block blobs and append blobs only)	Standard	Hot, Cool, Archive [3]	LRS, GRS, RA-GRS	Resource Manager	En

[1]Using the Azure Resource Manager deployment model is recommended. Storage accounts using the classic deployment model can still be created in some locations, and existing classic accounts continue to be supported. For more information, see Azure Resource Manager vs. classic deployment: Understand deployment models and the state of your resources.

[2]All storage accounts are encrypted using Storage Service Encryption (SSE) for data at rest. For more information, see Azure Storage Service Encryption for Data at Rest.

[3]The Archive tier is available at level of an individual blob only, not at the storage account level. Only block blobs and append blobs can be archived. For more information, see Azure Blob storage: Hot, Cool, and Archive storage tiers.

[4]Zone-redundant storage (ZRS) and geo-zone-redundant storage (GZRS) (preview) are available only for standard general-purpose v2 storage accounts. For more information about ZRS, see Zone-redundant storage (ZRS): Highly available Azure Storage applications. For more information about GZRS, see Geo-zone-redundant storage for highly availability and maximum durability (preview). For more information about other replication options, see Azure Storage replication.

[5]Premium performance for general-purpose v2 and general-purpose v1 accounts is available for disk and page blob only.

2)

Storage Types

Azure Storage provides four different storage services suited for varying application purposes.

Blob Storage

If your app needs to store unstructured object data, then blob storage is for you. Sometimes referred to as object storage, blob storage can be any form of text or binary data, such as a document or media file. *Unstructured* means that data isn't stored in columns and rows, as is the case in relational databases. Blob storage can be accessed from anywhere using REST APIs, but unlike file storage, can't be mounted using the SMB protocol.

File Storage

Intended mainly for legacy apps that you want to 'lift and shift' to the cloud, file storage allows files to be accessed from virtual machines using the standard SMB protocol. There's also a REST API so that on-premises apps can access data stored in shares.

Table Storage

This is where it gets interesting because while table storage might sound like it's just an SQL database, instead of storing data in rows and columns, data is stored in collections of individual documents, or as Microsoft refers to it, a *NoSQL key/attribute* data store. In other words, a JSON blob with no defined schema. The lack of structure enables more rapid development and faster access to data because unlike relational SQL databases, *NoSQL key/attribute* data stores don't require careful planning and can be adapted more easily as needs change.

Azure

Storage services (Image Credit: Microsoft)

Queue Storage

Designed for storing large numbers of messages that can be accessed from anywhere using authenticated HTTP or HTTPS calls, queue storage enables app components, which have been decoupled from one another so that they can scale independently, to communicate efficiently. Queue storage can handle traffic bursts so that servers aren't brought to their knees by a sudden rise in demand. And like the other types of storage it is accessible using REST APIs, and client libraries for .NET, Java, Android, C++, Node.js, PHP, Ruby, and Python are available.

Redundancy

Azure replicates data to ensure that it's always available in the event of a hardware failure. There are four different replication options available. Data within a region is replicated synchronously, and to secondary regions asynchronously in the background. For more information on synchronous and asynchronous replication, see Windows Server 2016: DFS-R vs. Storage Replica on *Petri*.

Locally Redundant Storage (LRS)

LRS ensures that your data stays within a single data center in your chosen region. Data is replicated three times. LRS is cheaper than the other types of redundancy and doesn't provide protection against data center failures.

Zone-Redundant Storage (ZRS)

Only available for block blobs, ZRS keeps three copies of your data across two or three data centers, either within your chosen region or across two regions.

Geo-Redundant Storage (GRS)

This is the type of redundancy that Microsoft recommends by default, and it keeps six copies of your data. Three copies stay in the primary region, and the remaining three are replicated to a secondary region.

Read-Access Geo-Redundant Storage (RA-GRS)

The default redundancy setting, RA-GRS replicates data to a secondary region, where apps also get read access to the data.

question

A **shared access signature** (**SAS**) is a URI that grants restricted access rights to **Azure** Storage resources. ... The URI query parameters comprising the **SAS** token incorporate all of the information necessary to grant controlled access to a storage resource

question

how many ways we can connect to azure storage account

q)Deployment slots in Azure

Deployment Slots are a feature of **Azure** App Service. They actually are live apps with their own hostnames. You can create different **slots** for your application (for e.g. Dev, Test or Stage). The Production **slot** is the **slot** where your live app resides

Azure DNS is a hosting service for **DNS** domains that provides name resolution by using Microsoft **Azure** infrastructure. By hosting your domains in **Azure**, you can manage your **DNS** records by using the same credentials, APIs, tools, and billing as your other **Azure** services

q)how to increase website performance without changing application service plan?

1. Enable HTTP/2

Microsoft announced support for HTTP/2 in App Services early in 2018. However, when you create a new App Service today, Azure will start you with HTTP 1.1 configured as the default protocol. HTTP/2 brings major changes to our favorite web protocol, and many of the changes aim to improve performance and reduce latency on the web. For example, header compression and binary formatting in HTTP/2 will reduce payload sizes.

2. Keep the App Service Always On

If you've deployed applications into IIS in the past, you'll know that IIS will unload idle websites after a period of inactivity. Azure App Services will also unload idle websites. Although the unloading can free up resources for other applications that might be running on the same App Service Plan, this strategy hurts the performance of the app because the next incoming request will wait as the web application starts from nothing. Web application startup time can be notoriously slow, regardless of the technologies involved. The caches are empty, the connection pools are empty, and all requests are slower than normal as the site needs to warms up.

To prevent the idle shutdown, you can set the Always On flag in the App Service Configuration blade.

sas in azure?

A shared access signature (**SAS**) is a URI that grants restricted access rights to **Azure** Storage resources. ... Additionally, with the account **SAS**, you can delegate access to operations that apply to a given service, such as Get/Set Service Properties and Get Service Stats

Grant limited access to Azure Storage resources using shared access signatures (SAS)

Types of shared access signatures

Azure Storage supports three types of shared access signatures:

- **User delegation SAS (preview).** A user delegation SAS is secured with Azure Active Directory (Azure AD) credentials and also by the permissions specified for the SAS. A user delegation SAS applies to Blob storage only. To create a user delegation SAS, you must first request a user delegation key, which is used to sign the SAS. For more information about the user delegation SAS, see Create a user delegation SAS (REST API).
- **Service SAS.** A service SAS is secured with the storage account key. A service SAS delegates access to a resource in only one of the Azure Storage services: Blob storage, Queue storage, Table storage, or Azure Files. For more information about the service SAS, see Create a service SAS (REST API).
- **Account SAS.** An account SAS is secured with the storage account key. An account SAS delegates access to resources in one or more of the storage services. All of the operations available via a service or user delegation SAS are also available via an account SAS. Additionally, with the account SAS, you can delegate access to operations that apply at the level of the service, such as **Get/Set Service Properties** and **Get Service Stats** operations. You can also delegate access to read, write, and delete operations on blob containers, tables, queues, and file shares that are not permitted with a service SAS. For more information about the account SAS, Create an account SAS (REST API).

Note

Microsoft recommends that you use Azure AD credentials when possible as a security best practice, rather than using the account key, which can be more easily compromised. When your application design requires shared access signatures for access to Blob storage, use Azure AD credentials to create a user delegation SAS when possible for superior security.

A shared access signature can take one of two forms:

- **Ad hoc SAS:** When you create an ad hoc SAS, the start time, expiry time, and permissions for the SAS are all specified in the SAS URI (or implied, if start time is omitted). Any type of SAS can be an ad hoc SAS.
- **Service SAS with stored access policy:** A stored access policy is defined on a resource container, which can be a blob container, table, queue, or file share. The stored access policy can be used to manage constraints for one or more service shared access signatures. When you associate a service SAS with a stored access policy, the SAS inherits the constraints—the start time, expiry time, and permissions—defined for the stored access policy.

q) how many ways to connect to storage account in azure?

Get started with Storage Explorer

Overview

Microsoft Azure Storage Explorer is a standalone app that enables you to easily work with Azure Storage data on Windows, macOS, and Linux. In this article, you learn several ways of connecting to and managing your Azure storage accounts.

Prerequisites

- Windows
- macOS
- Linux

Storage Explorer is supported on the following versions of Windows:

- Windows 10 (recommended)
- Windows 8
- Windows 7

For all versions of Windows, .NET Framework 4.6.2 or greater is required.

Download and install

Download and install Storage Explorer

Connect to a storage account or service

Storage Explorer provides several ways to connect to storage accounts. In general you can either:

- Sign in to Azure to access your subscriptions and their resources
- Attach a specific Storage or CosmosDB resource

Sign in to Azure

Note

To fully access resources after signing in, Storage Explorer requires both management (ARM) and data layer permissions. This means that you need Azure AD permissions which give you access to your Storage account, the containers in the account, and the data in the containers. If you only have permissions at the data layer, consider using **Attach with Azure AD**. For more information on the exact permissions Storage Explorer requires, see the **troubleshooting guide**.

1. In Storage Explorer, select **Manage Accounts** to go to the **Account Management Panel**.

2. The left pane now displays all the Azure accounts you've signed in to. To connect to another account, select **Add an account**

3. If you want to sign into a national cloud or an Azure Stack, click on the **Azure environment** dropdown to select which Azure cloud you want to use. Once you have chosen your environment, click the **Sign-in...** button. For more information, see Connect Storage Explorer to an Azure Stack subscription.

4. After you successfully sign in with an Azure account, the account and the Azure subscriptions associated with that account are added to the left pane. Select the Azure subscriptions that you want to work with, and then select **Apply** (Selecting **All subscriptions:** toggles selecting all or none of the listed Azure subscriptions).

The left pane displays the storage accounts associated with the selected Azure subscriptions.

Attach a specific resource

You can attach to a resource in Storage Explorer using different options:

- Add a resource via Azure AD: If you only have permissions at the data layer, then you can use this option to add a Blob container or an ADLS Gen2 Blob container.
- Use a connection string: If you have a connection string to a Storage Account. Storage Explorer supports both key and SAS connection strings.
- Use a SAS URI: If you have a SAS URI to a Blob Container, File Share, Queue, or Table. To get a SAS URI, you can either use Storage Explorer or the Azure portal.
- Use a name and key: If you know either of the account keys to your Storage Account, you can use this option to quickly connect. The keys for your Storage Account are located on the Storage Account **Access keys** panel at the Azure portal.
- Attach to a local emulator: If you're using one of the available Azure Storage emulators, use this option to easily connect to your emulator.
- Connect to an Azure Cosmos DB account by using a connection string: If you have a connection string to a CosmosDB instance.
- Connect to Azure Data Lake Store by URI: If you have a URI to an Azure Data Lake Store.

Add a resource via Azure AD

1. Open the **Connect Dialog** by clicking on the **connect button** on the left hand, vertical toolbar.

2. If you haven't already done so, use the **Add an Azure Account** option to sign in to the Azure Account that has access to the resource. After signing in return to the **Connect Dialog**.
3. Select the **Add a resource via Azure Active Directory (Azure AD)** option and click **Next**.
4. Select the Azure Account and tenant that has access to the Storage resource you want to attach to. Click **Next**.
5. Choose the resource type you want to attach, and then enter the information needed to connect. The inputs on this page will change depending on what type of resource you're

adding. Make sure to choose the correct type of resource. Once you've filled in the required information, click **Next**.

6. Review the connection summary and make sure all of the information is correct. If all of the information looks correct, click **Connect**. Otherwise, return to the previous pages with the **Back** button to correct any incorrect information.

Once the connection is successfully added, the resource tree will automatically navigate to the node representing the connection. You can also look under **Local & Attached** → **Storage Accounts** → **(Attached Containers)** → **Blob Containers** if for some reason it doesn't. If Storage Explorer was unable to add your connection, or if you can't access your data after successfully adding the connection, then consult the troubleshooting guide for help.

Use a connection string

1. Open the **Connect Dialog** by clicking on the **connect button** on the left hand, vertical toolbar.

2. Select the **Use a connection string** option and click **Next**.
3. Choose a display name for your connection and enter in your connection string. Then click **Next**.
4. Review the connection summary and make sure all of the information is correct. If all of the information looks correct, click **Connect**, otherwise return to the previous pages with the **Back** button to correct any incorrect information.

Once the connection is successfully added, the resource tree will automatically navigate to the node representing the connection. You can also look under **Local & Attached** → **Storage Accounts** if for some reason it doesn't. If Storage Explorer was unable to add your connection, or if you can't access your data after successfully adding the connection, then consult the troubleshooting guide for help.

Use a SAS URI

1. Open the **Connect Dialog** by clicking on the **connect button** on the left hand, vertical toolbar.

2. Select the **Use a shared access signature (SAS) URI** option and click **Next**.
3. Choose a display name for your connection and enter in your SAS URI. The service endpoint for the type of resource you're attaching should autofill. If you're using a custom endpoint, then it's possible it may not. Click **Next**.
4. Review the connection summary and make sure all of the information is correct. If all of the information looks correct, click **Connect**, otherwise return to the previous pages with the **Back** button to correct any wrong information.

Once the connection is successfully added, the resource tree will automatically navigate to the node representing the connection. You can also look under **Local & Attached** → **Storage Accounts** → **(Attached Containers)** → **the service node for the type of container you attached** if for some reason it doesn't. If Storage Explorer was unable to add your

connection, or if you can't access your data after successfully adding the connection, then consult the troubleshooting guide for help.

Use a name and key

1. Open the **Connect Dialog** by clicking on the **connect button** on the left hand, vertical toolbar.

2. Select the **Use a storage account name and key** option and click **Next**.
3. Choose a display name for your connection.
4. Enter in your Storage account name and either of its access keys.
5. Choose the **Storage domain** to use and then click **Next**.
6. Review the connection summary and make sure all of the information is correct. If all of the information looks correct, click **Connect**, otherwise return to the previous pages with the **Back** button to correct any incorrect information.

Once the connection is successfully added, the resource tree will automatically navigate to the node representing the connection. You can also look under **Local & Attached → Storage Accounts** if for some reason it doesn't. If Storage Explorer was unable to add your connection, or if you can't access your data after successfully adding the connection, then consult the troubleshooting guide for help.

Attach to a local emulator

Storage Explorer currently supports two official Storage emulators:

- Azure storage emulator (Windows only)
- Azurite (Windows, macOS, or Linux)

If your emulator is listening on the default ports, you can use the **Emulator - Default Ports** node (found under **Local & Attached → Storage Accounts**) to quickly access your emulator.

If you want to use a different name for your connection, or if your emulator isn't running on the default ports:

1. Start your emulator. When you do, make a note of what ports the emulator is listening on for each service type.

Important

Storage Explorer does not automatically start your emulator. You must start it yourself.

2. Open the **Connect Dialog** by clicking on the **connect button** on the left hand, vertical toolbar.

3. Select the **Attach to a local emulator** option and click **Next**.
4. Choose a display name for your connection and enter in the ports your emulator is listening on for each service type. The text boxes will start with the default port values for most emulators. The **Files port** is left blank, because neither of the official emulators currently support the Files service. If the emulator you're using does support Files, then you can enter the port that is being used. Click **Next**.

5. Review the connection summary and make sure all of the information is correct. If all of the information looks correct, then click **Connect**, otherwise return to the previous pages with the **Back** button to correct any wrong information.

Once the connection is successfully added, the resource tree will automatically navigate to the node representing the connection. You can also look under **Local & Attached** → **Storage Accounts** if for some reason it doesn't. If Storage Explorer was unable to add your connection, or if you can't access your data after successfully adding the connection, then consult the troubleshooting guide for help.

Connect to an Azure Cosmos DB account by using a connection string

Besides manage Azure Cosmos DB accounts through Azure subscription, an alternative way of connecting to an Azure Cosmos DB is to use a connection string. Use the following steps to connect using a connection string.

1. Find **Local and Attached** in the left tree, right-click **Azure Cosmos DB Accounts**, choose **Connect to Azure Cosmos DB...**

2. Choose Azure Cosmos DB API, paste your **Connection String**, and then click **OK** to connect Azure Cosmos DB account. For information on retrieving the connection string, see Get the connection string.

Connect to Azure Data Lake Store by URI

If you need access to a resource not in your subscription, you'll need someone with access to give you the resource URI. After signing in, you can connect to Data Lake Store using the URI by following these steps:

1. Open Storage Explorer.
2. In the left pane, expand **Local and Attached**.
3. Right-click **Data Lake Store**, and - from the context menu - select **Connect to Data Lake Store...**.

4. Enter the Uri, then the tool navigates to the location of the URL you just entered.

Generate a SAS in Storage Explorer

Account level SAS

1. Right-click the storage account you want share, and then select **Get Shared Access Signature...**.

2. In the **Generate Shared Access Signature** dialog box, specify the time frame and permissions that you want for the account. Click **Create**.

3. You can now either copy the **Connection string** or the raw **Query string** to your clipboard.

Service level SAS

How to get a SAS for a blob container in Storage Explorer

Search for storage accounts

If you need to find a storage resource, and don't know where it is, you can use the search box at the top of the left pane to search for the resource.

As you type in the search box, the left pane displays all resources that match the search value you've entered up to that point. For example, a search for **endpoints** is shown in the following screenshot:

Note

Use the **Account Management Panel** to deselect any subscriptions that do not contain the item you are searching for to improve the execution time of your search. You can also right-click on a node and choose **Search From Here** to start searching from a specific node.

q) what is the use of deployment slots in azure?

Deployment Slots are a feature of Azure App Service. They actually are live apps with their own hostnames. You can create different slots for your application (for e.g. Dev, Test or Stage). The Production slot is the slot where your live app resides. With deploymet slots, you can validate app changes in staging before swapping it with your production slot

Pre-Requisites

- Microsoft Azure Subscription (Sign up for free/ "Create your Azure free account today"))
- Microsoft Azure CLI (Install from here)

#Log in to Azure

Before executing any Azure CLI commands, you will need to login first.

- Open **Command Prompt** or a **Powershell** session
- Enter following command:

az login

The command will prompt you to log in with an authentication code via a website.

#Listing Deployment Slots

To list **deployment slots** in an **Azure App Service**, execute the following command:

az webapp deployment slot list -n "web app name" -g "resource group name"

#Creating Deployment Slot

To create a **new deployment slot** in an Azure App Service, execute the following command:

az webapp deployment slot create -n "web app name" -g "resource group name" -s "deployment slot name"

#Swapping Deployment Slot

To **swap a deployment slot** in an Azure App Service, execute the following command:

az webapp deployment slot swap -n "web app name" -g "resource group name" -s "source slot name" --target-slot "target slot"

#Deleting a Deployment Slot

To **delete a deployment slot** in an Azpp Service, execute the following command:

az webapp deployment slot create -n "web app name" -g "resource group name" -s "deployment slot name"

q)what is the use of custom domain name in azure?

Azure DNS provides DNS for a **custom domain** for any of your **Azure** resources that support **custom domains** or that have a fully qualified **domain name** (FQDN). An **example** is you have an **Azure** web app and you want your users to access it by either using contoso.com, or www.contoso.com as an FQDN.

Use Azure DNS to provide custom domain settings for an Azure service

- 07/13/2019
- 6 minutes to read
-
  -
  -
  -
  -
  -
  - +3

Azure DNS provides DNS for a custom domain for any of your Azure resources that support custom domains or that have a fully qualified domain name (FQDN). An example is you have an Azure web app and you want your users to access it by either using contoso.com, or www.contoso.com as an FQDN. This article walks you through configuring your Azure service with Azure DNS for using custom domains.

Prerequisites

In order to use Azure DNS for your custom domain, you must first delegate your domain to Azure DNS. Visit Delegate a domain to Azure DNS for instructions on how to configure your name servers for delegation. Once your domain is delegated to your Azure DNS zone, you are able to configure the DNS records needed.

You can configure a vanity or custom domain for Azure Function Apps, Public IP addresses, App Service (Web Apps), Blob storage, and Azure CDN.

Azure Function App

To configure a custom domain for Azure function apps, a CNAME record is created as well as configuration on the function app itself.

Navigate to **Function App** and select your function app. Click **Platform features** and under **Networking** click **Custom domains**.

Note the current url on the **Custom domains** blade, this address is used as the alias for the DNS record created.

Navigate to your DNS Zone and click **+ Record set**. Fill out the following information on the **Add record set** blade and click **OK** to create it.

Property	**Value**	**Description**
Name	myfunctionapp	This value along with the domain name label is the FQDN for the custom domain name.
Type	CNAME	Use a CNAME record is using an alias.
TTL	1	1 is used for 1 hour
TTL unit	Hours	Hours are used as the time measurement
Alias	adatumfunction.azurewebsites.net	The DNS name you are creating the alias for, in this example it is the adatumfunction.azurewebsites. net DNS name provided by default to the function app.

Navigate back to your function app, click **Platform features**, and under **Networking** click **Custom domains**, then under **Custom Hostnames** click **+ Add hostname**.

On the **Add hostname** blade, enter the CNAME record in the **hostname** text field and click **Validate**. If the record is found, the **Add hostname** button appears. Click **Add hostname** to add the alias.

Public IP address

To configure a custom domain for services that use a public IP address resource such as Application Gateway, Load Balancer, Cloud Service, Resource Manager VMs, and, Classic VMs, an A record is used.

Navigate to **Networking** > **Public IP address**, select the Public IP resource and click **Configuration**. Notate the IP address shown.

Navigate to your DNS Zone and click **+ Record set**. Fill out the following information on the **Add record set** blade and click **OK** to create it.

Property	**Value**	**Description**
Name	mywebserver	This value along with the domain name label is the FQDN for the custom domain name.
Type	A	Use an A record as the resource is an IP address.
TTL	1	1 is used for 1 hour
TTL unit	Hours	Hours are used as the time measurement
IP Address	<your ip address>	The public IP address.

Once the A record is created, run nslookup to validate the record resolves.

App Service (Web Apps)

The following steps take you through configuring a custom domain for an app service web app.

Navigate to **App Service** and select the resource you are configuring a custom domain name, and click **Custom domains**.

Note the current url on the **Custom domains** blade, this address is used as the alias for the DNS record created.

Navigate to your DNS Zone and click **+ Record set**. Fill out the following information on the **Add record set** blade and click **OK** to create it.

Property	**Value**	**Description**

Property	Value	Description
Name	mywebserver	This value along with the domain name label is the FQDN for the custom domain name.
Type	CNAME	Use a CNAME record is using an alias. If the resource used an IP address, an A record would be used.
TTL	1	1 is used for 1 hour
TTL unit	Hours	Hours are used as the time measurement
Alias	webserver.azurewebsites.net	The DNS name you are creating the alias for, in this example it is the webserver.azurewebsites.net DNS name provided by default to the web app.

Navigate back to the app service that is configured for the custom domain name. Click **Custom domains**, then click **Hostnames**. To add the CNAME record you created, click **+ Add hostname**.

Once the process is complete, run **nslookup** to validate name resolution is working.

To learn more about mapping a custom domain to App Service, visit Map an existing custom DNS name to Azure Web Apps.

To learn how to migrate an active DNS name, see Migrate an active DNS name to Azure App Service.

If you need to purchase a custom domain, visit Buy a custom domain name for Azure Web Apps to learn more about App Service domains.

Blob storage

The following steps take you through configuring a CNAME record for a blob storage account using the asverify method. This method ensures there is no downtime.

Navigate to **Storage** > **Storage Accounts**, select your storage account, and click **Custom domain**. Notate the FQDN under step 2, this value is used to create the first CNAME record

Navigate to your DNS Zone and click **+ Record set**. Fill out the following information on the **Add record set** blade and click **OK** to create it.

Property	Value	Description
Name	asverify.mystorageaccount	This value along with the dom FQDN for the custom doma
Type	CNAME	Use a CNAME record is using a
TTL	1	1 is used for 1 hour
TTL unit	Hours	Hours are used as the time me
Alias	asverify.adatumfunctiona9ed.blob.core.windows.net	The DNS name you are creati example it asverify.adatumfunctiona9 net DNS name provided by account.

Navigate back to your storage account by clicking **Storage** > **Storage Accounts**, select your storage account and click **Custom domain**. Type in the alias you created without the asverify prefix in the text box, check **Use indirect CNAME validation, and click **Save**. Once this step is complete, return to your DNS zone and create a CNAME record without the asverify prefix. After that point, you are safe to delete the CNAME record with the cdnverify prefix.

Validate DNS resolution by running nslookup

To learn more about mapping a custom domain to a blob storage endpoint visit Configure a custom domain name for your Blob storage endpoint

Azure CDN

The following steps take you through configuring a CNAME record for a CDN endpoint using the cdnverify method. This method ensures there is no downtime.

Navigate to **Networking** > **CDN Profiles**, select your CDN profile.

Select the endpoint you are working with and click **+ Custom domain**. Note the **Endpoint hostname** as this value is the record that the CNAME record points to.

Navigate to your DNS Zone and click **+ Record set**. Fill out the following information on the **Add record set** blade and click **OK** to create it.

Property	Value	Description
Name	cdnverify.mycdnendpoint	This value along with the domain name label is the FQDN for the custom domain name.
Type	CNAME	Use a CNAME record is using an alias.
TTL	1	1 is used for 1 hour
TTL unit	Hours	Hours are used as the time measurement
Alias	cdnverify.adatumcdnendpoint.azureedge.net	The DNS name you are creating the alias for, in this example it is the cdnverify.adatumcdnendpoint.azureedge.net DNS name provided by default to the storage account.

Navigate back to your CDN endpoint by clicking **Networking** > **CDN Profiles**, and select your CDN profile. Click **+ Custom domain** and enter your CNAME record alias without the cdnverify prefix and click **Add**.

Once this step is complete, return to your DNS zone and create a CNAME record without the cdnverify prefix. After that point, you are safe to delete the CNAME record with the cdnverify prefix. For more information on CDN and how to configure a custom domain without the intermediate registration step visit Map Azure CDN content to a custom domain.

q) how many ways we can enable login for website in azure?
No answer

q) what is the use of kudos azure?
Use Azure AD to manage user access and enable single sign-on with Kudos. Requires an existing Kudos subscription.

* Enterprise Single Sign-On - Azure Active Directory supports rich enterprise-class single sign-on with Kudos out of the box. Users sign in using their organizational accounts hosted in Active Directory.

* Easy Configuration - Azure Active Directory provides a simple step-by-step user interface for connecting Kudos to Azure AD.

Tutorial: Azure Active Directory integration with Kudos

- 03/26/2019
- 6 minutes to read
-
  -
  -
  -
  -
  -
  - +4

In this tutorial, you learn how to integrate Kudos with Azure Active Directory (Azure AD). Integrating Kudos with Azure AD provides you with the following benefits:

- You can control in Azure AD who has access to Kudos.
- You can enable your users to be automatically signed-in to Kudos (Single Sign-On) with their Azure AD accounts.
- You can manage your accounts in one central location - the Azure portal.

If you want to know more details about SaaS app integration with Azure AD, see What is application access and single sign-on with Azure Active Directory. If you don't have an Azure subscription, create a free account before you begin.

Prerequisites

To configure Azure AD integration with Kudos, you need the following items:

- An Azure AD subscription. If you don't have an Azure AD environment, you can get a free account
- Kudos single sign-on enabled subscription

Scenario description

In this tutorial, you configure and test Azure AD single sign-on in a test environment.

- Kudos supports **SP** initiated SSO

Adding Kudos from the gallery

To configure the integration of Kudos into Azure AD, you need to add Kudos from the gallery to your list of managed SaaS apps.

To add Kudos from the gallery, perform the following steps:

1. In the **Azure portal**, on the left navigation panel, click **Azure Active Directory** icon.

2. Navigate to **Enterprise Applications** and then select the **All Applications** option.

3. To add new application, click **New application** button on the top of dialog.

4. In the search box, type **Kudos**, select **Kudos** from result panel then click **Add** button to add the application.

Configure and test Azure AD single sign-on

In this section, you configure and test Azure AD single sign-on with Kudos based on a test user called **Britta Simon**. For single sign-on to work, a link relationship between an Azure AD user and the related user in Kudos needs to be established.

To configure and test Azure AD single sign-on with Kudos, you need to complete the following building blocks:

1. **Configure Azure AD Single Sign-On** - to enable your users to use this feature.
2. **Configure Kudos Single Sign-On** - to configure the Single Sign-On settings on application side.
3. **Create an Azure AD test user** - to test Azure AD single sign-on with Britta Simon.
4. **Assign the Azure AD test user** - to enable Britta Simon to use Azure AD single sign-on.
5. **Create Kudos test user** - to have a counterpart of Britta Simon in Kudos that is linked to the Azure AD representation of user.
6. **Test single sign-on** - to verify whether the configuration works.

Configure Azure AD single sign-on

In this section, you enable Azure AD single sign-on in the Azure portal.

To configure Azure AD single sign-on with Kudos, perform the following steps:

1. In the Azure portal, on the **Kudos** application integration page, select **Single sign-on**.

2. On the **Select a Single sign-on method** dialog, select **SAML/WS-Fed** mode to enable single sign-on.

3. On the **Set up Single Sign-On with SAML** page, click **Edit** icon to open **Basic SAML Configuration** dialog.

4. On the **Basic SAML Configuration** section, perform the following steps:

In the **Sign-on URL** text box, type a URL using the following pattern: https://<company>.kudosnow.com

Note

The value is not real. Update the value with the actual Sign-On URL. Contact **Kudos Client support team** to get the value. You can also refer to the patterns shown in the **Basic SAML Configuration** section in the Azure portal.

5. On the **Set up Single Sign-On with SAML** page, in the **SAML Signing Certificate** section, click **Download** to download the **Certificate (Base64)** from the given options as per your requirement and save it on your computer.

6. On the **Set up Kudos** section, copy the appropriate URL(s) as per your requirement.

a. Login URL
b. Azure AD Identifier
c. Logout URL

Configure Kudos Single Sign-On

1. In a different web browser window, sign into your Kudos company site as an administrator.
2. In the menu on the top, click **Settings icon**.

3. Click **Integrations > SSO** and perform the following steps:

a. In **Sign on URL** textbox, paste the value of **Login URL** which you have copied from Azure portal.
b. Open your base-64 encoded certificate in notepad, copy the content of it into your clipboard, and then paste it to the **X.509 certificate** textbox
c. In **Logout To URL** textbox, paste the value of **Logout URL** which you have copied from Azure portal.
d. In the **Your Kudos URL** textbox, type your company name.
e. Click **Save**.

Create an Azure AD test user

The objective of this section is to create a test user in the Azure portal called Britta Simon.

1. In the Azure portal, in the left pane, select **Azure Active Directory**, select **Users**, and then select **All users**.

2. Select **New user** at the top of the screen.

3. In the User properties, perform the following steps.

a. In the **Name** field enter **BrittaSimon**.

b. In the **User name** field type brittasimon@yourcompanydomain.extension For example, BrittaSimon@contoso.com

c. Select **Show password** check box, and then write down the value that's displayed in the Password box.

d. Click **Create**.

Assign the Azure AD test user

In this section, you enable Britta Simon to use Azure single sign-on by granting access to Kudos.

1. In the Azure portal, select **Enterprise Applications**, select **All applications**, then select **Kudos**.

2. In the applications list, select **Kudos**.

3. In the menu on the left, select **Users and groups**.

4. Click the **Add user** button, then select **Users and groups** in the **Add Assignment** dialog.

5. In the **Users and groups** dialog select **Britta Simon** in the Users list, then click the **Select** button at the bottom of the screen.
6. If you are expecting any role value in the SAML assertion then in the **Select Role** dialog select the appropriate role for the user from the list, then click the **Select** button at the bottom of the screen.
7. In the **Add Assignment** dialog click the **Assign** button.

Create Kudos test user

In order to enable Azure AD users to sign in to Kudos, they must be provisioned into Kudos. In the case of Kudos, provisioning is a manual task.

To provision a user account, perform the following steps:

1. Sign in to your **Kudos** company site as administrator.
2. In the menu on the top, click **Settings icon**.

3. Click **User Admin**.
4. Click the **Users** tab, and then click **Add a User**.

5. In the **Add a User** section, perform the following steps:

a. Type the **First Name**, **Last Name**, **Email** and other details of a valid Azure Active Directory account you want to provision into the related textboxes.

b. Click **Create User**.

Note

You can use any other Kudos user account creation tools or APIs provided by Kudos to provision AAD user accounts.

Test single sign-on

In this section, you test your Azure AD single sign-on configuration using the Access Panel.

When you click the Kudos tile in the Access Panel, you should be automatically signed in to the Kudos for which you set up SSO. For more information about the Access Panel, see Introduction to the Access Panel.

q) what is the use of availability option in application insight?

General availability of Azure Application Insights

Posted on 16 November, 2016

Shiva Sivakumar Director of Progam Management, Azure Monitoring & Diagnostics

Today at the Connect() 2016 event in New York, we announced the general availability of Azure Application Insights (previously Visual Studio Application Insights) and launched our new pricing structure. With this announcement, Application Insights now provides a financially backed SLA offering 99.9% availability.

Application Insights is an integrated application performance management (APM) and application analytics solution. It enables development teams to understand how application performance relates to user experience and how these impact business outcomes.

If you are new to Application Insights, here is a quick overview and demo:

The main areas of Application Insights are:

- **Intelligent APM**: Proactively monitor and improve the performance of the application you're developing with advanced tools. Visual application maps pinpoint performance issues. Smart detection based on machine learning sends you alerts with embedded diagnostics. With Live Metrics Stream you can monitor your application health metrics in real time, while you're deploying a change.
- **Analytics**: With its rich query language, Analytics gives you answers to complex questions about your application's performance and usage, almost instantly. Ask creative questions about the performance and behavior of your apps in flexible ways with interactive queries,

and refine them until you pin-point the problem that impedes a desired business outcome. Once you derive an insight from the ad-hoc queries, you can share them in the form of visuals across your organization in customizable dashboards or through integration with Power BI.

- **DevOps Integration &Extensibility**: Tightly integrated into the Visual Studio product family. Read performance data right there in the code of your app in Visual Studio IDE. Integrations with Visual Studio Team Services, Team Foundation Server, and GitHub enable you to find and fix quality issues early in your DevOps workflows. Integrations with System Center and Operations Management Suite enable you to share application performance and system performance metrics across team boundaries and shorten the time to find root causes of issues.

As part of being generally available, we have introduced a new pricing structure. You can still start for free with Application Insights, and there is no limitation on the APM and Analytics tools – you get the full feature set without cost. You only pay as your app grows and as you transmit more application telemetry to Application Insights, but you control how much you pay!

In addition, we are announcing these new features and enhancements to Application Insights, that we are proud to share these with you today.

1. Increased raw data retention to 90 days for Analytics Queries
2. European data center option for storing Application Insights data
3. Improvements in Application Performance Management:

- Smart detection of degradation in request performance
- Correlating Availability Monitoring results with server side telemetry which will let you diagnose failures of your synthetic tests
- Failure Samples in Live Metrics Stream which will let you get insight into details of failed requests, dependency calls and exceptions in real time
- Grid control in Azure Dashboards and additional charting options in Metric options such as percentage charts

4. Enhancements to the Codelens and Application Search capabilities inside Visual Studio provide more information in context to identify and fix issues sooner
5. Preview of Application Insights REST APIs to access all your queries, events, and metrics data

We are continuously adding new capabilities to Application Insights and learning from our customers to better address your needs. We are fully committed to Azure Privacy standards and Security & Compliance policies, so that your data remains safe.

I would like to conclude with some inspiring words from one of our customers, AkzoNobel, a global paint firm, with whom we recently published a case study:

"With Application Insights, our attention has been directed multiple times to issues that would otherwise have taken much longer to detect. As a result, we've been able to maintain the service levels required for our application."

--Rob Reijers, Manager ColorApps Development, AkzoNobel

What is Application Insights?

- 06/03/2019
- 5 minutes to read
-

Application Insights is an extensible Application Performance Management (APM) service for web developers on multiple platforms. Use it to monitor your live web application. It will automatically detect performance anomalies. It includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. It's designed to help you continuously improve performance and usability. It works for apps on a wide variety of platforms including .NET, Node.js and Java EE, hosted on-premises, hybrid, or any public cloud. It integrates with your DevOps process, and has connection points to a variety of development tools. It can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center.

How does Application Insights work?

You install a small instrumentation package in your application, and set up an Application Insights resource in the Microsoft Azure portal. The instrumentation monitors your app and sends telemetry data to Azure Monitor. (The application can run anywhere - it doesn't have to be hosted in Azure.)

You can instrument not only the web service application, but also any background components, and the JavaScript in the web pages themselves.

In addition, you can pull in telemetry from the host environments such as performance counters, Azure diagnostics, or Docker logs. 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 analytic and search tools to the raw data.

What's the overhead?

The impact on your app's performance is very small. Tracking calls are non-blocking, and are batched and sent in a separate thread.

What does Application Insights monitor?

Application Insights is aimed at the development team, to help you understand how your app is performing and how it's being used. It monitors:

- **Request rates, response times, and failure rates** - Find out which pages are most popular, at what times of day, and where your users are. See which pages perform best. If your response times and failure rates go high when there are more requests, then perhaps you have a resourcing problem.
- **Dependency rates, response times, and failure rates** - Find out whether external services are slowing you down.
- **Exceptions** - Analyze the aggregated statistics, or pick specific instances and drill into the stack trace and related requests. Both server and browser exceptions are reported.

- **Page views and load performance** - reported by your users' browsers.
- **AJAX calls** from web pages - rates, response times, and failure rates.
- **User and session counts**.
- **Performance counters** from your Windows or Linux server machines, such as CPU, memory, and network usage.
- **Host diagnostics** from Docker or Azure.
- **Diagnostic trace logs** from your app - so that you can correlate trace events with requests.
- **Custom events and metrics** that you write yourself in the client or server code, to track business events such as items sold or games won.

Where do I see my telemetry?

There are plenty of ways to explore your data. Check out these articles:

Smart detection and manual alerts Automatic alerts adapt to your app's normal patterns of telemetry and trigger when there's something outside the usual pattern. You can also set alerts on particular levels of custom or standard metrics.	
Application map The components of your app, with key metrics and alerts.	
Profiler Inspect the execution profiles of sampled requests.	
Usage analysis Analyze user segmentation and retention.	
Diagnostic search for instance data Search and filter events such as requests, exceptions, dependency calls, log traces, and page views.	
Metrics Explorer for aggregated data Explore, filter, and segment aggregated data such as rates of requests, failures, and exceptions; response times, page load times.	
Dashboards Mash up data from multiple resources and share with others. Great for multi-component applications, and for continuous display in the team room.	
Live Metrics Stream	

When you deploy a new build, watch these near-real-time performance indicators to make sure everything works as expected.	
Analytics Answer tough questions about your app's performance and usage by using this powerful query language.	
Visual Studio See performance data in the code. Go to code from stack traces.	
Snapshot debugger Debug snapshots sampled from live operations, with parameter values.	
Power BI Integrate usage metrics with other business intelligence.	
REST API Write code to run queries over your metrics and raw data.	
Continuous export Bulk export of raw data to storage as soon as it arrives.	

How do I use Application Insights?

Monitor

Install Application Insights in your app, set up availability web tests, and:

- Check-out the default application dashboard for your team room to keep an eye on load, responsiveness, and the performance of your dependencies, page loads, and AJAX calls.
- Discover which are the slowest and most failing requests.
- Watch Live Stream when you deploy a new release, to know immediately about any degradation.

Detect, Diagnose

When you receive an alert or discover a problem:

- Assess how many users are affected.
- Correlate failures with exceptions, dependency calls, and traces.
- Examine profiler, snapshots, stack dumps, and trace logs.

Build, Measure, Learn

Measure the effectiveness of each new feature that you deploy.

- Plan to measure how customers use new UX or business features.
- Write custom telemetry into your code.

- Base the next development cycle on hard evidence from your telemetry.

q) what is the use of kusto query in app insight?

A **Kusto query** is a read-only request to process data and return results. The request is stated in plain text, **using a** data-flow model designed to make the syntax easy to read, author, and automate.

Overview

- 03/07/2019
- 2 minutes to read
-
○

○

A Kusto query is a read-only request to process data and return results. The request is stated in plain text, using a data-flow model designed to make the syntax easy to read, author, and automate. The query uses schema entities that are organized in a hierarchy similar to SQL's: databases, tables, and columns.

The query consists of a sequence of query statements, delimited by a semicolon (;), with at least one statement being a tabular expression statement which is a statement that produces data arranged in a table-like mesh of columns and rows. The query's tabular expression statements produce the results of the query.

The syntax of the tabular expression statement has tabular data flow from one tabular query operator to another, starting with data source (e.g. a table in a database, or an operator that produces data) and then flowing through a set of data transformation operators that are bound together through the use of the pipe (|) delimiter.

For example, the following Kusto query has a single statement, which is a tabular expression statement. The statement starts with a reference to a table called StormEvents (the database that host this table is implicit here, and part of the connection information). The data (rows) for that table are then filtered by the value of the StartTime column, and then filtered by the value of the State column. The query then returns the count of "surviving" rows.

KustoCopy

```
StormEvents
| where StartTime >= datetime(2007-11-01) and StartTime < datetime(2007-12-01)
| where State == "FLORIDA"
| count
```

To run this query, click here. In this case, the result will be:

Count
23

q) vm creation in azure step by step?
Microsoft Azure - Deploying Virtual Machines

Advertisements
Previous Page
Next Page

A quick process of creating a virtual machine was included in the chapter 'Compute Module'. This chapter contains the detailed process including how to configure virtual machines.

Quick Create

Step 1 − Login to Azure Management Portal.

Step 2 − Locate and click on 'Virtual Machines' in the left panel and then click on 'Create a Virtual Machine'.

Step 3 − Alternatively, click 'New' at the bottom left corner and then click 'Compute' → 'Virtual Machine' →'Quick Create'.

Step 4 − Enter DNS name. This has to be unique. The DNS name is used to connect to the virtual machine.

Step 5 − Select the image and size from the dropdown list. The size affects the cost of running virtual machine.

Step 6 − Enter username and password. You must remember to log in to the virtual machine later.

Step 7 − Select the relevant region.

Step 8 − Click on 'Create a virtual machine' and you are ready to use your new machine. It will take a few seconds for the machine to be created.

Create Virtual Machine with Advanced Settings

Step 1 − Choose 'Custom Create' instead of 'Quick Create' in the options and you will be taken to the following screen.

Step 2 − Choose an image from the list. In this screen, you find that choosing an image is easier based on their category shown on the left side. Let us create a virtual machine for SQL Server for which we have chosen SQL Server on the left side and all the software in this category are shown in the middle.

Step 3 − Click on the Next arrow.

Step 4 − Choose Version Release Date and enter the VM's name.

Step 5 − Select the Tier. The size dropdown would change items according to tier. In the basic version, you will get only first 5 options, while in the standard version you will get more options. It should be according to you and you image's requirements. For example, in this case let's choose SQL server. It requires minimum A4 machine with 8 cores and 14GB memory.

Step 6 − Enter the username and password and click Next arrow.

Step 7 − Enter DNS name which should be unique as mentioned earlier and select the region.

Under the storage account, it will display the storage accounts that you have already created. As seen in the following screen, an account name is shown in the dropdown which is a storage account created earlier. You can choose an already created account or even use an automatically generated account.

Step 8 − Next is Availability set. This option lets you create a set of virtual machines that will ensure that if a single point fails, it doesn't affect your machine and keeps the work going on. Let's choose the option 'none' here.

The last option is End Points. End points are used to communicate with virtual machines by other resources you can leave. In a subsequent chapter, we will provide a detailed illustration to configure endpoints.

Step 9 − Click on Next and the virtual machine will be created in a few seconds for you.

Connecting with a Virtual Network

Step 1 − Create a virtual machine using the steps described earlier. If you already have a virtual network created in Azure, it will be diplayed in the highlighted dropdown list as shown in the following screen. You can choose the network as shown in following picture.

Step 2 − When you go to your Virtual Network and management portal created earlier, click on 'Dashboard'. The virtual machine will be displyed in the resources of that network as shown in the following picture.

q)What is VNet and subnet?

A **subnet** is a range of IP addresses in the **VNet**. You can divide a **VNet** into multiple **subnets** for organization and security. Each NIC in a VM is connected to one **subnet** in one **VNet**. NICs connected to **subnets** (same or different) within a **VNet** can communicate with each other without any extra configuration.

What is VNet and subnet in Azure?

Subnets. A **subnet** is a range of IP addresses in the **VNet**, you can divide a **VNet** into multiple **subnets** for organization and security. Additionally you can configure **VNet** routing tables and Network Security Groups (NSG) to a **subnet**

What is CIDR notation?

Classless inter-domain routing (**CIDR**) is a set of Internet protocol (IP) standards that is used to create unique identifiers for networks and individual devices. ... That system is known as **CIDR notation**. **CIDR** IP addresses consist of two groups of numbers, which are also referred to as groups of bits.

q)Classless Inter-Domain Routing (CIDR) Chart

The Classless Inter-Domain Routing (CIDR) is commonly known as the CIDR chart and is used by those running networks and managing IP addresses. It enables them to see the number of IP addresses contained within each "slash notation" and the size of each "slash notation" in bits.

Determining the network prefixEdit

An IPv4 subnet mask consists of 32 bits; it is a sequence of ones (*1*) followed by a block of zeros (*0*). The ones indicate bits in the address used for the network prefix and the trailing block of zeros designates that part as being the host identifier.

The following example shows the separation of the network prefix and the host identifier from an address (*192.0.2.130*) and its associated */24* subnet mask (*255.255.255.0*). The operation is visualized in a table using binary address formats.

	Binary form	**Dot-decimal notation**
IP address	11000000.00000000.00000010.10000010	192.0.2.130
Subnet mask	11111111.11111111.11111111.00000000	255.255.255.0
Network prefix	11000000.00000000.00000010.00000000	192.0.2.0
Host identifier	00000000.00000000.00000000.10000010	0.0.0.130

The result of the bitwise AND operation of IP address and the subnet mask is the network prefix *192.0.2.0*. The host part, which is *130*, is derived by the bitwise AND operation of the address and the one's complement of the subnet mask.

Configure a Point-to-Site connection by using certificate authentication (classic)

- 12/11/2018
- 20 minutes to read
-
  ○

○

-
-
-
- +5

Note

This article is written for the classic deployment model. If you're new to Azure, we recommend that you use the Resource Manager deployment model instead. The Resource Manager deployment model is the most current deployment model and offers more options and feature compatibility than the classic deployment model. For more information about the deployment models, see **Understanding deployment models**.

For the Resource Manager version of this article, select it from the drop-down list below, or from the table of contents on the left.

This article shows you how to create a VNet with a Point-to-Site connection. You create this Vnet with the classic deployment model by using the Azure portal. This configuration uses certificates to authenticate the connecting client, either self-signed or CA issued. You can also create this configuration with a different deployment tool or model by using options that are described in the following articles:

Azure portal (classic)

You use a Point-to-Site (P2S) VPN gateway to create a secure connection to your virtual network from an individual client computer. Point-to-Site VPN connections are useful when you want to connect to your VNet from a remote location. When you have only a few clients that need to connect to a VNet, a P2S VPN is a useful solution to use instead of a Site-to-Site VPN. A P2S VPN connection is established by starting it from the client computer.

Important

The classic deployment model supports Windows VPN clients only and uses the Secure Socket Tunneling Protocol (SSTP), an SSL-based VPN protocol. To support non-Windows VPN clients, you must create your VNet with the Resource Manager deployment model. The Resource Manager deployment model supports IKEv2 VPN in addition to SSTP. For more information, see **About P2S connections**.

Prerequisites

Point-to-Site certificate authentication connections require the following prerequisites:

- A Dynamic VPN gateway.

- The public key (.cer file) for a root certificate, which is uploaded to Azure. This key is considered a trusted certificate and is used for authentication.
- A client certificate generated from the root certificate, and installed on each client computer that will connect. This certificate is used for client authentication.
- A VPN client configuration package must be generated and installed on every client computer that connects. The client configuration package configures the native VPN client that's already on the operating system with the necessary information to connect to the VNet.

Point-to-Site connections don't require a VPN device or an on-premises public-facing IP address. The VPN connection is created over SSTP (Secure Socket Tunneling Protocol). On the server side, we support SSTP versions 1.0, 1.1, and 1.2. The client decides which version to use. For Windows 8.1 and above, SSTP uses 1.2 by default.

For more information about Point-to-Site connections, see Point-to-Site FAQ.

Example settings

Use the following values to create a test environment, or refer to these values to better understand the examples in this article:

- **Create virtual network (classic) settings**
  - **Name**: Enter *VNet1*.
  - **Address space**: Enter *192.168.0.0/16*. For this example, we use only one address space. You can have more than one address space for your VNet, as shown in the diagram.
  - **Subnet name**: Enter *FrontEnd*.
  - **Subnet address range**: Enter *192.168.1.0/24*.
  - **Subscription**: Select a subscription from the list of available subscriptions.
  - **Resource group**: Enter *TestRG*. Select **Create new**, if the resource group doesn't exist.
  - **Location**: Select **East US** from the list.
  - **VPN connection settings**
    - **Connection type**: Select **Point-to-site**.
    - **Client Address Space**: Enter *172.16.201.0/24*. VPN clients that connect to the VNet by using this Point-to-Site connection receive an IP address from the specified pool.
- **Gateway configuration subnet settings**
  - **Name**: Autofilled with *GatewaySubnet*.
  - **Address range**: Enter *192.168.200.0/24*.
- **Gateway configuration settings**:
  - **Size**: Select the gateway SKU that you want to use.
  - **Routing Type**: Select **Dynamic**.

Create a virtual network and a VPN gateway

Before you begin, verify that you have an Azure subscription. If you don't already have an Azure subscription, you can activate your MSDN subscriber benefits or sign up for a free account.

Part 1: Create a virtual network

If you don't already have a virtual network (VNet), create one. Screenshots are provided as examples. Be sure to replace the values with your own. To create a VNet by using the Azure portal, use the following steps:

1. Sign in to the Azure portal and select **Create a resource**. The **New** page opens.
2. In the **Search the marketplace** field, enter *virtual network* and select **Virtual network** from the returned list. The **Virtual network** page opens.
3. From the **Select a deployment model** list, select **Classic**, and then select **Create**. The **Create virtual network** page opens.
4. On the **Create virtual network** page, configure the VNet settings. On this page, you add your first address space and a single subnet address range. After you finish creating the VNet, you can go back and add additional subnets and address spaces.

5. Select the **Subscription** you want to use from the drop-down list.
6. Select an existing **Resource Group**. Or, create a new resource group by selecting **Create new** and entering a name. If you're creating a new resource group, name the resource group according to your planned configuration values. For more information about resource groups, see Azure Resource Manager overview.
7. Select a **Location** for your VNet. This setting determines the geographical location of the resources that you deploy to this VNet.
8. Select **Create** to create the VNet. From the **Notifications** page, you'll see a **Deployment in progress** message.
9. After your virtual network has been created, the message on the **Notifications** page changes to **Deployment succeeded**. Select **Pin to dashboard** if you want to easily find your VNet on the dashboard.
10. Add a DNS server (optional). After you create your virtual network, you can add the IP address of a DNS server for name resolution. The DNS server IP address that you specify should be the address of a DNS server that can resolve the names for the resources in your VNet.

To add a DNS server, select **DNS servers** from your VNet page. Then, enter the IP address of the DNS server that you want to use and select **Save**.

Part 2: Create a gateway subnet and a dynamic routing gateway

In this step, you create a gateway subnet and a dynamic routing gateway. In the Azure portal for the classic deployment model, you create the gateway subnet and the gateway through the same configuration pages. Use the gateway subnet for the gateway services only. Never deploy anything directly to the gateway subnet (such as VMs or other services).

1. In the Azure portal, navigate to the virtual network for which you want to create a gateway.
2. On the page for your virtual network, select **Overview**, and in the **VPN connections** section, select **Gateway**.

3. On the **New VPN Connection** page, select **Point-to-site**.

4. For **Client Address Space**, add the IP address range from which the VPN clients receive an IP address when connecting. Use a private IP address range that doesn't overlap with the on-premises location that you connect from, or with the VNet that you connect to. You can overwrite the autofilled range with the private IP address range that you want to use. This example shows the autofilled range.

5. Select **Create gateway immediately**, and then select **Optional gateway configuration** to open the **Gateway configuration** page.

6. From the **Gateway configuration** page, select **Subnet** to add the gateway subnet. It's possible to create a gateway subnet as small as /29. However, we recommend that you create a larger subnet that includes more addresses by selecting at least /28 or /27. Doing so will allow for enough addresses to accommodate possible additional configurations that you may want in the future. When working with gateway subnets, avoid associating a network security group (NSG) to the gateway subnet. Associating a network security group to this subnet may cause your VPN gateway to not function as expected. Select **OK** to save this setting.

7. Select the gateway **Size**. The size is the gateway SKU for your virtual network gateway. In the Azure portal, the default SKU is **Default**. For more information about gateway SKUs, see About VPN gateway settings.

8. Select the **Routing Type** for your gateway. P2S configurations require a **Dynamic** routing type. Select **OK** when you've finished configuring this page.

9. On the **New VPN Connection** page, select **OK** at the bottom of the page to begin creating your virtual network gateway. A VPN gateway can take up to 45 minutes to complete, depending on the gateway SKU that you select.

Create certificates

Azure uses certificates to authenticate VPN clients for Point-to-Site VPNs. You upload the public key information of the root certificate to Azure. The public key is then considered *trusted*. Client certificates must be generated from the trusted root certificate, and then installed on each client computer in the Certificates-Current User\Personal\Certificates certificate store. The certificate is used to authenticate the client when it connects to the VNet.

If you use self-signed certificates, they must be created by using specific parameters. You can create a self-signed certificate by using the instructions for PowerShell and Windows 10, or MakeCert. It's important to follow the steps in these instructions when you use self-signed root certificates and generate client certificates from the self-signed root certificate. Otherwise, the certificates you create won't be compatible with P2S connections and you'll receive a connection error.

Acquire the public key (.cer) for the root certificate

Use either a root certificate that was generated with an enterprise solution (recommended), or generate a self-signed certificate. After you create the root certificate, export the public certificate data (not the private key) as a Base64 encoded X.509 .cer file. Then, upload the public certificate data to the Azure server.

- **Enterprise certificate:** If you're using an enterprise solution, you can use your existing certificate chain. Acquire the .cer file for the root certificate that you want to use.
- **Self-signed root certificate:** If you aren't using an enterprise certificate solution, create a self-signed root certificate. Otherwise, the certificates you create won't be compatible with your P2S connections and clients will receive a connection error when they try to connect. You can use Azure PowerShell, MakeCert, or OpenSSL. The steps in the following articles describe how to generate a compatible self-signed root certificate:
  - Windows 10 PowerShell instructions: These instructions require Windows 10 and PowerShell to generate certificates. Client certificates that are generated from the root certificate can be installed on any supported P2S client.
  - MakeCert instructions: Use MakeCert if you don't have access to a Windows 10 computer to use to generate certificates. Although MakeCert is deprecated, you can still use it to generate certificates. Client certificates that you generate from the root certificate can be installed on any supported P2S client.
  - Linux instructions

Generate a client certificate

Each client computer that you connect to a VNet with a Point-to-Site connection must have a client certificate installed. You generate it from the root certificate and install it on each client computer. If you don't install a valid client certificate, authentication will fail when the client tries to connect to the VNet.

You can either generate a unique certificate for each client, or you can use the same certificate for multiple clients. The advantage to generating unique client certificates is the ability to revoke a single certificate. Otherwise, if multiple clients use the same client certificate to authenticate and you revoke it, you'll need to generate and install new certificates for every client that uses that certificate.

You can generate client certificates by using the following methods:

- **Enterprise certificate:**
  - If you're using an enterprise certificate solution, generate a client certificate with the common name value format *name@yourdomain.com*. Use this format instead of the *domain name**username* format.

- Make sure the client certificate is based on a user certificate template that has *Client Authentication* listed as the first item in the user list. Check the certificate by double-clicking it and viewing **Enhanced Key Usage** in the **Details** tab.

- **Self-signed root certificate:** Follow the steps in one of the following P2S certificate articles so that the client certificates you create will be compatible with your P2S connections. The steps in these articles generate a compatible client certificate:
  - Windows 10 PowerShell instructions: These instructions require Windows 10 and PowerShell to generate certificates. The generated certificates can be installed on any supported P2S client.
  - MakeCert instructions: Use MakeCert if you don't have access to a Windows 10 computer for generating certificates. Although MakeCert is deprecated, you can still use it to generate certificates. You can install the generated certificates on any supported P2S client.
  - Linux instructions

When you generate a client certificate from a self-signed root certificate, it's automatically installed on the computer that you used to generate it. If you want to install a client certificate on another client computer, export it as a .pfx file, along with the entire certificate chain. Doing so will create a .pfx file that contains the root certificate information required for the client to authenticate.

To export the certificate

For steps to export a certificate, see Generate and export certificates for Point-to-Site using PowerShell.

Upload the root certificate .cer file

After the gateway has been created, upload the .cer file (which contains the public key information) for a trusted root certificate to the Azure server. Don't upload the private key for the root certificate. After you upload the certificate, Azure uses it to authenticate clients that have installed a client certificate generated from the trusted root certificate. You can later upload additional trusted root certificate files (up to 20), if needed.

1. On the **VPN connections** section of the page for your VNet, select the clients graphic to open the **Point-to-site VPN connection** page.

2. On the **Point-to-site VPN connection** page, select **Manage certificate** to open the **Certificates** page.

3. On the **Certificates** page, select **Upload** to open the **Upload certificate** page.

4. Select the folder graphic to browse for the .cer file. Select the file, then select **OK**. The uploaded certificate appears on the **Certificates** page.

Configure the client

To connect to a VNet by using a Point-to-Site VPN, each client must install a package to configure the native Windows VPN client. The configuration package configures the native Windows VPN client with the settings necessary to connect to the virtual network.

You can use the same VPN client configuration package on each client computer, as long as the version matches the architecture for the client. For the list of client operating systems that are supported, see the Point-to-Site connections FAQ.

Generate and install a VPN client configuration package

1. In the Azure portal, in the **Overview** page for your VNet, in **VPN connections**, select the client graphic to open the **Point-to-site VPN connection** page.
2. From the **Point-to-site VPN connection** page, select the download package that corresponds to the client operating system where it's installed:
   - For 64-bit clients, select **VPN Client (64-bit)**.
   - For 32-bit clients, select **VPN Client (32-bit)**.
3. After the package generates, download it and then install it on your client computer. If you see a SmartScreen popup, select **More info**, then select **Run anyway**. You can also save the package to install on other client computers.

Install a client certificate

To create a P2S connection from a different client computer than the one used to generate the client certificates, install a client certificate. When you install a client certificate, you need the password that was created when the client certificate was exported. Typically, you can install the certificate by just double-clicking it. For more information, see Install an exported client certificate.

Connect to your VNet

Note

You must have Administrator rights on the client computer from which you are connecting.

1. To connect to your VNet, on the client computer, navigate to **VPN connections** in the Azure portal and locate the VPN connection that you created. The VPN connection has the same name as your virtual network. Select **Connect**. If a pop-up message about the certificate appears, select **Continue** to use elevated privileges.
2. On the **Connection** status page, select **Connect** to start the connection. If you see the **Select Certificate** screen, verify that the displayed client certificate is the correct one. If not, select the correct certificate from the drop-down list, and then select **OK**.
3. If your connection succeeds, you'll see a **Connected** notification.

Troubleshooting P2S connections

If you have trouble connecting, check the following items:

- If you exported a client certificate with **Certificate Export Wizard**, make sure that you exported it as a .pfx file and selected **Include all certificates in the certification path if possible**. When you export it with this value, the root certificate information is also exported. After you install the certificate on the client computer, the root certificate in the .pfx file is also installed. To verify that the root certificate is installed, open **Manage user**

certificates and select **Trusted Root Certification Authorities\Certificates**. Verify that the root certificate is listed, which must be present for authentication to work.

- If you used a certificate that was issued by an Enterprise CA solution and you can't authenticate, verify the authentication order on the client certificate. Check the authentication list order by double-clicking the client certificate, selecting the **Details** tab, and then selecting **Enhanced Key Usage**. Make sure *Client Authentication* is the first item in the list. If it isn't, issue a client certificate based on the user template that has *Client Authentication* as the first item in the list.
- For additional P2S troubleshooting information, see Troubleshoot P2S connections.

Verify the VPN connection

1. Verify that your VPN connection is active. Open an elevated command prompt on your client computer, and run **ipconfig/all**.
2. View the results. Notice that the IP address you received is one of the addresses within the Point-to-Site connectivity address range that you specified when you created your VNet. The results should be similar to this example:

Copy

```
PPP adapter VNet1:
   Connection-specific DNS Suffix .:
   Description.....: VNet1
   Physical Address.....:
   DHCP Enabled.....: No
   Autoconfiguration Enabled.....: Yes
   IPv4 Address.....: 192.168.130.2(Preferred)
   Subnet Mask.....: 255.255.255.255
   Default Gateway.....:
   NetBIOS over Tcpip.....: Enabled
```

Connect to a virtual machine

Create a Remote Desktop Connection to connect to a VM that's deployed to your VNet. The best way to verify you can connect to your VM is to connect with its private IP address, rather than its computer name. That way, you're testing to see if you can connect, not whether name resolution is configured properly.

1. Locate the private IP address for your VM. To find the private IP address of a VM, view the properties for the VM in the Azure portal or use PowerShell.
2. Verify that you're connected to your VNet with the Point-to-Site VPN connection.
3. To open Remote Desktop Connection, enter *RDP* or *Remote Desktop Connection* in the search box on the taskbar, then select **Remote Desktop Connection**. You can also open it by using the **mstsc** command in PowerShell.
4. In **Remote Desktop Connection**, enter the private IP address of the VM. If necessary, select **Show Options** to adjust additional settings, then connect.

To troubleshoot an RDP connection to a VM

If you're having trouble connecting to a virtual machine over your VPN connection, there are a few things you can check.

- Verify that your VPN connection is successful.
- Verify that you're connecting to the private IP address for the VM.
- Enter **ipconfig** to check the IPv4 address assigned to the Ethernet adapter on the computer from which you're connecting. An overlapping address space occurs when the IP address is within the address range of the VNet that you're connecting to, or within the address range of your VPNClientAddressPool. When your address space overlaps in this way, the network traffic doesn't reach Azure, it stays on the local network.
- If you can connect to the VM by using the private IP address, but not the computer name, verify that you have configured DNS properly. For more information about how name resolution works for VMs, see Name Resolution for VMs.
- Verify that the VPN client configuration package is generated after you specify the DNS server IP addresses for the VNet. If you update the DNS server IP addresses, generate and install a new VPN client configuration package.

For more troubleshooting information, see Troubleshoot Remote Desktop connections to a VM.

Add or remove trusted root certificates

You can add and remove trusted root certificates from Azure. When you remove a root certificate, clients that have a certificate generated from that root can no longer authenticate and connect. For those clients to authenticate and connect again, you must install a new client certificate generated from a root certificate that's trusted by Azure.

To add a trusted root certificate

You can add up to 20 trusted root certificate .cer files to Azure. For instructions, see Upload the root certificate .cer file.

To remove a trusted root certificate

1. On the **VPN connections** section of the page for your VNet, select the clients graphic to open the **Point-to-site VPN connection** page.

2. On the **Point-to-site VPN connection** page, select **Manage certificate** to open the **Certificates** page.

3. On the **Certificates** page, select the ellipsis next to the certificate that you want to remove, then select **Delete**.

Revoke a client certificate

If necessary, you can revoke a client certificate. The certificate revocation list allows you to selectively deny Point-to-Site connectivity based on individual client certificates. This method differs from removing a trusted root certificate. If you remove a trusted root

certificate .cer from Azure, it revokes the access for all client certificates generated/signed by the revoked root certificate. Revoking a client certificate, rather than the root certificate, allows the other certificates that were generated from the root certificate to continue to be used for authentication for the Point-to-Site connection.

The common practice is to use the root certificate to manage access at team or organization levels, while using revoked client certificates for fine-grained access control on individual users.

To revoke a client certificate

You can revoke a client certificate by adding the thumbprint to the revocation list.

1. Retrieve the client certificate thumbprint. For more information, see How to: Retrieve the Thumbprint of a Certificate.
2. Copy the information to a text editor and remove its spaces so that it's a continuous string.
3. Navigate to the classic virtual network. Select **Point-to-site VPN connection**, then select **Manage certificate** to open the **Certificates** page.
4. Select **Revocation list** to open the **Revocation list** page.
5. Select **Add certificate** to open the **Add certificate to revocation list** page.
6. In **Thumbprint**, paste the certificate thumbprint as one continuous line of text, with no spaces. Select **OK** to finish.

After updating has completed, the certificate can no longer be used to connect. Clients that try to connect by using this certificate receive a message saying that the certificate is no longer valid.

q)vnet peering step by step?

Tutorial: Connect virtual networks with virtual network peering using the Azure portal

- 08/16/2018
- 5 minutes to read
-

You can connect virtual networks to each other with virtual network peering. These virtual networks can be in the same region or different regions (also known as Global VNet peering). Once virtual networks are peered, resources in both virtual networks are able to communicate with each other, with the same latency and bandwidth as if the resources were in the same virtual network. In this tutorial, you learn how to:

- Create two virtual networks
- Connect two virtual networks with a virtual network peering
- Deploy a virtual machine (VM) into each virtual network
- Communicate between VMs

If you prefer, you can complete this tutorial using the Azure CLI or Azure PowerShell.

If you don't have an Azure subscription, create a free account before you begin.

Log in to Azure

Log in to the Azure portal at https://portal.azure.com.

Create virtual networks

1. Select **+ Create a resource** on the upper, left corner of the Azure portal.

2. Select **Networking**, and then select **Virtual network**.
3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **Create**:

Setting	**Value**
Name	myVirtualNetwork1
Address space	10.0.0.0/16
Subscription	Select your subscription.
Resource group	Select **Create new** and enter *myResourceGroup*.
Location	Select **East US**.
Subnet Name	Subnet1
Subnet Address range	10.0.0.0/24

4.
5. Complete steps 1-3 again, with the following changes:

Setting	**Value**
Name	myVirtualNetwork2
Address space	10.1.0.0/16
Resource group	Select **Use existing** and then select **myResourceGroup**.
Subnet Address range	10.1.0.0/24

Peer virtual networks

1. In the Search box at the top of the Azure portal, begin typing *MyVirtualNetwork1*. When **myVirtualNetwork1** appears in the search results, select it.
2. Select **Peerings**, under **SETTINGS**, and then select **+ Add**, as shown in the following picture:

3. Enter, or select, the following information, accept the defaults for the remaining settings, and then select **OK**.

Setting	Value
Name	myVirtualNetwork1-myVirtualNetwork2
Subscription	Select your subscription.
Virtual network	myVirtualNetwork2 - To select the *myVirtualNetwork2* virtual network, select **Virtual network**, then select **myVirtualNetwork2**. You can select a virtual network in the same region or in a different region.

4.
5. The **PEERING STATUS** is *Initiated*, as shown in the following picture:

6.
7. If you don't see the status, refresh your browser.
8. In the **Search** box at the top of the Azure portal, begin typing *MyVirtualNetwork2*. When **myVirtualNetwork2** appears in the search results, select it.
9. Complete steps 2-3 again, with the following changes, and then select **OK**:

Setting	Value
Name	myVirtualNetwork2-myVirtualNetwork1
Virtual network	myVirtualNetwork1

10. The **PEERING STATUS** is *Connected*. Azure also changed the peering status for the *myVirtualNetwork2-myVirtualNetwork1* peering from *Initiated* to *Connected.* Virtual network peering is not fully established until the peering status for both virtual networks is *Connected.*

Create virtual machines

Create a VM in each virtual network so that you can communicate between them in a later step.

Create the first VM

1. Select **+ Create a resource** on the upper, left corner of the Azure portal.
2. Select **Compute**, and then select **Windows Server 2016 Datacenter**. You can select a different operating system, but the remaining steps assume you selected **Windows Server 2016 Datacenter**.
3. Enter, or select, the following information for **Basics**, accept the defaults for the remaining settings, and then select **Create**:

Setting	Value
Name	myVm1
User name	Enter a user name of your choosing.
Password	Enter a password of your choosing. The password must be at least 12 characters long and meet the defined complexity requirements.
Resource group	Select **Use existing** and then select **myResourceGroup**.
Location	Select **East US**.

4. Select a VM size under **Choose a size**.
5. Select the following values for **Settings**, then select **OK**:

Setting	Value
Virtual network	myVirtualNetwork1 - If it's not already selected, select **Virtual network** and then select **myVirtualNetwork1** under **Choose virtual network**.
Subnet	Subnet1 - If it's not already selected, select **Subnet** and then select **Subnet1** under **Choose subnet**.

6.
7. Under **Create** in the **Summary**, select **Create** to start the VM deployment.

Create the second VM

Complete steps 1-6 again, with the following changes:

Setting	Value
Name	myVm2
Virtual network	myVirtualNetwork2

The VMs take a few minutes to create. Do not continue with the remaining steps until both VMs are created.

Communicate between VMs

1. In the *Search* box at the top of the portal, begin typing *myVm1*. When **myVm1** appears in the search results, select it.
2. Create a remote desktop connection to the *myVm1* VM by selecting **Connect**, as shown in the following picture:

3. To connect to the VM, open the downloaded RDP file. If prompted, select **Connect**.
4. Enter the user name and password you specified when creating the VM (you may need to select **More choices**, then **Use a different account**, to specify the credentials you entered when you created the VM), then select **OK**.
5. You may receive a certificate warning during the sign-in process. Select **Yes** to proceed with the connection.
6. In a later step, ping is used to communicate with the *myVm2* VM from the *myVm1* VM. Ping uses the Internet Control Message Protocol (ICMP), which is denied through the Windows Firewall, by default. On the *myVm1* VM, enable ICMP through the Windows firewall, so that you can ping this VM from *myVm2* in a later step, using PowerShell:

PowerShellCopy

New-NetFirewallRule –DisplayName “Allow ICMPv4-In” –Protocol ICMPv4

Though ping is used to communicate between VMs in this tutorial, allowing ICMP through the Windows Firewall for production deployments is not recommended.

7. To connect to the *myVm2* VM, enter the following command from a command prompt on the *myVm1* VM:

Copy

mstsc /v:10.1.0.4

8. Since you enabled ping on *myVm1*, you can now ping it by IP address:

Copy

ping 10.0.0.4

9. Disconnect your RDP sessions to both *myVm1* and *myVm2*.

Clean up resources

When no longer needed, delete the resource group and all resources it contains:

1. Enter *myResourceGroup* in the **Search** box at the top of the portal. When you see **myResourceGroup** in the search results, select it.
2. Select **Delete resource group**.
3. Enter *myResourceGroup* for **TYPE THE RESOURCE GROUP NAME:** and select **Delete**.

Next steps

In this tutorial, you learned how to connect two networks in the same Azure region, with virtual network peering. You can also peer virtual networks in different supported regions and in different Azure subscriptions, as well as create hub and spoke network designs with peering. To learn more about virtual network peering, see Virtual network peering overview and Manage virtual network peerings.

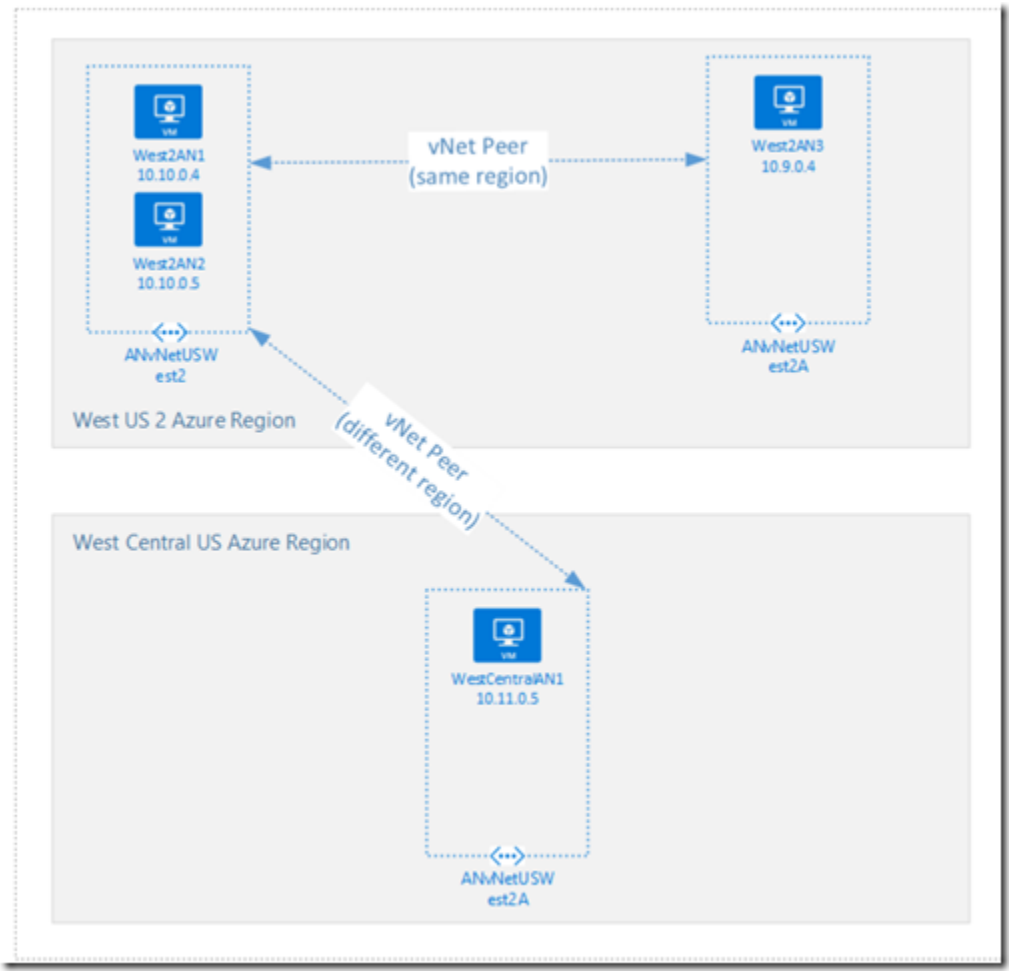
To connect your own computer to a virtual network through a VPN, and interact with resources in a virtual network, or in peered virtual networks, see Connect your computer to a virtual network.

q)global vnet peering azure?
How to setup Global VNet peering in Azure

Now that Global VNet peering has gone to GA I had a large university in California ask how to set this up.

What is Global VNet peering?

Global VNet peering in Azure is the ability to peer VNets or virtual networks across regions (see regional availability below).



Some benefits of Global VNet peering include:

- Private Peering traffic stays on Azure network backbone
- Low latency and high bandwidth VNet region to VNet region connectivity
- No more VNet to VNet VPN configuration which means no VPN encryption, no gateways, no public internet necessary

- No downtime setting up Global VNet peering with portal or ARM templates

What are some limitations of Global VNet peering?

You cannot use Global VNet peering to communicate with VIPs of load balancers in another region. VIP communication requires source IP to be on the same VNet as the LB IP:

- *Resources in one virtual network cannot communicate with the IP address of an Azure internal load balancer in the peered virtual network. The load balancer and the resources that communicate with it must be in the same virtual network.*

This is a big one as it bit us with a customer. You must not check 'use remote gateways' or 'allow gateway transit' like you select when setting up intra regional VNet peering:

- *You cannot use remote gateways or allow gateway transit. To use remote gateways or allow gateway transit, both virtual networks in the peering must exist in the same region.*

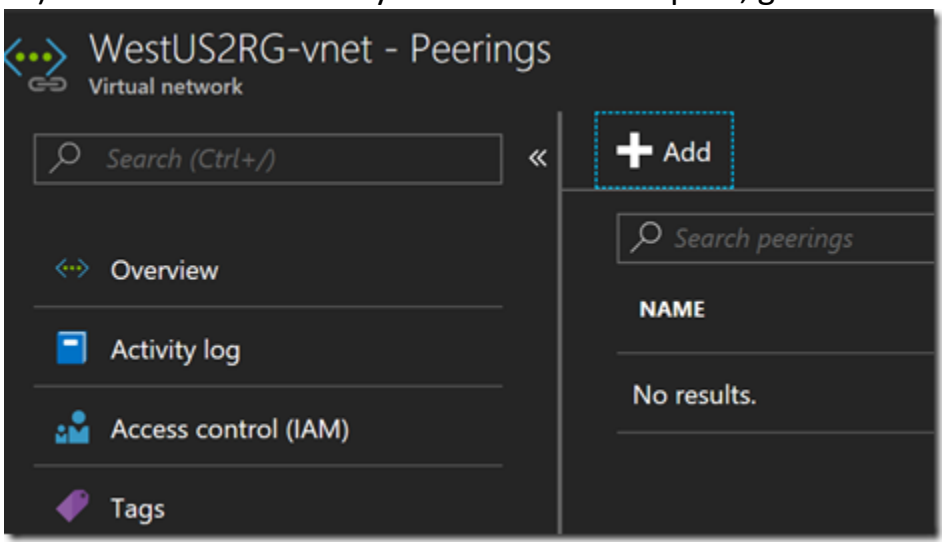
VNet Global peerings are not transitive meaning downstream VNets in one region cannot talk with downstream VNets in another region.

How do I setup Global VNet peering?

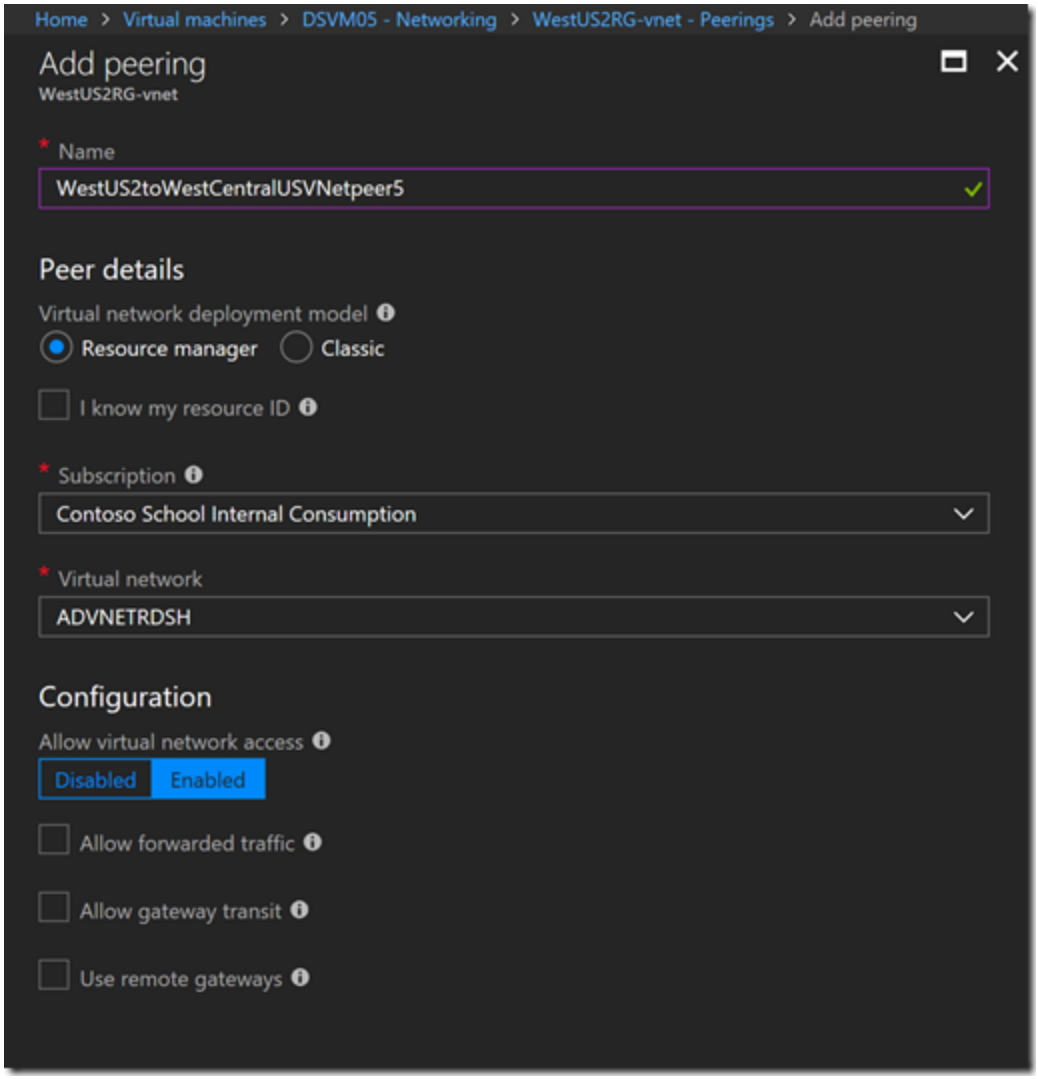
It is fairly straightforward to setup Global VNet peering and I documented the steps below:

1) Setup VNets in each region you want to Global VNet peer. There are a few Global VNet restrictions for regions see here for region restrictions. In my case, I created a VNet in **US West 2** and **West Central US**
2) For testing create a VM in each region and associate it to a VNet you are going to use with Global VNet peering to test VM connectivity. The VMs cannot be associated with a downstream VNet as you recall since transitive downstream VNets are not supported with Global VNet peering.
3) Enable Global VNet peering:

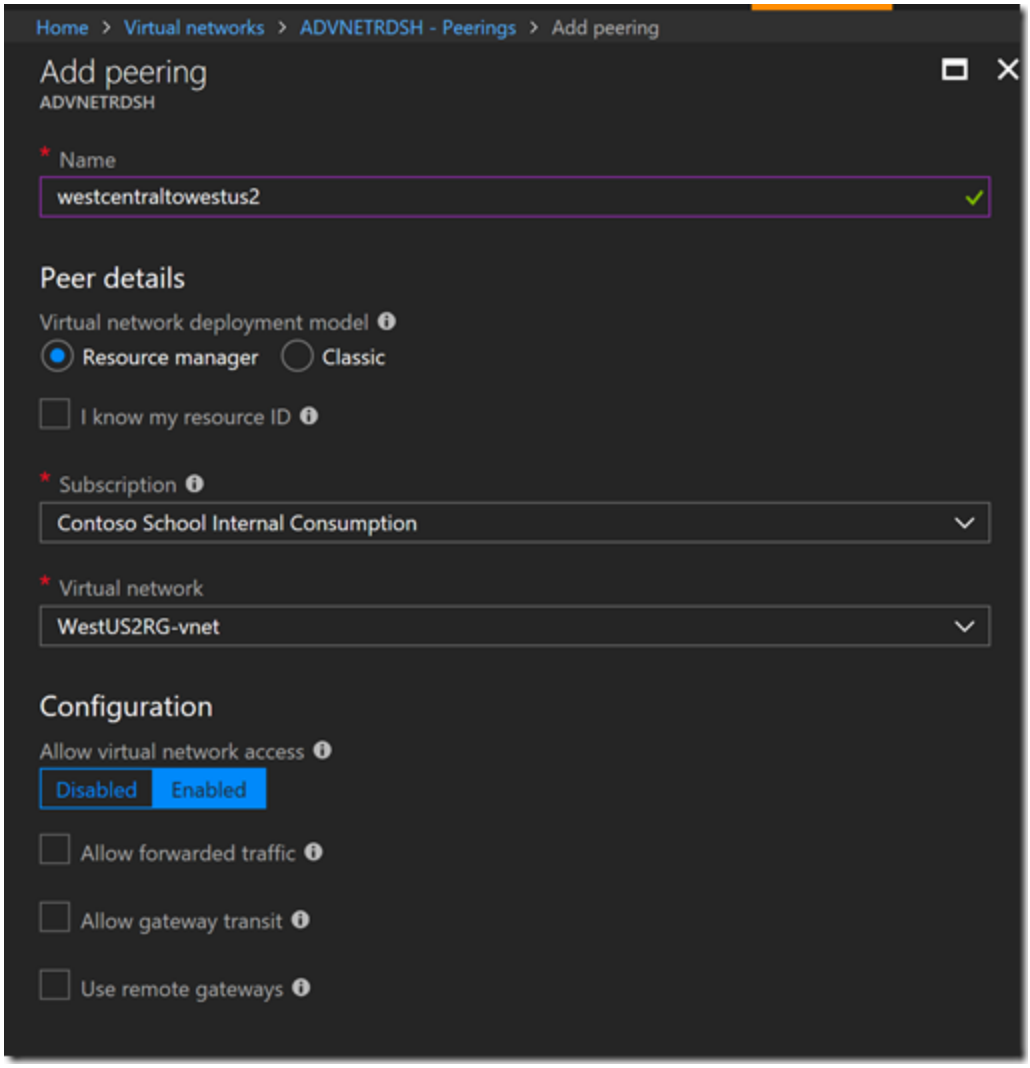
3a) On one of the VNets you want to Global peer, go to **Peerings** and click **Add**



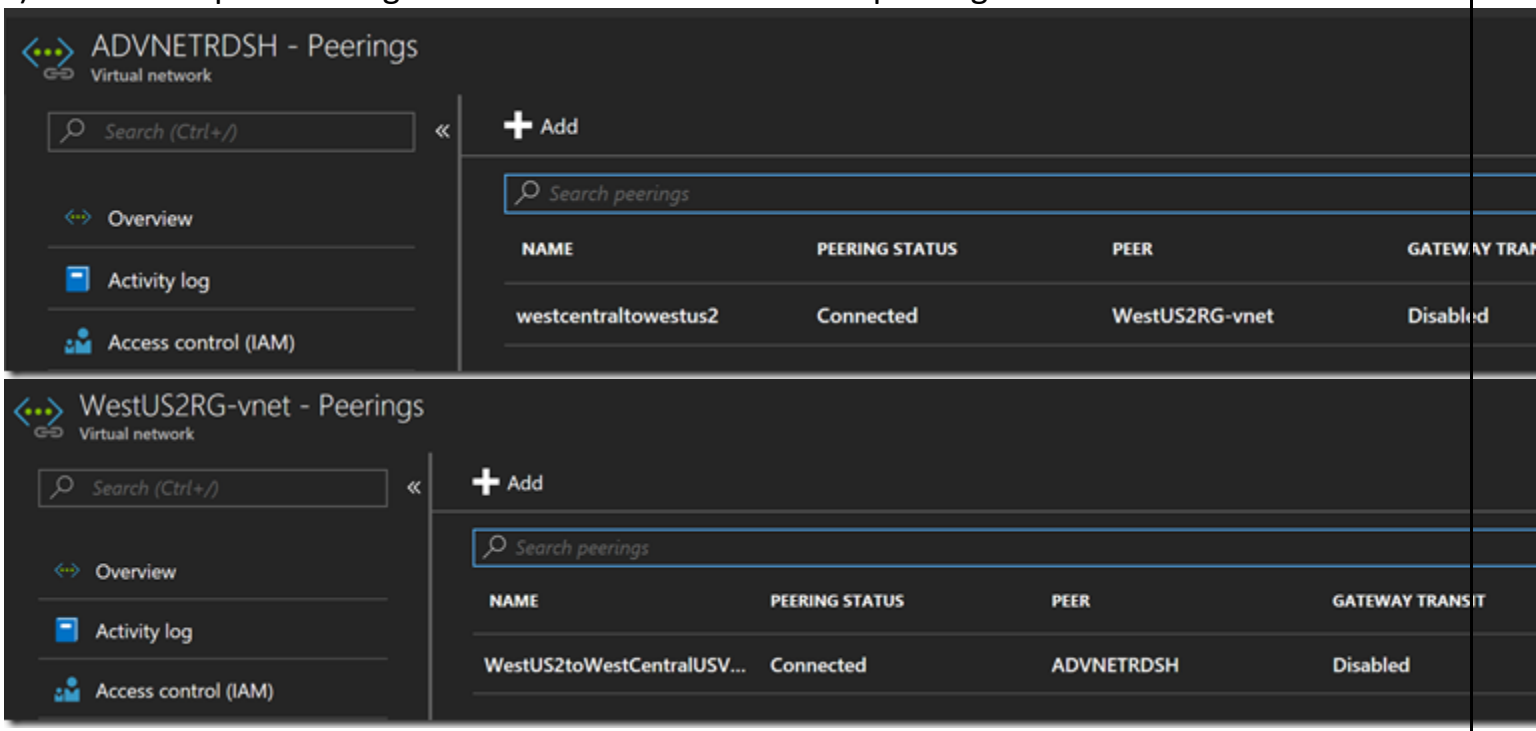
3b) Fill out the peering and select the VNet in the other region you want to Global Peer with – **Important** – don't check any of the checkboxes!



3c) Setup peering in other direction by repeating step 3a and 3b but with the VNet peering in the other region – also don't click any checkboxes and pick the other VNet in the other region you want to Global peer with:



4) Let it finish provisioning and then validate VNet Global peering is **Connected** on both VNets:



5) Test Global VNet connectivity with Telnet, PSPing or Ping Note: ensure firewalls allow ports you are telneting to, ICMP, etc.:

Test connectivity from one direction:

```
C:\>ping 10.0.0.8 -t

Pinging 10.0.0.8 with 32 bytes of data:
Reply from 10.0.0.8: bytes=32 time=21ms TTL=128
Reply from 10.0.0.8: bytes=32 time=24ms TTL=128
Reply from 10.0.0.8: bytes=32 time=23ms TTL=128
Reply from 10.0.0.8: bytes=32 time=23ms TTL=128
Reply from 10.0.0.8: bytes=32 time=22ms TTL=128
Reply from 10.0.0.8: bytes=32 time=21ms TTL=128
Reply from 10.0.0.8: bytes=32 time=25ms TTL=128
Reply from 10.0.0.8: bytes=32 time=21ms TTL=128
Reply from 10.0.0.8: bytes=32 time=21ms TTL=128
Reply from 10.0.0.8: bytes=32 time=25ms TTL=128
```

Then test connectivity from the other direction:

```
C:\>ping 172.16.7.4 -t

Pinging 172.16.7.4 with 32 bytes of data:
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=21ms TTL=128
Reply from 172.16.7.4: bytes=32 time=23ms TTL=128
```

6) You are done! Congratulations Global VNet peering is complete!

7) Optional – If you need reduced latency between the regional VNets you can optionally enable Virtual Machine Accelerated Networking for Windows or for Linux

q)Creating an Azure Point-to-Site VPN?

Step-By-Step: Creating an Azure Point-to-Site VPN

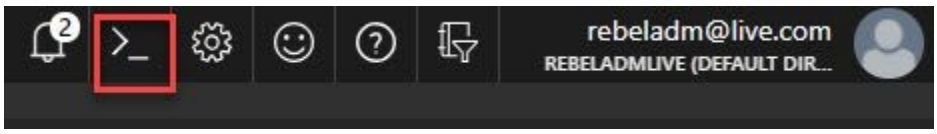
Site-to-Site VPN is the most common method organizations use to connect on-premises network to Azure vNet. This VPN connection is initiated in your edge firewall or router level. But what if you connecting from remote location such as home? We can use point-to-site method to do that. In this method it will use certificates to do the authentication between end point and azure virtual network.

So,
let's go ahead and see how we can do that,

Create Resource Group

In this exercise, I like to use separate resource group for virtual network and other components.

1. Log in to **Azure portal** as global administrator
2. Launch Cloud Shell



1. Then run **New-AzureRmResourceGroup -Name REBELVPNRG -Location "East US".** In here **REBELVPNRG** is resource group name and **East US** is the location.

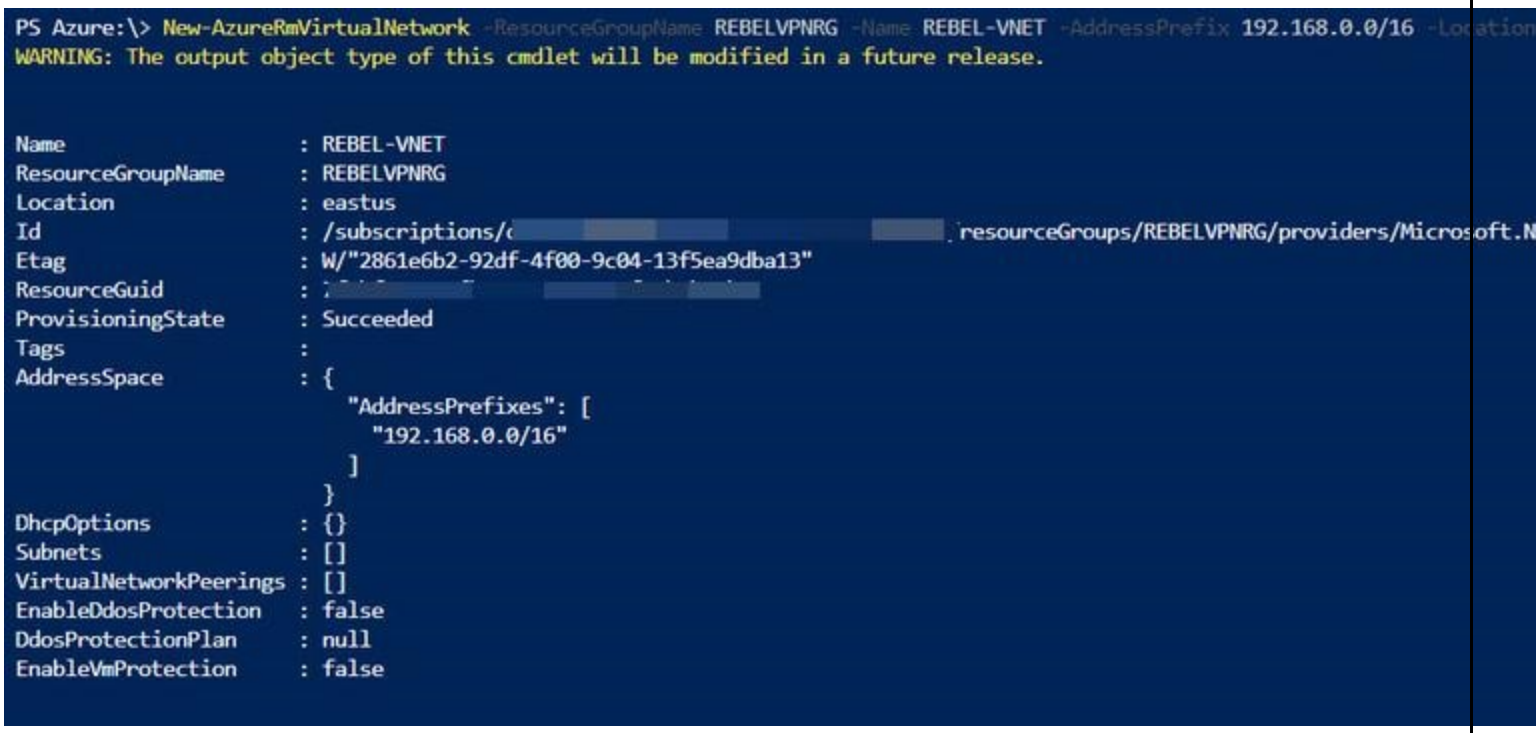
```
PS Azure:\> New-AzureRmResourceGroup -Name REBELVPNRG -Location "East US"


ResourceGroupName : REBELVPNRG
Location          : eastus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/                              /resourceGroups/REBELVPN
```

Create Virtual Network

Now we need to create new virtual network. We can create virtual network using,

New-AzureRmVirtualNetwork -ResourceGroupName REBELVPNRG -Name REBEL-VNET -AddressPrefix 192.168.0.0/16 -Location "East US"

In above, **REBEL-VNET** is the virtual network name. it uses **192.168.0.0/16** IP address range.



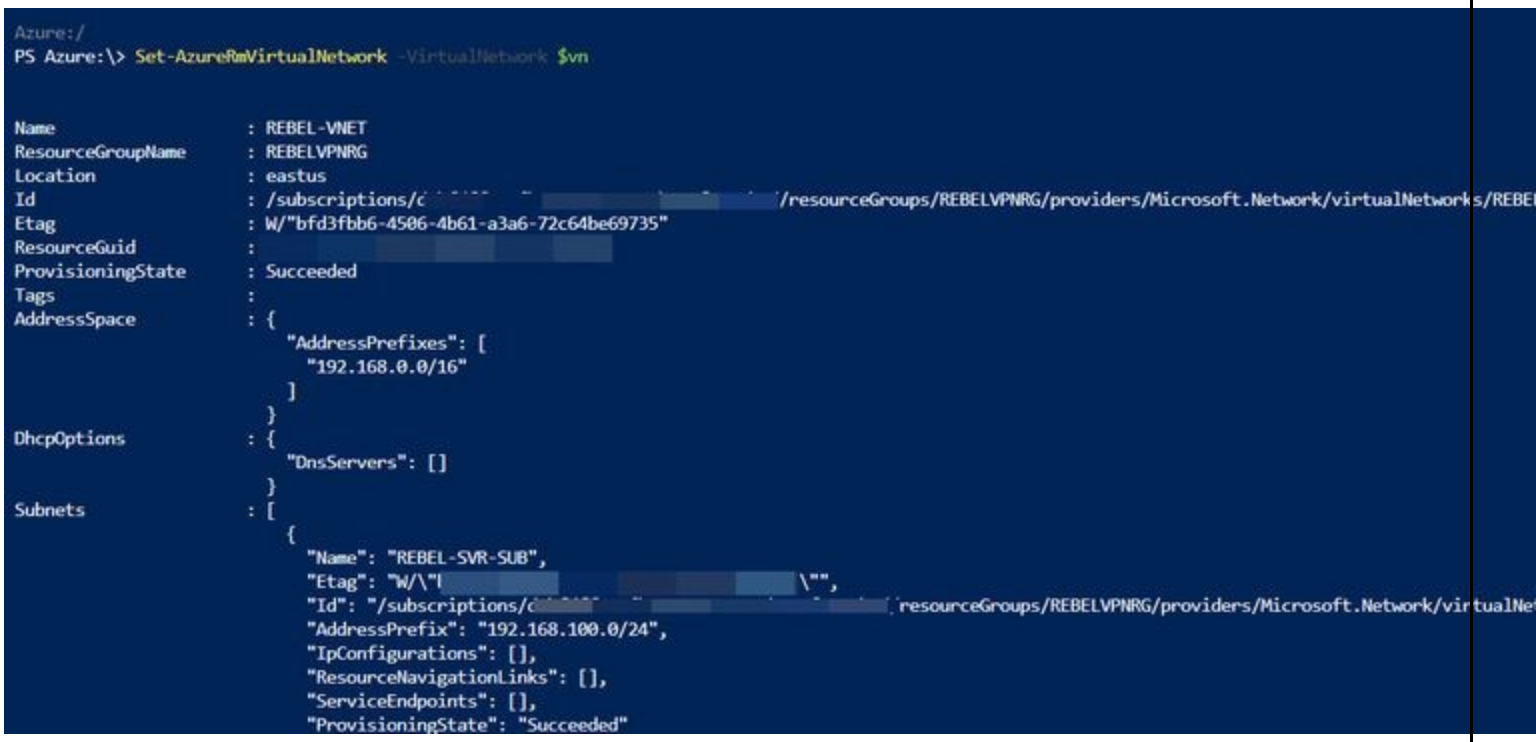
Create Subnets

Under the virtual network I am going to create a subnet for my servers. To create subnet use,

$vn = Get-AzureRmVirtualNetwork -ResourceGroupName REBELVPNRG -Name REBEL-VNET

Add-AzureRmVirtualNetworkSubnetConfig -Name REBEL-SVR-SUB -VirtualNetwork $vn -AddressPrefix 192.168.100.0/24

Set-AzureRmVirtualNetwork -VirtualNetwork $vn

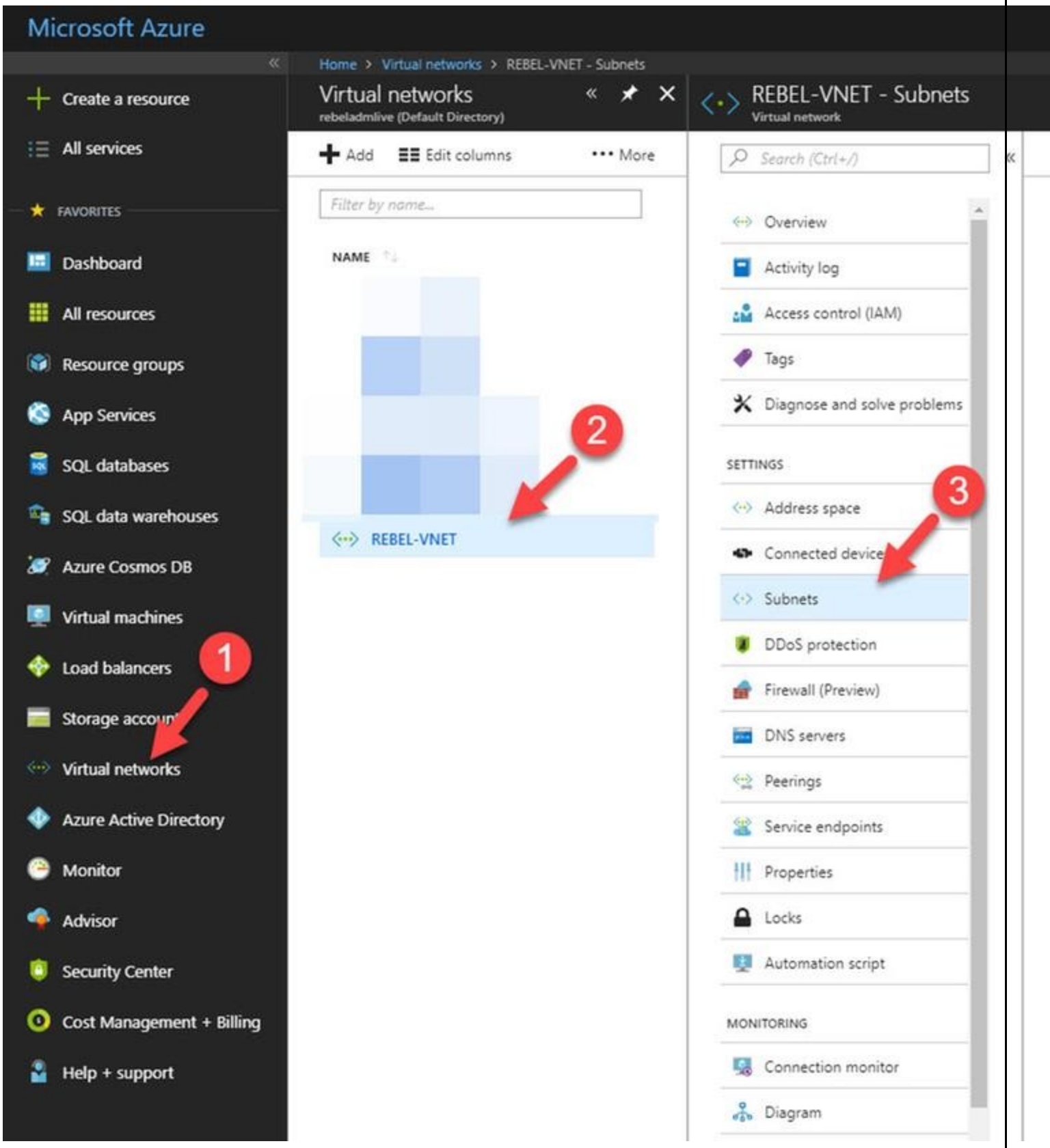


Create Gateway Subnet

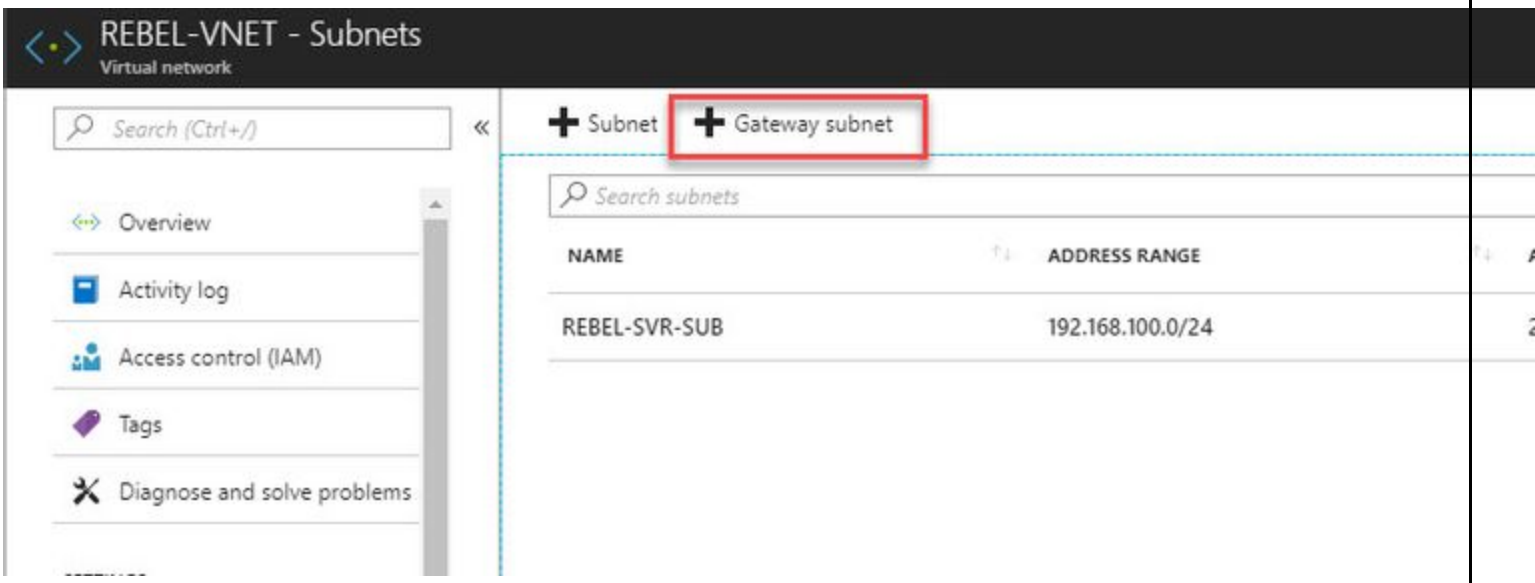
Before we create VN gateway, we need to create gateway subnet for it. so gateway will use ip addresses assigned in this subnet.

To do that,

- Log in to Azure portal as global administrator
- Go to **Virtual Networks | REBEL-VNET** (VNet created on previous steps) **| Subnets**



- Click on **Gateway Subnet**

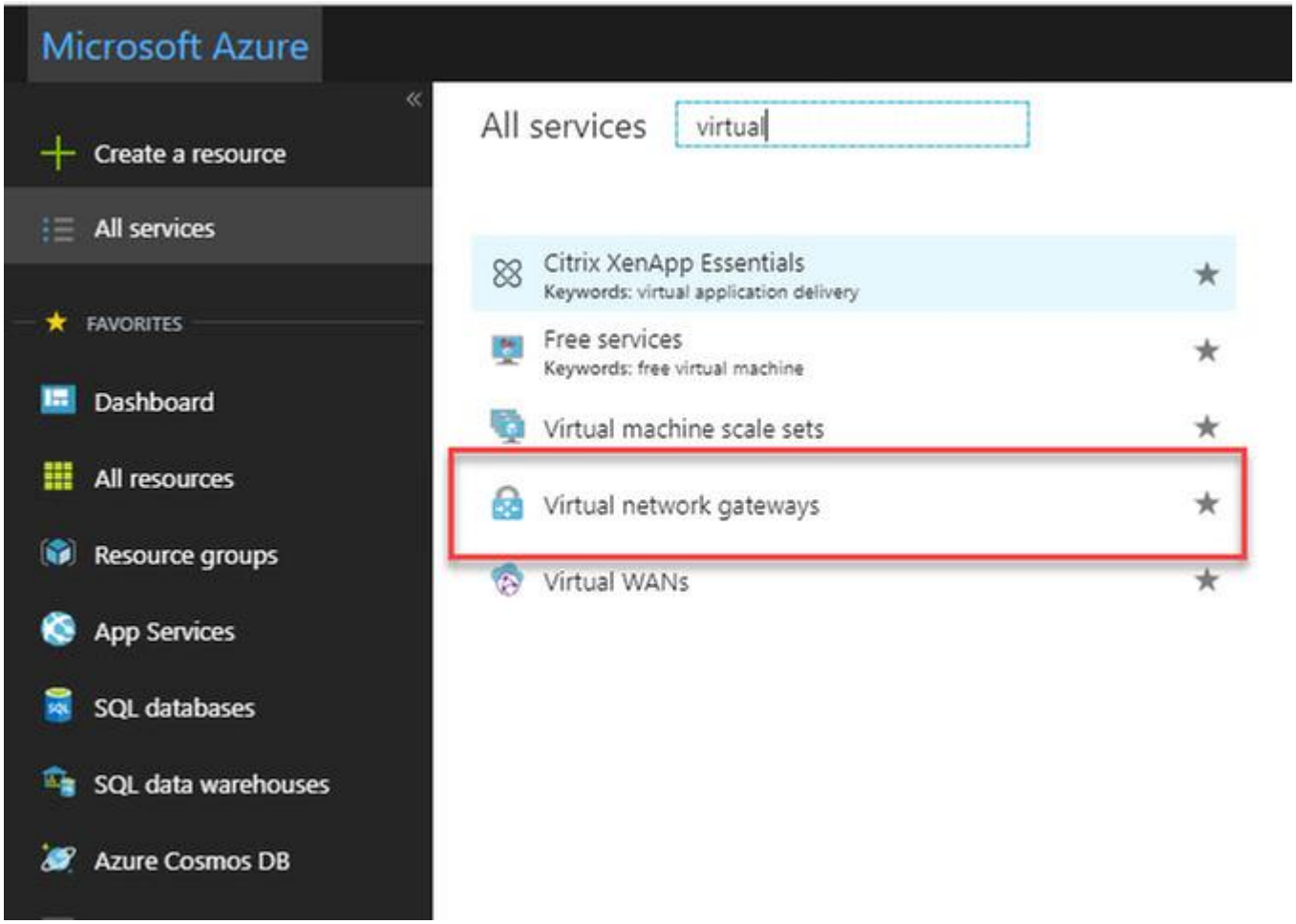


- In new window, define the ip range for gateway subnet and click **Ok**

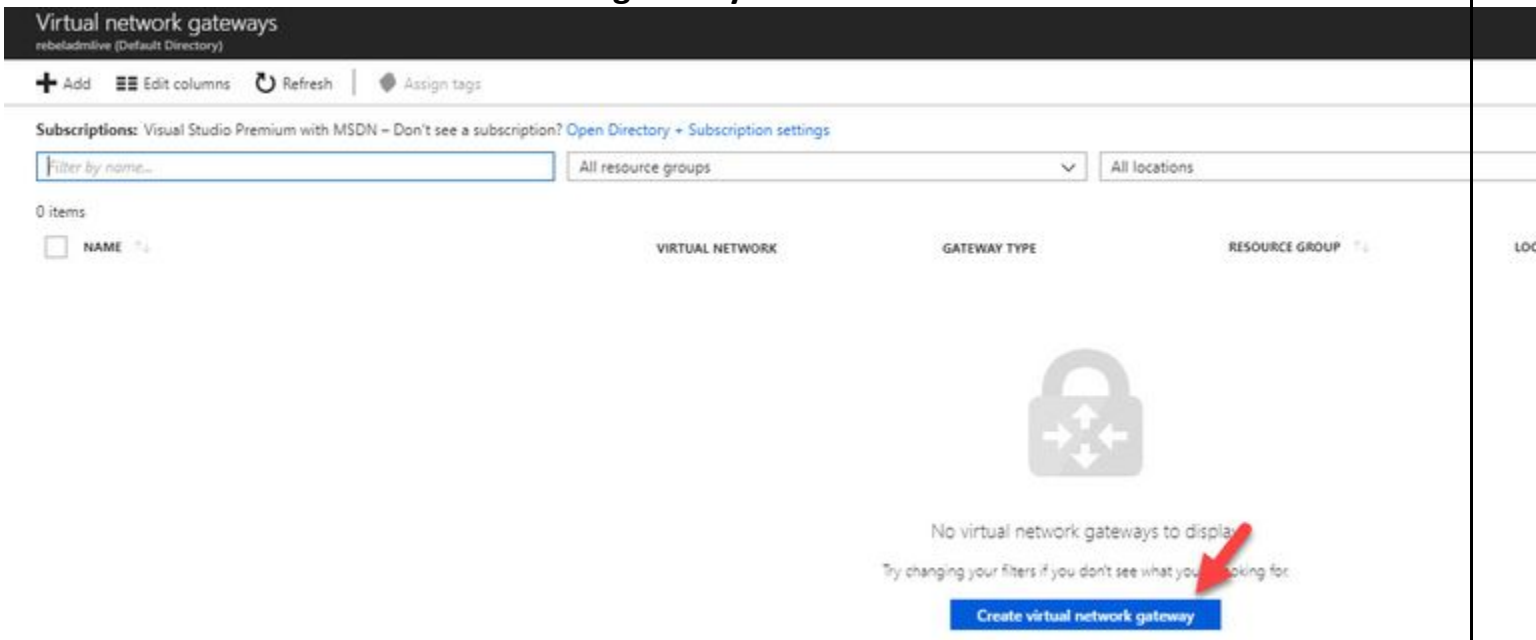
Create Virtual Network Gateway

Now we have all the things needed to create new VN gateway. To do that,

- Log in to Azure portal as global administrator
- Go to **All Services** and search for virtual network gateway. Once it is in list, click on it.

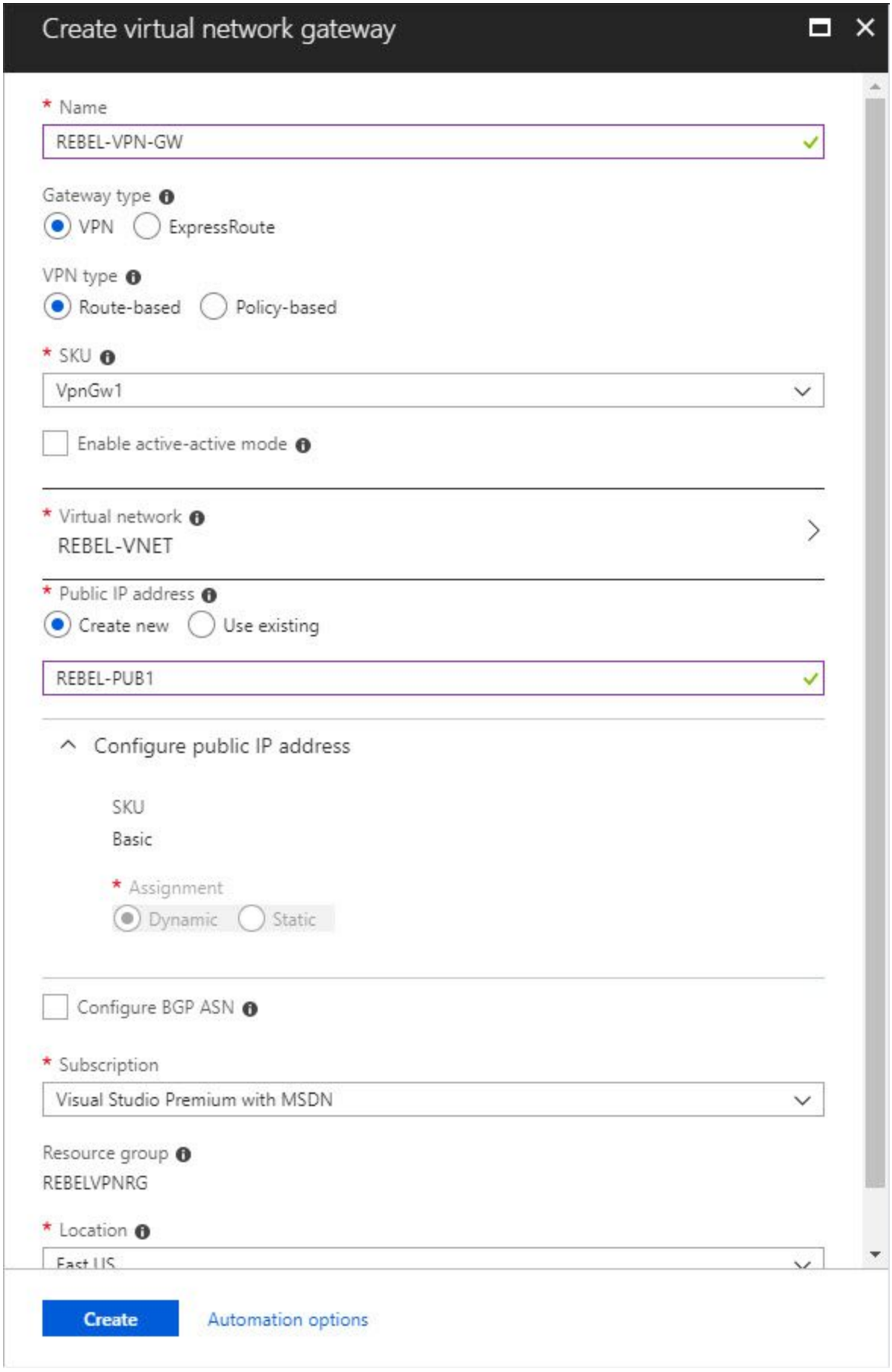


- Then click on **Create virtual network gateway**



- In new window fill relevant info and click on **Create**

In here, **REBEL-VPN-GW** is the gateway name. I have selected **REBEL-VNET** as the virtual network. I am also creating public ip called **REBEL-PUB1**. This is only supported with dynamic mode. This doesn't mean it is going to change randomly. It will only happen when gateway is deleted or read.



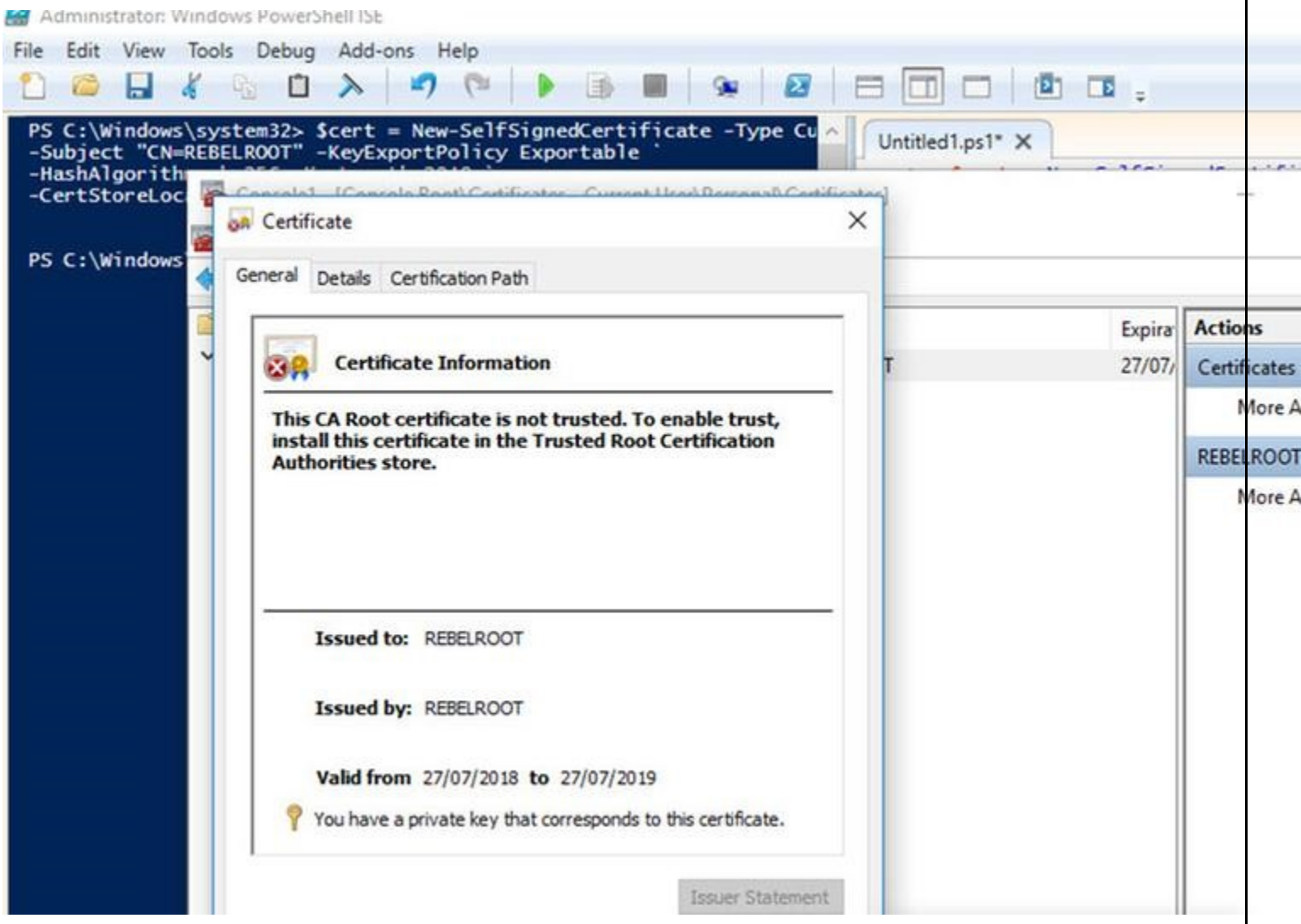
Create Self-sign root & client certificate

If your organization using internal CA, you always can use it to generate relevant certificates for this exercise. If you do not have internal CA, we still can use self-sign certs to do the job.

As first step I am going to create root certificate. In Windows 10 machine I can run this to create root cert first.

$cert = New-SelfSignedCertificate -Type Custom -KeySpec Signature `
-Subject "CN=REBELROOT" -KeyExportPolicy Exportable `
-HashAlgorithm sha256 -KeyLength 2048 `
-CertStoreLocation "Cert:\CurrentUser\My" -KeyUsageProperty Sign -KeyUsage CertSign

This will create root cert and install it under current user cert store.

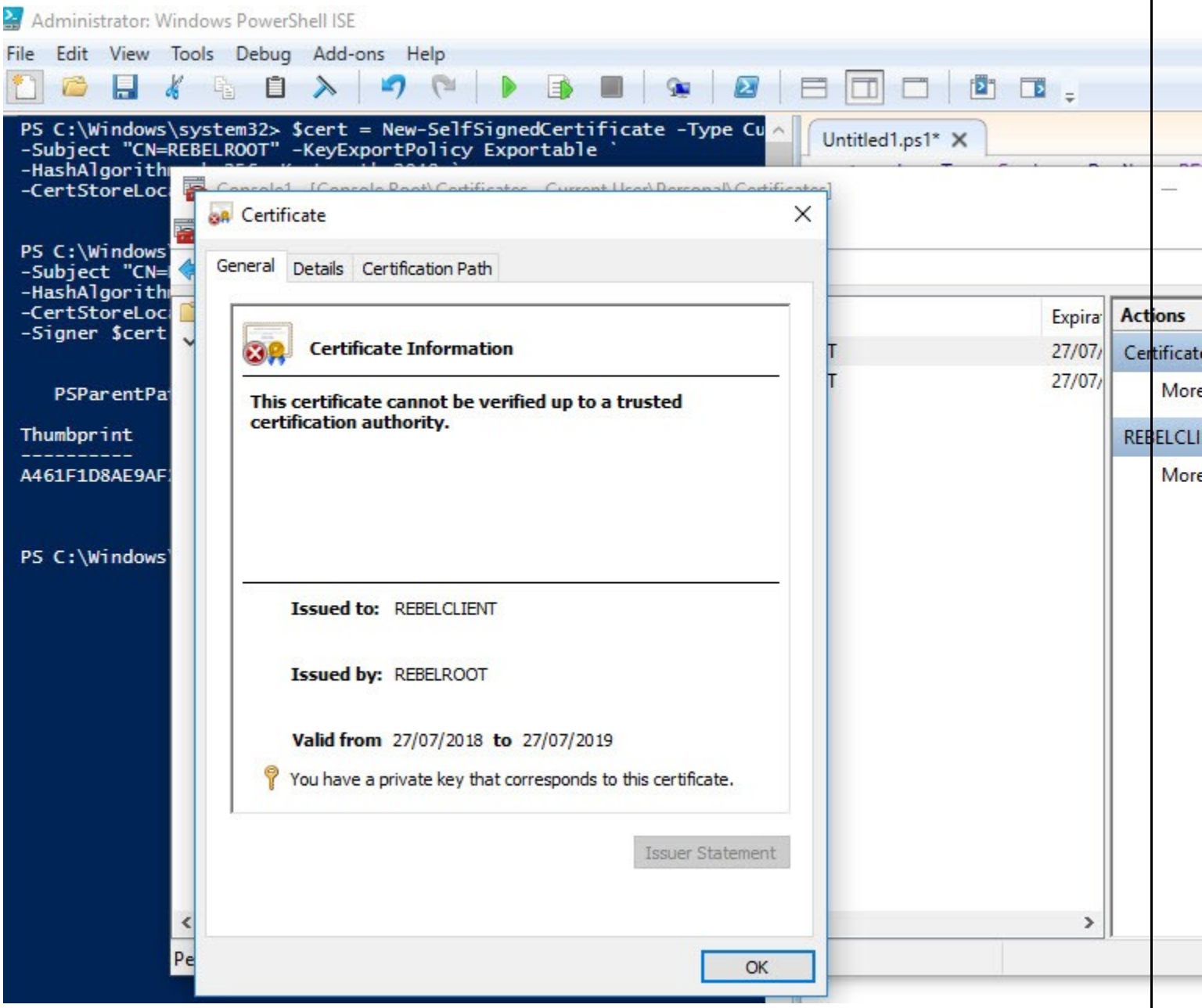


Then we need to create client certificate. We can do this using

New-SelfSignedCertificate -Type Custom -DnsName REBELCLIENT -KeySpec Signature `
-Subject "CN=REBELCLIENT" -KeyExportPolicy Exportable `
-HashAlgorithm sha256 -KeyLength 2048 `

-CertStoreLocation "Cert:\CurrentUser\My" `
-Signer $cert -TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.2")

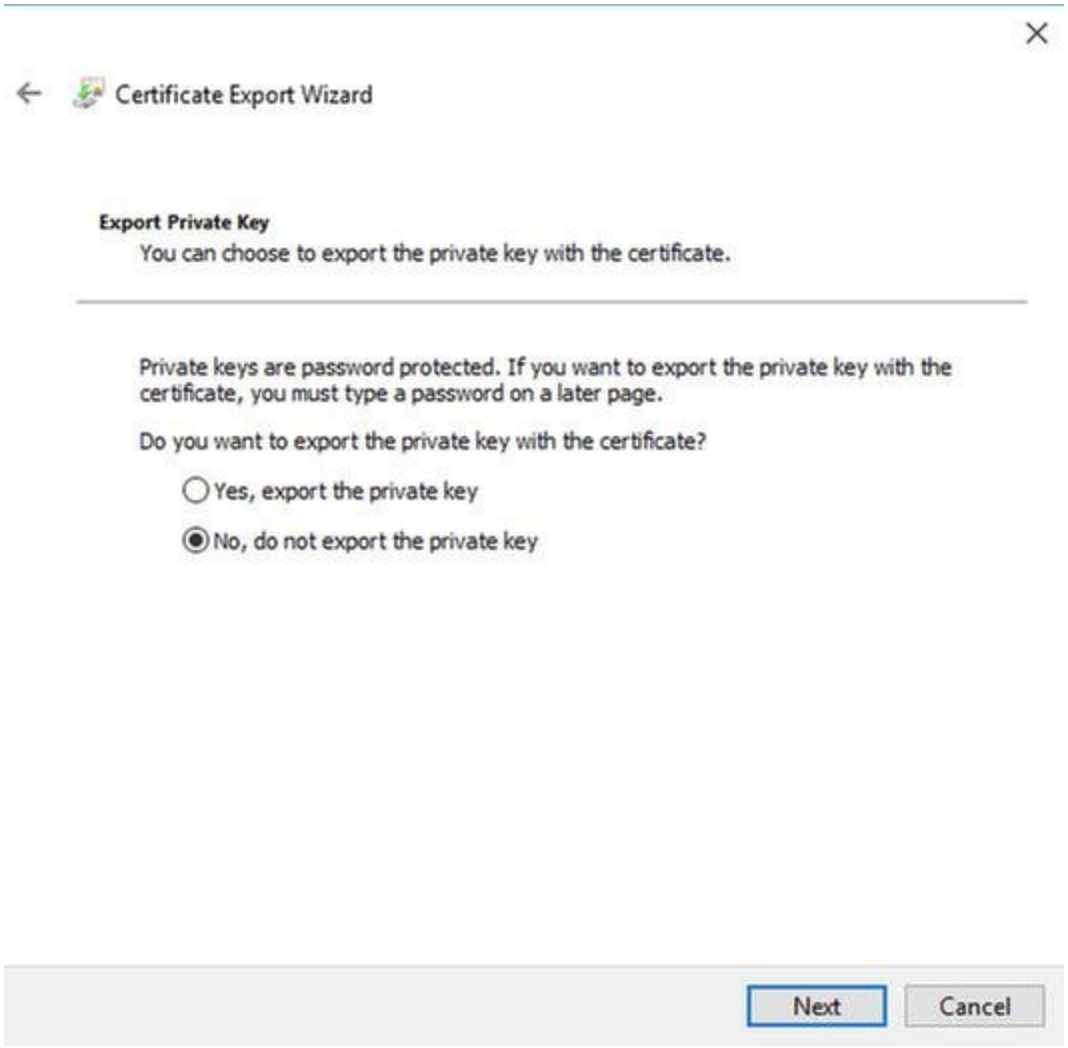
This will create cert called **REBELCLIENT** and install in same store location.



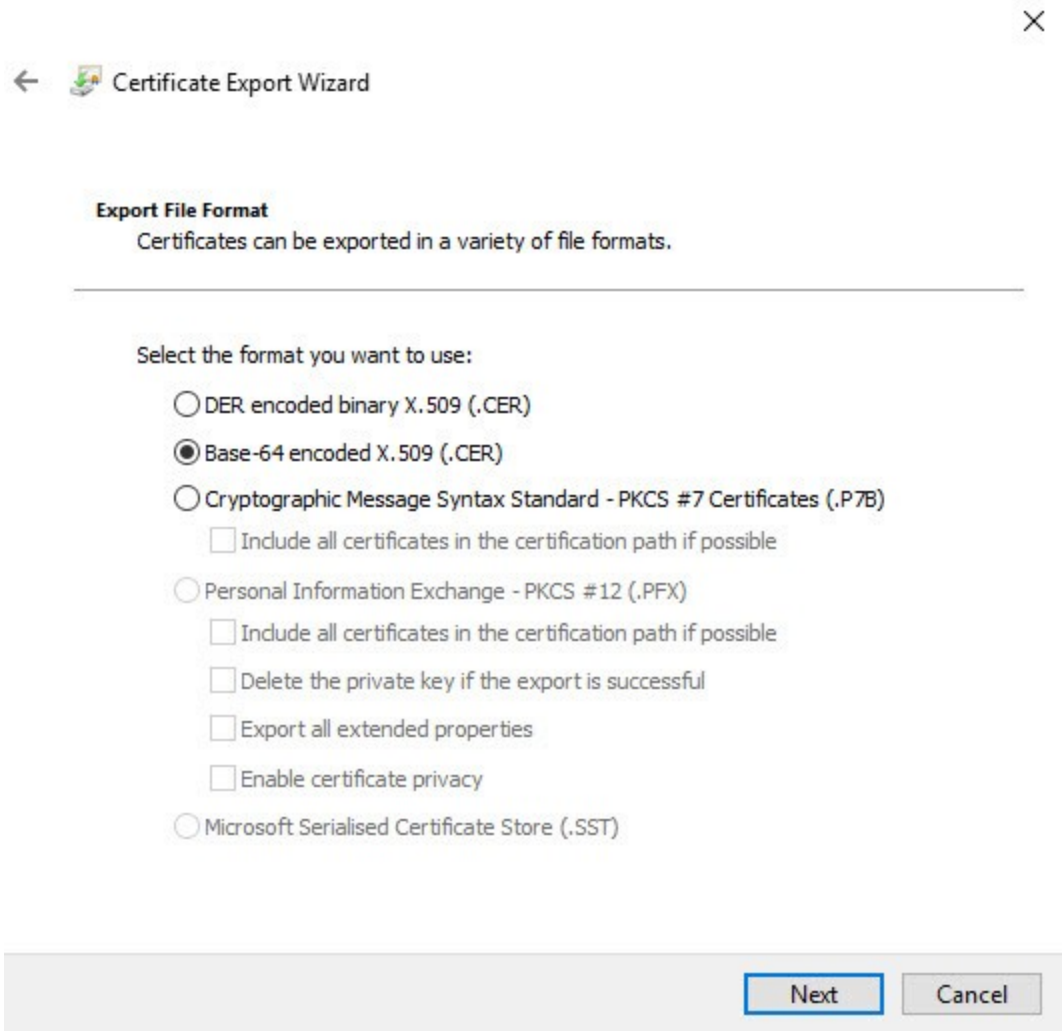
Now we have certs in place. But we need to export these so we can upload it to Azure.

To export root certificate,

- Right click on root cert inside certificate mmc.
- Click on **Export**
- In private key page, select not to export private key



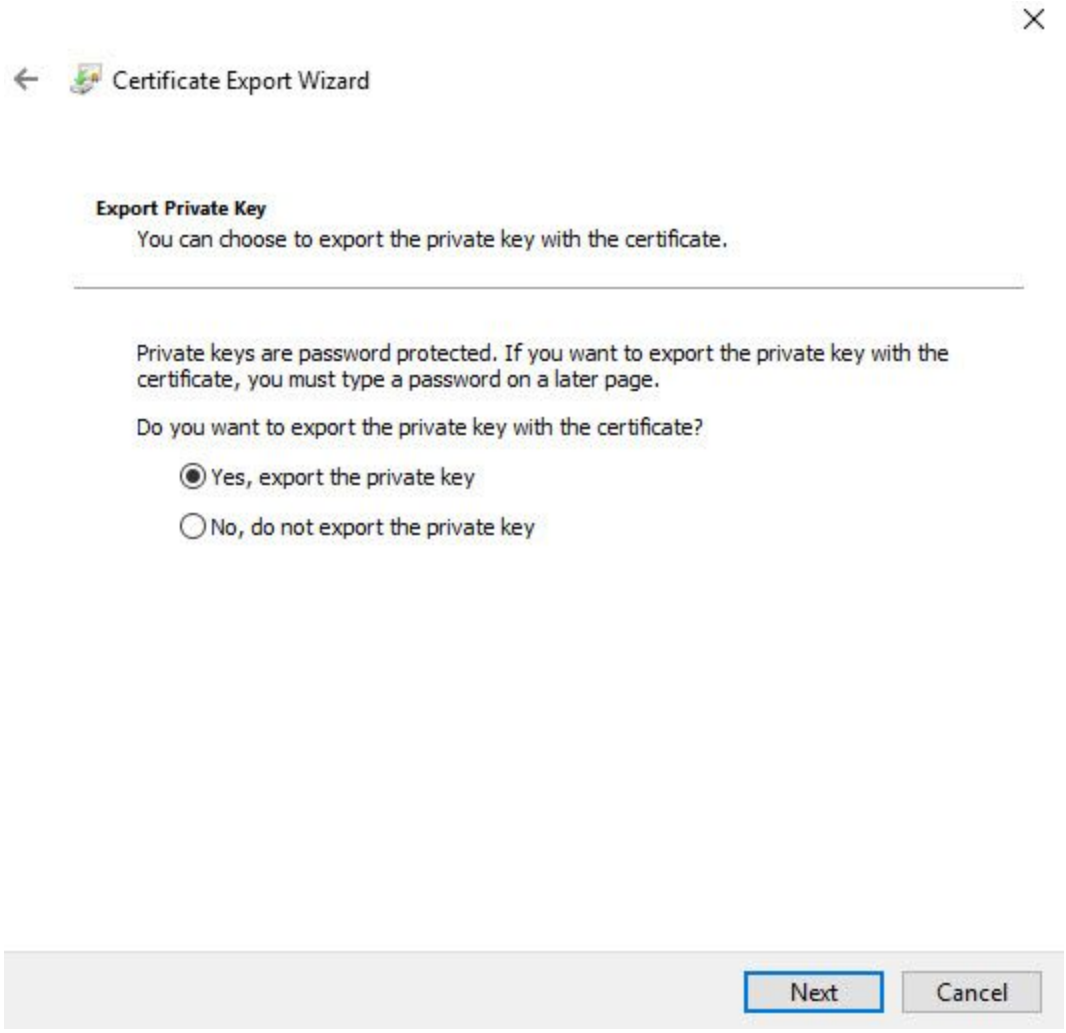
- Select Base-64 encoded X.509 as export file format.



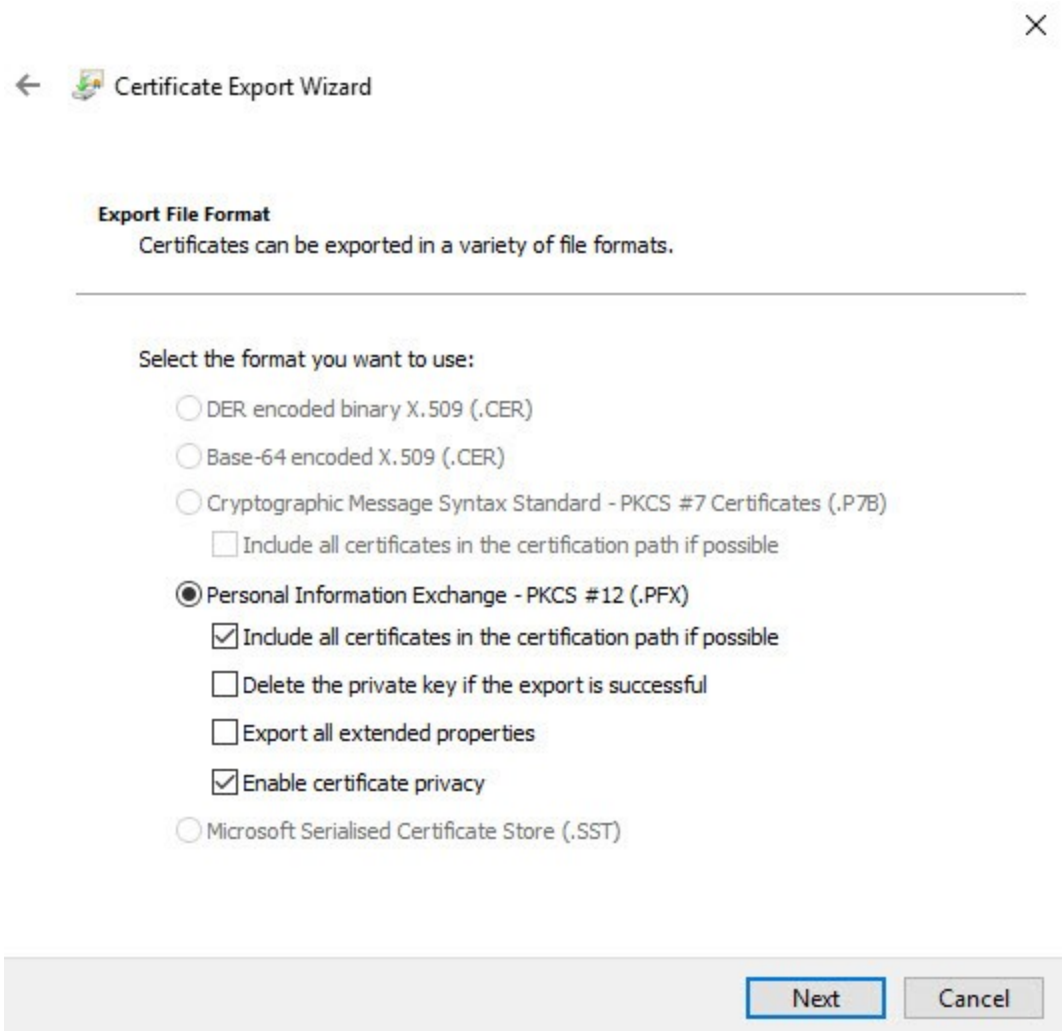
- Complete the wizard and save the cert in pc.

To export client certificate,

- Use same method to export as root cert, but this time under private key page, select option to export private key.



- In file format page, leave the default as following and click **Next**



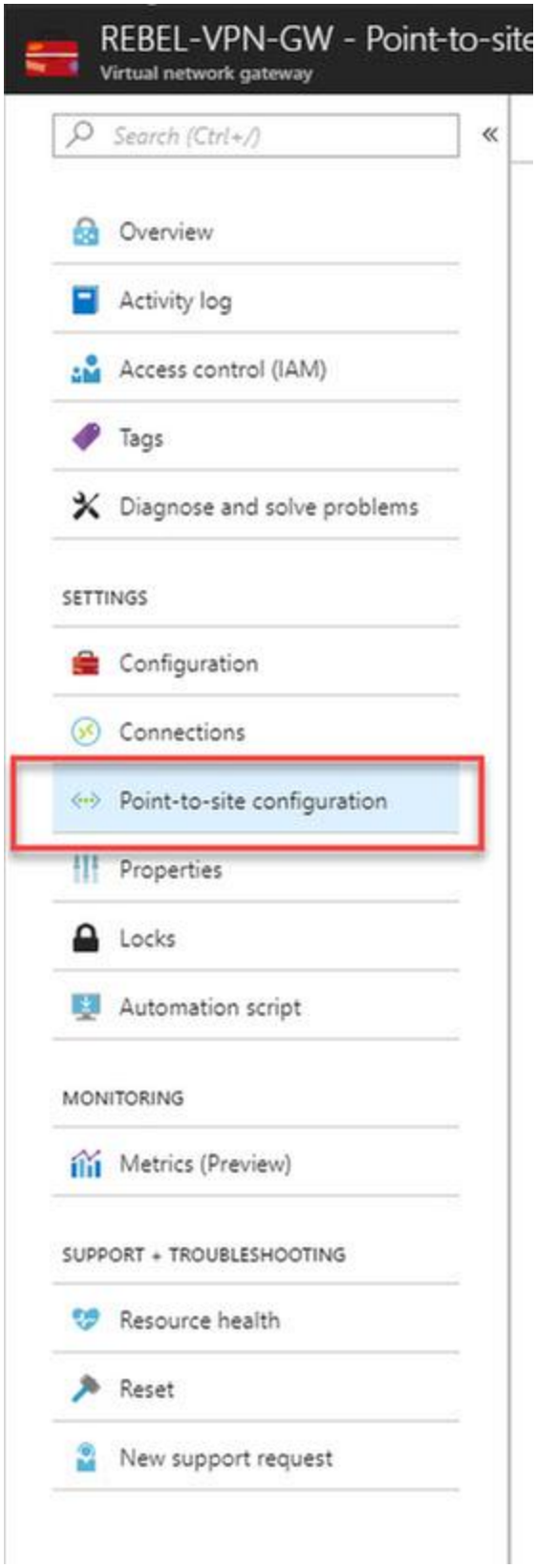
- Define password for the pfx file and complete the wizard.

Note – Only root cert will use in Azure VPN, client certificate can install on other computers which need P2S connections.

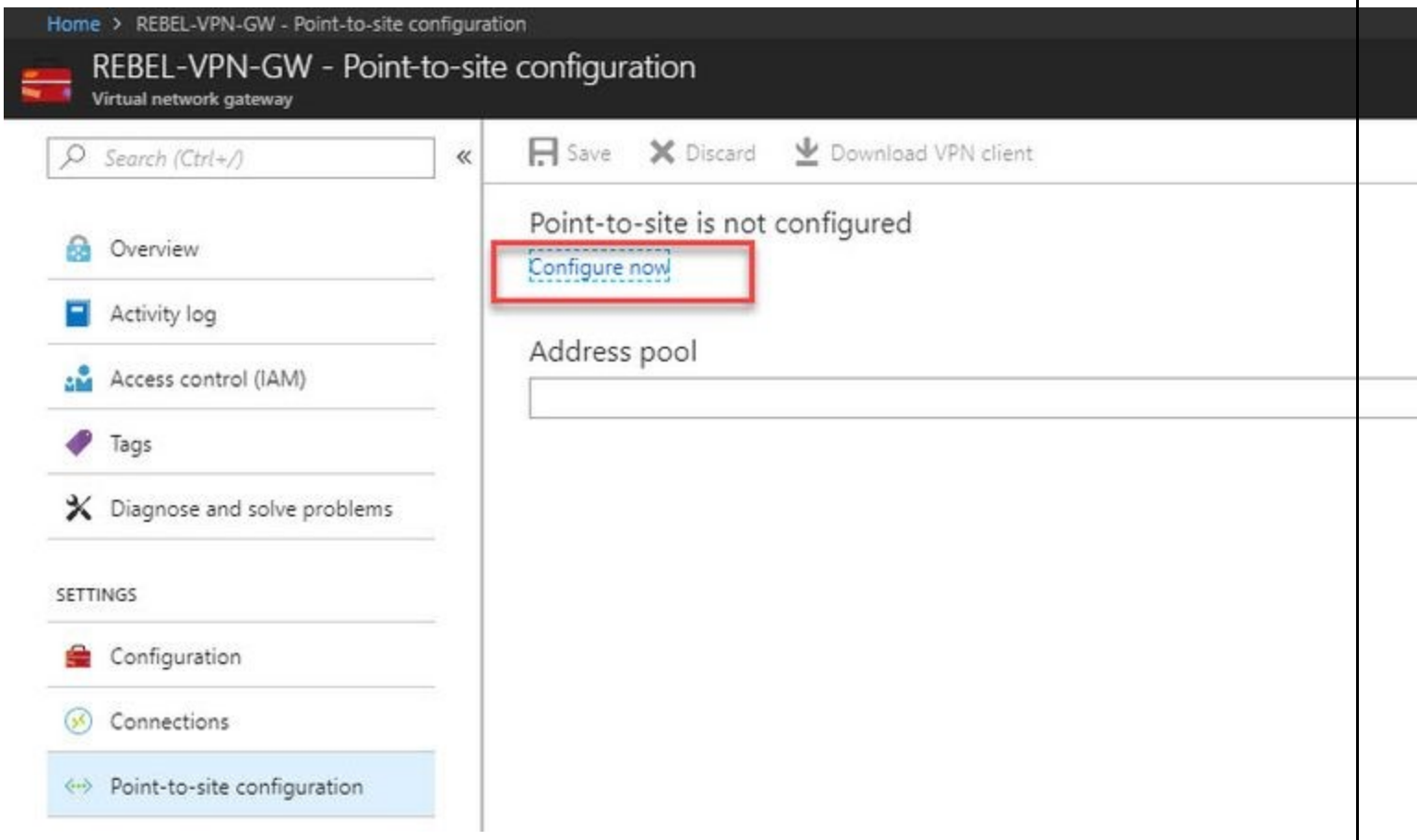
Configure Point-to-Site Connection

Next step of this configuration is to configure the point-to-site connection. In here we will define client ip address pool as well. It is for VPN clients.

- Click on newly created VPN gateway connection.
- Then in new window click on **Point-to-site configuration**

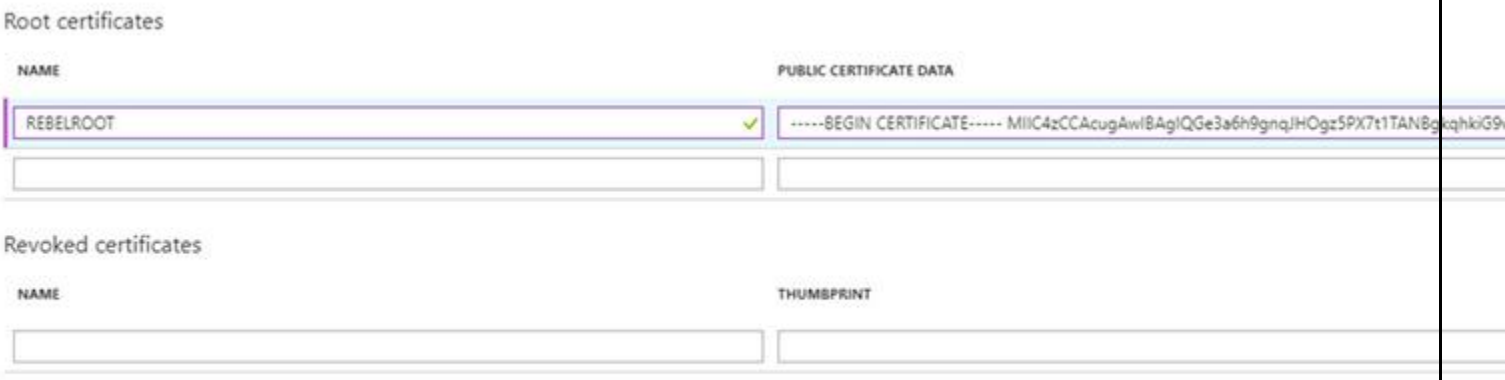


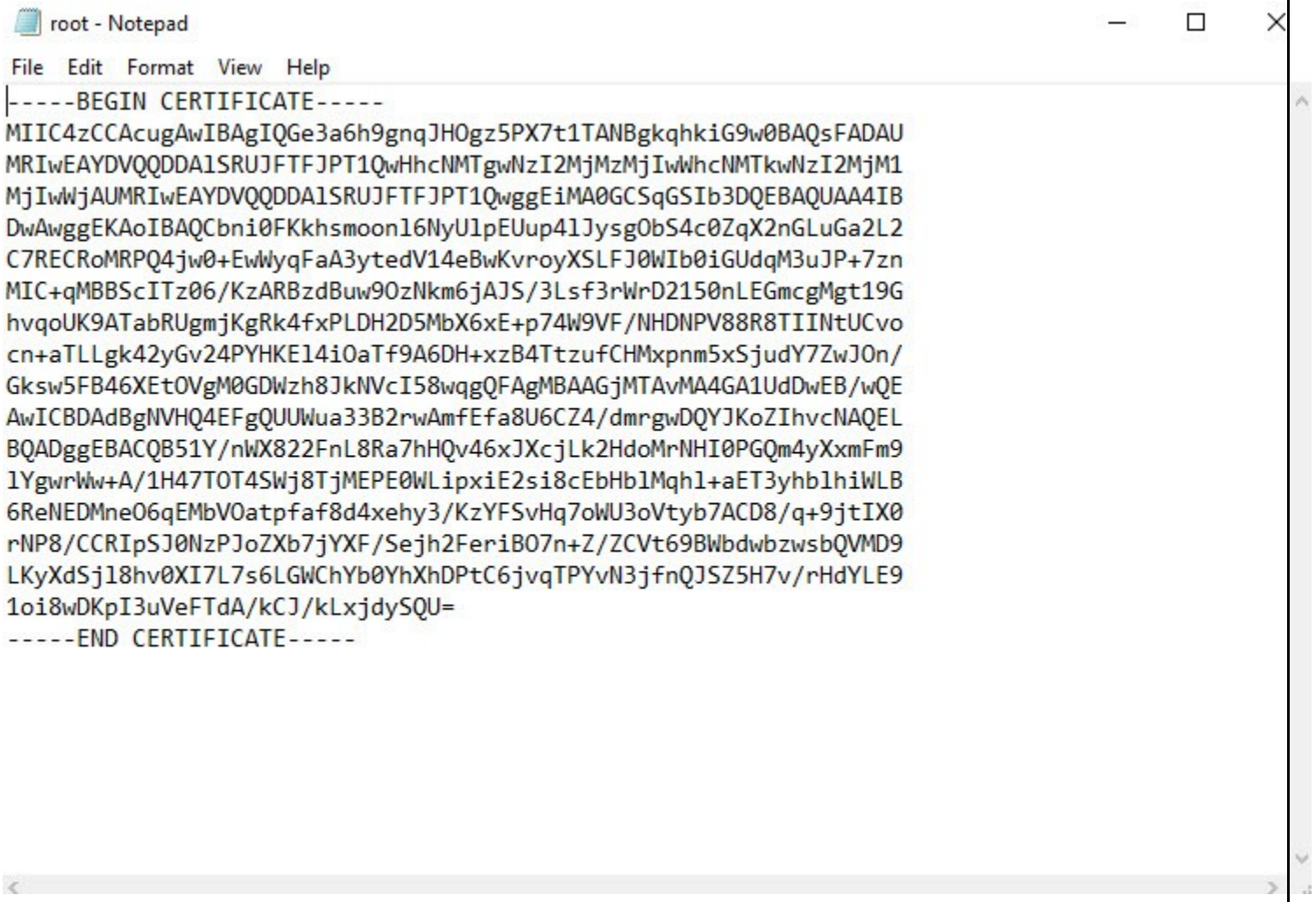
- After that, click on **Configure Now**



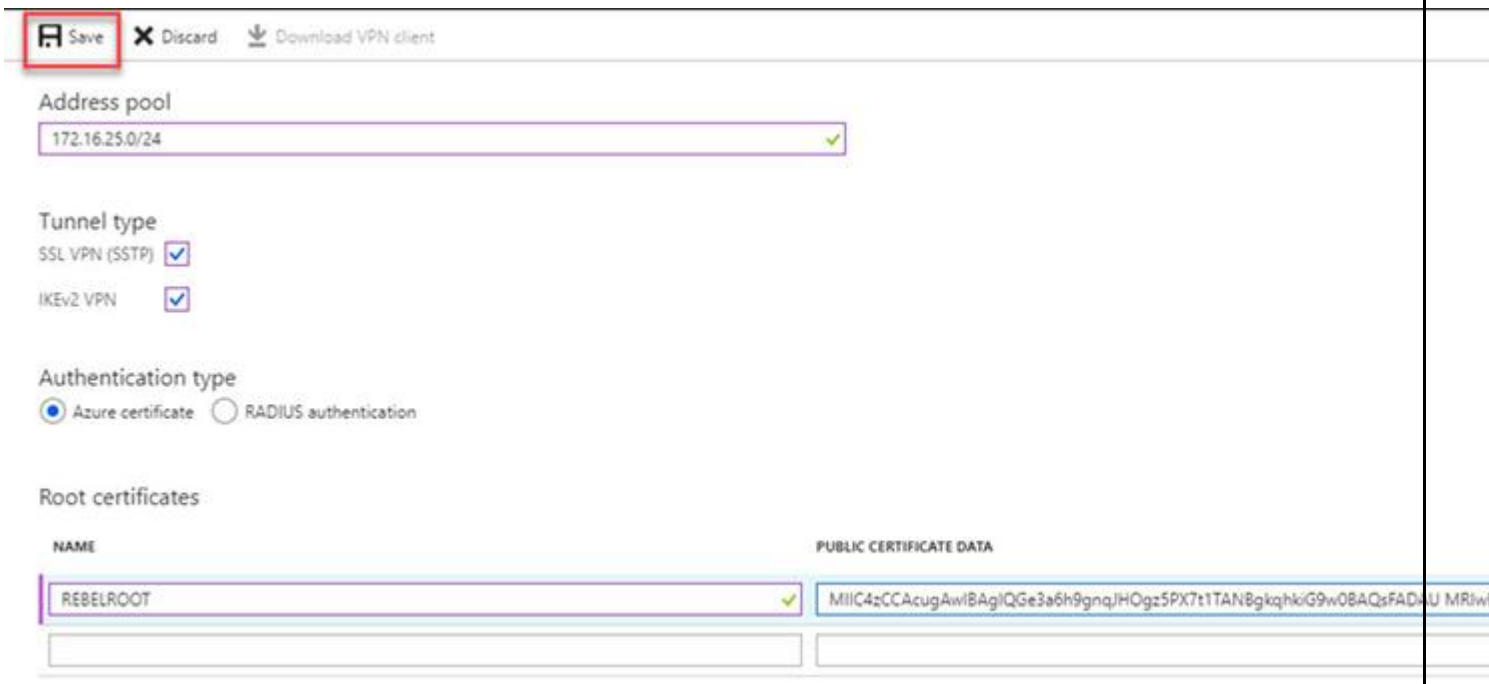
- In new window type IP address range for VPN address pool. In this demo I will be using **172.16.25.0/24**. For tunnel type use both **SSTP & IKEv2**. Linux and other mobile clients by default use **IKEv2** to connect. Windows also use **IKEv2** first and then try **SSTP**. For authentication type use **Azure Certificates**.

- In same window there is place to define root certificate. Under root certificate name type the cert name and under public certificate data, paste the root certificate data (you can open cert in notepad to get data).





- Then click on **Save** to complete the process.

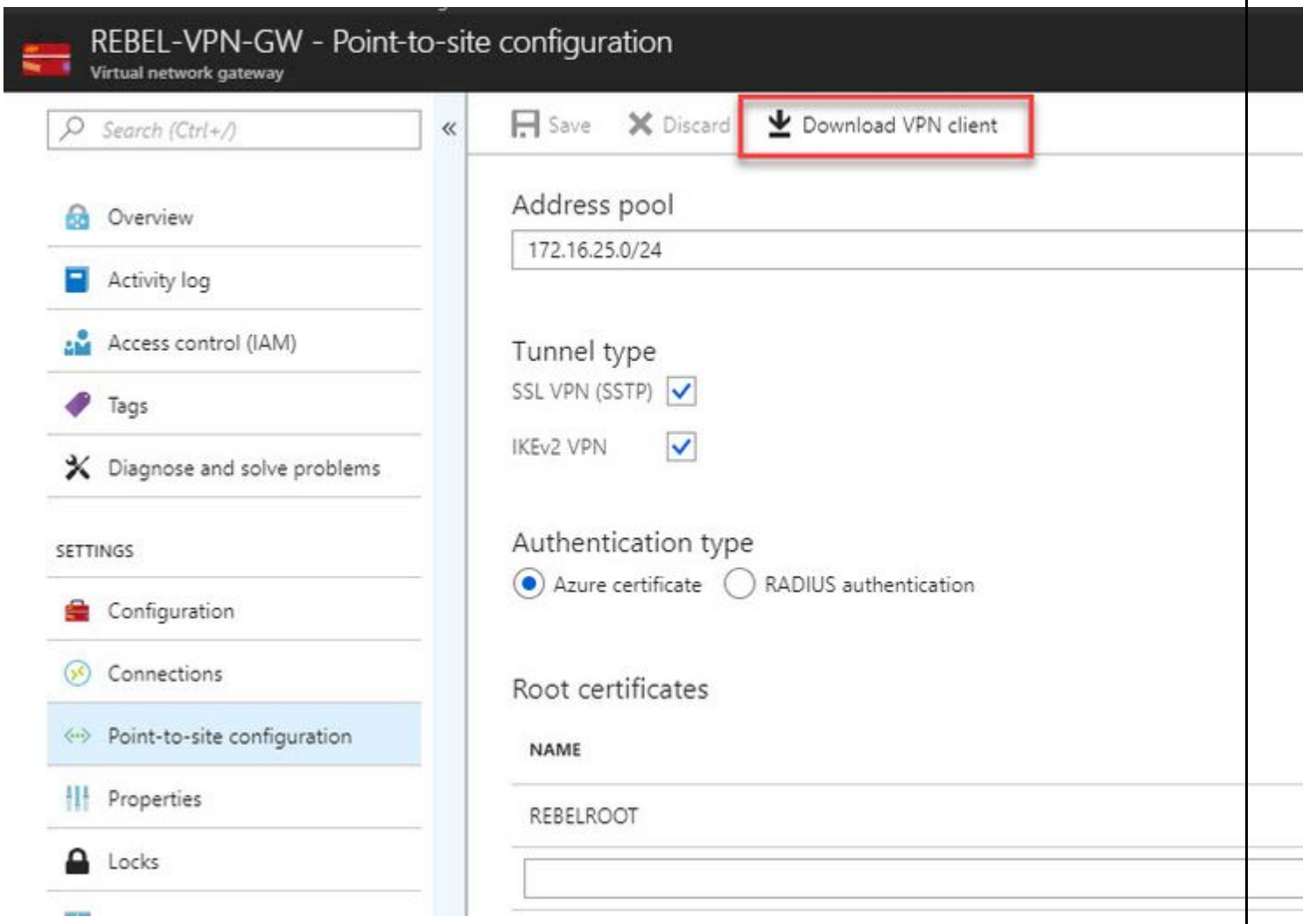


Note : when you paste certificate data, do not copy -----BEGIN CERTIFICATE----- & -----END CERTIFICATE----- text.

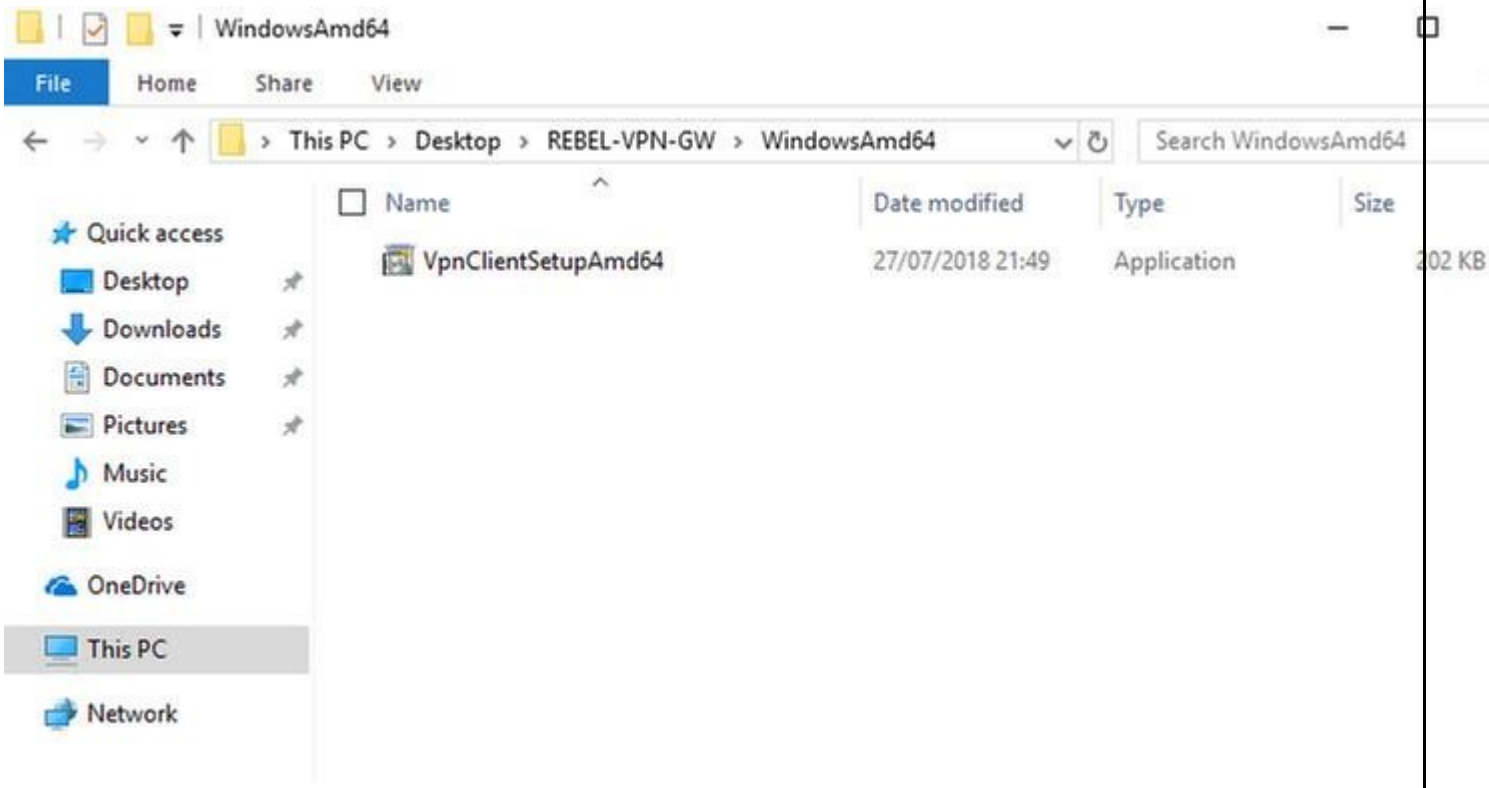
Testing VPN connection

Now we have finished with configuration. As next step, we need to test the connection. To do that log in to the same pc where we generate certificates. If you going to use different PC, first you need to import root cert & client certificate we exported.

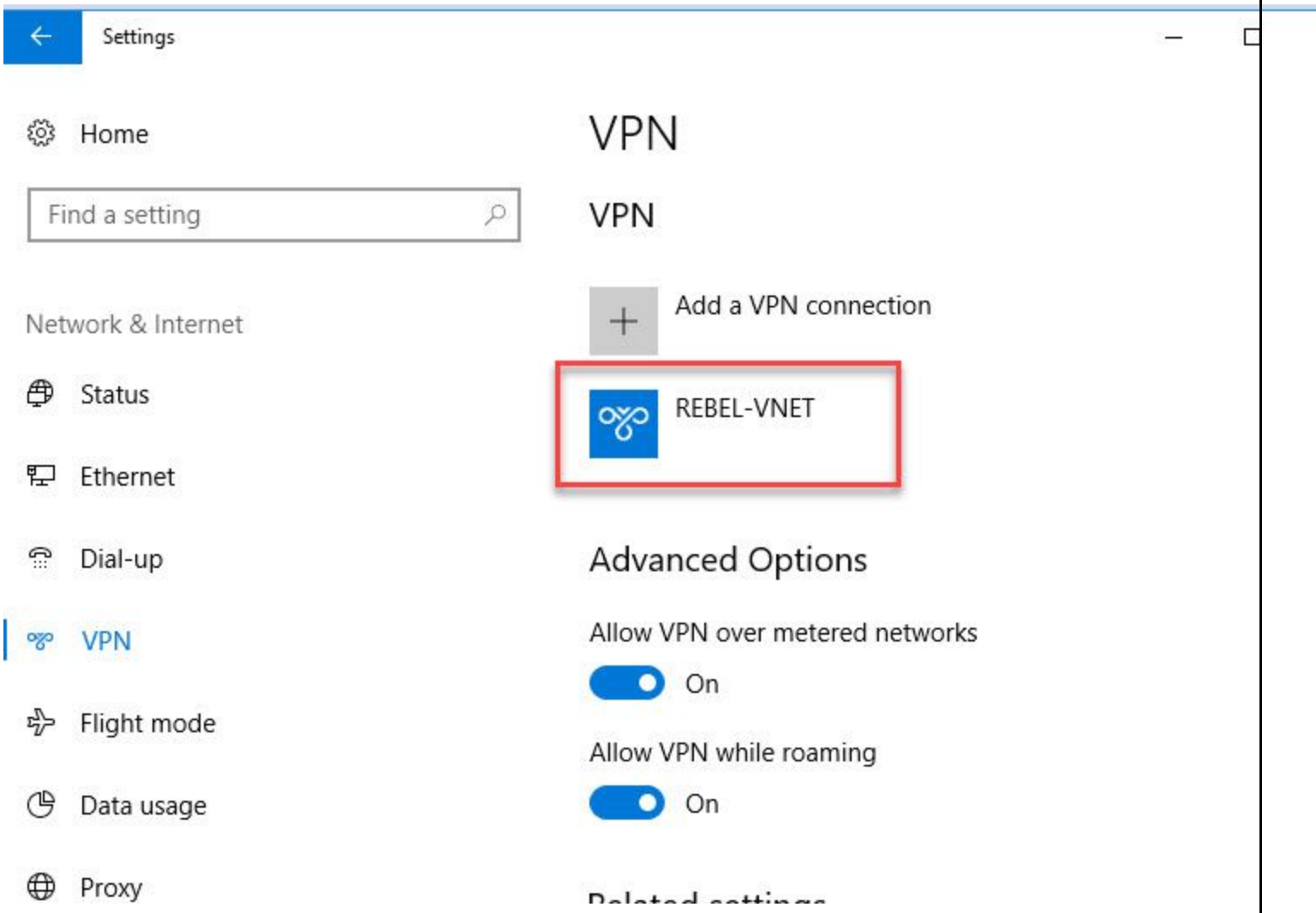
- Log in to Azure portal from machine and go to VPN gateway config page.
- In that page, click on **Point-to-site configuration**
- After that, click on **Download VPN client**



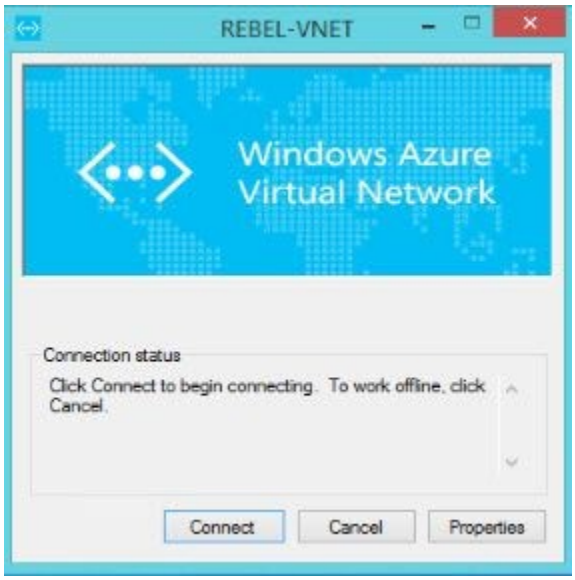
- Then double click on the VPN client setup. In my case I am using 64bit vpn client.



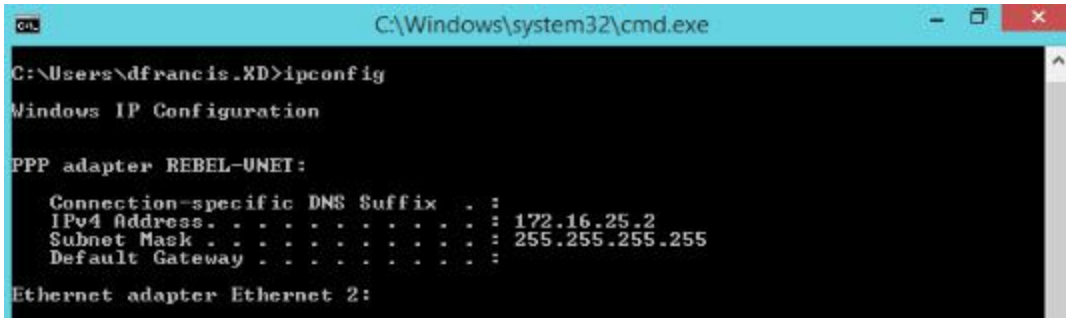
- After that, we can see new connection under windows 10 VPN page.



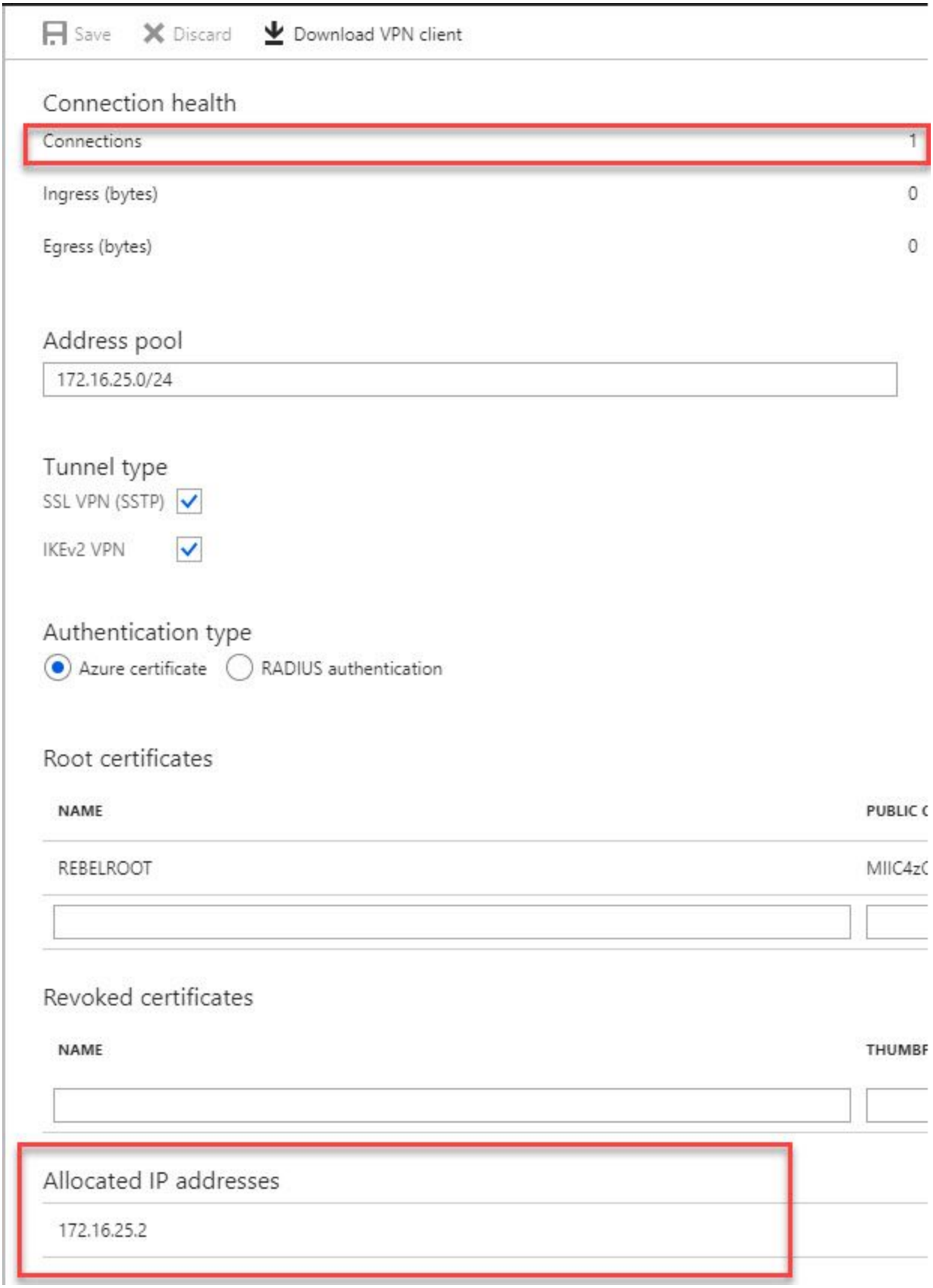
- Click on connect to VPN. Then it will open up this new window. Click on **Connect** in there.



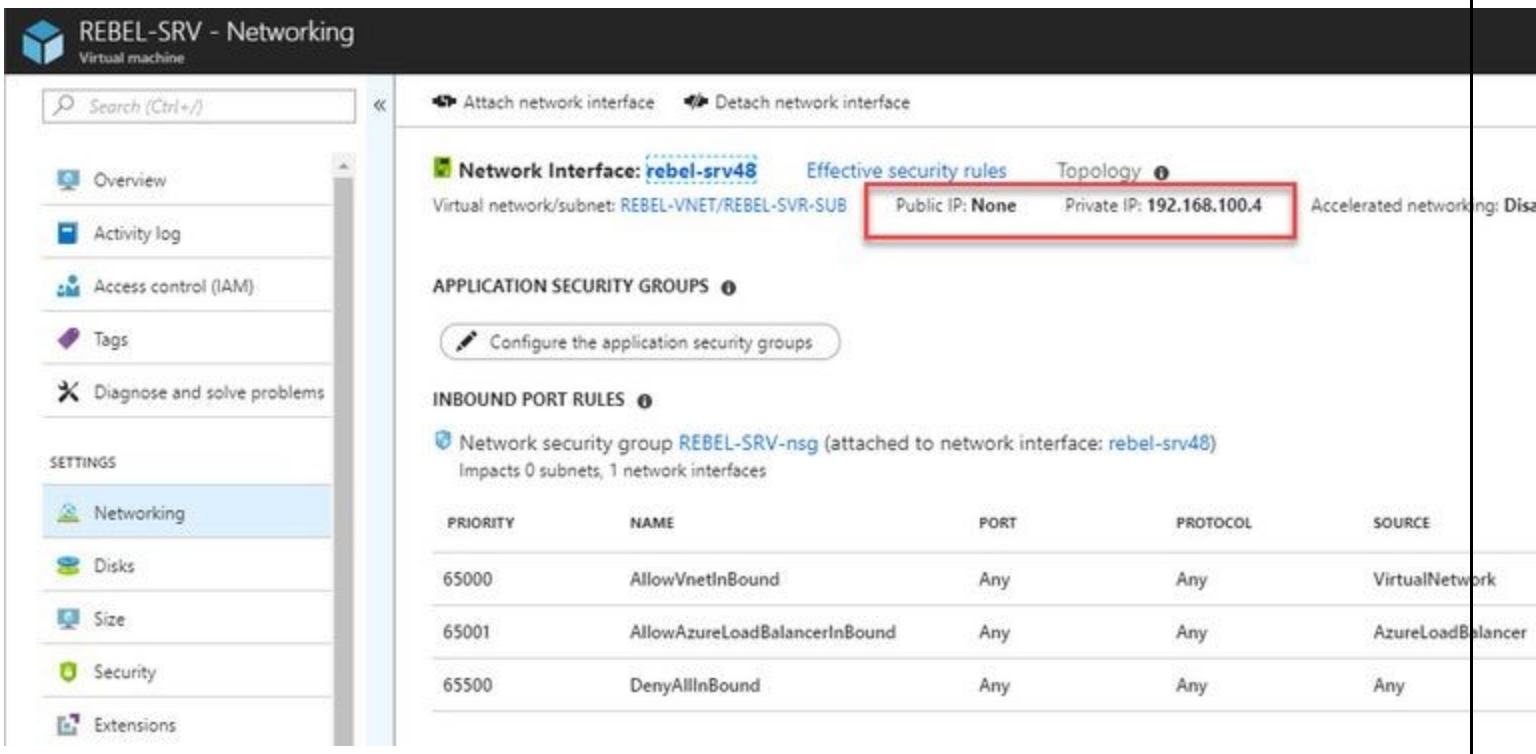
- Then run ip config to verify ip allocation from VPN address pool.



- In VPN gateway page also, I can see one connection is made.



- I have a server setup under new virtual network we created. This server only has private ip and its 192.168.100.4



- As expected, I can RDP to this via VPN.

q)What is the difference between point to site and site to site VPN?

- **Site-to-site** VPNs connect entire networks to each other -- for example, connecting a branch office network to a company headquarters network. **In a site-to-site VPN** configuration, hosts do not have **VPN** client software; they send and receive normal TCP/IP traffic through a **VPN** gateway

Azure Traffic Manager.

The job of Azure Traffic Manager is to route traffic globally based on flexible policies, enabling an excellent user experience that aligns with how you've structured your application across the world. Traffic Manager has several different policies:

Latency. Direct to the "closest service"

Round Robin. Distribute across all services

Failover. Direct to backup if primary fails

Nested. Flexible multi-level policies

Traffic Manager

Traffic Management Polic

Latency – Direct to "closest" servi
Round Robin – Distribute across
Failover – Direct to "backup" if pr
Nested – Flexible multi-level poli

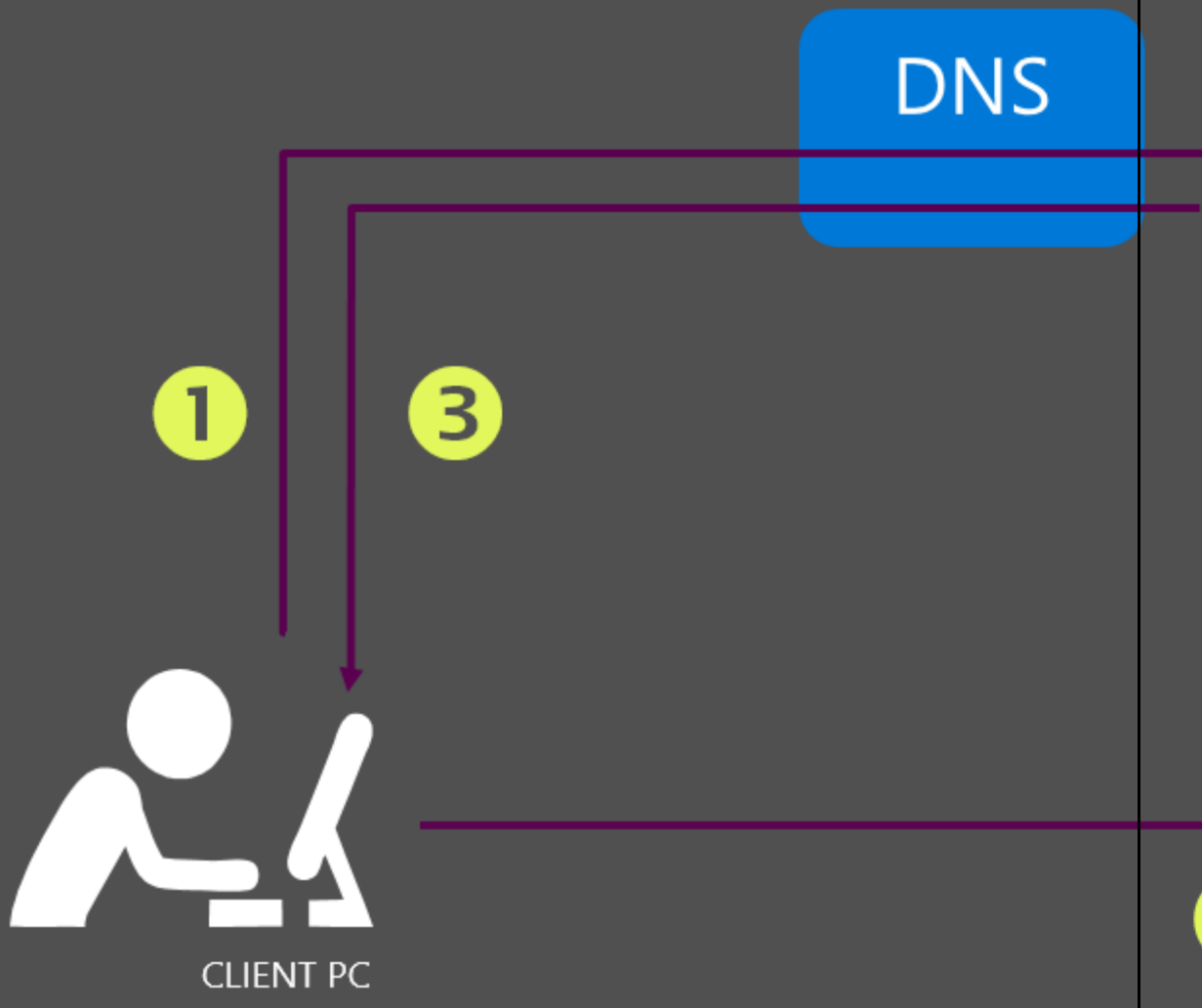
Azure Load Balancer

The job of Azure Load Balancer is to direct traffic inside a region. This is combined with Azure Traffic Manager, where traffic manager routes interior to a region between virtual machines. If you combine the two you get global traffic management combined with local

failover.

Customer Responsibility

Basic Topology



Load Balancer differences There are different options to distribute network traffic using Microsoft Azure. These options work differently from each other, having a different feature set and support different scenarios. They can each be used in isolation, or combining them.

Load Balancer differences

There are different options to distribute network traffic using Microsoft Azure. These options work differently from each other, having a different feature set and support different scenarios. They can each be used in isolation, or combining them.

Azure Load Balancer works at the transport layer (Layer 4 in the OSI network reference stack). It provides network-level distribution of traffic across instances of an application running in the same Azure data center.

Application Gateway works at the application layer (Layer 7 in the OSI network reference stack). It acts as a reverse-proxy service, terminating the client connection and forwarding requests to back-end endpoints.

Traffic Manager works at the DNS level. It uses DNS responses to direct end-user traffic to globally distributed endpoints. Clients then connect to those endpoints directly.

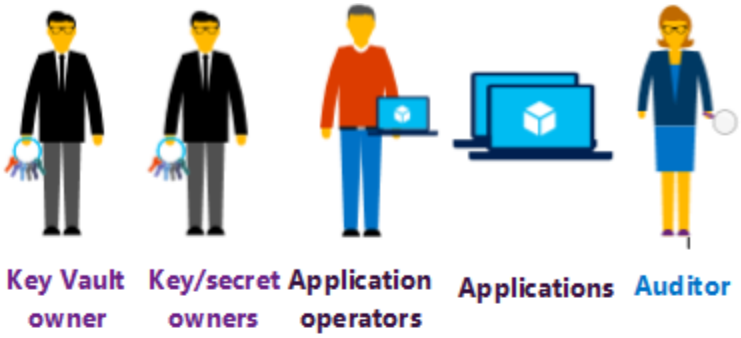
Service	Azure Load Balancer	Application Gateway	Traffic Manager
Technology	Transport level (Layer 4)	Application level (Layer 7)	DNS level
Application protocols supported	Any	HTTP and HTTPS	Any (An HTTP endpoint is required for endpoint monitoring)
Endpoints	Azure VMs and Cloud Services role instances	Any Azure Internal IP address or public internet IP address	Azure VMs, Cloud Services, Azure Web Apps, and external endpoints
Vnet support	Can be used for both Internet facing and internal (Vnet) applications	Can be used for both Internet facing and internal (Vnet) applications	Only supports Internet-facing applications
Endpoint Monitoring	Supported via probes	Supported via probes	Supported via HTTP/HTTPS GET

LINKS:
https://docs.microsoft.com/en-us/azure/traffic-manager/traffic-manager-overview
https://docs.microsoft.com/en-us/azure/load-balancer/load-balancer-overview
https://www.concurrency.com/blog/w/azure-traffic-manager-vs-azure-load-balancer

q)Microsoft Azure Key Vault Lifecycle?

In the previous post of Introduction to Azure Key Vault, we learnt about the basics of Azure Key Vault, keys and Authentication mechanism. In this blog we will be continuing with Key

Vault lifecycle i.e., go through the process of creating and managing Azure Key Vault with PowerShell.



Key Vault lifecycle

Azure Key Vault Lifecycle

1. To use the Key Vault service we have to create the Key Vault which can be done by using the cmdlet **New-AzureRmKeyVault**.
2. It is more than likely that this key vault would have been created for one or more applications to use. You must register those applications in Azure Active Directory. Also one must authorize them to use your vault with the cmdlet **Set-AzureRmKeyVaultAccessPolicy**. Optionally one can use the cmdlet: **Set-AzureRmKeyVaultAccessPolicy** to allow other users (say your team members) to add/remove keys to this key vault, the simple way is to authorize those users for those specific operations on your key vault.
3. An authorized user, can then add secrets (e.g. passwords) and cryptographic keys to the key vault. This can be done with the cmdlet: **Set-AzureKeyVaultSecret** and **Add-AzureKeyVaultKey** respectively. For each secret or key that we add, **we get a unique URI**.
4. The application operator is the one who configures applications to use the URI of your key vault (or of specific secrets or keys). The specific configuration steps are unique to each

application; generally the application will provide you a place in it's configuration to paste the URI of your key or secret.

5. Subsequently, the application we authorized can use the key vault programatically using the Key Vault REST API or Key Vault Client classes. The application can do the following:
   - It can **read or write secrets** into your key vault, in case it is authorized for those operations.
   - It can **use your keys** by calling methods of the service such as **decryptor sign**. The application cannot read (extract) your keys. This is because the Key Vault service performs the cryptographic operation on the application's behalf.
   - The application may also **add**, **remove**, or **update keys** in your key vault, if authorized for those specific operations, but this is less common.
6. The key vault owner, or a delegate such as an auditor, will be able to monitor operations performed on your keys and secrets by retrieving a usage log from the Key Vault service link.
7. At any point, we can update values of existing keys or secrets by using the cmdlet **Add-AzureKeyVaultKey** with the existing key name. **Set-AzureKeyVaultKey** can be used with the existing secret name.
8. When you are done using your keys and secrets, delete them using the cmdlets **Remove-AzureKeyVaultKey** and **Remove-AzureKeyVaultSecret** respectively.
9. At any point, you can revoke users and applications from accessing your key vault, or authorize other users and applications by using the cmdlets **Remove-AzureRmKeyVaultAccessPolicy** and **Set-AzureRmKeyVaultAccessPolicy**.
10. When you are done using the key vault, delete the entire key vault using the cmdlet **Remove-AzureRmKeyvault**.

For step-by-step instructions to create your first key vault and authorize an application using **PowerShell** and **REST APIs** stay tuned for next part

Step-by-Step Guide: Azure Key Vault

JUNE 10, 2018 BY DISHAN M. FRANCIS NO COMMENTS

People use safes, security boxes to protect their valuable things. In digital world "Data" is the most valuable thing. Passwords, Connection Strings, Secrets, Data encryption/decryption keys protects access to different data sets. Whoever have access to those will also have access to data behind it (of cause they need to know how to use those ?). So how we can protect those valuable info? People use different methods. Some use third party software installed on PC to do it. If its large environment some use web application so multiple people have access to it. different vendors use different methods to protect these types of valuable data. Microsoft Azure Key vault is a service which we can use to protect Passwords, Connection Strings, Secrets, Data encryption/decryption keys uses by cloud applications and services. Keys stored in vaults is protected by hardware security modules (HSMs). It is

also possible to import or generate keys using HSMs. Any keys process that way will be done according to **FIPS 140-2 Level 2** guidelines. You can find about FIPS 140-2 Level 2 using https://www.microsoft.com/en-us/trustcenter/Compliance/FIPS

Benefits of using Key Vault

- Keys saved in vault will be served via URLs. Developers, engineers do not need worry about securing keys. Application or service do not see the keys as vault service process behalf of them.
- Customers do not have to disclosure their keys to vendors or service providers. They can manage their own keys and allow to access those keys via urls in vendor or service provider applications. Vendor or service providers will not see the keys.
- By design Microsoft can't extract or see customer keys. So, its further protected in vendor level too.
- HSMs are FIPS 140-2 Level 2 validated. So, any industry required to comply with these standards are protected by default.
- Key usage details are logged. So, you know what's happening with your keys.

An Azure Administrator is allowed to do following using Azure Key Vault,

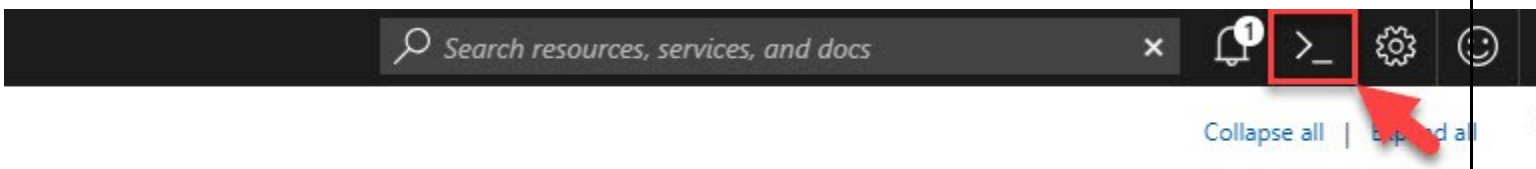
- Create or import a key or secret
- Revoke or delete a key or secret
- Authorize users or applications to access the key vault, which allow them to manage or use its own keys and secrets
- Configure key usage
- Record key usage

More info about Azure Key vault can find under https://docs.microsoft.com/en-us/azure/key-vault/key-vault-overview

Let's go ahead and see how we can setup and use Azure Key Vault service.

Create Azure Key Vault Instance

1) Log in to **Azure Portal** as global admin.
2) Click on **Cloud Shell** icon in top right-hand corner. (You also can setup this using portal, Azure CLI or locally installed Azure PowerShell. In this demo I am using Azure PowerShell directly from portal)



3) Then select **PowerShell** for the command type.
4) Then type **Get-AzureRmResourceGroup** to list down resource groups. So, we can select the resource group to associate the new key vault.

```
VERBOSE: Building your Azure drive ...
Azure:\
PS Azure:\> Get-AzureRmResourceGroup


ResourceGroupName : AppPool
Location          : southcentralus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/d7bfdffe-cfb6-4410-91a6-be45f88aab4d/resourceGroups/AppPool

ResourceGroupName : AzureBackupRG_westus_1
Location          : westus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/d7bfdffe-cfb6-4410-91a6-be45f88aab4d/resourceGroups/AzureBac

ResourceGroupName : cloud-shell-storage-northeurope
Location          : northeurope
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/d7bfdffe-cfb6-4410-91a6-be45f88aab4d/resourceGroups/cloud-sh

ResourceGroupName : Default-Networking
Location          : eastus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/d7bfdffe-cfb6-4410-91a6-be45f88aab4d/resourceGroups/Default-

ResourceGroupName : Default-Storage-EastUS
Location          : eastus
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/d7bfdffe-cfb6-4410-91a6-be45f88aab4d/resourceGroups/Default-
```

5) If you wish to create key vault under new resource group, you can do it using

New-AzureRmResourceGroup -Name RGName -Location WestUS

In above command **RGName** specify the resource group name and **WestUS** define the region. You can find the available locations using **Get-AzureRmLocation**

6) Now it's time to create the vault. We can create it using,

New-AzureRmKeyVault -VaultName 'Rebel-KVault1' -ResourceGroupName 'therebeladmin' -Location 'North Central US'

In above **VaultName** defines the Key Vault name. **ResourceGroupName** defines the resource group it is associated with. **Location** defines the location of resource.

```
Azure:\
PS Azure:\> New-AzureRmKeyVault -VaultName 'Rebel-KVault1' -ResourceGroupName 'therebeladmin'


Vault Name                      : Rebel-KVault1
Resource Group Name             : therebeladmin
Location                        : North Central US
Resource ID                     : /subscriptions/[redacted]/resource
Vault URI                       : https://[redacted].vault.azure.net/
Tenant ID                       : [redacted]
SKU                             : Standard
Enabled For Deployment?         : False
Enabled For Template Deployment? : False
Enabled For Disk Encryption?    : False
Soft Delete Enabled?            :
Access Policies                 :
Network Rule Set                :
                                  Default Action                    : Allow
                                  Bypass                            : AzureServices
                                  IP Rules                          :
                                  Virtual Network Rules             :

Tags                            :
```

7) We can view properties of existing key vault using,

Get-AzureRmKeyVault "Rebel-KVault1"

In above **Rebel-KVault1** is the key vault name.

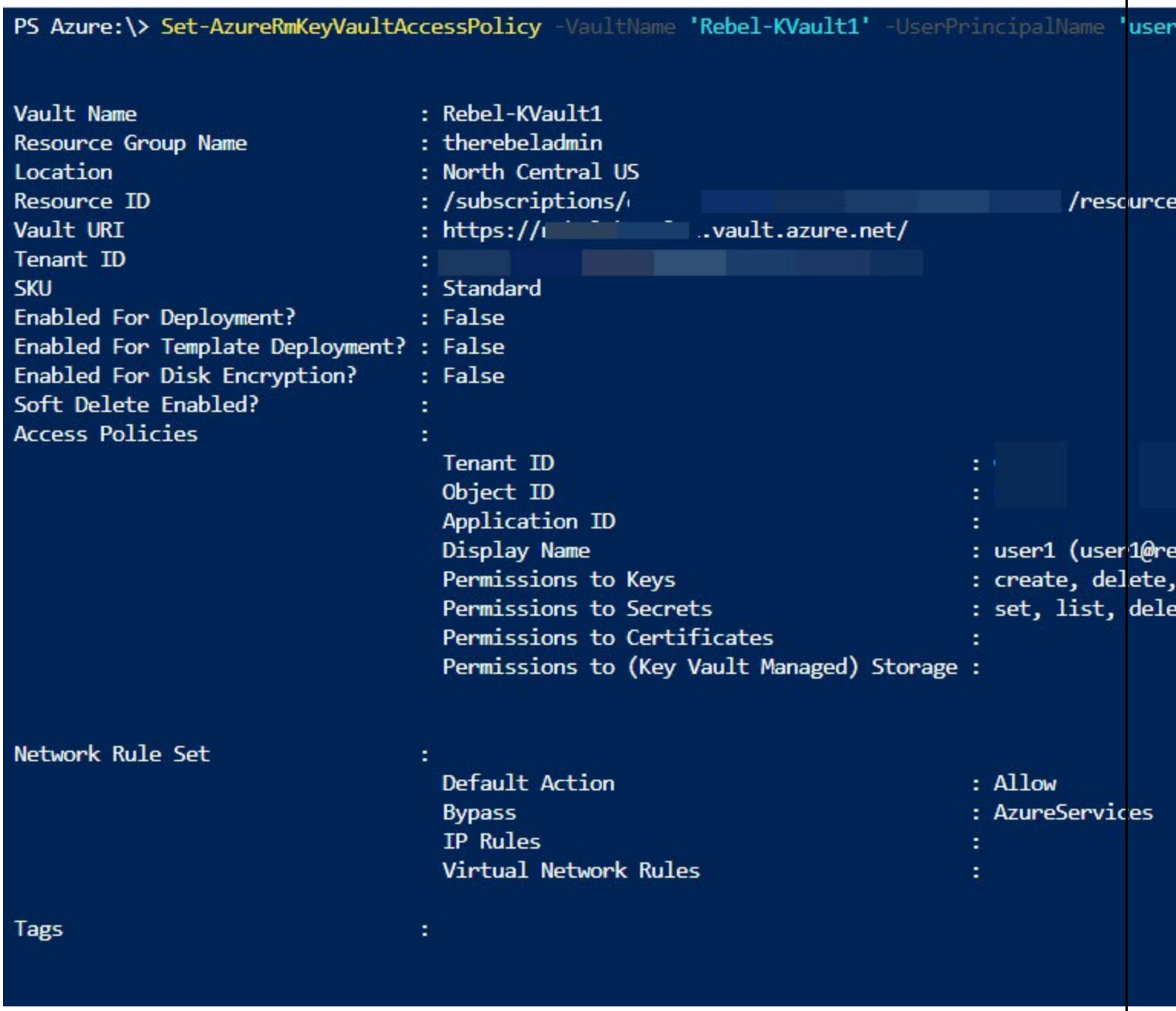
```
Azure:\
PS Azure:\> Get-AzureRmKeyVault "Rebel-KVault1"


Vault Name                      : Rebel-KVault1
Resource Group Name             : therebeladmin
Location                        : North Central US
Resource ID                     : /subscriptions/[redacted]/resource
Vault URI                       : https://[redacted]1.vault.azure.net/
Tenant ID                       : [redacted]
SKU                             : Standard
Enabled For Deployment?         : False
Enabled For Template Deployment? : False
Enabled For Disk Encryption?    : False
Soft Delete Enabled?            :
Access Policies                 :
Network Rule Set                :
                                  Default Action                    : Allow
                                  Bypass                            : AzureServices
                                  IP Rules                          :
                                  Virtual Network Rules             :

Tags                            :
```

Vault URI shows the URL which can use to access the key vault by applications and services.

8) Next step is to create **Access Policy** for the key vault. Using access policy we can define who have control over key vault, what they can do inside key vault and also what a application or service can do with it.

Set-AzureRmKeyVaultAccessPolicy -VaultName 'Rebel-KVault1' -UserPrincipalName 'user1@rebeladmlive.onmicrosoft.com' -PermissionsToKeys create,delete,list -PermissionsToSecrets set,list,delete -PassThru

In above command, **user1@rebeladmlive.onmicrosoft.com** can create,delete,list keys in **Rebel-KVault1**. He also can set,list,delete secrets under same vault.



We also can set permissions for application to retrieve secrets or keys.

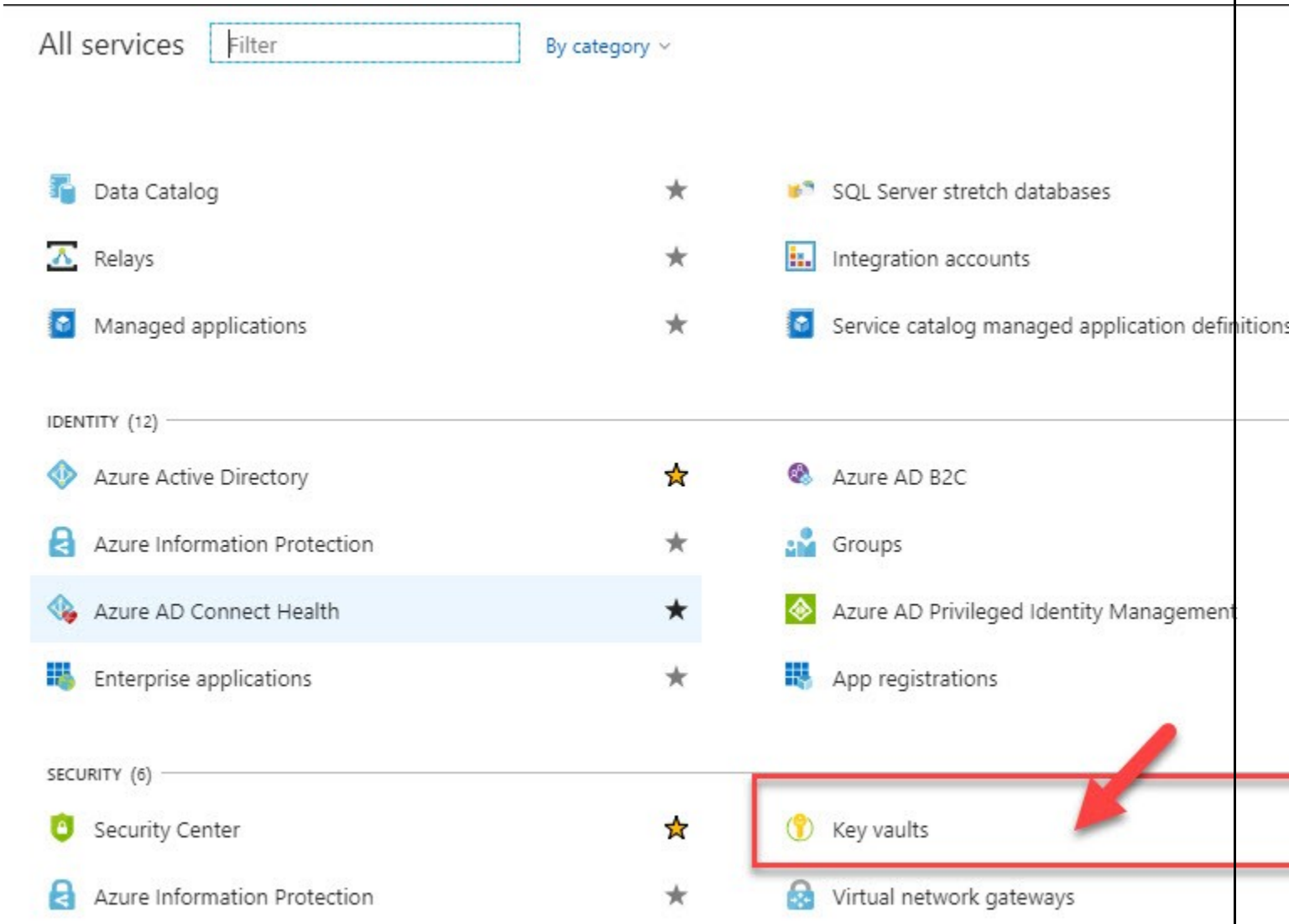
Set-AzureRmKeyVaultAccessPolicy -VaultName 'Rebel-KVault1' -ServicePrincipalName 'http://crm.rebeladmin.com' -PermissionsToSecrets Get

In above, service running on http://crm.rebeladmin.com will have permissions to retrieve secrets from the vault.

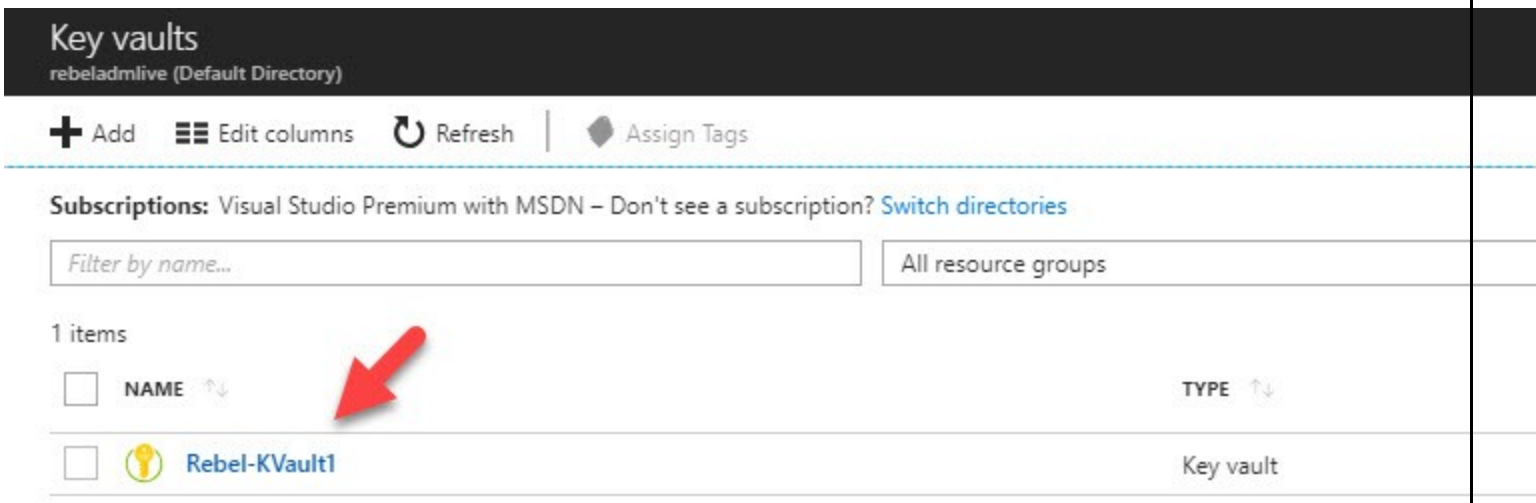
Key Management

Now we have a vault up and running. Next step is to see how to manage valued data using it. In this demo I am going to do this using Azure Portal. Same tasks still can be done using Azure CLI or Azure PowerShell.

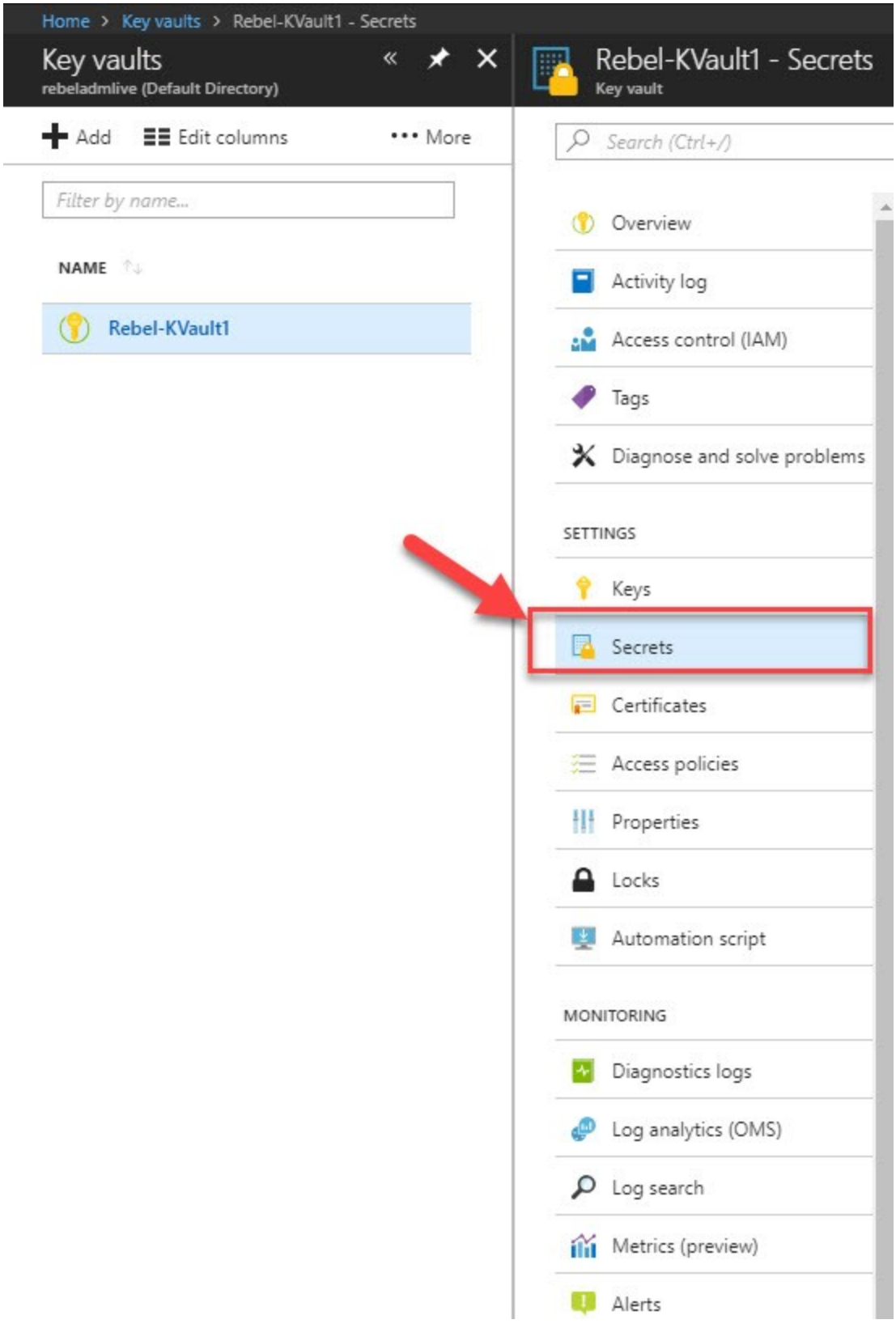
1) To access Key vault feature in portal, go to **Azure Portal > All Services > Key vaults**



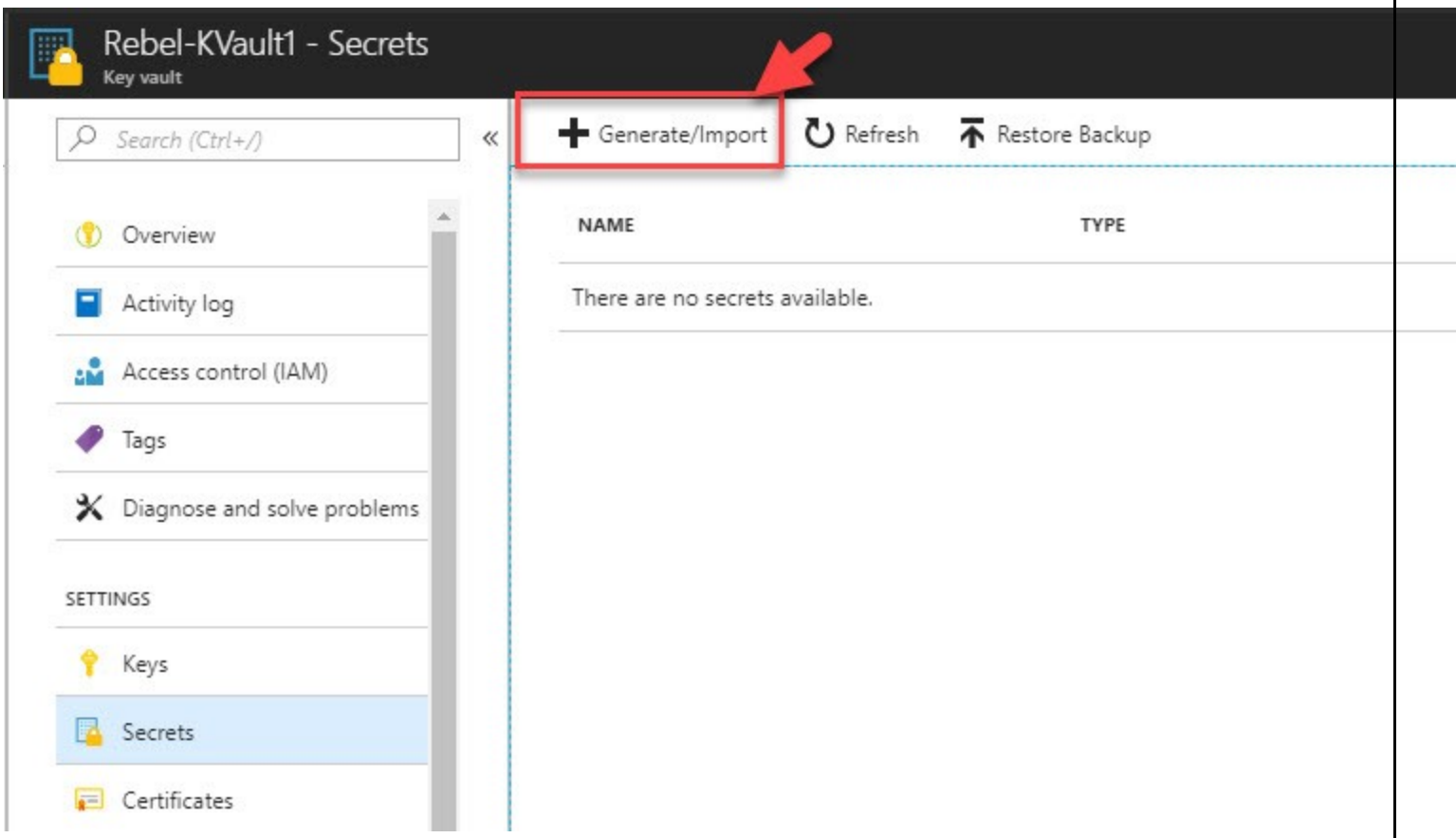
2) Then click on the relevant key vault from the list. In my demo it is **Rebel-KVault1** which we create on previous section.



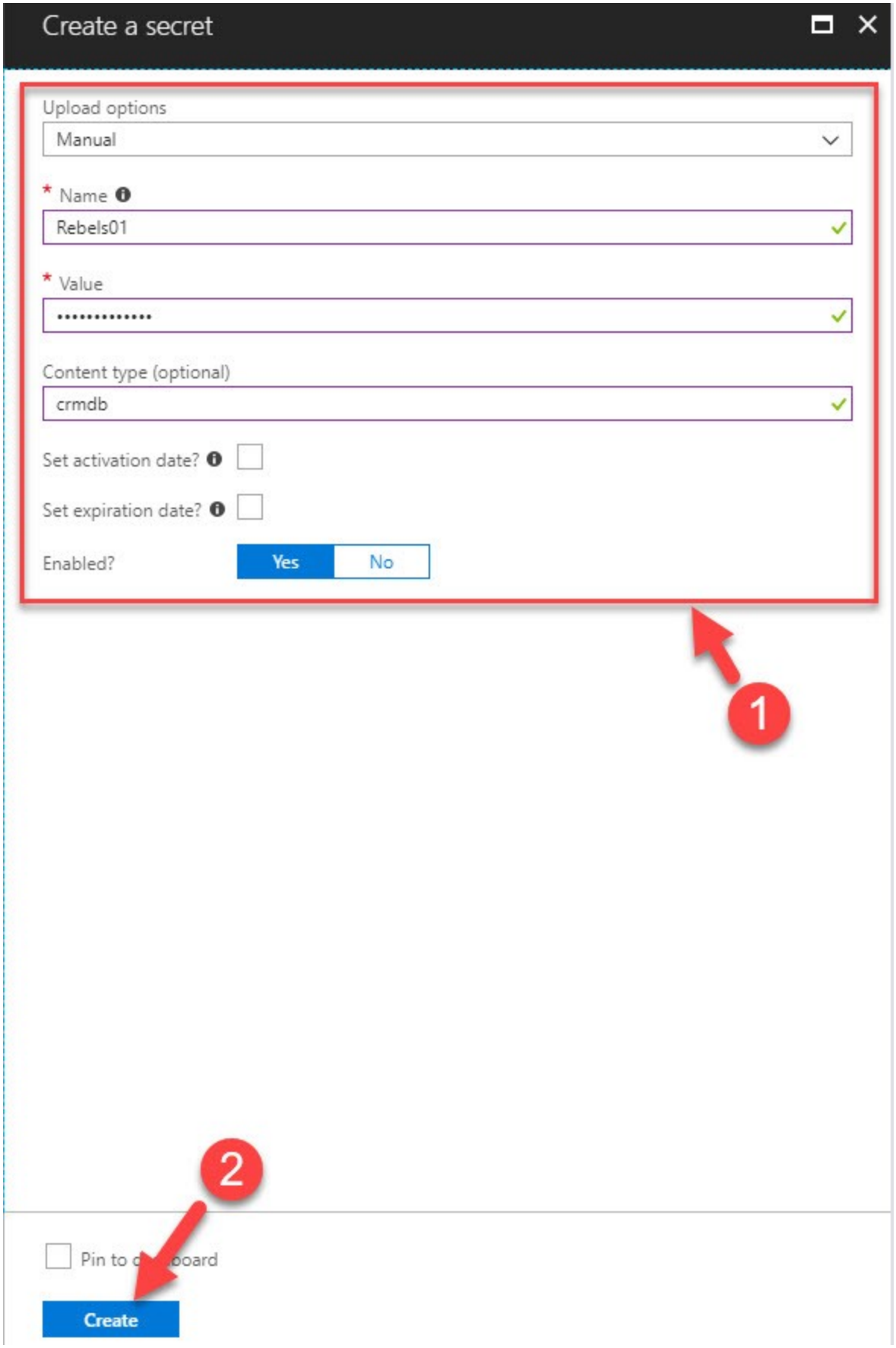
3) Then it will load new window. Let's go ahead and add a secret. To do that click on the **Secrets** option.



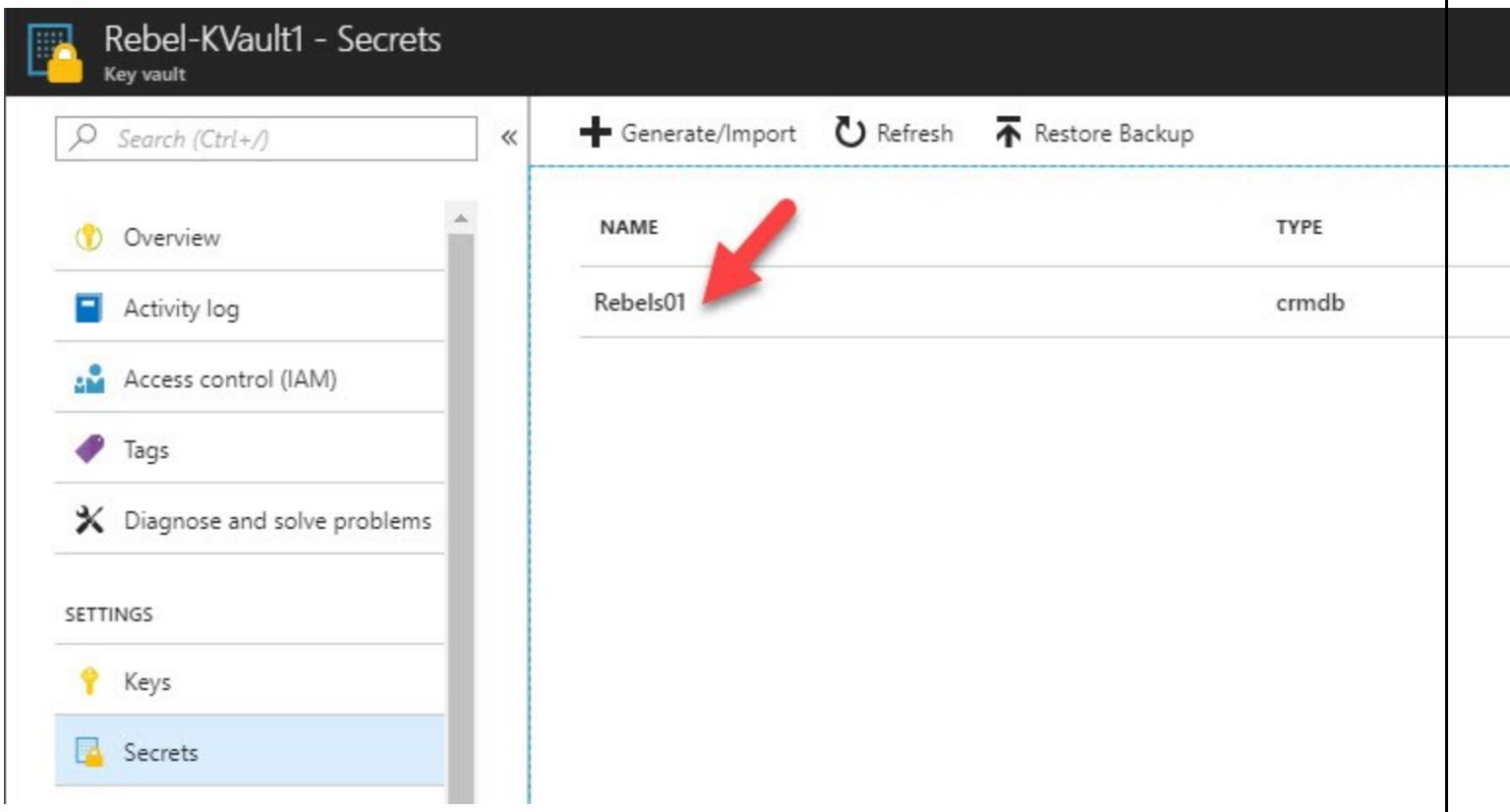
4) Then click on **Generate/Import**



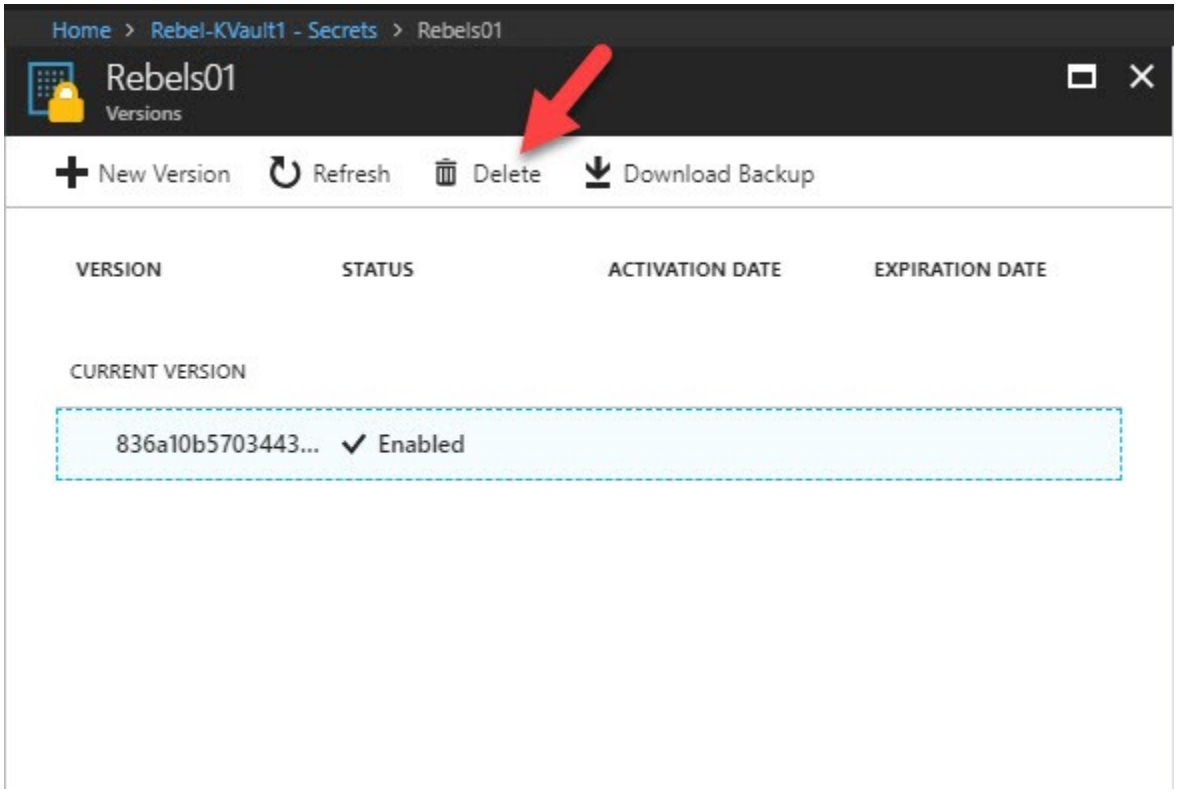
5) Then in the form fill the relevant info. **Value** defines the secret. After put relevant info click on **create**.



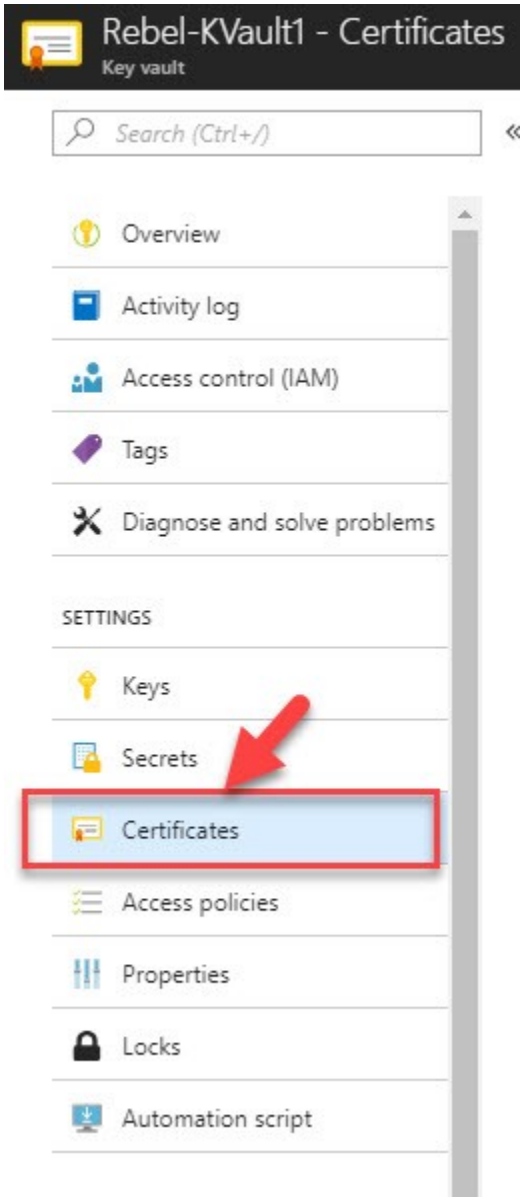
6) If you need to delete a secret, click on the relevant secret from the list.



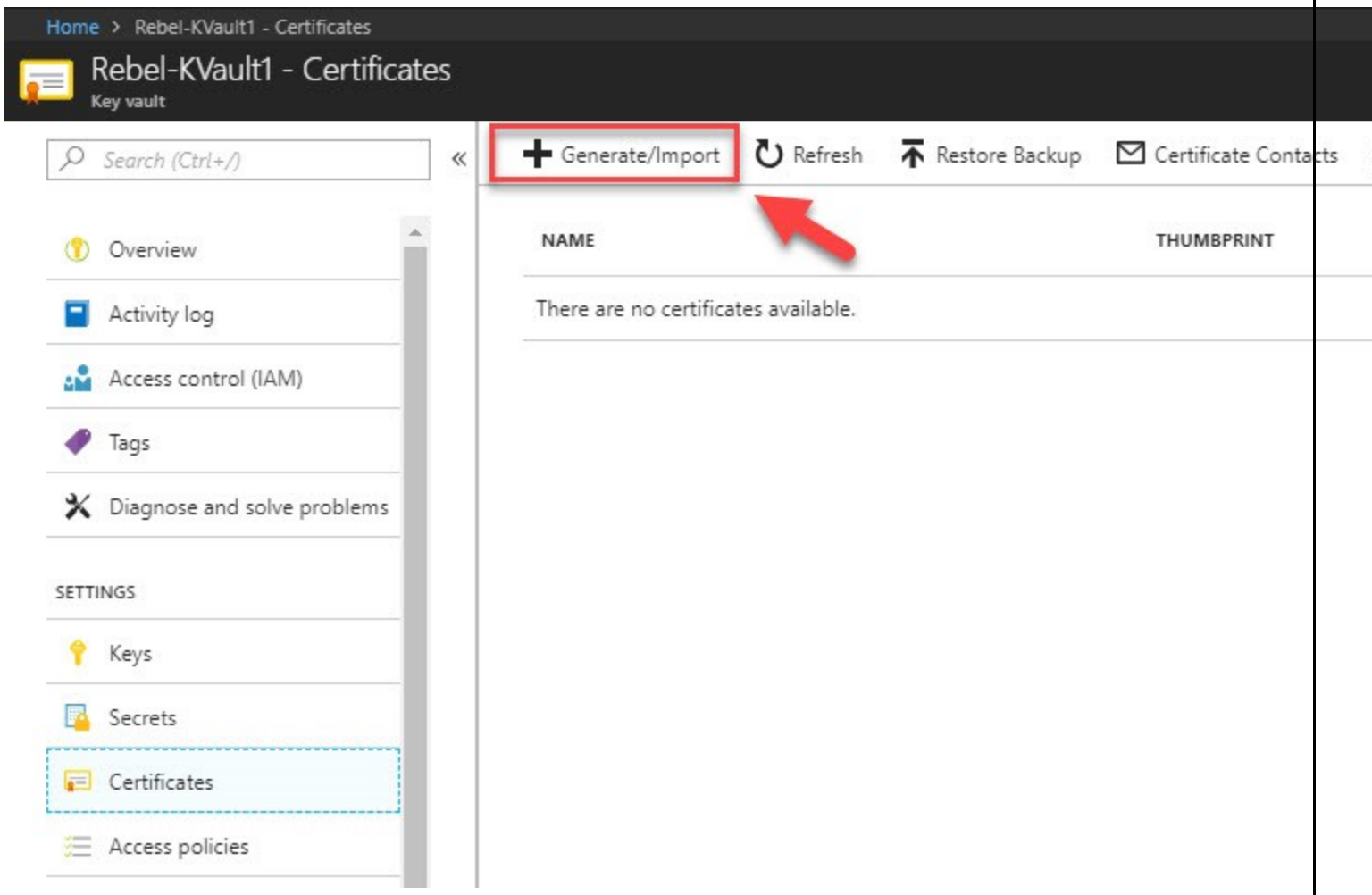
7) Then click on **Delete**.



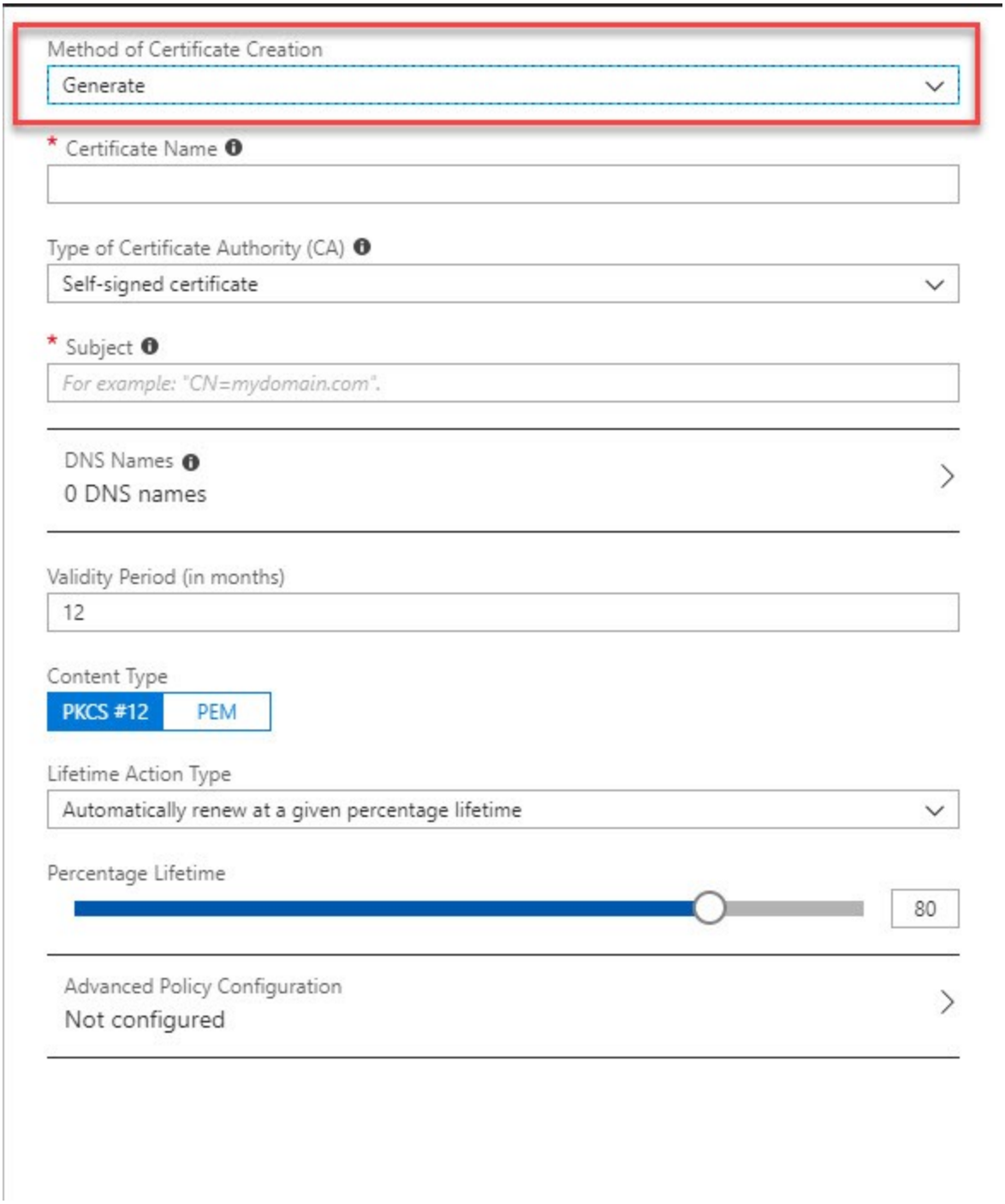
8) We also can **generate/import** certificates for use. In order to do so click on **Certificates** from the list.



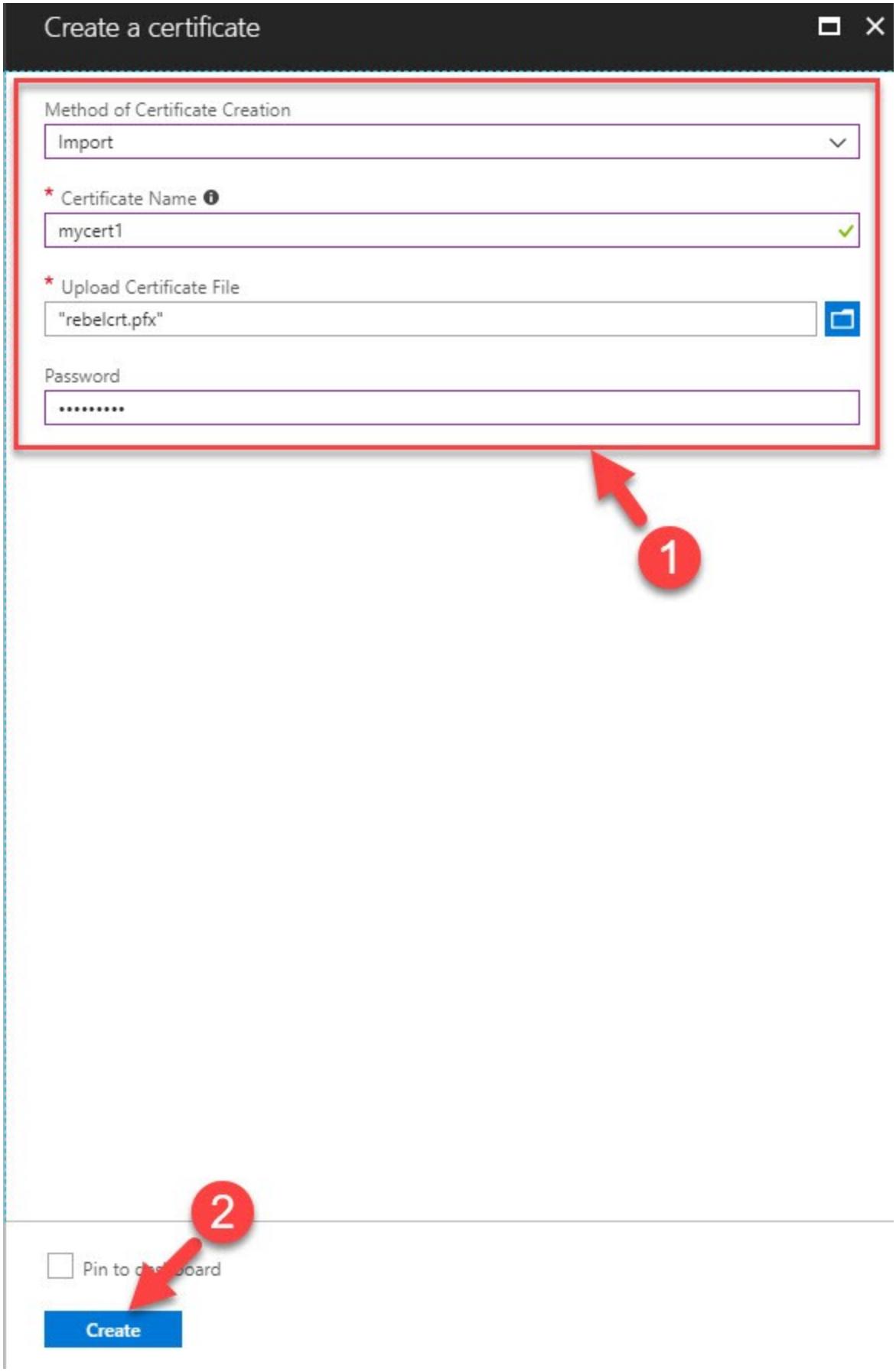
9) Then click on **Generate/Import**

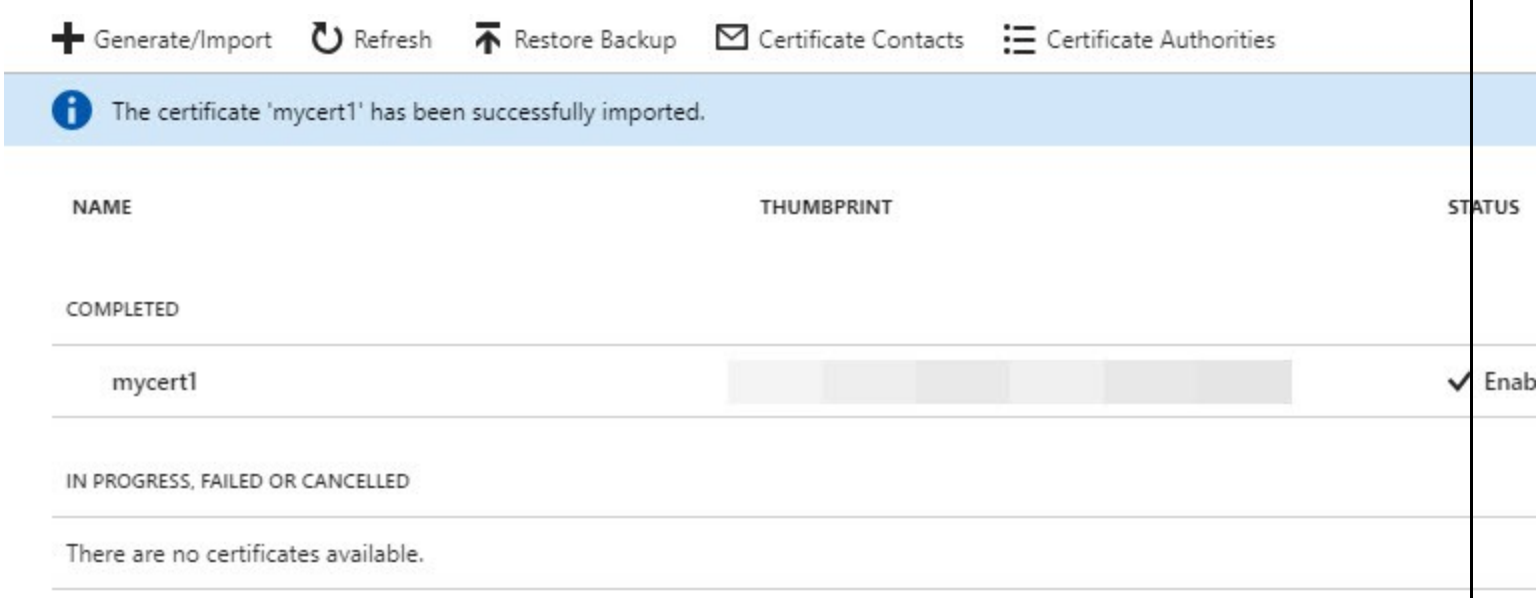


10) From the form, using **Generate** option we can create self-signed certificate.



11) Using Import option, we can import certificates in **.PFX** format. In the form, Upload Certificate File is the path for the .PFX file. You can use browse option to define the path. We can provide the PFX password under Password field. Once form is done, click on **Create**.


Create a certificate
Method of Certificate Creation
Import
* Certificate Name
mycert1
* Upload Certificate File
"rebelcrt.pfx"
Password
1
2
Pin to dashboard
Create



Hope now you have understanding about Azure key vault and how to use it. This marks the end of this blog post. If you have any questions feel free to contact me on **rebeladm@live.com** also follow me on twitter **@rebeladm** to get updates about new blog posts.

q)use of key vault in azure?

Microsoft **Azure Key Vault** is a cloud-hosted management service that allows users to encrypt **keys** and small secrets by using **keys** that are protected by hardware security modules (HSMs). Small secrets are data less than 10 KB like passwords and .PFX files

Azure Key Vault - An Introduction with step-by-step directions

20 DECEMBER 2017 on Microsoft Azure, Security, Azure Key Vault, Azure Active Directory

Wikipedia defines a Hardware Security Module (HSM) as:

A hardware security module (HSM) is a physical computing device that safeguards and manages digital keys for strong authentication and provides cryptoprocessing. These modules traditionally come in the form of a plug-in card or an external device that attaches directly to a computer or network server.

The HSM is used when security is paramount. As with other hardware devices, there is a fair bit of technicality involved in procuring, cost, installation, upgrade and maintenance (to name a few)... and that's ***before*** you can use it for all the benefit it provides!

To help you out of this hardware misery, Microsoft offers you ***Azure Key Vault*** (AKV) in the cloud. It offers the benefits of HSM, minus the headache in managing it.

How safe is storing sensitive information in AKV?

Storing information in a database and an HSM is ***very*** different. The data doesn't simply stay in a file on your server. This information is stored in hardware device and the device offers you many features like auditing, tamper-proofing, encryption, etc. What Microsoft provides in the form of AKV is an interface using which you can access the HSM device in a secure way.

If you want further assurance about the integrity of the key, you can generate it right inside of the HSM. How cool is that? Microsoft processes keys in FIPS 140-2 Level 2 validated HSMs and even Microsoft can't see or extract your keys. With logging enabled, you can pipe the logs to Azure HD Insight or SIEM solutions for threat detection.

Who takes care of patching, provisioning, and other infrastructure related issues?

Microsoft does, just like other Azure IaaS resources. They provide an SLA of 99.9% successful processing for Key Vault transactions within 5 seconds.

How can the Application access this key?

Applications have no direct access to the keys. You need to use an appropriate SDK based on your framework/language. Azure CLI, Portal & PowerShell provides an easy interface to work with the keys.

What kind of data should be stored?

Secrets which are less than 10KB should be stored in the AKV. You can also store PFX files and manage your SSL certificates using AKV. Database Connection strings, Social Network client keys, Subscription Keys, License Keys, and many other keys could be stored and managed easily using AKV.

What kind of operations are supported?

For Keys: Create, Import, Get, List, Backup, Restore, Delete, Update, Sign, Verify, Wrap, Unwrap, Encrypt & Decrypt

For Secrets: Create, Update, Get, List, Delete

For Certificates: Create, Update Policy, Contacts, Import, Renewal, Update

What about the cost?

- Zero setup fee
- Secrets: $0.03/10,000 requests
- Keys: $1 per key per month
- Certificates: $3 per renewal request

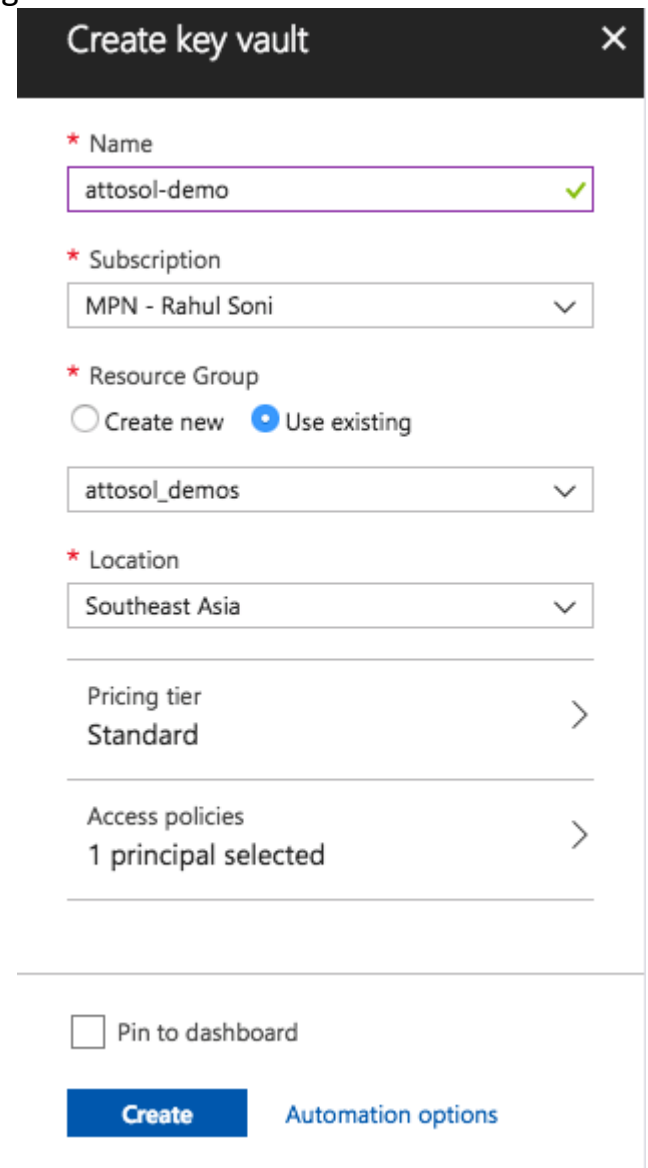
Azure Cost Calculator

Is there a quick way to test it out?

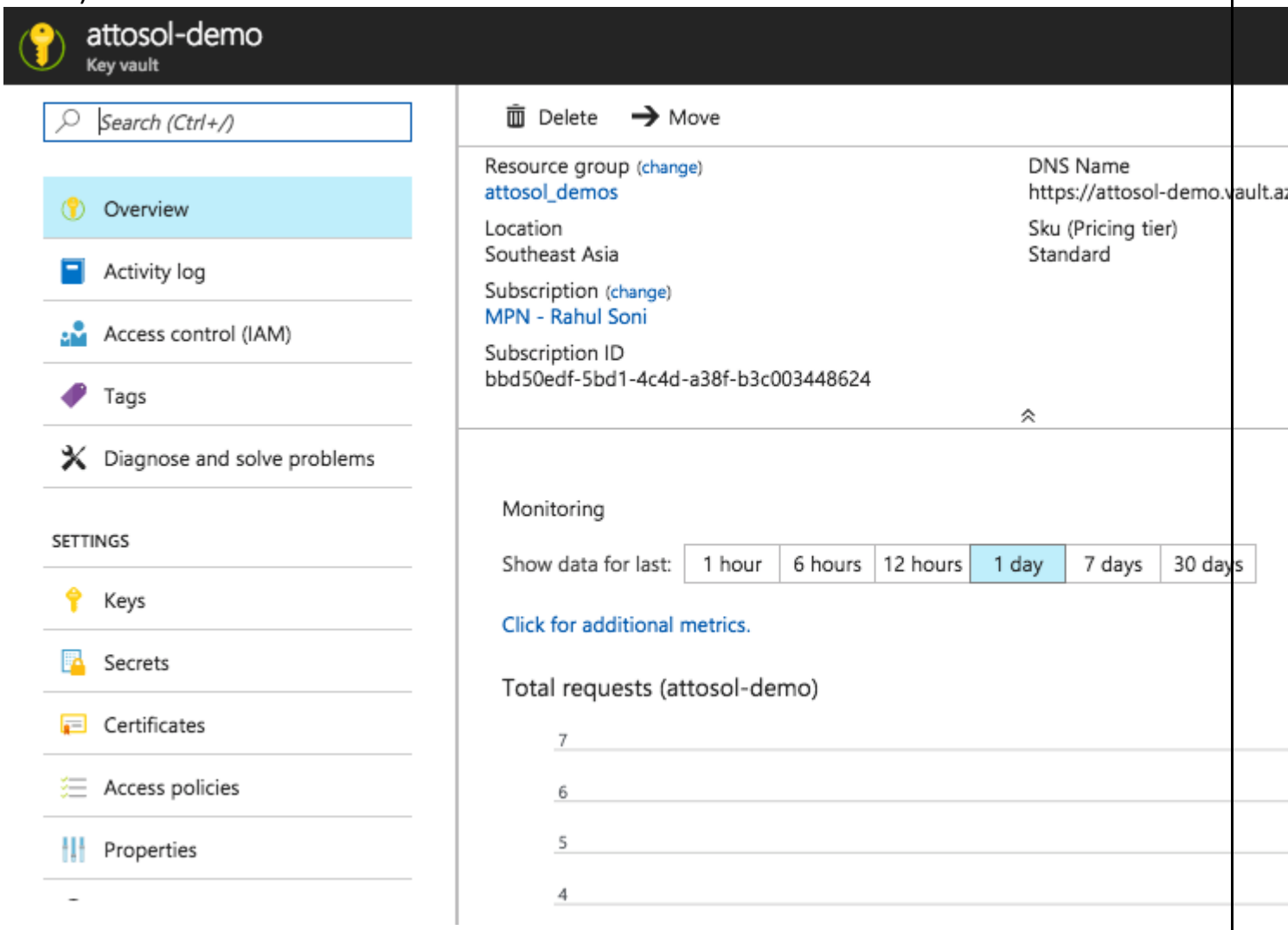
You bet! Follow along... (you can do it yourself, or simply checkout the screenshots to get an overall understanding).

Step 1: Create a Key Vault in Azure

Login > Click New > Key Vault > Create

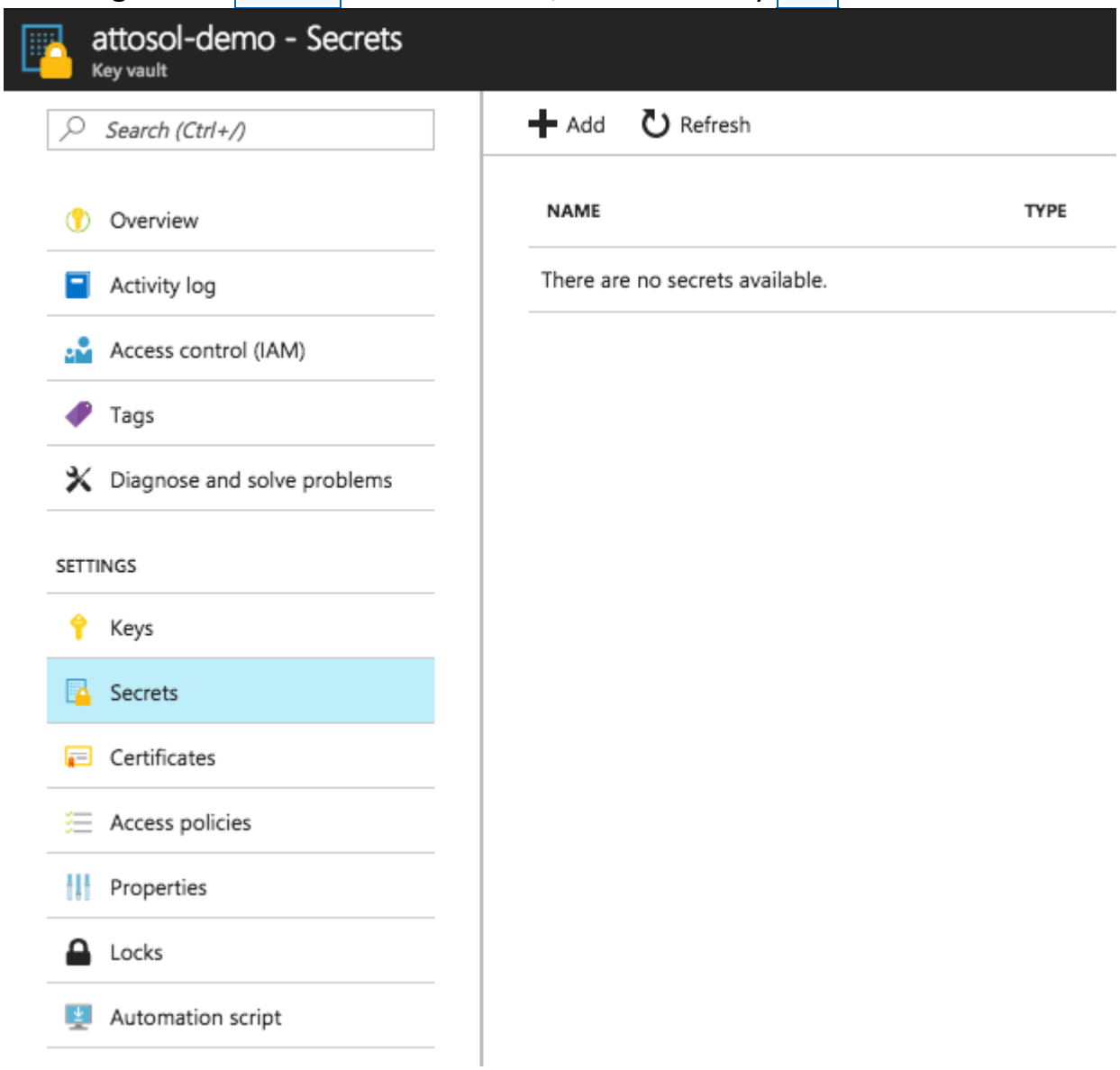


This process takes less than a minute usually. Note down the URL of your key vault (DNS Name). You will need it later.

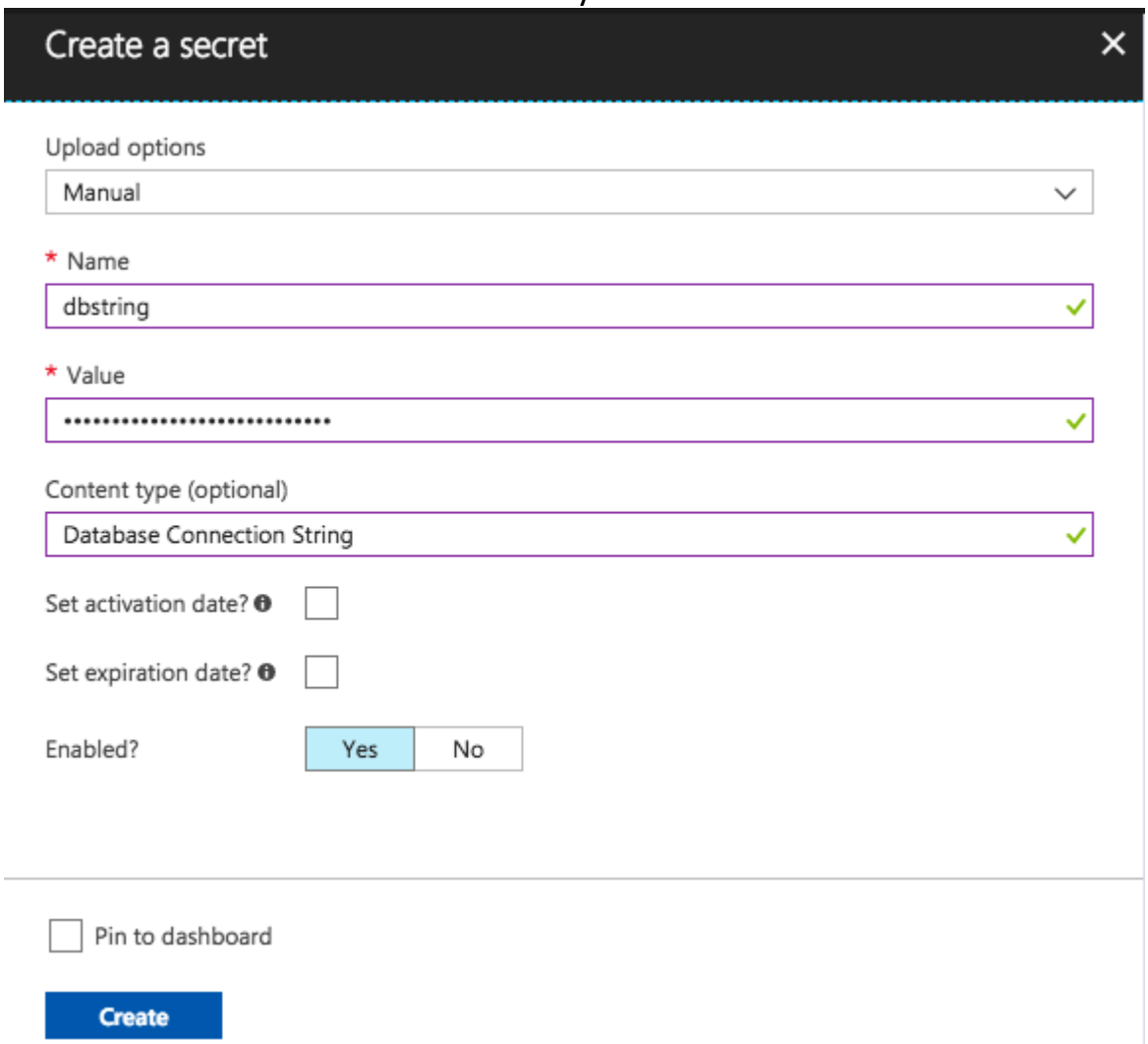


Step 2: Create a Secret

In the Azure Key Vault settings that you just created you will see a screen similar to the following. Click Secrets in the blade, followed by Add button on the top right.

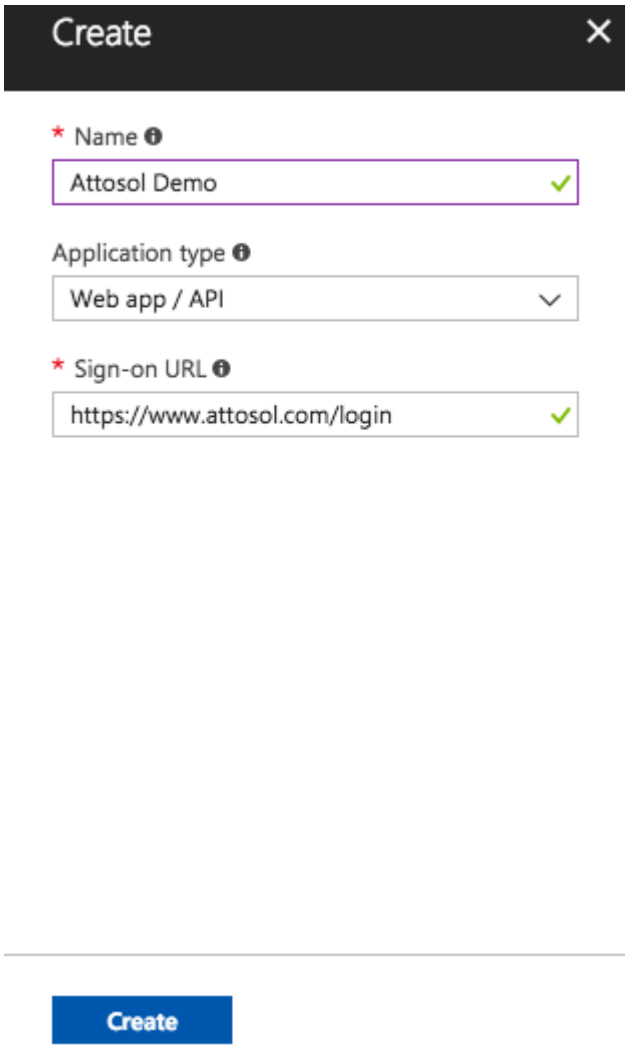


Type in your secret details:

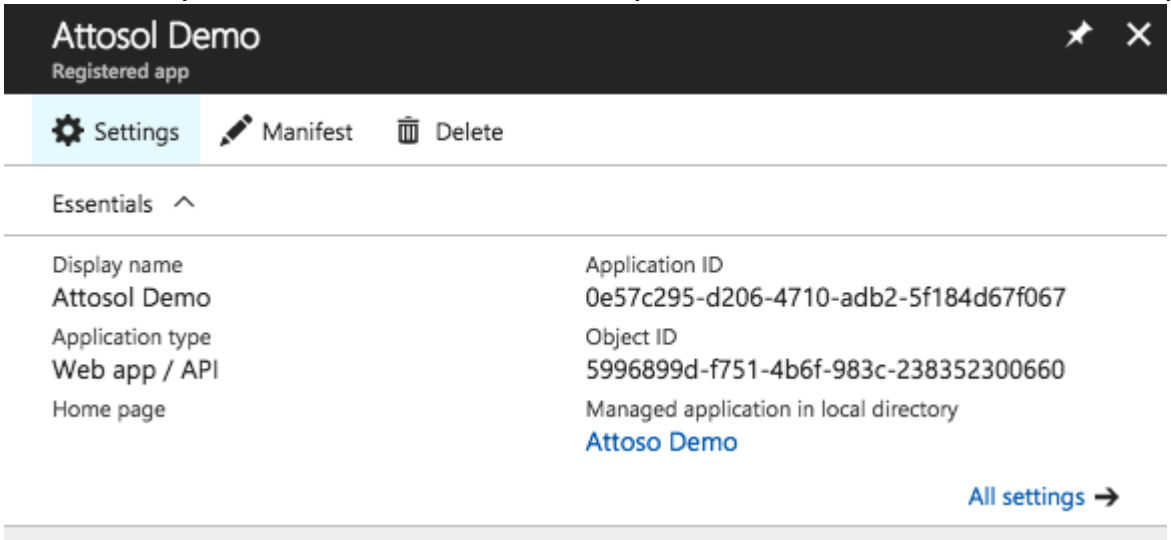


Step 3: Register an Azure Application and create Keys

Azure Portal > Azure Active Directory > App Registrations > New

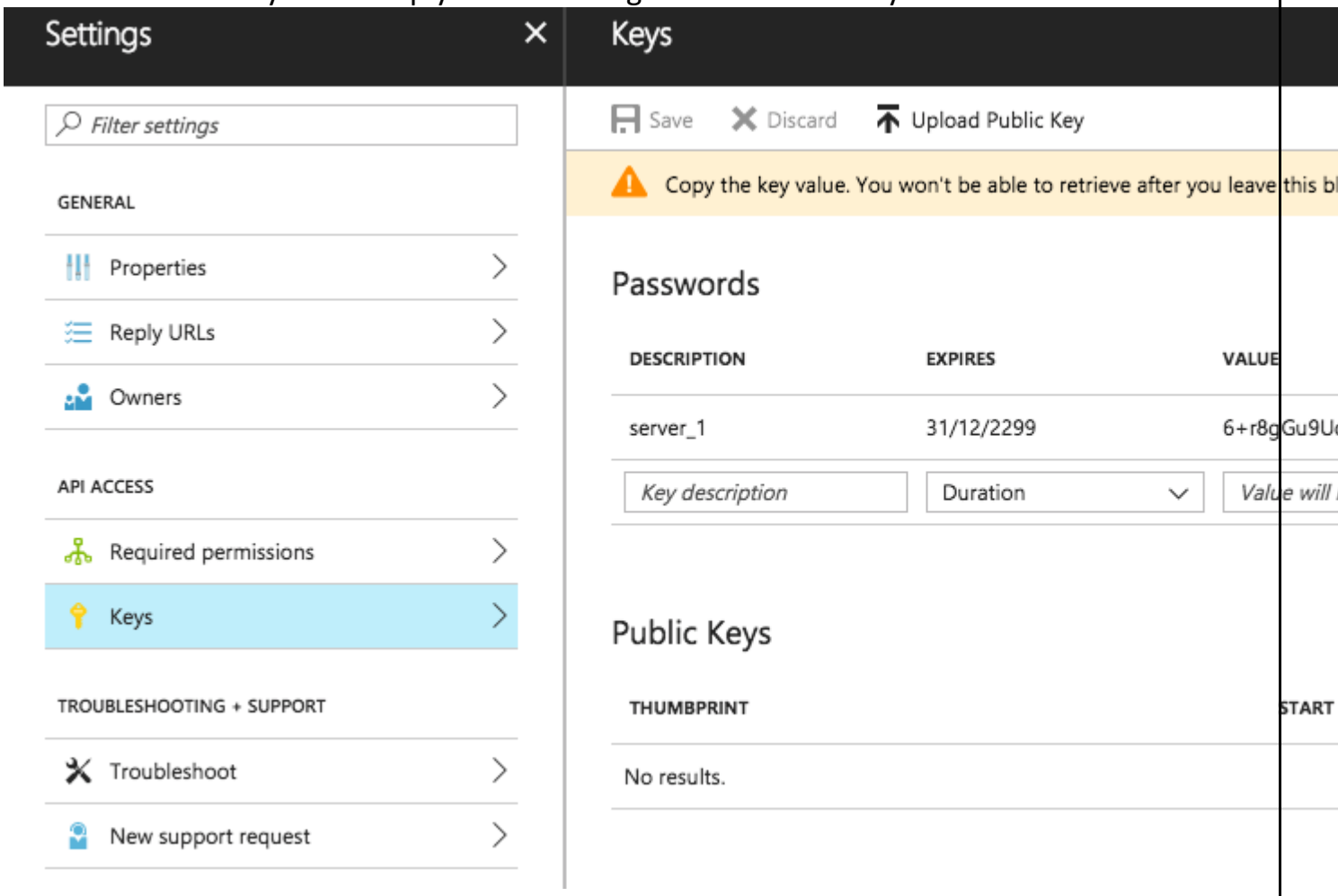


Note down your details. Remember, your client id is same as Application ID.



Let's say you have a server where you intend to access the key from. You can use the server's hostname as the key description. If this server is compromised, you can revoke the access

to AKV by simply deleting this key. That's neat!!!



Step 4: Retrieve a Secret

So far, we have created a key vault to store information, and also created an application that has access to read the key. Now, let's retrieve the secret! You will typically use a package or module, SDK, Azure CLI, PowerShell, or plain vanilla API. For the sake of simplicity, I will assume you know Node.js and use a package called masterkey.

Create a file

```
$ vi /usr/local/.masterkey/azuresecret.json
{
 "nodeAppName":{
  "clientId":"0e57c295-d206-4710-adb2-5f184d67f067",
  "clientSecret": "yW/T7ufh9bY5q0sSkwjhG6QkmXoBupz6omDMv2V9lD8=",
  "vaultUri": "https://attosol-demo.vault.azure.net/"
 }
}
```

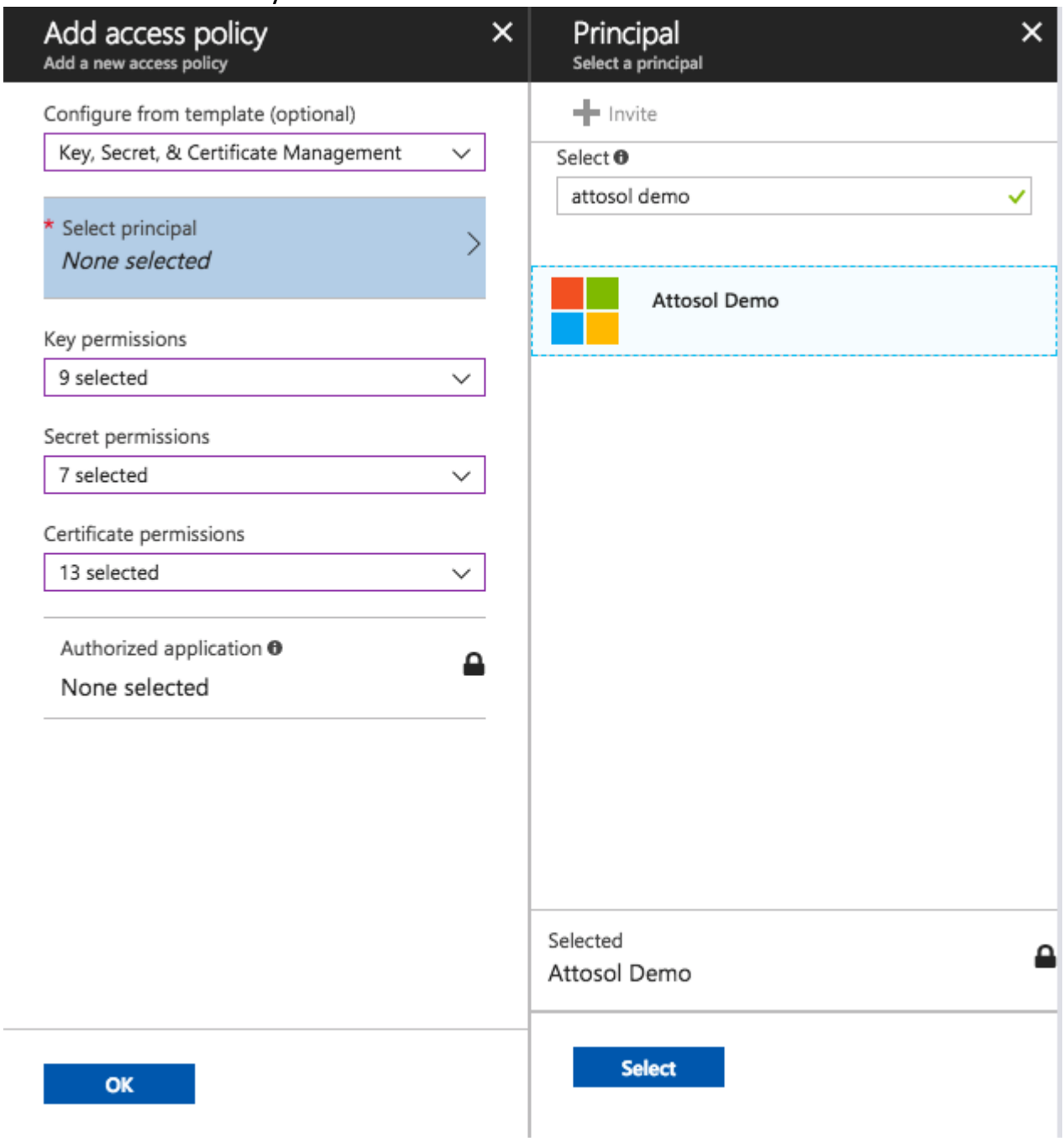
Install masterkey

```
npm install -g masterkey
```

Try to get the key, and you should hit an error! That's because, you have still not told Azure Key Vault to trust your application.

```
masterkey --get https://attosol-demo.vault.azure.net/secrets/dbstring --app nodeAppName
########################################
Listing all Secrets from the KeyVault
########################################
Unable to list secrets.
```

Give the permission to your app: Portal > Your Key Vault > Access Policies > Add New > Select Principal:



Select the principal, click Ok, and Save... the try the command again!

masterkey --get https://attosol-demo.vault.azure.net/secrets/dbstring --app nodeAppName
###################################
Get a Secret from the KeyVault
###################################
http://somevalue:9999/booyaa

There you go... Your secret!!! And nothing to worry about the secrets I revealed about my app, since I am gonna delete it right away ;-)

What next?

What's covered in this lab

In this lab, you will see how you can use Azure Key Vault in a pipeline

1. We will create a key vault, from the Azure portal, to store a MySQL server password
2. We will configure permissions to let a service principal to read the value
3. We will retrieve the password in an Azure pipeline and passed on to subsequent tasks

Before you begin

1. Refer the Getting Started page before you begin following the exercises.
2. Use the Azure DevOps Demo Generator to provision a new project

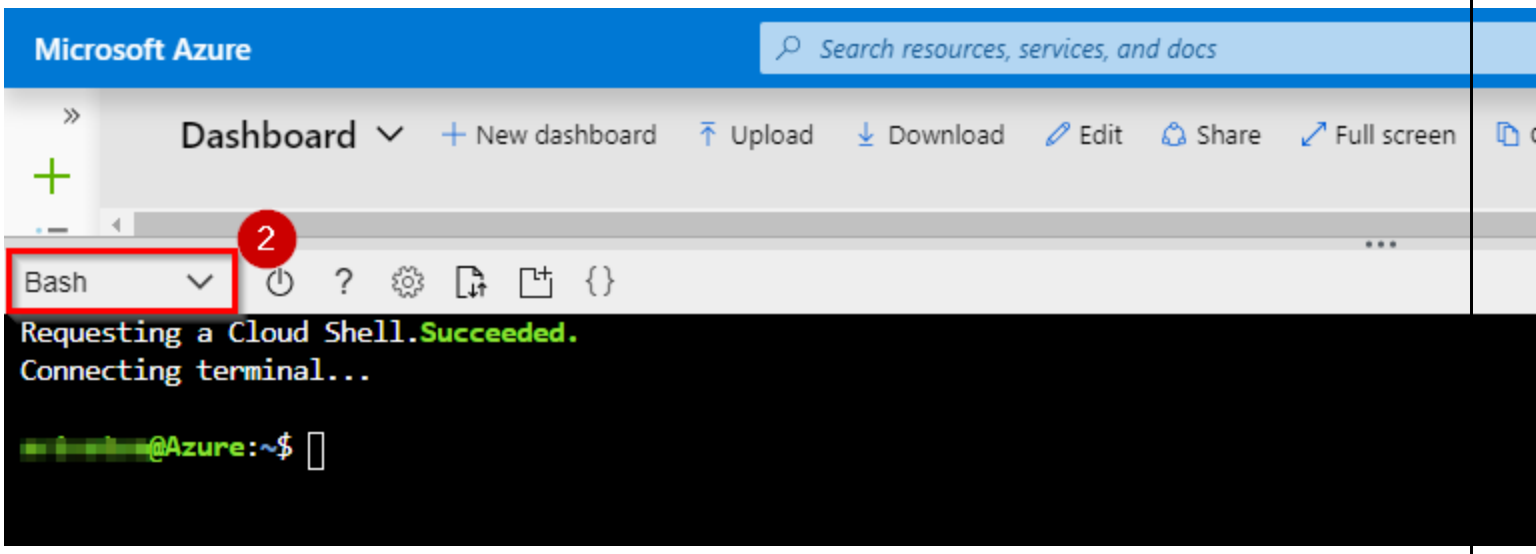
Task 1: Creating a service principal

You will need a service principal to deploy an app to an Azure resource from Azure Pipelines. Since we are going to retrieve secrets in a pipeline, we will need to grant permission to the service when we create the key vault.

A service principal is automatically created by Azure Pipeline when you connect to an Azure subscription from inside a pipeline definition or when you create a new service connection from the project settings page. You can also manually create the service principal from the portal or using Azure CLI, and re-use it across projects. It is recommended that you use an existing service principal when you want to have a pre-defined set of permissions

We will create one manually using the Azure CLI. If you do already have a service principal, you can skip this task.

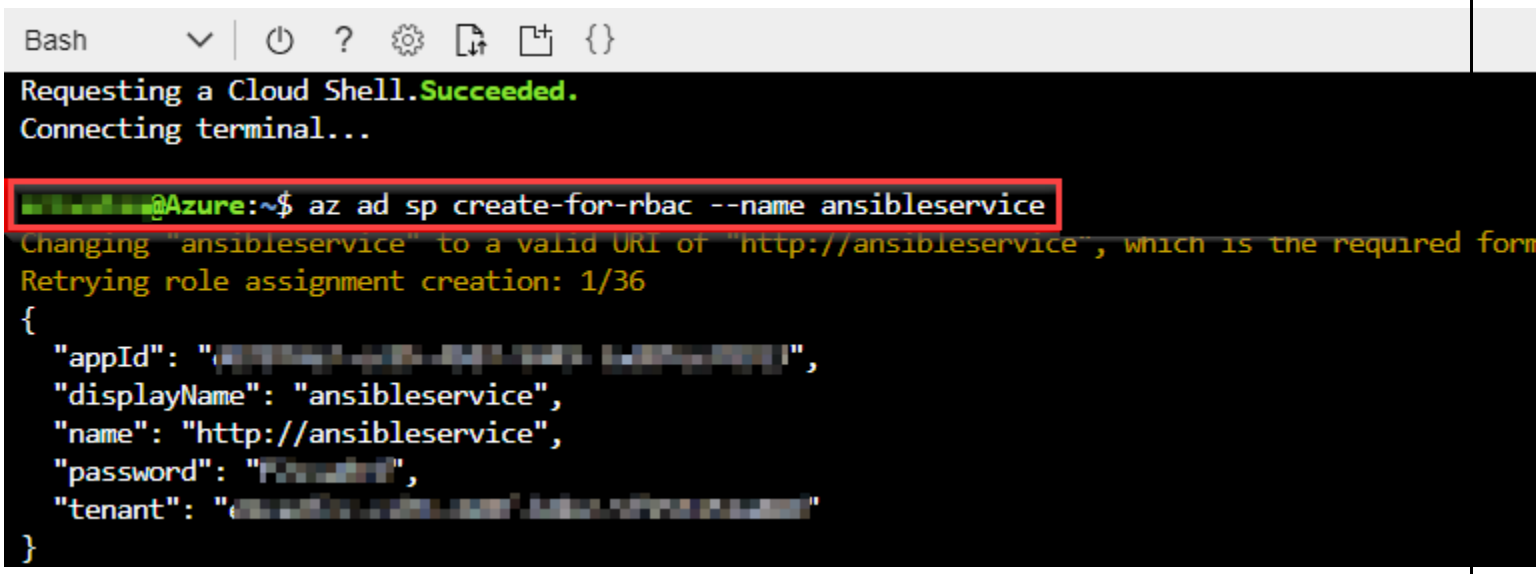
1. Login to the **Azure Portal**
2. Open the Azure cloud shell. Select **Bash** when prompted to choose shell.



3. Enter the following command by replacing ServicePrincipalName with your desired value.

az ad sp create-for-rbac -n ServicePrincipalName

It will give you a JSON output as shown in the image. Copy the output to notepad or text file. You will need them later.



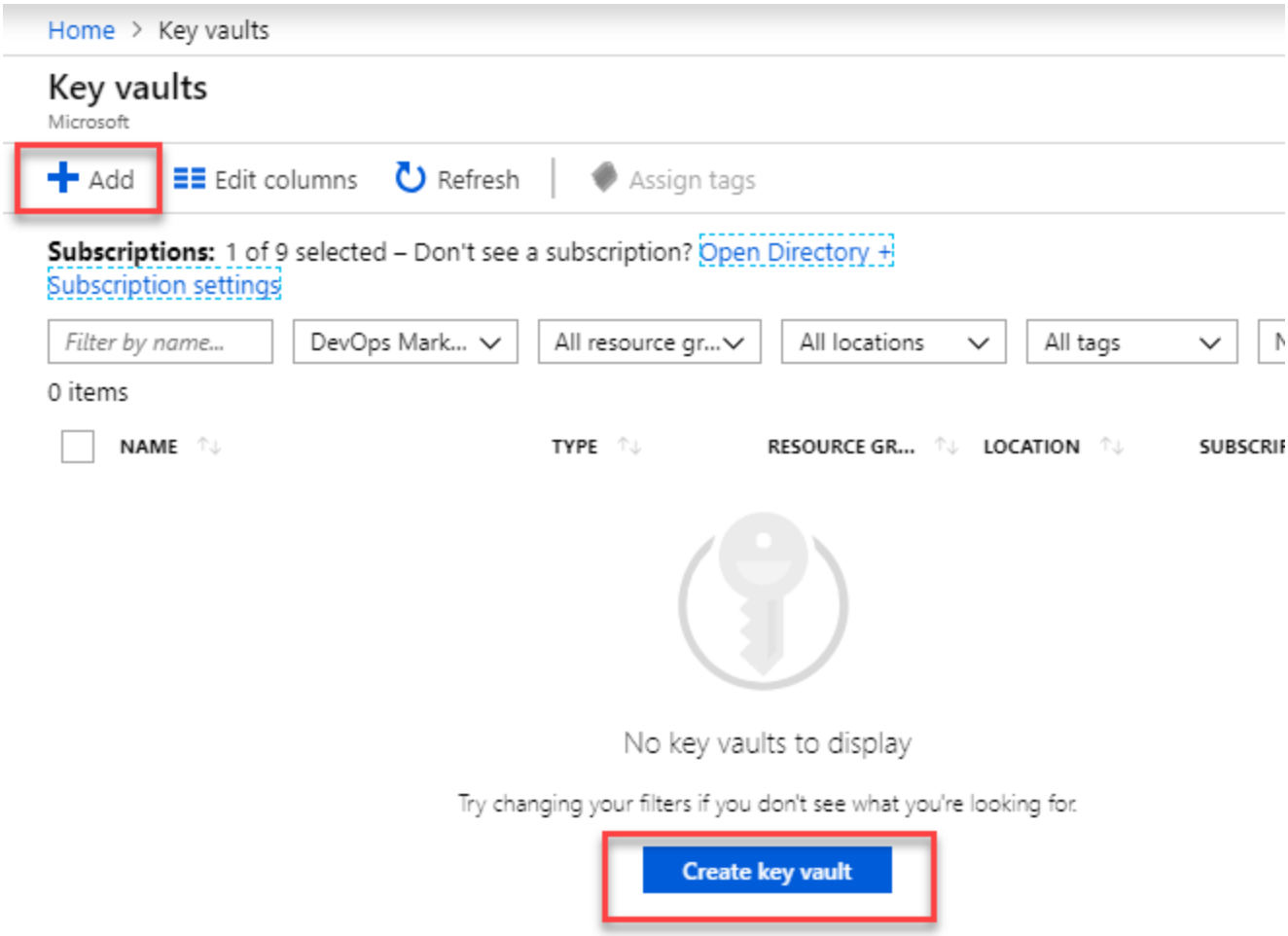
4. Enter the following command to get Azure SubscriptionID and copy the subscription ID and name to notepad.

az account show

Task 2: Creating a key vault

Next, we will create a key vault in Azure. For this lab scenario, since we are using a node-based app that connects to a MySQL database, we will store the password of MySQL database as a secret in the key vault.

1. If not already logged in, login to the Azure Portal
2. Enter “Key vault” in the search field and press enter. Select **Key Vaults** under services
3. Select **Add** or the **Create key vault** button to create a new key vault



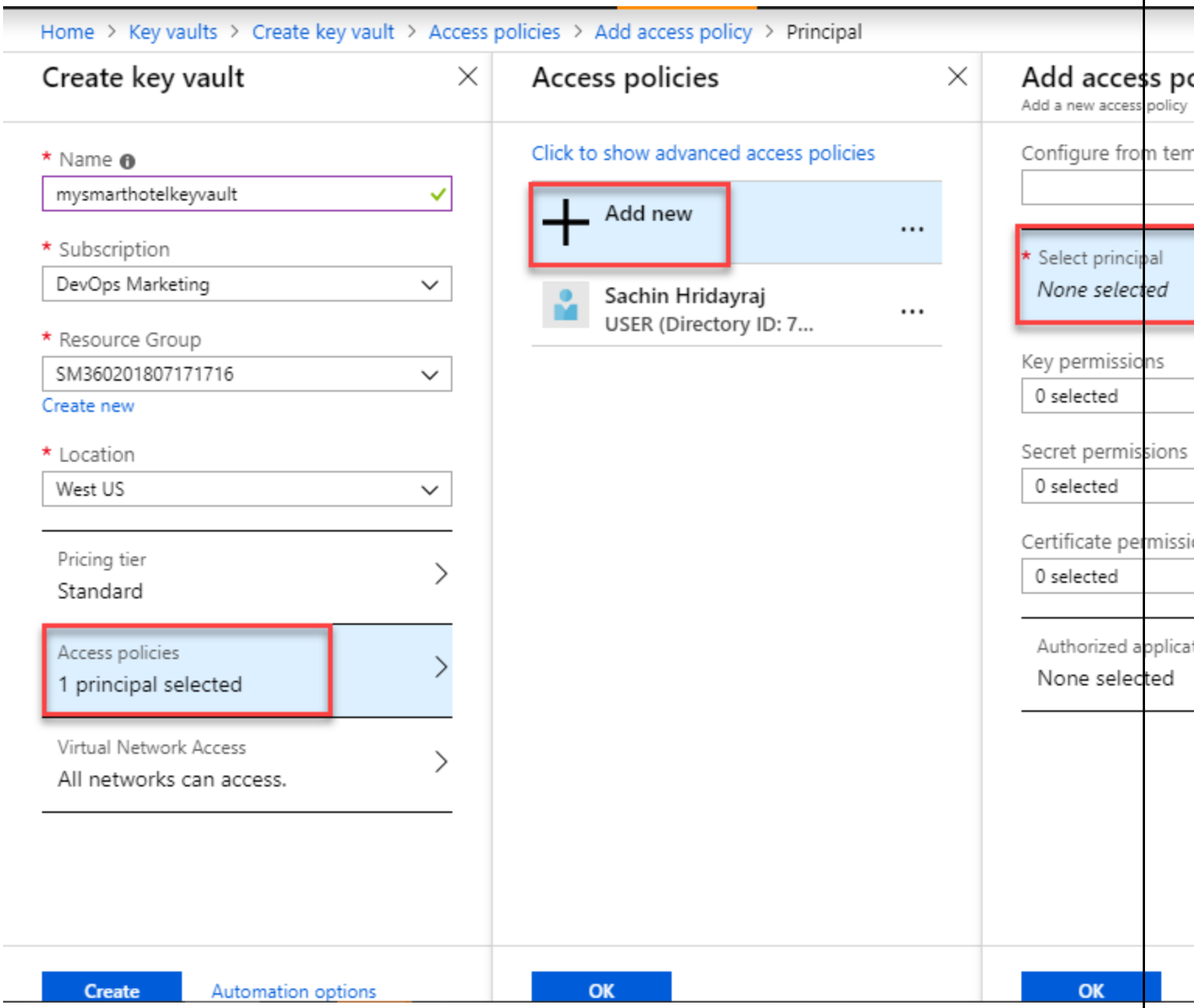
4. Provide a name, subscription, resource group and location for the vault

Because data in the Key Vaults are sensitive and business critical, you need to secure access to your key vaults by allowing only authorized applications and users. To access the data from the vault, you will need to provide read (Get) permissions to the service principal that you will be using for authentication in the pipeline.

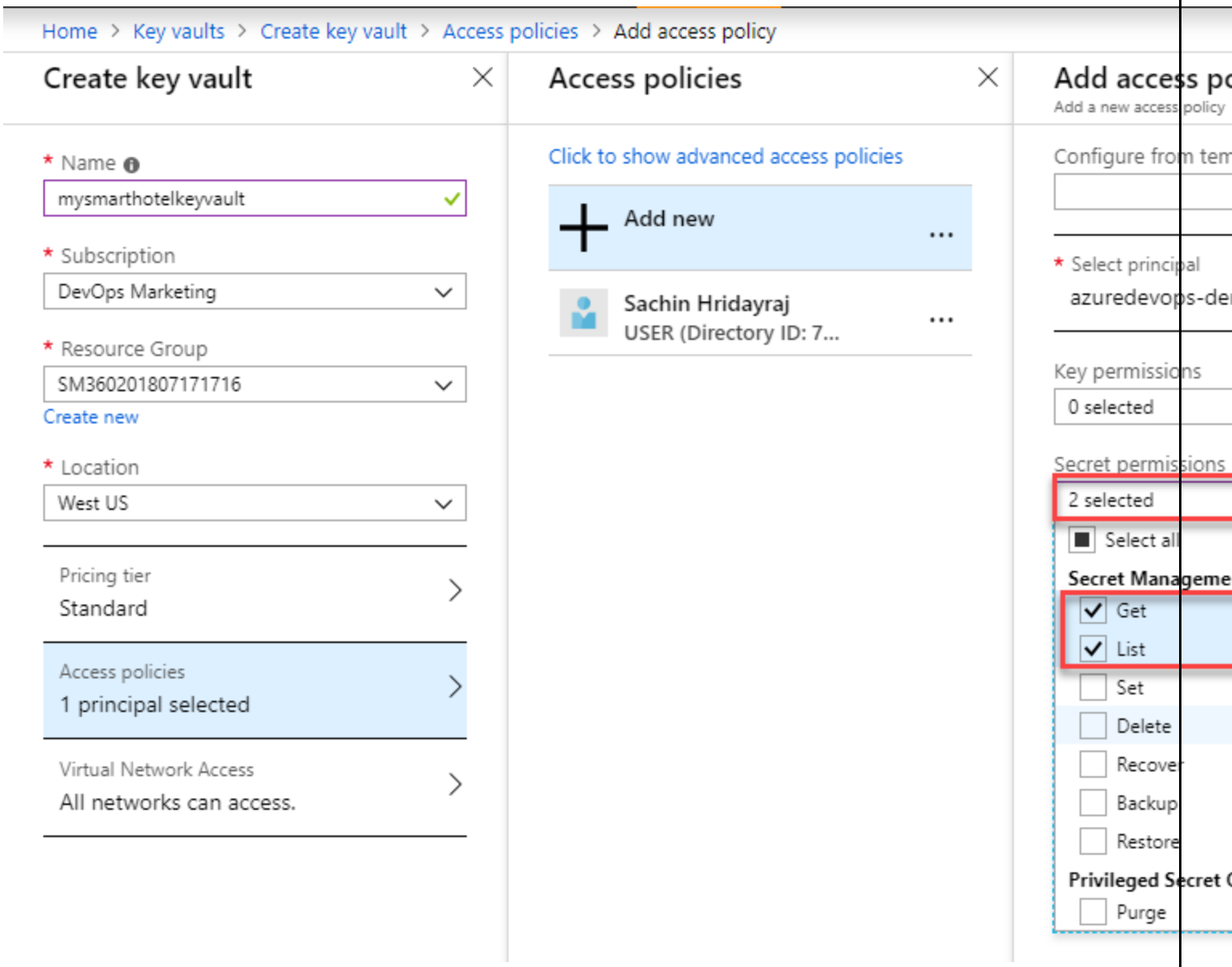
5. Select **Access policies** and then select **+Add new** to setup a new policy
6. You will need specify the permission that you intend to grant the application. This can be permissions to manage the keys, data(secrets), . In any case, applications can access the key vault in two ways:

- User + Application access: Access is granted to specific user who can then be allowed to use any application or can be restricted to use a specific application.
- Application-only access: Access is granted to the application which can be run as a daemon service or background job.

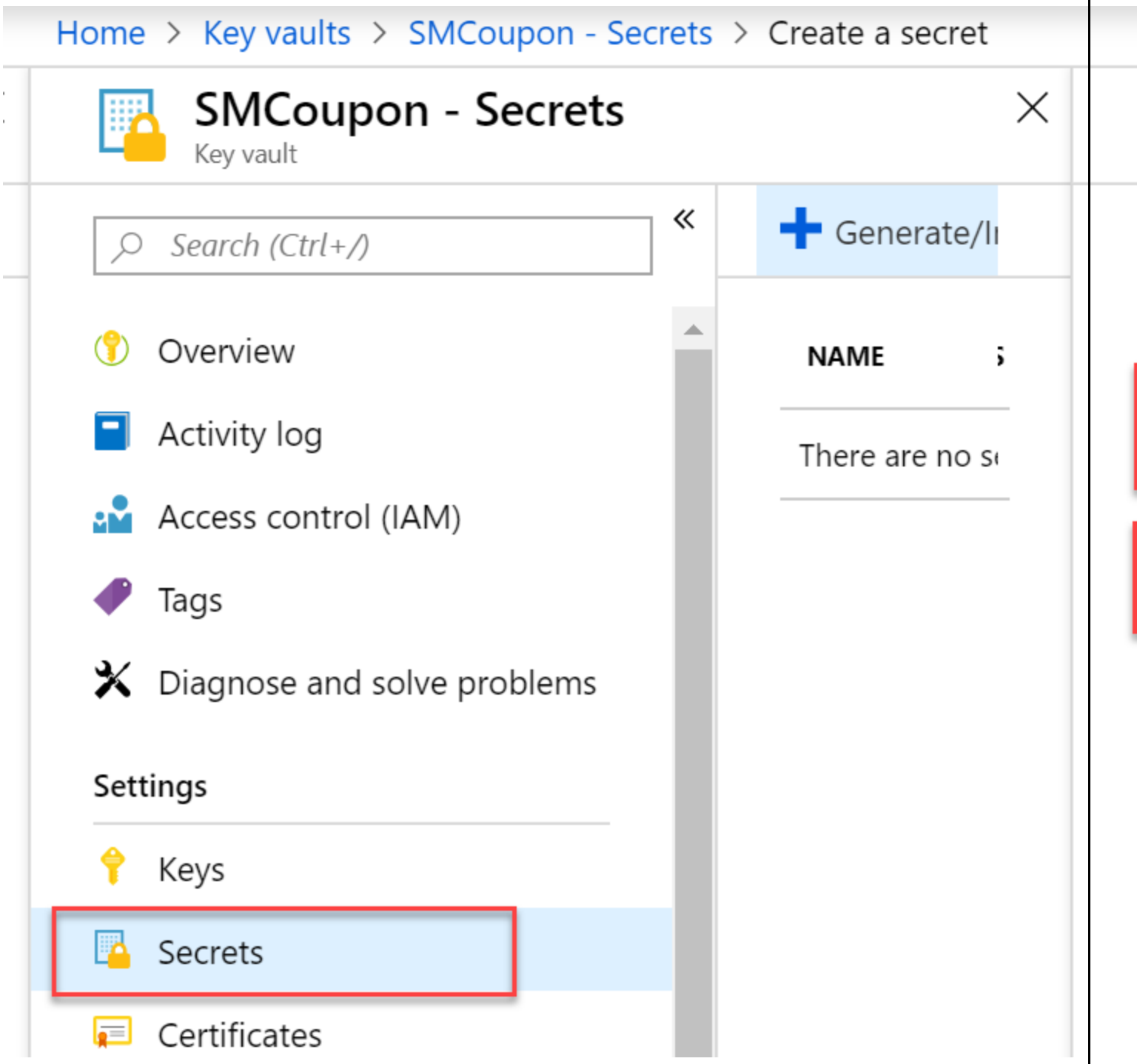
  Select the **Select principal** and search for the security principal that you created earlier and select it. You can search by name or ID of the principal.



Next, we will select the permission to be granted. For now, we will select to provide **read-only** permissions (Get, List) to just the secrets.



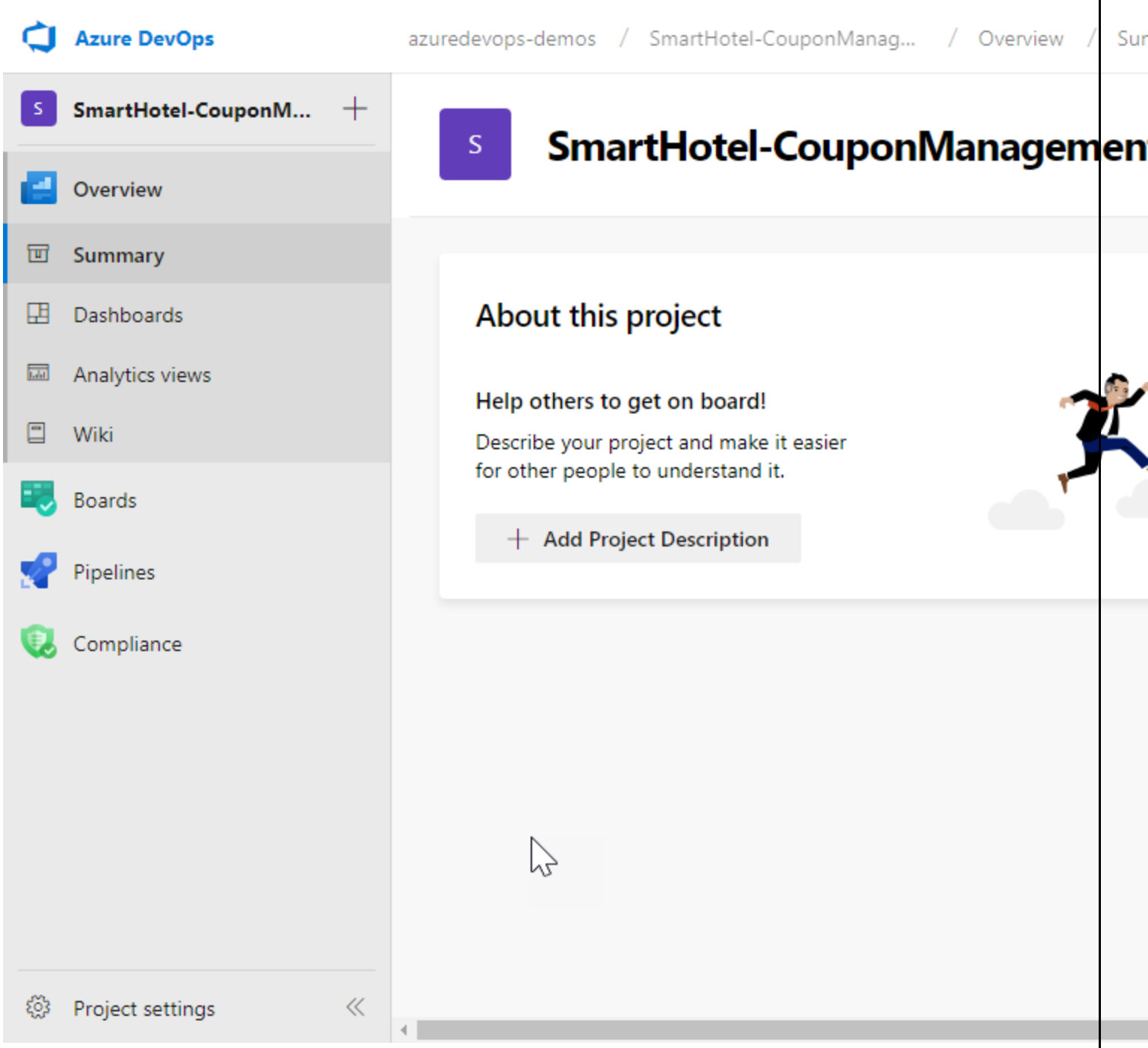
Click **OK** to close the open blades and select **Create** to create the vault.
It should only take a couple of minutes for the service to be created. Once it is provisioned, select the key vault and add a new secret. Let's name it **sqldbpassword**. Provide any value that will be accepted as a password for a MySQL database



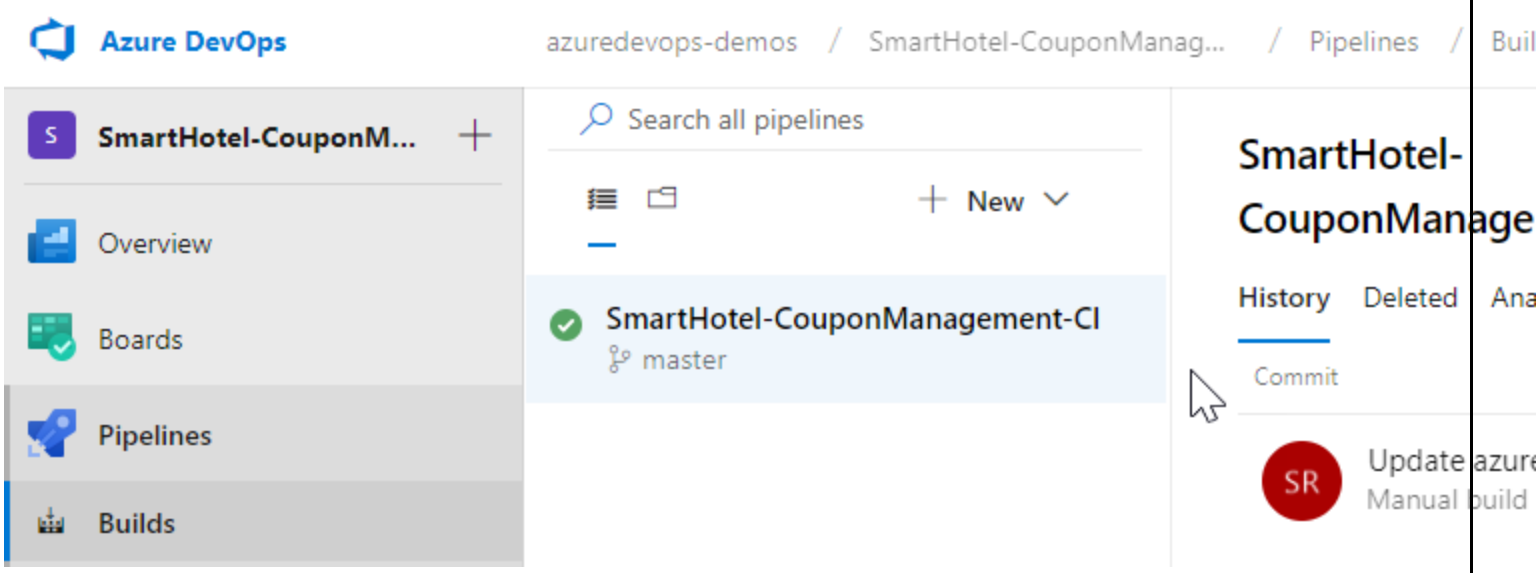
Task 3: Check the Azure Pipeline

Now, lets go to the Azure DevOps project that you provisioned using the demo generator and configure the Azure Pipelines to read the secret from the key vault

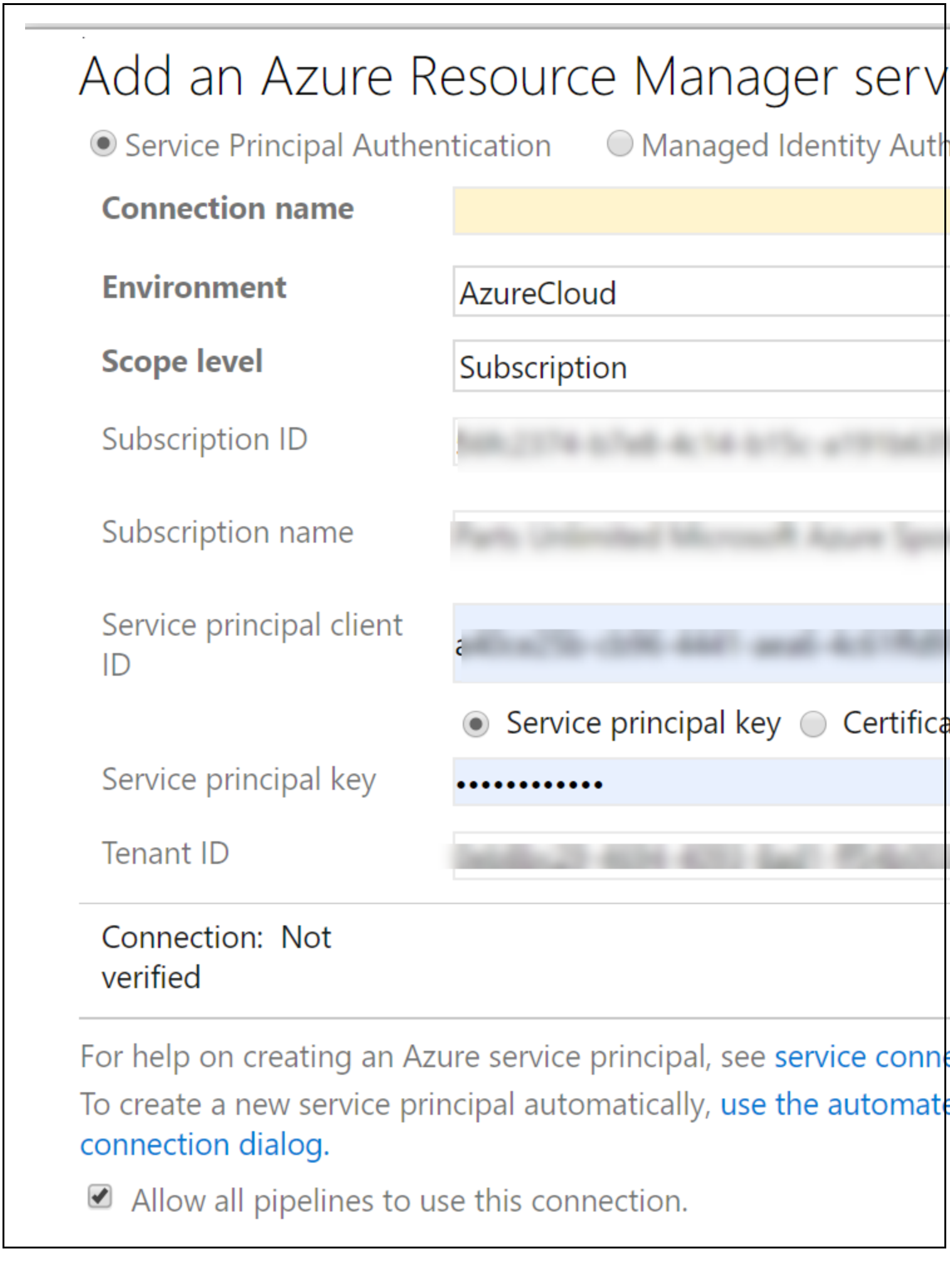
1. Navigate to the Azure DevOps project



2. Select **Pipelines** and choose **Builds**
3. Select the **Queue** button to manually queue the build definition. Wait for the build to complete



4. Go to **Releases** under **Pipelines** and then select and edit the **SmartHotel-CouponManagement-CD** definition
5. You will notice the release definition for **Dev** stage has **Azure Key Vault** task at the top. This task downloads *Secrets* from an Azure Key Vault. You will need to point to the subscription and the Azure Key Vault resource you created earlier in the lab.
6. You need to authorize the pipeline to deploy to Azure. Azure pipelines can automatically create a service connection with a new service principal but we want to use the one we created earlier. So, choose the **Advanced Options** from the drop down next to the *Authorize* button and use the full service dialog to enter the Service principal ID, passphrase, tenant ID, etc.,


Add an Azure Resource Manager servic
Service Principal Authentication
Managed Identity Authen
Connection name
Environment
AzureCloud
Scope level
Subscription
Subscription ID
Subscription name
Service principal client ID
Service principal key
Certificate
Service principal key
Tenant ID
Connection: Not verified
For help on creating an Azure service principal, see service connecti
To create a new service principal automatically, use the automated
connection dialog.
Allow all pipelines to use this connection.

7. Press **OK** to save and close the dialog. Once the connection is established, you can enter the name or select the key vault you created from the drop-down
8. In the **Secrets filter** field, you can specify an *asterisk* (*) to read all secrets or if you want only specific ones, you can provide the names of the secrets as comma-separated values

azuredevops-demos / SmartHotel-CouponManag... / Pipeline

All pipelines > SmartHotel-CouponManag

Pipeline | Tasks | Variables | Retention | Options | Histo

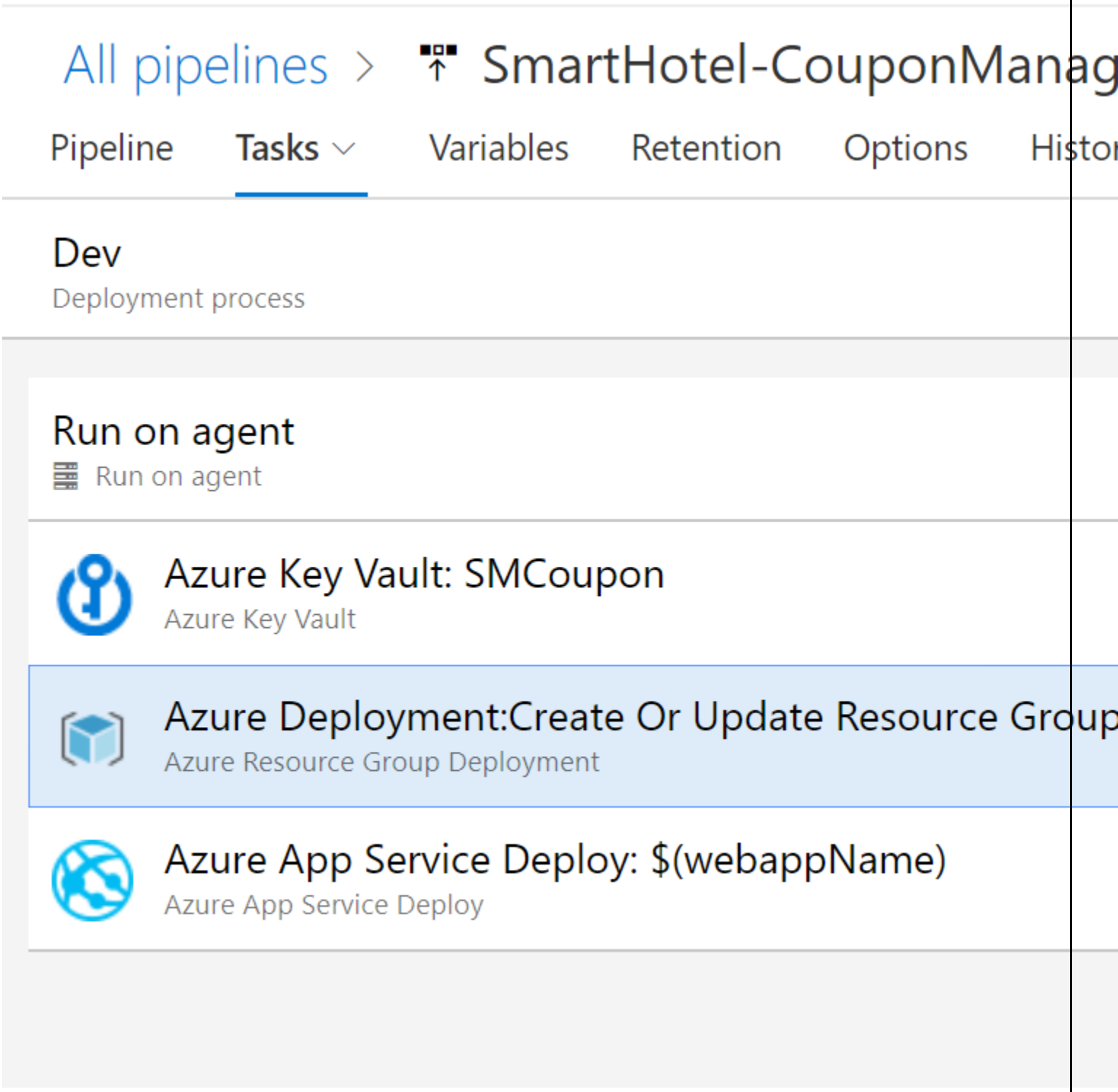
Dev
Deployment process

Run on agent
Run on agent

- Azure Key Vault: SMCoupon
  Azure Key Vault
- Azure Deployment:Create Or Update Resource Group
  Azure Resource Group Deployment
- Azure App Service Deploy: $(webappName)
  Azure App Service Deploy

At runtime, Azure Pipelines will fetch the latest values of the secrets and set them as task variables which can be consumed in the following tasks which means the password we stored earlier can be read using **$(sqldbpassword)**.

9. We pass this value in the next task, **Azure Deployment** where we deploy an ARM template



Notice the **Override template parameters** field has the database user name as a string but the password value is passed as a variable

-webAppName $(webappName) -mySQLAdminLoginName "azureuser" -mySQLAdminLoginPassword $(sqldbpassword)

This will provision the MySQL database defined in the ARM template using the password that you have specified in the key vault. It's that easy!!! ***It's that easy!!!***

You may want to complete the pipeline definition by specifying the subscription., location for the task. Repeat the same for the last task in the pipeline **Azure App Service Deploy**. Finally, save and create a new release to start the deployment

Note: You may wonder that we could have passed the value as a secret task variable itself within Azure Pipelines. While that is possible, task variables are specific to a pipeline and can't be used outside the definition it is created. Also, in most cases, secrets such as these are defined by Ops who may not want to set this for every pipeline

Exercise Challenge

Try creating a new secret to store the user name for the MySQL Database and change the pipeline to fetch and use the secret

Related Labs

- Embracing Continuous Delivery with Azure Pipelines
- GitHub integration with Azure Pipelines

q) how to configure backup for sql server azure?

Azure SQL Databases: Backups, Disaster Recovery, Import and Export

Azure SQL Database is a managed cloud database-as-a-service. It provides application developers with SQL Server databases which are hosted in the cloud and fully managed by Microsoft.

The service is very easy to start with. Several clicks in the portal and you have a database running. Now you can copy the connection string to your application config file, and boom - you have all up and running. No installation, no license to buy - just pay the hourly fee.

Any production database is a very important asset, so we are used to give it a good care in self-hosted scenario. A number of questions appear when you try to apply those practices to the cloud offering:

- How do I make a backup of my database? Where should I store it?
- How do I move my database including schema and data from on-premise to the cloud?
- How do I move it from the cloud to my local server?
- What is a point-in-time restore offered by Azure?
- Should I use geo-replication? What is geo-restore?

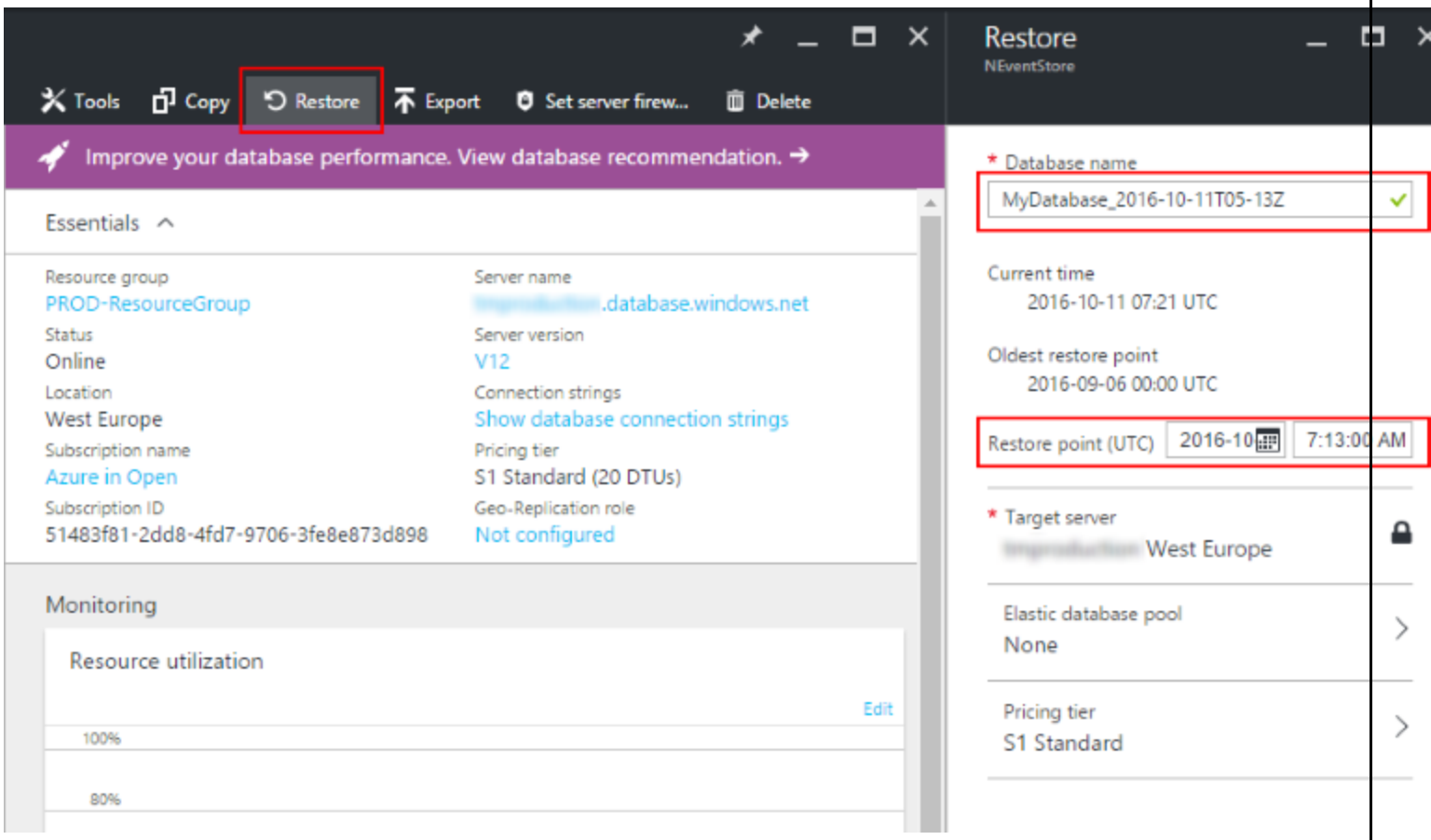
In this post I'll give the short answers to these questions and the links for further reading.

What is Point-in-time Restore?

All your databases are always automatically backed-up by Azure. They take full, differential and log backups in the background to guarantee you always have your data safe.

These backups are retained for 7 days for Basic, 14 days for Standard and 35 days for Premium tier.

Within this period, you can choose any *minute* and restore your database to that point in time. The restore always happens to a **new** database, it does not overwrite your current database.



That's very handy to recover from "oops" operations when data was deleted from one or more tables by a human or code error. In this case, you restore a copy of the database, and then move the data missing without stopping the original database.

If the restored database must replace the current one, be prepared to change connection strings once the restore operation is done. Alternatively, you can rename both databases to point applications to the new database without any other configuration changes.

Depending on the database size, the restore may take long time, up to several hours, 12 hours max guaranteed. So, point-in-time restore is very flexible but not instant.

Further reading: **Azure SQL Database Point in Time Restore**

What about disaster recovery?

The same Point-in-time Restore can be used for disaster recovery. The backups are automatically replicated to other Azure regions, and can be restored in *any* Azure region. This is called **Geo Restore**.

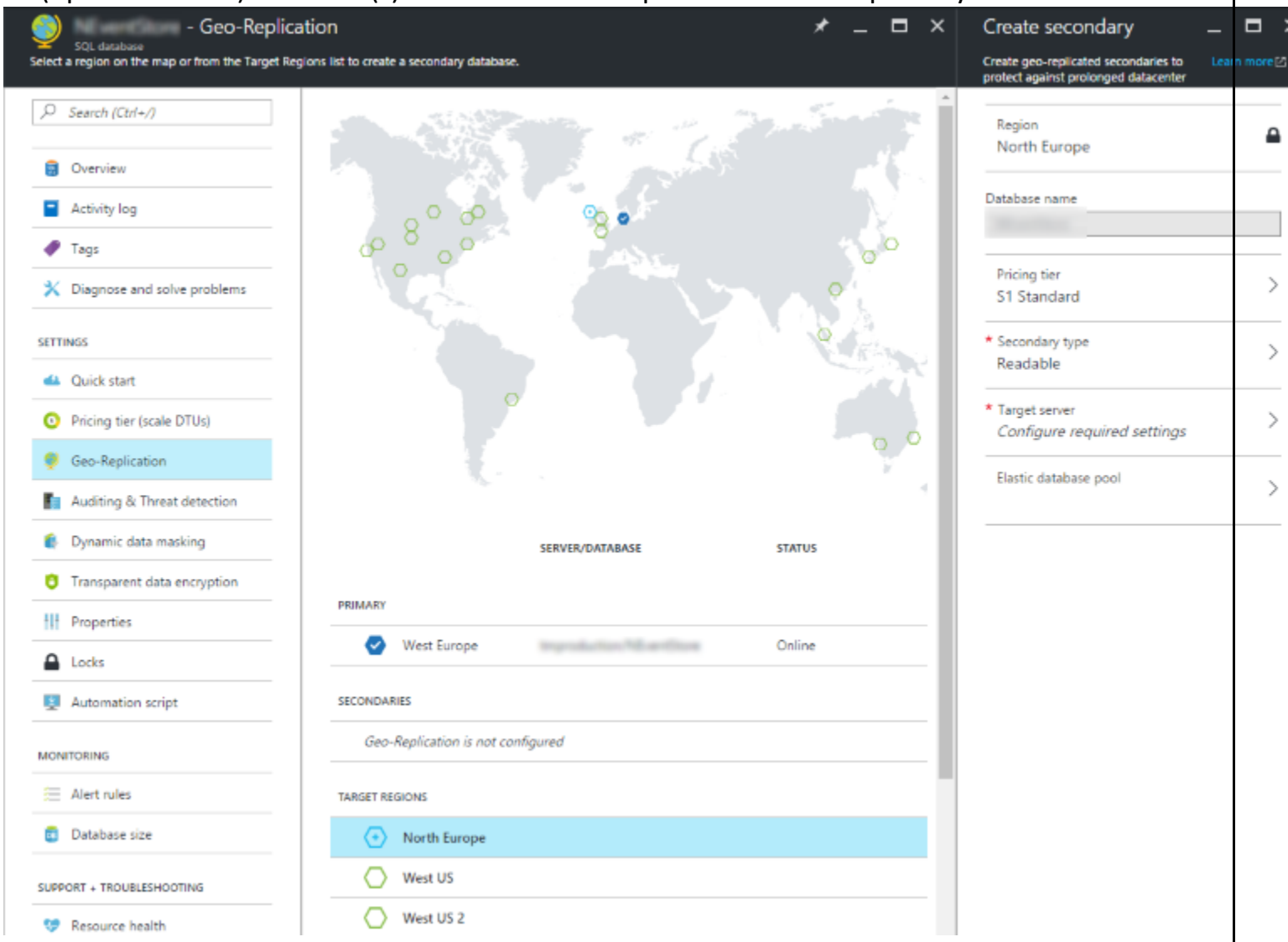
In case of failure of the primary region, you can immediately start restoring the database in another region. Remember that the restore might still take up to several hours depending on the database size.

Also, because the replication is done asynchronously, the geo-restore will probably lead to some data loss. Usually it will be under 5 minutes of data, but guarantee is 1 hour at max.

Further reading: **Azure SQL Database Geo-Restore**

Can I reduce the downtime and data loss?

If you want to be prepared to the failure of the database's Azure region and be able to fail over much faster, you can use **Active Geo Replication**. Effectively, you are creating other (up to 5 in total) database(s) which would be replicated from the primary database.



The replication happens asynchronously, which means that the latency of the primary database does not increase. That also means that some data may be lost when replica database is promoted to be the new primary. Microsoft guarantees that the loss will be limited to 5 seconds worth of data.

The failover can be done any time, manually or by your script.

Having replica databases means that you pay for them too. The performance level (and the fee) is configurable per database.

As a bonus, you can use secondary databases as read-only replicas. Just remember that the data might be slightly stale.

Geo Replication is only available for Standard and Premium pricing tiers.

Further reading: **Spotlight on SQL Database Active Geo-Replication**, **Overview: SQL Database Active Geo-Replication**

Do I still need to make manual backups?

Well, it's possible that you don't have to.

But there are at least two scenarios when you might still need to make manual backups:

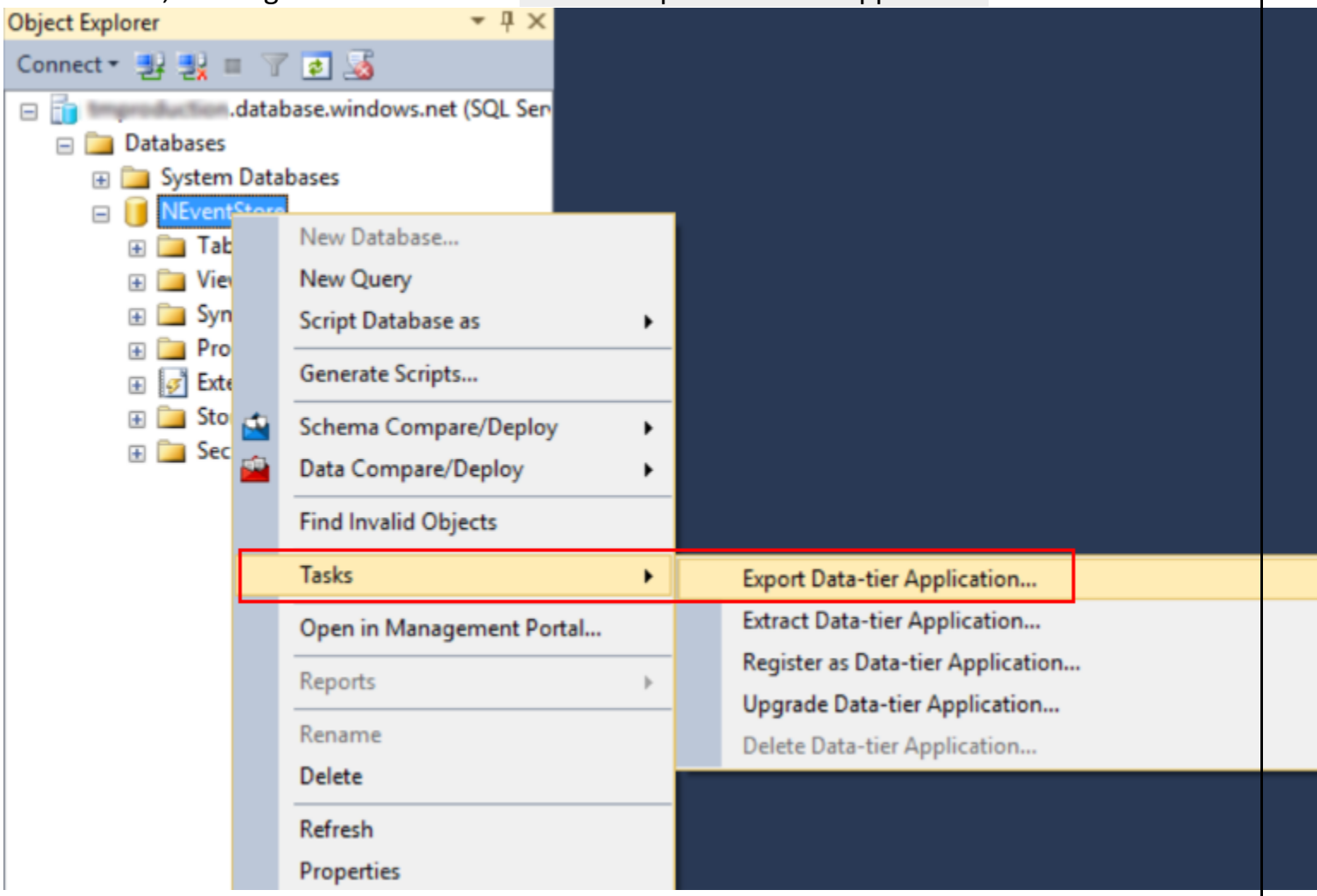
1. You need to keep a copy of your database for longer period than Point-in-time restore allows (7 to 35 days depending on the service tier).
2. You need a copy of your Azure database to be restored on premise.

Let's look at manual backups.

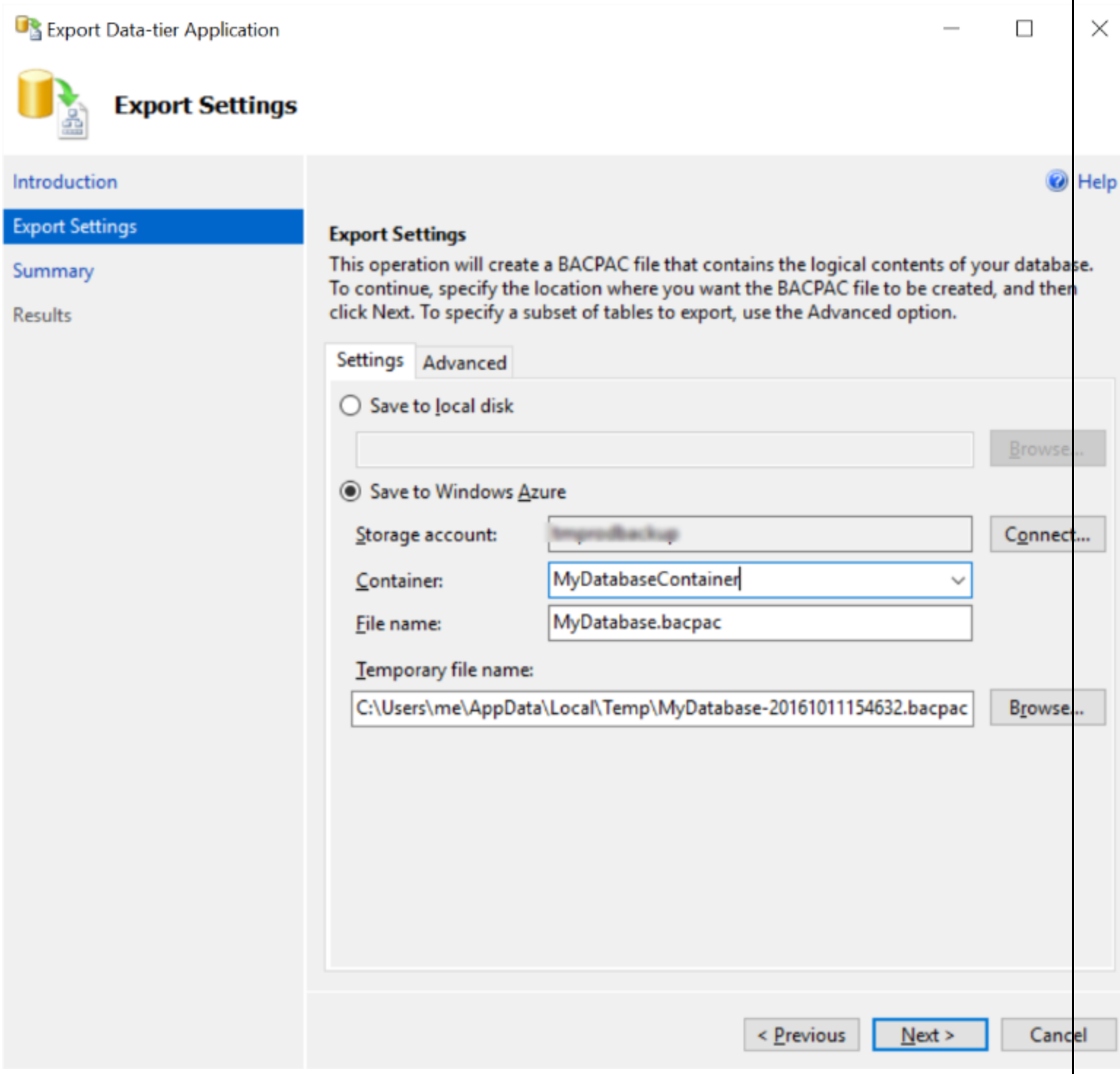
How do I make a BAK file from my Azure Database?

The BAK backup files are not directly supported by Azure SQL Databases. Instead, there is a feature called Export Data tier application, which creates a BACPAC file in Azure Storage account.

The easiest way to do that is to use SQL Server Management Studio, connect to Azure SQL Database, then right-click and select Tasks -> Export Data tier application in the menu.



You can export the file to the local storage name or Azure Storage account.

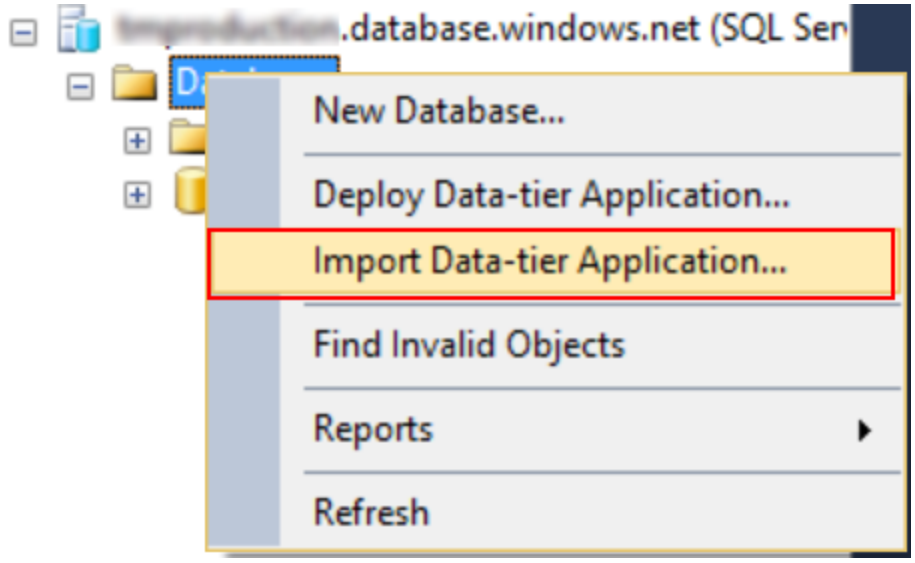


Export will take some time and will consume your database DTUs, so you shouldn't do it too often.

Export can also be triggered from Azure Portal and PowerShell scripts.

How do I restore a copy of my cloud database to a local server?

Now, when you have a BACPAC file, it's really easy to restore it to any SQL server instance. Right-click Databases node in SQL Server Management Studio and select Import Data-tier Application....



Then pick the location of the saved file.

How do I move my existing database to Azure SQL Database?

The process is exactly the same as described above, just the other direction:

- Export Data-tier Application from your local SQL Server to Azure Storage
- Import Data-tier Application to a new Azure SQL Database

Summary

Azure SQL Database is a production-ready fully managed service, which can dramatically reduce the amount of manual administration compared to on-premise setup. You can choose between several disaster recovery scenarios based on your objectives and budget. Import and export of databases are available, allowing operators to move databases between the cloud and self-hosted servers.

q)IOPS? DTU?

IOPS is the 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. OLTP applications like an online retail website need to process many concurrent user requests immediately

Database Throughput Unit (**DTU**): DTUs provide a way to describe the relative capacity of a performance level of Basic, Standard, and Premium databases. DTUs are based on a blended measure of CPU, memory, reads, and writes. As DTUs increase, the power offered by the performance level increases

How do I check my Azure DTU?

In the **Azure** portal, click SQL databases, select the database, and then use the Monitoring chart to look for resources approaching their maximum utilization. **DTU** consumption is shown by default

q)IAM?

Identity and access management (**IAM**)

Secure access to your resources with **Azure** identity and access management solutions. ... Defend against malicious login attempts and safeguard credentials with risk-based access controls, identity protection tools and strong authentication options—without disrupting productivity

Q)RBAC?

RBAC for **Azure** resources documentation

Role-based access control (**RBAC**) is a system that provides fine-grained access management of **Azure** resources. Using **RBAC**, you can segregate duties within your team and grant only the amount of access to users that they need to perform their jobs.

Q)AZURE POLICIES?

Azure Policy is a service in **Azure** that you use to create, assign, and manage **policies**. ... **Azure Policy** meets this need by evaluating your resources for non-compliance with assigned **policies**. For example, you can have a **policy** to allow only a certain SKU size of virtual machines in your environment

Overview of the Azure Policy service

Governance validates that your organization can achieve its goals through effective and efficient use of IT. It meets this need by creating clarity between business goals and IT projects.

Does your company experience a significant number of IT issues that never seem to get resolved? Good IT governance involves planning your initiatives and setting priorities on a strategic level to help manage and prevent issues. This strategic need is where Azure Policy comes in.

Azure Policy is a service in Azure that you use to create, assign, and manage policies. These policies enforce different rules and effects over your resources, so those resources stay compliant with your corporate standards and service level agreements. Azure Policy meets this need by evaluating your resources for non-compliance with assigned policies. For example, you can have a policy to allow only a certain SKU size of virtual machines in your environment. Once this policy is implemented, new and existing resources are evaluated for compliance. With the right type of policy, existing resources can be brought into compliance. Later in this documentation, we'll go over more details on how to create and implement policies with Azure Policy.

Important

Azure Policy's compliance evaluation is now provided for all assignments regardless of pricing tier. If your assignments do not show the compliance data, please ensure that the subscription is registered with the Microsoft.PolicyInsights resource provider.

Note

This service supports **Azure Delegated Resource Management** which lets service providers manage resources and subscriptions that customers have delegated from within the service provider's tenant. For more info, see **Azure Lighthouse**.

How is it different from RBAC?

There are a few key differences between Azure Policy and role-based access control (RBAC). RBAC focuses on user actions at different scopes. You might be added to the contributor role for a resource group, allowing you to make changes to that resource group. Azure Policy focuses on resource properties during deployment and for already existing resources. Azure Policy controls properties such as the types or locations of resources. Unlike RBAC, Azure Policy is a default allow and explicit deny system.

RBAC Permissions in Azure Policy

Azure Policy has several permissions, known as operations, in two Resource Providers:

- Microsoft.Authorization
- Microsoft.PolicyInsights

Many Built-in roles grant permission to Azure Policy resources. The **Resource Policy Contributor (Preview)** role includes most Azure Policy operations. **Owner** has full rights. Both **Contributor** and **Reader** can use all read Azure Policy operations, but **Contributor** can also trigger remediation.

If none of the Built-in roles have the permissions required, create a custom role.

Policy definition

The journey of creating and implementing a policy in Azure Policy begins with creating a policy definition. Every policy definition has conditions under which it's enforced. And, it has a defined effect that takes place if the conditions are met.

In Azure Policy, we offer several built-in policies that are available by default. For example:

- **Require SQL Server 12.0**: Validates that all SQL servers use version 12.0. Its effect is to deny all servers that don't meet these criteria.
- **Allowed Storage Account SKUs**: Determines if a storage account being deployed is within a set of SKU sizes. Its effect is to deny all storage accounts that don't adhere to the set of defined SKU sizes.
- **Allowed Resource Type**: Defines the resource types that you can deploy. Its effect is to deny all resources that aren't part of this defined list.
- **Allowed Locations**: Restricts the available locations for new resources. Its effect is used to enforce your geo-compliance requirements.
- **Allowed Virtual Machine SKUs**: Specifies a set of virtual machine SKUs that you can deploy.
- **Apply tag and its default value**: Applies a required tag and its default value if it's not specified by the deploy request.
- **Enforce tag and its value**: Enforces a required tag and its value to a resource.
- **Not allowed resource types**: Prevents a list of resource types from being deployed.

To implement these policy definitions (both built-in and custom definitions), you'll need to assign them. You can assign any of these policies through the Azure portal, PowerShell, or Azure CLI.

Policy evaluation happens with several different actions, such as policy assignment or policy updates. For a complete list, see Policy evaluation triggers.

To learn more about the structures of policy definitions, review Policy Definition Structure.

Policy assignment

A policy assignment is a policy definition that has been assigned to take place within a specific scope. This scope could range from a management group to a resource group. The term *scope* refers to all the resource groups, subscriptions, or management groups that the policy definition is assigned to. Policy assignments are inherited by all child resources. This design means that a policy applied to a resource group is also applied to resources in that resource group. However, you can exclude a subscope from the policy assignment.

For example, at the subscription scope, you can assign a policy that prevents the creation of networking resources. You could exclude a resource group in that subscription that is intended for networking infrastructure. You then grant access to this networking resource group to users that you trust with creating networking resources.

In another example, you might want to assign a resource type allow list policy at the management group level. And then assign a more permissive policy (allowing more resource types) on a child management group or even directly on subscriptions. However, this example wouldn't work because policy is an explicit deny system. Instead, you need to exclude the child management group or subscription from the management group-level policy assignment. Then, assign the more permissive policy on the child management group or subscription level. If any policy results in a resource getting denied, then the only way to allow the resource is to modify the denying policy.

For more information on setting policy definitions and assignments through the portal, see Create a policy assignment to identify non-compliant resources in your Azure environment. Steps for PowerShell and Azure CLI are also available.

Policy parameters

Policy parameters help simplify your policy management by reducing the number of policy definitions you must create. You can define parameters when creating a policy definition to make it more generic. Then you can reuse that policy definition for different scenarios. You do so by passing in different values when assigning the policy definition. For example, specifying one set of locations for a subscription.

Parameters are defined when creating a policy definition. When a parameter is defined, it's given a name and optionally given a value. For example, you could define a parameter for a policy titled *location*. Then you can give it different values such as *EastUS* or *WestUS* when assigning a policy.

For more information about policy parameters, see Definition structure - Parameters.

Initiative definition

An initiative definition is a collection of policy definitions that are tailored towards achieving a singular overarching goal. Initiative definitions simplify managing and assigning policy definitions. They simplify by grouping a set of policies as one single item. For example, you could create an initiative titled **Enable Monitoring in Azure Security Center**, with a goal to monitor all the available security recommendations in your Azure Security Center.

Under this initiative, you would have policy definitions such as:

- **Monitor unencrypted SQL Database in Security Center** – For monitoring unencrypted SQL databases and servers.
- **Monitor OS vulnerabilities in Security Center** – For monitoring servers that don't satisfy the configured baseline.
- **Monitor missing Endpoint Protection in Security Center** – For monitoring servers without an installed endpoint protection agent.

Initiative assignment

Like a policy assignment, an initiative assignment is an initiative definition assigned to a specific scope. Initiative assignments reduce the need to make several initiative definitions for each scope. This scope could also range from a management group to a resource group.

Each initiative is assignable to different scopes. One initiative can be assigned to both **subscriptionA** and **subscriptionB**.

Initiative parameters

Like policy parameters, initiative parameters help simplify initiative management by reducing redundancy. Initiative parameters are parameters being used by the policy definitions within the initiative.

For example, take a scenario where you have an initiative definition - **initiativeC**, with policy definitions **policyA** and **policyB** each expecting a different type of parameter:

Policy	Name of parameter	Type of parameter	Note
policyA	allowedLocations	array	This parameter expects a list of strings for a value since the parameter type has been defined as an array
policyB	allowedSingleLocation	string	This parameter expects one word for a value since the parameter type has been defined as a string

In this scenario, when defining the initiative parameters for **initiativeC**, you have three options:

- Use the parameters of the policy definitions within this initiative: In this example, *allowedLocations* and *allowedSingleLocation* become initiative parameters for **initiativeC**.
- Provide values to the parameters of the policy definitions within this initiative definition. In this example, you can provide a list of locations to **policyA's parameter – allowedLocations** and **policyB's parameter – allowedSingleLocation**. You can also provide values when assigning this initiative.

- Provide a list of *value* options that can be used when assigning this initiative. When you assign this initiative, the inherited parameters from the policy definitions within the initiative, can only have values from this provided list.

When creating value options in an initiative definition, you're unable to input a different value during the initiative assignment because it's not part of the list.

Maximum count of Azure Policy objects

There's a maximum count for each object type for Azure Policy. An entry of *Scope* means either the subscription or the management group.

Where	What	Maximum count
Scope	Policy definitions	500
Scope	Initiative definitions	100
Tenant	Initiative definitions	1,000
Scope	Policy or initiative assignments	100
Policy definition	Parameters	20
Initiative definition	Policies	100
Initiative definition	Parameters	100
Policy or initiative assignments	Exclusions (notScopes)	400
Policy rule	Nested conditionals	512

Recommendations for managing policies

Here are a few pointers and tips to keep in mind:

- Start with an audit effect instead of a deny effect to track impact of your policy definition on the resources in your environment. If you have scripts already in place to autoscale your applications, setting a deny effect may hinder such automation tasks already in place.
- Consider organizational hierarchies when creating definitions and assignments. We recommend creating definitions at higher levels such as the management group or subscription level. Then, create the assignment at the next child level. If you create a definition at a management group, the assignment can be scoped down to a subscription or resource group within that management group.
- We recommend creating and assigning initiative definitions even for a single policy definition. For example, you have policy definition *policyDefA* and create it under initiative

definition *initiativeDefC*. If you create another policy definition later for *policyDefB* with goals similar to *policyDefA*, you can add it under *initiativeDefC* and track them together.

- Once you've created an initiative assignment, policy definitions added to the initiative also become part of that initiatives assignments.
- When an initiative assignment is evaluated, all policies within the initiative are also evaluated. If you need to evaluate a policy individually, it's better to not include it in an initiative.

Video overview

The following overview of Azure Policy is from Build 2018. For slides or video download, visit Govern your Azure environment through Azure Policy on Channel 9.

Next steps

Now that you have an overview of Azure Policy and some of the key concepts, here are the suggested next steps:

- Assign a policy definition using the portal.
- Assign a policy definition using the Azure CLI.
- Assign a policy definition using PowerShell.
- Review what a management group is with Organize your resources with Azure management groups.
- View Govern your Azure environment through Azure Policy on Channel 9.

What are Azure Blueprints?

Blueprints, in the traditional sense, are used by architects and engineers to design and build new things. They are used to ensure that the final products are built to specifications and in compliance with certain standards and requirements.

Azure Blueprints are used in much the same way as traditional blueprints are. In much the same manner that an engineer or architect uses a traditional blueprint to design and build to spec, IT engineers can use Azure Blueprints to design and deploy a repeatable collection of Azure resources that adhere to certain requirements and standards. By leveraging Azure Blueprints, engineers can quickly build and deploy new environments that are always compliant with organizational standards – and they can do so far more quickly than building new each time.

Azure Blueprints allow the IT professional to orchestrate the deployment of resource templates and other Azure artifacts, including role assignments, policy assignments, resource groups, and resource manager templates. The service is back-ended by Azure Cosmos DB, which is globally distributed. Objects are replicated to multiple Azure regions to provide both highly available and low-latency access to those objects, regardless of where the Azure Blueprints objects are deployed.

Watch this short video to learn what CosmosDB can do.

The Lifecycle of an Azure Blueprint

Most resources in Azure have a natural lifecycle. Blueprints in Azure Blueprints are no different as they are created and then deployed. When they are no longer needed, they are

deleted. As such, Azure Blueprints supports typical lifecycle operations and even builds upon them. Azure Blueprints provides support for typical continuous integration and for continuous deployment pipelines for companies that manage infrastructure as code.

The typical Azure Blueprint lifecycle consists of:

- Creation of a blueprint
- Publishing of the blueprint
- Creating or editing a new version of the blueprint
- Publishing a new version of the blueprint
- Deletion of a specific version of the blueprint
- Deleting the blueprint altogether

Azure Blueprints vs Resource Manager Templates

You may be asking yourself, "why not just use Resource Manager templates instead of blueprints?" It's a fair question, given the fact that Azure Blueprints do indeed appear to overlap the functionality of Resource Manager templates.

Understanding the differences between Azure Blueprints and Resource manager templates is key to understanding which to use and when.

Azure Blueprints is intended to assist with environment setup. Such environments often include Azure resource groups, role assignments, different policies, and Resource Manager template deployments. Blueprints are essentially packages that pull these types of resources and artifacts together. These packages can then be composed, versioned, and assigned to a subscription. Such blueprint packages can also be audited and tracked.

Now, with all of that being said, almost everything that you'd want to do with Blueprints can also be done with Resource manager templates. So what's the difference?

Resource Manager templates are documents that don't natively exist in Azure. Such templates are typically stored locally or in source control. Once a Resource Manager template has been used to deploy Azure resources, there are no longer any active connections or relationships between the template and the resources deployed from it.

Azure Blueprints differ from templates because even after deployment of resources from a blueprint, the relationship between the blueprint definition and blueprint assignment (i.e., what was deployed from the blueprint) remains intact. Preserving this connection allows for tracking and auditing of deployments. Another benefit of blueprints is that a single blueprint can be used to upgrade multiple subscriptions at once.

Because blueprints can consist of Resource Manager template artifacts, previously developed Resource Manager templates are reusable with Azure Blueprints.

Azure Blueprints vs Azure Policy

So, we've compared Azure Blueprints to Resource Manager templates. What about Azure Policy? What is the overlap, if any?

An Azure policy is essentially an access system that provides default allow and explicit deny on new and existing resource properties to which the policy is applied. An Azure Blueprint is a package for creating specific sets of standards and requirements that govern the

implementation of Azure services, security, and design. Such packages are reusable so that consistency and compliance among resources can be maintained.

A policy included in a blueprint offers the ability to create the correct pattern or design when the blueprint is assigned. Additionally, a policy inclusion ensures that only approved changes can be made to the resources or environment to which the blueprint was assigned. This ensures ongoing compliance with the blueprint.

Blueprint Resources

As mentioned previously, Azure Blueprints are made up of artifacts. Resources supported as artifacts include resource groups, resource manager templates, policy assignments, and role assignments.

Resource Groups allow an administrator to organize resources and to structure them as needed. They also serve as scope limiters for policy assignment artifacts, role assignment artifacts, and Azure Resource Manager templates.

Azure Resource Group Templates are useful when designing complex environments, such as those managed with Azure Automation State Configuration. Leveraging templates makes standardization of such environments far easier than building them individually.

Policy Assignments provide a means for applying policy to a subscription to which a blueprint is assigned. That said, the policy must be within the scope of the blueprint containing the policy. Parameters defined with a policy are assigned during blueprint creation or during blueprint assignment.

Role Assignments provide a means for adding existing users or groups to a built-in role. This is done to ensure the correct people have the proper access to Azure resources. Role assignments are often defined for an entire subscription, but they can also be scoped to a specific resource group that's included in the blueprint.

Blueprint Parameters

Azure Blueprints can pass parameters to policies, initiatives, or Resource Manager templates. When the blueprint author adds an artifact to a blueprint, they must decide on a value to define for each blueprint assignment or they must allow the blueprint assignment to provide a value at assignment time. With this flexibility, the author has the option to either define a pre-set value for all uses of the blueprint or to allow that decision to occur at assignment time.

Although a blueprint can have its own parameters, such parameters can only be created when the blueprint is generated from the REST API. They cannot be created when generating the blueprint via the Portal.

Publishing and Assigning an Azure Blueprint

A new blueprint, when created, is in "draft mode". Before assigning the blueprint, it first needs to be published. The publishing process requires a version string to be defined, along with change notes that must be provided. The version, which consists of a maximum of 20 characters (letters, numbers, and hyphens), differentiates the blueprint from future changes to the blueprint – and allows each different version to be assigned separately.

As changes are made to a blueprint, the published version, along with unpublished changes, continue to coexist. After all changes to the blueprint are completed, the updated blueprint is published with a new version. Each separate (published) version of a blueprint can then be assigned to a subscription. When assigning blueprints via the portal, the blueprint defaults to the version most recently published. If there are any parameters, those are defined during the assignment process.

Summary

Azure Blueprints are essentially no different from architectural blueprints that building architects use to design and build large buildings. They are actually used in much the same way. While architects use traditional blueprints to design and construct building to certain standards and specifications, Azure architects and admins use Azure Blueprints to design and architect Azure solutions to specific standards and specifications.

The lifecycle of an Azure Blueprint begins with the creation of the blueprint and then the publishing of the blueprint. New versions of the blueprint are then created and published as needed. The lifecycle of an Azure Blueprint ends with deletion of specific versions of the blueprint, and then of the blueprint itself.

While Azure Blueprints are like Resource Manager Templates and Azure Policies, the blueprints differ because they are "living and breathing", so to speak. As such, they retain their relationships to the resources they have been assigned to and can be tracked and audited. This is not possible with templates and policies.

With its versioning capabilities, Azure Blueprints offers the ability to create newer "versions" of specific blueprints, and to leverage those versions independently from the original blueprint from which the new versions were created. This further streamlines the configuration and architecting process of Azure environments since it doesn't require the reinvention of the wheel each time a new change is needed.

When all is said and done, Azure Blueprints is a service that affords cloud architects the ability to define an easily repeatable set of Azure resources that conforms to the organization's standards and requirements. With blueprints, it is possible to quickly build and deploy new environments with a set of built-in components. As such, it is possible to not only deploy consistent environments, but it's also possible to do so in a much more streamlined fashion, allowing organizations to speed up development and delivery of such solutions.

q)ddatabricks vs datalake?

Cloud Analytics on Azure: Databricks vs HDInsight vs Data Lake Analytics

2019 is proving to be an exceptional year for Microsoft: for the 12th consecutive year they have been positioned as Leaders in Gartner's Magic Quadrant for Analytics and BI Platforms:

Figure 1. Magic Quadrant for Analytics and Business Intelligence Platforms

Source: Gartner (February 2019)

As a Microsoft Gold Partner, and having delivered many projects using the Azure stack, it's easy to see why: as Cloud technologies have become key players in BI and Big Data, Microsoft has worked wonders to create Azure, a platform that exploits the main benefits of Cloud (agility, reliability and cost) and helps all kinds of enterprise to achieve their maximum potential thanks to its flexibility.

In What can Cloud do for BI and Big Data?, we explored the different Cloud service models and how they compare to an on-premise deployment. In the Azure ecosystem, there are three main PaaS (Platform as a Service) technologies that focus on BI and Big Data Analytics:

- Azure Data Lake Analytics (ADLA)
- HDInsight
- Databricks

Deciding which to use can be tricky as they behave differently and each offers something over the others, depending on a series of factors.

At ClearPeaks, having worked with all three in diverse ETL systems and having got to know their ins and outs, we aim to offer a guide that can help you choose the platform that best adapts to your needs and helps you to obtain value from your data as quickly as possible.

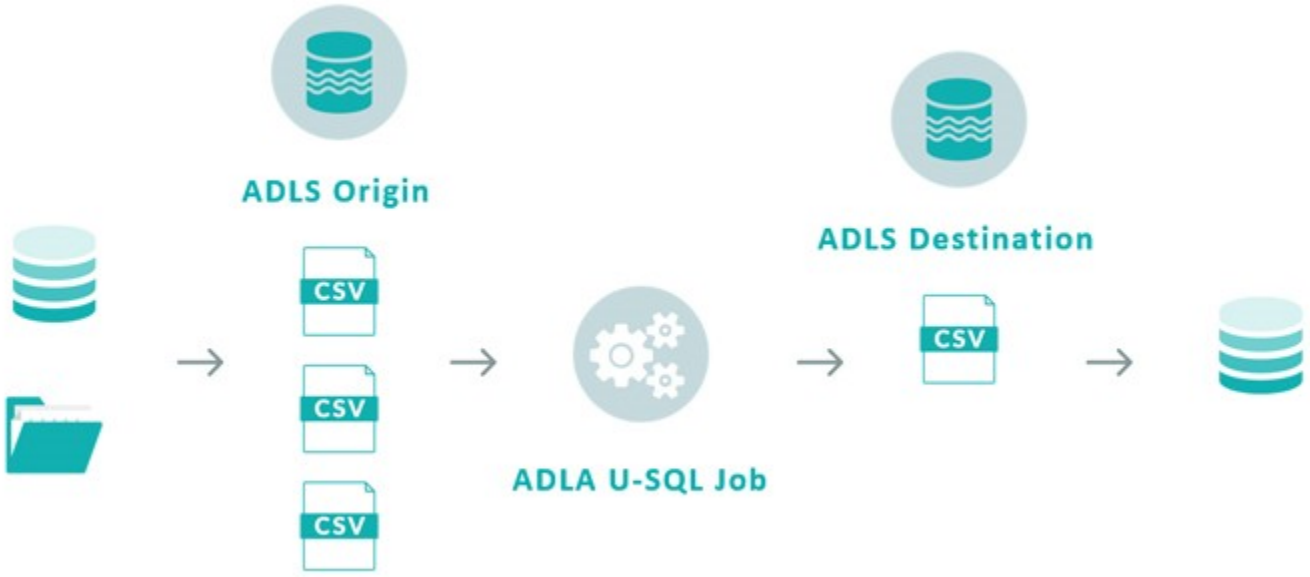
Let's review them one by one:

1. Azure Data Lake Analytics

Azure Data Lake is an on-demand scalable cloud-based storage and analytics service. It can be divided in two connected services, Azure Data Lake Store (ADLS) and Azure Data Lake Analytics (ADLA). ADLS is a cloud-based file system which allows the storage of any type of data with any structure, making it ideal for the analysis and processing of unstructured data.

Azure Data Lake Analytics is a parallelly-distributed job platform which allows the execution of U-SQL scripts on Cloud. The syntax is based on SQL with a twist of C#, a general-purpose programming language first released by Microsoft in 2001.

The general idea of ADLA is based on the following schema:



Text files from different sources are stored in Azure Data Lake Store and are joined, manipulated and processed in Azure Data Lake Analytics. The results of the operation are dumped into another location in Azure Data Lake Store.

ADLA jobs can only read and write information from and to Azure Data Lake Store. Connections to other endpoints must be complemented with a data-orchestration service such as Data Factory.

2. HDInsight

Azure HDInsight is a cloud service that allows cost-effective data processing using open-source frameworks such as Hadoop, Spark, Hive, Storm, and Kafka, among others.

Using Apache Sqoop, we can import and export data to and from a multitude of sources, but the native file system that HDInsight uses is either Azure Data Lake Store or Azure Blob Storage.

Of all Azure's cloud-based ETL technologies, HDInsight is the closest to an IaaS, since there is some amount of cluster management involved. Billing is on a per-minute basis, but activities can be scheduled on demand using Data Factory, even though this limits the use of storage to Blob Storage.

3. Databricks

Azure Databricks is the latest Azure offering for data engineering and data science. Databricks' greatest strengths are its zero-management cloud solution and the collaborative, interactive environment it provides in the form of notebooks.

Databricks is powered by Apache Spark and offers an API layer where a wide span of analytic-based languages can be used to work as comfortably as possible with your data: R, SQL, Python, Scala and Java. The Spark ecosystem also offers a variety of perks such as Streaming, MLib, and GraphX.

Data can be gathered from a variety of sources, such as Blob Storage, ADLS, and from ODBC databases using Sqoop.

4. Azure ETL showdown

Let's look at a full comparison of the three services to see where each one excels:

	ADLA	**HDInsight**	**Databricks**
Pricing	Per Job	Per Cluster Time	Per Cluster Time (VM cost + DBU processing time)

Engine	Azure Data Lake Analytics	Apache Hive or Apache Spark	Apache Spark, optimized for Databricks since founders were creators of Spark
Default Environment	Azure Portal, Visual Studio	Ambari (HortonWorks), Zeppelin if using Spark	Databricks Notebooks, RStudio for Databricks
De Facto Language	U-SQL, (Microsoft)	HiveQL, open source	R, Python, Scala, Java, SQL, mostly open-source languages
Integration with Data Factory	Yes, to run U-SQL	Yes, to run MapReduce jobs, Pig, and Spark scripts	Yes, to run notebooks, or Spark scripts (Scala, Python)
Scalability	Easy, based on Analytics Units	Not scalable, requires cluster shutdown to resize	Easy to change machines and allows autoscaling
Testing	Tedious, each query is a paid script execution, and always generates output file (Not interactive)	Easy, Ambari allows interactive query execution (if Hive). If using Spark, Zeppelin	Very easy, notebook functionality is extremely flexible
Setup and managing	Very easy as computing is detached from user	Complex, we must decide cluster types and sizes	Easy, Databricks offers two main types of services and clusters can be modified with ease
Sources	Only ADLS	Wide variety, ADLS, Blob and databases with sqoop	Wide variety, ADLS, Blob, flat files in cluster and databases with sqoop
Migratability	Hard, every U-SQL script must be translated	Easy as long as new platform supports MapReduce or Spark	Easy as long as new platform supports Spark
Learning curve	Steep, as developers	Flexible as long as developers know	Very flexible as almost all analytic-based

	need knowledge of U-SQL and C#	basic SQL	languages are supported
Reporting services	Power BI	Tableau, Power BI (if using Spark), Qlik	Tableau, open-source packages such as ggplot2, matplotlib, bokeh, etc.

5. Use case in all three platforms

Now, let's execute the same functionality in the three platforms with similar processing powers to see how they stack up against each other regarding duration and pricing:

In this case, let's imagine we have some HR data gathered from different sources that we want to analyse. On the one hand, we have a .CSV containing information about a list of employees, some of their characteristics, the employee source and their corresponding performance score.

On the other hand, from another source, we've gathered a .CSV that tells us how much we've invested in recruiting for each platform (Glassdoor, Careerbuilder, Website banner ads, etc).

Our goal is to build a fact table that aggregates employees and allows us to draw insights from their performance and their source, to pursue better recruitment investments.

5.1. ADLA:

In ADLA, we start off by storing our files in ADLS:

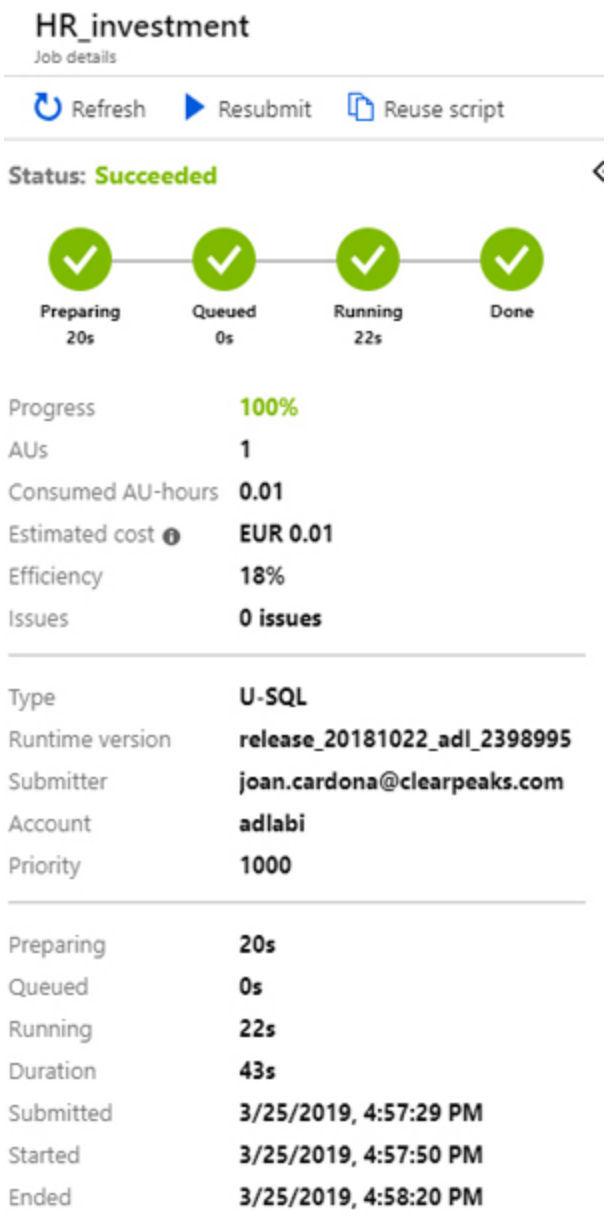
We then proceed to write the U-SQL script that will process the data in the Azure portal:

```
DECLARE @employee_file string = "/TEST/HR_Recruitment/employees.csv";
DECLARE @costs_file string = "/TEST/HR_Recruitment/recruiting_costs.csv";

DECLARE @fact_output string = "/TEST/HR_Recruitment/output/fact.csv";

// Input rowset extractions and column definition

@input_employee =
  EXTRACT
    [Employee Name] string
    ,[Employee Number] string
    ,[State] string
    ,[Zip] string
    ,[DOB] DateTime
    ,[Age] int
    ,[Sex] string
    ,[MaritalDesc] string
    ,[CitizenDesc] string
    ,[Hispanic/Latino] string
    ,[RaceDesc] string
    ,[Date of Hire] DateTime
    ,[Date of Termination] DateTime?
    ,[Reason For Term] string
    ,[Employment Status] string
    ,[Department] string
    ,[Position] string
    ,[Pay Rate] float
    ,[Manager Name] string
    ,[Employee Source] string
    ,[Performance Score] string
  FROM @employee_file
  USING Extractors.Csv(skipFirstNRows:1);

@input_costs =
  EXTRACT
    [Employee Source] string
    ,[January] int
    ,[February] int
```

```
    ,[March] int
    ,[April] int
    ,[May] int
    ,[June] int
    ,[July] int
    ,[August] int
    ,[September] int
    ,[October] int
    ,[November] int
    ,[December] int
    ,[Total] int
  FROM @costs_file
  USING Extractors.Csv(skipFirstNRows:1);

// We now grab the info we actually need and join by employee source:

@employee_performance =
  SELECT
    [Employee Number] AS emp_id,
    [Employee Source] AS emp_source,
    [Performance Score] AS performance_score
  FROM @input_employee;

@recruitment_costs =
  SELECT
    [Employee Source] AS emp_source,
    [Total] AS total_cost
  FROM @input_costs;

@fact_performance_recruitment_costs =
  SELECT
    ep.performance_score,
    rc.emp_source,
    rc.total_cost
  FROM @employee_performance AS ep
  INNER JOIN @recruitment_costs AS rc
  ON rc.emp_source == ep.emp_source;

@fact_performance_recruitment_costs =
  SELECT
```

```
    performance_score,
    emp_source,
    total_cost,
    COUNT(1) AS total_employees
    FROM @fact_performance_recruitment_costs
  GROUP BY
    performance_score,
    emp_source,
    total_cost;

// Output the file to ADLS

OUTPUT @fact_performance_recruitment_costs
TO @fact_output
USING Outputters.Csv(outputHeader:true,quoting:true);
```

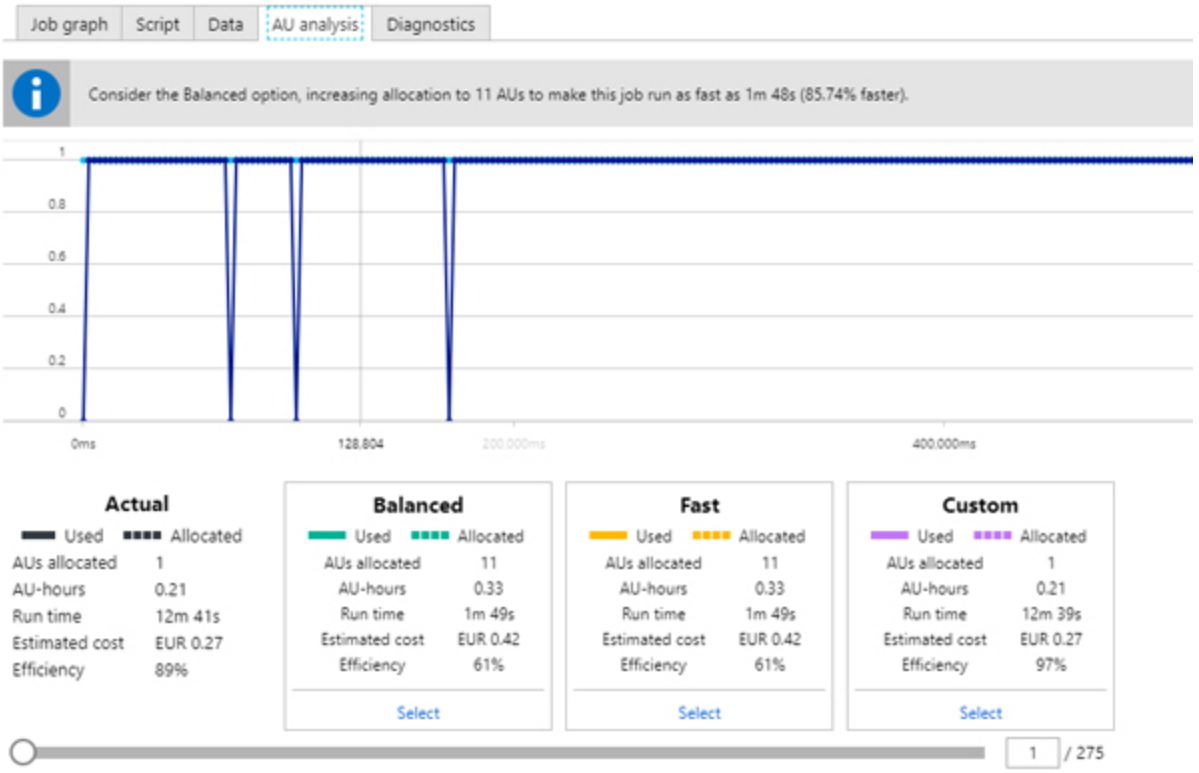
After running, we can monitor how this job was executed and how much it cost in the Azure Portal for ADLA:



As we can see, the total duration was 43 seconds and it had an approximate cost of 0.01€.

The ADLA Service offers a neat functionality that tells us the efficiency of any job after running it, so we know if it's worth augmenting or reducing the AUs of the job (the computing power).

Here we can see another job with 1 allocated AU: it recommends increasing the AUs for the job, so it runs 85.74% faster, but it also costs more.



It's worth considering, but in cases like this, higher speed is unnecessary, and we prefer the reduced costs.

5.2. HDInsight:

In this case, we store the same files in ADLS and execute a HiveQL script with the same functionality as before:

```
CREATE TABLE IF NOT EXISTS hr_recruitment_employees

(
    EmployeeName varchar(500)
    ,EmployeeNumber varchar(500)
    ,State varchar(50)
    ,Zip varchar(500)
    ,DOB varchar(500)
    ,Age int
    ,Sex varchar(50)
    ,MaritalDesc varchar(50)
    ,CitizenDesc varchar(50)
    ,Hispanic_Latino varchar(50)
    ,RaceDesc varchar(50)
```

```
	,DateofHire varchar(50)
	,DateofTermination varchar(50)
	,ReasonForTerm varchar(50)
	,EmploymentStatus varchar(50)
	,Department varchar(50)
	,Position varchar(50)
	,PayRate float
	,ManagerName varchar(50)
	,EmployeeSource varchar(50)
	,PerformanceScore varchar(50)
)

ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
  "separatorChar" = ",",
  "quoteChar"     = "\""    --"
)
STORED AS TEXTFILE LOCATION
    'adl://cphdinsight.azuredatalakestore.net/TEST/HR_Recruitment/HDI/input_employees'
tblproperties ("skip.header.line.count"="1");

CREATE TABLE IF NOT EXISTS hr_recruitment_costs
(
	EmployeeSource string
	,January int
	,February int
	,March int
	,April int
	,May int
	,June int
	,July int
	,August int
	,September int
	,October int
	,November int
	,December int
	,Total int
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
WITH SERDEPROPERTIES (
```

```
  "separatorChar" = ",",
  "quoteChar"     = "\""    --"
)
STORED AS TEXTFILE LOCATION
    'adl://cphdinsight.azuredatalakestore.net/TEST/HR_Recruitment/HDI/input_costs'
tblproperties ("skip.header.line.count"="1");

CREATE TABLE fact_perf_recruitment_costs AS
select
        PerformanceScore,
        EmployeeSource,
        Total,
        count(1) as TotalEmployees
        from
(select
        ep.PerformanceScore,
        rc.EmployeeSource,
        rc.Total
from hr_recruitment_employees ep
inner join hr_recruitment_costs rc
on ep.EmployeeSource = rc.EmployeeSource) joined_rc
group by
        PerformanceScore,
        EmployeeSource,
        Total;
```

In this case the duration of the creation of the two temporary tables and their join to generate the fact took approximately 16 seconds:

```
Query Execution Summary
----------------------------------------------------------------------------------
OPERATION                            DURATION
----------------------------------------------------------------------------------
Compile Query                           0.47s
Prepare Plan                            6.58s
Submit Plan                             0.64s
Start DAG                               0.47s
Run DAG                                 8.00s
----------------------------------------------------------------------------------
```

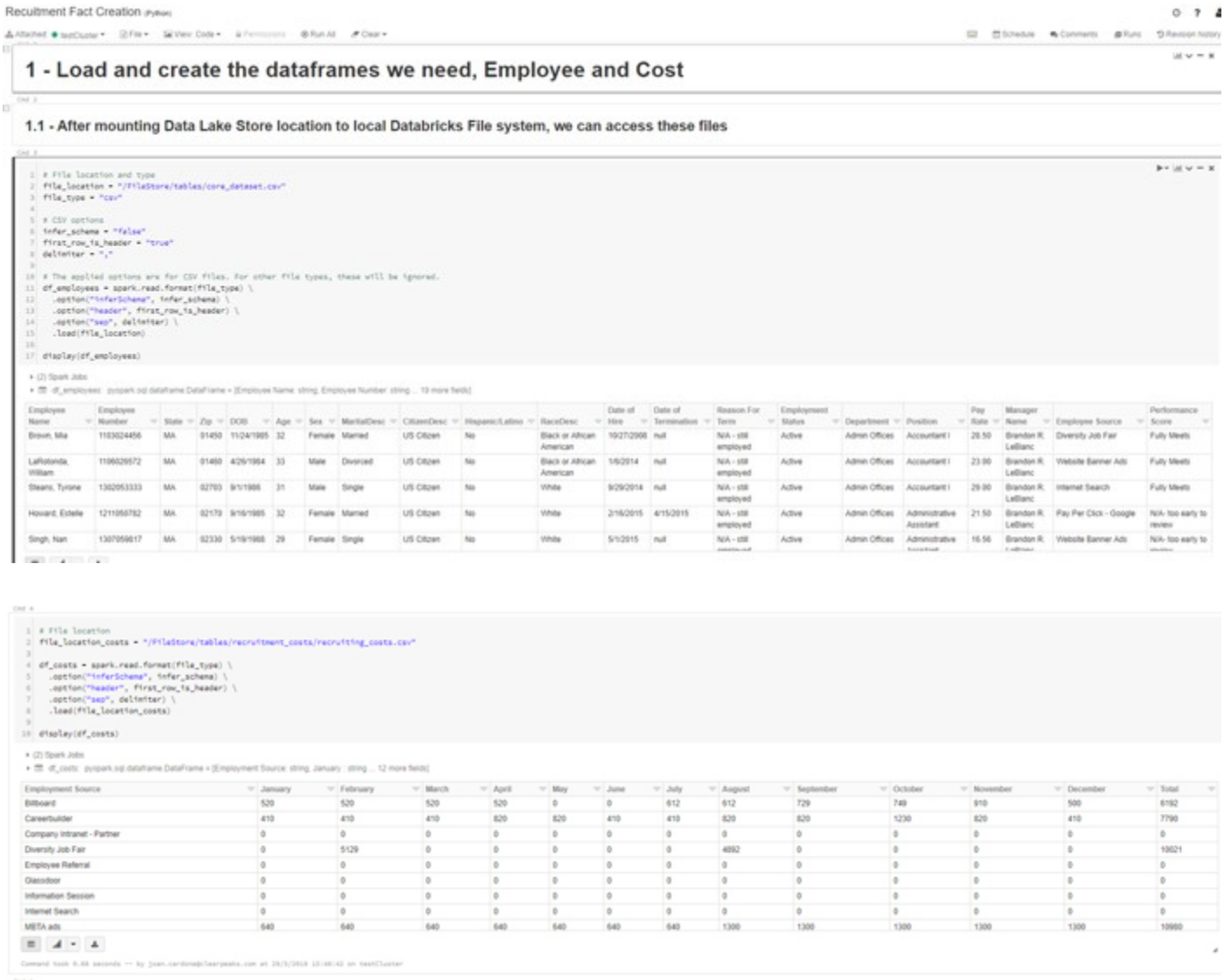
Taking into account the Azure VMs we're using (2 D13v2 as heads and 2 D12v2 as workers), following the pricing information (https://azure.microsoft.com/en-au/pricing/details/hdinsight/) this activity cost approximately 0.00042 €, but as

HDInsight is not an on-demand service, we should remember that per-job pricings are not as meaningful as they were in ADLA.

Another important thing to mention is that we are running Hive in HDInsight. As Hive is based on MapReduce, small and quick processing activities like this are not its strength, but it shines in situations where data volumes are much bigger and cluster configurations are optimized for the type of jobs they must execute.

5.3. Databricks:

Finally, after loading data from ADLS using a Mount point, we execute a notebook to obtain a table which can be accessed by anyone with credentials to use the cluster and its interactive notebooks:



Now that we have both dataframes, we can procede to join them using PySpark and generate our Fact table

```
employees = df_employees.select(df_employees["Employee Source"].alias("EmployeeSource"), df_employees["Performance Score"].alias("PerformanceScore"))
costs = df_costs.select(df_costs["Employment Source"].alias("EmploymentSource"), "Total")

rc = costs.alias('rc')
em = employees.alias('em')

inner_join = em.join(rc, em['EmployeeSource'] == rc['EmploymentSource'])
inner_join.drop("EmployeeSource")

fact_performance_recruitment = inner_join.groupBy("PerformanceScore", "EmployeeSource", "Total").count()
fact_performance_recruitment.show()
```

```
▸ (4) Spark Jobs
▸ employees: pyspark.sql.dataframe.DataFrame = [EmployeeSource: string, PerformanceScore: string]
▸ costs: pyspark.sql.dataframe.DataFrame = [EmploymentSource: string, Total: string]
▸ rc: pyspark.sql.dataframe.DataFrame = [EmploymentSource: string, Total: string]
▸ em: pyspark.sql.dataframe.DataFrame = [EmployeeSource: string, PerformanceScore: string]
▸ inner_join: pyspark.sql.dataframe.DataFrame = [EmployeeSource: string, PerformanceScore: string ... 2 more fields]
▸ fact_performance_recruitment: pyspark.sql.dataframe.DataFrame = [PerformanceScore: string, EmployeeSource: string ... 2 more fields]
+--------------------+--------------------+-----+
|   PerformanceScore|      EmployeeSource|count|
+--------------------+--------------------+-----+
|N/A- too early to...|  Website Banner Ads| 7143|    2|
|       Fully Meets|           Billboard| 6192|   10|
|       90-day meets|      Vendor Referral|    0|    2|
|       90-day meets| Diversity Job Fair|10821|    2|
|       Fully Meets|   Employee Referral|    0|   16|
|       90-day meets|               Other| 3995|    2|
|        Exceptional|           Billboard| 6192|    1|
|       Fully Meets|       Word of Mouth|    0|    8|
|       90-day meets|   Employee Referral|    0|    5|
|       Fully Meets|     Internet Search|    0|    4|
|                PIP|  Website Banner Ads| 7143|    2|
```

We finally save it as a table to be accessed by anyone who needs it, and queries can be launched at it using SQL, to make it easier for users who know one but not the other:

```
fact_performance_recruitment.write.saveAsTable("fact_performance_recruitment")
```

▸ (1) Spark Jobs

Command took 5.44 seconds -- by joan.cardona@clearpeaks.com at 29/3/2019 14:22:32 on testCluster

Cmd 8

```
%sql
SELECT PerformanceScore, EmployeeSource, count, Total AS RecruitmentTotal FROM fact_performance_recruitment ORDER BY PerformanceScore, EmployeeSource
```

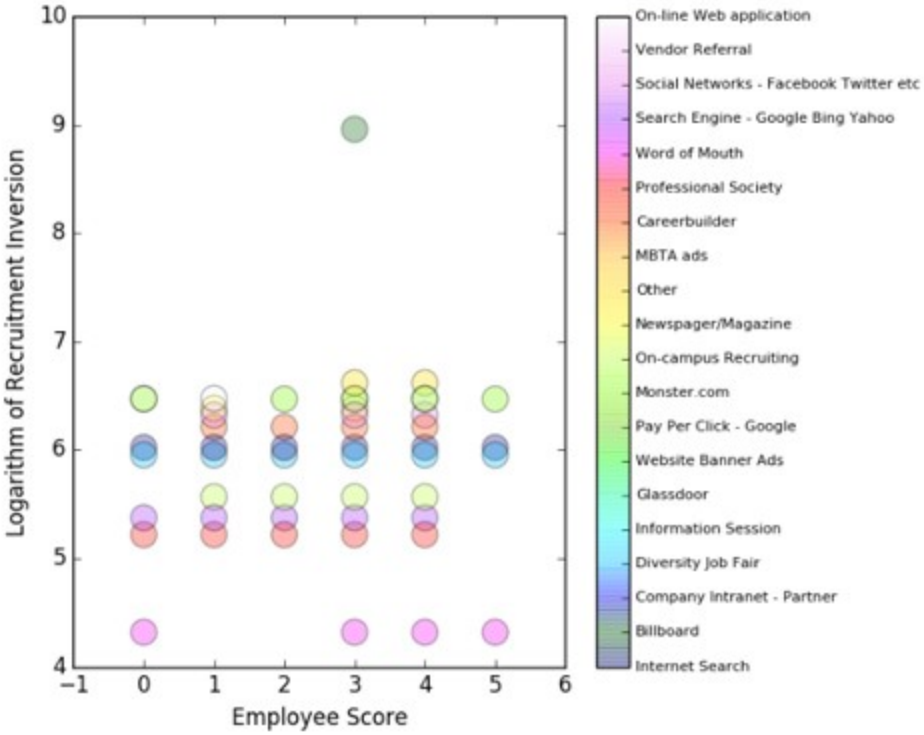
▸ (1) Spark Jobs

PerformanceScore	EmployeeSource	count
90-day meets	Billboard	1
90-day meets	Diversity Job Fair	2
90-day meets	Employee Referral	5
90-day meets	Glassdoor	2
90-day meets	Monster.com	2
90-day meets	Newspaper/Magazine	4
90-day meets	Other	2
90-day meets	Pay Per Click - Google	2
90-day meets	Search Engine - Google Bing Yahoo	2

Command took 1.44 seconds -- by joan.cardona@clearpeaks.com at 29/3/2019 14:49:56 on testCluster

Having these final fact tables, plus the ease of running a quick analysis in our notebook, we can answer questions like “Where are we, as a company, getting our better performers and how much are we spending on those platforms?” This can help companies detect steep spending without many returns so as to avoid them, or invest more money where the better performers come from:

Using Pandas and Matplotlib inside the notebook, we can sketch the answer to this question and draw our corresponding insights:



It seems balanced, but we can see that too much has been spent on Billboard advertising for just one recruit whose performance is only middling.

In this case, the VMs we're using are 3 Standard_D3_v2, and the notebook took a total of approximately 5 seconds, which in pricing information reflects a total of 0.00048 €. In this case, however, Spark is optimized for these types of job, and bearing in mind that the creators of Spark built Databricks, there's reason to believe it would be more optimized than other Spark platforms.

6. Big Data Situations

Until now we've seen how these systems deal with reasonably small datasets. To fully unleash their potential, we will proceed to study how they react to a much bigger file with the same schema and comment on their behaviour. The employee file size is now 9.5 GB, but the script will be the same.

6.1. ADLA:

When executing the ADLA job, these are the results we obtain:

In this case, we allocated 10 AUs for the job, but we see that the AU analysis gives us a more balanced option of 6 AUs that takes a little longer but is also cheaper. The total cost was 0.18€ just for this one job.

6.2. HDInsight:

In HDInsight we execute the same query with the larger dataset in the same configuration we used before to compare pricings (which are based on cluster times) and we achieve the following Query Execution Summary:

```
Query Execution Summary
----------------------------------------------------------------------------------
OPERATION                                 DURATION
----------------------------------------------------------------------------------
Compile Query                               2.61s
Prepare Plan                               11.10s
Submit Plan                                 0.82s
Start DAG                                   0.63s
Run DAG                                  1140.63s
----------------------------------------------------------------------------------
```

In this case the query took approximately 20 minutes. According to the pricings of the cluster configuration we are using, this corresponds to an estimated cost of 0.63 €. In this case it's clear we should use a more powerful cluster configuration in order to balance out the time of execution; if we had to run a

lot of tasks like this, each would need to take much less than 20 minutes. This is a good example of when scaling becomes tedious: since we now know that this cluster is not appropriate for our use case, we must eliminate the cluster and create a new one and see if it's what we're looking for. We can also see that it's about 4 times more expensive than the ADLA job, as well as not showing us what an appropriate cluster configuration would be.

6.3. Databricks:

In the Databricks service, we create a cluster with the same characteristics as before, but now we upload the larger dataset to observe how it behaves compared to the other services:

```
▸ (8) Spark Jobs
▸ df_costs_2: pyspark.sql.dataframe.DataFrame = [Employment Source: string, January : integer ... 12 more fields]
▸ df_employees_2: pyspark.sql.dataframe.DataFrame = [Employee Name: string, Employee Number: integer ... 19 more fields]
▸ employees: pyspark.sql.dataframe.DataFrame = [EmployeeSource: string, PerformanceScore: string]
▸ costs: pyspark.sql.dataframe.DataFrame = [EmploymentSource: string, Total: integer]
▸ rc: pyspark.sql.dataframe.DataFrame = [EmploymentSource: string, Total: integer]
▸ em: pyspark.sql.dataframe.DataFrame = [EmployeeSource: string, PerformanceScore: string]
▸ inner_join: pyspark.sql.dataframe.DataFrame = [EmployeeSource: string, PerformanceScore: string ... 2 more fields]
▸ fact_performance_recruitment: pyspark.sql.dataframe.DataFrame = [PerformanceScore: string, EmployeeSource: string ... 2 more fields]
+--------------------+--------------------+-----+--------+
|    PerformanceScore|      EmployeeSource|Total|   count|
+--------------------+--------------------+-----+--------+
|             Exceeds|  Diversity Job Fair|10021|  750005|
|        90-day meets|           Glassdoor|    0|  300002|
|         Fully Meets|               Other| 3995|  450003|
|         Exceptional|Professional Society| 1200|  300002|
|                 PIP|Pay Per Click - G...| 3509|  150001|
|        90-day meets|           Billboard| 6192|  150001|
|         Fully Meets|       Word of Mouth|    0| 1200008|
|         Fully Meets|     Internet Search|    0|  600004|
|         Fully Meets|   Employee Referral|    0| 2400016|
|N/A- too early to...|  Diversity Job Fair|10021|  450003|
|         Fully Meets|       Careerbuilder| 7790|  150001|
|N/A- too early to...|     Vendor Referral|    0|  600004|
|   Needs Improvement|       Word of Mouth|    0|  150001|
|N/A- too early to...|Search Engine - G...| 5183|  150001|
|         Fully Meets|  Diversity Job Fair|10021| 2100014|
|        90-day meets|         Monster.com| 5760|  300002|
|N/A- too early to...|       Word of Mouth|    0|  300002|
|        90-day meets|               Other| 3995|  300002|
|        90-day meets|Search Engine - G...| 5183|  300002|
|N/A- too early to...|     Internet Search|    0|  150001|
+--------------------+--------------------+-----+--------+
only showing top 20 rows

Command took 6.82 minutes -- by joan.cardona@clearpeaks.com at 8/4/2019 14:01:51 on blogClusterDb
```

As we can see, the whole process took approximately 7 minutes, more than twice as fast as HDInsight with a similar cluster configuration. In this case, the job cost approximately 0.04€, a lot less than HDInsight. This is a good example of how Spark jobs can generally run faster than Hive queries.

Conclusion

As we have seen, each of the platforms works best in different types of situation: ADLA is especially powerful when we do not want to allocate any amount of time to administrating a cluster, and when ETLs are well defined and are not subject to many changes over time. It also helps if developers are familiar with C# to get the full potential of U-SQL.

On the other hand, HDInsight has always been very reliable when we know the workloads and the cluster sizes we'll need to run them. Scaling in this case is tedious, are machines must be deleted and activated iteratively until we find the right choice. Using Hive is a perk, as its being open source and very similar to SQL allows us to get straight down to developing without further training. By using Hive, we take full advantage of MapReduce power, which shines in situations where there are huge amounts of data.

And finally, Databricks seems an ideal choice when the notebook interactive experience is a must, when data engineers and data scientists must work together to get insights from data and adapt smoothly to different situations, as scalability is extremely easy. Another perk of using Databricks is its speed, thanks to Spark.

If you'd like to know more about the questions raised in this brief article, please don't hesitate to contact us here at ClearPeaks – we´ll be glad to help!

q) use of active directory app registration?

Purpose of app registration

- It is used to integrate the application and service with Azure AD.
- Using Azure App, we can generate the token to authenticate the application.
- If we want to use the Azure AD capabilities, we must register the app.
- After we register the app, we can get the "Client ID, Secret key".

Steps to register the new application in Azure AD

Follow the below-listed steps to register the application.

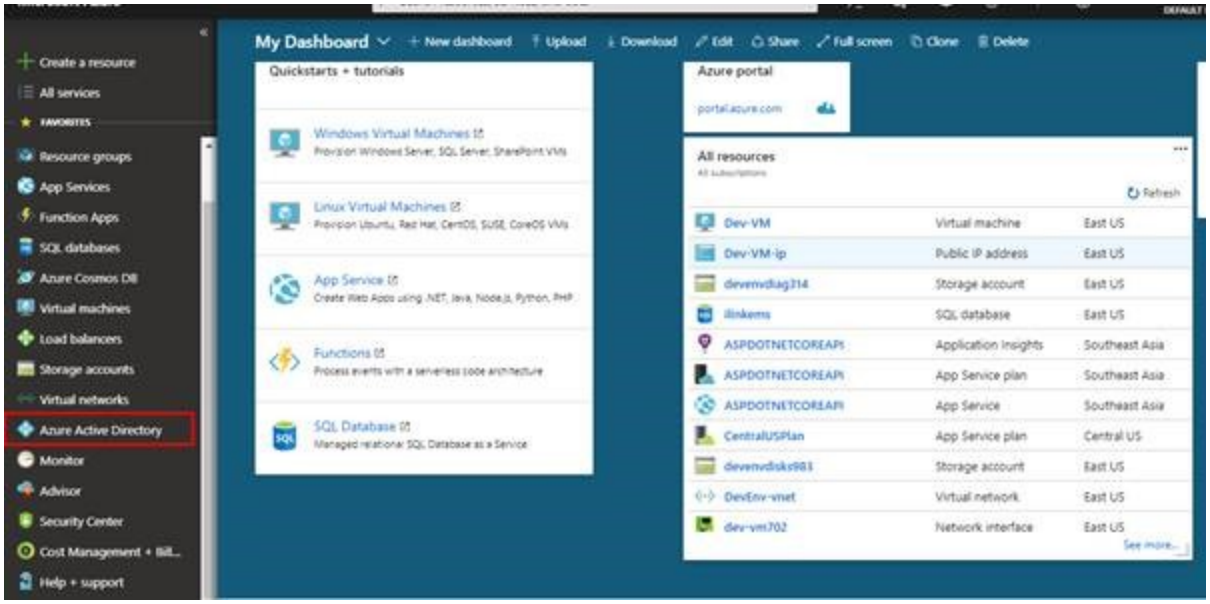
Step 1

Log into the Azure portal using your Azure account.

URL - *https://portal.azure.com/*

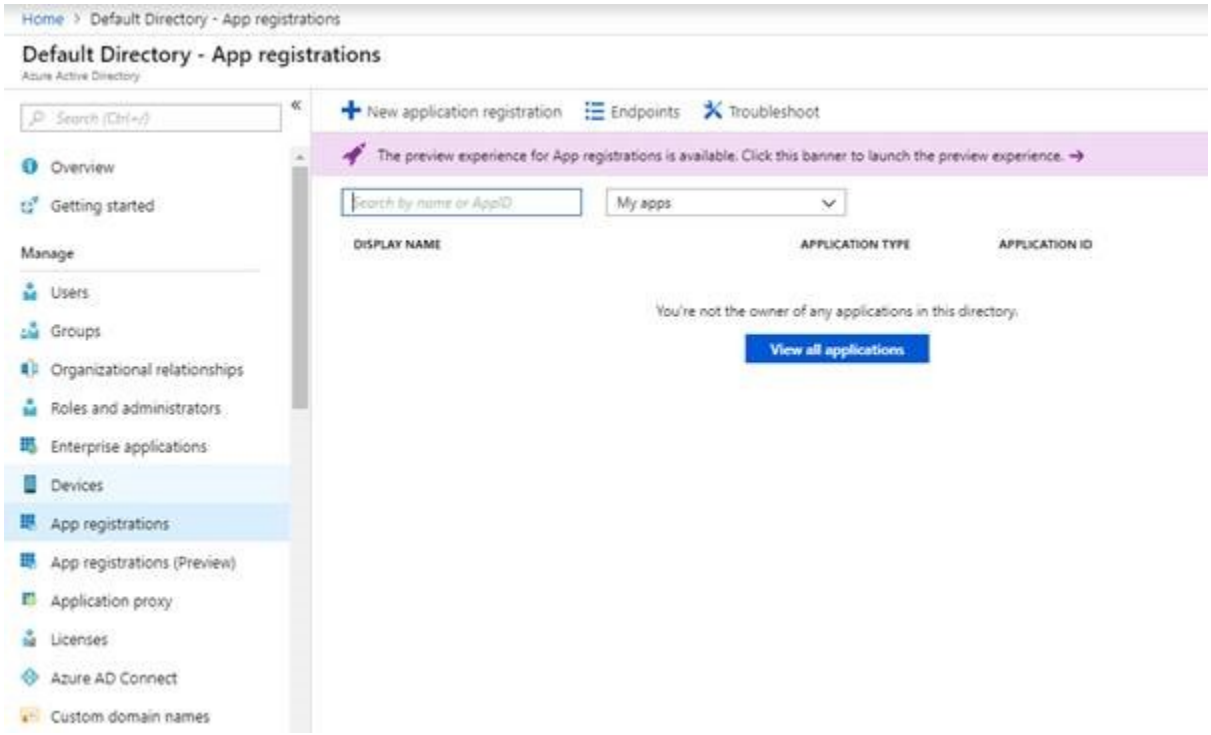
Step 2

Select Azure Active Directory from the left navigation.



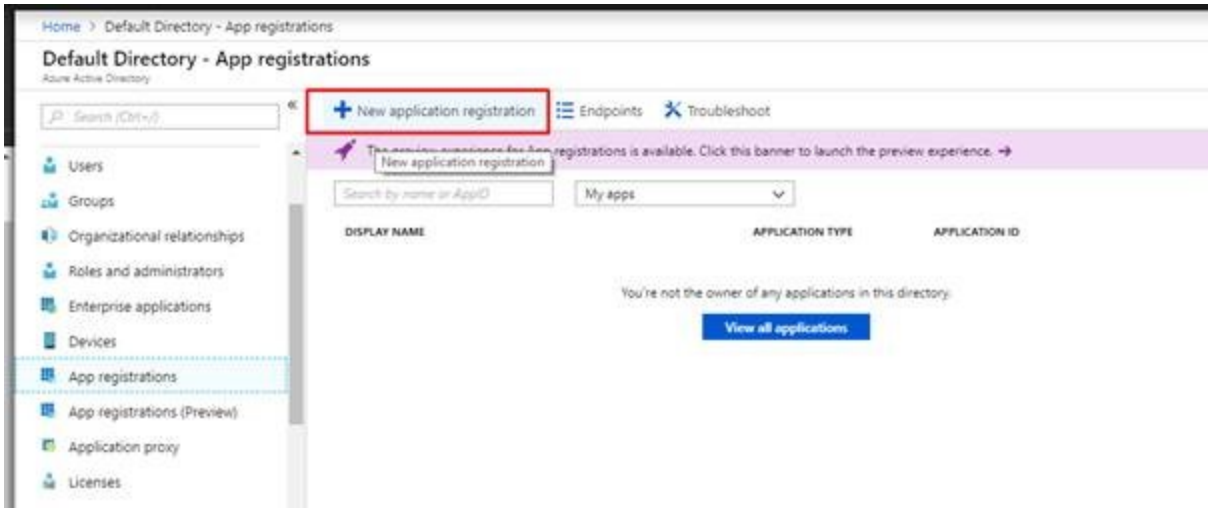
Step 3

On the "Default directory" page, select the "*App registrations"* from the left panel, as shown below.



Step 4

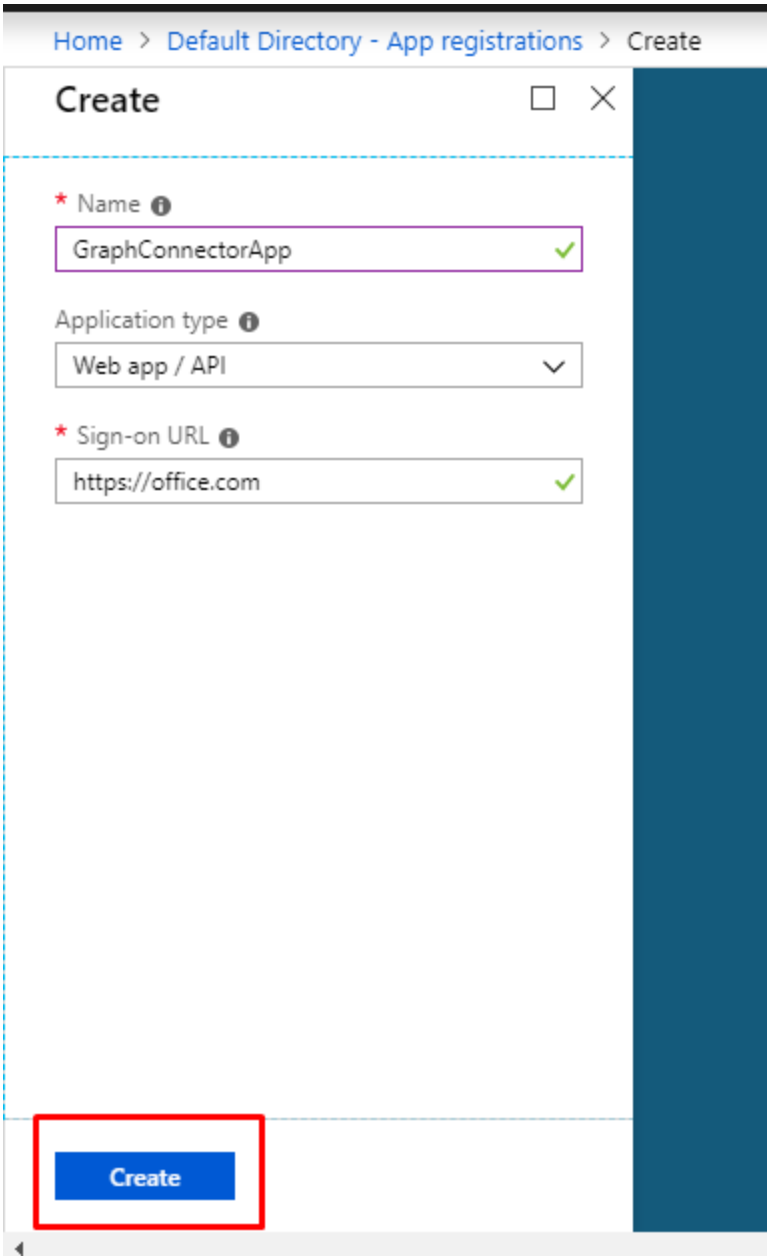
From App Registration pane, click NewApplication Registration option.



Step 5

On the "Create" pane, type in the following information and then click the "Create" button.

Field	Description	Sample
Name	Name for the new application. Type in the desired application name.	GraphConnectorApp
Application type	Leave as Web app/API	Web app/API
Sign in URL	Login URL. (To get the data from Graph API, we no need to provide the proper login URL. Just give office URL)	https://office.com



Step 6

Once the application is created, edit the manifest file and change the value of oauth2AllowImplicitFlow parameter to true.

Before	

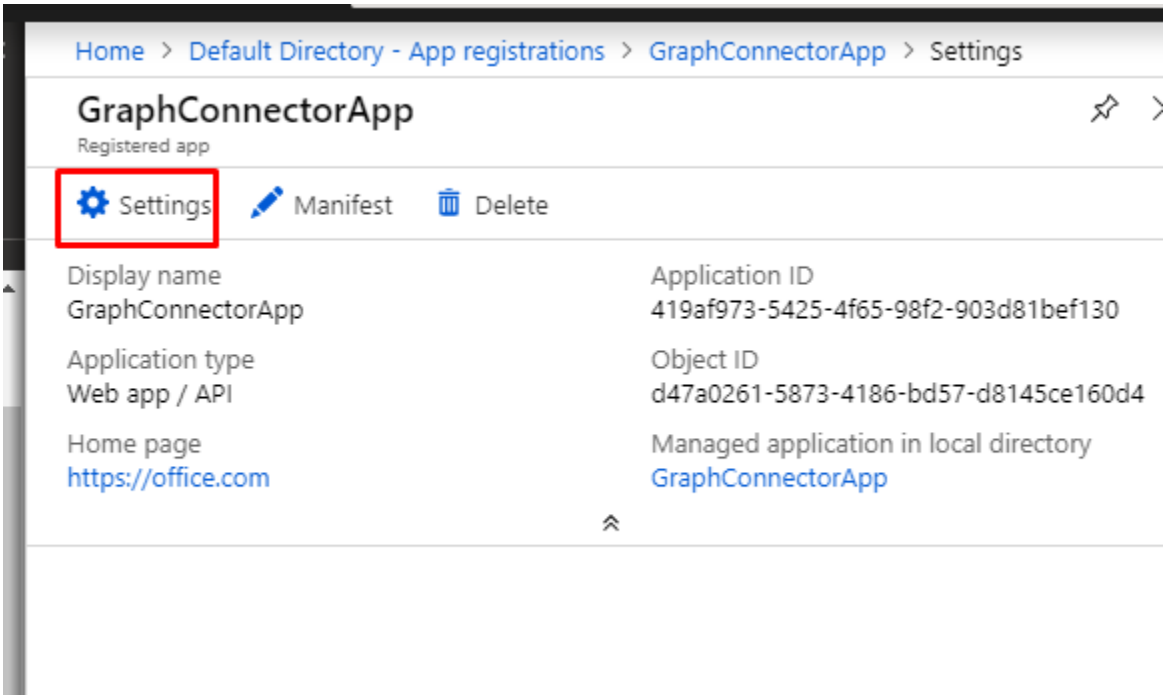
After

Step 7

Save the changes first and then click X to close the pane.

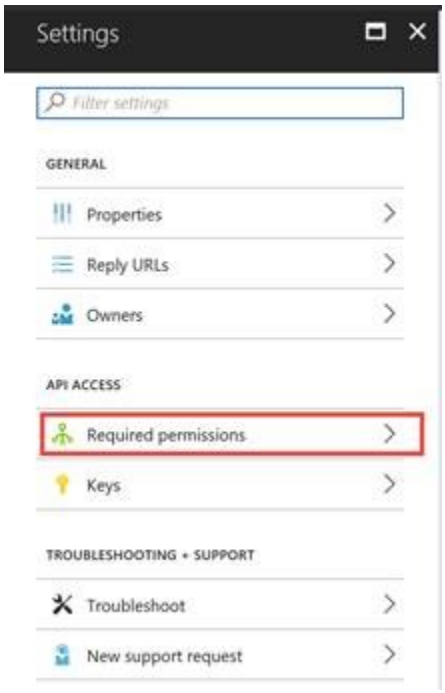
Step 8

Then, we need to add the permissions. From Registered App pane, click the "Settings" option.



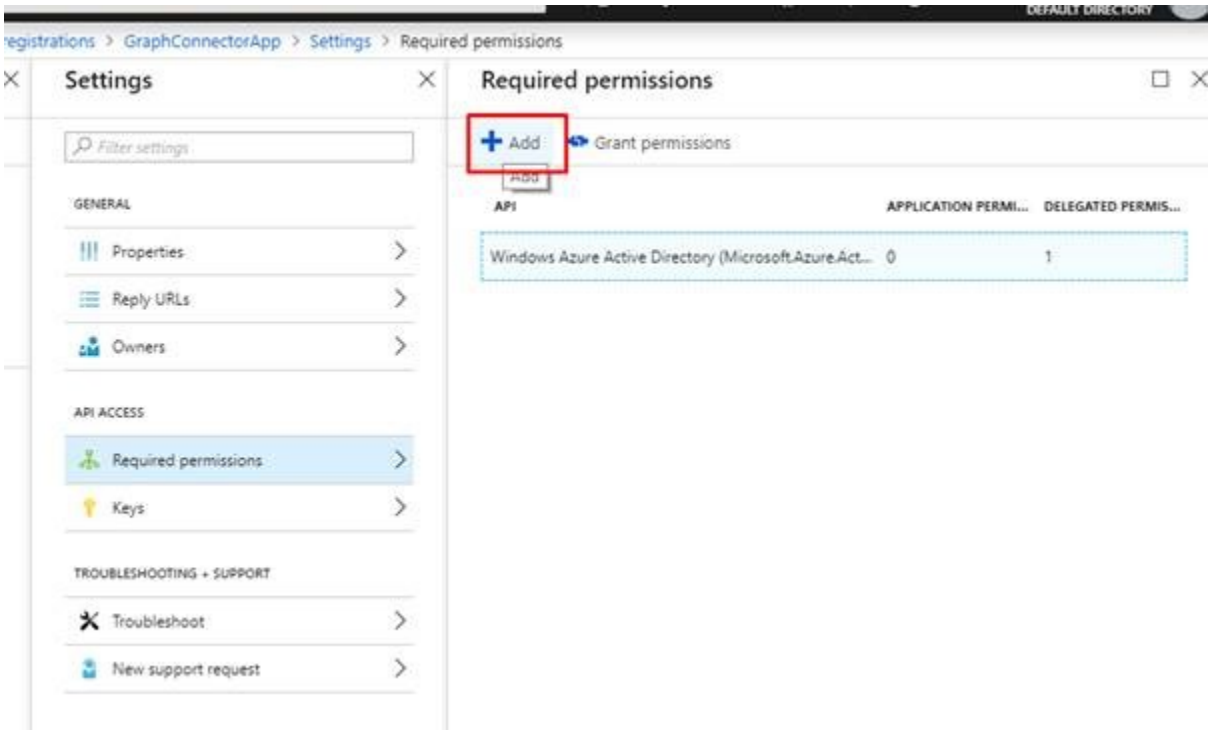
Step 9

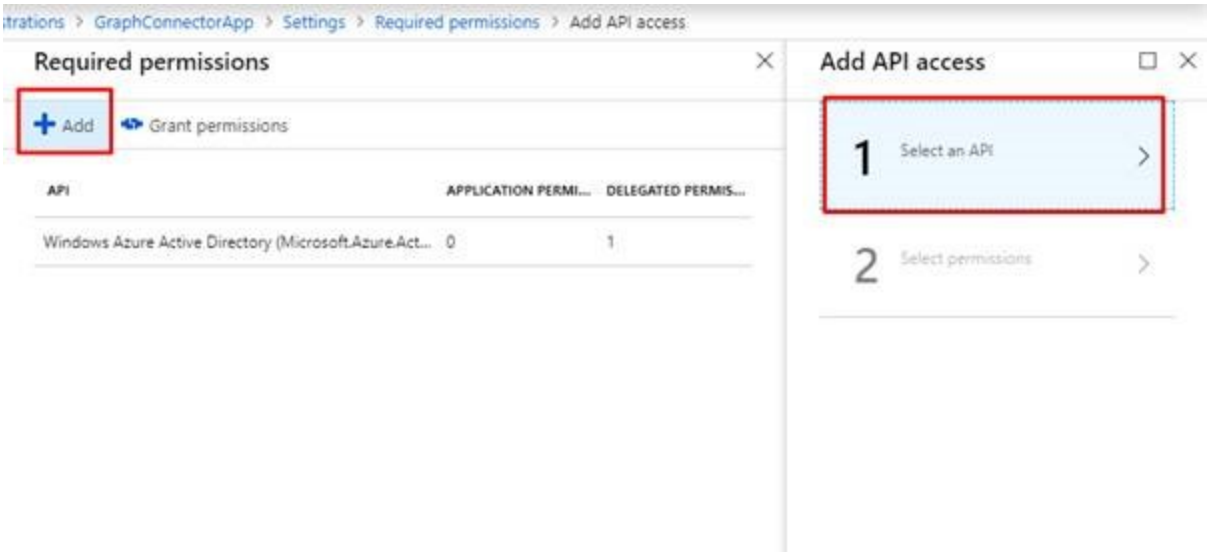
From Settings pane, click the "Required Permissions" option.



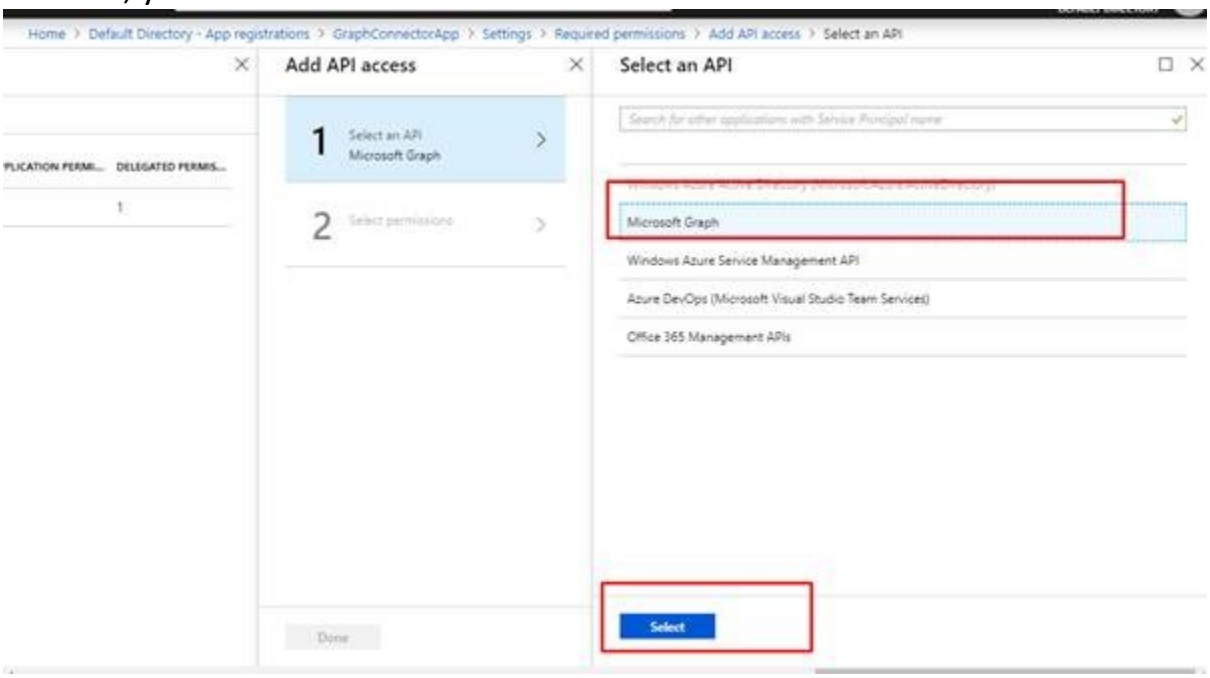
Step 10

Now, click on + Add, then choose the “Select API”.





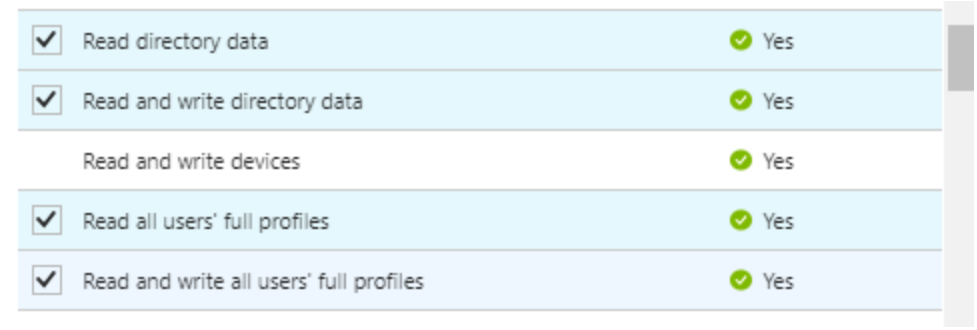
I've planned to use the new app for Graph API so I've selected Microsoft Graph. As per your need, you can select a different API.

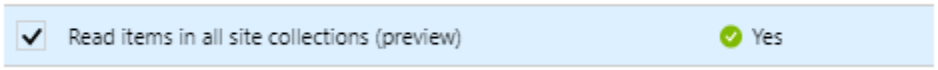


Step 11

From the "Enable Access" pane, place a check in the checkbox in front of the following permissions from the Application Permissions and Delegated Permissions sections. Then, click the "Select" button.

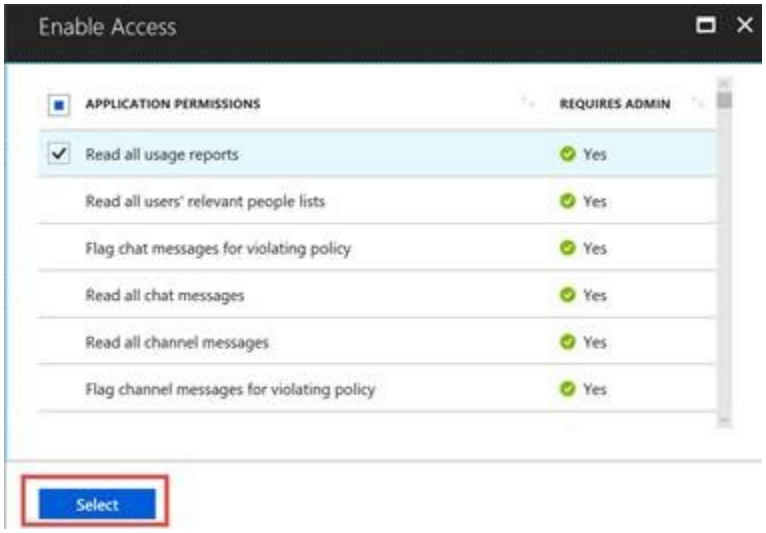
Application Permissions





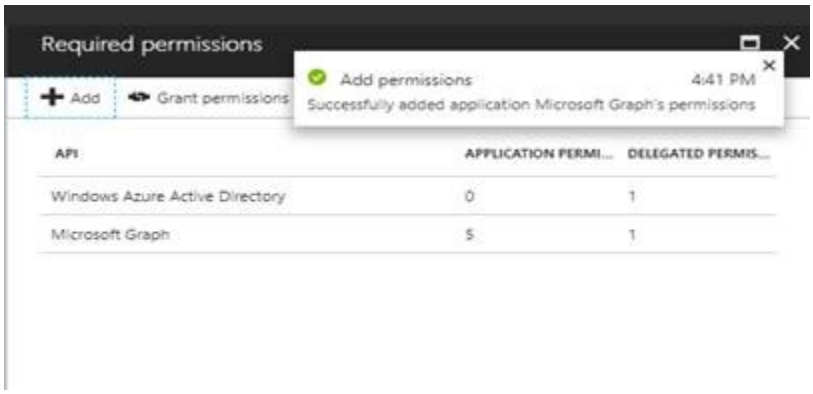
Delegated Permissions

Save the permissions.



Step 12

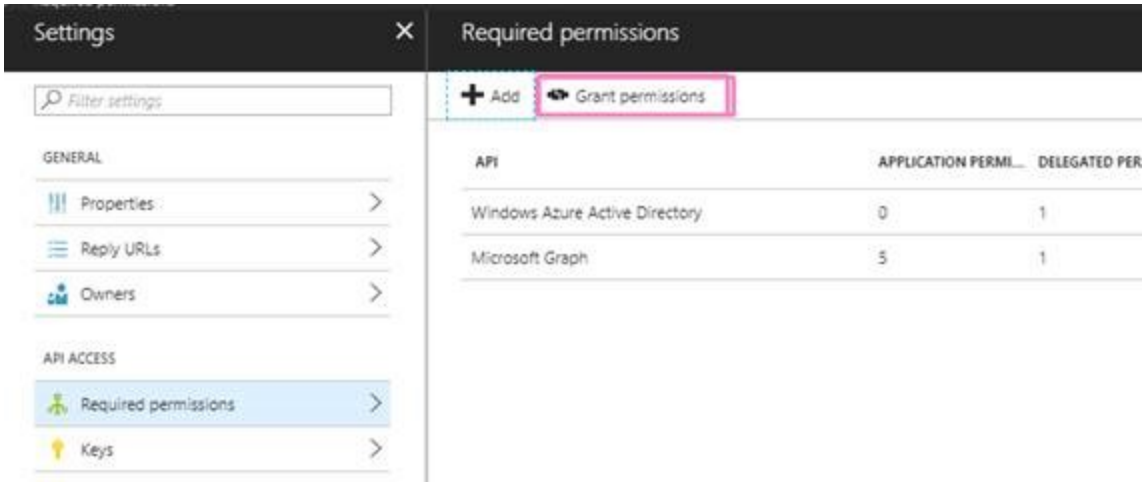
From Add API Access pane, verify the count of roles and scope. Then click the Done button.

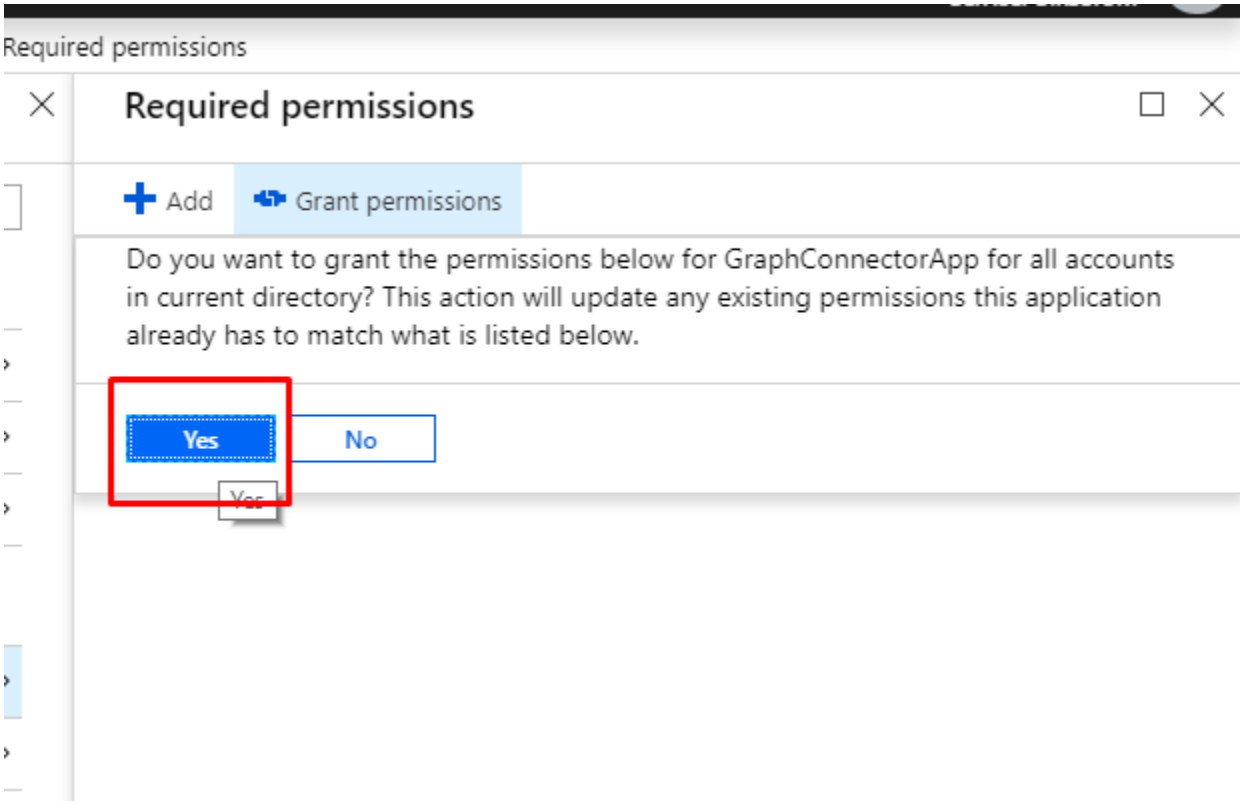


Step 13

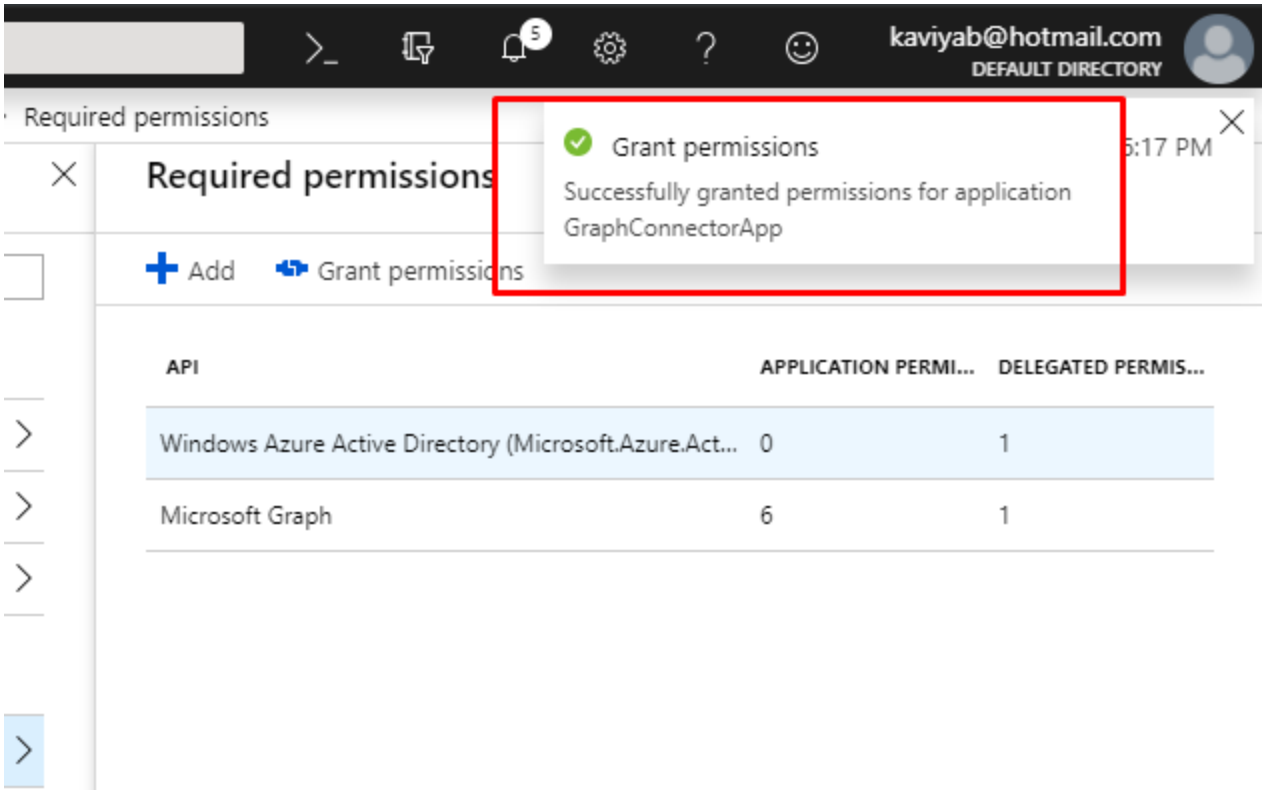
On Required Permissions pane, click the "Grant Permissions" option.

Note: Only Azure directory admin is able to do this.





Finally, you have granted permission to the Azure app.



Generate Secret key

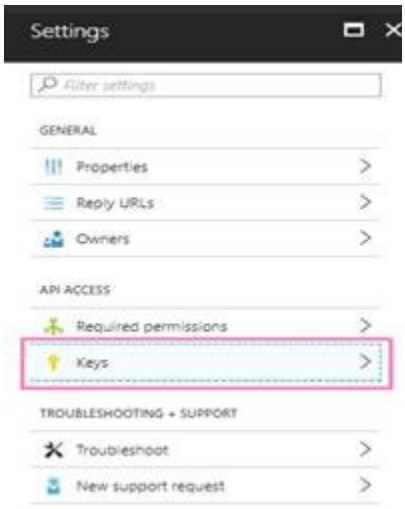
Step 1

From Azure Active Directory Admin Centre, open the created application and click Settings option.

Log into https://aad.portal.azure.com/ > Azure Active Directory > App Registrations > select the created application name

Step 2

From Settings pane, click Keys option.



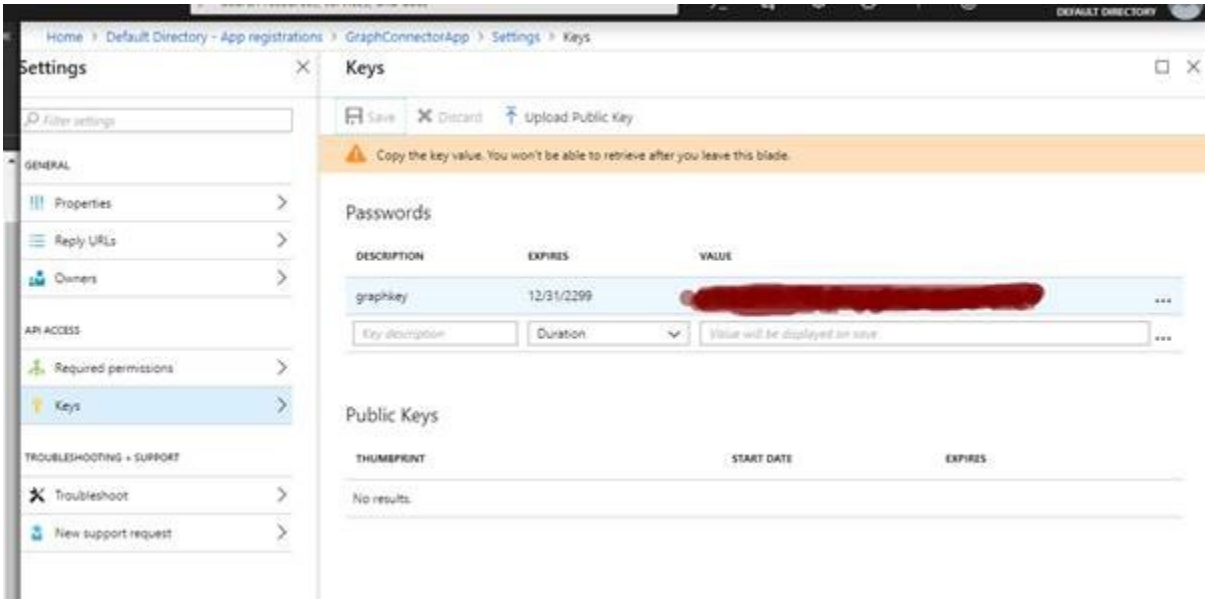
Step 3

From Keys pane, type in the following information and then click the "Save" button.

Field	Description	Example
Description	Name for the key. Type in a descriptive name.	Secret Key
Expires On	Select Never expire option.	Never expire
Value	You will get the key when you click on the save	

Step 4

From the Keys pane, copy the encoded key value. This key value cannot be retrieved after leaving this pane. This encoded key value is the Client Secret Key that will be a part of the authentication credential.



Then, close the key pane.

Steps to get the Tenant ID

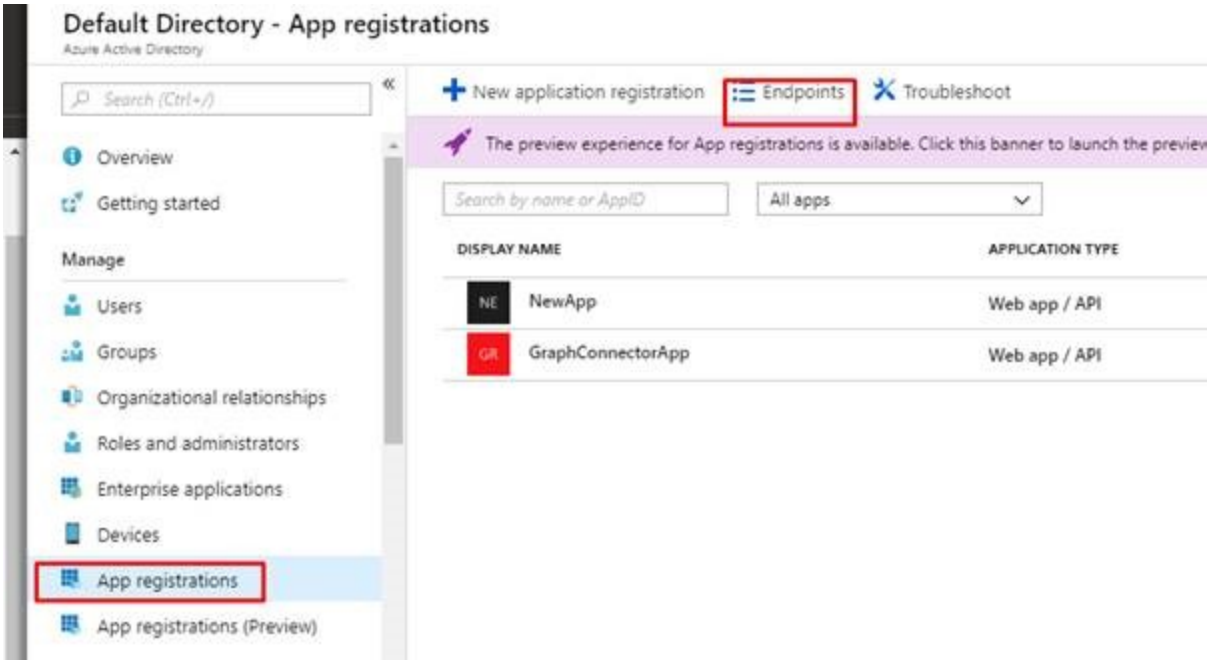
Step 1

From Azure Active Directory Admin Center, navigate to the App Registrations pane.

Log into https://aad.portal.azure.com/ > Azure Active Directory > App Registrations

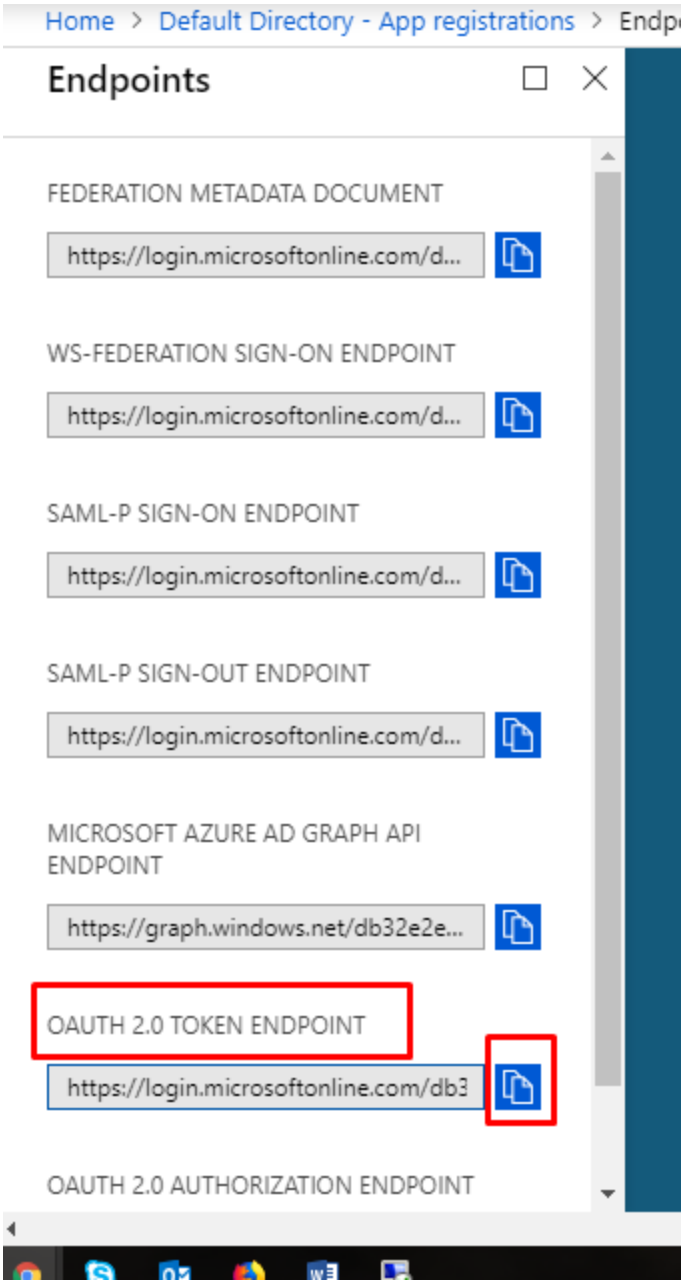
Step 2

From App Registrations pane, click Endpoints option.



Step 3

From Endpoints pane, click on the copy icon next to OAuth 2.0 Token Endpoint option and save the value.



From the copied endpoint URL, copy the value between microsoftonline.com/ ….and / oauth2/token. This is the Tenant ID that will be part of the authentication credential.

Get the Client ID

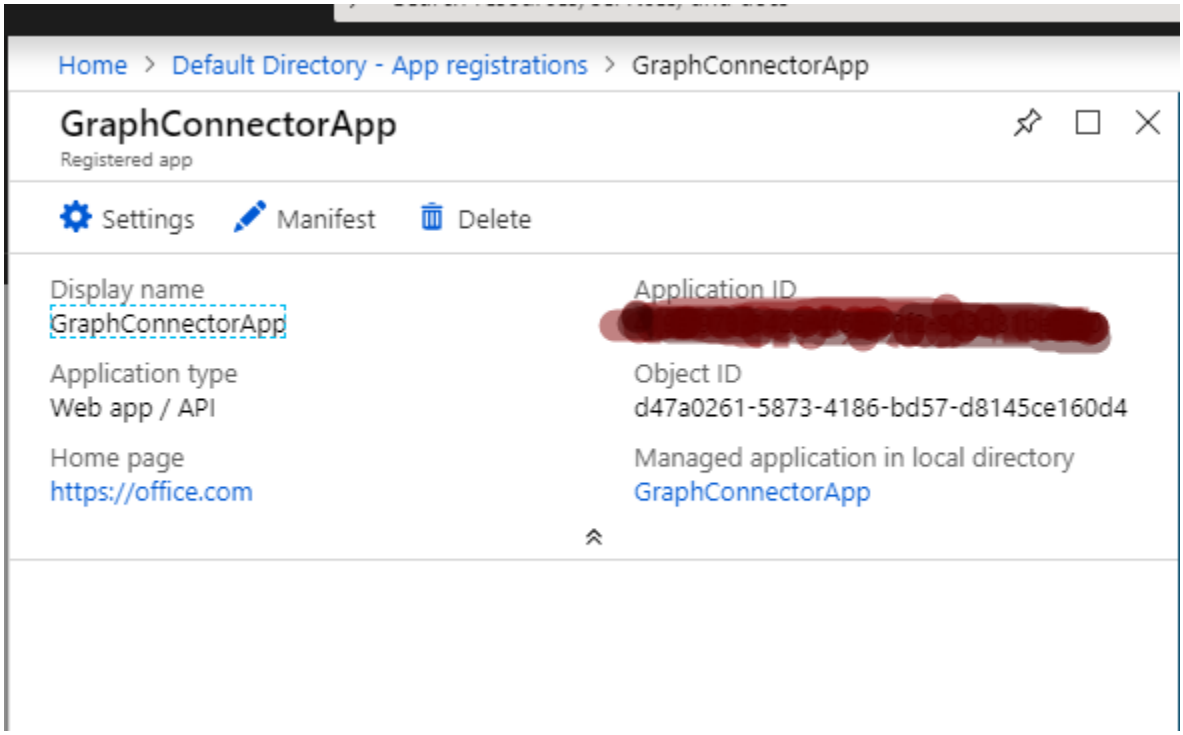
Step 1

From Azure Active Directory Admin Center, open the created application and click Settings option.

Log into https://aad.portal.azure.com/ > Azure Active Directory > App Registrations > select the created application name .

Step 2

From Settings pane, copy the Application ID value. This is the Client ID that will be part of the authentication credential.



Summary

In this article, we have explored how to register an app in Azure active directory. We also saw that we need client ID, secret key and Tenant ID to generate the oAuth token for Graph API.

Happy learning.

q) use of active directory custom domain?

Managing custom domain names in your Azure Active Directory

- 01/31/2019
- 5 minutes to read
-
-
-
-
-
-

A domain name is an important part of the identifier for many directory resources: it's part of a user name or email address for a user, part of the address for a group, and is sometimes part of the app ID URI for an application. A resource in Azure Active Directory (Azure AD) can include a domain name that's owned by the directory that contains the resource. Only a Global Administrator can manage domains in Azure AD.

Set the primary domain name for your Azure AD directory

When your directory is created, the initial domain name, such as 'contoso.onmicrosoft.com,' is also the primary domain name. The primary domain is the default domain name for a new user when you create a new user. Setting a primary domain name streamlines the process for an administrator to create new users in the portal. To change the primary domain name:

1. Sign in to the Azure portal with an account that's a Global Administrator for the directory.
2. Select **Azure Active Directory**.
3. Select **Custom domain names**.

4. Select the name of the domain that you want to be the primary domain.
5. Select the **Make primary** command. Confirm your choice when prompted.

You can change the primary domain name for your directory to be any verified custom domain that isn't federated. Changing the primary domain for your directory won't change the user name for any existing users.

Add custom domain names to your Azure AD tenant

You can add up to 900 managed domain names. If you're configuring all your domains for federation with on-premises Active Directory, you can add up to 450 domain names in each directory.

Add subdomains of a custom domain

If you want to add a third-level domain name such as 'europe.contoso.com' to your directory, you should first add and verify the second-level domain, such as contoso.com. The subdomain is automatically verified by Azure AD. To see that the subdomain you added is verified, refresh the domain list in the browser.

What to do if you change the DNS registrar for your custom domain name

If you change the DNS registrars, there are no additional configuration tasks in Azure AD. You can continue using the domain name with Azure AD without interruption. If you use your custom domain name with Office 365, Intune, or other services that rely on custom domain names in Azure AD, see the documentation for those services.

Delete a custom domain name

You can delete a custom domain name from your Azure AD if your organization no longer uses that domain name, or if you need to use that domain name with another Azure AD.

To delete a custom domain name, you must first ensure that no resources in your directory rely on the domain name. You can't delete a domain name from your directory if:

- Any user has a user name, email address, or proxy address that includes the domain name.
- Any group has an email address or proxy address that includes the domain name.
- Any application in your Azure AD has an app ID URI that includes the domain name.

You must change or delete any such resource in your Azure AD directory before you can delete the custom domain name.

ForceDelete option

You can **ForceDelete** a domain name in the Azure AD Admin Center or using Microsoft Graph API. These options use an asynchronous operation and update all references from the custom domain name like "user@contoso.com" to the initial default domain name such as "user@contoso.onmicrosoft.com."

To call **ForceDelete** in the Azure portal, you must ensure that there are fewer than 1000 references to the domain name, and any references where Exchange is the provisioning service must be updated or removed in the Exchange Admin Center. This includes Exchange Mail-Enabled Security Groups and distributed lists; for more information, see Removing mail-enabled security groups. Also, the **ForceDelete** operation won't succeed if either of the following is true:

- You purchased a domain via Office 365 domain subscription services
- You are a partner administering on behalf of another customer tenant

The following actions are performed as part of the **ForceDelete** operation:

- Renames the UPN, EmailAddress, and ProxyAddress of users with references to the custom domain name to the initial default domain name.
- Renames the EmailAddress of groups with references to the custom domain name to the initial default domain name.
- Renames the identifierUris of applications with references to the custom domain name to the initial default domain name.

An error is returned when:

- The number of objects to be renamed is greater than 1000
- One of the applications to be renamed is a multi-tenant app

Frequently asked questions

Q: Why is the domain deletion failing with an error that states that I have Exchange mastered groups on this domain name?

A: Today, certain groups like Mail-Enabled Security groups and distributed lists are provisioned by Exchange and need to be manually cleaned up in Exchange Admin Center (EAC). There may be lingering ProxyAddresses which rely on the custom domain name and will need to be updated manually to another domain name.

Q: I am logged in as admin@contoso.com but I cannot delete the domain name "contoso.com"?

A: You cannot reference the custom domain name you are trying to delete in your user account name. Ensure that the Global Administrator account is using the initial default domain name (.onmicrosoft.com) such as admin@contoso.onmicrosoft.com. Sign in with a different Global Administrator account that such as admin@contoso.onmicrosoft.com or

another custom domain name like “fabrikam.com” where the account is admin@fabrikam.com.

Q: I clicked the Delete domain button and see In Progress status for the Delete operation. How long does it take? What happens if it fails?

A: The delete domain operation is an asynchronous background task that renames all references to the domain name. It should complete within a minute or two. If domain deletion fails, ensure that you don’t have:

- Apps configured on the domain name with the appIdentifierURI
- Any mail-enabled group referencing the custom domain name
- More than 1000 references to the domain name

If you find that any of the conditions haven’t been met, manually clean up the references and try to delete the domain again.

Use PowerShell or Graph API to manage domain names

Most management tasks for domain names in Azure Active Directory can also be completed using Microsoft PowerShell, or programmatically using Azure AD Graph API.

- Using PowerShell to manage domain names in Azure AD
- Using Graph API to manage domain names in Azure AD

Next steps

- Add custom domain names
- Remove Exchange mail-enabled security groups in Exchange Admin Center on a custom domain name in Azure AD
- ForceDelete a custom domain name with Microsoft Graph API

q) use of active directory ad connect?

Azure **Active Directory** (Azure **AD**) **Connect** Health provides robust monitoring of your on-premises identity infrastructure. It enables you to maintain a reliable connection to Office 365 and **Microsoft** Online Services.Feb 25, 2019

What is Azure AD Connect?

- 02/26/2019
- 3 minutes to read
-
  -
  -
  -

Azure AD Connect is the Microsoft tool designed to meet and accomplish your hybrid identity goals. It provides the following features:

- Password hash synchronization - A sign-in method that synchronizes a hash of a users on-premises AD password with Azure AD.
- Pass-through authentication - A sign-in method that allows users to use the same password on-premises and in the cloud, but doesn't require the additional infrastructure of a federated environment.
- Federation integration - Federation is an optional part of Azure AD Connect and can be used to configure a hybrid environment using an on-premises AD FS infrastructure. It also provides AD FS management capabilities such as certificate renewal and additional AD FS server deployments.
- Synchronization - Responsible for creating users, groups, and other objects. As well as, making sure identity information for your on-premises users and groups is matching the cloud. This synchronization also includes password hashes.
- Health Monitoring - Azure AD Connect Health can provide robust monitoring and provide a central location in the Azure portal to view this activity.

What is Azure AD Connect Health?

Azure Active Directory (Azure AD) Connect Health provides robust monitoring of your on-premises identity infrastructure. It enables you to maintain a reliable connection to Office 365 and Microsoft Online Services. This reliability is achieved by providing monitoring capabilities for your key identity components. Also, it makes the key data points about these components easily accessible.

The information is presented in the Azure AD Connect Health portal. Use the Azure AD Connect Health portal to view alerts, performance monitoring, usage analytics, and other information. Azure AD Connect Health enables the single lens of health for your key identity components in one place.

Why use Azure AD Connect?

Integrating your on-premises directories with Azure AD makes your users more productive by providing a common identity for accessing both cloud and on-premises resources. Users and organizations can take advantage of:

- Users can use a single identity to access on-premises applications and cloud services such as Office 365.
- Single tool to provide an easy deployment experience for synchronization and sign-in.
- Provides the newest capabilities for your scenarios. Azure AD Connect replaces older versions of identity integration tools such as DirSync and Azure AD Sync. For more information, see Hybrid Identity directory integration tools comparison.

Why use Azure AD Connect Health?

When with Azure AD, your users are more productive because there's a common identity to access both cloud and on-premises resources. Ensuring the environment is reliable, so that users can access these resources, becomes a challenge. Azure AD Connect Health helps

monitor and gain insights into your on-premises identity infrastructure thus ensuring the reliability of this environment. It is as simple as installing an agent on each of your on-premises identity servers.

Azure AD Connect Health for AD FS supports AD FS 2.0 on Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2 and Windows Server 2016. It also supports monitoring the AD FS proxy or web application proxy servers that provide authentication support for extranet access. With an easy and quick installation of the Health Agent, Azure AD Connect Health for AD FS provides you a set of key capabilities.

Key benefits and best practices:

Key Benefits	**Best Practices**
Enhanced security	Extranet lockout trends Failed sign-ins report In privacy compliant
Get alerted on all critical ADFS system issues	Server configuration and availability Performance and connectivity Regular maintenance
Easy to deploy and manage	Quick agent installation Agent auto upgrade to the latest Data available in portal within minutes
Rich usage metrics	Top applications usage Network locations and TCP connection Token requests per server
Great user experience	Dashboard fashion from Azure portal Alerts through emails

License requirements for using Azure AD Connect

Using this feature is free and included in your Azure subscription.

Next steps

- Hardware and prerequisites
- Express settings
- Customized settings
- Install Azure AD Connect Health agents

Feedback

q) what is cdn IN AZURE?

Azure Content Delivery Network (**CDN**) is a global **CDN** solution for delivering high-bandwidth content. ... With **Azure CDN**, you can cache static objects loaded from **Azure** Blob storage, a web application, or any publicly accessible web server, by using the closest point of presence (POP) server.

q) disks vs snapshots?

---

q) What is the difference between **public and private IP** address?

A **public IP** address is an **IP** address that can be accessed over the Internet. ... **Private IP** address, on the other hand, is used to assign computers within your **private** space without letting them directly expose to the Internet

q) Azure Container Registry?

Azure Container Registry is a managed, private **Docker registry** service based on the open-source **Docker Registry** 2.0. ... Use **Azure container registries** with your existing **container** development and deployment pipelines, or use **Azure Container Registry** Tasks to build **container** images in **Azure**

q)Cosmos DB?

Azure Cosmos DB is Microsoft's proprietary globally-distributed, multi-model database service "for managing data at planet-scale" launched in May 2017. It is schema-agnostic, horizontally scalable and generally classified as a NoSQL database

Welcome to Azure Cosmos DB

Today's applications are required to be highly responsive and always online. To achieve low latency and high availability, instances of these applications need to be deployed in datacenters that are close to their users. Applications need to respond in real time to large changes in usage at peak hours, store ever increasing volumes of data, and make this data available to users in milliseconds.

Azure Cosmos DB is Microsoft's globally distributed, multi-model database service. With a click of a button, Cosmos DB enables you to elastically and independently scale throughput and storage across any number of Azure regions worldwide. You can elastically scale throughput and storage, and take advantage of fast, single-digit-millisecond data access

using your favorite API including SQL, MongoDB, Cassandra, Tables, or Gremlin. Cosmos DB provides comprehensive service level agreements (SLAs) for throughput, latency, availability, and consistency guarantees, something no other database service offers.

You can Try Azure Cosmos DB for Free without an Azure subscription, free of charge and commitments.

Try Azure Cosmos DB for Free

You can also use the Cosmos DB Bootstrap Program to accelerate building or migrating your applications on Azure Cosmos DB. When you sign up for this program, the Azure Cosmos DB engineers are assigned to assist with your project and they can help you migrate your data to Azure Cosmos DB or building new apps on Azure Cosmos DB.

Sign up for the Cosmos DB bootstrap program

Key Benefits

Turnkey global distribution

Cosmos DB enables you to build highly responsive and highly available applications worldwide. Cosmos DB transparently replicates your data wherever your users are, so your users can interact with a replica of the data that is closest to them.

Cosmos DB allows you to add or remove any of the Azure regions to your Cosmos account at any time, with a click of a button. Cosmos DB will seamlessly replicate your data to all the regions associated with your Cosmos account while your application continues to be highly available, thanks to the *multi-homing* capabilities of the service. For more information, see the global distribution article.

Always On

By virtue of deep integration with Azure infrastructure and transparent multi-master replication, Cosmos DB provides 99.999% high availability for both reads and writes. Cosmos DB also provides you with the ability to programmatically (or via Portal) invoke the regional failover of your Cosmos account. This capability helps ensure that your application is designed to failover in the case of regional disaster.

Elastic scalability of throughput and storage, worldwide

Designed with transparent horizontal partitioning and multi-master replication, Cosmos DB offers unprecedented elastic scalability for your writes and reads, all around the globe. You can elastically scale up from thousands to hundreds of millions of requests/sec around the globe, with a single API call and pay only for the throughput (and storage) you need. This capability helps you to deal with unexpected spikes in your workloads without having to over-provision for the peak. For more information, see partitioning in Cosmos DB, provisioned throughput on containers and databases, and scaling provisioned throughput globally.

Guaranteed low latency at 99th percentile, worldwide

Using Cosmos DB, you can build highly responsive, planet scale applications. With its novel multi-master replication protocol and latch-free and write-optimized database engine,

Cosmos DB guarantees less than 10-ms latencies for both, reads (indexed) and writes at the 99th percentile, all around the world. This capability enables sustained ingestion of data and blazing-fast queries for highly responsive apps.

Precisely defined, multiple consistency choices

When building globally distributed applications in Cosmos DB, you no longer have to make extreme tradeoffs between consistency, availability, latency, and throughput. Cosmos DB's multi-master replication protocol is carefully designed to offer five well-defined consistency choices - *strong*, *bounded staleness*, *session*, *consistent prefix*, and *eventual* — for an intuitive programming model with low latency and high availability for your globally distributed application.

No schema or index management

Keeping database schema and indexes in-sync with an application's schema is especially painful for globally distributed apps. With Cosmos DB, you do not need to deal with schema or index management. The database engine is fully schema-agnostic. Since no schema and index management is required, you also don't have to worry about application downtime while migrating schemas. Cosmos DB automatically indexes all data and serves queries fast.

Battle tested database service

Cosmos DB is a foundational service in Azure. For nearly a decade, Cosmos DB has been used by many of Microsoft's products for mission critical applications at global scale, including Skype, Xbox, Office 365, Azure, and many others. Today, Cosmos DB is one of the fastest growing services on Azure, used by many external customers and mission-critical applications that require elastic scale, turnkey global distribution, multi-master replication for low latency and high availability of both reads and writes.

Ubiquitous regional presence

Cosmos DB is available in all Azure regions worldwide, including 54+ regions in public cloud, Azure China 21Vianet, Azure Germany, Azure Government, and Azure Government for Department of Defense (DoD). See Cosmos DB's regional presence.

Secure by default and enterprise ready

Cosmos DB is certified for a wide array of compliance standards. Additionally, all data in Cosmos DB is encrypted at rest and in motion. Cosmos DB provides row level authorization and adheres to strict security standards.

Significant TCO savings

Since Cosmos DB is a fully managed service, you no longer need to manage and operate complex multi datacenter deployments and upgrades of your database software, pay for the support, licensing, or operations or have to provision your database for the peak workload. For more information, see Optimize cost with Cosmos DB.

Industry leading comprehensive SLAs

Cosmos DB is the first and only service to offer industry-leading comprehensive SLAs encompassing 99.999% high availability, read and write latency at the 99th percentile, guaranteed throughput, and consistency.

Globally distributed operational analytics and AI with natively built-in Apache Spark

You can run Spark directly on data stored in Cosmos DB. This capability allows you to do low-latency, operational analytics at global scale without impacting transactional workloads operating directly against Cosmos DB. For more information, see Globally distributed operational analytics.

Develop applications on Cosmos DB using popular Open Source Software (OSS) APIs

Cosmos DB offers a choice of APIs to work with your data stored in your Cosmos database. By default, you can use SQL (a core API) for querying your Cosmos database. Cosmos DB also implements APIs for Cassandra, MongoDB, Gremlin and Azure Table Storage. You can point client drivers (and tools) for the commonly used NoSQL (e.g., MongoDB, Cassandra, Gremlin) directly to your Cosmos database. By supporting the wire protocols of commonly used NoSQL APIs, Cosmos DB allows you to:

- Easily migrate your application to Cosmos DB while preserving significant portions of your application logic.
- Keep your application portable and continue to remain cloud vendor-agnostic.
- Get a fully-managed cloud service with industry leading, financially backed SLAs for the common NoSQL APIs.
- Elastically scale the provisioned throughput and storage for your databases based on your need and pay only for the throughput and storage you need. This leads to significant cost savings.

Solutions that benefit from Azure Cosmos DB

Any web, mobile, gaming, and IoT application that needs to handle massive amounts of data, reads, and writes at a global scale with near-real response times for a variety of data will benefit from Cosmos DB's guaranteed high availability, high throughput, low latency, and tunable consistency. Learn about how Azure Cosmos DB can be used to build IoT and telematics, retail and marketing, gaming and web and mobile applications.

What is a project in Azure DevOps?

A **project** provides a repository for source code and a place for a group of people to plan, track progress, and collaborate on building software solutions. It represents a fundamental container where data is stored when added to **Azure DevOps**.