IT Innovation Series:   
ATTENDEE GUIDE

“What’s new in Azure Infrastructure: Improving Datacenter Flexibility with Microsoft, Open Source and Other Technologies” rev3.2

**Table of Contents**

[**Attendee guide** 1](#_Toc441831072)

[How to use this guide 1](#_Toc441831073)

[What this guide is for 2](#_Toc441831074)

[Today’s goals 2](#_Toc441831075)

[Event format 2](#_Toc441831076)

[Agenda 2](#_Toc441831077)

[Today: Step-by-step guidance 2](#_Toc441831078)

[Lab 1: Deploy Azure Resources using ARM Templates 10](#_Toc441831079)

[Exercise 1: Prepare the Azure Infrastructure 11](#_Toc441831080)

[Exercise 2: Understand Azure Templates and Resource Groups 12](#_Toc441831081)

[Exercise 3: Configure Git repository and Visual Studio Code 18](#_Toc441831082)

[Exercise 4: Modify and Deploy Azure Quickstart Templates 25](#_Toc441831083)

[Exercise 5: Remove resource group used for lab 33](#_Toc441831084)

[Lab 2: Design Azure Storage and Compute Infrastructure 34](#_Toc441831085)

[Exercise 1: Prepare the Azure Infrastructure 34](#_Toc441831086)

[Exercise 3: Compare Standard and Premium Storage 39](#_Toc441831087)

[Lab 3: Design Azure Networking for Advanced Security 42](#_Toc441831088)

[Exercise 1: Prepare the Azure Infrastructure 43](#_Toc441831089)

[Exercise 2: Run the Lab03Start.ps1 Script 44](#_Toc441831090)

[Lab 4: Designing Identity Solutions with Azure Active Directory 64](#_Toc441831091)

[Exercise 1: Installing and Configuring Azure Active Directory Connect 65](#_Toc441831092)

[Exercise 2: Confirm Azure Active Directory Connect Synchronization 68](#_Toc441831093)

[Exercise 3: Establishing SSO with a Third-Party Provider and Azure Active Directory 69](#_Toc441831094)

[Exercise 4: Cloud App Discovery 72](#_Toc441831095)

[**Worksheet Exercise: Designing for Appropriate Workloads with Questions and Answers** 75](#_Toc441831096)

[At home: Resources to continue learning 81](#_Toc441831097)

[Cloud Infrastructure 81](#_Toc441831098)

[TechNet on Tour Resources – Jan-Mar 2016 81](#_Toc441831099)

[FREE Online Training: Microsoft Virtual Academy 83](#_Toc441831100)

# Attendee guide

## How to use this guide

1. **Use this guide only**, disregard the online .PDF lab guide files.
2. Review [Attendee guide](#_Overview) to understand event goals, format and agenda
3. Today: Follow the [Step-by-step guidance](#_Step-by-step_Guidance) to complete the technical exercises
4. At home: Review [Next Steps](#_At_home:_Resources) for continued learning.
5. Online Labs are valid for 30-days.

## What this guide is for

Thank you for joining us for today’s IT Innovation Series event, focused on “What’s new in Cloud Infrastructure: Improving Datacenter Flexibility with Microsoft, Open Source and Other Technologies”. This document will guide you through today’s instructor-led and individual activities, and provide you with the key resources and information necessary to continue building your skills to be able to implement in your environment.

## Today’s goals

By the time you leave today, you should walk away with:

1. A deep technical understanding of the latest Azure Premium Services and Advanced performance configurations
2. The knowledge to integrate on-premises solutions to Azure, like Disaster Recovery and Virtualization
3. An understanding of the best practices around compute, storage, network and Identity architectures

## Event format

Today’s event will include a combination of presentation, short activities and hands-on work, organized in a step-by-step approach to lead you through overview to implementation. You will use your personal laptop and a free Azure Pass we’ll provide to complete today’s activities directly in Azure.

**Your instructor will provide the instructions to access the lab environment.**

**Hands on Lab URL is:** [**http://aka.ms/iti**](http://aka.ms/iti)

**Training Key is: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## Agenda

|  |  |
| --- | --- |
| 9:00 – 9:30 | **Introduction - Datacenter and IT role changes in a cloud first world** |
| 9:30 – 10:30 | **Deploy Azure Resources using ARM Templates** |
| 10:30 – 12:00 | **Designing Azure Networking Infrastructure for advanced security** |
| 12:00PM | Lunch |
| 12:45 – 1:45 | **Designing Azure Networking Infrastructure for advanced security** |
| 1:45 – 2:30 | **Designing identity solutions with Active Directory** |
| 2:30 – 2:45 | Wrap up / Evaluations / Raffle |
| 2:45 – 4:30 | Hands-on labs with Proctor |

# Today: Step-by-step guidance

Following the introduction and overview, you’ll have the chance to try out real-world scenarios directly in our preconfigured labs, aimed to help give you the skills necessary to deploy in your own environment.

#### Before You Begin

The portal is located at [https://portal.azure.com](https://portal.azure.com/).

#### Azure Subscriptions

This IT Innovation Series event lab requires a valid Azure subscription. While you may use an existing subscription such as a subscription associated MSDN account or existing corporate account, it would preferable to use an Azure Trial subscription for this event. By using a trial subscription, you will avoid any charges against your MSDN or corporate subscription that would result from doing the activities in this camp.

Your instructor may be able to provide you with a pre-provisioned Microsoft Account that already has an Azure subscription associated with it. Or, you may use a CLEAN and UNUSED Azure Trial account - details on how to set one up are detailed below.

| **Description** | **Steps** |
| --- | --- |
| Creating a new Azure trial account perform the following steps.  Valid for $200 in Azure credits and 30-days evaluation time  Configuring an Azure Pass perform the following steps.  Valid for $100 in Azure credits and 30 days evaluation time | **Creating a new Azure 30-day Trial Account**   1. Open an InPrivate window in your browser 2. Navigate to [www.live.com](http://www.live.com) and click **Sign up now**. 3. Follow the on-screen instructions to create a new Microsoft Account. 4. Navigate to [www.azure.com](http://www.azure.com) and click **Free Trial**.   Note: Below are the Free Trial details – If you are using an Azure Pass get the pass from the Instructor.   1. Follow the on-screen instructions to activate a new Windows Azure Trial. 2. Navigate to Manage.windowsazure.com and sign in. 3. In Microsoft Azure portal, in the upper left, click your user name, and then click **View my bill**. 4. Click your current trial subscription, and then click **Edit subscription details.** 5. Type a name you will recognize in SUBSCRIPTION NAME, such as ITCamps, and then click the **Done** icon.   **Configuring an Azure Pass – (Provided by instructor if available)**  Your instructor may be able to provide you with a pre-provisioned Microsoft Account that already has an Azure Pass subscription associated with it.  Alternatively, your instructor may be able to provide you with an Azure Pass promotional code.  To activate the promotional code and create a new Azure Pass account perform the following steps.   1. If you are not using the lab virtual machine to activate your Azure Pass promotional code, ensure you open an InPrivate browser session before performing these steps. It is critically important that you do not accidently associate the promotional code with any account that has previously been associated with or linked to an Azure subscription. Use an InPrivate browser session to ensure that no credentials are unintentionally forwarded during the process to activate and redeem the promotional code. If you fail to activate the code because you logged in with the wrong account, you will render the code useless and will not be able to use it again. 2. Navigate to **www.live.com** and click **Sign up now**.. Follow the on-screen instructions to create a new Microsoft Account. Please ensure, you create an outlook.com, live.com or Hotmail.com account. Do not use accounts that have country code suffixes, such as .dk, ca, uk, etc. in their names. 3. Navigate to **http://www.microsoftazurepass.com** and follow the onscreen instructions to redeem the Azure Pass promotional code. (Note - You may be getting this code from your instructor). Once you have submitted the promotional code, it will take a few minutes for the account to become activated. Only one promo code can be redeemed per the life of the Microsoft ID. Click **Purchase**, even though you will not be charged for this pass. 4. Follow the on-screen instructions to activate a new Azure Pass. 5. Click on **Start managing my service** and enter the Azure Portal |

#### Logging into Hosted Workstations

Labs in this camp are written to be completed on a pre-configured workstation. Additional labs require an on-premises environment consisting of multiple servers. A hosted virtual machine environment is provided for this purpose. All you will need is the required software below to run the labs.



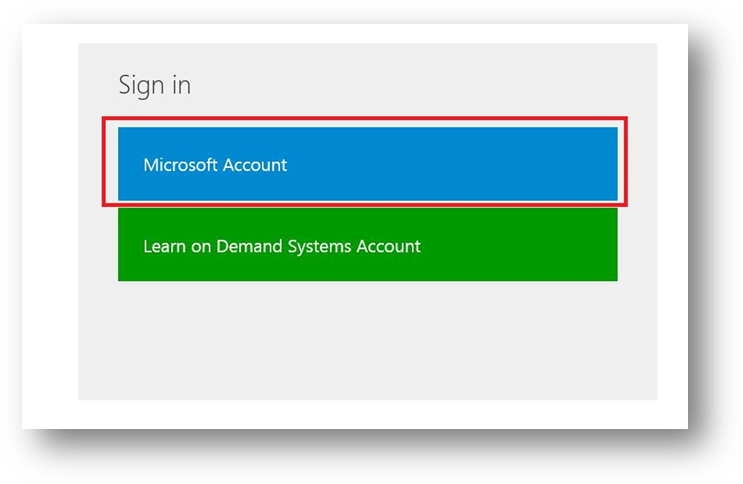
**How to Access Labs with a Training Key**

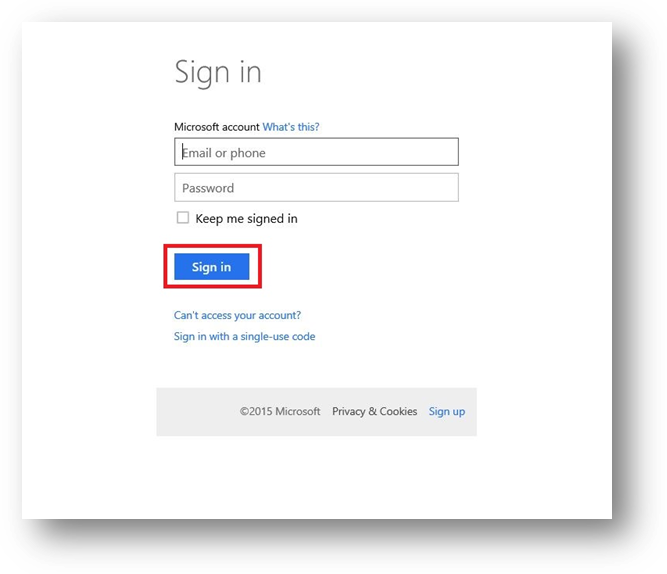
Training Key will be provided by Instructor.

To access labs, navigate to: [**https://aka.ms/iti**](https://aka.ms/iti) You will then enter in your Training Key.

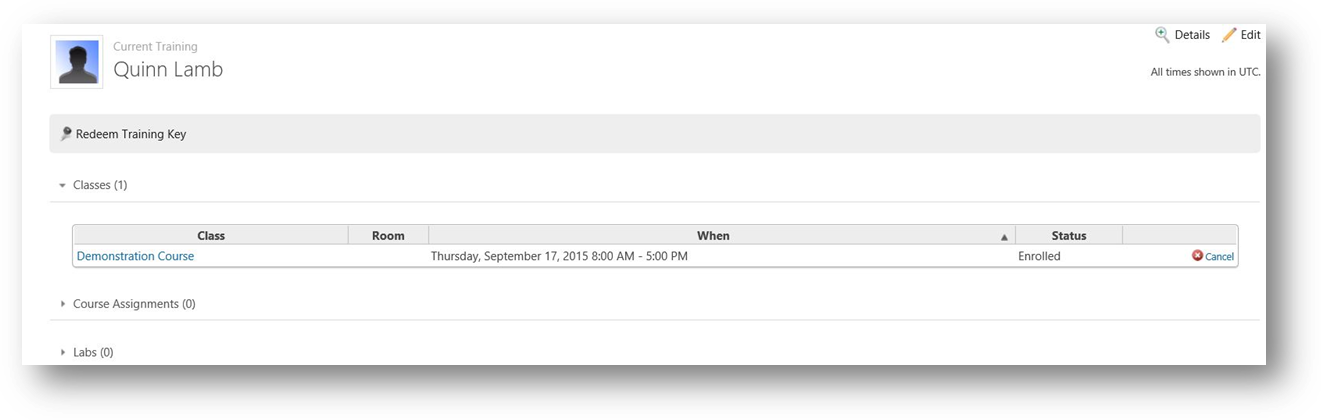
On the portal landing page, you will be presented with two options to sign in. Please select the option to sign in with a **Microsoft Account (Live ID**). If you do not have a Microsoft Account, is unable to use one, or already has a Learn on Demand Systems account, then please refer to the Learn on Demand Systems Account Creation section towards the bottom of this guide.

**You will need to enter your Microsoft Account credentials**, then click the **blue Sign in** button.

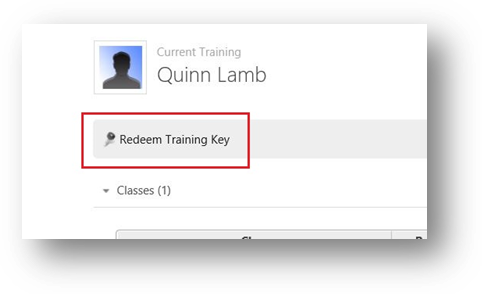




After you have successfully signed in you will be presented with a page similar to the one below. If you have already accessed labs through this system, you will see a list of all of the previous classes they have enrolled in.

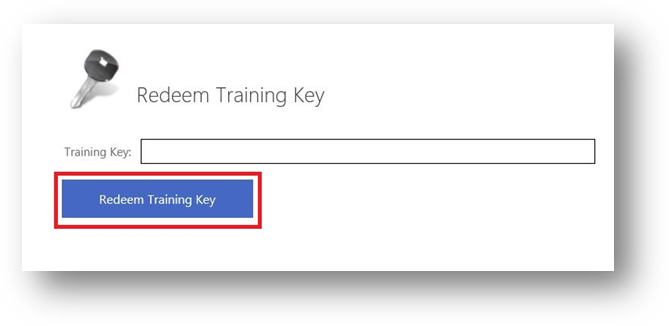


Click the **Redeem Training Key** link at the top of the page.



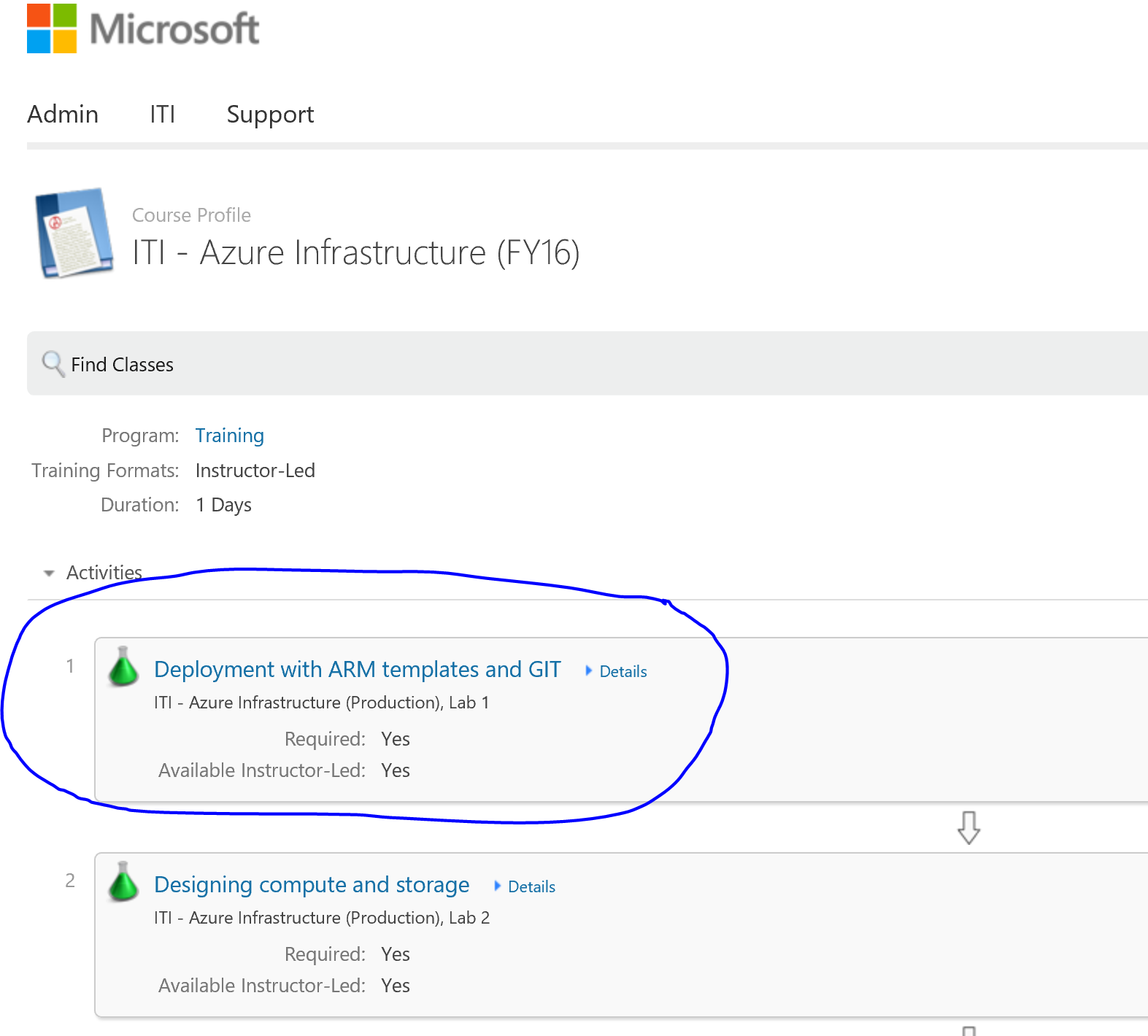
**You will need to enter the Training Key**, and then click the **Redeem Training Key** button. You will only need to redeem the Training Key once for this class. For the duration of the class, you will be able to sign back in to the portal and select this class from their enrollment list and access labs (for the allotted time).

**Labs are valid for 30 days.**

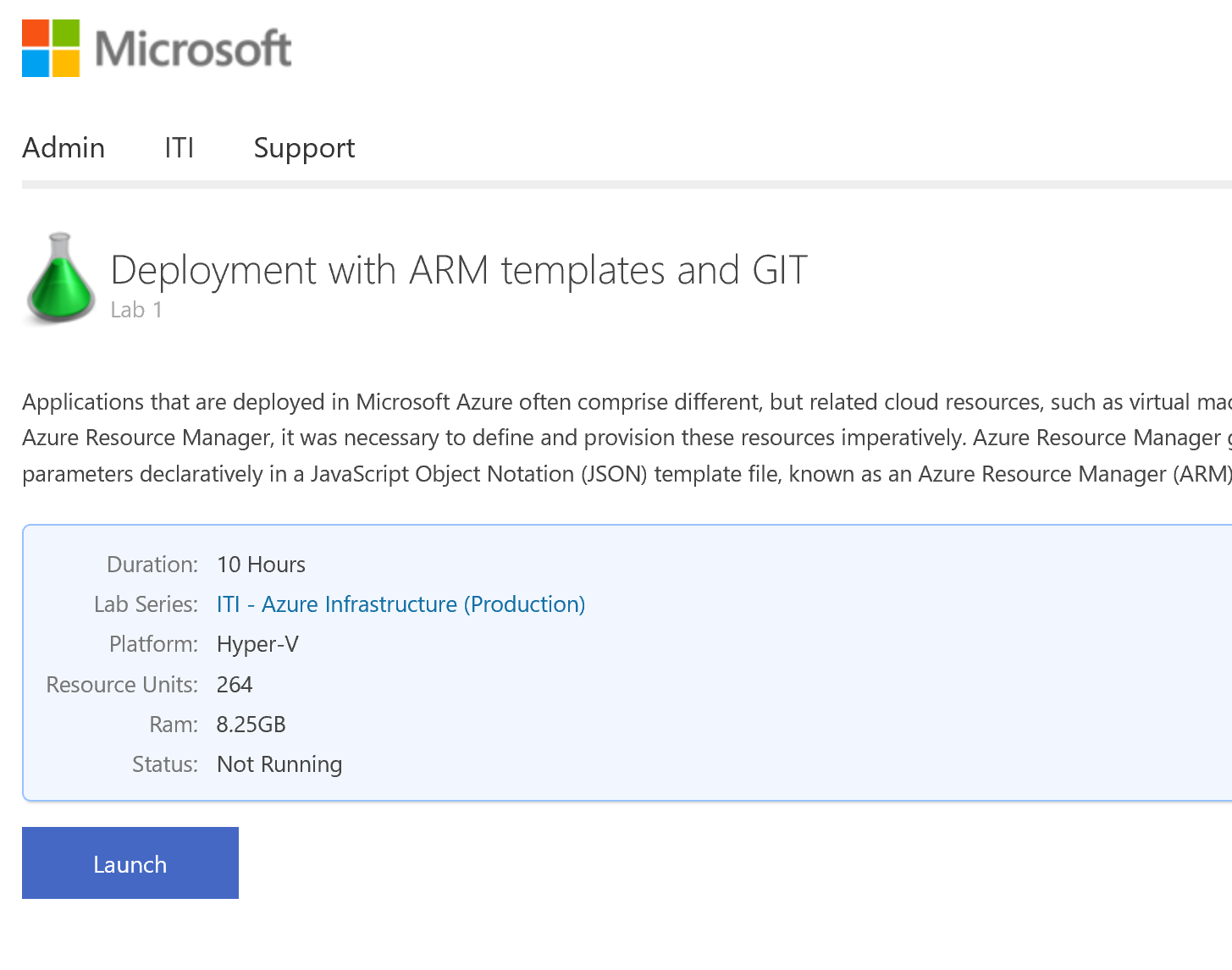


Once you have redeemed the Training Key you will be taken to the class page. Here you will see a link to launch a test lab, links to launch the course labs (under Activities), and any downloadable content for the class.

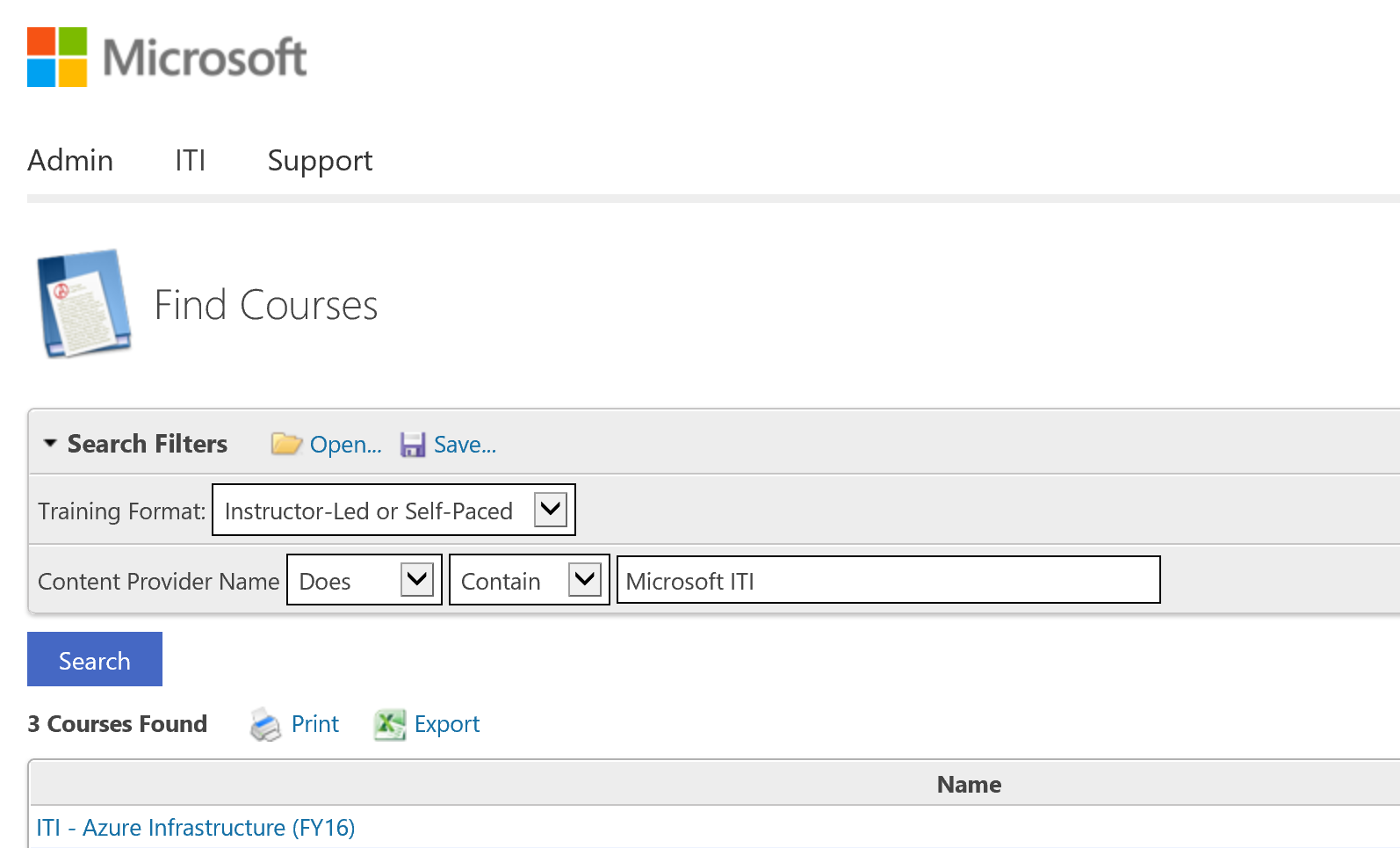
**Do not click on the test lab!**



To launch a lab, simply find the desired lab under the **Activities** section, and then click the blue Launch button. Once the lab is launched, it will take a few minutes to load. Please be patient and wait. Do not use the online .PDF files, use the guide that is provided by the Instructor.

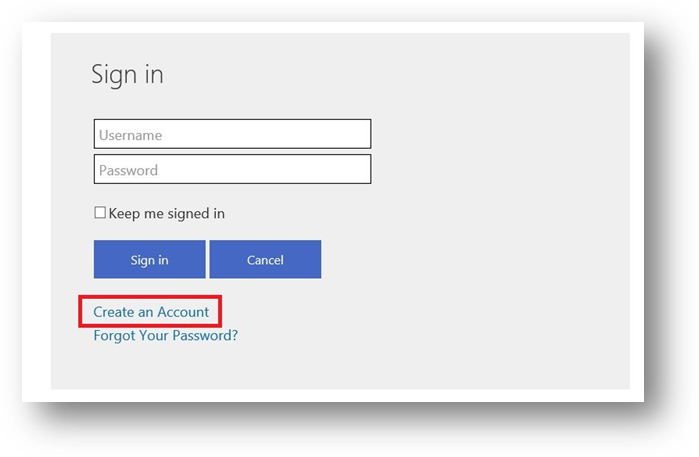


**Note:** If you’ve already accessed labs for this class, you will need to use the same sign in method and credentials as you did when you initially registered for the class. Once signed in, you can select the desired class from the displayed list where they will then be taken to the class page.



**Learn on Demand Systems Account Creation**

If you do not have a **Microsoft Account** or do not wish to use one, you can select the Learn on Demand Systems Account option on the portal landing page. After clicking the green Learn on Demand Systems Account button, you can create an account in the system, or use an existing account if they already have one. To enter the system with existing credentials, you will simply need to enter their Username and Password.



You will need to fill out the required information pictured below, and then click the blue Save button. You will then be taken to the class listing page where you can follow the prompts listed above to **Redeem Training Key**.



***(Optional) Using your own local machine instead of the Hosted Environment***

If using the hosted lab, you may complete lab instructions using your own workstation instead of the online lab environment (with either Windows 10 or Windows 8.1), providing you download the files used for the lab from GitHub and have the following software installed.

#### GitHub repository for Lab Files if using your own machine

If you are not using the hosted virtual machine and are using your own workstation, any custom files the lab instruction call out can be found in a GitHub repository. The repository is located here: <https://github.com/AZITCAMP/Labfiles>.

#### Required Software – Note - you can click on the links in the document located here for the software below, <http://aka.ms/azure-software-tools>

#### Required Software

| **Description** | **Steps** |
| --- | --- |
| Required software will be called out throughout the lab. | 1. Microsoft Azure PowerShell - <http://go.microsoft.com/?linkid=9811175&clcid=0x409> (also installs the Web Platform Installer, minimum version 0.9.8 and higher) 2. Visual Studio Code - <https://code.visualstudio.com/> 3. Install Git at: <http://git-scm.com/download/win> 4. GitHub Desktop for Windows - <https://desktop.github.com/> 5. Windows Credential Store for Git (if VSCode won't authenticate with GitHub) - [http://gitcredentialstore.codeplex.com/](http://gitcredentialstore.codeplex.com/%20) 6. Iometer - <http://sourceforge.net/projects/iometer/> |

#### Optional Software

| **Description** | **Software** |
| --- | --- |
| Any additional software that you require will be called out in the lab. The following software may be useful when working with Azure in general. | 1. Remote Server Administration Tools - [http://support.microsoft.com/kb/2693643](http://support.microsoft.com/kb/2693643%20) (Windows 8.1) or [http://www.microsoft.com/en-ca/download/details.aspx?id=45520](http://www.microsoft.com/en-ca/download/details.aspx?id=45520%20) (Windows 10) 2. AzCopy - <http://aka.ms/downloadazcopy> (in Azure Tools) 3. Azure Storage Explorer - <http://azurestorageexplorer.codeplex.com/downloads/get/891668> 4. Microsoft Azure Cross-platform Command Line Tools (installed using the Web Platform Installer) 5. Visual Studio Community 2015 with Microsoft Azure SDK - 2.8.1 (installed using the Web Platform Installer) 6. Msysgit - <http://msysgit.github.io> 7. PuTTY and PuTTYgen – (Use the Windows Installer) <http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html> 8. Microsoft Online Services Sign-In Assistant for IT Professionals RTW - <http://go.microsoft.com/fwlink/?LinkID=286152> 9. Azure Active Directory Module for Windows PowerShell (64-bit version) - <http://go.microsoft.com/fwlink/p/?linkid=236297> |

Lab 1: Deploy Azure Resources using ARM Templates

|  |
| --- |
| Applications that are deployed in Microsoft Azure often comprise different, but related cloud resources, such as virtual machines, web applications, SQL databases, virtual networks, and others. Before the introduction of Azure Resource Manager, it was necessary to define and provision these resources imperatively. Azure Resource Manager gives you the ability to define and provision these resources with their configuration and associated parameters declaratively in a JavaScript Object Notation (JSON) template file, known as an Azure Resource Manager (ARM) template.  In this lab, you will learn how to create and deploy IaaS applications using ARM templates. |

#### Overview

Contoso, Inc. has successfully been using Azure for some time now. However, Contoso's use of Azure has, up until now, been limited to the use of the Azure Service Management API. The Service Management API allows Contoso to manage Azure resources, such as storage accounts, virtual machines, and virtual networks as individual entities. The Service Management API model serves Contoso well, but it does not reflect the fact that these entities are often, in fact, related and independent parts of a larger, single entity.

In contrast to the Azure Service Management API, the Azure Resource Management API allows organizations such as Contoso to deploy, manage, and monitor Azure resources as a single group. In addition to this benefit, ARM makes it possible to use declarative templates to define and provision resources in a deployment. Furthermore, Contoso could implement Role-Based Access Controls (RBAC) to provide greater security. Using ARM, Contoso would be able to apply tags to resources to organize them logically within the same subscription.

A significant and additional benefit is that by using Azure Resource Management templates, Contoso can start to realize an important goal of bringing the best of DevOps practices into its infrastructure by treating its infrastructure as code. Following this paradigm, Contoso will be able to modularize its infrastructure into various components and then join or separate the constituent components as necessary in a highly automatable and scalable manner.

Because of the benefits that Azure Resource Management can provide to Contoso, you have been asked to explore how ARM can be used to deploy and manage resources. Specifically, you have been asked to learn how templates can be used to define and deploy Azure Resources.

#### Virtual Machines

For this lab you will be accessing a hosted environment that contains all the VMs and resources you require. Your instructor will provide a link to the hosted lab environment.

You should be able to connect with any recent web browser, including Microsoft Edge. Once you have connected to the lab environment, take a few minutes to familiarize yourself with Launchpad.

For this course there are four VMs that you will work in. If you look at the Machines tab on the right side of the lab environment you will find a listing of all the VMs. To switch to another VM, just click on the appropriate name in the Machines list. Below you will find a listing of the VMs for this course.

| **Virtual machine** | **Description** |
| --- | --- |
| **AZRCamp-Admin** | Windows 10. A member of the Contoso.com domain. Used for Azure management. |
| **AZRCamp-Edge** | A Stand-alone Windows Server 2012 R2 Server. Routing and Remote Access has been installed and it is acting as the default gateway for all outbound traffic. |
| **AZRCamp-DC** | Windows Server 2012 R2 domain controller and DNS server. |
| **AZRCamp-Sync** | Directory Sync for use in other Labs. |

The password for all logons in these VMs is **Passw0rd!**

**Note:** If you are having configuration or syntax issues with your JSON files not working, you can always go to the **Solutions** folder for the properly configured JSON lab files.

**Exercise 1: Prepare the Azure Infrastructure**

In this exercise, you will use the **Lab01Start.ps1** script to log on to your Azure subscription and create an Azure resource group that you will use for the remaining lab activities. The script will also determine a globally unique name that you can use to create a storage account in subsequent lab activities.

***Activity 1: Run the Lab01Start.ps1 script***

| **Description** | **Steps** |
| --- | --- |
| To perform the subsequent lab activities, you need to **create an Azure Resource Group** and determine a globally unique name you can use to **create a storage account.** | The lab will open to the desktop, click on **Machines,** you can click on the push pin to hide vm’s and stay on the desktop.  ***Note*** – Please only use the lab guide handed out at this event for the lab instructions.  Perform the following activities on **AZRCAMP-ADMIN**:  **Contoso\Administrator** using **Passw0rd!** as the password:   1. Open File Explorer from taskbar and you will see folders to chose from. Select and navigate to **C:\LabFiles\AZRITPROCamp\Lab01-Deploy Azure resources using templates**. 2. Right-click **Lab01Start.ps1**, and click **Edit**.   The Windows PowerShell ISE console opens.   1. In Windows PowerShell ISE, on the upper Ribbon, click on the word **SYNOPSIS** go to **Edit** and **Select All**, and click **Run Selection** 2. When prompted, enter a lower-case string that represents your initials, and press ENTER.   The storage account name must contain only lower case letters and numbers and must be globally unique.   1. In the **Sign in to Windows Azure PowerShell** dialog box, enter the **email address** of the account associated with your Azure subscription, and click **Continue**. 2. On the sign in page, enter your password, and click **Sign in**.   The script starts running and then pauses to display a name verified as unique for use a storage account name. (Example: bbstore1).  **Storage Account Name**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  **Note:** Android users may need to download the Microsoft Account app to get the authentication to activate Azure. For iphone users, use the Google Authenticator App from istore.   1. Record the unique storage account name, and press ENTER to continue.   You will need to know the unique name for the storage account in a subsequent lab activity.  ***Troubleshooting tip on Subscriptions*** - If you have error on your subscription name, try using a different account and add the name into line 50 of the PowerShell script. Replace $subname with your subscription name. If the subscription name has spacing in it, rename the account in azure without spaces and run the script again with the new name.  This script creates a **Resource Group** called **RG-AZITCAMP-LAB01**  **DO NOT CLOSE POWERSHELL**  Leave the Windows PowerShell ISE console open for subsequent lab activities. |

**Exercise 2: Understand Azure Templates and Resource Groups**

Applications that are deployed in Microsoft Azure often comprise different, but related cloud resources, such as virtual machines, web applications, SQL databases, virtual networks, and others. Before the introduction of Azure Resource Manager, it was necessary to define and provision these resources imperatively, for example, with Azure PowerShell cmdlets such as Add-AzureVMDataDisk. Azure Resource Manager gives you the ability to define and provision these resources with their configuration and associated parameters declaratively in a JavaScript Object Notation (JSON) template file. The format for template files is described in detail here: <https://msdn.microsoft.com/en-us/library/azure/dn790564.aspx>.

A JSON template file is a text file that contains descriptions of the resources, configurations code and extensions. JSON templates are can be run multiple times without changing the outcome beyond initial deployment. A consequence of this characteristic is that templates can be used to upgrade applications, for example, by scaling out applications with additional VMs. You modify the template to include the specifications for the additional virtual machines. When you deploy the template, Azure Resource Manager will recognize the resources that have previously been deployed and create only the resources that have been added.

Another advantage of template files is that they can be parameterized. This simplifies orchestration and deployment of applications because the parameterization allows you to reuse templates to create deployments with different configurations based on the parameters you define in the JSON template and that you specify at run time.

Once you create a JSON-based template, you can deploy the resources described in the template by using the Microsoft Azure Portal or using Windows PowerShell commands. An advantage of using Windows PowerShell commands is that you can include them in a Windows PowerShell script that provides additional automation.

When you execute the template, the resources are provisioned to a new or existing Azure resource group. An Azure resource group is a collection of related cloud resources that you can manage as a single entity. Resource groups allow you to start, stop, or delete all resources in a group at once; to apply Role-Based Access Controls rules to provide more granular security permissions on resources; to audit operations within the resource as a whole; and to enable better tracking through the use of metadata tags.

One way to think of Azure resource groups is to consider them as containers where you place logically related resources. For example, a resource group might contain an Internet-facing web site, a backend SQL database, one or more virtual networks, and a storage account for additional assets. Alternatively, you could create a separate resource group for your backend SQL database as this component is likely to have a different lifecycle than your front-end web applications. Therefore, when considering whether resources should be placed in the same or different resource group, you should also keep in mind the lifecycle of those resources, and group resources accordingly.

In this exercise, you gain a basic understanding of Azure Resource Manager templates in general by closely examining one of the **Quickstart Templates available on GitHub**.

***Activity 1: Understand Azure Quickstart templates***

JSON templates simply provide a different way of defining the same types of resources that you may be familiar with using the classic Azure model. For example, to create a new VM using the Azure Service Management mode, you would specify the image name, the virtual machine name, the size, the administrative account, the password credential, and other values. When deploying resources using JSON templates, you specify the same information, but in a different format.

It is rarely necessary to create a JSON template from scratch. Microsoft maintains a large and growing library of community contributed Azure templates on GitHub that you can leverage to learn about JSON templates and to deploy your own applications.

| Description | Steps |
| --- | --- |
| In this activity, you will view the **Quickstart Templates** that are available on GitHub and examine one of them in depth. | 1. Open **Microsoft Edge (or Internet Explorer)** and click on “Azure Quickstart” tab to navigate to <https://github.com/Azure/azure-quickstart-templates>.  * GitHub is a repository service, based on Git, that provides source control management (SCM) and revision control. GitHub expands on the functionality of the command-line based Git by providing a GUI and collaboration features.  1. On the **Azure Quickstart Templates** page, spend a few moments examining the titles and scroll down the page. 2. Templates that have a prefix of 101 are simple templates; templates that have a prefix of 201 are more advanced and complex templates.  * A searchable index of the quickstart templates is available at <https://azure.microsoft.com/en-us/documentation/templates>.  1. Scroll down the page, then locate and click on **101-vm-simple-windows**  * Note that the folder contains four files: 1) **README.md** is an HTML file that contains the description and the button that you can click to deploy the template to an Azure subscription; 2) The **azuredeploy.json** file is the main file used to deploy the cloud resources; 3) The **azuredeploy.parameters.json** is the file where you can provide values for the parameters defined in the azuredeploy.json file and pass them in at execution time; and 4) The **metadata.json** file provides additional data about the deploy template.  1. Click **azuredeploy.json**. 2. The JSON file opens as shown below:   (Tip – use your UP and DOWN arrow keys to move through the file)     * A JSON file is constructed of key/value pairs. For example, in the image above, “ContentVersion“ is the key and “1.0.0.0” is the value. A key is always a string enclosed in quotation marks. A value can be a string, securestring, number, Boolean expression, array, or object. A JSON object is enclosed in curly braces, “{ }”. In a key/value pair, the key is always followed by a colon. Key/value pairs are separated by commas. * A JSON template may also contain functions and expressions. Expressions are enclosed in square brackets, “[ ]”, and can appear anywhere in a JSON string. Function calls have the format functionName(arg1,arg2,arg3). Properties are referenced using the dot and index operators. * A JSON template may have as many as six sections: 1) “$schema” (a required element that provides the location of the file that describes the version of the template language; 2) contentVersion (a required element that provides the version of the template); 3) “parameters” (an optional element that defines values that are passed in when the template is executed); 4) “variables” (an optional element that defines the values that are used when template is executed); 5) “resources” (a required element that defines the resources that are deployed or updated in a resource group); and 6) “outputs” (an optional element that defines values that are returned after a deployment.)  1. At the top of the file, note the first line that references a JSON schema.  * The schema reference is used by intelligent JSON clients to determine the schema that is applicable to the JSON file and to provide additional functionality such as autocomplete and IntelliSense.  1. Note the Parameters key.  * The value for the Parameters key is an array of parameter objects that represent the dynamic input for the JSON template. Each of the parameter objects has a name that is used to pass values in at run time and is referenced within the JSON file itself in other sections. For example, “NewStorageAccount” is the name of the parameter that is supplied as an input and used to provide the name of the storage account resource specified in the JSON file.  1. In the “parameters” section, note the “adminPassword” parameter, and note the “type” value on line 12.   Machine generated alternative text: 16  17  18  19  21  admi nPassword'  " type "  "securestring'  "description" :  "Password for the Virtual Machine.     * Each parameter has a type, such as string, securestring, number, and so on. Securestring is a special data type that is used to ensure that the value is not persisted anywhere in the Azure platform. This ensures that sensitive information such as passwords are not displayed.  1. Scroll down to view the **WindowsOSVersion** parameter object around line 23.   Machine generated alternative text: 29  31  32  33  34  35  36  37  38  "windowsOSVe rsion "  " strin  "defaultVa1ue"  " allowedVa1ues  "2eø8-R2-SP1",  "2912 -Datacenter"  "2912-R2-Datacenter"  R2-Datacenter"  Windows version for the VM.  This w:  me a a a  "description" :  "The   * Note that a parameter can include a default value. When you execute the JSON template, you will be prompted for any parameter values that are missing from the input, unless you have specified a default value. Note also that a parameter can include an array of allowed values, which is useful if you want to constrain the input to a set of specific values—for example, specific regions. Finally, note the metadata key, where you can include a description of the parameter.  1. Scroll down to the “variables” section that begins at around line 55.   Machine generated alternative text: 42  43  44  46  47  48  49  51  52  53  54  55  56  57  58  59  "variables  "location  "Wes  " Mi crosoftWindowsServer"  "imagePub1isher"  " imageOffer "  "WindowsServer"  "OSDi skName"  " osdiskforwindowssimple"  "nicName" : "myW',lic  "addressPrefix• :  "1B.e.ø.e/16" ,  "subnetName"  "Subnet"  "subnetPrefix" .  • "le.ø.e.ø/24",  " storageAccountType "  publ icIPAddressName" :  " publ icIPAddressType "  "Standard LRS",  "myPub1icIP" ,  "Dynamic  "vmStorageAccountContainerName": "vhds "  "vmName  "vmSize  "MyWindowsVM" ,  "Standard A2" ,  "virtualNetworkName": "MyVNET" ,  "vnetID" :  "subnetRef" :  resourceld 'Microsoft . Network/virtua1Networks ' , variables 'virtualNetworkP,lame '  " C •vnetID ' ) , ' /subnets/ , variables( subnetName ' ) ) )   * Variables, in general, represent static values that are already present in the template, in contrast to parameter values, which are passed in at execution time. Variables can also be used to simplify template language by using expressions and functions, such as “[resourceId…]” and “[concat…]”, as highlighted above. In the case of the concat function, variables can be composed of other variables or parameter values that are input at run time. ResourceId is a helper function that is used to get unique IDs of resources.  1. Scroll down to the “resources” section that begins at around line 59.   Machine generated alternative text: 61  62  64  65  66  67  68  69  " resources  " type"  name  " Mi crosoft . Storage/ storageAccounts  " (parameters( ' newStorageAccountName ' ) J"  "2B15-e5-e1-preview",  " (variables( ' location ' ) ) " ,  " apiVersion  " location  •properties  • "(variables( ' storageAccountType' ) J"  " account Type   * In the “resources” section, you define the resources that you wish to deploy or update. Each resource must include a type, name, and “apiVersion”. The “type” value is the namespace of the resource provider and the supported resource type. The value for the type is a substring of the URL you would use if you wanted to create a resource, such as a storage account, against the Azure representational state transfer (REST) application programming interface (API). When the template is deployed, the template engine makes the same HTTP PUT calls that would be made if you were directly interacting with the API. * If you wish to find out the values and the properties associated with a resource, you can find the documentation on the Azure REST APIs here: <https://msdn.microsoft.com/en-us/library/azure/mt420159.aspx>. For example, for more information about the Microsoft.Storage/storageaccounts resource, you can consult the Azure Storage Provider REST API reference at <https://msdn.microsoft.com/en-us/library/azure/mt163683.aspx>. * Optional key/value pairs for resource objects include “location” (where you define the geo-location for the resource), “properties” (where you define resource-specific settings), “tags” (which provide metadata associated with the resource), “resources” (child resources that depend on the resource being defined), and “**dependsOn**” (resources that the resource being defined depends on).  1. Scroll down to the beginning of the virtual machine resource definition, which begins at around line 129.   Machine generated alternative text: 131  132  133  134  135  136  137  " apiVersion  "2B15-e6-15",  " type"  name  " Mi crosoft . Compute/ vi rtua1Machi nes  " (variables( ' vmName ) J",  " (variables( ' location ' ) ) " ,  " location  " dependsOn" •  "L concat( 'Microsoft. Storage/ storageAccounts/' , parameters( 'newStorageAccountName ) ) J"  "L concat( 'Microsoft. Network/networklnterfaces/' , variables( ' nicName ) ) )   * Note the “**dependsOn**” key/value pair. Before the virtual machine can be created, the storage account and the network interface—which in turn has dependencies of its own (see line 134) —must exist. The script is evaluated for dependencies to create objects in the appropriate order. If no dependencies are specified, ARM attempts to create them in parallel. This can result in much faster deployments for large and complex infrastructures over the Azure Service Manager mode.  1. In the Microsoft Edge browser, click **Back** to return to the previous page. 2. Click on the **101-vm-simple-windows** page, click **Deploy to Azure**.      On the Microsoft Azure sign-in page, enter the email address and password associated with your subscription, and sign in.   * The Azure Portal opens, showing the Custom deployment and Parameters blades. * Note that the **Parameters** blade provides fields for text input. In the case of the **WINDOWSOSVERSION** parameter, note the presence of a default value and a drop-down list of allowed values.     Machine generated alternative text: Microsoft Azure  Create  Browse  Create  Custom deployment  Parameters  Custom deployment  De p by a template  Template  Edit template  Parameters  Edit parameters  Subscription  Free Trial  Resource group  Select a resource group  Or create new  Resource group  East US  Legal terms  Review terms and agreements(de„  Pin to Startboard  Parameters  late  NEWSTOUGEACCOUNTNAME  ADMINUSERNAME (sting)  ADMINPASSWORD (securestring)  DNSNAMEFORPUBUCIP (sting)  WINDOWSOSVERSION (string)  2012-R2-Datacenter  BROWSE ALL  NOTIFICATIONS  BILLING  1  Compute  Web + Mobile  Data + Storage  Data + Analytics  Networking  Media + CDN  Hybrid Integration  Security + Identity  Developer Services  Management  Container Apps  Marketplace  Recent  Template deployment  Microsoft  Windows Server 2012 R2 Datac...  Microsoft     1. On the Custom deployment blade, click **Edit template**.  * You can edit the template before deploying it—for example, to change variable names or to modify resources that the template describes for deployment.  1. Close the Edit template blade. (scroll upper right-hand **X** Close). 2. In the **Custom deployment** blade, in **Resource group**, click **Or create new**. 3. In the Create a new resource group text box, type **test** 4. Click Resource group location.  * Note that once you enter a name for a new resource group, you can also specify a location.  1. Review the locations, and then close the Location blade. Do not select a location at this time. 2. Close the Custom deployment blade.  * You are not going to deploy the template at this time. Rather, you will modify the template in a subsequent exercise and then explore other ways to deploy the template.  1. If prompted by a warning indicating **“This site says, your unsaved edits will be discarded**”, click **OK**. 2. Leave the Azure Portal open for a subsequent lab exercise. |

**Exercise 3: Configure Git repository and Visual Studio Code**

As you saw in the previous activities, GitHub is used to store ARM templates and other code-related objects. GitHub is a web-based Git repository to ensure availability and provide a version control system for files.

Version control systems record changes to files or a set of files so that they can retrieved at a later time. There are different kinds of version control systems, the simplest being the manual use of multiple folders to store different versions of your files.

More sophisticated and fault tolerant version control systems use central or distributed databases to store the files and their changes.

Git is a distributed version control system in which clients clone entire repositories to ensure that there is no single point of failure. If a particular repository fails, the contents can be restored from a cloned copy that exist elsewhere.

In this exercise, you will learn more about GitHub and Git repositories. You will also learn how to configure Visual Studio Code to use Git.

***Activity 1: Create GitHub account and public repository***

| Description | Steps |
| --- | --- |
| The remaining lab activities require that you have a GitHub account and repository named Templates. In this activity, you create a GitHub account and then configure a public repository named Templates. | * This lab activity requires that you have a GitHub account. * Note that, if you do have a GitHub account and you have enabled two-factor authentication for it, you might have difficulties authenticating to your account when using Visual Studio Code. If you have two-factor authentication enabled on your Git repository, consider creating a Git account for this lab.  1. On **AZRCAMP-ADMIN**, open Microsoft Edge. 2. Navigate to [**https://github.com**](https://github.com)**.** 3. On the home page, click **Sign up**. 4. On the Join GitHub page, enter a username, email address, and password.  * Please note that when you perform certain operations on GitHub, such as commits, your email address is publicly viewable. For more information, please see <https://help.github.com/articles/keeping-your-email-address-private/>.  1. Click **Create an account.** 2. On the Welcome to GitHub page, click **Finish sign up**. 3. On the GitHub Bootcamp page, click **Create a repository**. 4. Open your email client and open the verification email sent to you by the GitHub website. 5. Click **Verify email address**. 6. Switch to the GitHub page. 7. Click **Create a repository** (alternatively, in the upper right corner, click the plus sign (**+**), and then click **New repository**). 8. In Repository name, type Templates, then select Initialize this repository with a README and then click Create repository.  * You will clone the repository in the next activity.   Leave the Microsoft Edge browser open for the next activity. |

#### Activity 2: Configure a GitHub Desktop and Clone Repository

| Description | Steps |
| --- | --- |
| In this activity, you will launch GitHub, configure settings for the application, and then clone the Templates repository you configured. | 1. Switch to the desktop of the **AZRCAMP-ADMIN**. 2. On the desktop, double-click **GitHub**.   The GitHub desktop application was installed as part of the lab setup.   1. On the Welcome page, enter your GitHub username and password, and then click **Log in**. 2. Click **Continue**. 3. Under No local repositories found, click **dashboard**.  * The GitHub desktop application opens, showing a preconfigured Git repository named Tutorial that contains .gitattributes and .gitignore files. Before proceeding with the next steps, feel free to launch the tutorial by clicking Got it!. Otherwise, close the tutorial.  1. In the upper right corner, click **Settings** (upper right corner, the gear icon).     Machine generated alternative text: TutMial  master  Welcome! Let's walk through the  basics of GitHub Flow:  l. Make a branch.  Z Make changes,  3. Open a pull request  If want to this at  time go to ard  Tutorial.  Got it! more  No uncommitted changes History  Added .gitattributes & .gitlgnore files  .gitattributes  .gitignore  n Pull æquest  Publish  Expand all  Reæ-t     1. Click **Options**.     Machine generated alternative text: n Pull request  Undo most recent commit  Delete master...  Repository settings...  Open n Explorer  open n Git Shell  View cn GitHub  Tutorial  Options...  About GitHub Desktop...     1. On the Options page, in Clone path, enter **C:\GitHub**, and then click **Save**.   (Enter your username and email address if needed).  Machine generated alternative text: @ Options  Accounts  Free plan (O private repositories)  + Add GitHub Enterprise account  Configure git  Log out  Manage  Clone path  CAGitHub  Create and clone repositories into this directory by default.  Q Scan for repositories  Find repositories on your hard drive.  Default shell  Git  Privacy  Help us improve by sending anonymous usage data  This Will be used in the commits you Create. Keep in mind that if you  publish commits. anyone will have access to this email.  This will change your global gitconfig.  save @cancel       1. On the GitHub home page, in the upper left corner, click **Add** (plus sign).     Machine generated alternative text: Local path  Git ignore  master  Add Crea  B rows e  Clone     1. Click **Clone**. 2. Click **Templates**, and then click **Clone Templates**.   Machine generated alternative text: Add Create С1опе  Н Тег  Тетр|аЖ  С|опе Templates     1. In the Browse for Folder dialog box, ensure that **C:\GitHub** is selected, and then click **OK**.  * If you receive an error, ensure that the Tutorial repository is closed/removed and perform step 12 again.  1. On the desktop, double-click **Git-Shell**.  * A PowerShell window opens. The option to integrate the Git commands with PowerShell is configured in the GitHub client.  1. At the PowerShell prompt, type the following command and press ENTER.   git config --global credential.helper wincred   * This command is necessary to ensure that Visual Studio Code can authenticate against the GitHub website. The command is case-sensitive.  1. At the PowerShell prompt, type the following command and press ENTER.   git config --list   * This command shows the settings that are configured for Git. Note that the bottom of the output displays your GitHub username and email address. These settings were configured when you logged on to the GitHub Windows client. Normally, you would configure Git using commands from a shell, such as Bit Bash or the integrated PowerShell prompt you see in the lab. For more information about configuring and using Git, please see <http://git-scm.com/doc>.  1. Open File Explorer, and browse to **C:\LabFiles\AZITPROCamp\Lab01-Deploy Azure resources using templates**. 2. Select and right-click **101-simple-windows-vm**, and then click **Copy**.  * For your convenience, the Azure quickstart template you examined in the previous activity was previously downloaded as part of the lab setup.  1. In File Explorer, navigate to **C:\GitHub\Templates.** 2. Press CTRL-V to paste the folder to **C:\GitHub\Templates**.  * In File Explorer, on the Ribbon, click **View**, and then select **Hidden items**. * The hidden .git folder appears. This folder is created upon initialization of the Git repository. The .git folder includes the Git database and other files necessary for staging and committing changes to files for source version control.          1. Close File Explorer. |

#### Activity 3: Examine Visual Studio Code Integration with Git and Push Commits to Remote Repository

| Description | Steps |
| --- | --- |
| Visual Studio Code has integrated support for the most common Git commands—for example, to stage and commit changes.  In this activity, you will see a demonstration of Visual Studio Code integration with Git and push commits (changes you wish to be permanent) to the remote repository on GitHub. | 1. On the desktop or from the taskbar, open Visual Studio Code.     C:\Users\martin\AppData\Local\Temp\msohtmlclip1\02\clip_image007.png   1. If you see a message indicating that you need to install Visual Studio Code, please ignore the message (or click **Later**) and complete the steps below. 2. In Visual Studio Code, click **File**, and click **Open Folder**. 3. In the Select Folder dialog box, navigate to **C:\GitHub**, click **Templates**, and then click **Select Folder**. 4. The folder opens and Visual Studio Code provides a visual indication that Git detects uncommitted files in the folder. These are the files you copied to the folder earlier. 5. Click on the **folders** icon on the left   **Note -** You might have to close and reopen the folder to see the uncommitted indication as per the screenshot below.  To close the folder, click **File**, and click **Close Folder**.  Machine generated alternative text: Template Studio  File Vie•.•.' Goto kelp  WORKING FILES  -simple-windows-vm  azured  R EA OME md   1. Click the Git icon on the left.   Machine generated alternative text: • Code  Goto  Message (press Ctrl* Enter to  azuredeploy.json  azured40y.pararneteø.'s—   1. Press the CTRL key down, and select both files. 2. Click **ellipsis** above , and then click **Commit All.**  * NOTE: if you see an update notification for VS Code, just ignore it. It’s likely hiding the ellipses. The labs don’t require this update. * Git files have three states: modified, staged, and committed. Modified means that the files are in a workspace and have been changed but have not yet been committed to the database in the repository. Staged means the files have been moved to a staging area prior to being committed—for example, as a holding area pending review. Committed means that a snapshot of the file is committed to the database.     Machine generated alternative text: azuredeploy.json - Templates - Visual Studio Code  File Edit View Goto Help  Message (press Ctrl+EntE  azuredeploy.json ICI  azuredeploy.paramet  azuredeploy.jsol  Sync  Push  Commit Staged  o mm It  Undo Last Commit  Unstage All  Clean All  Show Git Output   1. In Message, type **Initial Version**, and press CTRL+ENTER.   These files will represent a starting point for changes you will make in a subsequent activity.  Machine generated alternative text: - Template - Viwal  Initial Version  azure&pkry.json  parametersjs„  The changes are committed to the local repository. The files disappear from view in the Git node. In the next step, you will push the committed files to the remote repository.   1. Click the ellipsis, and then click **Push**. 2. When prompted by the Visual Studio Code dialog box, enter your Git username and password, and click **OK**.   **Error occurs** - The authentication fails to show that the authentication has not happened.   1. When prompted by an error message indicating that authentication failed on the Git remote, click **Close**. 2. Open File Explorer, and browse to **C:\LabFiles\Utils** and double-click **git-credential-winstore.exe**.   Machine generated alternative text: Appliation  re  , Thispc  Quick access  Desktop  git-eredential-win  Picture  AZITPRocamp  Utils  LabFilE  Date modified  9/14/2015 PM  PM  9,'21/201BSEPM  geos AM  9,114/2015 4:30 PM  Type  File folde  File folde  Application  Application  Application  1goKB  Utils   1. When prompted, click **Yes**.   This small executable file downloaded from Codeplex.com resolves an issue with authentication to GitHub using Visual Studio Code.  If you have enabled multi-factor authentication on your GitHub repository, this workaround might not be effective.   1. Switch to Visual Studio Code, click the ellipsis, and then click **Push**. 2. When prompted by the Git Credentials dialog box, enter your Git username and password, and click **OK**. 3. The committed files in the local repository are pushed to the repository on GitHub. 4. Leave Visual Studio Code open for subsequent lab steps, and switch to the instance of Microsoft Edge that you left open in a previous activity. 5. Ensure that you are viewing the Templates page for your repository, and then press F5 to refresh the page.   You should see the 101-simple-windows-vm folder from your local repository.   1. Click 101-simple-windows-vm. 2. You should see the two JSON files in the folder.   Machine generated alternative text: This repository  Branch: master  Initial Version  Search  / Templates  Pull requests  Issues  Gist  @ Unwatch  Star  0  Fork  Templates / 101-simple-windows-vm / +  latest commit dfe8ab618d  20 minutes ago  20 minutes ago  authored 20 minutes ago  azuredeploy_json  azuredeploy_parameters_json  Initial Version  Initial Version   1. Leave the browser open for subsequent steps in the next activity. |

**Exercise 4: Modify and Deploy Azure Quickstart Templates**

As you learned in the previous exercise, it is rarely necessary to construct a JSON-based Azure Resource Manager template from scratch. In many instances, you will be able to find a sample template to use as a starting point for your own template. A basic understanding of how templates are constructed, combined with an appropriate sample template to use as a starting point, will allow you to create your own custom templates with relative ease.

For additional information on authoring and modifying template files, please see Authoring Azure Resource Manager templates at <https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/>.

In this exercise, you will build on the understanding that you acquired in a previous exercise to customize a sample template and then deploy it.

***Activity 1: Modify sample template and parameter JSON files***

In a previous exercise, you examined the 101-simple-windows-vm template that is available on GitHub. You subsequently added this template to your own Git repository.

| Description | Steps |
| --- | --- |
| In this activity, you will customize this template and its related parameters’ JSON file using Visual Studio Code to meet your specific and additional requirements. Also, you will stage and commit your changes to the local and remote Git repository. | 1. If it is not already open, open **Visual Studio Code**, and navigate to the C:\**GitHub\Templates\101-simple-windows-vm** folder. 2. In the tree pane, click **azuredeploy.json**.  * Note that IntelliSense and autocomplete make the activity of entering JSON easier. * Recall that the parameters are passed to the template at run time and allow users to specify values.  1. Next, above the “**WindowsOSVersion**” parameter, add the “**location**” parameter with a default value, restricting choice to locations where premium storage is available.  * Type in the following syntax listed in the screenshot. * **Tip:** You can find the completed solution in the C:\LabFiles\AZIT{ROCamp\Solutions folder for Lab01 in the azuredeploy.json file. You can copy and paste code from the solution into your JSON script.        1. Scroll down to the “**variable**s” section, and delete the **“location”: “West US”** key/value pair.     Machine generated alternative text: "variables" .  " imagePub1 isher"  "MicrosoftWindowsServer"  " imageOffer" :  "OSDiskName" :  "WindowsServer"  "osdiskforwindowssimple"    Machine generated alternative text: "variables"  " imagePub1 isher" :  "MicrosoftWi ndowsServer"  " imageOffer" :  "OSDiskName" :  "WindowsServer"  "osdiskforwindowssimple"   1. Click **Edit** and then click **Replace**.   Machine generated alternative text: • azuredeploy.json - Templates - Visu,  View Goto Help  Copy  Select All  Replace  Find in Files  Ctrl*Z  Ctrl*Y  Ctrl*X  Ctrl*C  Ctrl*V  Ctrl*A  Ctrl*F  Ctrl+Shift+F   1. In Find, type **variables(‘location’)**; in Replace, type **parameters(‘location’)**, and then click **Replace All**, as shown below. Note – They are not quotes “      1. Scroll up to the end of the “**parameters**” section and the start of the “**variables**” section. 2. At the end of the second-to-last closing brace, before the “**variables**” section, add a comma, as shown below. Look for line 34.   Machine generated alternative text: "description" :  "variables"  " imagePub1 isher" :  "The Windows version for the W.  "MicrosoftWi ndowsServer"   1. Then, add the following lines of code to create a parameter for the size of the data disk.  * **Tip:** You can find the completed solution in the C:\LabFiles\AZIT{ROCamp\Solutions folder for Lab01 in the azuredeploy.json file. If you like, you can copy and paste code from the solution into your JSON script.        1. Scroll to the end of the “**variables**” section, and **add a comma at the end of the “subnetRef**” key/value pair:        1. At the end of the “**variables**” enter a new line. 2. Type the following **on a single line** OR cut and paste it from the solutions file.   **“dataDisk1VhdName”: “[concat(‘http://’,parameters(‘newStorageAccountName’),’.blob.core.windows.net/’,variables(‘vmStorageAccountContainerName’),’/’,variables(‘vmName’),’dataDisk1.vhd’)]”**   * You can find the completed solution in the C:\LabFiles\AITPROCamp\Solutions folder for Lab01 in the **azuredeploy.json** file.  1. Scroll down to the “**resources**” section and locate the “**osDisk**” section. 2. Between the “**osDisk**” around line 153 and “**networkProfile**” sections, add the following resource: found in Lab01 azuredeploy.json file.   Machine generated alternative text: 171  172  174  176  177  178  181  182  187  "osDisk" :  name" :  "vhd":  " url "  "osdisk" ,  " L concat( http://'.parameters('newStorageAccount  "caching .  "ReadWrite"  "createoption" :  " Fromlmage "  "dataDisks" .  " name  "datadiskl " ,  "diskSizeGB" :  " L parameters( ' sizeOfDiskInGB ) ) "  "Nun": e,  "vhd":  "Uri" .  " (variables( ' dataDisk1VhdName ' ) I "  " Empty"  " createoption" :  "networkprofile":  " networklnterfaces " .  (resourceld( Microsoft. Network/networklnterfaces   * Be very careful where you place this code section. Before proceeding to the next step, scroll to the end of the file and double-check for any red squiggles that might indicate improper placement of this snippet.  1. Click **File** and then click **Save**. 2. Click the Git icon on the left.   Machine generated alternative text: azuredeploy.json - Templates - Visual Studic  File Edit View Goto Help  Message (press Ctrl+Enter to  CHANGES  azuredeploy.json 101-sim...   1. Under **CHANGES**, click **azuredeploy.json**.  * Two screens appear. The left screen shows the original file; the right screen shows the changes.   Review your changes, right-click **azuredeploy.json**, and then click **Stage**.   1. On the left, click the Explore icon. 2. Click **azuredeploy.parameters.json**. 3. In **azuredeploy.parameters.json**, add key/value pairs for the **location**, **WindowsOSVersion**, and **sizeOfDiskInGB** parameters. 4. Below, provide your unique names for the **storage account** and **DNS name**, the storage account name that was documented and created in exercise 1, for your DNS name use your initials followed by at least 4 or more integers, as follows:   **Do not change “adminUsername”,** but only change “newStorageAccountName”, adminPassword”, “dnsNameForPublicIP” and add “location” and “sizeofDiskInGB”.    Machine generated alternative text: • azuredeploy.parameters.json  I O I -si mple-windows-vm  "$schema" .  "http://schema.management.azure.com/schemas/2915-91-Bl/deploymentparameters.json*" ,  " contentVersion" :  "l.ø.e.ø",  "parameters" :  " newStorageAccountName " :  "value" .  " unique-name-you-recorded -in-exercise-2"  " adminUsername":  "value" .  " adminPass  "value" .  " admin123"  "Passwerd ! "  " dnsNameForPub1icIP" :  "value".  " location .  "value".  " your- initials -plus -4 -or -more- random- integers- to-ensure-name-uniqueness "  "West US"  "sizeOfDiskInGB":  "value".   1. Add **Location** snipetwhich can be found in the **azuredeploy.parameter.json** solution file. 2. Add **sizeOfDiskInGB** snipet can be found in the **azuredeploy.parameter.json** solution file. 3. Save the **azuredeploy.parameter.json** file. 4. Click the Git icon in the navigation pane, right-click **azuredeploy.parameters.json**, and click **Stage**. 5. Click the ellipsis, and then click **Commit Staged**. 6. In the Message box, type **Update1**, and press CTRL+ENTER.  * A snapshot of the files is now committed to the local repository.  1. Click the ellipsis, and then click **Push**.  * Note that you are not prompted for credentials. You configured credential caching earlier with the **git config --global credential.helper wincred** command.  1. Switch to the Microsoft Edge browser you left open in the previous activity, and press F5 to refresh the page.  * You should see that the two JSON files have been pushed to the remote repository.      1. Leave the browser open for the next activity. |

***Activity 2: Deploy custom JSON files***

In the previous activity, you modified the sample JSON to meet specific requirements and subsequently committed your changes to both a local and remote Git repository.

| Description | Steps |
| --- | --- |
| In this activity, you will examine deploying the custom template using the Azure Portal and Windows PowerShell. | 1. If it is not already open, open Visual Studio Code, and navigate to the C:\**GitHub\Templates\101-simple-windows-vm** folder. Make sure you copy the local template and not the one in the lab environment! 2. In the tree pane, click **azuredeploy.json**. 3. Highlight and copy the JSON code 4. Switch to Microsoft Edge. 5. If you are not already logged on to the Azure Portal, open a new tab, navigate to <https://portal.azure.com>, and log on the account you are using for your Azure subscription.   Note: You may need to refresh your Azure Portal   1. In the Azure Portal, in the navigation pane, click **New**. 2. In the New blade, click **Marketplace – See all**. 3. In the **Search everything** text box, type Template, click Template deploymentfrom the drop-down list, and press ENTER.      1. Double-click **Template deployment** in the results. 2. In the **Template deployment** blade, click **Create**. 3. The **Custom deployment** blade appears. Scroll to the right to see the blade. 4. In the Custom deployment blade, click Edit template. 5. In the Edit template blade, **delete all the lines of JSON script**. 6. Place the cursor in the template area, and press **CTRL+V** to paste the JSON script that you copied to the clipboard earlier. 7. Click **Save**.      1. Click Edit parameters.   The Parameters blade should appear immediately. If it does not, this means that there is something wrong with your JSON script, most likely a mismatch between opening and closing braces or brackets.  If the template has the proper formatting, you will see a small green rectangle in the upper right side of the template; otherwise, you will see small red squares on the right side that provide an approximate location of the source of the problem.   1. Review, in the Parameters blade, note the presence of the **LOCATION** (string) and **SIZEOFDISKINGB** (string) parameters that you added to the original JSON script.   Machine generated alternative text: Custom deployment  Depby a  Template  Edit template  Parameters  Edit parameters  S ubscription  Free Trial  Resource group  Select a resource group  Or create new  Resource group location  West US  Legal terms  Review terms and agreements(de...  Pin to Startboard  x  Parameters  temp ate  ADMINUSERNAME (string)  ADMINPASSWORD (securestring)  DNSNAMEFORPUBLICIP (string)  LOCATION (string)  West US  WINDOWSOSVERSION (string)  2012-R2-Datacenter  SIZEOFDISKINGB (string)  x   1. Close the Parameters and Custom deployment blades without creating the deployment. 2. You are going to deploy your custom template using a PowerShell command in a later time. You examined how to deploy a custom template in the Azure Portal as a demonstration and to do a final verification on your template. You can always deploy the template again at a later time. 3. When prompted, click **OK to discard unsaved edits.** 4. Close all open blades in the portal to return to the Start page. 5. Switch to **Microsoft Edge**. 6. In the tab that displays the contents of your **GitHub Templates repository**, click **azuredeploy.json**. 7. On the **azuredeploy.json** page, click **Raw** 8. Copy the URL displayed for the raw version of the **azuredeploy.json** file to the value of the **$templatefileURI** on line 57 of the Lab01Start.ps1 script.      1. Hit the 🡨 in the browser to go back from RAW 2. Repeat steps 58-59, open the **Github Templates Repository** to copy the URL for the **azuredeploy.parameters.json** file to the value for the **$parameterfileURI** variable on line 59 of the Lab01Start.ps1 script. 3. The resulting section of the script should look like something this:      1. Select the entire block of variables, and click **Run Selection**.      1. Scroll to the bottom of the script, and remove the # at the beginning of the second command to perform a resource group deployment. Look for line 106. 3. Select the entire line, and then click **Run Selection**.   After a few minutes, the command should return a successful result. You can verify the results by viewing the new resources in the Azure Portal.   1. Please leave the Windows PowerShell ISE console open for the next activity. |

**Exercise 5: Remove resource group used for lab**

Because each lab in this series begins with an empty resource and because Azure resources are potentially billable, it is necessary to remove the resource group you created and used in this lab. Also, because Azure trial accounts are limited to 4 compute cores, it is important that you remove the resource group to ensure you do not run out of resources, if you have an Azure trial account.

***Activity 1: Remove Azure resource group***

| Description | Steps |
| --- | --- |
| In this activity, you will run a Windows PowerShell script to remove the resource group you created and used in this lab. | 1. If it is not already open, open the Windows PowerShell ISE. 2. Click **File**, click **Open**, browse to **C:\LabFiles\AZITPROCamp\Scripts, select RGCleanup.ps1**, and click **Open**. 3. On the menu, click **Run**. 4. When prompted, log on to your Azure subscription. 5. When prompted to delete **RG-AZITCAMP-LAB01**, click **Yes**. 6. If you used a different resource group for the lab, you can modify the PowerShell script to delete the resource group. |

**Proceed to Lab 2 “Designing Compute and Storage” in the online learning portal**.

Lab 2: Design Azure Storage and Compute Infrastructure

|  |
| --- |
| Microsoft Azure offers many choices for designing compute, storage, and other resources that provide an optimal fit for a particular workload or set of workloads. Designing resources for particular workloads requires an understanding not only of the capabilities and cost associated with particular resources but also of subscription and service limits, quotas, and constraints.  In this lab, you will learn about the relevant capabilities within Azure that will enable a cost-effective design that meets performance requirements. |

**Introduction**

Contoso is still in the process of evaluating recently added new features to Azure, such as ARM. Contoso has a number of large applications that use significant amounts of disk space for both read and write operations. Consequently, Contoso is interested in investigating how recently added features, such as new virtual machine scale units and Premium Storage, can provide both performance and cost benefits. You have been asked to investigate these features and create a sample ARM template to deploy a relatively high-performant application.

**Exercise 1: Prepare the Azure Infrastructure**

In this exercise, you will use the **Lab02Start.ps1** script to log on to your Azure subscription and create an Azure resource group that you will use for the remaining lab activities. The script will also determine a globally unique name that you can use to create a storage account in subsequent lab activities.

Additionally, you will also deploy a STANDARD\_D2 virtual machine that has four striped data disks that use Standard Storage. You will use this virtual machine to compare and contrast disk performance with a STANDARD\_DS2 virtual machine that uses Premium Storage.

***Activity 1: Run the Lab02Start.ps1 Script***

| **Description** | **Steps** |
| --- | --- |
| To perform the subsequent lab activities, you need to create two Azure resource groups and determine two globally unique names that you can use to create your storage accounts. | * Perform the following activities on **AZRCAMP-ADMIN**:  1. Open File Explorer and navigate to **C:\LabFiles\AZRITPROCamp\Lab02 – Design Azure compute and storage.** 2. Right-click **Lab02Start.ps1**, and click **Edit**.    1. The Windows PowerShell Integrated Scripting Environment (ISE) console opens. 3. In the PowerShell ISE, on the upper Ribbon, go to **Edit** – **Select All** - click **Run Script** (green arrow). You may get a red screen until you put in your Azure credentials. 4. When prompted, enter a lowercase string that represents your initials, and press **ENTER**.   The storage account name must contain only lowercase letters and numbers and must be globally unique.   1. In the **Sign In to Windows Azure PowerShell** dialog box, enter the email address of the account associated with your Azure subscription, and click **Continue**. 2. On the sign-in page, enter your password, and click **Sign in**.   The script starts running and then pauses to display the two names verified as unique for use with the storage account names in your lab.   1. Write down the unique storage account names below:   **Standard Storage Account Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Premium Storage Account Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**   1. Press **ENTER** to continue.   You will need to know the unique name for the storage account in a subsequent lab activity.   1. Leave the **PowerShell ISE** console open for subsequent lab activities. |

***Activity 2: Deploy a D2 Virtual Machine with Standard Storage***

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will deploy a D2 virtual machine from a preconfigured template. You will subsequently use this virtual machine to compare it with a DS2 virtual machine that uses Premium Storage. | 1. Perform the following activities on **AZRCAMP-ADMIN**: 2. Open **Microsoft Edge**, and browse to [**https://github.com/AZITCAMP/Labfiles/tree/master/lab02**](https://github.com/AZITCAMP/Labfiles/tree/master/lab02)   The lab02 repository contains all the files you need to complete the lab.     1. Scroll down, In the **Readme.md** section, click **Deploy to Azure**. 2. If prompted, log on to the Azure Portal.   The Parameters tab for the virtual machine custom deployment appears.   1. In **NEWSTORAGEACCOUNTNAME**, enter the unique name of the Standard Storage account that you documented in the previous activity.   The name will look like **[abc]stdstore#**, where [abc] represents your initials and # is an integer. It might look like bbstdstore1.   1. In **DNSNAMEFORPUBLICIP**, enter your initials plus four random digits to ensure uniqueness. It might look like bb5555. 2. In **ADMINPASSWORD**, enter Passw0rd!.   Take note of this account and password for use in future activities.  **Storage Account Name : \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Password :\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**   1. Leave the remaining parameters at their default value, and click **OK**. 2. in the **Resource group section,** click Select a resource group, and select R**G-AZITCAMP-STD**.   Scroll down in the windows below “Resource Group Location” to see **Legal terms -**   1. Click **Legal terms**. Click **Create.** 2. On the Custom deployment tab, click **Create**.   You might see errors regarding the deployment of Desired State Configuration Extensions. Wait a few minutes, and the problem will resolve itself.   1. Leave the **Microsoft Edge browser open** for subsequent steps, and proceed to the next exercise without waiting for the deployment to complete. |

***Deploy a Virtual Machine with Premium Storage***

In a previous exercise, you deployed a D2 virtual machine from a template. In this exercise, you will customize the template used to deploy the D2 virtual machine to deploy a DS2 virtual machine that uses Premium Storage. In a subsequent lab exercise, you will compare the performance of the two virtual machines.

***Activity 1: Modify the Azuredeploy.json Template File***

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will download an azuredeploy.json file from GitHub and then modify it to deploy a virtual machine that uses Premium Storage rather than Standard Storage. | 1. If it is not already open, open the Microsoft Edge browser and navigate to <https://github.com/AZITCAMP/Labfiles/tree/master/lab02>. 2. Click **azuredeploy.json**. 3. On the azuredeploy.json page, click **Raw**. 4. Press CTRL+A to select all the text in the window, and then press CTRL+C to copy the text to the clipboard. 5. From the activity bar, open Visual Studio Code. 6. In Visual Studio Code, place the cursor in the blank window and then press CTRL+V to copy the text. 7. Click **File**, and then click **Save As**. 8. In the Save As dialog box, browse to **C:\LabFiles\AZITPROCAMP\Lab02**. 9. In File name, type **azuredeploy-premstorage.json**, and click **Save**. 10. Locate the **“vmSize”** parameter at around line 30. 11. Change all the instances for Standard\_D# to **Standard\_DS#**, where # is an integer.   To use Premium Storage, you must deploy a virtual machine that belongs to either the DS or the GS tiers.   1. Machine generated alternative text:    -vmSize":     •type": -string",     -descri ion": "Virtual Machine Size"     1     - defaultVaIue- :     "StandaM DS2",     osr,     -Standard     cyst,     -Standard     Dsy-,     -standard 2. Scroll down to the “variables” section, and locate the **“storageAccountType”** key/value pair at around line 85. 3. Change Standard\_LRS to **Premium\_LRS**. 4. A couple of lines below storageAccountType, locate the **“vmName”** key/value pair. 5. Change STD-D2-VM to **PREM-DS2-VM** 6. Machine generated alternative text:    - latest"     -publiclPAddressName": "nyPubIicIP".     -stora     vne t     -nicUame-:     count pe": "Premiurl CRS",     "PREM-DS2-VH"     resource     crosoft. Network/virtuaINetwork5 , vat     -subnet1Ref" :     'vnet1D'), Isubnets/' , variab     -configurationFunction": "ConfigureVN.psftNConfigureVM-, 7. Scroll down to the “resources” section, and locate the “dataDisks” section that starts at around line 192.   Note that read-only disk caching is configured here, along with other aspects of the disk, such as the LUN.   1. Scroll down to the end of the file and note the “extension” section that begins at around line 253.   In the “extension” section, you can specify the Desired State Configuration (DSC) properties. DSC allows you to, among other things, automate the post-deployment configuration of the virtual machines. In this case, the PowerShell script used by the DSC extension initializes the four data disks, creates a storage pool, creates a striped volume from the disks in the pool, and then copies some files to the virtual machine for later use in the lab.  Machine generated alternative text: "type": "Microsoft. Compute/virtual"achines/extensions",  "nam": '/' ,  variables(  "apiversion-.  -2e1s-e5-a1-preview" ,  "location";  "dependsOn-:  (concat( 'Hicrosoft.compute/virtualnachines/' ,  "properties - : (  -publisher"  "Microsoft.powershell".  -type-: - DSC -  -typeHandIerVersion": "1.9",  -settings": (  "SasToken":  variables( ' vmNue.  "ConfigurationFunction": "(variables( 'configurationFunction  "Properties": (  -protectedSettings " :  null   1. Click **File**, and then click **Save**. 2. Leave the file open for the next activity. |

***Activity 2: Deploy a Virtual Machine from a Template***

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will deploy a virtual machine that uses Premium Storage rather than Standard Storage. | 1. Switch to the **Microsoft Edge** browser. 2. If you are not already logged on to the Azure Portal, navigate to <https://portal.azure.com> and log on with the account associated with your subscription. 3. In the Azure Portal, in the left pane, click **New** 4. Click **Search** for **Template Deployment** 5. In the New blade, click **Template deployment**. Click **Create**   If you do not see this item for selection, you will need to search for it in the Marketplace.   1. In the **Custom deployment** blade, click **Edit template**. 2. In the Edit template blade, select all the text and then delete it. 3. Place the cursor in the Edit template blade, and then press CTRL+V to paste the code you copied to the clipboard earlier. 4. Click **Save**. 5. In the Custom deployment blade, click **Parameters, Edit Parameters** 6. In **NEWSTORAGEACCOUNTNAME**, enter the unique name of the Premium Storage account that you determined in the previous activity.   The name will look something like [abc]premstore#, where [abc] represents your initials and # is an integer.   1. In **DNSNAMEFORPUBLICIP**, enter your initials plus four random digits to ensure uniqueness. 2. In **ADMINPASSWORD**, type **Passw0rd!** 3. Leave the remaining parameters at their default value, and click **OK**. 4. On the Custom deployment blade, in the Resource group section, click **Select a resource group**, and select R**G-AZITCAMP-PREM** select “Legal Terms” 5. On the **Custom deployment** tab, click **Create**. 6. **Leave the Microsoft Edge browser open** for subsequent steps, and proceed to the next activity without waiting for the deployment to complete. |

**Exercise 3: Compare Standard and Premium Storage**

In the previous exercise, you deployed both a D2 virtual machine that uses Standard Storage and a DS2 virtual machine that uses Premium Storage. In this exercise, you will connect to the virtual machines, install Iometer software to measure disk IOPS and throughput, and then perform a basic benchmark test to compare the performance of the virtual machines.

***Activity 1: Connect to STD-D2-VM and Run a Disk Performance Test***

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will connect to STD-D2-VM, install and configure Iometer, and then run a bench mark test. | 1. In the Azure Portal, in the left navigation, click **All resources**, and then click **STD-D2-VM**. 2. In the STD-D2-VM blade, click **Connect**. 3. When prompted, click **Open**. 4. In the Remote Desktop Connection dialog box, click **Connect**. 5. In the Windows Security dialog box, click **Use another account.** 6. In Username, type **itcampadmin**, in Password, type **Passw0rd!**, and then click **OK**. 7. In the Remote Desktop connection dialog box, click **Yes**. 8. If prompted to choose network preferences, click **No**. 9. Open **File Explorer**, and navigate to **C:\Source**. 10. Right-click **iometer.zip**, click **Extract all**, and then click **Extract**. 11. In C:\Source\iometer, double-click **iometer-setup.exe**. 12. On the Welcome to Iometer 1.1 Setup Wizard page, click **Next**. 13. On the License Agreement page, click **I Agree**. 14. On the Choose Components page, click **Next**. 15. On the Choose Install Location page, click **Install**. 16. Click **Finish**. 17. Click **Start**. 18. On the Start page, type **Iometer**, and then click **Iometer**. 19. Click **I Agree**. 20. Click **Open Test Configuration File** (the folder icon in the upper left). 21. In the Open Test Configuration File dialog box, navigate to C**:\Source\Iometer, select iometer.icf**, and click **Open**.   The iometer.icf file was configured earlier as part of this lab setup.   1. Click **Start Tests** (the green flag). Wait some time to gather data.   At the bottom right of the dialog box, you will see the message, “Preparing Drives.” This is expected and can continue for some time.   1. Wait a few minutes, and then close the Iometer application.   Iometer is preparing a test file. We are aborting the test file creation process.   1. Launch Iometer again and repeat steps 20–23. 2. Click **green flag** on toolbar to **Start Tests** ().   When the test completes, take note of the metrics, which should be similar to those you see below. |

***Activity 2: Connect to PREM-D2-VM and Run a Disk Performance Test***

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will connect to PREM-DS2-VM, install and configure Iometer, and then run a benchmark test. | 1. In the Azure Portal, in the left navigation, click **All resources**, and then click **PREM-DS2-VM**. 2. In the PREM-D2-VM blade, click **Connect**. 3. When prompted, click **Open**. 4. In the Remote Desktop Connection dialog box, click **Connect**. 5. In the Windows Security dialog box, click Use another account. 6. In Username, type **itcampadmin**, in Password, type **Passw0rd!**, and then click **OK**. 7. In the Remote Desktop Connection dialog box, click **Yes**. 8. If prompted to choose network preferences, click **No**. 9. Open **File Explorer**, and navigate to **C:\Source**. 10. Right-click **iometer.zip**, click **Extract all**, and then click **Extract**. 11. In C:\Source\iometer, double-click **iometer-setup.exe**. 12. On the Welcome to Iometer 1.1 Setup Wizard page, click **Next**. 13. On the License agreement page, click **I Agree**. 14. On the Choose Components page, click **Next**. 15. On the Choose Install Location page, click **Install**. 16. Click **Finish**. Click **Start**. 17. On the Start page, type **Iometer**, and then click **Iometer**. 18. Click **I Agree**. 19. Click Open Test Configuration File (the folder icon in the upper left). 20. In the Open Test Configuration File dialog box, navigate to **C:\Source\Iometer**, select **iometer.icf**, and click **Open**.   The iometer.icf file was configured earlier as part of this lab setup.   1. Click **Run** (the green flag).   At the bottom right of the dialog box, you see a message, “Preparing Drives.” This is expected and can continue for some time.   1. Wait a few minutes and then close the Iometer application.   Iometer is preparing a test file. We are aborting the test file creation process.   1. Launch **Iometer** again and repeat steps **20–23**. 2. Click **Start Tests** (green flag). 3. When the test completes, take note of the metrics, which should be similar to those you see below.      1. Note that the total IOPS and throughput are significantly higher than STD-D2-VM. Aside from the storage account type, the virtual machines have an identical configuration. |

***Activity 3: Remove the Resource Group Used for the Lab***

Because each lab in this series begins with an empty resource and because Azure resources are potentially billable, it is necessary to remove the resource group you created and used in this lab. Also, because Azure trial accounts are limited to four compute cores, it is important that you remove the resource group to ensure that you do not run out of resources, if you have an Azure trial account.

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will run a PowerShell script to remove the resource group you created and used in this lab. | * Perform the following activities on AZRCAMP-ADMIN logged on as **Contoso\Administrator** using **Passw0rd!** as the password:  1. If it is not already open, open the PowerShell ISE. 2. Click **File**, click **Open**, browse to **C:\LabFiles\AZITPROCamp\Scripts**, select **RGCleanup.ps1**, and click **Open**. 3. On the menu, click **Run**. 4. When prompted, log on to your Azure subscription. 5. When prompted to delete **RG-AZITCAMP-PREM** and **RG-AZITCAMP-STD**, click **Yes**. 6. If you used different resource groups for the lab, you can modify the PowerShell script to delete those resource groups. |

**Proceed to and start Lab 3 “Design Azure Networking for Advanced Security” in the online learning portal**.

Lab 3: Design Azure Networking for Advanced Security

|  |
| --- |
| Microsoft Azure offers a number of mechanisms for mitigating risks to virtual machines, networks, and other resources. Site-to-site and VNet-to-VNet virtual private networks (VPNs) allow data to be transmitted securely between an on-premises site and a virtual network or between virtual networks. **Network Security Groups (NSGs**) allow designers to configure network access controls at the subnet, virtual machine, or NIC level to provide an additional layer of defense. VPNs and NSGs can complement each other—for example, to provide a means of allowing management access over a secure VPN to reduce the number of ports that have to be exposed on the Internet.  In this lab, you will learn about the VPN and NSG resources that are available in Azure. In addition, you will learn how to configure and deploy these resources by using Azure Resource Manager (ARM) templates. |

#### Overview

Contoso, Inc. has asked you to continue to investigate the advantages of deploying infrastructure as a service (IaaS) using ARM. A significant benefit to this approach is the security of applications that reside in Azure. Contoso would like to see its best practices reflected in the configuration of Azure networking to help ensure the integrity and confidentiality of its data. In particular, Contoso is interested in learning how ARM templates might be used to quickly deploy an infrastructure that uses a VPN to provide connectivity to manage resources and at the same time uses NSGs to control access.

In this lab, you will analyze the JSON objects that make it possible to deploy network configurations. You will then complete a partially finished template to deploy NSGs.

#### Access the Lab Environment

For this lab, you will be accessing a hosted environment that contains all the virtual machines and resources you require.



**Exercise 1: Prepare the Azure Infrastructure**

In this exercise, you will use the **Lab03Start.ps1** script to log on to your Azure subscription and deploy the Azure infrastructure required for this set of lab exercises. The script creates a new resource group called **RG-AZITCAMP-LAB03**. The script then uses an ARM template to deploy three subnets and two virtual machines to the resource group.

The script requires your initials and a password as inputs. The unique storage account and public DNS parameters are generated by the script and passed to the ARM template at run time, along with the password. The script takes about 10 to 20 minutes to complete the deployment. When the script completes, it displays the two dynamic public IP addresses assigned to the two virtual machines.

***Activity 1: Configure a Public IP Address on the Edge Server***

| Description | Steps |
| --- | --- |
| To configure a site-to-site VPN, you require a public IP address that is connected to your VPN device. It is important to note that, while your VPN device may be behind a firewall device or load balancer of some kind, it cannot be behind a NAT (Network Address Translation) device.  In our lab environment, the RRAS server (AZRCAMP-EDGE) is behind a NAT device to preserve public IP addresses. In this exercise you will **IMPORTANT: Change the Launchpad configuration to directly connect the external interface of AZRCAMP-EDGE to the Internet and the change the configuration of the external adapter get a public IP address via DHCP.** | * Perform the following exercises on **AZRCAMP-EDGE** logged on as **Contoso\Administrator** using **Passw0rd!** as the password: * Note: If the virtual machine is not already logged on, on the menu bar, click Commands, and then click Ctrl+Alt+Delete to display the log on screen.  1. Switch to the **AZRCamp-Edge** in Launchpad. 2. In Launchpad, on the **Machines** tab, change the Internet connection to **Internet-Direct**. 3. On the desktop, right mouse click **get-public-ip.cmd**. Edit the file and add **cmd** at the end of the file and click **Save**. Double-click and open the file. Record the IP address.  * The script uses the netsh command to change the address type to DHCP. When a public IP address is acquired for the external network interface, it is displayed in the command prompt output..  1. Record the public IP address displayed in the command prompt.  * You will need to know this address in later steps. * **Public IP: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |

**Exercise 2: Run the Lab03Start.ps1 Script**

|  |  |
| --- | --- |
| **Description** | **Steps** |
| In this exercise, you will run the Lab03Start.ps1 script to deploy the Azure VM and network infrastructure for Lab 03. As an alternative to running this script, you can also deploy the infrastructure from a GitHub repository. Please see the alternate instructions below, if you wish to use this alternate method. | * Perform the following exercises on **AZRCAMP-ADMIN**:  1. Open File Explorer and navigate to **C:\LabFiles\AZRITPROCamp\Lab03 – Design Azure Networking for Advanced Security**.  * You may also download files used for this lab from the GitHub repository for the course at <https://github.com/AZITCAMP/Labfiles>.  1. Right-click **Lab03Start.ps1**, and click **Edit**.  * The Windows PowerShell ISE console opens.  1. In Windows PowerShell ISE, on the upper Ribbon, click in upper windows go to **Edit** and **Select All,** click **Run Script** (green arrow). 2. When prompted, enter a lower-case string that represents your initials, and press **ENTER**.  * Your initials are used to create unique names for the storage account and public DNS names.  1. In the Sign in to Windows Azure PowerShell dialog box, enter the email address of the account associated with your Azure subscription, and click **Continue**. 2. On the sign in page, enter your password, and click **Sign in**.  * The script starts running and then pauses to display the storage account and public DNS names that will be used for the lab.  1. Press **ENTER** to continue. 2. When prompted for the Admin Password parameter, type Passw0rd! and press **ENTER**. 3. In the command pane, at the prompt for the **localGatewayIpAddress** parameter, type the IP address of the external adapter on the AZRCAMP-EDGE server that you determined earlier, and press ENTER.  * The deployment starts. * After 10 - 20 minutes or more minutes, the infrastructure is deployed. The VPN gateway will take the longest to deploy, as long as 30 minutes. Please be patient. If you log on to <https://portal.azure.com> you can watch the progress of the deployment. * When the script completes, the public IP addresses and DSNs name are displayed as the output. * Please continue with the lab steps. You do not have to wait for the deployment to complete in its entirety to do subsequent steps.  1. Record the public IP addresses and the DNS name. 2. **Public IP:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** 3. **DNS:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_\_\_\_\_\_\_\_\_ 4. Leave the Windows PowerShell ISE console open for subsequent lab exercises. |

***(Alternate Setup Instructions) – (optional only if needed) Exercise 2a: Deploy Lab 03 Infrastructure from GitHub***

Perform these instructions only if you have NOT performed the instructions above. These instructions are provided as an alternate method for setting up the Azure lab infrastructure.

|  |  |
| --- | --- |
| **Description** | **Steps** |
| As an alternative to deploying the Azure infrastructure using a PowerShell script, you can set up the infrastructure using a template deployment directly from GitHub. | * Perform the following exercises on **AZRCAMP-ADMIN**:  1. Open Microsoft Edge, and browse to[**https://github.com/AZITCAMP/Labfiles/tree/master/lab03**](https://github.com/AZITCAMP/Labfiles/tree/master/lab03)   The lab03 repository contains all files you need to complete the lab.   1. In the Readme.md section, click **Deploy to Azure**. 2. If prompted, log on to the Azure portal.   The Parameters tab for the virtual machine custom deployment appears.   1. In the **PUBLICDNSNAME**, type a unique name, such as your initials plus a random number between 10,000 and 99,000 (e.g, abc12345). 2. In **NEWSTORAGEACCOUNTNAME**, type a unique name such as [abc]store#, where [abc] represents your initials and # is a random 3 or 4 digit number.   This name must be unique, must all be lower case, and contain only letters and numbers.   1. In **ADMINPASSWORD**, type **Passw0rd!** 2. In **LOCALGATEWAYIPADDRESS**, enter the IP address of the external interface of AZRCAMP-EDGE that you determined earlier. 3. Leave the remaining parameters at their default value, and click **OK**. 4. On the custom deployment blade, in the Resource group section, **Or create new**. 5. In the text box, type **RG-AZITCAMP-LAB03**.   You can name the resource group anything you want. However, keep in mind that the RGCleanup.ps1 script relies on the existence of a resource group(s) named \*AZITCAMP\*.   1. Click **Legal terms**. 2. In the Buy blade, click **Buy**. 3. On the Custom deployment tab, click **Create**. 4. Leave the Microsoft Edge browser open for subsequent steps. |

**Analyze the ARM Template Used to Deploy Network Related Resources**

The ARM template you use to deploy the infrastructure creates a number of network-related resources, including a virtual network, subnets, and virtual network gateway. The virtual network and the subnets can be created either by using the Azure Portal or by using an ARM template that relies on the Network REST API (<https://msdn.microsoft.com/en-us/library/azure/mt163658.aspx>).

At the time of this writing, it is not possible to use the Azure Portal to create virtual network gateways or virtual network gateway connections. This means that if you want your deployment to include a site-to-site or a point-to-site VPN and you want to use the Resource Manager deployment model to create a virtual network gateway and virtual network connection for the VPN, you must use either Azure PowerShell cmdlets or ARM templates. In either case, the PowerShell cmdlet or the ARM template that you use to create your virtual network gateway and connection will rely on the Azure Network Gateway REST API (<https://msdn.microsoft.com/en-us/library/azure/mt163859.aspx>).

In this lab exercise, you will examine how the network-related resources are deployed by using the ARM template used to create the infrastructure for Lab 03.

|  |  |
| --- | --- |
| **Description** | **Steps** |
| In this exercise, you will open a copy of the azuredeploy.json template file in Visual Studio Code. | Open azuredeploy.json template file in Visual Studio Code   * These instructions assume that you are using the provided lab environment. If you do not have the lab files stored locally, you can acquire them at <https://github.com/AZITCAMP/Labfiles/tree/master/lab03>. * Perform the following exercises on AZRCAMP-ADMIN and AZRCAMP-EDGE logged on as **Contoso\Administrator** using **Passw0rd!** as the password: * Note: If the virtual machine is not already logged on, on the menu bar, click Commands, and then click Ctrl+Alt+Delete to display the log on screen.  1. On AZRCAMP-ADMIN, open File Explorer, and navigate to **C:\LabFiles\AZITPROCamp\Lab03**. 2. Double-click **azuredeploy.json**.   The azuredeploy.json file opens in Visual Studio Code.   1. Leave Visual Studio Code open for subsequent lab steps. |

**Examine the Azure Networking Resources Deployment in the ARM Template**

Aside from the need for a local gateway—which could be another Azure network that has its own virtual network gateway—to create a virtual network gateway, you need to have a virtual network that contains at least two subnets. One of the subnets must be named **GatewaySubnet** and is used as a routing domain between the remote site (or point) and your IaaS deployment. As such, the GatewaySubnet requires that it be configured with at least a /29 Classless Inter-Domain Routing (CIDR) specification to provide a minimum number of IP addresses to create the routes.

**Virtual Network Resource**

To meet the prerequisite of the gateway subnet, it is necessary for you to use the Microsoft.Network/virtualNetworks resource in your template.

The following shows a simplified version of the Microsoft.Network/virtualNetworks resource:

For the sake of clarity, some of the elements, such as tags, dhcpOptions, and dnsServers, have been omitted. For a complete list of the elements used in the Microsoft.Network/virtualNetworks resource, please see: <https://msdn.microsoft.com/en-us/library/azure/mt163661.aspx>.

To create a subnet or subnets, you need to create a virtual network to contain the subnets. The virtual network resource requires, at a minimum, a name and location in order to create it. You may configure the resource so that it depends on other resources being present as a prerequisite. The following is a brief description of the other resource elements.

**addressSpace:** The addressSpace element contains an array of IP address ranges denoted by a list of IP addresses and prefixes.

**addressPrefix:** This element denotes the range of IP addresses in CIDR notation that can be consumed by subnets in your deployment.

**subnets:** This element contains an array of subnets. Like virtual networks, subnets have a name and an addressPrefix. If you wish to associate an NSG, you do it in this section (please see later lab exercises for an explanation of this).

In the azuredploy.json template that you opened earlier, the Microsoft.Network/virtualNetworks resource starts at about line 117:

{

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/virtualNetworks”,

“name”: “name-of-VNET”,

“location”: “location”,

“dependsOn”: [

],

“properties”: {

“addressSpace”: {

“addressPrefixes”: [

“10.0.0.0/16”

]

},

“subnets”: [

{

“name”: “subnet1”,

“properties”: {

“addressPrefix”: “10.0.0.0/24”,

}

]

}

}

The relevant variables are declared earlier in the template, as follows:

“VNETName”: “Lab03\_VNET”,

“FESubnetPrefix”: “10.0.0.0/24”,

“FESubnetName”: “FESubnet”,

“BESubnetPrefix”: “10.0.1.0/24”,

“BESubnetName”: “BESubnet”,

“GWSubnetPrefix”: “10.0.200.0/28”,

“GWSubnetName”: “GatewaySubnet”,

“vnetAddressRange”: “10.0.0.0/16”

Given the values provided for the resource, the template causes a virtual network named Lab03\_VNET to be created using a CIDR block of 10.0.0.0/16. Within the VNet, three subnets are created: FESubnet, BESubnet, and GatewaySubnet. These subnets are assigned CIDR blocks of 10.0.0.0/24, 10.0.1.0/24, and 10.0.200.0/28, respectively. Note that the location value is provided by a function: [resourceGroup().location].

**Public IP Addresses Resource**

 A virtual network gateway requires a public IP address to be assigned to it. You assign public IP addresses to resources such as virtual network gateways and virtual machines by using the Microsoft.Network/publicIPAddresses resource, which may be found here: <https://msdn.microsoft.com/en-us/library/azure/mt163590.aspx>.

In the deployment, we require three public IP addresses: one for each of the two servers and another for the virtual network connection endpoint. In the template, these resources are created immediately after the “variables” section starting at about line 70.

 {

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/publicIPAddresses”,

“name”: “[concat(variables(‘publicIPAddressName’),’0’)]”,

“location”: “[resourceGroup().location]”,

“properties”: {

“publicIPAllocationMethod”: “Dynamic”,

“dnsSettings”: {

“domainNameLabel”: “[parameters(‘publicDnsName’)]”

}

}

},

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/publicIPAddresses”,

“name”: “[concat(variables(‘publicIPAddressName’),’1’)]”,

“location”: “[resourceGroup().location]”,

“properties”: {

“publicIPAllocationMethod”: “Dynamic”

}

},

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/publicIPAddresses”,

“name”: “[variables(‘gatewayPublicIPName’)]”,

“location”: “[resourceGroup().location]”,

“properties”: {

“publicIPAllocationMethod”: “Dynamic”

}

},

In the **Microsoft.Network/publicIPAddresses**, a number of settings are required, such as the name and the **publicIPAllocationMethod**. For the **publicIPAllocationMethod** setting, only two values are possible: Static or Dynamic.

The relevant variables and parameters are defined earlier in the JSON file. The public DNS name must be unique and is defined as a parameter to allow the template user the ability to enter a unique name upon deployment.

”publicDnsName”: {

“type”: “string”,

“metadata”: {

“description”: “Unique public DNS prefix for the deployment. The fqdn will look something like ‘<dnsname>.westus.cloudapp.azure.com’. Up to 62 chars, digits or dashes, lowercase, should start with a letter: must conform to ‘^[a-z][a-z0-9-]{1,61}[a-z0-9]$’.”

}

},

The public IP address name is created by using two variables: “**publicIPAddressName**”: “**PubIP**” and “**gatewayPublicIPName**”: “**GWIP**”. The PubIP variable is used in conjunction with the concatenate function to provide the unique names PubIP0 and PubIP1. These resources are assigned to the virtual machines, FE1 and BE1, in the template.

**Network Gateway Resources**

The Azure Network Gateway resources include Local Network Gateways and Virtual Network Gateways. A complete description of the APIs that comprise these resources is found here: <https://msdn.microsoft.com/en-us/library/azure/mt163859.aspx>.

As with any resource, these resources may be created by using either PowerShell cmdlets or ARM templates.

Consider the following PowerShell commands that could be used to create the same VPN deployment in the lab environment as the template:

$rgname = RG-AZITCAMP-LAB03

$loc = “West US”

$vnet = Get-AzureVirtualNetwork -Name Lab03\_VNET `

-ResourceGroupName “$rgname”

$GWSubnet = Get-AzureVirtualNetworkSubnetConfig -Name `

‘GatewaySubnet’ -VirtualNetwork ‘$vnet’

$gwip = New-AzurePublicIpAddress -Name gwip `   
 -ResourceGroupName”$rgname” -Location “$loc” `

-AllocationMethod Dynamic

$gwipconfig = New-AzureVirtualNetworkGatewayIpConfig –Name`

vnetGatewayConfig -SubnetId $GWSubnet.Id -PublicIpAddressId $gwip.Id

New-AzureVirtualNetworkGateway -Name VNETGW -ResourceGroupName`

“$rgname” -Location “$loc” `

-IpConfigurations $gwipconfig -GatewayType vpn

New-AzureLocalNetworkGateway -Name LocalSite –ResourceGroupName`

“$rgname” -Location “$loc” -GatewayIpAddress “128.136.x.y”`

-AddressPrefix “192.168.10.0/24”

ARM resources in the template require the same settings as the PowerShell cmdlets. Examine the **New-AzureVirtualNetworkGateway** cmdlet above. To create a new Azure Virtual Network resource, we need to give the virtual network gateway a name, specify a resource group, specify a location, ensure that a public IP address is available for its use, ensure that a private IP address (dynamic or static) is allocated, specify a gateway type, and so on.

The ARM template provides identical information in a different format. Otherwise, there is no difference between the resource defined by the template and the PowerShell cmdlet, as shown below.

Note that in this resource, we use the “**dependsOn**” element to ensure that a public IP address is available.

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/virtualNetworkGateways”,

“name”: “[variables(‘gatewayName’)]”,

“location”: “[resourceGroup().location]”,

“dependsOn”: [

“[concat(‘Microsoft.Network/publicIPAddresses/’, variables(‘gatewayPublicIPName’))]”,

“[concat(‘Microsoft.Network/virtualNetworks/’, variables(‘VNetName’))]”

],

“properties”: {

“ipConfigurations”: [

{

“properties”: {

“privateIPAllocationMethod”: “Dynamic”,

“subnet”: {

“id”: “[variables(‘GWsubnet-id’)]”

},

“publicIPAddress”: {

“id”: “[resourceId(‘Microsoft.Network/publicIPAddresses’,variables(‘gatewayPublicIPName’))]”

}

},

“name”: “vnetGatewayConfig”

}

],

“gatewayType”: “Vpn”,

“vpnType”: “RouteBased”,

“enableBgp”: false

}

}

Note that, by default, the virtual network gateway is configured using the default SKU. The VPN default SKU provides up to ~80 Mbps throughput for a site-to-site VPN (~500 Mbps for ExpressRoute) and up to 10 site-to-site-tunnels. The VPN high-performance SKU can provide up to ~200 Mbps throughput for a site-to-site VPN (~1,000 Mbps for ExpressRoute) and up to 30 site-to-site tunnels. To create a VPN that uses the high-performance SKU or to modify an existing VPN that uses the default SKU, you can use the “gatewaySize” element in the virtualNetworkGateways resource to specify the SKU. Please see <https://msdn.microsoft.com/en-us/library/azure/mt130667.aspx> and <https://azure.microsoft.com/en-us/blog/azure-virtual-network-gateway-improvements/> for more information.

Likewise, the PowerShell cmdlet to create the local gateway, which refers to the settings for the on-premises or remote network, provides the same settings as the resource defined in the template. In the cmdlet, we need to specify a name for the local gateway, the resource group, the location, the IP address of the endpoint connection, and an address range for the local site.

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/localNetworkGateways”,

“name”: “[variables(‘localGatewayName’)]”,

“location”: “[resourceGroup().location]”,

“properties”: {

“localNetworkAddressSpace”: {

“addressPrefixes”: [

“[variables(‘localGatewayAddressPrefix’)]”

]

},

“gatewayIpAddress”: “[parameters(‘localGatewayIpAddress’)]”

}

}

**Complete Site-To-Site VPN Configuration**

Site-to-site VPNs are used to create secure connections between Azure sites, whether on-premises locations or other virtual networks (more typically known as VNet-to-VNet). Point-to-site VPNs are used to create a secure connection between an Azure VNet and a local computer when it is not possible or desirable to use a VPN device to create the secure connection. Site-to-site, VNet-to-VNet, and point-to-site connections all require that you configure a virtual network gateway in Azure.

To configure a virtual network gateway in Azure, you need to determine the Gateway SKU (Basic, Standard, or High Performance) you want to use. As you learned in the previous lab exercise, the VPN default (standard) SKU provides up to ~80 Mbps throughput for a site-to-site VPN (~500 Mbps for ExpressRoute) and up to 10 site-to-site-tunnels. The VPN high-performance SKU can provide up to ~200 Mbps throughput for a site-to-site VPN (~1,000 Mbps for ExpressRoute) and up to 30 site-to-site tunnels. In the lab environment, the virtual network gateway is created using the standard (default) SKU.

As part of your virtual network gateway configuration, you need to specify the gateway types, either static routing (aka, policy-based VPN) or dynamic routing (aka route-based VPN). The setting you use in the virtual network gateway configuration is determined by the VPN device used on the other end of the VPN connection, your on-premises VPN device. For example, RRAS in Windows Server 2012 R2 supports only dynamic routing. Furthermore, not all configurations are possible using static (policy-based) routing. For example, point-to-site or multi-site configurations are not possible using static routing.

Once you have made your choices with regard to Gateway SKU and gateway type and have configured the virtual network gateway, you need to configure your VPN device. In the classic mode, after configuring the VPN gateway, you could download a configuration file to assist you with the configuration on the local VPN device. However, at the time of this writing, the Portal does not provide a link to download the configuration file. Therefore, to configure the VPN, you will need to follow device-specific instructions that are available here: <https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/>.

In the following lab exercise, you will complete the configuration of the VPN connection settings and the local VPN device to establish a secure connection between the on-premises network (192.168.10.0/24) and the Azure VNet (10.0.0.0/16).

**Configure the Local VPN Device for the Site-to-Site VPN Connection**

Once the VPN gateway has finished provisioning, you can configure the local on-premises VPN device to connect to the Azure VPN gateway.

Please note that, although these steps assume you are using the provided lab environment, these steps will work for any Windows Server 2012 R2 server configured with RRAS, as long as it is not behind a NAT device. If you are not using the provided lab environment and are using some other VPN device, you will need to follow instructions specific for your device. You can find these instructions at <https://azure.microsoft.com/en-us/documentation/articles/vpn-gateway-about-vpn-devices/>.

|  |  |
| --- | --- |
| **Description** | **Steps** |
| In this exercise, you will configure AZRCAMP-EDGE server to connect to the Azure VPN gateway.  To configure a site-to-site VPN, you require a public IP address that is connected to your VPN device. It is important to note that, while your VPN device may be behind a firewall device or load balancer of some kind, it cannot be behind a NAT (Network Address Translation) device. | * These instructions assume that the template deployment you initiated earlier has successfully completed deploying. If the template has not finished deploying, please wait until it has done so. * Perform the following exercises on **AZRCAMP-ADMIN** and **AZRCAMP-EDGE** logged on as **Contoso\Administrator** using **Passw0rd!** as the password: * Note: If the virtual machine is not already logged on, on the menu bar, click Commands, and then click Ctrl+Alt+Delete to display the log on screen.  1. On AZRCAMP-ADMIN, in the Windows ISE PowerShell console you left open in the previous exercise, type the following command and press ENTER.   Get-AzureRMPublicIpAddress -Name gwip -ResourceGroupName RG-AZITCAMP-LAB03 | select name, IPAddress  The output of the command displays the public IP address of the VPN gateway.   1. Record the IP address for use in later steps. 2. In Windows PowerShell ISE console, click **File** and then click **Open**. 3. In the Open dialog box, navigate to **C:\LabFiles\AZITPROCAMP\Lab03**. 4. Click **AZVPNConnectionConfig.ps1**, and click **Open**.   The AZVPNConnectionConfig.ps1 script runs the New-AzureVirtualNetworkGatewayConnection cmdlet to configure the connection in Azure between the remote site VPN endpoint and the VPN gateway in Azure. In subsequent steps, you will configure the on premises VPN device (RRAS) and initiate the VPN connection.   1. On the ribbon, click **Run Script**.   After a few moments, the output of the command returns with something similar to the following:    Leave the Windows PowerShell ISE console open for subsequent steps.   1. Switch to the **AZRCamp-Edge** in Launchpad. 2. On the activitybar, right-click **Windows PowerShell**, and click **Window PowerShell ISE**. 3. In Windows PowerShell ISE console, click **File** and then click **Open**. 4. In the Open dialog box, navigate to **C:\Scripts\**. 5. Click **RRASConfig,ps1**, and click **Open**. 6. In the script, locate the two instances of **<SP\_AzureGatewayIpAddress>**. 8. Replace both instances of **<SP\_AzureGatewayIpAddress>** with the IP address you determined in step 1 above. 9. On the ribbon click **Run Script**. 10. Click **OK**. 11. In the output of the command, you may see an error, such as the following. This is expected. RRAS may not be reporting its status accurately.      1. Open **Server Manager**. 2. In Server Manager, click **Tools**, and then click **Routing and Remote Access**. 3. In Routing and Remote Access console, in the tree pane, expand **Edge**, and then click **Network Interfaces**.   You should see the network interface that is represented by the IP Address of the VPN Gateway. If you do not see the interface, return to the Windows PowerShell ISE console, run the command **restart-service RemoteAccess**, and then rerun the **RRASConfig.ps1** script.   1. Right-click the connection, and click **Connect**.   In a few moments, the connection should be established     1. Switch to AZRCAMP-ADMIN.   In these next steps, you will verify that the VPN site-to-site connection is functioning as intended.   1. In the Windows PowerShell ISE console, in the command pane, type the following command, and press ENTER.   (Get-AzureRMNetworkInterface).IpConfigurations.PrivateIpAddress  The output of the command shows the two private IP address that are assigned to the virtual machines BE1 and FE1 in the lab environment. The IP addresses will likely be 10.0.1.4 and 10.0.0.4, as .4 is the first address in the octet assigned to the first virtual machines that are provisioned in a subnet.     1. Open **File Explorer**, and navigate to **C:\LabFiles\Utils\PortQryUI**. 2. Double-click **portqueryui.exe**. 3. In Enter destination IP or FQDN to query, type **10.0.0.4** (or the actual private IP address of FE1, if different). 4. Click **Manually input query ports**. 5. In Ports to Query, type **23, 80, 3389**, and click **Query**.   The Port Query application almost instantly displays the output showing the status of the 3 ports on FE1 as LISTENING. This means that the ports are open. Had any of the ports been blocked, the output would have taken a few moments to be displayed and the status for the blocked ports would be denoted by FILTERED  Machine generated alternative text: Pod Query  File Help  Enter destinabal IP or FQC'-I to que  Query  Query credefrned service:  to Dzmans Trusts  ports.  Ports to .•ry  Enter pct rum*r and,"' port ran— sep-rated by commas.  Pro :  For  Result:  Cancel  starblg -n 10.0.0.4. 0 TCP  Queryirg target system calkd:  10.0.0.4  Attempting to resdve IP ö*ress to a name.  address to FEL  sort 23  sort 30 service): LISTENING  3389 service) : LISTENING  -p TCP exits •.return ade 010033000.   1. **Repeat steps 25-28** to query BE1 at its private IP address 10.0.1.4 to see if the ports 23 and 3389 are accessible through the VPN connection.   IIS is not installed on BE1.    Machine generated alternative text: 酗 pod Query  n - R  addressredved L  一 t 名 5 IP 甘 緲 「  Queryi-stargetsystem d …  51 一 1 •nLO.O.L4 13 9 TCP :  .e 411PP1- 2 3389 TCP ) ; 」 O 0 〔 「 〕 00.  「 【 3 9 ( t se 一 “ LISTB 」  23( servic = 5 司 一 G  FM n 「 3P5 1324 亠 35D  「 ; ft 「 t 「 5 紉 「 or ,  to 「  一 4 'y 一 「 query 名 一  5 ♂ r 、  23 9   * Leave PortQuery open for subsequent lab steps. |

**Configure NSGs by Using ARM Templates**

NSGs are a recent feature of Azure that allow you to configure firewall policies at the network level to control the flow of inbound and outbound traffic to and from virtual machine instances. An NSG contains access control rules, similar to firewall rules that allow or deny access to virtual machines in your subscription. Unlike network ACLs, which can only be applied to the public endpoints, an NSG can be applied to virtual machines, NICs, and subnets. This means that all instances of virtual machines that reside in a subnet can have the same access control rules applied to them and that each virtual machine can have access control rules specifically tailored for them. If a virtual machine has multiple NICs, and an NSG is applied to only one of the NICs, the other NICs are not subject to the access control rules of the NSG.

Only one NSG can be associated with each subnet, virtual machine, or NIC. However, NSGs can contain up to 200 rules. NSG rules have the following characteristics:

NSGs contain rules that consist of the following properties:

**Name:** Unique identifier for the rule

**Type:** Inbound/Outbound

**Priority:** An integer between 0 and 4096 that determines the order of processing; lower-priority numbers are processed before higher-priority numbers

**Source IP Address:** CIDR of source IP range

**Source Port Range:** Range between 0 and 65500

**Destination IP Range:** CIDR of destination IP range

**Protocol**: TCP, UDP or \* for all

**Access:** Allow/Deny

Rules are stateful. This means that NSGs keep track of the communication. When you create an inbound rule, it is not necessary to configure a corresponding outbound rule and vice versa to enable bidirectional communication between the local and remote entity. For example, when you configure an inbound rule to provide inbound access to a web server on TCP port 80, you do not have to create a corresponding outbound rule to allow the web server to respond to the request on port 80.

Rules are processed according to priority value, which is an integer between 100 and 4096. The lower the number, the higher the priority (order of processing). When a traffic match is made by a higher-priority rule, processing of rules stops for that particular traffic.

When you implement an NSG on a subnet, virtual machine, or NIC, a set of default rules is also implemented. The default rule can be overridden by rules that have a lower-priority value. The default rules are as follows:

**Allow VNET Inbound:** Allows inbound traffic from all hosts in the VNet. Priority = 65,000.

**Allow Azure Load Balancer Inbound:** Allows the Azure load balancer to probe the health of the virtual machine. Priority = 65,001.

**Deny All Inbound:** Denies all inbound traffic. Priority = 65,500.

**Allow All VNET Outbound:** Allows outbound traffic from all hosts in the VNet to all hosts in the VNet. Priority = 65,000.

**Allow Internet Outbound:** Allows all outbound traffic for the Internet outbound. Priority = 65,001

**Deny All Outbound:** Denies all outbound traffic. Priority = 65,500.

Some other important characteristics of NSGs include the following:

Default tags and special characters are used to identify a category of IP addresses. The default tags are:

**VIRTUAL\_NETWORK:** Denotes the entire address space of the virtual network

**AZURE\_LOADBALANCER:** Denotes Azure’s load balancer

**INTERNET:** Denotes the Internet

**\***: Special character to indicate “all”

**NSGs** are available only for virtual machines in a regional VNet: they are not available for VNets associated with affinity groups.

**Endpoint-based ACLs** and **NSGs** are not compatible for the same virtual machine. You must use either ACLs or NSGs, not both.

You can have up to 100 NSGs per subscription.

**Example: Defense in Depth Using NSGs**

The flexibility and scope provided by NSGs enable defense-in-depth scenarios, where NSGs at the network level can be combined with firewall rules at the operating system level. Because NSGs can control traffic at the network level between subnets, they can be used to realize secure multi-tier application architectures. In these architectures, typically only one tier of the application is exposed to the Internet in a perimeter network (also known as a DMZ, demilitarized zone, and screen subnet). The remaining tiers of the application are segmented from the perimeter network by firewalls. Generally, following the principle of least privilege, no hosts in any of the tiers would be allowed to initiate communication with hosts on the Internet. Furthermore, only the minimum traffic necessary for the application is allowed to traverse the firewalls that segment the application tiers from each other.

In the case of Azure IaaS deployment, exceptions would be required for the Key Management Service (KMS) license server and any platform as a service (PaaS) services hosted in Azure, such as a SQL Server database. For KMS, the Azure virtual machines would need to be able to communicate with the Internet on TCP port 1688.

Consider a three-tier application that comprises a web front end, a business logic middle tier, and a back-end data tier. In this case, the only traffic allowed from the Internet would be web traffic on TCP ports 80 and 443. The web servers in the perimeter network would not be allowed to initiate traffic to the Internet. The web servers in the perimeter network would be able to communicate across a firewall to the middle tier, perhaps on well-known ports or on custom ports. The servers in the middle tier would be allowed to initiate communication with the perimeter network or the Internet, but would be allowed to communicate with the data tier. The servers in the data tier would not be able to initiate communicate to the host on the Internet or the other tiers of the applications.

**ARM Template Examples for NSGs**

To create a Network Security Gateway resource, we need to use the **Microsoft.Network/networkSecurityGroups** resource, which can be found here: <https://msdn.microsoft.com/en-us/library/azure/mt163615.aspx>.

The following shows a somewhat abbreviated version of the elements that comprise the NSG resource:

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/networkSecurityGroups”,

“name”: “NSG\_Name”,

“location”: “NSG\_Location”,

“properties”: {

“securityRules”: [

{

“name”: “rule-name”,

“properties”: {

“description”: “rule-description”,

“protocol”: “Tcp,UDP, or \*”,

“sourcePortRange”: “source-port-range”,

“destinationPortRange”: “destination-port-range”,

“sourceAddressPrefix”: “CIDR-of-source-IP-range”,

“destinationAddressPrefix”: “destination-CIDR-range”,

“access”: “Allow or Deny”,

“priority”: integer-value-for-rule-priority,

“direction”: “Inbound or Outbound”

}

}

]

}

}

It is possible to create an NSG without associating it with a subnet. However, we would typically want to associate the NSG with a subnet (or, using classic mode, a virtual machine, or NIC). To associate a rule with a subnet, we need to use the **Microsoft.Network/virtualNetworks** resource, which can be found here: <https://msdn.microsoft.com/en-us/library/azure/mt163650.aspx>.

For clarity, the following shows an abbreviated version of the elements in the virtualNetworks resource that enable the association of an NSG with a subnet:

{

“apiVersion”: “2015-05-01-preview”,

“type”: “Microsoft.Network/virtualNetworks”,

“name”: “name-of-VNET”,

“location”: “location”,

“dependsOn”: [

],

“properties”: {

“addressSpace”: {

“addressPrefixes”: [

“10.0.0.0/16”

]

},

“subnets”: [

{

“name”: “subnet1”,

“properties”: {

“addressPrefix”: “10.0.0.0/24”,

“networkSecurityGroup”: {

“id”: .../networkSecurityGroups/NSG-name”

}

}

}

]

}

}

**More Information**

For more information, please see the following blog posts and articles:

* <http://azure.microsoft.com/blog/2014/11/04/network-security-groups/>
* <https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-nsg/>
* <http://blog.kloud.com.au/2014/11/07/secure-azure-virtual-network-and-create-dmz-on-azure-vnet-using-network-security-groups-nsg/>

**Verify Connectivity to Virtual Machines**

|  |  |
| --- | --- |
| **Description** | **Steps** |
| In the previous lab exercise, you verified connectivity. In this lab exercise, you will verify the connectivity to FE1 and BE1 to establish a baseline for determining the effects of associating Network Security Groups with subnets in the subsequent lab exercise. | * Perform the following exercise on AZRCAMP-ADMIN logged on as **Contoso\Administrator** using **Passw0rd!** as the password:  1. In the Windows PowerShell ISE console that you left open in the previous exercise, in the command pane, type the following command on a single line, and press ENTER.   Get-AzurePublicIpAddress | where {$\_.Name -like "Pub\*"} | Select Name, IPaddress  The output of the command shows the two public IP address that are assigned to the virtual machines FE1 and BE1 in the lab environment. PupIP0 corresponds to FE1; PubIP1 corresponds to BE1.   1. Record both IP addresses that are displayed in the output. 2. Open File Explorer, if not already open, and navigate to **C:\LabFiles\Utils\PortQryUI**. 3. Double-click **portqueryui.exe**. 4. In Enter destination IP or FQDN to query, type ***[PubIP0]*** (where PubIP0 is the public IP address of FE0). 5. Click **Manually input query ports**. 6. In Ports to Query, type **23, 80, 3389**, and click **Query**.   The Port Query application almost instantly displays the output showing the status of the 3 ports on FE1 as LISTENING. This means that the ports are open. Had any of the ports been blocked, the output would have taken a few moments to be displayed and the status for the blocked ports would be denoted by FILTERED.     1. **Repeat steps 4-7** to query BE1 at its public IP address to verify that ports 23 and 3389 on BE1 are accessible from the Internet.   IIS is not installed on BE1.  Machine generated alternative text: Q-eryResult:  、 .e 104-4 215 3 2 3 9 T 「 Pe  T 「 【 3 9( t - 5 ! serv ) “ LISTB 」  23()é nc = 5 司  addressredved toSEL  一 t 名 「 IP 甘 緲 「  Queryi-s targetsystem d …  51 一 1 •n 1 旵 ,  Marualynwtqueyports:  Pod Query  FM 31 一 「 1324 亠 35D  「 ; t 「 55 紉 「 or ,  P to  5 ♂ 一 Do-nails—adTrusts  23 9  13 TCP -   1. Click **Manually input query ports**. |

**Deploy NSGs by Using an ARM Template**

In this lab exercise, you will modify an incomplete starter template that contains an incomplete set of rules for two NSGs. This lab exercise does not provide you with detailed step-by-step instructions for completing the exercise. Rather, the lab exercise asks you to create a template to deploy NSGs that meets specific design goals in the scenario provided below.

***Lab Scenario Goals***

You have deployed a two-tiered Azure IaaS application. The front-end tier consists of a subnet (10.0.0.0/24) and single web server. The back-end tier consists of a subnet (10.0.1.0/24) and a virtual machine that runs an application on TCP port 23. (In fact, this is the port for the Telnet server; in the lab environment, we use this port to stand in for a database application so that we can deploy the virtual machines as quickly as possible.) You have a VPN connection between your on-premises environment and the virtual network (10.0.0.0/16).

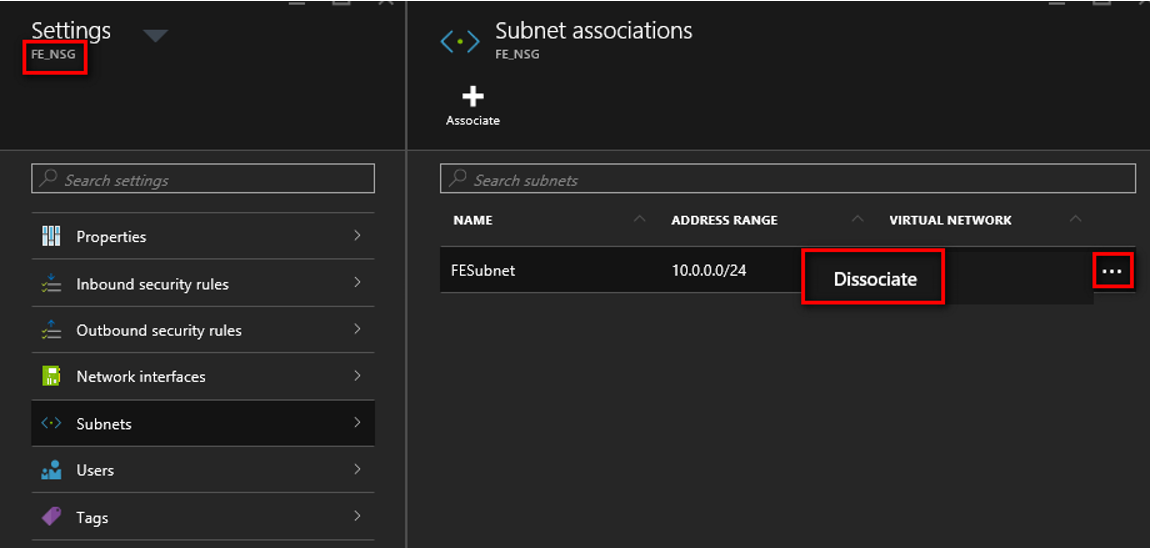
|  |  |
| --- | --- |
| **Description** | **Steps** |
| You need to ensure the following controls are enforced with your network security groups: | 1. Clients on the Internet can access the Web server in the frontend subnet using it public IP address. 2. All other Inbound traffic to both the front-end the back-end subnets is denied. 3. The virtual machines in the frontend (10.0.0.4/24) and the backend (10.0.1.0/24) subnets must be allowed access to the KMS server, which resides on the Internet at TCP port 1688. 4. All other outbound Internet access from the backend and frontend servers is denied. 5. All outbound access from the backend subnet to the frontend subnet is denied. 6. TCP port 23 must be accessible from the frontend subnet to the backend subnet; all other outbound traffic from the frontend to the backend subnet is denied. 7. RDP access on TCP port 3389 from the on-premise subnet (192.168.10.0/24) must be allowed ingress to the frontend and backend networks through the site-to-site VPN for management purposes. 8. All other traffic from 192.168.10.0/24 to the frontend and backend subnets through the site-to-site VPN must be explicitly denied. 9. Optionally, create an exception to allow traffic on TCP port 80 from the on-premises subnet to FE1 through the site-site VPN. 10. Because NSGs can have unpredictable effects on VPNS, no NSGs should be applied to the gateway subnet (10.0.200.0/28). |

|  |  |
| --- | --- |
| **Description** | **Steps** |
| * Perform the following exercises on AZRCAMP-ADMIN logged on as **Contoso\Administrator** using **Passw0rd!** as the password: | 1. Open File Explorer, if not already open. 2. Navigate to **C:\LabFiles\AZITPROCamp\Lab03**. 3. Double-click **start.azuredeploy.nsg.json** to open the file in Visual Studio Code. 4. Start.azuredeploy.nsg.json represents an incomplete deployment of Network Security Group rules to meet the criteria specified above. It can, however, be deployed as is.  * Using the criteria above, add the appropriate Network Security Group rules to the start.deploy.nsg.json file. * You do not have to edit any existing resources, parameters, or variables. You only have to add rules to the backend and frontend subnet NSGs. * If you get stuck or just want to peek quickly at a possible solution for more guidance, please see the azuredeploy.nsg.json file in the **C:\LabFiles\AZITPRO\Camp\Solutions\Lab03** folder.  1. When you have completed the modifications to the start.azuredeploy.nsg.json, save it as **azuredeploy.nsg.json**.  * Before saving the file, make sure that the file does not display any red squiggles that indicate a problem with the file format.  1. In the azuredeploy.nsg.json file, press CTRL+A, and then press CTRL+C to copy the entire contents of the file to the clipboard. 2. Leave the azuredeploy.nsg.json open for subsequent steps. 3. Open Edge browser, browse to[**https://portal.azure.com**](https://portal.azure.com), and log on to your Azure subscription. 4. Click **New**, and then click **Template deployment**. 5. If you do not see Template deployment in the New blade, search for it in the Marketplace. 6. On the Custom deployment blade, click **Edit template**. 7. In the Edit template blade, select all the lines of JSON code and delete them. 8. Press CTRL+V to past the contents of the clipboard (the azuredeploy.nsg.json) file to template area.  * When you have pasted the file, note the small green square in the upper right. This indicates that the file has the correct format. * If you do not see the green square, it is likely you have missed a comma, inserted a comma in a wrong place, have missing braces, etc. You will see a red square indicating the approximate location of the issue. Please resolve the problem before proceeding.      1. Click **Save**. 2. In the Custom deployment blade, click **Edit parameters** to review the parameter values, and then click **OK**. 3. Click **Select a resource group**, and click **RG-AZITCAMP-LAB03**. 4. Click **Legal terms**, and then click **Buy**. 5. In the Custom deployment blade, click **Create**. |

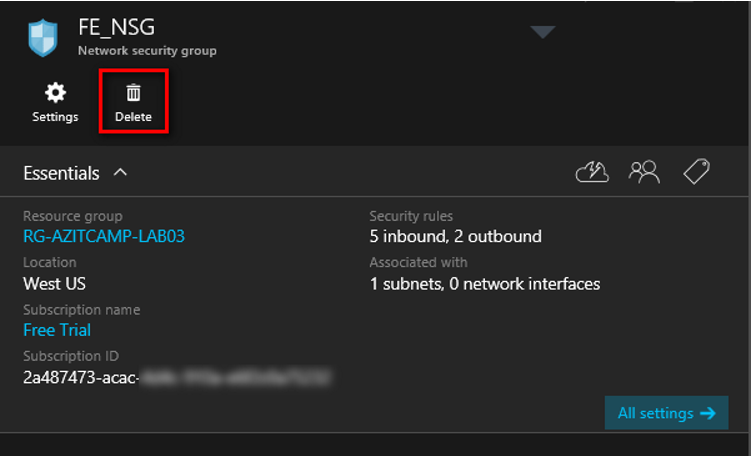
**Troubleshooting Template Deployment (If Necessary)**

If your template fails to deploy properly, please examine the error message that accompanies the failure notification. Common reasons for the failure are typically related to copying and pasting rules to create new ones from the copies. Please ensure that you use unique names and unique priority values for the rules you create.

iIf your NSG is associated with a subnet, you will not be able to delete it. You must first disassociate the NSG from the subnet. You can do so using a PowerShell cmdlet, or you can use the Portal as shown in the screenshot below.



Once you have disassociated the NSG from the subnet, you can delete the NSG resource, as shown below.



**Verify NSG Rules**

|  |  |
| --- | --- |
| **Description** | **Steps** |
| In the following exercise, you will verify that your NSG rules are properly implemented. | * Perform the following exercises on **AZRCAMP-ADMIN**:  1. Using the **PortQuery** application that you left open in a previous step, verify that the only port that is accessible from the Internet for either FE1 or BE1 is port 80 on FE1. 2. You can also use Microsoft Edge to verify that IIS is available at the public IP address for FE1.   Machine generated alternative text: Q-eryResult:  F 「 re ♂ name  一 t 名 「 IP 甘 緲 「  Queryi-s targetsystem d …  51 一 1 * 一  Marualynwtqueyports:  Pod Query  一 ♂ 11 1 .155 21 一 SO 一 3 9 TCP 」 ; b 」 一 n code 0 0H01.  ; 9 ( ms ~ ) “ F 1 一  【 30 5 •LIS 」 一  FM 31 一 「 1324 亠 35D  「 ; t 「 55 紉 「 or ,  to 「  5 ♂ 一 Do-nails—adTrusts  23 一 80 一 3389  一 一 'V 1 9 TCP :     1. Using PortQuery, verify that only TCP port 3389 is available for FE1 and BE1 through the site-to-site VPN.  * If you also created the option rule to allow TCP port 80 from the on premise subnet to the frontend subnet, TCP port 80 will also be available.  1. **On AZRCAMP-ADMIN,** open a PowerShell or a command prompt, if you do not have one of these open already. 2. At the prompt, type **mstsc**, and press **ENTER**. 3. In the Remote Desktop Connection dialog box, type **10.0.0.4** (or the actual private IP address of FE1, if different), and click **Connect**. 4. In the Windows Security dialog box, click **Use another account**. 5. In User name, enter **ITCampAdmin**; in Password, enter **Passw0rd!**, and click **OK**. 6. In the Remote Desktop Connection dialog box, click **Yes**. 7. When you have logged on to FE1, open **File Explorer**, and navigate **C:\Source**. 8. Right-click **PortQryUI**, and click **Extract All**. 9. On the Select a Destination and Extract Files page, shorten the path to **C:\Source**, and click **Extract**. 10. Open the **PortQryUI** folder, and double-click **portqueryui.exe**. 11. In Enter destination IP or FQDN to query, type **10.0.1.4** (or the actual private IP address of BE1, if different). 12. Click **Manually input query ports**. 13. In Ports to Query, type **23, 3389**, and click **Query**.  * The output should show that Port 23 on BE1 is open (LISTENING) and that port 3389 is closed (FILTERED).   Machine generated alternative text: port Que  File Help  Query  Query credefrned service:  Ser-vice to Dcmans yd Trusts  que ports  Ports to ..ry:  23,3389  Enter port rum*r and,"' port ran— sep-rated by commas.  For exa-npk:  Vy Result:  startng -n 10.0. L.4.e 23,333 -p „  Q.eryir-g target system calkd:  10.0.1.4  Attewbng to IP to a  address redved to EEL  TCP sort 23  TCP 3389 service): FILTEE  prtqry.exe 4110.0.1.4 .e 23, 3389 .pTCP exits return cade oxooocom.     1. Repeat steps 6 - 16 to establish a Remote Desktop session with BE1 (10.0.1.4) and to query ports 23 and 3389 on FE1.  * All ports on FE1 should be unavailable to BE1. |

**Remove Resource Group Used for lab**

Because each lab in this series begins with an empty resource and because Azure resources are potentially billable, it is necessary to remove the resource group you created and used in this lab. Also, because Azure trial accounts are limited to four compute cores, it is important that you remove the resource group to ensure that you do not run out of resources, if you have an Azure trial account.

**Remove Azure Resource Group**

|  |  |
| --- | --- |
| **Description** | **Steps** |
| In this exercise you will run a Windows PowerShell script to remove the resource group you created and used in this lab. | * Perform the following exercises on **AZRCAMP-ADMIN** logged on as **Contoso\Administrator** using **Passw0rd!** as the password:  1. If not already open, open Windows PowerShell ISE. 2. Click **File**, click **Open**, browse to **C:\LabFiles\AZITPROCamp\Scripts**, select **RGCleanup.ps1**, and click **Edit**. 3. On the menu, click **Run Script**. 4. When prompted, log into your Azure subscription. 5. When prompted to delete **RG-AZITCAMP-LAB03**, click **Yes**. 6. If you used a different resource group for the lab, you can modify the PowerShell script to delete that resource group. |

**Proceed to Lab 4 “Designing Identity Solutions with Azure Directory”**

## Lab 4: Designing Identity Solutions with Azure Active Directory

|  |
| --- |
| This lab is intended for IT professionals who need to design and manage Azure Active Directory solutions. In this lab, you will learn how to use Azure Active Directory Connect to integrate Azure Active Directory with your on-premises Active Directory, use Azure Active Directory to authenticate with software as a service (SaaS) services, and discover which SaaS services are being used. |

#### Overview

Your Active Directory is one of your most valuable assets when it comes to the IT services you provide to your users. Are you leveraging it to its utmost potential? With your users signing up for third-party cloud-based SaaS services, what sort of control and visibility do you have to ensure that you can help them access their data securely? Learn how you can sync select user identities with Azure Active Directory to replace one-off logons by third-party providers in a secure and scalable fashion once you design an updated identity solution with Azure Active Directory.

In this lab, you will learn how to install and configure the Azure Active Directory Connect tool, as well as use your Azure Active Directory service to enable single sign-on (SSO) with third-party providers. Finally, you will learn how to use Cloud App Discovery to identify which third-party SaaS solutions are being used.

**Before You Begin**

In this lab, you will create Azure Active Directory Connect synchronization, SSO for third-party applications, and Cloud App Discovery. This lab is self-contained, but there are dependencies between the activities and exercises. Due to the nature of the lab environment, please ensure that you have enough time to complete the lab in a single sitting. The directory synchronization will be broken when the lab environment is shut down.

**IMPORTANT: Additionally, because you are working with directory services, rather than Microsoft Azure resources, these objects are not encapsulated as parts of resource groups. As a result, there is no cleanup script to undo the directory work you have done. Consequently, you might want to ensure that you use a trial account, rather than your own personal account.**

#### Introduction

Contoso has been using Azure primarily to provide infrastructure as a service (IaaS) and platform as a service (PaaS) services. It has not been leveraging any security and identity services, such as Azure Active Directory, that are available in Azure.

One of the management challenges that Contoso is facing is caused by its increasing reliance on applications that are built, hosted, and maintained by third-party vendors, commonly known as SaaS applications. For example, Contoso recently migrated from an on-premises customer relationship management (CRM) solution to a cloud-based solution. This is just one example among many SaaS applications that Contoso is increasingly reliant on.

These SaaS applications have provided significant benefits to Contoso in the form of reduced costs and greater agility in managing public relations through social media. However, there has been a management cost. Managing account passwords for sanctioned SaaS applications is proving difficult. Additionally, there is an increasing and unauthorized use of consumer SaaS applications. The extent of this use is unknown at the present time.

Management has asked you to explore how Azure Active Directory might benefit the organization by providing SSO capabilities for SaaS applications. Additionally, you have a mandate to explore how Cloud App Discovery, which is part of Azure Active Directory Premium, can assist Contoso in discovering consumer cloud applications that are currently in use and that can be potentially integrated with Azure Active Directory SSO.

### Exercise 1: Installing and Configuring Azure Active Directory Connect

Azure Active Directory provides the heart of your authentication and identity services in Azure, just as Active Directory does for your on-premises infrastructure. Azure Active Directory Connect allows you to quickly and easily leverage your existing Active Directory investment by synchronizing users and groups from your on-premises Active Directory to Azure Active Directory.

In this lab, you will download and install Azure Active Directory Connect, configure it with the default configuration—which includes password synchronization—and test and verify synchronization.

#### Activity 1: Create a DirSync Account in Azure Active Directory

| **Description** | **Steps** |
| --- | --- |
| Azure Active Directory Connect requires an Azure Active Directory account with the Global Admin role to enable the synchronization. You should not use the default Global Admin account—that is, the Microsoft ID you used to create the Azure subscription. In this activity, you will create a Global Admin account to be used for Azure Active Directory Connect directory synchronization. | * Begin this activity logged on to AZRCamp-Admin*,* as Contoso\Administrator with a password of Passw0rd!  1. Open the Azure Portal (<https://portal.azure.com>), and log on to your Azure subscription. 2. The new portal will open, go to the bottom of the list and click **Browse** 3. Click on **Active Directory** This will open the legacy Azure Portal. At this time, Azure Active Directory can only be managed in this portal. 4. In the navigation bar of the Azure Portal, scroll down until “Active Directory” appears. 5. Click ACTIVE DIRECTORY*.* 6. In the Active Directory pane, clickDefault Directory*.* 7. Click theUSERSlink. At the bottom of page 8. Click ADD USER*.* 9. In the “Tell us about this user” screen, in the “new user in your organization”**USER NAME**box, typeDirSync*.* 10. Record the entire username that will be created.  * You will need this information in later activities. It should be in the format of **<user>@<yoursubscription>.onmicrosoft.com.**   Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   1. Click Next (the right arrow in the lower right corner of the page). 2. Complete the “user profile” screen with the following information, then click Next.   **First Name:** Directory **Last Name:** Sync **Display Name:** DirSync **Role:** Global Admin **Alternate Email Address:** <Use an email address you can access>  **Do not check** Multi-Factor Authentication   1. Click **Next** 2. On the “Get temporary password” page, click create. 3. **Record password**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 4. click Complete (click check mark). 5. In your Edge Browser, click on the upper right hand corner for the ellipses **…**  and click “**New InPrivate window**” in the web browser. 6. Log on to the Azure Portal using the DirSync account (**DirSync@<directoryname>.onmicrosoft.com**) and temporary password you just created. 7. When prompted to update your password, create and confirm a new password of **Passw0rd!** 8. Click Update password and sign in.  * It might state “No Subscription Found”, procedd. * You have now configured an account that will be used for directory synchronization.  1. Log out of the Azure Portal that is running in the InPrivate window. |

#### Activity 2: Install Azure Active Directory Connect

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will download the Azure Active Directory Connect tool from the Azure Portal, and then you will do an express installation of the tool on a dedicated synchronization server. The default installation will automatically do an initial directory synchronization. | 1. Go back to Active Directory   If required, open the Azure Portal (<https://portal.azure.com>), and log on to your Azure subscription.   1. Click Azure Portal scroll down to the bottom of the new portal click **Active Directory**.   This will open the legacy Azure Portal. At this time, Azure Active Directory can only be managed in this portal.   1. In the navigation bar of the Azure Portal, scroll down to the bottom of the portal until **“Active Directory”** appears. 2. Click ACTIVE DIRECTORY. 3. In the Active Directory pane, click Default Directory. 4. Click your directory on the **Active Directory** main list and then click the Quick Start link to the left of the Users link. 5. In the “**GET STARTED**” section of the Default Directory page, and on #2 “Integrate your local directory” click Download Azure AD Connect.  * This will open a download page in a new tab of your web browser.  1. Click Download, and wait for the download to complete to the Downloads folder. 2. Click View downloads. 3. In the **DOWNLOADS** pane, click Open folder.  * The File Explorer icon in the activitybar will flash.  1. Click on the program and confirm that AzureADConnect.msi is in the Downloads folder. 2. Copy and paste **AzureADConnect.msi** to C:\.  * This will make it easier to find in later steps.  1. If prompted to provide administrator permission, click Continue.  * Perform the following steps logged on to AZRCamp-Sync, as Contoso\Administrator with a password of Passw0rd!  1. Open **File Explorer** go to the top of the network icon bar and navigate to [**\\Admin\c$**](file:///\\Admin\c$) and click ENTER      1. **Copy and paste** the **AzureADConnect.msi** file to C:\.      1. Double-click AzureADConnect.msi. 2. When prompted with a security warning, click Run. 3. The **Microsoft Azure Active Directory Connect** wizard will start.  * It might open behind the current active window.  1. Minimize the File Explorer window. 2. On the **Welcome to Azure AD Connect** page, agree to the license terms and privacy notice, and then click Continue. 3. On the Express Settings page, review what the wizard will do, and then click Use express settings. 4. On the **Connect to Azure AD** page, enter ***username and password*** *(Passw0rd!)*of the *DirSync account* you created in the previous activity, then click Next. 5. On the Connect to Active Directory Domain Services (AD DS) page, enter the following credentials, then click Next. Username: Contoso\Administrator Password: Passw0rd! 6. Review the information on the **“Ready to configure”** page. 7. Confirm that there is a check mark in the box to start synchronization, click Install. 8. It might take up to 5 minutes for the configuration to complete. When it does, review the information on the page, and then click Exit. 9. Switch to ***AZRCamp-Admin*** as***Contoso\Administrator*** with a password of ***Passw0rd!*** |

### Exercise 2: Confirm Azure Active Directory Connect Synchronization

#### Activity 1: Confirm Initial Synchronization

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will verify that the Azure Active Directory Connect directory synchronization has occurred, and then you will force a manual delta synchronization to confirm that future changes will also synchronize. | 1. In the **ACTIVE DIRECTORY**, if not, go to Classic Azure Portal.   This will open the classic Azure Portal. At this time, Azure Active Directory can only be managed in this portal.   1. Click ACTIVE DIRECTORY. 2. In the Active Directory pane, your Directory, click Default Directory. 3. In the “default directory” page, click USERS. 4. The first user listed should be “**Admin**” and be sourced from Local Active Directory, as seen below. |

#### Activity 2: Test Delta Synchronization

| **Description** | **Steps** |
| --- | --- |
| In this exercise, you will test delta synchronization and enable ongoing synchronization. | 1. Minimize your web browser. 2. In Windows 10, click on Start Menu and type in “**Active Directory Users and Computers**.” 3. Double click and expand **contoso.com** 4. Right-click **Users**, click **New** then **User** and create a new user account in the Users container with the following details:  * **First Name:** Test * **Last Name:** User * **Full Name:** Test User * **User logon name:** TestUser * **User logon name (pre–Windows 2000):** TestUser * **Password:** Passw0rd! * *Uncheck* **- User must change password at next logon**: * **Next** and **Finish,** * **Close** Active Directory Users and Computers   Begin this activity logged on to AZRCamp-Sync*,* as Contoso\Administrator with a password of Passw0rd!   1. Use the shortcut on the desktop to open the Windows Azure Active Directory Module for Windows PowerShell. 2. Type the following command, and press Enter.   cd ‘c:\program files\microsoft azure ad sync\bin’   1. Type the following command, and press Enter. This will manually trigger a delta directory synchronization.   .\directorysyncclientcmd.exe delta  Begin this activity logged on to AZRCamp-Admin*,* as Contoso\Administrator with a password of Passw0rd!   1. Open up Azure Active Directory to your directory 2. In Edge, refresh the browser, Refresh the “**USERS**” page of the Azure default directory. **Test User** should now appear in the Azure default directory.  * You have just confirmed that ongoing synchronization is successful. |

### Exercise 3: Establishing SSO with a Third-Party Provider and Azure Active Directory

Azure is not the only SaaS provider that organizations use. Azure Active Directory allows you to create an SSO solution that can securely store the credentials for these third-party providers, as well as act as a broker to use these authentication mechanisms for your own web apps.

In this exercise, you will configure Azure Active Directory to store the credentials for a third-party SaaS provider, and then you will test the SSO capabilities provided by the Azure Active Directory sign-in page.

#### Activity 1: Adding Third-Party Providers to Azure Active Directory

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will add a third-party provider to your Azure Active Directory and configure it with predefined credentials. | 1. If you are not already there, open the **Azure Portal**, and open the default Azure Active Directory page. 2. Click APPLICATIONS. 3. At the bottom of the screen, click ADD. 4. On the “What do you want to do?” page, click Add an application from the gallery. 5. On the “Add an application for my organization to use” page, click Social in the left column, then click Facebook.  * If you have an account for a different application that is in the gallery, you may use that application instead.  1. In the lower right corner click Complete). 2. On the Facebook (or whichever application you are using) page, click Configure single sign-on.      1. On the “**How would you like users to sign on to Facebook**?” page, select “**Password Single Sign-On**,” and then click the check mark, Complete. 2. Click Assign Accounts. 3. Click on the line for “**Test User**,” and then click ASSIGN. 4. *Check* “**I want to enter Facebook credentials on behalf of the user**” box.  * If you choose not to enter credentials, the user will be given the opportunity to provide the credentials when he or she first attempts to access the SaaS application from the Azure Active Directory sign-in page.  1. Enter a Facebook logon email address and password that you know, and then click Complete.  * It will be securely stored as part of Azure Active Directory. * Tip: Type the password in Notepad to ensure that it is correct, and then past it to the Azure screen. If you make a mistake, SSO will fail.  1. Click DASHBOARD.      1. From the “quick glance” section of the application’s dashboard, click and copy the **Single Sign-On URL**. |

#### Activity 2: Test Third-Party SSO

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will install the access agent on a client machine and test that Azure SSO is passing the credentials to the third-party application. | 1. Click the Start button and search for **Internet Explorer** 2. Open an **Internet Explorer inPrivate browsing** window, and paste in the URL you just copied into the address bar.  * If prompted for credentials, use the Test User username and password. Your user name will be the one in the directory **EXAMPLE**  TestUser@xxxxxxxx.onmicrosoft.com  1. When prompted to install some software first, click install this software. 2. In the download bar at the bottom of your browser, click Run. 3. In the Welcome to the **Access Panel Extension** Setup Wizard, click Next. 4. Click Install. 5. In the User Account Control dialog box, click Yes. 6. Click Finish. 7. In the Set Up Internet Explorer 11 box, choose to use the recommended settings, and then click OK. 8. In the add-in box at the bottom of Internet Explorer 11, click **OK**, Enable. 9. Close Internet Explorer.  * Internet Explorer needs to be restarted for the Access Control add-in to function properly.  1. Click the Start Menu and search for Internet Explorer 11 then open IE 2. Type [**https://myapps.microsoft.com**](https://myapps.microsoft.com) into the address bar, and then press Enter. 3. Log in with your TestUser@<directoryname>.onmicrosoft.com account. 4. You will see the application appear on the access panel.      1. **Note:** You will not be able to log in as this is a test account, you will need to provision your FaceBook accounts before assigning the users to the accounts on the Facebook side. (You can try to add your own account into your own Active Directory for Facebook or another account of your liking. ) 2. Minimize the browser. |

### Exercise 4: Cloud App Discovery

Azure Cloud App Discovery (a feature of the Azure Active Directory Premium services) allows you to track what SaaS applications are being accessed by your staff. The Cloud App Discovery agent, once deployed to a user, can evaluate the web and application traffic and report back to Azure which SaaS applications are being used.

Cloud App Discovery can then compare and analyze that data and generate reports to help you better understand which known (or registered) apps are being used, as well as which unregistered apps are being used. With this report, you can take appropriate action to meet your needs, whether that is to register the apps for SSO in Azure Active Directory or find ways to block access.

In this exercise, you will enable the Azure Active Directory Premium trial, enable the Azure Active Directory Cloud App Discovery feature, take an inventory, and explore the Cloud App Discovery reports.

#### Activity 1: Enable the Azure Active Directory Premium Trial

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will enable the Azure Active Directory Premium functionality in a trial state, and assign an initial Azure Active Directory Premium license to your Global Admin account. | 1. In **Azure** **Active Directory**, open the default directory quickstart page, if it isn’t already open. 2. In the #3 **Get Azure AD Premium** section, click Try it now. 3. ClickTRY AZURE ACTIVE DIRECTORY PREMIUM NOW 4. Review the Azure Active Directory Premium trial terms, may need to click refresh, then click the check mark, Complete. 5. Click the link to refresh the status of the Azure Active Directory Premium license until the license appears in the portal. 6. On the bottom of the page, click Assign. 7. Choose your current logged-on user and Test User, then click Complete |

#### Activity 2: Enable Azure Active Directory Cloud App Discovery

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will enable the Azure Cloud App Discovery feature for your subscription. | 1. Open the new Azure Portal, and log on with your Azure administration account.  * If you were logged on to the Portal while you activated Azure Active Directory Premium, then you will need to log out of the Portal and log back on. Until you do that, Azure will not recognize that you have an Azure Active Directory Premium license.  1. Click New, then click Security + Identity. 2. In the Security + Identity blade, click See All, scroll down .      1. Click Azure AD Cloud App Discovery. 2. On the Azure AD Cloud App Discovery blade, click Create. 3. On the Cloud App Discovery blade, click Create. |

#### Activity 3: Configure Cloud App Discovery and Generate Data

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will configure Cloud App Discovery to search for all known (by Azure) SaaS cloud apps. You will then download, install, and test the Cloud App Discovery agent. | 1. Once the Cloud App Discovery object is created, click Settings. 2. On the Settings blade, click Manage Agent. 3. On the Manage Agent blade, click Please select a consent option. 4. Put a check mark in the box next to No notification or consent required. 5. Click Update. 6. Close this window and go to **Settings** blade, click Data Collection. 7. Set the Collection options to All Apps. 8. Click Save, then **Close** the Data Collection blade. 9. On the Manage Agent blade, click Download. 10. When it finishes downloading, click Open. 11. Copy and paste the two files in the directory to C:\. If prompted for permission, click Continue. 12. Double-click **C:\**EndpointAgentSetup.exe. 13. In the Cloud App Discovery–Endpoint Agent setup, click Install. 14. Click Yes to allow changes to be made. 15. When the installation is complete, click Close. 16. Close the File Explorer |

#### Activity 4: Generate Cloud App Activity

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will generate some traffic to various cloud services to generate traffic for the Cloud App Discovery service to report on. | 1. Click Start, then click Github. (You can access any other cloud services you are subscribed to, if you so choose.) 2. If you have a Github account, log on using that account. Leave Github running. 3. Use the Start menu to open Groove Music. 4. Leave these applications running.  * It might take up to 15 minutes for the data to appear in the Azure Portal.  1. If you are concerned that data is not appearing in the Azure Portal, confirm that the Microsoft Cloud App Discovery Endpoint Agent service is running on the computer. |

#### Activity 5: Explore the Cloud App Discovery Report

| **Description** | **Steps** |
| --- | --- |
| In this activity, you will view the Cloud App Discovery reports and register an app for management with Azure Active Directory. | 1. Once the data starts being passed to Azure, the Cloud App Discovery charts will start to appear, similar to what you see below. 2. Log into the Azure portal with your original Azure credentials after about 15 minutes and scroll to the right, you will see Cloud App Discovery,      1. In the Activity chart, click Edit Chart.      1. In the **Edit Chart** blade, choose a view other than “**Count**,” and click Update. 2. You will notice that the data is still categorized by **Managed** and **Unmanaged,** but the summary values reflect the data being displayed. 3. Click on **Users** - Note that Azure itself is reporting as an app.      1. You should also notice that every app has a status of either **Managed** or **Unmanaged**. 2. Click on the listing for one of the unmanaged apps. 3. This opens a more detailed set of reports for that app, but it might also suggest that you manage the app with Azure Active Directory. Click Manage GitHub with Azure AD (where <xxx> is the name of the app you chose). 4. If it is a supported app, the Azure Portal opens directly to the homepage of your Azure Active Directory service, specifically to the page to add SaaS applications.   If you choose to manage this service, you would configure it from here.   1. Close the Azure “Classic” Portal and Azure ”New” Portal. 2. End of labs. |

**Worksheet Exercise: Designing for Appropriate Workloads with Questions and Answers**

Designing for appropriate workloads requires that performance benefits are relevant to the workloads and are cost-effective. Different applications will have different workloads. Some will be write-intensive, while others will be more read-intensive. And the data that comprises the workloads can vary from small to large. Within an Azure subscription, there is a cost associated with storage, compute, and other resources. Furthermore, software licensing models might be directly related to resources. For example, some database applications are licensed on a per-CPU basis; the greater the number of CPUs, the greater the cost.

Microsoft Azure offers many choices for designing compute, storage, and other resources to provide an optimal fit for a particular workload or set of workloads. Designing resources for particular workloads requires an understanding not only of the capabilities and cost associated with particular resources but also of subscription and service limits, quotas, and constraints.

In this lab exercise, you will learn about some of the more relevant capabilities within Azure and considerations that will allow you to design for cost-effective performance that matches your desired requirements.

***Overview of Azure Storage and Compute Resource Capabilities***

Microsoft recently added Premium Storage and two new series of virtual machines (D and G series) to provide additional, high-performance capacity for your workloads and applications.

Virtual machine sizes are categorized by tier (basic or standard) and series (A, D, DS, G, and GS).

***Basic Tier***

The basic tier provides only A series virtual machines. The basic tier is adequate for small to medium applications or workloads that do not require advanced features, such as load-balancing or auto-scaling. Alternatively, basic tier virtual machines might be a cost-effective option for test/dev cycles. However, consideration would have to be given to the ease of scaling up or out of applications or workloads. Basic tier virtual machines belong to a scale unit that includes only A0–A4 virtual machines. If you need to scale up to a larger virtual machine, such as an A5, you will not be able to do so directly in the UI; you must do a more time-consuming manual migration.

The basic tier A series virtual machines represent the smallest virtual machines you can deploy. The smallest virtual machine is Basic\_A0, which provides 1 CPU core and 768 MB of RAM; for storage, the Basic\_A0 provides up to 1 1,023 GB operating system disk, 1 1,068 GB data disk, and 300 input/output operations (IOPS) per disk. The largest basic tier virtual machine is Basic\_A4, which provides 8 CPU cores, 14 GB of RAM, 1 1,023 GB operating system disk (max size), and up to 16 1,023 GB data disks. The maximum IOPS for each data disk is 300 IOPS. So, if the data disks are striped in a storage pool on a Basic\_A4 virtual machine, it is possible to achieve a theoretical maximum of 16×300= 4,800 IOPS.

***Standard Tier***

Standard tier offers more capabilities, such as load balancing and auto-scaling. Additionally, it offers a wide range of virtual machine sizes and performance capabilities. The standard tier includes A series virtual machines, as well as the D and G series. Like basic tier virtual machines, these virtual machines are also grouped into scale units (also known as stamps). The scale unit groupings are as follows:

**SU 2:** A0–A7

**SU 3:** A8–A11 (HPC virtual machines)

**SU 4:** A0–A7 and D1–D14

**SU 5:** G1–G5

Scale units have consequences if you are using cloud services and need to ensure that you can scale up. For example, to ensure that you can scale from an A series virtual machine to a D series virtual machine directly in the UI, you need to create your first virtual machine in the cloud service as a D series virtual machine. This will ensure that any subsequent A series virtual machines you create can be upgraded to D series virtual machines.

Within a series, the greater the numerical value, the more resources (such as CPU cores, RAM, disks, and IOPS) the virtual machines can have. For the A0–A4 series of virtual machines, the capabilities of the virtual machines for the standard tier are similar to those of the basic tier, except that virtual machines in the standard tier have larger temporary drives and have higher maximum IOPS (500) per disk. An A4 extra-large virtual machine could theoretically have a maximum IOPS of 16 striped data disks × 500 IOPS = 8,000 total IOPS.

***D and G Series***

The D and G series of virtual machines offer more memory and faster processing power than the A series machines, with the possible exception of A8–A11 virtual machines. Both the D and G series virtual machines offer local solid-state drive (SSD) storage for the temporary D: drive that is created by default when you provision a virtual machine in Azure. At the extreme upper end of the G series (G5), a virtual machine can have up to 32 vCPUs, 448 MB of memory, and 6.59 TB of SSD space for the temporary drive. Additionally, the G5 series can have up to 64 persistent drives, each supporting a maximum of 500 IOPS.

Because not all datacenters have the same hardware capacity, not all Azure datacenters can support G series virtual machines. For example, at the time of this writing, G series virtual machines could be created only in the East US 2, West US, and West Europe datacenters. Other considerations for deciding whether to use D or G series virtual machines are whether you want to be able to resize and whether other types of virtual machines can exist in the same cloud service. For example, it is not possible to resize to a G series virtual machine from any other series, and G series virtual machines cannot be in the same cloud service as other series.

***Premium Storage (DS and GS series)***

Premium Storage is available for the DS and GS series virtual machines. Premium Storage provides durable data storage on SSD drives. Using SSD drives for durable storage allows Azure to support I/O-intensive workloads. Using Premium Storage, applications can have up to 64 TB of storage per virtual machine and, depending on the configuration, achieve more than 80,000 IOPS per virtual machine and 2,000 MBps disk throughput.

The following two tables show the virtual machine and disk sizes and limits DS and GS series virtual machines and the P10, P20, and P30 storage disk types.

| **Virtual Machine Size** | **Number of CPU Cores** | **Maximum Disk IOPS  (per virtual machine)** | **Maximum Disk Bandwidth**  **(per virtual machine)** | **Cache Size (GB)** |
| --- | --- | --- | --- | --- |
| **STANDARD\_DS1** | 1 | 3,200 | 32 MBps | 43 |
| **STANDARD\_DS2** | 2 | 6,400 | 64 MBps | 86 |
| **STANDARD\_DS3** | 4 | 12,800 | 128 MBps | 172 |
| **STANDARD\_DS4** | 8 | 25,600 | 256 MBps | 344 |
| **STANDARD\_DS11** | 2 | 6,400 | 64 MBps | 72 |
| **STANDARD\_DS12** | 4 | 12,800 | 128 MBps | 144 |
| **STANDARD\_DS13** | 8 | 25,600 | 256 MBps | 288 |
| **STANDARD\_DS14** | 16 | 50,000 | 512 MBps | 576 |
| **STANDARD\_GS1** | 2 | 5,000 | 125 MBps | 264 |
| **STANDARD\_GS2** | 4 | 10,000 | 250 MBps | 528 |
| **STANDARD\_GS3** | 8 | 20,000 | 500 MBps | 1056 |
| **STANDARD\_GS4** | 16 | 40,000 | 1,000 MBps | 2112 |
| **STANDARD\_GS5** | 32 | 80,000 | 2,000 MBps | 4224 |

| **Storage Disk Type** | **P10** | **P20** | **P30** |
| --- | --- | --- | --- |
| **Disk size** | 128 GB | 512 GB | 1,024 GB (1 TB) |
| **IOPS per disk** | 500 | 2,300 | 5,000 |
| **Throughput per disk** | 100 MBps | 150 MBps | 200 MBps |

***Planning Considerations for Using Premium (or Standard) Storage***

Premium Storage is more expensive than Standard Storage and requires that you have a Premium Storage account. However, because of the performance gains offered by the storage subsystem, Premium Storage might be more cost-effective than virtual machines that are using Standard Storage. For example, a database application that is licensed per CPU core might have a lower cost overall if it can run on a DS virtual machine that has fewer cores than a D or A series virtual machine.

Premium Storage is locally redundant storage (LRS). Three copies are kept within the local region.

You can use both Standard and Premium Storage in DS or GS series virtual machines. This can provide cost benefits in that you can place the high-performance storage where you need it. For example, if the operating system disk is used only for booting, you can use a standard disk for the operating system disk. An SSD disk will not provide a significant benefit in this case.

By default, when you create Premium Storage data disks, the caching policy is read-only; for operating system disks, the caching policy is read-write. If you use the data disks for write-intensive applications (e.g., SQL Server logs), you should disable disk caching to ensure that writes are flushed to the disk to mitigate risk to data integrity.

Premium Storage is subject to bandwidth and IOPS limits and constraints that differ according to the Premium Storage disk type. There are three disk types that you can choose from, each with its own cost profile: P10, P20, and P30. A P10 disk is 128 GB in size, provides a maximum of 500 IOPS per disk, and offers 100 MB throughput per disk. A P30 disk is 1 TB in size, provides a maximum of 5,000 IOPS per disk, and offers 200 MB throughput per disk.

The limits for Premium Storage disks are for the disk traffic alone and do not include cache hits or network traffic, which is measured separately. For example, cache hits from disks configured as read-only are not subject to the Premium Storage disk limitations but are subject to separate IOPS/throughput limits based on the virtual machine size. A DS series virtual machine provides approximately 4,000 IOPS and 33 MBps per core for cache and local SSD I/Os. If you have disks configured to use read-only caching and you are mainly performing reads, you can achieve higher IOPS than the maximum limits indicated for the Premium Storage disks.

To achieve maximum throughput and IOPS with multiple data disks, you can stripe the disks using Storage Spaces. It is recommended that you use one column for each disk. If you have more than eight disks, you must use the NumberOfColumns parameter in the New-VirtualDisk cmdlet to specify more than eight columns. By default, the UI creates a maximum of eight columns, even if there are more than eight disks. The total IOPS and throughput are a product of the number of disks in a stripe set:   
  
**number of disks in a stripe set × IOPS per disk = total IOPS**

Be aware of the different throttling limits in your design choices. For example, both the DS1 and DS2 virtual machine series limit disk bandwidth to 32 MBps and 64 MBps, respectively. This means that a P10 disk will not reach it maximum throughput limit of 100 MBps if it is used in these virtual machines. A GS5 series virtual machine can provide up to 2,000 MBps throughput across all disks. In some cases, the limiting factor will be the disk. For example, a DS4 virtual machine provides up to 256 MBps for disk throughput; however, a single P30 disk provides only 200 MBps and is the limiting factor.

The maximum number of IOPS for a storage account is 20,000. Your IOPS will be throttled if the sum of the IOPS for disks in the same storage account exceeds 20,000. For example, if you have a single GS5 virtual machine with five striped P30 disks that belong to the same storage account, your IOPS will be throttled at 20,000. To achieve the maximum possible IOPS (25,000) in this scenario, you need to place the disks across two storage accounts. For each storage account, use the following formula to determine the number of storage accounts you require, where x equals the number of standard or P10 disks, y equals the number of P20 disks, and z equals the number of P30 disks. If the result is greater than 1, you need two or more storage accounts.  
  
**(x × 500 IOPS) + ( y × 2,300 IOPS) + ( z × 5,000 IOPS)/20,000**

IOPS and throughput are related, but the limits for each are applied separately. The relationship between IOPS and throughput is described by this formula: **IOPS × I/O size = throughput**. For example, 1,000 IOPS × 8 KB = 8 MBps.

I/O size is configurable and has implications for throughput and IOPS. If your applications use larger files, increase the I/O size to increase throughput. If you require more IOPS, reduce the I/O size.

Outstanding I/O requests are also known as queue depth (QD) and are related to performance as follows: **IOPS × latency = QD**. For example, if you 5,000 IOPS with a latency of 1 ms, your QD equals 5 (5 is an optimal QD). Give this example, if your QD is more than 5, the IOPS is a limiting factor and your disk is a bottleneck. Likewise, if your QD is consistently lower than 5, you are not taking advantage of the full capacity of the disk subsystem.

Be aware of account and other scale limits. For example, the maximum allowable disk capacity for Premium Storage accounts is 35 TB. Therefore, you would need to get an additional storage account to accommodate the 64 1 TB disks allowed for a G5 virtual machine.

For more information, please see the following:

Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads — <https://azure.microsoft.com/en-us/documentation/articles/storage-premium-storage-preview-portal/>

Azure Subscription and Service Limits, Quotas, and Constraints —   
<https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/>

Azure Storage Scalability and Performance Targets —   
<https://azure.microsoft.com/en-us/documentation/articles/storage-scalability-targets/>

Sizes for virtual machines —   
<https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-size-specs/>

New D-Series Virtual Machine Sizes —  
 <https://azure.microsoft.com/en-us/blog/new-d-series-virtual-machine-sizes/>

Largest VM in the Cloud —   
<https://azure.microsoft.com/en-us/blog/largest-vm-in-the-cloud/>

***Review Questions***

*The answers can be found in the appendix.*

The following provides a number of questions to reinforce your understanding of throttling limits and capacity:

1. You configure a DS1 virtual machine with two striped P10 data disks. Excluding read-only cache, what is the maximum theoretical disk throughput? What is the maximum throughput for the virtual machine? What is the limiting factor, the disk or the virtual machine?
2. Your application is capable of writing 128 KB blocks to a P10 disk on a DS2 virtual machine. What is the maximum throughput that your design is capable of achieving?
3. You need to design a solution that can theoretically achieve the highest possible disk throughput of 2,000 MBps using the minimum number of striped disks. Your application reads and writes data in 64 KB blocks. Specify the virtual machine series, the disk type, and the IOPS needed to achieve the maximum throughput.
4. Using the same design as provided in the answer to question 3 above, if your application uses data in 32 KB blocks, would you be able to achieve the 2,000 MBps throughput?
5. Given a latency of 1 ms, what is the QD needed to achieve 2,300 IOPS?

You have five DS3 virtual machines, each with four striped P20 disks that belong to a single storage account. What limit is going to affect disk performance? What can you do to achieve optimal performance?

1. **Answers to Review Questions**
   1. You configure a DS1 virtual machine with two striped P10 data disks. Excluding read-only cache, what is the maximum theoretical disk throughput? What is the maximum throughput for the virtual machine? What is the limiting factor, the disk or the virtual machine?   
      **Answer: 32 MBps. The virtual machine limit will cause the throughput to be throttled. Each virtual disk is capable of 100 MBps throughput to provide a maximum of 200 MBps when striped. However, a DS1 virtual machine has a maximum disk capacity throughput of 32 MBps.**
   2. Your application is capable of writing 128 KB blocks to a P10 disk on a DS2 virtual machine. What is the maximum throughput that your design is capable of achieving?   
      **Answer: (500 IOPS × 128 KB)/1,024 = 62.5 MBps. A DS2 virtual machine is capable of disk throughput of 64 MBps, so the disk is (just barely) the limiting factor.**
   3. You need to design a solution that can theoretically achieve the highest possible disk throughput of 2,000 MBps using the minimum number of striped disks. Your application reads and writes data in 64 KB blocks. Specify the virtual machine series, the disk type, and the IOPS needed achieve the maximum throughput.   
      **Answer: At 200 MBps, a P30 disk provides the highest throughput. To achieve 2,000 MBps, you need 10 striped P30 disks. Given a block size of 64 KB, you would need to achieve 3,200 IOPS per disk to achieve the goal of 2,000 MBps: (3,200 IOPS × 64 KB)/1,024 = 200 MBps per disk.**
   4. Using the same design as provided in the answer to question 3 above, if your application uses data in 32 KB blocks, would you be able to achieve the 2,000 MBps throughput?  
      **Answer: No. To achieve the desired throughput of 200 MBps per disk, you need 6,400 IOPS.**
   5. Given a latency of 1 ms, what is the QD needed to achieve 2,300 IOPS?  
      **Answer: IOPS × latency = QD = 2,300 × .001 = 2.3**
   6. You have five DS3 virtual machines, each with four striped P20 disks that belong to a single storage account. What limit is going to affect disk performance? What can you do to achieve optimal performance?  
      **Answer: Each storage account is limited to 20,000 IOPS. Each of the DS3 virtual machines is capable of using 9,200 IOPS (4 × 2,300 = 9,200) for a combined total of 46,000 IOPS (9,200 × 5 = 46,000). To achieve the maximum performance target of 46,000 IOPS, you need to spread the disks across three separate storage accounts (46,000/20,000 = 2.3)**.

# At home: Resources to continue learning

By participating in today’s event you will have exclusive access to download the presentation, step-by-step setup guidance, and latest online training and demos at [aka.ms/itinnovationresources](http://www.aka.ms/itinnovationresources).

|  |  |
| --- | --- |
| Resources | What you’ll learn |
| Event presentation. | Today’s presentation on “What’s new in Azure Infrastructure: Improving Datacenter Flexibility with Microsoft, Open Source and Other Technologies.” |
| Step-by-step guidance (attendee guide) | This document guides you through today’s instructor-led and individual activities, and provide you with the key resources and information necessary to continue building your skills to be able to implement in your environment. |
| Try Azure | Sign up for a free one-month trial of Azure. |
| Access Azure videos and online training | The most recent Enterprise Mobility, On Premises, In Cloud (Azure) and DevOps news and resources, from Microsoft and in the industry. |
| Blog: Regular IT Guy | Stay up to date on the latest in Azure. |

# Cloud Infrastructure

# TechNet on Tour Resources – Jan-Mar 2016

*You can find all* ***session slides****, this* ***resource document****, plus other related materials here:* [**http://aka.ms/TNEventResources**](http://aka.ms/TNEventResources)

*Do you have a question? Ask on our Event Forum:* [**http://aka.ms/tne**](http://aka.ms/tne)

**Referenced Event Resources**

**Azure FREE Trial –** [**http://aka.ms/try-azure**](http://aka.ms/try-azure)

Hands-on-Lab files - <https://github.com/AZITCAMP/Labfiles>

Azure Portal – <http://portal.azure.com>

Azure Trust Center (Security/Compliance/Privacy) <http://aka.ms/AzureCompliance>

Azure Resource Manager overview

<https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/>

Understanding Resource Manager deployment and classic deployment

<https://azure.microsoft.com/en-us/documentation/articles/resource-manager-deployment-model/>

Resource Manager Architecture

<https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-azure-resource-manager-architecture/>

Azure Quick-Start Templates (searchable)

<https://azure.microsoft.com/en-us/documentation/templates/>

Azure Quick-Start Templates (repository)

<https://github.com/Azure/azure-quickstart-templates>

Authoring Azure Resource Manager templates

<https://azure.microsoft.com/en-us/documentation/articles/resource-group-authoring-templates/>

Deploy an application with Azure Resource Manager template

<https://azure.microsoft.com/en-us/documentation/articles/resource-group-template-deploy/>

Develop and share your code in Git using Visual Studio

<https://www.visualstudio.com/en-us/get-started/code/share-your-code-in-git-vs>

Premium Storage: High-Performance Storage for Azure Virtual Machine Workloads

<https://azure.microsoft.com/en-us/documentation/articles/storage-premium-storage-preview-portal/>

Sizes for virtual machines   
<https://azure.microsoft.com/en-us/documentation/articles/virtual-machines-size-specs/>

Virtual Machines Pricing

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

New D-Series Virtual Machine Sizes  
<https://azure.microsoft.com/en-us/blog/new-d-series-virtual-machine-sizes/>

Largest VM in the Cloud

<https://azure.microsoft.com/en-us/blog/largest-vm-in-the-cloud/>

How Azure storage pricing works

<https://azure.microsoft.com/en-us/pricing/details/storage/>

Virtual Machines Documentation

<https://azure.microsoft.com/en-us/documentation/services/virtual-machines/>

Azure Storage Scalability and Performance Targets  
<https://azure.microsoft.com/en-us/documentation/articles/storage-scalability-targets/>

Azure Subscription and Service Limits, Quotas, and Constraints

<https://azure.microsoft.com/en-us/documentation/articles/azure-subscription-service-limits/>

Best Practices for Large Deployments of Azure Nodes with Microsoft HPC Pack

<https://technet.microsoft.com/en-us/library/jj994092.aspx>

Designing for Big Scale in Azure

<http://blogs.msdn.com/b/kdot/archive/2014/10/09/designing-for-big-scale-in-azure.aspx>

Azure Virtual Network

<https://azure.microsoft.com/en-us/services/virtual-network/>

Virtual Network Documentation

<https://azure.microsoft.com/en-us/documentation/services/virtual-network/>

Instance Level Public IP Overview

<https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-instance-level-public-ip/>

Reserved IP addresses for Cloud Services & Virtual Machines

<https://azure.microsoft.com/en-us/blog/reserved-ip-addresses/>

Azure Datacenter IP Address Ranges

<https://msdn.microsoft.com/en-us/library/azure/dn175718.aspx>

Azure DNS

<https://azure.microsoft.com/en-us/services/dns/>

User Defined Routes and IP Forwarding

<https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-udr-overview/>

Multiple VM NICs and Network Virtual Appliances in Azure

<https://azure.microsoft.com/en-us/blog/multiple-vm-nics-and-network-virtual-appliances-in-azure/>

Create a Multi-NIC VM with a Public IP in Azure

<http://blogs.msdn.com/b/rslaten/archive/2014/11/18/create-a-multi-nic-vm-with-a-public-ip-in-azure.aspx>

Create a VM with multiple NICs

<https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-multiple-nics/>

About VPN Devices and Gateways for Virtual Network Connectivity

<https://msdn.microsoft.com/en-us/library/azure/jj156075.aspx>

ExpressRoute

<https://azure.microsoft.com/en-us/services/expressroute/>

ExpressRoute Pricing

<https://azure.microsoft.com/en-us/pricing/details/expressroute/>

New Networking features and partnerships for Enterprise scenarios

<https://azure.microsoft.com/en-us/blog/networking-enterprise/>

Azure Standard VPN Gateway

<https://azure.microsoft.com/en-us/updates/azure-standard-vpn-gateway/>

Cloud basics - Security in the cloud

<http://www.microsoft.com/industry/government/guides/cloud_computing/3-security.aspx>

Microsoft Azure Trust Center: Security

<https://azure.microsoft.com/en-us/support/trust-center/security/>

Network Security Groups

<https://azure.microsoft.com/en-us/blog/network-security-groups/>

What is a Network Security Group (NSG)?

<https://azure.microsoft.com/en-us/documentation/articles/virtual-networks-nsg/>

Azure Active Directory

<https://azure.microsoft.com/en-us/services/active-directory/>

Active Directory Federation Services (AD FS) & Azure Active Directory Sync (DirSync) Resources

<http://simon-may.com/adfs-dirsync-waad-resources/>

Authentication Scenarios for Azure AD

<https://azure.microsoft.com/en-us/documentation/articles/active-directory-authentication-scenarios/>

Azure Active Directory application gallery: Federated SaaS apps

<http://social.technet.microsoft.com/wiki/contents/articles/20235.azure-active-directory-application-gallery-federated-saas-apps.aspx>

Cloud App Discovery Tool

<https://appdiscovery.azure.com/>

### FREE Online Training: Microsoft Virtual Academy

<http://aka.ms/MSVirtAcademy>

This document supports a preliminary release of a software product that may be changed substantially prior to final commercial release. This document is provided for informational purposes only and Microsoft makes no warranties, either express or implied, in this document. Information in this document, including URL and other Internet Web site references, is subject to change without notice. The entire risk of the use or the results from the use of this document remains with the user. Unless otherwise noted, the companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in examples herein are fictitious. No association with any real company, organization, product, domain name, e-mail address, logo, person, place, or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Copyright 2014 © Microsoft Corporation. All rights reserved.

Microsoft Active Directory, Azure Active Directory, Azure, Hyper-V, Windows, and Windows Server 2012 are trademarks of the Microsoft group of companies.

All other trademarks are property of their respective owners.