# ^AZ00001: Azure Portal and Navigation

18 September 2018 11:42

AZ00001: Azure Portal and Navigation

1 Hr 55 Min Remaining

Instructions

Resources

Help

100%

## Introduction

## Azure Foundations & Management

*Getting started with the Azure portal*

## Overview

Microsoft Azure is a multi-tenant, public cloud computing platform. It is designed for building, deploying, and managing applications and services through a global network of Microsoft-managed datacenters. It provides SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service) and supports many different programming languages, tools, and frameworks, including both Microsoft and third-party software and systems.
In this lab, we will be focusing on the Azure foundation. The following topics/objectives will be covered:

- Portals and Navigation
- Accounts, Subscriptions, and Usage
- Classic vs. Resource Manager

## Portals and Navigation

- The portal is a central place to deploy and manage Azure resources. It is a comprehensive marketplace that allows the user to browse through thousands of items from Microsoft (and others) that can be purchased and/or provisioned on the Azure cloud platform.
- The portal provides a unified and scalable browser experience that makes it easy to find resources that can be used to perform various management operations.
- The portal contains management web pages (referred to as **blades**) including settings, actions, billing information, health monitoring, usage data, and other configuration options that the user can manage.
- The portal provides a customizable experience that allows the user to create a dashboard that will display information that is important to the individual user. Some settings and options (which are displayed as **tiles** ) that can be displayed on the dashboard can include:
  - **All resources** (displays deployed resources in all subscriptions by default)
  - **Service health** (displays the availability status of the Azure datacenters)
  - **Help + support** (Log support tickets and connect to the support community)
  - **Marketplace** (browse Microsoft and third-party offerings to purchase and/or provision in Azure)

  - **Settings** (set default preferences for dashboard color, subscription filter, language settings, and more)

## Capabilities or components used in this scenario:

- **Azure:** Azure Resource Manager (ARM)

Next: Scenario 1 - Navigation

From <https://labondeman

AZ00001: Azure Portal and Navigation

1 Hr 53 Min Remaining

Instructions

Resources

Help

100%

## Scenario 1 - Navigation

Perform the following steps to get familiar with navigating the Azure portal.

## Sign in to the Azure portal

## Home page and tiles

### Part A - Settings and notifications

1. At the top of the **Dashboard** page, click on the **Fullscreen**button to change the page to the fullscreen mode. Press the **Esc** key on your keyboard to exit fullscreen mode.
2. In the top right-hand corner, click on **Settings**.
3. Choose a different theme color, and click **Apply**.
4. In the top right-hand corner, click on Notifications.
5. Close the Notifications blade.
   **Note:** You can review the status of any activity performed within the Azure portal.

### Part B - Tiles

You will notice several tiles on the **Dashboard** page such as All resources, Get started, Marketplace, and Service health. Just like your desktop icons, these tiles provide quick access to Azure resources and different parts of the portal. Perform the following steps to observe the tile functions:

1. In the menu on the left side (referred to as the **Favorites** menu), click the **Show menu** button
   and then click on **Azure Active Directory**.
2. Click on the pin

at the top of the **Azure Active Directory** blade which will create a tile on the Dashboard.

3. Close the **Azure Active Directory** blade and note the new **Azure Active Directory** tile that now appears on the Dashboard.
4. On the **Dashboard** page, click the **Show menu** button

   to expand the **Favorites** menu, if it is collapsed.
5. Click on **All resources** to review the list of resources used for this lab.
6. Close the **All resources** blade.
7. Click **Service Health** tile on the **Dashboard**.
8. Click on **Health history**, select an option in the **Time range**dropdown menu to view all past issues for that time period.
9. Close the **Service Health** blade.
10. Click on **Monitor** in the **Favorites** menu.
11. Click on **Activity log**.
12. Select an option in the **Timespan** dropdown menu and select **All categories** in the **Event category** dropdown menu then click **Apply** to view a log of activities for that time period.
13. Close the **Activity log** blade.
14. Click the **Marketplace** tile and note the different offerings and categories that appear on the **Everything** blade.
15. Close the **Everything** blade.
16. Close the **Marketplace** blade.
17. On the **My Dashboard** page, click the **Show menu** button

   to expand the **Favorites** menu, if it is collapsed.
18. Click **Help + support** at the bottom of the **Favorites** menu and note the links to different support resources.
19. On the **Help + support** blade, click **New support request** to open the **New support request** blade which you can use to create a support ticket.
20. Close the **New support request** blade.

This completes this scenario. Click on **Next** to go to the next scenario.

PreviousNext: Scenario 2 - Pinning a resource group...

From <https://labondemand.com/CloudClient/fd4f8054-e54a-43de-a46d-5ad10b0f29be>

d.com/CloudClient/fd4f8054-e54a-43de-a46d-5ad10b0f29be>

# ^AZ00002: Azure Networking Concepts

18 September 2018 13:25

AZ00002: Azure Networking Concepts

2 Hours Remaining

Instructions

Resources

Help

100%

# Intro

## Networking Concepts

*Explore and deploy Azure networking resources.*

## Overview

Microsoft Azure is a multi-tenant, public cloud computing platform. It is designed for building, deploying, and managing applications and services through a global network of Microsoft-managed datacenters. It provides SaaS (Software as a Service), PaaS (Platform as a Service), and IaaS (Infrastructure as a Service) and supports many different programming languages, tools, and frameworks, including both Microsoft and third-party software and systems.
In this lab, we will be focusing on virtual networking concepts in Azure.

## Capabilities or components used in this scenario:

- **Azure:** Azure Resource Manager (ARM), Azure virtual networking, network security groups, load balancers

Next: Scenario 1 - Building a Virtual Network

From <https://labondemand.com/CloudClient/9879509e-8f56-42e0-80bf-a8d27f9e0602>

AZ00002: Azure Networking Concepts

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

# Scenario 1 - Building a Virtual Network

An **Azure Virtual Network (VNet)** is a logical extension of your physical network topology in the cloud, dedicated and isolated to your Azure subscription. You can fully control the IP address blocks, DNS settings, security policies, and route tables within this network. Click the following link to learn more about Azure VNets: Learn more.

## Sign in to the Azure portal

1. Click the **+New** button.
2. On the **New** blade, click **Networking** in the **Azure Marketplace** menu.
3. Within the virtual network blade, still in the Marketplace under resource manager, click **Create**
4. In the **Featured** menu, select the **Virtual network** app.
5. On the **Create virtual network** blade, type DemoLabVnet1 in the **Name** field.
6. In the **Address space** field, change the address space from **/24** to **/16**.
   This will change the total number of available IP addresses from **256** to **65,536**
7. In the **Subscription** dropdown menu, verify that a valid subscription is listed.
8. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(178).Name** resource group from the drop-down.
9. In the **Location** field, change the location to **West Central US**.
10. In the **Subnet name** field, leave **default** in the name field.
11. In the **Subnet address range** field, leave the **/24** default setting.
12. Click the **Create** button to deploy the **Virtual network**.

## Advanced configuration steps

In these optional steps, you will create additional subnets to demonstrate building multiple subnets for network segmentation in a virtual network.

1. On the **Dashboard** page, click **Virtual networks** in the menu on the left side (referred to as the **Favorites** menu).
2. On the **Virtual networks** blade, click **DemoLabVnet1** that was deployed in Scenario 1.
3. In the **SETTINGS** section of the **DemoLabVnet1** blade, click the **Subnets** field.
4. Click the **+Subnet** button to display the **Add subnet** blade.
5. In the **Name** field, type Production.
6. Verify that the following address range has been created: **10.0.1.0/24**.
7. Verify that **None** is selected for **Network security group**.
8. Verify that **None** is selected for **Route table**.
9. Click the **OK** button.
10. Verify that **Production** is now listed in the **Subnet** list.
11. Click the **+Subnet** button to display the **Add subnet** blade.
12. In the **Name** field, type Dev.
13. Verify that the following address range has been created: **10.0.2.0/24**.
14. Verify that **None** is selected for **Network security group**.
15. Verify that **None** is selected for **Route table**.
16. Click the **OK** button.
17. Verify that **Dev** is now listed in the **Subnet** list.
    You have completed this scenario. Click **Next** to go to the next scenario.

PreviousNext: Scenario 2 - Building network...

From <https://labondemand.com/CloudClient/9879509e-8f56-42e0-80bf-a8d27f9e0602>

AZ00002: Azure Networking Concepts

1 Hr 51 Min Remaining

Instructions

Resources

Help

100%

## Scenario 2 - Building network security groups

An Azure **Network Security Group (NSG)** lets you create **access control lists (ACLs)** to **block** or **allow** traffic on your subnets or individual virtual machines. To learn more about Azure NSGs, click the following link: Learn more.

1. Click the **+New** button.
2. On the **New** blade, type network security group in the **Search the marketplace** field and press **Enter**.
3. On the **Everything** blade, click **Network security group** in the **Results** list.
4. On the **Network security group** blade, leave the **Select a deployment model** option set to **Resource Manager** in the dropdown menu and click the **Create** button.
5. On the **Create network security group** blade, type prodNSG in the **Name** field.
6. In the **Subscription** dropdown menu, verify that a valid subscription is listed.
7. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(178).Name** resource group from the drop-down.
8. In the **Location** field, select **West Central US**.
9. Click the **Create** button.

## Advanced configuration steps

In these optional steps, you will modify the newly created **prodNSG**network security group to allow TCP port 80 for web server traffic, and apply/associate the NSG to the **DemoLabVnet1** virtual network that was created in Scenario 1.

1. On the **Dashboard** page, click **More services >** at the bottom of the **Favorites** menu.
2. In the **Filter** search box on the next blade, type Network security groups and press **Enter**.
3. On the **Network security groups** blade, click **prodNSG** in the list of NSGs.
4. In the **SETTINGS** section of the **prodNSG** blade, click**Inbound security rules**.
5. Click the **+Add** button.
6. In the **Action** field, choose **Allow**.
7. Verify the following default values are set: **Priority – 100** and **Source – Any**.
8. In the **Name** field, type Webserver.
9. Click on **Basic**.
10. In the **Service** dropdown menu, select **HTTP**.
    This will automatically configure the following settings: Protocol – TCP, Port Range – 80
11. Click on **Advanced**.
12. Click **OK** to apply your changes.
13. To verify the new rule has been created and is active, click the **Overview** link in the top left of the **prodNSG** network security group blade.
    **Note:** The **Webserver** rule should now be listed under **Inbound security rule**
14. In the **SETTINGS** section of the **prodNSG** blade, click the **Subnets** link.
15. Click the **+Associate** button to launch the **Associate subnet**blade.
16. Click the **Virtual network** step to display a list of virtual networks.
17. Click **DemoLabVnet1** to select the virtual network.
18. Click the **Production** to select the subnet.
19. Click the **OK** button.
    You have completed this scenario. Click **Next** to go to the next scenario.

PreviousNext: Scenario 3 - Building a load balancer

From <https://labondemand.com/CloudClient/9879509e-8f56-42e0-80bf-a8d27f9e0602>

AZ00002: Azure Networking Concepts

1 Hr 46 Min Remaining

Instructions

Resources

Help
100%

## Scenario 3 - Building a load balancer

Azure load balancer is a Layer 4 load balancer that distributes incoming traffic across healthy virtual machine instances using a hash-based distribution algorithm. By default, Azure load balancers use a quintuple (source IP, source port, destination IP, destination port, protocol type) hash to distribute traffic across available servers. Load balancers can either be Internet-facing (accessible via public IP addresses) or internal (only accessible from a virtual network). Azure load balancers also support network address translation (NAT) to route traffic between public and private IP addresses.

1. Click the **+New** button.
2. On the **New** blade, type load balancer in the **Search the marketplace** field and press **Enter**.
3. On the **Everything** blade, click **Load Balancer** in the **Results**list.
4. On the **Load balancer** blade, click the **Create** button.
5. On the **Create load balancer** blade, type DemoLabLB1 in the **Name** field.
6. Click the **Internal** radio button in the **Type** field to select it.
   You can use internal load balancers to balance traffic from private IP addresses. Public load balancers can balance traffic originating from public IP addresses. Click these links to learn more about each type:
   Internal, Public
7. Click the **Virtual network** field and select **DemoLabVnet1**.
8. Click the **Subnet** field and select **Production**.
9. In the **IP address assignment** section, verify that **Dynamic** is selected.
10. In the **Subscription** dropdown menu, verify that a valid subscription is listed.
11. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(178).Name** resource group from the drop-down.
12. In the **Location** field, verify that **West Central US** is selected.
13. Click the **Create** button to deploy the load balancer.
    You have completed this scenario. Click **Next** to go to the next scenario.

PreviousNext: Scenario 4 - Creating a Virtual...

From <https://labondemand.com/CloudClient/9879509e-8f56-42e0-80bf-a8d27f9e0602>

AZ00002: Azure Networking Concepts

1 Hr 45 Min Remaining

Instructions

Resources

Help
100%

## Scenario 4 - Creating a Virtual Network Gateway

A VPN gateway is a type of virtual network gateway that sends encrypted traffic between your virtual network and your on-premises location across a public connection. You can also use a VPN gateway to send traffic between virtual networks across the Azure backbone.

1. Click the **+New** button.
2. On the **New** blade, type virtual network gateway in the **Search the marketplace** field and

press **Enter**.

3. On the **Everything** blade, click **Virtual network gateway** in the **Results** list.
4. On the **Virtual network gateway** blade, click the **Create**button.
5. On the **Create virtual network gateway** blade, type DemoVNetGW1 in the **Name** field.
6. Verify that **VPN** is selected in the **Gateway type** field.
   Click the following link to learn more about the different types of gateways: Learn more
7. Verify that the **Route-based VPN** type is selected.
   The type of VPN you can choose depends on your VPN device and the kind of VPN connection you intend to create. Point-to-site, inter-virtual network, and multiple site-to-site VPN connections are only supported with a route-based virtual network gateway. In addition, if you are creating a VPN-type gateway to coexist with an ExpressRoute gateway, the VPN gateway must be route-based and the ExpressRoute gateway must be created first. Click the following link to learn more: Learn more.
   more...
8. In the **SKU** field, verify that **VpnGw1** is the selected type.
   Route-based VPN gateway types are offered in three SKUs: Basic, Standard, and High Performance. Standard or High Performance must be chosen if the gateway is being created to coexist with an ExpressRoute gateway. Click the following link to learn more: Learn more.
9. Click **Choose a virtual network** in the **Virtual network**section to display the **Choose virtual network** blade.
10. Click the **DemoLabVnet1** virtual network.
11. Verify that the following default **Gateway** subnet address range has been auto-selected: **10.0.3.0/24**.
    This should be the next subnet available based on subnets already in use.
12. Click **Create gateway IP configuration** in the **First IP configuration** section to display the **Choose public IP address** section.
13. Click **+Create new** to display the **Create public IP address**blade.
14. Type DemoVNetGW1-pip in the **Name** field.
    **Note:** This naming convention identifies the resource being created as a public IP address by ending the name with **-pip**.
15. Click the **OK** button.
16. Verify that the **(new) DemoVNetGW1-pip** name is displayed in the **First IP configuration** section.
17. In the **Subscription** dropdown menu, verify that a valid subscription is listed.
18. In the **Location** field, verify that **West Central US** is selected.
19. Click the **Create** button
    **Note:** This configuration can take up to 45 minutes to complete. Therefore you will NOT be completing the deployment in this lab scenario. Click the **Next** button to go to the lab conclusion.

PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/9879509e-8f56-42e0-80bf-a8d27f9e0602>

AZ00002: Azure Networking Concepts

1 Hr 26 Min Remaining

Instructions

Resources

Help

100%

## Conclusion

The **Azure Networking Concepts** lab has now been completed. The following networking elements have been reviewed and deployed:

- Virtual networks
- Additional subnets
- Network security groups
- Access control lists
- Load balancers
- Virtual network gateways

PreviousEnd

From <https://labondemand.com/CloudClient/9879509e-8f56-42e0-80bf-a8d27f9e0602>

# ^AZ00003: Azure Virtual Machine and Compute

18 September 2018 14:05

AZ00003: Azure Virtual Machine and Compute

2 Hours Remaining

Instructions

Resources

Help

100%

## Introduction

## Virtual Machines & Compute

*Deploy a Windows Server 2016 Datacenter Virtual Machine in Azure*

### Overview

Microsoft Azure is one of the key cloud platforms currently providing cloud computing to the enterprise market. Azure launched in 2010 and has quickly matured as many new services are continually being added. One of the first and most popular services offered in Azure is **Virtual Machines** , an IaaS solution.
This lab will create and deploy a Windows Server 2016 Datacenter virtual machine.
In this lab, the essential concepts for deploying a virtual machine will be explored by creating the following components:

- Resource Group
- Virtual Network
- Storage Account

During the creation of the virtual machine, additional properties will be reviewed, including the following:

- Availability Sets
- Dynamic vs Static IP Address
- Network Security Groups

## Capabilities or Components Used in this Story:

**Azure:** Azure Resource Manager Portal, Windows Server 2016 Virtual Machine, Azure Storage, Azure Virtual Networking, Azure Availability Sets, Azure Network Security Groups
Next: Scenario 1 - Building a Virtual Network

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

AZ00003: Azure Virtual Machine and Compute

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Scenario 1 - Building a Virtual Network

An **Azure virtual network** (VNet) is a representation of your own **network** in the cloud. It is a logical isolation of the **Azure** cloud dedicated to your subscription. You can fully control the IP address blocks, DNS settings, security policies and route tables within this **network.** Learn more.

## Login to the Azure Portal

## Building a Virtual Network

1. Click on the green **+New** button.
2. In the **New** blade, click on **Networking** in the **Azure Marketplace** menu.
3. In the **Featured** menu, select the **Virtual network** app.
4. In the Create virtual network blade, type DemoLabVnet1 in the **Name** field.
5. Leave the default **Address space**.
6. Verify a **Subscription** is listed in the **Subscription** drop-down.
7. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(180).Name** resource group from the drop-down.
8. In the **Location** field, choose **South Central US**.
9. Change the Subnet name to demolabsubnet.
10. Change the **Subnet address range** to a /26.
    This will supply 64 IP addresses to be used in the Virtual Network.
11. Click on the **Create** button to deploy the **Virtual Network**.
    This completes this scenario. Click on **Next** to go to the next scenario.

PreviousNext: Scenario 2 - Building an Azure Storage...

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

AZ00003: Azure Virtual Machine and Compute

1 Hr 55 Min Remaining

Instructions

Resources

Help

100%

## Scenario 2 - Building an Azure Storage Account

An **Azure Storage Account** provides a unique namespace to store and access your Azure storage data objects. All objects in an **Azure Storage Account** are billed together as a group. By default, the data in your account is available only to you, the account owner. Learn more.

1. Click the green **+New** button.
2. On the **New** blade, click on **Storage** in the **Azure Marketplace** menu.
3. In the **Featured** menu, select the **Storage account - blob, file, table, queue** app.
4. On the **Create storage account** blade, type demolabstdsto7312608 in the **Name** field.
   Storage account names must be unique across all existing storage account names in Azure. They must be 3-24 characters long and can only contain lowercase letters and numbers. If the portal advises the name is not unique, simply adjust the name. Once you create a unique name that is available, you will see a check mark at the end of the **Name** field.
5. Verify that the **deployment model** selected is **Resource manager**.

6. Verify that the **Account kind** selected is **General purpose v1**(It may be grayed-out, but should still show **General purpose**).
   To learn more about the different **Azure Storage Account** types follow the link. Learn more.
7. Verify that the storage account **Performance** is set to **Standard**.
   To learn more about the different storage performance types follow the link. Learn more.
8. Change the **Replication** type to **Locally-redundant storage (LRS)**.
   To learn more about the different replication types follow the link. Learn more.
9. Leave **Secure transfer required** set to the default value.
10. Verify a **Subscription** is listed in the **Subscription** drop-down.
11. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(180).Name** resource group from the drop-down.
12. In the **Location** field, choose **South Central US**.
13. Click on the **Create** button to deploy the **Storage Account**.
    This completes this scenario. Click on **Next** to go to the next scenario.

PreviousNext: Scenario 3 - Create an Azure...

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

AZ00003: Azure Virtual Machine and Compute

1 Hr 52 Min Remaining

Instructions

Resources

Help

100%

## Scenario 3 - Create an Azure Availability Set

An **Azure Availability Set** is a logical grouping of virtual machines (VMs) in Azure. When you create VMs within an **Availability Set** , the Azure platform distributes the placement of those VMs across the underlying infrastructure. During planned maintenance on the Azure platform, or in the event of an unexpected fault in the underlying hardware/infrastructure, **Availability Sets** ensure that at least one VM remains operational. To learn more about **Azure Availability Sets**. Lean more.

1. Click the green **+New** button.
2. On the **New** blade, type Availability Set in the Marketplace search window and press **Enter**.
3. Select **Availability Set** from the **Everything** blade.
4. On the **Availability Set** blade, click the **Create** button.
5. On the **Create availability set** blade, type demolabavaset1 in the name field.
6. Verify a **Subscription** is listed in the **Subscription** drop-down.
7. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(180).Name** resource group from the drop-down.
8. In the **Location** field, choose **South Central US**.
9. Verify that **Fault domains** is set to **2**.
   Virtual machines in the same **Fault domain** share a common power source and physical network switch.
10. Verify that **Update domains** is set to **5**.
    Virtual machines in the same **Update Domain** will be restarted together during planned maintenance. Azure never restarts more than one **Update Domain**at a time.
11. Click on **Yes (Aligned)** under **Use managed disks**.
    Azure Managed Disks simplifies disk management for Azure IaaS VMs by managing the storage accounts associated with the VM disks. You only have to specify the type (Premium

or Standard) and the size of disk you need, and Azure creates and manages the disk for you. Learn more

12. Click **Create**.
    This completes this scenario. Click on **Next** to go to the next scenario.

PreviousNext: Scenario 4 - Deploy a Virtual Machine

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

AZ00003: Azure Virtual Machine and Compute

1 Hr 51 Min Remaining

Instructions

Resources

Help

100%

## Scenario 4 - Deploy a Virtual Machine

In this part of the lab, we will deploy an **Azure Virtual Machine**using the foundational elements we created in Part I of this lab.

1. Click the green **+New** button.
2. On the **New** blade, click **Compute**.
3. On the **Featured** menu, choose the **Windows Server 2016 Datacenter** option.
4. On the **Basics** blade, type DEMOLABVM01 in the **Name**field.
5. Change the **VM disk type** drop-down menu option from **SSD** to **HDD**.
   It is recommended that if you migrate any virtual machine disk requiring high IOPS that you choose Azure Premium Storage to achieve the best performance for your applications. If your application does not require high IOPS, you can limit costs by maintaining it in Standard Storage, which stores virtual machine disk data on hard disk drives (HDDs) instead of solid-state drives (SSDs). Learn More.
6. In the **User Name** field, type labadmin.
7. In the **Password** and **Confirm password** fields, type Password2017.
8. Verify a **Subscription** is listed in the **Subscription** drop-down.
9. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(180).Name** resource group from the drop-down.
10. In the **Location** field, choose **South Central US**.
11. Leave **No** selected under **Already have a Windows Server License?**.
    If you own Windows Server licenses with active Software Assurance (SA), use Hybrid Use Benefit to save money. Learn More
12. Click the **OK** button.
13. On the **Choose a size** blade, click the **D1_V2 Standard** box.
    For the purposes of this lab, we are choosing a **Dv2**series VM as they are available across all regions. Learn More
14. Click **Select**.
15. On the **Settings** blade, choose **Yes** under **Managed disks**.
    Azure Managed Disks simplifies disk management for Azure IaaS VMs by managing the storage accounts associated with the VM disks. You only have to specify the type (Premium or Standard) and the size of disk you need, and Azure creates and manages the disk for you. Learn more
16. Click the **Virtual network** field.
17. Select the **demolabvnet1** virtual network on the **Choose virtual network** blade.
18. Verify that **demolabsubnet /26** is selected in the **Subnet**field.
19. Verify that the **Public IP address** field has a status that begins with **(new)**.

Use a public IP address if you want to communicate with the virtual machine from outside the Virtual Network. For example, if you will need to RDP to the VM, you will need a public IP address .

20. Verify that the **Network security group (firewall)** field has a status that begins with **(new)**.
    A **Network Security Group** is a set of firewall rules that control traffic to and from your virtual machine. Learn more.
21. Leave the **Extensions** field with **No extensions**.
    You can add features like Configuration Management or Antivirus Protection to your virtual machine using extensions. Learn more.
22. Click the **Availability set** field (at the top).
23. Click on **demolabavaset1**.
24. Back on the **Settings** blade, leave the defaults for **Boot diagnostics**, **Guest OS diagnostics**, and **Diagnostics storage account** under the **Monitoring** section.
    Azure Monitor enables you to consume telemetry to gain visibility into the performance and health of your workloads on Azure. The most important type of Azure telemetry data is the metrics (also called performance counters) emitted by most Azure resources. Learn more.
25. Click the **OK** button on the **Settings** blade.
26. The **Create** blade will be displayed with a **Validation Passed**message at the top of the blade.
27. Click **Create** to deploy the virtual machine that has been configured.
    You can check the status of the virtual machine deployment by clicking on the notification (Bell) icon at the top of the page. **(Deployment can take anywhere from a few minutes to about 15. Please wait for the Deployment to Succeed before heading to the next Scenario)**
    This completes this scenario. Click on **Next** to go to the next scenario.

PreviousNext: Scenario 5 - Backup and Recovery

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

AZ00003: Azure Virtual Machine and Compute

1 Hr 30 Min Remaining

Instructions

Resources

Help

100%

## Scenario 5 - Backup and Recovery

A virtual machine is protected by a locally redundant storage account, which means that it is replicated and only accessible to Microsoft in the event of a datacenter outage. In this lab, we will configure a separate backup of the VM that will be user-accessible for recovery.

1. From the **Dashboard** screen, click the expand menu icon
   from the top left to expand the **Favorites** menu if it is collapsed.
2. From the **Favorites** menu, click on the **Virtual machines**shortcut.
3. Click **DEMOLABVM01** on the **Virtual machines** blade.
4. In the **OPERATIONS** section, click **Backup**.
   This lab uses the **Azure Backup Recovery Services Vault** to back up the virtual machine with the configured policy, and will be charged as per-backup pricing. To learn more about **Azure Backup Recovery Services Vault** follow the link. Learn more.
5. On the **Enable backup** blade, ensure the **Create New** radio button under **Recovery**

**Services vault** is selected.
6. In the field below the **Create new** radio button, type demolabRSV.
7. In the **Resource group** field, change the radio button to **Use existing** and select the **@lab.CloudResourceGroup(180).Name** resource group from the drop-down.
8. Verify that the backup policy field is set to **(new) DailyPolicy**.
9. Click the **Enable Backup** button.
   **Note:** You can check the status of the **Backup Recovery Services Vault** deployment by clicking on the notification icon. Once deployment has succeeded you can go back into the backup area to review settings.
   Please click on the **Next** button to continue to the lab conclusion.

PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

AZ00003: Azure Virtual Machine and Compute

1 Hr 21 Min Remaining

Instructions

Resources

Help

100%

## Conclusion

The Virtual Machines & Compute lab has now been completed. With the completion of this lab, the following foundational elements have been reviewed, briefly explained, and used to successfully deploy an Azure based Virtual Machine.

- Resource Groups
- Virtual Networks
- Storage Accounts
- Availability Sets

For more information and follow up resources about key topics and elements discussed within this lab click associated links above.

PreviousEnd

From <https://labondemand.com/CloudClient/a7e40648-c68f-420e-ac33-7f9f74fe5981>

# ^AZ00004: Understanding Azure Networking

18 September 2018 15:25

AZ00004: Understanding Azure Networking

2 Hours Remaining

Instructions

Resources

Help

100%

## Understanding Azure Networking

*Learn Azure networking basics.*

### Overview

This lab provides an overview of Azure networking including VM network interfaces and virtual networks (VNets).

**The following topics will be covered in this lab:**

- Reviewing Azure VM Network Interface Options
- Creating an Azure Virtual Network (VNet)

**Capabilities or components used in this lab:**

- Azure Resource Manager (ARM)
- Virtual networks (VNets)

Next

From <https://labondemand.com/CloudClient/97549fa6-dc9c-49ee-a572-2882ba42ff29>

AZ00004: Understanding Azure Networking

2 Hours Remaining

Instructions

Resources

Help

100%

### Scenario A - Reviewing Azure VM Network Interface Options

A network interface enables an Azure VM to communicate with Internet, Azure, and on-premises resources. When creating a virtual machine using the Azure portal, the portal creates one network interface with default settings for you. You may instead choose to create network interfaces with custom settings and add one or more network interfaces to a virtual machine when you create it. You may also want to change default network interface settings for an existing network interface.

In this lab, you will review Azure VM network interface configuration options.

1. Login to Azure Portal
2. In the **Favorites** menu, click **Virtual machines**
   **Note:** A virtual machine named **BackendVM1** has already been created for this lab.
3. On the **Virtual machines** blade, click **BackendVM1**
4. On the **BackendVM1** blade in the right pane, note that you can view the following information about the VM:
    - Resource group

    - Status
    - Location
    - Subscription
    - Subscription ID
    - Computer name
    - Operating system
    - Size
    - Public IP address (if assigned)
    - Virtual network/subnet
    - DNS name (if assigned)
5. In the left pane, click **Networking** under **SETTINGS**
6. On the **BackendVM1 - Networking** blade, click **BackendVM1nic** in the right pane
7. On the **BackendVM1nic** blade under **Essentials** in the right pane, note that you can view the following information about the network interface:
    - Resource group
    - Location
    - Subscription
    - Subscription ID
    - Private IP address
    - Virtual network/subnet
    - Public IP address (if assigned)
    - Network security group (if assigned)
    - VM attached to
8. In the left pane under **SETTINGS**, click **IP configurations**
9. On the **BackendVM1nic - IP configurations** blade in the right pane, click **ipconfig1** in the table under **IP configurations**
10. On the **ipconfig1** blade under **Public IP address settings**, verify that **Public IP address** is **Disabled**
    **Note:** You can add a public IP address by clicking **Enabled** under **Public IP address settings** and entering the required information. To add a public IP address, you must have a registered public IP address associated with your Azure subscription. There is no public IP address associated with the subscription for this lab.
11. Under **Private IP address settings**, leave **Dynamic** selected under **Assignment**
    **Note:** IP addresses must be unique on your network. If you attempt to assign a duplicate IP address, the **IP address** field will turn red when you click **Save** to indicate an IP address conflict. When you change an IP address, the virtual machine associated with the network interface will be restarted to utilize the new private IP address. The network interface will be re-provisioned and network configuration settings, including secondary IP addresses, subnet masks, and default gateway, will need to be manually reconfigured within the virtual machine.
12. Click the **X** near the top right corner of the blade to close the **ipconfig1** blade
13. On the **BackendVM1nic - IP configurations** blade in the left pane under **SETTINGS**, click **DNS servers**
14. On the **BackendVM1nic - DNS servers** blade, note that you can select to inherit DNS server settings from the virtual network (default) or provide an IP address of a DNS server in your network which may already exist, such as your domain controller setup as a DNS server

In the next section we will review how to create a new Azure Virtual Network and associate it with

a new Virtual Machine.
PreviousNext

From <https://labondemand.com/CloudClient/fd83cc9c-fe62-43ca-ae89-7626906d357e>

AZ00004: Understanding Azure Networking

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario B - Creating an Azure Virtual Network (VNet)

An **Azure virtual network (VNet)** enables Azure resources to securely communicate with each other in a virtual network. A VNet is a representation of your own network in the cloud and is a logical isolation of the Azure cloud dedicated to your subscription. You can connect VNets to other VNets and to your on-premises network.
In this lab, you will learn how to create an Azure virtual network (VNet) and associate a new VM with the VNet.

1. In the **Favorites** menu, click **Virtual networks**
2. Make note of the **Location** of the **corpnet**virtual network, this will be used later in this lab
3. On the **Virtual networks** blade, click **Add**near the top left of the blade
4. On the **Create virtual network** blade under **Name**, type **myVNet**
5. Under **Address space**, type **10.3.0.0/16**
6. In the **Subscription** dropdown menu, select the available subscription
7. Under **Resource group**, click the **Use existing** radio button and select **@lab.CloudResourceGroup(757).Name** in the dropdown menu
8. In the **Location** dropdown menu, select **South Central US**
9. Under **Subnet** under **Name**, type **mySubnet-1**
10. Under **Address range**, type **10.3.1.0/24**
    **Note:** A virtual network has already been created for this lab, so you do not need to click **Create** to complete the virtual network deployment.
11. Click the **X** near the top right corner of the blade to close the **Create virtual network**blade
12. Click **OK** in the popup window to discard your unsaved edits
13. In the **Favorites** menu, click **Virtual machines**
14. On the **Virtual machines** blade, click **Add**near the top left corner of the blade
15. On the **Compute** blade in the **Search Compute** box, type **windows server 2016 datacenter** and press **Enter**
16. In the **Results** list, click **Windows Server 2016 Datacenter**
17. On the **Windows Server 2016 Datacenter**blade, click **Create**
18. On the **Create virtual machine - Basics**blade under **Name**, type **myVM1**
19. In the **VM disk type** dropdown menu, leave the default selection **(SSD)**
20. Under **User name** type **labadmin**
21. Under **Password** and **Confirm password**type **Password2018**
    **Note:** Do not use weak passwords when actually creating VMs.
22. In the **Subscription** dropdown menu, select the available subscription
23. Under **Resource group**, change the radio button to **Use existing** and select **@lab.CloudResourceGroup(757).Name** in the **Resource group** dropdown menu
24. In the **Location** dropdown menu, select **South Central US**
25. Click **OK**
26. On the **Choose a size** blade, search for and select **DS1_V2 Standard**
27. Click **Select**

28. On the **Settings** blade under **High availability** click **None** under **Availability set**
29. On the **Change availability set** blade, click **Create new**
30. On the **Create new** blade under **Name**, type **myAS**
31. Under **Fault domains** leave the default setting **(2)**
32. Under **Update domains** change the setting to **2**
33. Click **OK**
34. On the **Settings** blade under **Storage**, leave the **Use managed disks** default setting **(Yes)**
35. Under **Network**, click **Virtual network**
36. On the **Choose virtual network** blade, click **corpnet**
37. Click the **X** near the top right corner of the **Settings** blade to close the blade
38. Click **OK** in the popup window to discard your unsaved edits
39. Click the **X** near the top right corner of the **Create virtual machine** blade to close the blade
40. Click **OK** in the popup window to discard your unsaved edits
    **Note:** When you create a new VM, you can create a new VNet or associate it with an existing VNet. In the preceding steps you learned how to create a VNet, then associated a new VM with the VNet while creating the VM. You have completed this lab so you do not need to continue creating the VM you started in the previous steps.

PreviousNext

From <https://labondemand.com/CloudClient/e79c7a02-39e8-4b90-9bd2-630c874b02d9>

AZ00004: Understanding Azure Networking

2 Hours Remaining

Instructions

Resources

Help

100%

## Conclusion

In this lab, you learned about Azure VM network interface configuration options and Azure virtual networks (VNets).

PreviousEnd

From <https://labondemand.com/CloudClient/e79c7a02-39e8-4b90-9bd2-630c874b02d9>

# ^AZ00005: Deploy Network Security Groups and Load Balancers

19 September 2018 10:26

AZ00005: Deploy Network Security Groups and Load Balancers

## Understanding Azure Networking

*Deploy and configure Azure networking resources including network security groups (NSGs), load balancers, and application gateways.*

### Overview

This lab provides an introduction to deploying and configuring Azure network security groups (NSGs), load balancers, and application gateways.
**The following topics will be covered in this lab:**

- Creating a Network Security Group (NSG) and Inbound Security Rule
- Deploying an Internal Load Balancer
- Creating an Application Gateway

**Capabilities or components used in this lab:**

- Azure Resource Manager (ARM)
- Network security groups (NSGs)
- Load balancers
- Application gateways

### Scenario A - Creating a Network Security Group (NSG) and Inbound Security Rule

A **network security group (NSG)** contains a list of security rules that allow or deny network traffic to resources connected to Azure VNets. NSGs can be associated to subnets, individual VMs (classic), or individual network interfaces attached to VMs (Resource Manager). When an NSG is associated to a subnet, the rules apply to all resources connected to the subnet. Traffic can further be restricted by also associating an NSG to a VM or network interface.
In this lab, you will create a network security group (NSG) and an inbound security rule to allow RDP access.

1. Login to Azure Portal
2. In the menu on the left side of the Azure portal, click **Create a resource**
3. In the **Search the Marketplace** box, type **network security group** and press **Enter**
4. On the **Everything** blade, click **Network security group** in the **Results** list
5. On the **Network security group** blade, leave **Resource Manager** selected in the **Select a deployment model** dropdown menu
6. Click **Create**
7. On the **Create network security group**blade under **Name**, type **backend-nsg**
8. In the **Subscription** dropdown menu, select the available subscription
9. Under **Resource group**, change the radio button to **Use existing** and select **@lab.CloudResourceGroup(760).Name** in the **Resource group** dropdown menu
10. In the **Location** dropdown menu, select the location **South Central US**
11. Click **Create**
    **Note:** The NSG will take several minutes to deploy in Azure. You can track the deployment progress in the Azure portal by clicking on the **Notifications** icon
    .
12. In the menu on the left side of the Azure portal, click **All services**
13. On the **All services** blade under **NETWORKING**, click the **Favorite** icon

next to **Network security groups** to add it to your **Favorites** menu

14. Click **Network security groups**
15. On the **Network security groups** blade, click **backend-nsg**
16. On the **backend-nsg** blade under **SETTINGS**, click **Inbound security rules**
17. On the **backend-nsg - Inbound security rules** blade, note the following default rules:
    - **AllowVnetInBound** (permits source traffic from the virtual network to destinations in the virtual network)
    - **AllowAzureLoadBalancerInBound**(permits Azure load balancers to send health probes)
    - **DenyAllInBound** (blocks all traffic that is not explicitly permitted by another rule)
18. Click **Add** near the top of the blade
19. On the **Add inbound security rule** blade, click **Basic** near the top left corner of the blade
20. In the **Service** dropdown menu, select **RDP**
    **Note:** The **service** specifies the destination protocol and port range for this rule. You can choose a predefined service from the dropdown menu or select **Custom** and specify a **Protocol**(*Any, TCP, or UDP*) and **Port range** (*single port, range of ports, or any port*).
21. Under **Priority**, leave the default value **(100)**
    **Note:** Rules are processed in priority order; the lower the number, the higher the priority. Each rule must be assigned a unique priority number between 100 and 4096. It is highly recommended that you leave gaps between rules (for example, 100, 200, 300, etc.) so that it's easier to add new rules without having to edit existing rules.
22. Under **Name**, type **rdp-rule**
23. Near the top left corner of the **Add inbound security rule** blade, click **Advanced**
24. Under **Source** select **Any**
    **Note:** The **Any** filter permits/blocks traffic from any source IP address. The **CIDR block** filter permits/blocks traffic from an IP address range. The **Tag** filter permits/blocks traffic from a default Azure tag (such as *Internet, VirtualNetwork, or AzureLoadBalancer*).
25. Under **Source port ranges** leave the default wildcard **( * )** entry
26. Under **Destination** select **Any**
27. Under **Destination port ranges** leave the default port **(3389)** entry
28. Under **Protocol** leave the default selection **(TCP)**
29. Under **Action** click **Allow**
30. Click **Add**
    **Note:** The inbound security rule will take several minutes to deploy in Azure. You can track the deployment progress in the Azure portal by clicking on the **Notifications** icon
    .
31. After the deployment has succeeded, note the **rdp-rule** has been added to the list on the **backend-nsg - Inbound security rules** blade

In the next section we will review deploying internal load balancers.

## Scenario B - Deploying an Internal Load Balancer

**Azure load balancer** delivers high availability and network performance to your applications. It is a Layer 4 (TCP, UDP) load balancer that distributes incoming traffic among healthy instances of services defined in a load-balanced set.
In this lab you will deploy an internal load balancer to distribute inbound SQL cluster (port 1433) traffic.

1. In the **Favorites** menu, click **Virtual networks**
2. On the **Virtual networks** blade, note the location of the **corpnet** VNet
3. In the **Favorites** menu, click **Load balancers**

4. On the **Load balancers** blade, click **Add** near the top left corner of the blade
5. On the **Create load balancer** blade under **Name** near the top, type **webintlb1**
6. Under **Location**, select **South Central US**
   **Note:** The load balancer must be created in the same location as the vNet for the VM resources you want to deploy with the load balancer.
7. Under **Type**, select **Internal**
8. Under **SKU**, leave the default setting **(Basic)**
9. Under **Virtual network**, click **Choose a virtual network**
   **Note:** A **virtual network** named **corpnet** and associated subnet has already been created for this lab.
10. On the **Choose a virtual network** blade, click **corpnet**
11. On the **Create load balancer** blade under **Subnet**, click **Choose a subnet**
12. On the **Choose subnet** blade, click **Subnet-2**
13. On the **Create load balancer** blade under **IP address assignment**, select **Static**
14. In the **Private IP address field**, type **10.1.1.110**
15. Under **Subscription**, select the available subscription
16. Under **Resource group**, change the radio button to **Use existing** and select **@lab.CloudResourceGroup(760).Name** in the **Resource group** dropdown menu
17. Click **Create**
18. In the **Favorites** menu, click **Load balancers**
19. On the **Load balancers** blade, click **webintlb1** in the list
20. On the **webintlb1** blade under **SETTINGS**, click **Frontend IP configuration**
21. On the **webintlb1 - Frontend IP configuration** blade, note the **LoadBalancerFrontEnd** in the list in the right pane
    **Note:** The **LoadBalancerFrontEnd** and associated **IP address** were configured when you created the load balancer.
22. Under **SETTINGS** in the left pane, click **Backend pools**
23. On the **webintlb1 - Backend pools** blade, click **Add** near the top left corner of the right pane
24. On the **Add backend pool** blade under **Name**, type **backendpool1**
25. In the **Associated to** dropdown menu, select **Availability set**
    **Note:** An **availability set** named **backendAS** has already been created for this lab.
26. Under **Availability set**, select **backendAS**
27. Click **+ Add a target network IP configuration**
    **Note:** Virtual machines named **BackendVM0** and **BackendVM1** have already been created for this lab.
28. Under **Target virtual machine**, select **BackendVM0**
29. Under **Network IP configuration**, select **ipconfig1**
30. Click **Add a target network IP configuration**
31. Under **Target virtual machine**, select **BackendVM1**
32. Under **Network IP configuration**, select **ipconfig1**
33. Click **OK**
    **Note**: Wait for the backend pool to finish. You can monitor the status from the notifications menu at the top of the Azure portal.
    🔔
34. On the **webintlb1 - Backend pools** blade in the right pane, click the **right arrow** icon
    ▶
    next to **backendpool1** in the list to view additional information about the virtual machines in the pool
35. On the left pane under **SETTINGS**, click **Health probes**
36. On the **webintlb1 - Health probes** blade, click **Add** near the top left corner of the right pane
37. On the **Add health probe** blade under **Name**, type **probe1**

38. Select **TCP** under **Protocol**
39. Change **Port** to **1433**
40. Change **Interval** to **60**
41. Change **Unhealthy threshold** to **3**
42. Click **OK**
    **Note**: Wait for the health probe to finish deploying. You can monitor the status from the notifications menu at the top of the Azure portal.
43. On the left pane under **SETTINGS**, click **Load balancing rules**
44. On the **webintlb1 - Load balancing rules**blade, click **Add** near the top left corner of the right pane
45. On the **Add load balancing rule** blade under **Name**, type **sqlsvr-rule**
46. Under **IP Version**, verify **IPv4** is selected
47. Verify the **(LoadBalancerFrontEnd)** IP address is selected in the **Frontend IP address** dropdown menu
48. Verify **Protocol** is **TCP**
49. Under **Port**, type **1433**
50. Under **Backend port**, type **1433**
51. In the **Backend pool** dropdown menu, verify **backendpool1** is selected
52. In the **Health probe** dropdown menu, verify **probe1** is selected
53. In the **Session persistence** dropdown menu, verify **None** is selected in
    **Note:** Session persistence specifies that traffic from a client should be handled by the same virtual machine in the backend pool for the duration of a session. **None**specifies that successive requests from the same client may be handled by any virtual machine. **Client IP** specifies that successive requests from teh same client IP address will be handled by the same virtual machine. **Client IP and protocol**specifies that successive requests from the same client IP address and protocol combination will be handled by the same virtual machine.
54. Use the slider bar to increase the **Idle timeout (minutes)** to **20**
55. Verify **Floating IP (direct server return)** is **Disabled**
    **Note:** The floating IP feature is recommended only when configuring a SQL AlwaysOn Availability Group Listener. It can only be enabled when creating a rule and if the port and backend port match.
56. Click **OK**

In the next section we will review how to create an Azure application gateway.

## Scenario C - Creating an Application Gateway

**Azure application gateway** is a dedicated virtual appliance providing application delivery controller (ADC) as a service. It offers various Layer 7 load balancing capabilities for your application. It allows customers to optimize web farm productivity by offloading CPU-intensive SSL termination to the application gateway. It also provides other Layer 7 routing capabilities including round robin distribution of incoming traffic, cookie-based session affinity, URL path-based routing, and the ability to host multiple websites behind a single application gateway.
A web application firewall (WAF) is also provided as part of the application gateway WAF SKU. It provides protection to web applications from common web vulnerabilities and exploits.
Application gateway can be configured as an Internet-facing gateway, internal-only gateway, or a combination of both.
In this lab you will deploy a basic application gateway.

1. In the menu on the left side of the Azure portal, click **All services**
2. On the **All services** blade, type **application gateways** in the search box and press **Enter**
3. Click **Add** near the top left corner of the **Application gateways** blade

4. On the **Basics** blade located to the right of the **Create application gateway** blade, type **appgw1** under **Name**
5. Under **Tier**, select **Standard**
6. Under **SKU size**, select **Small**
7. Leave the **Instance count** set to **2**
8. Under **Subscription**, select the available subscription
9. Under **Resource group**, change the radio button to **Use existing** and select **@lab.CloudResourceGroup(760).Name** in the **Resource group** dropdown menu
10. Under **Location**, select **South Central US**
    **Note:** Your virtual network and public IP address must be in the same location as your gateway. If you plan on using existing resources, ensure that you select the correct location .
11. Click **OK**
12. On the **Settings** blade under **Subnet configuration**, click **Choose a virtual network** under **Virtual network**
13. On the **Choose virtual network** blade, click **corpnet**
14. In the **Subnet** dropdown menu, select **appgw-subnet(10.1.2.0/24)**
15. Under **Frontend IP configuration**, leave the default **IP address type** setting **(Public)**
16. Under **Public IP address**, select **Create new**and type **appgw1-ip**
17. Under **SKU**, leave the default setting **(Basic)**
18. Under **Listener configuration**, leave the default **Protocol** setting **(HTTP)**
19. Under **Port**, leave the default setting **(80)**
20. Click **OK**
21. On the **Summary** blade, click **OK**
    **Note:** The application gateway will take several minutes to deploy in Azure.
22. Click the **Notifications** icon
    in the top menu bar to open the **Notifications** window
23. When the deployment completes, click **Go to resource** in the popup window
24. On the **appgw1** blade under **SETTINGS**, click **Backend pools**
25. On the **appgw1 - Backend pools** blade in the right pane, click **appGatewayBackendPool**
26. Under Targets, select **Virtual machine**
    **Note:** Virtual machines named **FrontendVM0** and **FrontendVM1** have already been created for this lab.
27. In the **Virtual machine** dropdown menu, select **FrontendVM0**
28. Next to **FrontendVM0**, under **NETWORK INTERFACES**, select **FrontendVM0nic**
29. Click *Select a virtual machine* to add another virtual machine and select **FrontendVM1**
30. Next to **FrontendVM1**, under **NETWORK INTERFACES**, select **FrontendVM1nic**
31. Click **Save**
32. Under **SETTINGS**, click **Listeners**
    **Note:** The **appGatewayHttpListener** is automatically created when you create an application gateway. A listener directs traffic to the appropriate backend pool based on the configured rules.
33. Under **SETTINGS**, click **Rules**
34. On the **appgw1 - Rules** blade, note that a default rule named **rule1** is automatically created
    **Note:** When you create a basic rule, you select a **Listener** and a **Backend pool**. A **basic** rule implements port forwarding. You can also do **path-based** routing rules.

## Conclusion

In this lab, you created an NSG and inbound security rule to allow RDP connections, deployed an internal load balancer for a SQL Server pool, and created an application gateway for an application server pool.

# AZ00006: Azure Web App Service

19 September 2018    10:36

AZ00006: Azure Web App Service

2 Hours Remaining

Instructions

Resources

Help

100%

## Managing Web Applications in Azure

*Explore and deploy Azure Web App service*

### Overview

This lab provides foundational knowledge and hands-on experience for deploying and managing web applications in Azure. The Web Apps feature is a part of the Azure App Service offering which allows developers to quickly build, deploy, manage, and host web applications in the Azure Platform as a Service (PaaS). These applications can include small to enterprise-scale websites and web applications for mobile and full desktop experience. The following objectives will be covered in this lab:

- Deployment of a web application
- Management of a web application
- Introduction to the high availability and scalability concepts in web applications
- Introduction to the Azure Marketplace

**Capabilities or components used in this scenario:**

- **Azure:** Azure Resource Manager (ARM), App Service, Web App

Next: Scenario 1 – Building a...

From <https://labondemand.com/CloudClient/e426b357-943e-48bc-a926-6595f59c6ab6>

AZ00006: Azure Web App Service

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario 1 – Building a Web App Service

**Web Apps** is a fully managed compute platform that is optimized for hosting websites and web applications. This platform-as-a-service (PaaS) offering of Microsoft Azure lets you focus on your business logic while Azure takes care of the infrastructure to run and scale your apps. Learn more.

**Sign in to the Azure portal**

1. In the menu on the left, click **Create a resource**
   **Note:** If the menu on the left side of the portal is collapsed (no text labels appears beside the icons), click **Show text labels**

to expand the menu

2. On the **New** blade under the **Azure Marketplace** menu, click **Web**
3. Under **Featured** on the right side of the blade, click **Web App**
4. On the **Web App** blade under **App name**, enter a unique name like labapp7322382
   **Note:** Once you enter a name that is available, you will see a green check mark on the right side of the **App name** box. Web app names must be unique across all existing Web app names in Azure. It must be 3-24 characters long. The URL to your Web App will use the domain name .azurewebsites.net. For example, the URL for your Web App will be similar to http://webapp123.azurewebsite.net
5. In the **Subscription** dropdown menu, verify a subscription is selected
6. Under **Resource group**, change the radio button to **Use existing** and select **@lab.CloudResourceGroup(761).Name** in the dropdown menu
7. Leave the default OS selection (**Windows**)
8. Click **App Service plan/Location**
   **Note:** An App Service plan is the container for your app. The App Service plan settings will determine the location, features, cost, and compute resources associated with your app. Click here to learn more about Azure App Service plans.
9. On the **App Service plan** blade, click **Create new**
10. On the **New App Service Plan** blade, under **App Service plan** type labwebappplan7322382
11. In the **Location** dropdown menu, choose the location that is geographically nearest to you
12. Click **Pricing tier**
13. On the **Choose your pricing tier** blade, click **S1 Standard**
14. Click **Apply**
15. On the **New App Service Plan** blade, click **OK**
16. On the **Web App** blade, change the **Application Insights** selection to **On**
    **Note:** Application Insights helps you detect and diagnose quality issues in your .NET web apps and web services, and helps you better understand what your users actually do with your app.
17. In the **Application Insights Location**dropdown menu, leave the default selection
18. Click **Create**

This completes this scenario. Click **Next** to go to the next scenario.
PreviousNext: Scenario 2: Management of...

From <https://labondemand.com/CloudClient/e426b357-943e-48bc-a926-6595f59c6ab6>

AZ00006: Azure Web App Service

2 Hours Remaining

Instructions

Resources

Help

100%

# Scenario 2: Management of a Web Application

In this scenario, you will navigate to the web app you created in Scenario 1 of this lab and view its configuration and settings. You will also configure the scalability options and setup Application Insights to monitor your Web App.

## Web App configuration and settings

1. In the **Favorites** menu, click **App Services**
2. On the **App Services** blade, click the **Name**of the Web App that you created in the

previous scenario

3. On the **App Service** blade in the right pane, click the **link** under **URL**
   **Note:** This will open a new browser tab and display a web site that says **Your App Service app is up and running"** which confirms that the web app is available over the public Internet.
4. Click the **Azure portal** browser tab
5. On the **App Service** blade in the left pane under **DEPLOYMENT**, click **Deployment credentials**
   **Note:** Git and FTP cannot authenticate using the account you are signed into Azure with, so you will need to create a new user name and password to use with Git and FTP.
6. Under **FTP/deployment Username**, type a unique username, for example ftpadminlabapp7322382
7. Under **Password** and **Confirm password**, type SPftpadmin01
8. Click **Save**
9. In the left pane under **DEPLOYMENT**, click **Deployment slots**
   **Note:** Deployment slots let you deploy different versions of your web app to different URLs. You can test a particular version and then swap content and configuration between slots.
10. On the **Deployment slots** blade in the right pane, click **Add Slot** near the top of the blade
11. On the **Add a slot** blade under **Name**, type labstaging
    **Note:** This value will be appended to the URL of your main web app and will serve as the public address for the slot. For example, if you have a web app named **contoso** and a slot named **labstaging**then the new slot will have the following URL: **https://contoso-labstaging.azurewebsites.net**
12. In the **Configuration Source** dropdown menu, leave the default selection (**Don't clone configuration from an existing slot**)
13. Click **OK**
14. Near the top left corner of the browser, click **Microsoft Azure** to return to the main Azure Portal page
15. If a popup window appears, click **OK** to discard your unsaved edits

## Advanced Configuration Steps (Optional)

Perform these steps to configure the following advanced features:

- Enable Java support
- Add a default document type

## Enable Java Support and additional default document types

1. In the **Favorites** menu, click **App Services**
2. On the **App Services** blade, click the **Web App** that you created in Scenario 1 of this lab
3. On the **App Service** blade in the left pane under **SETTINGS**, click **Application settings**
4. In the right pane under **General settings** in the **Java version** dropdown menu, select **Java 7**
5. Scroll down in the right pane to the **Default documents** section
6. At the bottom of the **Default documents**section, click **Add new document**
7. In the **Enter a name** box, type start.html
8. Click **Save** at the top of the blade

This completes this scenario. Click **Next** to go to the next scenario.

PreviousNext: Scenario 3: Introduction...

From <https://labondemand.com/CloudClient/e426b357-943e-48bc-a926-6595f59c6ab6>

AZ00006: Azure Web App Service

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario 3: Introduction to high availability and scalability concepts in web applications

In this scenario, you will review high availability and scalability options and upgrade your web app to make it more scalable and highly available.

### Changing Scale Up and Scale Out Options

1. In the **Favorites** menu, click **App Services**
2. On the **App Services** blade, click the **Name**of the web app that you created in Scenario 1 of this lab
3. On the **App Service** blade in the left pane under **SETTINGS**, click **Scale up (App Service plan)**
4. On the **Choose your pricing tier** blade, click **P1 Premium**
   **Note:** There are many options to help you scale up in the event that additional resources are needed. The **P1 Premium**option enables you to have up to 20 instances as needed. This is an increase from the 10 instances deployed with the **S1 Standard** option. Learn more
5. Click **Apply**
6. On the **App Service** blade under **SETTINGS**, click **Scale out (App Service plan)**
7. On the **Scale out (App Service plan)** blade, move the **Instance count** slider bar to **10**
   **Note:** A scale-out operation is the equivalent of creating multiple copies of your web site and adding a load balancer to distribute the demand across them. When you scale out a web site in Windows Azure Web Sites, there is no need to configure load balancing separately since it is already provided by the platform.
8. Click **Save** at the top of the blade

This completes this scenario. Click **Next** to go to the next scenario.

PreviousNext: Scenario 4: Application...

From <https://labondemand.com/CloudClient/e426b357-943e-48bc-a926-6595f59c6ab6>

AZ00006: Azure Web App Service

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario 4: Application Insights

Application Insights helps you detect and diagnose quality issues in your web apps and web services, and helps you better understand what your users actually do with your web apps. Learn more
Follow these steps to configure the following settings:

- Availability ping test
- Alerts
- Application Map

## Configure Availability Ping Test

1. In the **Favorites** menu, click **App Services**
2. On the **App Services** blade, click the **Name**of the web app that you created in Scenario 1 of this lab
3. On the **App Service** blade in the left pane under **SETTINGS**, click **Application Insights**
4. On the **Application Insights** blade, click **VIEW MORE IN APPLICATION INSIGHTS**near the bottom right corner of the blade
5. In the left pane under **INVESTIGATE** click **Availability**
6. On the **Availability** blade, click **Add test** at the top of the blade
7. On the **Create test** blade under **Test name**, type Ping test
8. In the **Test type** dropdown menu, select **URL ping test**
9. Under **URL**, verify the URL to your web app is displayed
10. Leave the default selections for the remaining settings and click **Create**
11. In the top left corner of the browser window, click **Microsoft Azure** to return to the main Azure Portal page
12. If a popup window appears, click **OK** to discard your unsaved edits

## Configure Alerts

1. In the **Favorites** menu, click **App Services**
2. On the **App Services** blade, click the **Name**of the web app that you created in Scenario 1 of this lab
3. On the **App Services** blade in the left pane under **SETTINGS**, click **Application Insights**
4. On the **Application Insights** blade, click **VIEW MORE IN APPLICATION INSIGHTS**near the bottom right corner of the blade
5. In the left pane under **CONFIGURE**, click **Alerts (classic)**
6. On the **Alerts (classic)** blade, click **Add metric alert (classic)** at the top of the blade
7. On the **Add rule** blade under **Name**, type Alert1
8. In the **Resource** dropdown menu, select the name of your web app
9. In the **Metric** dropdown menu, select **Availability**
10. In the **Condition** dropdown menu, select **Less than**
11. Under **Threshold**, leave the default setting (**1**)

12. Leave the default selections for the remaining settings and click **OK**
13. In the top left corner of the browser window, click **Microsoft Azure** to return to the main Azure Portal page
14. If a popup window appears, click **OK** to discard your unsaved edits

## Application Map

In Azure Application Insights, **Application Map** is a visual layout of the dependency relationships of your application components. Each component shows KPIs such as load, performance, failures, and alerts, to help you discover any component causing a performance issue or failure. You can click through from any component to more detailed diagnostics, such as Application Insights events. If your app uses Azure services, you can also click through to Azure diagnostics, such as SQL Database Advisor recommendations. Learn more

### Enabling Application Insights Monitoring

1. In the **Favorites** menu, click **App Services**
2. On the **App Services** blade, click the **Name**of the web app that you created in Scenario 1 of this lab
3. On the **App Service** blade in the left pane under **SETTINGS**, click **Application Insights**
4. On the **Application Insights** blade, click **VIEW MORE IN APPLICATION INSIGHTS**near the bottom right corner of the blade
5. In the left pane under **INVESTIGATE**, click **Application map**
6. On the **Application map** blade, click **Refresh**near the top of the blade
7. On the **Application map** blade in the right pane, review the details of the alerts and tests performed

This completes the lab. Click **Next** to go to the Conclusion.
PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/e426b357-943e-48bc-a926-6595f59c6ab6>

AZ00006: Azure Web App Service

2 Hours Remaining
Instructions

Resources

Help
100%

## Conclusion

In this lab you created a simple Web App for hosting a web site and configured monitoring and scalability options in the platform.
Web App Services
Azure Application Insights
PreviousEnd

From <https://labondemand.com/CloudClient/e426b357-943e-48bc-a926-6595f59c6ab6>

# AZ00007: Understanding Azure Resource Manager Templates

19 September 2018 10:40

AZ00007: Understanding Azure Resource Manager Templates

2 Hours Remaining

Instructions

Resources

Help

100%

## Understanding Azure Resource Manager Templates

## Overview

This lab will guide the user through the structure and use of Azure Resource Manager (ARM) templates. It will walk the user through the process of loading an ARM template into Visual Studio, explain various sections of the template, and show the user how to run a simple pre-configured template in Azure.

### Lab Goals

- Understand ARM template structure and schema, including:
    - Parameters
    - Variables
    - Resources
- Understand PowerShell-driven Deployment
- Deploy PaaS resources via an ARM template
- Deploy IaaS resources via an ARM template

### Overview

ARM templates have a core structure which includes various components. The components include:

- **Schema** - All templates require an ARM schema reference in order for ARM to validate the template. ARM template schema's are hosted at https://github.com/Azure/azure-resource-manager-schemas.git.
- **Content version** - This component is used by the developer to keep track of which version of the template is current.
- **Parameters** - Parameters are pieces of information which ARM needs to know about. Examples are names of VM's, regions, resource groups, etc.
- **Variables** - Variables contain details which ARM can determine on its own, for example resource ID's of resources created in Azure.
- **Resources** - The Resources component is the main component of the ARM template and contains all the resources that the template will create and deploy.
- **Outputs** - Outputs are information you want to pass from the template to other

applications such as PowerShell.

ARM template parameters have a specified structure:

- **name** - The name of the parameter and how it is referred to in other components of the ARM template.
- **type** - Defines the type of value for the parameter (for example, string, integer, array, etc.).
- **defaultValue** - The value to be used if no value has been provided.
- **allowedValues** - A list of values that are allowed as the value property.
- **Metadata** - A description of what this parameter is used for in the ARM template.

ARM template variables are optional pieces of information that are part of the ARM template and can be determine by ARM during the deployment process.
ARM template resources have the same structure:

- **Type** - Defines the type of resource that will be created.
- **Name** - Identifies the name of resource.
- **apiVersion** - Identifies which version of the API should be used during the creation of this resource.
- **Location** - The geographical location where the resource is created.
- **Properties** - The list of properties required by the ARM template to create the resource.

**Capabilities or components used in this scenario:**

- PowerShell ISE
- PaaS Resource Deployment
- IaaS Resource Deployment
- Azure Portal

Next

From <https://labondemand.com/CloudClient/2f1cb961-0ef3-4ec4-87c8-cb8b81f1fcf9>

AZ00007: Understanding Azure Resource Manager Templates

2 Hours Remaining

Instructions

Resources

Help

100%

## Understanding Azure Resource Manager Templates

In this lab, you will create simple Azure Resource Manager templates for both a Platform as a Service (PaaS) scenario and an Infrastructure as a Service (IaaS) scenario. You will deploy the ARM templates using PowerShell and verify their deployment through the Azure portal.

### Scenario A - Deploy a PaaS Resource

In this scenario, you will create a simple template to deploy a PaaS resource into Azure. To get started we will RDP into a jumphost where we have pre-loaded PowerShell along with ARM templates that we'll be working with.

1. In this window (the guide application) click the **Resources** tab up above, then click **Jumphost RDP Profile** to download the jumphost RDP profile. Usually this will download to your downloads folder
2. Open the downloaded RDP profile, copy and paste @lab.CloudResourceGroup(769).Name.southcentralus.cloudapp.azure.cominto

the **Computer** field and click **Connect**, on first connect you will need to click the checbox [ ] Don't ask me again for connections to this computer
3. For the password, copy and paste k!DgeHA70$ and click **Connect**
4. On the next dialog click [ ] Don't ask me again for connection to this computer then click [Yes], this will bypass certificate issues
5. In the jumphost, click the **Start** button, type **Powershell ISE** and select the **Windows PowerShell ISE Desktop App**
note: may take up to 1minute to load initially.
6. In **PowerShell ISE**, click **File > Open**
7. In the **Documents** folder, open the **HOL_Lab_Files** folder
8. Open the **1_Understanding_Azure_Resource_Manager_Templates**folder, and double-click the PaaS PowerShell file **Deploy-SimplePaaS**
9. Next we need to establish the Azure Account we want to run our powershells scripts under. In the **blue PowerShell window**, at the bottom paste the following line and press **Enter** to launch the Azure login:
```
Add-AzureRmAccount
```
**Note:** If the blue window at the bottom isn't viewable you may need to expose the bottom window by dragging up the horizontal bar to see the blue PowerShell command window.
10. In the login screen's **Email or phone** field, enter User1-7322410 @cloudplatimmersionlabs.onmicrosoft.comand click **Next**
11. In the **Password** field, enter k!DgeHA70$and click **Sign in**
**Note:** Now you will define some variables in the PowerShell file. You should not hard code variables as part of an ARM template because it makes it more difficult to re-use the template. Variables can be referenced throughout the ARM template using the 'variables' object.
12. In the open **Deploy-SimplePaas.ps1** file, replace **[resource group name]** with @lab.CloudResourceGroup(769).Name
13. Replace **[resource group location]** with South Central US
14. In the **Define Deployment Variables** section of the **Deploy-SimplePaas.ps1 file**, highlight the following nine lines and click **Run selection**
```
$resourceGroupName...
$resourceProviderNamespace...
$resourceTypeName...
$resourceGroupLocation...

$randomString = ([char[]]([char]'a'..[char]'z') + 0..9 | Sort-Object
{Get-Random})[0..8] -join ''
$appNamePrefix = 'contoso'
$appServicePlanName = $appNamePrefix + $randomString
$webAppName = $appNamePrefix + $randomString
```
**Note:** You have now established definitions for variables that will be used later in the deployment script.
15. In the **Create App Service Plan** section of the **Deploy-SimplePaas.ps1 file**, highlight the following seven lines and click **Run selection**
```
$appServicePlan = New-AzureRmAppServicePlan `
    -ResourceGroupName $resourceGroupName `
    -Location $resourceGroupLocation `
    -Name $appServicePlanName `
    -Tier Standard `
    -WorkerSize Small `
    -Verbose
```
**Note:** You have now deployed an App Service Plan into the @lab.CloudResourceGroup(769).Nameresource group with a randomized name for the App Service using a Standard tier pricing plan.

16. In the **Create Web App** section of the **Deploy-SimplePaas.ps1** file, highlight the following six lines and click **Run selection**

```
New-AzureRmWebApp `
    -ResourceGroupName $resourceGroupName `
    -Location $resourceGroupLocation `
    -AppServicePlan $appServicePlan.ServerFarmWithRichSkuName `
    -Name $webAppName `
    -Verbose
```
**Note:** Now that you have deployed a web app and an associated app service plan, you can verify the settings in the Azure portal.
17. If you already logged into the azure portal per instructions on the Resource tab (above), please open that browser to the Azure portal and skip the steps for this instruction.
The next steps will log you into the Azure portal for this lab.
. Click here to open the Azure portal
. In the **Email or phone** field, enter User1-7322410 @cloudplatimmersionlabs.onmicrosoft.comand click **Next**
. In the **Password** field, enter k!DgeHA70$and click **Sign in**
. In the **Stay signed in?** window, click **No**
. If a **Welcome to Microsoft Azure** pop-up window appears, click **Maybe Later** to skip the tour
18. In the Azure **Favorites** menu on the left, click **App Services**
19. On the **App services** blade, click the contoso app service created for this lab
20. On the **Overview** blade of the **App Service**, click the **URL** link to view the external facing web application created for this lab
21. click Next on the bottom right of this window (the Guid Application)

PreviousNext

From <https://labondemand.com/CloudClient/2f1cb961-0ef3-4ec4-87c8-cb8b81f1fcf9>

AZ00007: Understanding Azure Resource Manager Templates

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario B - Deploy an IaaS Resource

In this scenario, you will create a simple template to deploy an IaaS resource into Azure

1. In **PowerShell ISE**, click **File > Open**
2. In the **HOL_Lab_Files/1_Understanding_Azure_Resource_Manager_Templates**folder, double-click the **Deploy-SimpleIaaS**file
**Note:** Now you will define some variables in the **Deploy-SimpleIaaS.ps1** file.
3. In the **Define Deployment Variables** section of the **Deploy-SimpleIaas.ps1** file, replace **[resource group name]** with @lab.CloudResourceGroup(769).Name
4. In the **Define Deployment Variables** section of the **Deploy-SimpleIaaS.ps1** file, replace **[resource group location]** with South Central US
5. In the **Define Deployment Variables** section of the **Deploy-SimpleIaas.ps1** file, highlight the following four lines and click **Run selection**

```
$resourceGroupName...
$resourceProviderNamespace...
$resourceTypeName...
```

```
$resourceGroupLocation...
```

6. Highlight the following six lines that define variables for a **Virtual Network** and click **Run selection**

```
$vNetName = 'vnet-contoso'
$vNetAddressPrefix = '172.16.0.0/16'
$vNetSubnet1Name = 'subnet-1'
$vNetSubnet1Prefix = '172.16.1.0/24'
$vNetSubnet2Name = 'subnet-2'
$vNetSubnet2Prefix = '172.16.2.0/24'
```

   **Note:** You have now defined network and subnet variables that will be used when you deploy the virtual network.

7. In the **Create Virtual Network Subnets**section of the **Deploy-SimpleIaas.ps1 file**, highlight the following eight lines and click **Run selection**

```
$vNetSubnet1 = New-AzureRmVirtualNetworkSubnetConfig `
-Name $vNetSubnet1Name `
-AddressPrefix $vNetSubnet1Prefix `
-Verbose

$vNetSubnet2 = New-AzureRmVirtualNetworkSubnetConfig `
-Name $vNetSubnet2Name `
-AddressPrefix $vNetSubnet2Prefix `
-Verbose
```

   **Note:** You have now defined variables for subnets that will be created at the same time the virtual network is being deployed.

8. In the **Create Virtual Network** section of the **Deploy-SimpleIaas.ps1 file**, highlight the following seven lines and click **Run selection**

```
$vNet = New-AzureRmVirtualNetwork `
    -ResourceGroupName $resourceGroup.ResourceGroupName `
    -Location $resourceGroup.Location `
    -Name $vNetName `
    -AddressPrefix $vNetAddressPrefix `
    -Subnet $vNetSubnet1,$vNetSubnet2 `
    -Verbose -Force
```

   **Note:** Now that you have created a virtual network with subnets, you can verify the deployed resources in the Azure portal.

9. Go to the **Azure portal**
10. In the **Favorites menu** on the left, click **Virtual Networks**
11. Click the **contosonet** virtual network to view its configuration
12. Go to the **PowerShell** window
13. In the **Define Deployment Variables** section of the **Deploy-SimpleIaas.ps1 file** file, highlight the following four lines and click **Run selection**

```
$randomString = ([char[]]([char]'a'..[char]'z') + 0..9 | Sort-Object
{Get-Random})[0..8] -join ''
$storageAccountNamePrefix = 'storage'
$storageAccountType = 'Standard_LRS'
$storageAccountName = $storageAccountNamePrefix +
($storageAccountType.Replace('Standard_','')).ToLower() + $randomString
```

   **Note:** You have now defined variables for the storage accounts that will be created.

14. In the **Create Storage Account** section of the **Deploy-SimpleIaas.ps1 file** file, highlight the following six lines and click **Run selection**

```
$storageAccount = New-AzureRmStorageAccount `
-ResourceGroupName $resourceGroup.ResourceGroupName `
```

```
-Location $resourceGroup.Location `
-Name $storageAccountName `
-Type $storageAccountType `
-Verbose
```
**Note:** Now that you have created a storage account, you can verify the deployed resources in the Azure portal.

15. After the **Run selection** command executes, go to the **Azure portal**
16. In the **Favorites menu** on the left, click **Storage Accounts**
17. Click the **storage account that begins with 'storage'** to view the account you just created and its configuration

PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/2f1cb961-0ef3-4ec4-87c8-cb8b81f1fcf9>

AZ00007: Understanding Azure Resource Manager Templates

2 Hours Remaining

Instructions

Resources

Help

100%

## Conclusion

In this lab, you explored the core structure of an ARM architecture including resources and their interdependencies. You also deployed two simple ARM templates - a PaaS and an IaaS solution - using PowerShell.

PreviousEnd

From <https://labondemand.com/CloudClient/2f1cb961-0ef3-4ec4-87c8-cb8b81f1fcf9>

# AZ00008: Building Azure Resource Manager Services

19 September 2018 10:43

AZ00008: Building Azure Resource Manager Services

1 Hr 56 Min Remaining

Instructions

Resources

Help

100%

## Building ARM Infrastructure

***Creating an IaaS solution***

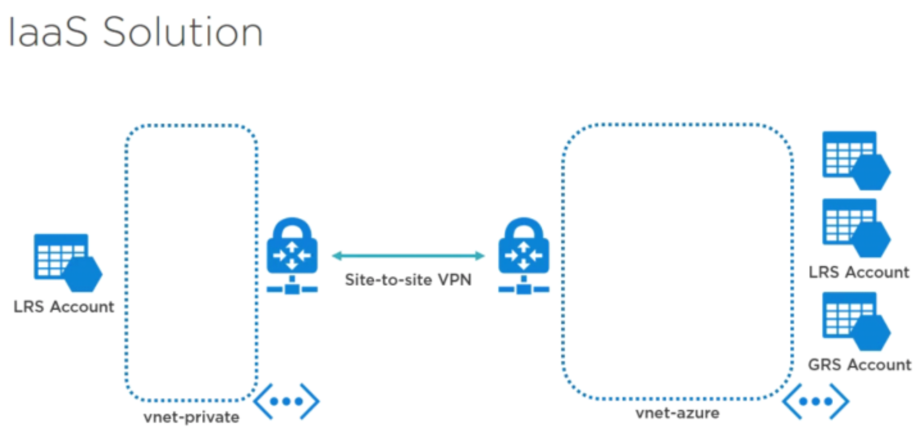
## Overview

This lab will guide the user through creating an IaaS solution using ARM templates.

### Lab Goals

- Review ARM template
- Deploy storage resources
- Deploy networking resources
- Deploy vNet peering
- Validate deployment of the above resources

### Product overview



**Capabilities or components used in this scenario:**The following resource types will be created as part of the lab:
• Storage accounts • Virtual networks • Public IP addresses • Virtual network peering • Connections

Next

From <https://labondemand.com/CloudClient/7973c76c-a3fa-4ed4-b9b4-25e7a1cb1627>

Instructions

Resources

Help

100%

# Building ARM Infrastructure

In this lab you will setup a private environment consisting of a network, locally redundant storage, and a network gateway. You will also deploy Azure services consisting of a virtual network, two locally redundant storage accounts, and a geo-redundant storage account. Finally, you will setup vNet peering so that you can communicate from one network to the other.

## Scenario A - Create storage account resources

In this scenario, you will start with a basic JSON template file and customize it to create an IaaS ARM template. You will begin with defining storage account resources in the template. To get started we will RDP into a jumphost where we have pre-loaded PowerShell along with ARM templates that we'll be working with.

1. On the **Resources** tab in the guide application, click **jumphost RDP profile** to download the jumphost RDP profile
2. Open the downloaded RDP profile
3. Copy and paste @lab.CloudResourceGroup(789).Name.southcentralus.cloudapp.azure.cominto the **Computer** field and click **Connect**, on first connect you will need to click the checbox [ x ] Don't ask me again for connections to this computer
4. For the password, copy and paste jG6bPp*M5+ and click **Connect**
5. On the next dialog click [ x ] Don't ask me again for connection to this computer then click [Yes], this will bypass certificate issues
6. On the jumphost desktop, click the **Start**button and click **Visual Studio 2017** - it may take a few minutes for Visual Studio to open the first time
   **Note:** The Blend for Visual Studio icon (at the top) is very similar to Visual Studio 2017
7. You will be prompted to sign in to Visual Studio, click **Sign in**
8. In the **Email or phone** field, type User1-7322435 @cloudplatimmersionlabs.onmicrosoft.com
9. Click **Next**
10. In the **Password** field, type jG6bPp*M5+
11. Click **Sign in**
12. On the **Choose your color theme** screen, select a color theme
13. Click **Start Visual Studio**
14. In the main navigation window, click **View > Other Windows > JSON Outline** to activate the view
    **Note:** Depending on your screen resolution, you can close the **Solution Explorer** and **Team Explorer** panels on the right side of your screen, if desired.
15. In **Visual Studio**, click **File > Open > File**
16. In the **Quick access** menu, click **Documents**
17. Double-click **HOL_Lab_Files** folder, double-click the **2 _Building_Azure_Resource_Manager_Infrastructure**folder, and then double-click the **ContosoIaas.json** file which may take a few moments to fully load
18. In the **JSON Outline**, right-click **Resources**and choose **Add New Resource**
19. In the **Search** box at the top, type **storage**
20. In the **search results**, select **Storage Account** as the resource type

21. In the **Name** field, type **Temporary**
22. Click **Add**

    **Note:** Now you will clean up some of the variables and parameters that were automatically created and refactor properties of the new storage account.
23. In the **JSON Outline**, double-click **variables**to expand the list
24. In the **variables** section, right-click the **TemporaryName** variable and choose **Delete**
25. In the **JSON Outline**, click **parameters**
26. Under **"parameters": {** on the first line, paste the following lines:
```
"storageAccountType": {
  "type": "array",
  "defaultValue": [ "Standard_LRS", "Standard_LRS", "Standard_GRS" ]
},
```
    **Note:** This will create two standard locally redundant storage accounts and one standard geographically redundant storage account.
27. Under the **"storageAccountType";**parameter, paste the following lines:
```
"storageAccountNamePrefix": {
  "type": "string",
  "defaultValue": "contoso",
  "minLength": 1
},
```
    **Note:** This sets the storage account name to start with the word *contoso* as a prefix.
28. In the **resources** section of the JSON Outline, click the **storageAccount** resource you named **Temporary**
29. In the **storageAccount** resource, replace **[variables('TemporaryName')]** with **[concat(parameters('storageAccountNamePrefix'),copyIndex(),uniqueString(resourceGroup().id))]**in the **"name":** property
30. In the **"displayName":** property, replace **Temporary** with **[concat(parameters('storageAccountNamePrefix'),copyIndex(),uniqueString(resourceGroup().id))]**

    **Note:** This will create a naming convention for each storage account that will include the prefix identified in the parameters section with a unique number for each storage account
31. Between the **apiVersion** property and the **sku** property in the **storageAccount**resource, paste the following lines to add a copy task function:
```
  "copy": {
    "count": "[length(parameters('storageAccountType'))]",
    "name": "storageCopy"
  },
```
    **Note:** This will allow ARM to look at the number of entries in the storageAccountType parameter and use the resulting integer as the number for the count
32. In the **"sku":** property, replace **[parameters('temporaryType')]** with **[parameters('storageAccountType')[copyIndex()]]**
33. Click **File > Save ContosoIaas.json** to save your changes

    **Note:** You will now use PowerShell to deploy the ARM template.
34. Click the **Start** button, type **Powershell ISE**and select the **Windows PowerShell ISE Desktop App**
35. Click **File > Open**
36. Double-click the **HOL_Lab_Files** folder, double-click the **2 _Building_Azure_Resource_Manager_Infrastructure**folder, and then double-click **Deploy-ContosoIaas_2** to open it
37. In the **Define Deployment Variables**section, replace **[resource group location]**with South Central US
38. In the **Define Deployment Variables**section, replace **[resource group name]** with @lab.CloudResourceGroup(789).Name
39. In the blue **PowerShell** window, paste the following line and press **Enter** to launch the Azure login:

```
Add-AzureRmAccount
```

40. In the login screen's **Email or phone** field, enter User1-7322435 @cloudplatimmersionlabs.onmicrosoft.com
41. Click **Next**
42. In the **Password** field, enter jG6bPp*M5+
43. Click **Sign in**
44. Highlight all the lines in the **Define Deployment Variables** section and click **Run selection** icon
    from the toolbar above
45. In the **Deploy Resources** section, highlight the **New-AzureRmResourceGroupDeployment**cmdlet and click **Run selection**
    **Note:** After the PowerShell deployment is completed, you can verify the deployment in the Azure portal.
46. If you already logged into the azure portal per instructions on the Resource tab (above), please open that browser to the Azure portal and skip the steps for this instruction.
    The next steps will log you into the Azure portal for this lab.
    . Click here to open the Azure portal
    . In the **Email or phone** field, enter User1-7322435 @cloudplatimmersionlabs.onmicrosoft.comand click **Next**
    . In the **Password** field, enter jG6bPp*M5+and click **Sign in**
    . In the **Stay signed in?** window, click **No**
    . If a **Welcome to Microsoft Azure** pop-up window appears, click **Maybe Later** to skip the tour
47. In the **Favorites** menu on the left, click **Resource groups**
48. Click on @lab.CloudResourceGroup(789).Name to view the resources deployed to that resource group
49. Click on the **Contoso storage accounts** that were just created and verify the type, storage location, and replication setting.

PreviousNext

From <https://labondemand.com/CloudClient/7973c76c-a3fa-4ed4-b9b4-25e7a1cb1627>

AZ00008: Building Azure Resource Manager Services

1 Hr 56 Min Remaining

Instructions

Resources

Help

100%

## Scenario B - Network resources

Next you will add network resources to the ARM template.

1. Return to the **ContosoIaas.json** file in **Visual Studio**
2. In the **JSON Outline**, right-click **Resources**and choose **Add New Resource**
3. In the **Search** box at the top, type **virtual network**
4. In the **search results**, select **Virtual Network**as the resource type
5. In the **Name** field, type **private**
6. Click **Add**
7. In the new resource's **"name":** property, change **private** to **[parameters('vNetPrivateName')]**
   **Note:** You have just changed the name from a static name to a name derived from a

parameter, but the parameter has not yet been defined in the template. You will see a red indicator on the right hand side of the template in Visual Studio to identify that there is an error in the template. These error indicators are a helpful way to make sure you complete all necessary steps when creating or modifying ARM templates.

8. In the **"displayName":** property, change **private** to **[parameters('vNetPrivateName')]**
9. In the **"properties":** property, change the **"addressPrefixes":** variable from **privatePrefix** to **vnetPrivatePrefix**
10. In the **"subnets":** property, change the **"name":** variables from **privateSubnet1Name** to **vNetPrivateSubnet1Name** and **privateSubnet2Name** to **vNetPrivateSubnet2Name**
11. In the **"subnets":** property, change the **"properties": { "addressPrefix":** variables from **privateSubnet1Prefix** to **vNetPrivateSubnet1Prefix** and **privateSubnet2Prefix** to **vNetPrivateSubnet2Prefix**

    **Note:** You will now add a second virtual network and re-factor its associated variables.
12. In the **JSON Outline**, right-click **Resources**and choose **Add New Resource**
13. In the **Search** box at the top, type **virtual network**
14. In the **search results**, select **Virtual Network**as the resource type
15. In the **name** field, type **azure**
16. Click **Add**
17. In the new resource's **"name":** property, change **azure** to **[parameters('vNetAzureName')]**
18. In the **"displayName":** property, change **azure** to **[parameters('vNetAzureName')]**
19. In the **"properties":** property, change the **"addressPrefixes":** variable from **azurePrefix** to **vNetAzurePrefix**
20. In the **"subnets":** property, change the **"name":** variables from **azureSubnet1Name**to **vNetAzureSubnet1Name** and **azureSubnet2Name** to **vNetAzureSubnet2Name**
21. In the **"subnets":** property, change the **"properties": { "addressPrefix":** variables from **azureSubnet1Prefix** to **vNetAzureSubnet1Prefix** and **azureSubnet2Prefix** to **vNetAzureSubnet2Prefix**

    **Note:** There are now a number of undefined parameters and variables in the template that you will need to clean up to prepare the template for deployment. First, you will modify the variables.
22. In the **JSON Outline**, click **variables**
23. For each of the five variables that contain the **private** label (lines 50-54), change **private** to **vNetPrivate**
24. For each of the five variables that contain the **azure** label (lines 55-59), change **azure** to **vNetAzure**
25. In the **"vNetPrivatePrefix":** variable, change the **IP address** to **172.16.0.0/16**
26. In the **"vNetPrivateSubnet1Name":**variable, change **"Subnet-1"** to **private-subnet-1**
27. In the **"vNetPrivateSubnet1Prefix":**variable, change the **IP address** to **172.16.1.0/24**
28. In the **"vNetPrivateSubnet2Name":**variable, change **Subnet-2** to **private-subnet-2**
29. In the **"vNetPrivateSubnet2Prefix":**variable, change the **IP address** to **172.16.2.0/24**
30. In the **"vNetAzureSubnet1Name":** variable, change **Subnet-1** to **azure-subnet-1**
31. In the **"vNetAzureSubnet1Prefix":** variable, change the **IP address** to **10.0.1.0/24**
32. In the **"vNetAzureSubnet2Name":** variable, change **Subnet-2** to **azure-subnet-2**
33. In the **"vNetAzureSubnet2Prefix":** variable, change the **IP address** to **10.0.2.0/24**

    **Note:** All of the variables have now been renamed to be more descriptive, and the IP addresses have been changed so there will be no address conflicts when the networks are deployed. Now you can add the missing parameters to the file.
34. In the **JSON Outline**, click **parameters**
35. After the closing **}** of the **"storageAccountNamePrefix":** parameter, paste the following lines:

```
  ,
    "vNetPrivateName": {
```

```
    "type": "string",
    "defaultValue": "contoso-vnet-private",
    "minlength": 1
    },
    "vNetAzureName": {
      "type": "string",
      "defaultValue": "contoso-vnet-azure",
      "minlength": 1
    },
```
**Note:** You have now added parameters for both the private and azure virtual networks that you defined while adding the new resources.

36. Click **File > Save ContosoIaas.json** to save your changes
37. In **PowerShell ISE**, return to the **Deploy-ContosoIaas_2.ps1** file
38. In the **Deploy Resources** section, highlight the **New-AzureRmResourceGroupDeployment**cmdlet and click **Run selection**

**Note:** This will deploy the two virtual networks and related subnets to the resource group.

PreviousNext

From <https://labondemand.com/CloudClient/7973c76c-a3fa-4ed4-b9b4-25e7a1cb1627>

AZ00008: Building Azure Resource Manager Services

1 Hr 56 Min Remaining

Instructions

Resources

Help

100%

## Scenario C - vNet Peering

You will now configure a vNet peering resource to establish a relationship between the two virtual networks you just created to allow the private and Azure networks to talk to each other.

1. In **Visual Studio**, return to the **ContosoIaas.json** file
2. In the **resources** section of the **JSON Outline**, click **[vNetPrivateName]**
3. In the **[vNetPrivateName]** code block, add a line above the closing **}**
4. On the new line, **paste** the following lines:
```
"resources":[
{
    "name": "[variables('vNet1tovNet2PeeringName')]",
    "type": "virtualNetworkPeerings",
    "apiVersion": "2017-03-01",
    "location": "[resourceGroup().location]",
```
**Note:** This establishes a vNet peering resource for the private vNet in the same location the resource group is deployed.

5. After the **"location":** property, **paste** the following lines:
```
"dependsOn": [
  "[concat('Microsoft.Network/virtualNetworks/', 
parameters('vNetPrivateName'))]",
  "[concat('Microsoft.Network/virtualNetworks/', 
parameters('vNetAzureName'))]"
  ],
  "comments": "This is the peering from the private vNet to the azure 
vNet",
```
**Note:** This adds a dependency on the two virtual networks that are deployed in the template.

6. After the **"comments":** property, **paste** the following lines:

```
"properties": {
  "allowVirtualNetworkAccess": "true",
  "allowForwardedTraffic": "false",
  "allowGatewayTransit": "false",
  "useRemoteGateways": "false",
  "remoteVirtualNetwork": {
    "id":
"[resourceId('Microsoft.Network/virtualNetworks',parameters('vNetAzureNam
e'))]"
  }
 }
}
],
```
**Note:** This sets the properties for the vNetPeering resource.

7. In the **resources** section of the **JSON Outline**, click **[vNetAzureName]**
8. In the **[vNetAzureName]** code block, add a line above the closing **}**
9. On the new line, **paste** the following lines:

```
"resources":[
{
    "name": "[variables('vNet2tovNet1PeeringName')]",
    "type": "virtualNetworkPeerings",
    "apiVersion": "2017-03-01",
    "location": "[resourceGroup().location]",
"dependsOn": [
  "[concat('Microsoft.Network/virtualNetworks/',
parameters('vNetPrivateName'))]",
  "[concat('Microsoft.Network/virtualNetworks/',
parameters('vNetAzureName'))]"
  ],
  "comments": "This is the peering from the azure vNet to the private
vNet",
"properties": {
  "allowVirtualNetworkAccess": "true",
  "allowForwardedTraffic": "false",
  "allowGatewayTransit": "false",
  "useRemoteGateways": "false",
  "remoteVirtualNetwork": {
    "id":
"[resourceId('Microsoft.Network/virtualNetworks',parameters('vNetPrivateN
ame'))]"
  }
 }
}
],
```

10. In the **JSON Outline**, click **variables**
11. Below the **"vNetAzureSubnet2Prefix":**variable, paste the following lines:

```
  "vNet1tovNet2PeeringName": "[concat(parameters('vNetPrivateName'), '-',
parameters('vNetAzureName'))]",
  "vNet2tovNet1PeeringName": "[concat(parameters('vNetAzureName'), '-',
parameters('vNetPrivateName'))]"
```
**Note:** These variables define the names for the vNet peering and complete the steps necessary for adding the peering resource to the template.

12. Click **File > Save ContosoIaas.json** to save your changes
13. In **PowerShell ISE**, return to the **Deploy-ContosoIaas_2.ps1** file
14. In the **Deploy Resources** section, highlight the **New-AzureRmResourceGroupDeployment**cmdlet and click **Run selection**

    **Note:** Now you can verify the deployment in the Azure portal.
15. Return to the **Azure portal**
16. Refresh the Resource Group @lab.CloudResourceGroup(789).Nameblade

17. You can now review the contoso privatevNet, azurevNet, and vNetPeering resources that you just added to the resource group.

PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/7973c76c-a3fa-4ed4-b9b4-25e7a1cb1627>

AZ00008: Building Azure Resource Manager Services

1 Hr 56 Min Remaining

Instructions

Resources

Help

100%

## Conclusion

In this lab, you reviewed and modified simple IaaS and PaaS templates to deploy resources such as storage accounts, virtual networks, and an App service plan. After deploying the resources, you verified the deployments in the Azure portal.

PreviousEnd

From <https://labondemand.com/CloudClient/7973c76c-a3fa-4ed4-b9b4-25e7a1cb1627>

# AZ00009: Building Azure Resource Manager Infrastructure

19 September 2018 10:49

AZ00009: Building Azure Resource Manager Infrastructure

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Deploying Virtual Machines

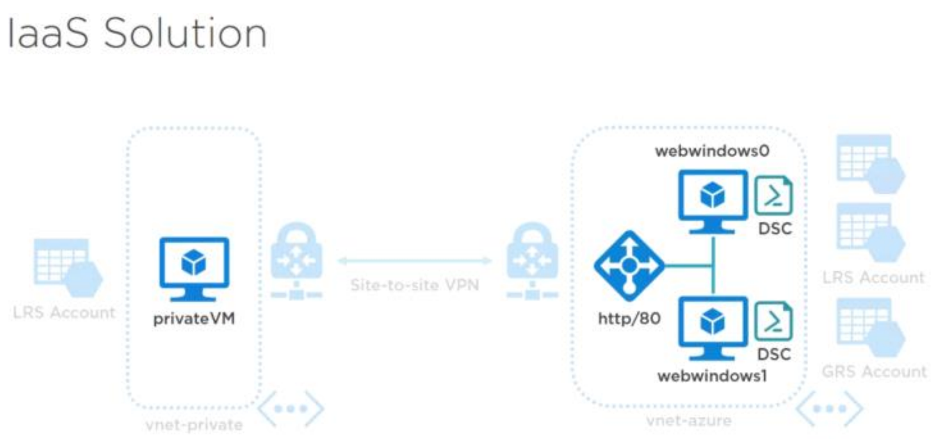
## Overview

In this lab you will extend an ARM template to deploy virtual machines and a load balancer into the infrastructure.

### Lab Goals

- Deploy "private" virtual machine
- Deploy remote virtual machines
- Configure remote virtual machines with DSC
- Deploy load balancer
- Validate the deployment and configuration of all of these resources

### Product overview



**Capabilities or components used in this scenario:** • Network Interfaces • Virtual Machines • Virtual Machine Extensions • Availability Sets • Load Balancers

Next

From <https://labondemand.com/CloudClient/e8113556-1ed4-4179-8baa-edb932b8866d>

AZ00009: Building Azure Resource Manager Infrastructure

1 Hr 58 Min Remaining

Instructions

Resources

Help
100%
# Deploying Virtual Machines

## Lab Overview

In this lab, you will deploy and configure virtual machines and deploy a load balancer to balance web services.

### Scenario A - Build and Deploy Private Virtual Machine

To get started we will RDP into a jumphost where we have pre-loaded PowerShell along with ARM templates that we'll be working with.

1. On the **Resources** tab in the guide application, click **jumphost RDP profile** to download the jumphost RDP profile
2. Open the downloaded jumphost RDP profile
3. Copy and paste rg-lod7332272.southcentralus.cloudapp.azure.com into the **Computer** field and click **Connect**, on first connect you will need to click the checbox [ x ] Don't ask me again for connections to this computer
4. For the password, copy and paste rNqS8e!0D+ and click **Connect**
5. On the next dialog click [ x ] Don't ask me again for connection to this computer then click [Yes], this will bypass certificate issues
6. On the jumphost desktop, click the **Start** button and click **Visual Studio 2017** - it may take couple of minutes for Visual Studio to open the first time
7. If prompted to sign in to **Visual Studio**, click **Sign in**
8. In the **Email or phone** field, type User1-7332272 @cloudplatimmersionlabs.onmicrosoft.com
9. Click **Next**
10. In the Password field, type rNqS8e!0D+
11. Click **Sign in**
12. If prompted with the **Choose your color theme** screen, select a color theme
13. Click **Start Visual Studio**
14. In the main navigation, click on **View > Other Windows > JSON Outline** to activate the view.
15. In **Visual Studio**, click **File > Open > File**
16. In the **Quick access** menu, click on **Documents**
17. In the **HOL_Lab_Files/3_Deploying_Virtual_Machines**folder, double-click the **contosoIaaS_3.json** file
18. In the **JSON Outline**, right-click on resources and select **Add New Resource**
19. In the **Search** box, type **windows**
20. In the **search results**, select **Windows Virtual Machine**
21. In the **Name** field, enter **vmPrivate**
22. In the **Virtual Network/subnet:** field, select **variables('vNetPrivateSubnet1Name')**
23. Click **Add**
24. In the **JSON Outline**, right-click on resources and select **Add New Resource**
25. In the **Search** box, type **public**
26. In the **search results**, select **Public IP Address**
27. In the **Name:** field, enter **vmPrivatePublicIP**
28. In the **Network interface:** box, select **vmPrivateNic**
29. Click **Add**
   **Note:** Now that you have added all of the necessary resources to the template, you can consolidate and refactor the template.

30. In the **JSON Outline**, double click on the **parameters** header
31. In the **parameters** section, click on **vmPrivatePublicIPDnsName**
32. After the **"type": "string",** line, paste **"defaultValue": "contoso-vm",** (including the **,**) to identify a default name
33. In the **JSON Outline**, double click on the **variables** header
34. In the **variables** section, click on **vmPrivateNicName**
35. Replace the **"vmPrivateNicName":** variable definition with **"[concat(parameters('vmPrivateName'), '-', 'nic-0')]",**
36. In the **resources** section, click on **vmPrivateNic**
37. In **"tags":** property, locate the **displayName:"vmPrivateNic"** variable and replace the variable definition with **"[variables('vmPrivateNicName')]"**
38. In the **resources** section, click on **vmPrivate**
39. In the **"dependsOn":** property, change **"[resourceId('Microsoft.Storage/storageAccounts', concat(parameters('storageAccountNamePrefix'),copyIndex(),uniqueString(resourceGroup().id)))]",**to **"storageCopy",**
40. In the **"osDisk:"** property of the **vmPrivate** resource, locate the **"uri"**: string
41. In the **"uri"**: string, replace **copyIndex()** with **'0'**
42. In the **parameters** section, click on **vmPrivateName**
43. Under the **"type": "string",** line, add the line **"defaultValue": "vmPrivate",**
44. In the **parameters** section, click on the **vmPrivateWindowsOSVersion** parameter and change the default value from **"2012-R2-Datacenter"** to **"2016-Datacenter"**
45. In the **"allowedValues":** property, replace the **"Windows-Server-Technical-Preview"** value with **"2016-Datacenter"**
46. In the **parameters** section, click on**vmPrivateAdminUserName**
47. Under the **"type": "string",** line, add the line **"defaultValue": "vmAdmin",**
48. In the **resources** section of the **JSON Outline**, click on **vmPrivatePublicIP**
49. In the **properties**, change the **"domainNameLabel:"** value from **"[parameters('vmPrivatePublicIPDnsName')]"** to **"[concat(parameters('vmPrivatePublicIPDnsName'),'-',uniqueString(resourceGroup().id))]"**
50. **Save** the **contosoIaaS_3.json** file

    **Note:** You need to pass the vmPrivateAdminPassword value into the template, but it is a secure value. To accomplish this, you will use PowerShell to build a PowerShell variable.
51. Click the **Start** button, type **Powershell ISE** and select the **Windows PowerShell ISE Desktop App**
52. Click on **File > Open**
53. In the **HOL_Lab_Files/3_Deploying_Virtual_Machines**folder, double-click **Deploy-ContosoIaas_3.ps1** to open the file
54. In the **PowerShell** window, paste the following line and press **Enter** to launch the Azure login

    ```
    Add-AzureRmAccount
    ```

    **Note:** A login screen should appear. If it does not display for you, use the Alt + Tab key to find and open the login screen.
55. In the login screen's **Email or phone** field, enter User1-7332272 @cloudplatimmersionlabs.onmicrosoft.com
56. Click **Next**
57. In the **Password** field, enter rNqS8e!0D+
58. Click **Sign in**
59. In the **Define Deployment Variables** section, replace **[resource group location]** with South Central US
60. In the **Define Deployment Variables** section, replace **[resource group name]** with rg-lod7332272
61. Highlight all of the lines in the **Define Deployment Variables** section and click **Run selection**

62. At the beginning of the **Deploy Resources** section, paste the following lines to add an additional parameters hashtable
```
$additionalParameters = New-Object -TypeName Hashtable
$additionalParameters['vmPrivateAdminPassword'] = $securePassword
```
63. Highlight both of the **additional parameter lines** and click **Run selection**
64. In the **New-AzureRmResourceGroupDeployment**command under the **-TemplateFile $template** ` line, paste the following line
```
@additionalParameters `
```
65. Highlight the **new-AzureRMResourceGroupDeployment command** and click **Run selection**

    **Note:** Deploying a virtual machine can take 5-10 minutes. After you sign into the Azure portal, you may have to wait for the deployment to finish and refresh the resource group blade.
66. If you already logged into the azure portal per instructions on the Resource tab (above), please open that browser to the Azure portal and skip the steps for this instruction
    The next steps will log you into the Azure portal for this lab:
    . Click here to open the Azure portal
    . In the **Email or phone** field, enter User1-7332272 @cloudplatimmersionlabs.onmicrosoft.com and click **Next**
    . In the **Password** field, enter rNqS8e!0D+ and click **Sign in**
    . In the **Stay signed in?** window, click **No**
    . If a **Welcome to Microsoft Azure** pop-up window appears, click **Maybe Later** to skip the tour
67. In the **Favorites** menu on the left, click **Resource groups**
68. In the **Resource groups** blade, click on **rg-lod7332272**
69. In the rg-lod7332272 blade, you can identify the newly deployed resources from the **ContosoIaaS_3.json** template, including the **vmPrivate** VM, storage accounts, virtual networks, and network interface with the associated public IP address
70. Click on the **vmPrivate** VM to open the virtual machine blade
    **Note:** In this blade, you can review the details of the virtual machine's configuration and obtain an RDP file to allow you to connect to the virtual machine.
71. **Close** the virtual machine blade

PreviousNext

From <https://labondemand.com/CloudClient/e8113556-1ed4-4179-8baa-edb932b8866d>

AZ00009: Building Azure Resource Manager Infrastructure

1 Hr 58 Min Remaining

Instructions

Resources

Help

100%

## Scenario B - Deploy virtual machines

Now that you have created the private virtual machine in a private virtual network, you can deploy remote virtual machines into the remote virtual network.

1. Return to the **contosoIaaS_3.json** file in Visual Studio
2. In the **JSON Outline**, right-click on **resources** and select **Add New Resource**
3. In the **Search** box, type **windows**
4. In the **search results**, select **Windows Virtual Machine**
5. In the **Name** field, enter the name **vmweb**
6. In the **Virtual network/subnet** box, select the **vNetAzureSubnet1Name** for the subnet

7. Click **Add**
8. In the **JSON Outline**, right-click on **resources** and select **Add New Resource**
9. In the **Search** box, type **availability**
10. In the **search results**, select **Availability Set**
11. In the **Name:** box, type **as-vmweb**
12. Click **Add**
13. In the **parameters** section of the **JSON Outline** delete the following parameters: **vmwebName, vmwebAdminUserName, vmwebAdminPassword, and vmwebWindowsOSVersion**
14. In the **parameters** section, click on **vmPrivateAdminUserName** and rename the parameter to **vmAdminUserName**
15. In the **parameters** section, click on **vmPrivateAdminPassword** and rename the parameter to **vmAdminPassword**
16. Click on the **vmPrivateWindowsOSVersion** parameter and rename it to **vmWindowsOSVersion**
17. In the **resources** section of the **JSON Outline**, click on the **vmweb** resource
18. In the **"name":** property, change **"[parameters('vmwebName')]"** to **"[concat('vmweb','-',copyIndex())]"**
19. In the **"tags":** property locate the **"displayName"** section, change **"vmweb"** to **"[concat('vmweb','-',copyIndex())]"**
20. Between the **"apiVersion":** property and the **"dependsOn":**property, paste the following to insert a copy task
```
"copy": {
  "count": 2,
  "name": "vmwebCopy"
},
```
21. Highlight the **"dependsOn":** property and replace it with the following lines
```
    "dependsOn": [
        "storageCopy",
        "vmwebNicCopy",
        "[concat('Microsoft.Compute/availabilitySets/','as-vmweb')]"
    ],
```
**Note:** In this step, you have changed the "dependsOn": property to wait for the storageCopy function, vmwebNicCopy function, and the availability set resource before beginning the vmweb resource deployment.
22. In the **"properties":** property, change the parameter values in the **"osProfile":** section for the **"adminUsername":** and **"adminPassword":** to **'vmAdminUserName'** and **'vmAdmin Password'**
23. In the **"properties":** property in the **"storageProfile":**section, change the **"sku":** parameter value to **vmWindowsOSVersion**
**Note:** Now you will delete some duplicated variables and rename the existing variables to more generic names.
24. In the **JSON Outline** in the **variables** section, delete the **vmwebImagePublisher** and **vmwebImageOffer** variables
25. Click on the **vmPrivateImagePublisher** variable and rename the variable to **vmImagePublisher**
**Note:** The renamed variable should now be displayed as the following: **"vmImagePublisher": "MicrosoftWindowsServer"**
26. Click on the **vmPrivateImageOffer** variable and rename the variable name to **vmImageOffer**
**Note:** The renamed variable should now be displayed as the following: **"vmImageOffer": "WindowsServer"**
27. In the **resources** section of the **JSON Outline**, click on the **[concat('vmweb','-',copyIndex())]** resource
28. In the **"properties":** property, locate the **"storageProfile":**settings and change the **"publisher":** variable to **vmImagePublisher** and the **"offer":** variable

to **vmImageOffer**

29. In the **variables** section of the **JSON Outline**, delete the **vmwebVmSize** variable
30. In the **variables** section, click on the **"vmPrivateVmSize":**variable and change the name to **"vmSize":**
31. In the **resources** section of the **JSON Outline**, click on **vmwebNic**
32. In the **"name":** property,
    change **"[variables('vmwebNicName')]"** to **"[concat('vmweb','-',copyIndex(),'-nic-0')]"**
33. In the **"tags":** property,
    change **"displayName":** value **"vmwebNic"** to **"[concat('vmweb','-',copyIndex(),'-nic-0')]"**
34. In between the **"apiVersion":** property and the **"dependsOn":** property, paste the following to insert a copy task
    ```
    "copy": {
      "count": 2,
      "name": "vmwebNicCopy"
    },
    ```
35. In the **resources** section, click on **[concat('vmweb','-',copyIndex())]**
36. In the **"properties":** of the **"[concat('vmweb','-',copyIndex())]"** resource, scroll down to find the **"networkProfile"** property
37. In the **"id":** line,
    replace **variables('vmwebNicName')** with **concat('vmweb','-',copyIndex(),'-nic-0')**
38. In the **"properties":** section of the **"[concat('vmweb','-',copyIndex())]"** resource, paste the following lines after the **"hardwareProfile":** property's closing **}** to create an availabilitySet property
    ```
        "availabilitySet": {
          "id": "[resourceId('Microsoft.Compute/availabilitySets/','as-
    vmweb')]"
        },
    ```
    **Note:** This tells the template to which availabilitySet the vmweb resource needs to be bound. Now you can begin to clean up the template, removing any errors and updating any variables or parameter references to clear the errors in the template.
39. In the **variables** section of the **JSON Outline**, delete **vmwebNicName**
40. In the **variables** section, delete**vmwebStorageAccountContainerName**
41. In the **variables** section,
    change **"vmPrivateStorageAccountContainerName":** to **"vmStorageAccountContainerName":**
42. In the **resources** section, click on the **vmPrivate** resource and replace **'vmPrivateStorageAccountContainerName'** in the **"OSDisk":** - **"uri":** setting to **'vmStorageAccountContainerName'**
43. In the **resources** section, click on the **[concat('vmweb','-',copyIndex())]** resource and replace **'vmwebStorageAccountContainerName'** in the **"OSDisk"**:- **"uri":** setting with **'vmStorageAccountContainerName'**
44. In the resources section, click on **as-vmweb** resource
45. In the **"displayName":** property, change the value **"as-vmweb"** to to **"[parameters('as-vmwebName')]"**
46. In the **"properties":** property, change
    the **platformUpdateDomainCount** and **platformFaultDomainCount** values to **2**
47. In the **parameters** section, click on **as-vmwebName**
48. Under the **"type":** property, add a new line with **"defaultValue": "as-vmweb",**
49. In the **resource** section, click on the **[concat('vmweb","-',copyIndex())]** virtual machine resource
50. In the **"properties":** - **"hardwareProfile":** section, change **'vmwebVmSize'** to **'vmSize'**
51. In the **"osProfile":** section, change the **"computerName"**: parameter
    from **"[parameters('vmwebName')]"** to **"[concat('vmweb','-',copyIndex())]"**
52. In the **"osProfile":** section, change the values
    for **"adminUsername":** and **"adminPassword":** to **'vmAdminUsername'** and **'vmAdminP**

**assword'**respectively

53. In the **"storageProfile": - "osDisk":** section, change the **"name":** value from **"vmwebOSDisk"** to **"[concat('vmweb','-',copyIndex(),'-','osDisk')]"**
54. Also in the **osDisk** settings at the end of the **"uri":** property string, replace **variables('vmwebOSDiskName'),** with **'vmweb','-',copyIndex(),'-','osDisk',**
55. In the **resources** section, click on the **vmPrivate** resource
56. In **"properties":**, find the **hardwareProfile** section and change **'vmPrivateVmSize'** to **'vmSize'**
57. In **"properties":**, find the **osProfile** section and remove the word **Private** from **vmPrivateAdminUsername** and**vmPrivateAdminPassword**
58. In the **storageProfile** section, remove the word **Private** from**vmPrivateImagePublisher**, **vmPrivateImageOffer**, and **vmPrivateWindowsOSVersion**
    **Note:** Before returning to PowerShell to deploy the template, check the right side of your ARM template to make sure all conflicts are resolved.
59. Click **File > Save** to save the **ContosoIaaS_3.json** file in Visual Studio
60. Return to **PowerShell ISE**
61. In the **@additionalParameters hashtable**, change **'vmPrivateAdminPassword'** to **'vmAdminPassword'**
62. Highlight both lines of the **@additionalParameters hashtable** and click **Run selection**
63. Highlight the **New-AzureRmResourceGroupDeployment**command and click **Run selection**
    **Note:** This deployment may take 5-10 minutes to complete. You can wait for the deployment to complete before finishing the steps below to verify the resources deployed, or you can click Next at the bottom of the screen to proceed with the next step in the ARM template development while the deployment completes.
64. After the deployment is done, return to the Azure portal and refresh the rg-lod7332272 resource group
65. In the **Overview** blade, you can review the list of resources that are associated with the deployment you have just completed
66. Click on **vmweb-0** to open the private virtual machine
67. In the **settings** menu, click on **Disks** and review the configuration details
    **Note:** Here you can review the disk settings for the virtual machine.
68. In the **settings** menu, locate and click on **Networking**
69. Click on **Network interface** -> **vmweb-0-nic-0** object
70. In the **Network interface** blade, review the configuration details
71. In the **breadcrumb** at the top of the blade, click on the rg-lod7332272 link to return to the resource group blade
72. In the **resources** list, click on **as-vmweb** Availability set resource
73. In the **as-vmweb** blade, you can verify the two virtual machines created are running and identify the fault and update domains that have been assigned
74. **Close** the **as-vmweb** blade
75. In the **resources** list, click on **contoso-vnet-azure** Virtual network resource
76. In **Settings**, locate and click on **Peerings**, from here locate and click on **contoso-vnet-azure-contoso-vnet-private**
77. Review the following statuses, **Peering status** should reflect a status of **Connected** and **Provisioning state** should reflect a status of **Succeeded**
78. Use the bottom scroll bar to scroll left to the **contoso-vnet-azure-contoso-vnet-private** blade, and then **Close** the blade

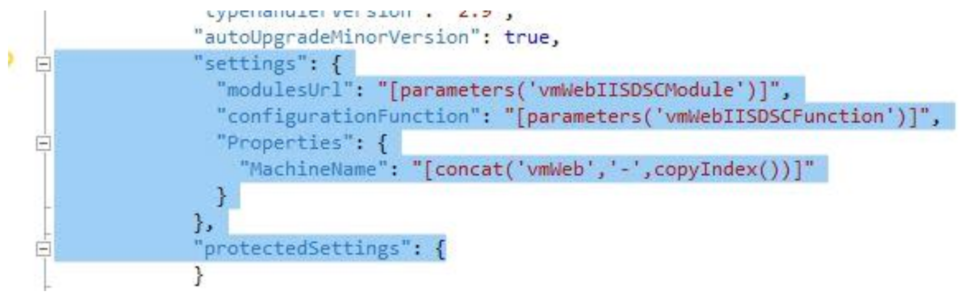
PreviousNext

From <https://labondemand.com/CloudClient/e8113556-1ed4-4179-8baa-edb932b8866d>

## Scenario C - Configure Windows Servers with PowerShell DSC VM Extensions

Now you will configure the basic VM servers that were just deployed into your remote Azure network. ARM can be used to configure the VM servers as application web servers. You will do this by adding a Virtual Machine Extension using PowerShell Desired State Configuration (DSC).

1. Return to the **Azure portal** and navigate to the rg-lod7332272 resource group blade
2. Click **vmPrivate** to open the **Overview** blade for the private VM
3. Click **Extensions** in the **SETTINGS** menu
4. The **MicrosoftMonitoringAgent** extension *may* show installed; however this should be the only extension that displays
5. Close the **Extensions** blade
6. Return to **Visual Studio** and the **contosoIaaS_3.json** file
7. In the **JSON Outline**, click **resources**
8. In the **resources** section, right-click on the **[concat('vmweb','-',copyIndex())]** virtual machine resource and choose **Add New Resource**
9. In the **Search** box, type **PowerShell**
10. In the **search results**, click on **PowerShell DSC Extension**
11. In the **Name:** section, type **vmweb**
12. In the **Virtual machine:** dropdown menu, select **[concat('vmweb','-',copyIndex())]** from the list
13. Click **Add**
14. In the **JSON Outline**, you can see that there is now a child resource to the **[concat('vmweb','-',copyIndex())]** virtual machine resource
    **Note:** This means, when the virtual machine is deployed, the extension will be deployed automatically. It is also possible to define a VM extension as a completely independent resource with a dependency for a parent virtual machine. Now you will adjust the resource settings for your ARM deployment and identify the specific extension needed on each VM.
15. In the **"name":** property of the **vmweb** child resource, change the name value from **"microsoft.Powershell.DSC"**to **"IIS"**
16. In the **"tags":** property of the **vmweb** child resource, change the **"displayName":** value from **"vmweb"** to **"IIS"**
17. In **"properties":** replace the **"settings"**: code block on lines 413-422 with the following:
```
          "settings": {
          "modulesUrl": "[parameters('vmwebIISDSCModule')]",
          "configurationFunction": "[parameters('vmwebIISDSCFunction')]",
          "Properties": {
            "MachineName": "[concat('vmweb','-',copyIndex())]"
          }
        },
```
18. Refer to the image below to verify the code block pasted correctly, as the image represents what the code should look like



```
"autoUpgradeMinorVersion": true,
"settings": {
  "modulesUrl": "[parameters('vmWebIISDSCModule')]",
  "configurationFunction": "[parameters('vmWebIISDSCFunction')]",
  "Properties": {
    "MachineName": "[concat('vmWeb','-',copyIndex())]"
  }
},
"protectedSettings": {
}
```

19. In the **"protectedSettings":** property, **delete** the **"configurationUrlSasToken":** line
20. In the **JSON Outline**, click on **variables**
21. Right-click on the **vmwebArchiveFolder** variable and choose **Delete**
22. Right-click on the **vmwebArchiveFileName** variable and choose **Delete**
23. In the **JSON Outline**, click on **parameters**
24. Right-click on the **_artifactsLocation** parameter and choose **Delete**
25. Right-click on the **_artifactsLocationSasToken** parameter and choose **Delete**
26. At the end of the **parameters** section, but before the last **},**paste the following lines to add the vmwebIISDSCModule parameter
```
"vmwebIISDSCModule": {
  "type": "string",
  "defaultValue": "https://github.com/Azure/azure-quickstart-templates/raw/master/dsc-extension-iis-server-windows-vm/ContosoWebsite.ps1.zip"
},
```
**Note:** This tells ARM the location to install the IIS server extension on each virtual machine as they are deployed from the Azure Quickstart Template. For this lab, you will be using a zip file that is loaded on a GitHub repository.
27. After the **vmwebIISDSCModule** parameter, paste the following lines:
```
"vmwebIISDSCFunction": {
  "type": "string",
  "defaultValue": "ContosoWebsite.ps1\\ContosoWebsite"
}
```
**Note:** This will tell ARM which configuration function to run - this will apply the ContosoWebsite configuration.
28. Click **File > Save** to save the **ContosoIaaS_3.json** file in Visual Studio
29. Return to **PowerShell ISE**
30. Highlight the **New-AzureRmResourceGroupDeployment**command and click **Run selection**
**Note:** The deployment and installation will take approximately five minutes. As the script runs, you can see that the resources look pretty much the same, except for right at the end where the Microsoft.compute/virtualmachine/extension resource type is provisioned. You can see that each of them are a child resource of the parent virtual machine.
31. After the deployment is done, return to the **Azure portal**and refresh the rg-lod7332272 resource group
32. In the **Resources** list, click on the **as-vmweb** Availability set
33. Now, click on either one of the virtual machines listed, **vmweb-0** or **vmweb-1**
34. In the **SETTINGS** menu, click on **Extensions**
35. Verify that the **IIS** extension is listed and the **STATUS** is **Provisioning succeeded**
36. If you wish, you can return to the **as-vmweb** Availability set blade, and click on the other virtual machine to verify that the **IIS** extension shows the same status if you wish

So, we have now configured our remote servers to be application web servers, using ARM and PowerShell desired state configuration.

PreviousNext

From <https://labondemand.com/CloudClient/e8113556-1ed4-4179-8baa-edb932b8866d>

## Scenario D - Load Balancer

Now you will build a highly available web application service, accessible from our private virtual network. You will need an internal load balancer to manage traffic to both web servers.

1. In **Visual Studio**, return to the **contosoIaas_3.json** file
2. In the **JSON Outline**, click on **resources**
3. After the **as-vmwebName** availabilitySet resource at the end of the resource list, paste the following lines starting after the **}** on line 435

```
,
    {
  "name": "[parameters('vmwebLoadBalancerName')]",
  "type": "Microsoft.Network/loadBalancers",
  "apiVersion": "2015-06-15",
  "location": "[resourceGroup().location]",
  "dependsOn":  [
    "[parameters('vNetAzureName')]"
  ],
  "properties": {
  }
},
```

   **Note:** This adds the framework for a Load Balancer resource to the ARM template. It identifies a parameter for the load balancer name, a type, the API version, and a resource group. The load balancer must have a virtual network with the subnet already provisioned. So it needs to be added as a dependency. Now you will define different properties in the Load Balancer resource.
4. After the **"properties": {** line, paste the following to identify a front end IP configuration for the load balancer resource

```
  "frontendIPConfigurations": [
    {
      "properties": {
        "subnet": {
          "id": "[variables('vmwebSubnetRef')]"
        },
        "privateIPAddress": "10.0.1.20",
        "privateIPAllocationMethod": "Static"
      },
      "name": "LoadBalancerFrontend"
      }
  ],
```

   **Note:** An internal load balancer requires a front end IP configuration, which is the IP address. This tells the load balancer which subnet it's going use and assigns a static address in the Azure virtual network environment.
5. After the **frontendIPConfigurations** property, add a new line and paste the following to identify a backend pool

```
    "backendAddressPools": [
      {
        "name": "BackendPool1"
      }
    ],
```

6. After the **"backendAddressPools" property**, add a new line and paste the following to

identify load balancing rules

```
    "loadBalancingRules": [
      {
        "properties": {
          "frontendIPConfiguration": {
            "id":
"[concat(resourceId('Microsoft.Network/loadBalancers',parameters('vmwebLoadBalancerName')),'/frontendIpConfigurations/LoadBalancerFrontEnd')]"
          },
          "backendAddressPool": {
            "id":
"[concat(resourceId('Microsoft.Network/loadBalancers',parameters('vmwebLoadBalancerName')),'/backendAddressPools/BackendPool1')]"
          },
          "probe": {
            "id":
"[concat(resourceId('Microsoft.Network/loadBalancers',parameters('vmwebLoadBalancerName')),'/probes/lbprobe')]"
          },
          "protocol": "Tcp",
          "frontendPort": 80,
          "backendPort": 80,
          "idleTimeoutInMinutes": 15
        },
        "name": "lbRule"
      }
    ],
```

**Note:** Load balancing rules to determine exactly how traffic is going to be balanced. This load balancing rule ties together the front end IP configuration, the backend address pool, and the probe.

7. At the end of the **"loadBalancingRules"** property, add a new line paste the following to identify the probes needed by the load balancer resource

```
    "probes": [
      {
        "properties": {
          "protocol": "Tcp",
          "port": 80,
          "intervalInSeconds": 15,
          "numberofProbes": 2
        },
        "name": "lbprobe"
      }
    ]
```

**Note:** A network probe is needed so the load balancer can determine the health of any services in the backend address pool.

Now that you have defined a load balancer resource, you need to modify the network interface resources which are bound to the virtual machines. They are the resources that contain the configuration of the load balancer to which they are attached.

8. In the **resources** section of the **JSON Outline**, click on the **[concat('vmweb','-',copyIndex(),'-nic-0')]** network interface resource
9. In **"properties":** after the **closing ] bracket** of the **"ipConfigurations": "properties":** code block on line 336, paste the following

```
,
    "loadBalancerBackendAddressPools": [
        {
          "id":
"[concat(resourceId('Microsoft.Network/loadBalancers',parameters('vmwebLoadBalancerName')),'/backendAddressPools/BackendPool1')]"
        }
    ]
```

10. In the **"dependsOn":** property, add the following dependency after the current **vNet** dependency
    ```
    "[concat('Microsoft.Network/loadBalancers/', 
    parameters('vmwebLoadBalancerName'))]"
    ```
11. In the **JSON Outline**, click on **parameters**
12. After the **"vmwebIISDSCFunction":** parameter, paste the following lines
    ```
    "vmwebLoadBalancerName": {
      "type": "string",
      "defaultValue": "contoso-web-lb",
      "minlength": 1
    }
    ```
    **Note:** This adds the undefined vmwebLoadBalancerName parameter and assigns it a default value.
13. **Save** the **contosoIaas_3.json** file
14. Return to **PowerShell ISE**
15. Highlight the **New-AzureRmResourceGroupDeployment**cmdlet and click **Run selection**
16. After the deployment is done, return to the **Azure portal**and refresh the rg-lod7332272 resource group
17. In the **Resource group** blade, click on the **Load balancer**resource to review the settings that were deployed
18. **Close** the **Load balancer** blade

PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/e8113556-1ed4-4179-8baa-edb932b8866d>

AZ00009: Building Azure Resource Manager Infrastructure

1 Hr 58 Min Remaining

Instructions

Resources

Help

100%

## Conclusion

In this lab, you have gained experience deploying both private and remote VMs, configuring the VMs with DSC to enable IIS, and adding a load balancer to help distribute traffic across your network.

PreviousEnd

From <https://labondemand.com/CloudClient/e8113556-1ed4-4179-8baa-edb932b8866d>

# AZ00010: Deploying VMs in an ARM Template

19 September 2018 10:49

AZ00010: Deploying VMs in an ARM Template

2 Hours Remaining

Instructions

Resources

Help

100%

## Building ARM Services

***Creating PaaS services in Azure with ARM Templates***

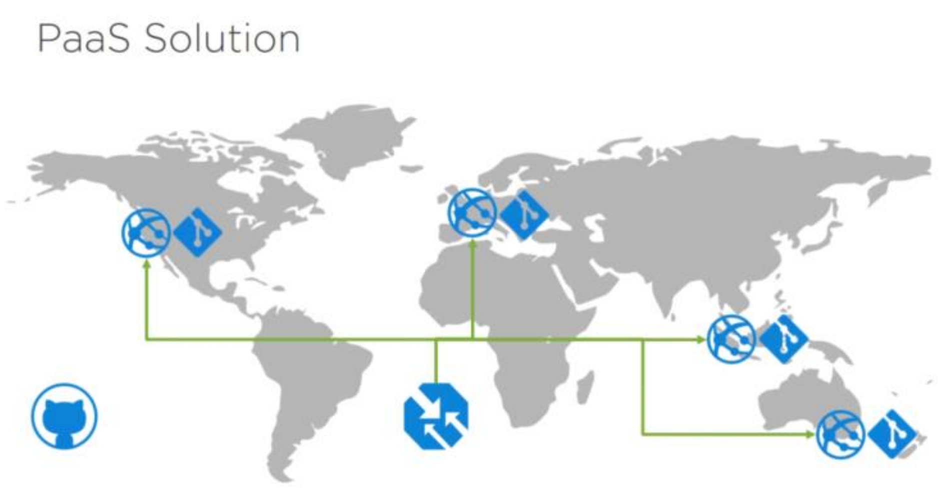
## Overview

This lab will guide the user through creating PaaS services in Azure with ARM templates. You will deploy web server farms in different geographical locations, deploy websites to these server farms, configure and deploy a Traffic Manager, and configured web applications from code stored on GitHub.

### Lab Goals

- Create new ARM template
- Deploy full PaaS solution
- Populate web site from Github source control

### Product overview



**Capabilities or components used in this scenario:** • Web Farms • Virtual Machines • Traffic Manager • GitHub Web Application code

Next

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

2 Hours Remaining

Instructions

Resources

Help

100%

## Building ARM Services Overview

In this lab, you are going to deploy a combination of web server farms and websites throughout the world. Four of them in total. You will then populate those sites with web applications from GitHub source control, and manage access to the websites using an Azure Traffic Manager. You will use the Microsoft.Web resource provider, provisioning server farms, sites, and source control resources. You will also deploy a Traffic Manager, which falls under the Microsoft.Network resource provider.

### Scenario A - Create a new template

To get started we will RDP into a jumphost where we have pre-loaded PowerShell along with ARM templates that we'll be working with.

1. On the **Resources** tab in the guide application, click **jumphost RDP profile** to download the jumphost RDP profile
2. Open the downloaded RDP profile
3. Copy and paste @lab.CloudResourceGroup(787).Name.southcentralus.cloudapp.azure.cominto the **Computer** field and click **Connect**, on first connect you will need to click the checbox [ x ] Don't ask me again for connections to this computer
4. For the password, copy and paste j!DaeMJ60+ and click **Connect**
5. On the next dialog click [ x ] Don't ask me again for connection to this computer then click [Yes], this will bypass certificate issues
6. On the desktop, click the **Start** button and click **Visual Studio 2017** - it may take couple of minutes for Visual Studio to open
7. If prompted to sign in to Visual Studio, click **Sign in**
8. In the **Email or phone** field, type User1-7332388 @cloudplatimmersionlabs.onmicrosoft.com and click **Next**
9. In the **Password** field, type j!DaeMJ60+
10. Click **Sign in**
11. If prompted with the **Choose your color theme** screen, select a color theme
12. Click **Start Visual Studio**
13. In the main navigation, click on **View > Other Windows > JSON Outline** to activate the view
14. In Visual Studio, click **File > New > File**
15. In the **Search Installed Templates** box, type **JSON**
16. In the **search results**, double-click the **JSON file** template option
17. In between the brackets, **Paste** the following lines into the blank line
```
"$schema":
"http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate
.json#",
"contentVersion": "1.0.0.0",
```
18. Click on **File > Save json1.json As**
19. In the **Quick access** menu, click on **Documents**
20. In the **file list**, open **HOL_Lab_Files/4_Building_Azure_Resource_Manager_Services**and name the file **contosoPaaS.json** and click **Save**
**Note:** A JSON file has a structure that includes a schema declaration, parameters, variables, resources, and outputs. We will now add these sections to the file.
21. At the end of the **"contentVersion":** line, type a **comma** and press **Enter**

22. Begin typing **parameters** and Intellisense will present you with the **parameters** option
23. Press **tab** to add a parameter section to the template
    **Note:** If you pause for a few seconds after accepting the parameter suggestion, Intellisense will add the required : {} at the end of the "parameters" line.
24. Add a **comma** to the end of the **parameters** entry and press **Enter**
25. After the **parameters** entry, begin typing **variables**
26. Press **tab** to accept the **variables** option
27. Add a **comma** to the end of the **variables** entry and press **Enter**
28. After the **variables** entry, begin typing **resources**
29. Press **tab** to accept the **resources** option
30. Add a **comma** to the end of the **resources** entry and press **Enter**
31. After the **resources** entry, begin typing **outputs**
32. Press **tab** to accept the **outputs** option
33. **Save** the contosoPaaS.json file
34. **Close** the contosoPaaS.json file
35. In **Visual Studio**, click on **File > Open > File**
36. Click on **Documents > HOL_Lab_Files/4_Building_Azure_Resource_Manager_Services**
37. Double-click **contosoPaas.json** to re-open the file with the JSON Outline view activated

PreviousNext

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario B - Prepare Deployment Script

1. In the jumphost, click the **Start** button, type **Powershell ISE**and select the **Windows PowerShell ISE Desktop App**
2. In the **PowerShell** window, paste the following line and press **Enter** to log in to Azure
   `Add-AzureRmAccount`
3. In the **Email or phone** field, enter User1-7332388 @cloudplatimmersionlabs.onmicrosoft.com and click **Next**
4. In the **Password** field, enter j!DaeMJ60+
5. Click **Sign In**
6. Click **File > Open**
7. In the **HOL_Lab_Files/4_Building_Azure_Resource_Manager_Services**folder, double-click **Deploy-ContosoPaas_4.ps1** to open it
8. In the **Define Deployment Variables** section of the **Deploy-ContosoPaas_4.ps1** file, replace **[resource group location]**with **South Central US**
9. In the **Define Deployment Variables** section, replace **[resource group name]** with @lab.CloudResourceGroup(787).Name
10. **Save** the Deploy-ContosoPaas_4.ps1 file

The PowerShell script is now ready to use to deploy resources from the JSON template. Now you will add resources to the template.

PreviousNext

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

## Scenario C - Web Farm Template Resources

You will now create the necessary resources to deploy web server farms, and deploy them before validating the deployment was successful.

1. In **Visual Studio**, return to the **contosoPaaS.json** file
2. In the **JSON Outline**, right-click on **Resources** and select **Add New Resource**
3. In the **Search** box, type **App**
4. In the **search results**, choose the **App Service Plan (Server Farm)**
5. In the **Name** box, type **farm**
6. Click **Add**
   **Note:** Because the template will be deploying multiple versions of the resource using the copy property, the name parameter now needs to be refactored so its unique for each copy.
7. In the **JSON Outline**, double-click on the **parameters**header
8. In the **parameters** section, click on **farmName**
9. Change the parameter name **farmName** to **webAppNamePrefix**
10. After the **"minLength":** property, paste the following to add a default name value
    ```
    ,
    "defaultValue": "contoso"
    ```
11. In the JSON Outline in the **parameters** section, click on **farmSkuName**
12. In the **"defaultValue":** property, change **F1** to **S1** which will set the SKU used to deploy the servers to Standard-1
13. After the **farmSkuName** parameter code block, paste the following lines before the **closing }, bracket** of the parameters code block which will identify the geographical locations where the servers will be created
    ```
    ,
    "webAppLocations": {
      "type": "array",
      "defaultValue": [ "Australia Southeast", "West US", "West Europe", 
    "Southeast Asia" ]
    },
    ```
14. In the **resources** section of the **JSON Outline**, click on the **farm** resource
15. In between the **"apiVersion":** property and the **"sku":**property, paste the following lines
    ```
      "copy": {
        "count": "[length(parameters('webAppLocations'))]",
        "name": "farmCopy"
      },
    ```
    **Note:** This copy function will calculate the number of entries in the webAppLocations parameter and iterate the copy function as many times as needed to create a server in each location.
16. In the **"name":** property, replace **"[parameters('farmName')]",** with the following value
    ```
    "[concat(parameters('webAppNamePrefix'),'-',copyIndex(),'-',uniqueString(
    resourceGroup().id))]",
    ```
    **Note:** This name definition will create a name for each server resource deployed that will contain the name contoso, the number of the server, and a unique string to identify it within the resource group.
17. In the **"location":** property, replace **"[resourceGroup().location]",** with the following
    ```
    "[parameters('webAppLocations')[copyIndex()]]",
    ```
    **Note:** As the copy job iterates through the resource deployment, the interpreted value will change to match the corresponding location in the webAppLocations array.

18. In **"tags":,** replace the line **"displayName": "farm"** with **"displayName": "[concat(parameters('webAppNamePrefix'),'-',copyIndex(),'-',uniqueString(resourceGroup().id))]"**
19. In **"properties":**, replace the line **"name": "[parameters('farmName')]",** with **"name": "[concat(parameters('webAppNamePrefix'),'-',copyIndex(),'-',uniqueString(resourceGroup().id))]",**
20. Click **File > Save**
21. Return to **PowerShell ISE**
22. In the **Define Deployment Variables** section, highlight all the lines and click **Run selection**
23. In the **Deploy Resources** section, highlight the **New-AzureRMResourceGroupDeployment** command and click **Run selection**
24. If you already logged into the azure portal per instructions on the Resource tab (above), please open that browser to the Azure portal and skip the steps for this instruction
    The next steps will log you into the Azure portal for this lab:
    . Click here to open the Azure portal
    . In the **Email or phone** field, enter User1-7332388 @cloudplatimmersionlabs.onmicrosoft.com and click **Next**
    . In the **Password** field, enter j!DaeMJ60+ and click **Sign in**
    . In the **Stay signed in?** window, click **No**
    . If a **Welcome to Microsoft Azure** pop-up window appears, click **Maybe Later** to skip the tour
25. In the **Favorites** menu on the left, click on **Resource groups**
26. In the **Resource group** blade, click on the @lab.CloudResourceGroup(787).Name resource group
27. In the list of resources, you can see four new App Service plans all beginning with contoso-
28. Click on the **contoso-0 App Service plan** where you can review its settings and see it was created in the **Australia Southeast** region
29. Close the **contoso-0 App Service plan** blade

PreviousNext

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

2 Hours Remaining

Instructions

Resources

Help

100%

## Scenario D - Web App Template Resources

Now you will add functional websites, attach them to the server farms, and verify their deployment.

1. In **Visual Studio**, return to the **ContosoPaaS.json** file
2. In the **JSON Outline**, right-click on **resources**
3. Choose **Add New Resource**
4. In the **Search** box, type **Web**
5. In the **search results**, select **Web App**
6. In the **Name:** box, type **site**
7. Click **Add**

**Note:** You will now delete the siteName variable and replace it in the webapp "name:" property with a function similar to what was used for the web server resources.

8. In the **JSON Outline**, double-click on the **variables** header
9. In the **variables** section of the **JSON Outline**, right-click on **siteName** and select **Delete**
10. In the **resources** section of the **JSON Outline**, click on **site**
11. In the **"name":** property, replace **"[variables('siteName')]"**with the following line
    ```
    "[concat(parameters('webAppNamePrefix'),'-
    site-',copyIndex(),'-',uniqueString(resourceGroup().id))]"
    ```
12. In the **"tags": "displayName":** property, replace **"site"** with the following
    ```
    "[concat(parameters('webAppNamePrefix'),'-
    site-',copyIndex(),'-',uniqueString(resourceGroup().id))]"
    ```
13. In the **"dependsOn": [],** property, replace **[resourceId('Microsoft.Web/serverfarms....]** with
    ```
        "farmCopy"
    ```
14. Above the **"dependsOn":** property, insert a new line and paste in the **copy function** below
    ```
          "copy": {
        "count": "[length(parameters('webAppLocations'))]",
        "name": "siteCopy"
      },
    ```
15. In **"properties": "name"** property, replace **"[variables('siteName')]"** with **"[concat(parameters('webAppNamePrefix'),'-site-',copyIndex(),'-',uniqueString(resourceGroup().id))]"**
16. Save the **contosoPaaS.json** file
17. Return to **PowerShell ISE**
18. In the **Deploy Resources** section, highlight the **New-AzureRMResourceGroupDeployment** command
19. Click **Run selection**
20. Return to the **Azure portal**
21. In the **Resource group** blade, click **Refresh**
22. In the **Resource** list, you can verify the four **App service**resources have been deployed
23. Click on the **App Service** resources starting with **contoso-site-0**
24. In the **Overview** blade for the **contoso-site-0** App Service, click on the **URL** link to open a new browser tab and view the new web application
25. Take notice of the text reading **"Your App Service app is up and running"**

PreviousNext

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Scenario E - Azure Traffic Manager

Now that you have created web server farms and deployed web sites running across the world, you will create a public-facing service so the public can access the websites without having to know the specific URL for a specific web app instance. You will deploy an Azure Traffic Manager resource which will help direct a customer to the endpoint that will provide the best possible experience. Azure Traffic Manager can also be used for failover protection, redundancy, round robbin, and load distribution.

1. Return to the **ContosoPaaS.json** file in **Visual Studio**

2. In the **resources** section, paste the following lines in between the **}** and closing **],** bracket of the resources section to begin a new code block for the traffic manager resource
```
,
{
  "name": "[concat(parameters('webAppNamePrefix'),'-
tm-',uniqueString(resourceGroup().id))]",
  "type": "Microsoft.Network/trafficManagerProfiles",
  "apiVersion": "2015-11-01",
  "location": "global",
}
```
   **Note:** The traffic manager will use the same function for naming as the other resources you have added. The location is set to global because the traffic manager is a resource that doesn't require a specific geographical location.
3. Insert a new blank line under **"location": "global",**
4. On the new blank line, under the **"location":** property, paste the following
```
      "dependsOn": [
    "siteCopy"
  ],
```
   **Note:** This creates a dependency in the template requiring the siteCopy function to be completed before the traffic manager resource can be deployed.
5. Under the **"dependsOn":** property, paste the following
```
  "properties": {
    "profileStatus": "Enabled",
    "trafficRoutingMethod": "Performance",
    "dnsConfig": {
      "relativeName": "[concat(parameters('webAppNamePrefix'),'-
tm-',uniqueString(resourceGroup().id))]",
      "ttl": 30
    }
  },
```
   **Note:** This will automatically enable the traffic manager, establish the routing method, and configure the DNS for the traffic manager
6. Under the **"properties":** property, before the closing **}** paste the following
```
   "monitorConfig": {
      "protocol": "HTTP",
      "port": 80,
      "path": "/"
    },
```
   **Note:** This establishes the monitoring configuration the Traffic Manager will apply to each endpoint using the HTTP protocol on port 80 with a flat relative path.
7. After **"monitorConfig":** property, in the properties code block paste the following to open a code block for the endpoints
```
    "endpoints": [
],
```
8. In the **blank line** in between the [ ] brackets, paste the following endpoint code block **FOUR** times - one for each endpoint that needs to be configured
```
      {
        "name": "[concat(parameters('webAppNamePrefix'),'-
endpoint-','0')]",
        "type":
"Microsoft.Network/trafficManagerProfiles/azureEndpoints",
        "properties": {
          "targetResourceId":
"[resourceId('Microsoft.Web/sites/',concat(parameters('webAppNamePrefix')
,'-site-','0','-',uniqueString(resourceGroup().id)))]",
          "endpointStatus": "Enabled"
        }
      },
```
   **Note:** The name of the endpoint will be **webAppNamePrefix-endpoint-** followed by a

resource number.

9. In the **"name":** property of the second endpoint code block, change the **-0-** to **-1-**
10. In the **"targetResourceId":** property of the second endpoint code block, change the **-0-** to **-1-**
11. In the **"name":** property of the third endpoint code block, change the **-0-** to **-2-**
12. In the **"targetResourceId":** property of the second endpoint code block, change the **-0-** to **-2-**
13. In the **"name":** property of the last endpoint code block, change the **-0-** to **-3-**
14. In the **"targetResourceId":** property of the second endpoint code block, change the **-0-** to **-3-**
15. After the **closing ] bracket** of the endpoints code block, verify the end of the file has the following bracket structure
    **Note:** The end of the file should now look like this:
```
                "endpointStatus": "Enabled"
            }
          }
        ]
      }
    }
],
"outputs": { }
}
```
16. **Save** the **ContosoPaaS.json** file
17. Return to **PowerShell ISE**
18. Highlight the **New-AzureRMResourceGroupDeployment**cmdlet and click **Run selection**
19. Return to the **Azure portal**
20. **Close** the **contoso-site-0** blade
21. **Refresh** the **Resource group** blade
22. In the **resource** list, click on the **Traffic Manager profile**resource to open it
    **Note:** It will be the resource with TYPE = Traffic Manager profile
23. From the **Overview** blade, review the four deployed endpoints and the regions in which the endpoints were deployed
24. Click on the **DNS Name URL** to open the **App Service** via the newly deployed **Traffic Manager** based **URL**
    **Note:** Leave this tab open so you can easily return to it to view the new content you are now going to deploy to the sites.

Now that we've deployed a fully functional web platform, it's time to take the last step, which is to populate our websites with a specific web application.
PreviousNext

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

1 Hr 59 Min Remaining
Instructions

Resources

Help
100%

## Scenario F - Configure Web Site from GitHub Source Control

You will use a sub-resource to deploy content to each of the web sites you just deployed using a GitHub Source repository to build the web site content.

1. Return to the **contosoPaaS.json** file in **Visual Studio**

2. In the **resources** section of the **JSON Outline**, click on the -site- **Web App resource** named **[concat(parameters('webAppNamePrefix'),'-site-',copyIndex(),'-',uniqueString(resourceGroup().id))]**
3. At the end of the **"properties":** property, paste the following
```
,
  "resources": [
    {
      "name": "web",
      "type": "sourcecontrols",
      "apiVersion": "2015-08-01",
      "dependsOn": [
        "[concat(parameters('webAppNamePrefix'),'-
site-',copyIndex(),'-',uniqueString(resourceGroup().id))]"
      ],
```
   **Note:** This sub-resource will run as a dependent resource to the web site resource.
4. After the **"dependsOn": closing ] bracket**, paste the following
```
    "properties": {
        "RepoUrl": "[parameters('repoURL')]",
        "branch": "[parameters('branch')]",
        "IsManualIntegration": true
      }
    }
  ]
```
   **Note:** These properties will help you identify which GitHub source control repository the web site will be built from.
5. In the **parameters** section, paste the following after the **webAppLocations** parameter to add a repoURL parameter
```
,
"repoURL": {
  "type": "string",
  "defaultValue": "https://github.com/davidebbo-
test/Mvc52Application.git"
},
```
6. After the **"repoURL":** parameter, paste the following to add a branch parameter
```
    "branch": {
  "type": "string",
  "defaultValue": "master"
}
```
   **Note:** You have now defined, in the ARM template, where ARM should obtain the code to populate the site resources as they are built.
7. **Save** the **ContosoPaaS.json** file
8. Return to **PowerShell ISE**
9. Highlight the **New-AzureRMResourceGroupDeployment**command
10. Click **Run selection**
11. Return to the open browser session that is pointing to the **Traffic Manager**
12. In the **browser**, click **Refresh** to view the site content reflecting an **ASP.NET** based application versus the previous **App Service** based application that you just deployed from GitHub source control

PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

AZ00010: Deploying VMs in an ARM Template

1 Hr 59 Min Remaining

Instructions

Resources

Help
100%

## Conclusion

In this lab, you have created an ARM template for a PaaS solution, or Platform as a Service solution, which is the foundation of Contoso's public-facing services. In order to do that, you deployed web server farms, deployed websites, configured and deployed a Traffic Manager in a performance algorithm, and configured web applications from code stored in GitHub.
PreviousEnd

From <https://labondemand.com/CloudClient/5a2e822b-a04a-4f0d-9a78-4d9c562cd1cc>

# AZ00011: Authoring Deconstructed Templates

20 September 2018 09:34

AZ00011: Authoring Deconstructed Templates

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Authoring Deconstructed Templates

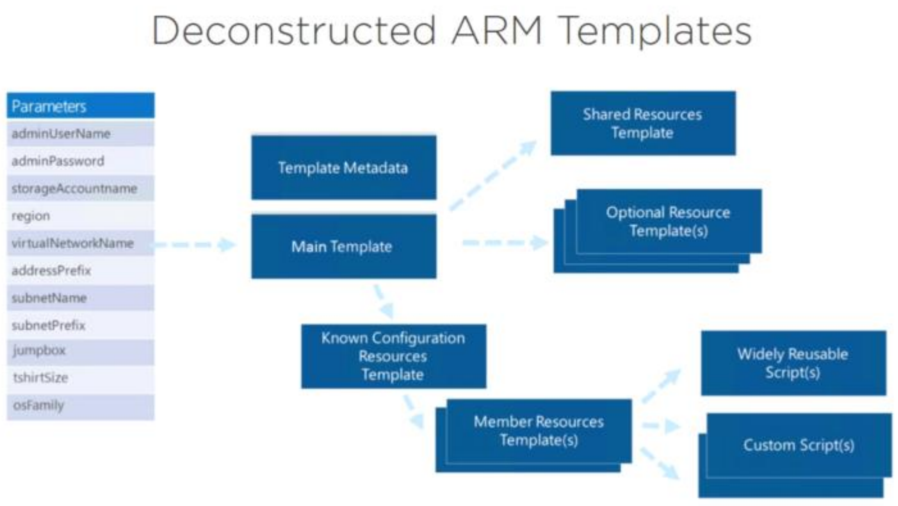
## Overview

Deconstructed templates break down an ARM template into reusable templates that can be shared among several different solutions. This is accomplished by creating a master template, which is then used to orchestrate several different resource templates to create the resources required by the solution.

### Lab Goals

- Understand deconstructed template structure
- Share outputs between templates
- Explore nested deployments
- Store deployment secrets in a Key Vault
- Link to shared templates
- Deploy a nested template solution

### Product overview



**Capabilities or components used in this scenario:**

- ARM template file
- ARM parameters file

- Individual resource template files
- Key Vault

Next

From <https://labondemand.com/CloudClient/0bf6522c-11d0-4764-9f09-68085ae07ec0>

AZ00011: Authoring Deconstructed Templates

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Authoring Deconstructed Templates

In a deconstructed template, you use a core feature of the ARM template - outputs - to make it work. In this structure, other templates can use the outputs values from a shared template. In this lab, you will learn how outputs are created and shared by viewing examples of ARM nested templates.
A sample deconstructed template file can be found in the Documents library in the **HOL_Lab_Files/5_Authoring_Deconstructed_Templates/nested**folder. When you look at the structure, you will note the following file structure:

- Master template
- Parameter file
- Shared templates for specific resource deployments
  - NIC template
  - publicIP template
  - storageAccounts template
  - virtualMachine template

The individual resource templates define parameters and produce outputs which are used by the master template. The master template can then pass outputs to other shared templates.

### Scenario A - Using Outputs in Nested Deployments

In this scenario, you examine the master template file and the parameters file, then review the shared template files that are used by the master template. To get started we will RDP into a jumphost where we have pre-loaded PowerShell along with ARM templates that we'll be working with.

1. On the **Resources** tab in the guide application, click **jumphost RDP profile** to download the jumphost RDP profile
2. Open the downloaded RDP profile
3. Copy and paste @lab.CloudResourceGroup(786).Name.southcentralus.cloudapp.azure.cominto the **Computer** field and click **Connect**
4. For the password, copy and paste Bh0De0!!De and click **Connect**
5. On the next dialog click [ x ] Don't ask me again for connection to this computer then click [Yes], this will bypass certificate issues
6. On the desktop, click the **Start** button and click **Visual Studio 2017** - it may take a few minutes for Visual Studio to open
7. If prompted to sign in to Visual Studio, click **Sign in**
8. In the **Email or phone** field, enter User1-7332421

@cloudplatimmersionlabs.onmicrosoft.com

9. Click **Next**
10. In the **Password** field, enter Bh0De0!!De
11. Click **Sign in**
12. If prompted with a **Choose your color theme** window, select a color theme
13. Click **Start Visual Studio**
14. In the main navigation, click on **View > Other Windows > JSON Outline** to activate the view
15. In Visual Studio, click **File > Open > File**
16. From the **Quick access** list, click the **Documents** folder
17. From the list of folders, locate the **HOL_Lab_Files** folder and double-click to display the contents
18. Navigate to the following folder - **5_Authoring_Deconstructed_Templates** by double-clicking the folder
19. Navigate to the following folder - **nested** by double-clicking the folder to display the contents
20. In the folder, double-click the **azureDeploy_5.parameters.json** parameters file
21. Click **File > Open > File**
22. In the **HOL_Lab_Files/5_Authoring_Deconstructed_Templates/nested**folder, double-click **azureDeploy_5.json** which is the master template of the nested deployment
23. In the **JSON Outline** panel, double-click the **resources**section to review the five different resources that are deployed via external shared templates as a deployment resource type
    **Note:** You'll see that this template is designed to deploy five resources. All of the resources are being deployed via external shared templates and are being called from the master template using the Microsoft.Resources resource provider and the deployment resource type.
24. In the **resources** section of the **JSON Outline**, click on the first resource - **[nestedDeploymentNameVnet]**
    **Note:** This code block creates a virtual network from an ARM template in a nested deployment .
```
{
  "apiVersion": "[variables('deploymentApiVersion')]",
  "name": "[parameters('nestedDeploymentNameVnet')]",
  "type": "Microsoft.Resources/deployments",
  "properties": {
    "mode": "Incremental",
    "templateLink": {
      "uri": "[concat(parameters('baseTemplateUri'), '/shared/',
parameters('sharedTemplateNameVnet'))]",
      "contentVersion": "1.0.0.0"
    },
    "parameters": {
      "vNetName": { "value": "[parameters('vNetName')]" },
      "vNetApiVersion": { "value": "[variables('vNetApiVersion')]" },
      "vNetRange": { "value": "[parameters('vNetRange')]" },
      "subnetProdRange": { "value": "[parameters('subnetProdRange')]" },
      "subnetDevRange": { "value": "[parameters('subnetDevRange')]" },
      "deploymentType": { "value": "[parameters('deploymentType')]" }
    }
  }
},
```
    **Note:** This deployment has a name which is populated from a parameter, **nestedDeploymentNameVnet**, indicating that this is the name of the nested deployment. The nested deployment is a template which deploys a virtual network.
25. In the **"properties":** for this resource, note the **mode**setting:
```
"properties": {
  "mode": "Incremental",
```

**Note:** The incremental mode indicates that the deployment will only deploy changes into the ARM environment. It will not redeploy the entire solution.

26. Also, in the **"properties":** for this resource a **templateLink** is provided:

```
"templateLink": {
  "uri": "[concat(parameters('baseTemplateUri'), '/shared/', 
parameters('sharedTemplateNameVnet'))]",
      "contentVersion": "1.0.0.0"
},
```

**Note:** The template link is a concatenation of the baseTemplateUri, the folder, and then a shared template name.

27. In **Visual Studio**, click the **azureDeploy_5.parameters.json**tab
28. Locate the **baseTemplateUri** parameter definition:

```
"baseTemplateUri": {
  "value": 
"https://raw.githubusercontent.com/jamesbannan/pluralsight/master/microso
ft-azure-resource-manager-mastering/nested"
}
```

**Note:** The baseTemplateUri in this example above is a public location, but the shared templates can also reside in a local file path.

29. In **Visual Studio**, return to the **azureDeploy_5.json** tab

**Note:** The shared template structure looks exactly the same as a traditional ARM template: it's expecting values to be provided for several parameters. These values are being provided directly by the master template.

30. In the **resources** section of the **JSON Outline**, click on the second resource named **[nestedDeploymentNamePublicIp]**

```
    {
  "apiVersion": "[variables('deploymentApiVersion')]",
  "name": "[parameters('nestedDeploymentNamePublicIp')]",
  "type": "Microsoft.Resources/deployments",
  "properties": {
    "mode": "Incremental",
    "templateLink": {
      "uri": "[concat(parameters('baseTemplateUri'), '/shared/', 
parameters('sharedTemplateNamePublicIp'))]",
      "contentVersion": "1.0.0.0"
    },
    "parameters": {
      "publicIpNamePrefix": { "value": 
"[parameters('publicIpNamePrefix')]" },
      "publicIpApiVersion": { "value": 
"[variables('publicIpApiVersion')]" },
      "deploymentType": { "value": "[parameters('deploymentType')]" }
    }
  }
},
```

**Note:** The code structure is consistent across all deployments. The first two resources you looked at did not have dependencies upon any other resource, so they can be deployed in isolation. Now you will look at a deployed resource that has dependencies.

31. In the **resources** section of the **JSON Outline**, click on the fourth resource **[nestedDeploymentNameNic]**

```
{
  "apiVersion": "[variables('deploymentApiVersion')]",
  "name": "[parameters('nestedDeploymentNameNic')]",
  "type": "Microsoft.Resources/deployments",
  "dependsOn": [
    "[concat('Microsoft.Resources/deployments/', 
parameters('nestedDeploymentNameVnet'))]",
    "[concat('Microsoft.Resources/deployments/', 
parameters('nestedDeploymentNamePublicIp'))]"
```

```
  ],
  "properties": {
    "mode": "Incremental",
    "templateLink": {
      "uri": "[concat(parameters('baseTemplateUri'), '/shared/', 
parameters('sharedTemplateNameNic'))]",
      "contentVersion": "1.0.0.0"
    },
    "parameters": {
      "nicName": { "value": "[parameters('nicName')]" },
      "nicApiVersion": { "value": "[variables('nicApiVersion')]" },
      "subnetID": { "value": 
"[reference(parameters('nestedDeploymentNameVnet')).outputs.subnetID.valu
e]" },
      "publicIpID": { "value": 
"[reference(parameters('nestedDeploymentNamePublicIp')).outputs.publicIpI
D.value]" }
    }
  }
},
```

**Note:** This deployment is configured to deploy a network interface resource. Note it has a **dependsOn** block, but the dependencies are actually **dependent upon deployments**, so this deployment will not start until ARM has received confirmation that those deployments have completed successfully.

You also need to pass information from both the vNet deployment and the public IP deployment to the **virtualNetwork.json** shared template in order for it to be provisioned properly.

32. In the **[nestedDeploymentNameNic]** resource, review the **parameters** section

```
    "parameters": {
      "nicName": { "value": "[parameters('nicName')]" },
      "nicApiVersion": { "value": "[variables('nicApiVersion')]" },
      "subnetID": { "value": 
"[reference(parameters('nestedDeploymentNameVnet')).outputs.subnetID.valu
e]" },
      "publicIpID": { "value": 
"[reference(parameters('nestedDeploymentNamePublicIp')).outputs.publicIpI
D.value]" }
    }
```

**Note:** There are two parameters, **subnetID** and **publicIpID**, that receive their values from an output in the associated deployment name. The **subnetID** is being provided by an output in the **virtualNetwork.json**template and the **publicIpID** is being provided by an output in the **publicIP.json** template.

**Note:** The values of these deployments have been passed from the individual shared templates to the master template, which is now passing it on to another template for the deployment of a different resource.

33. Click **File > Open > File > Nested**
34. Double-click the **shared** folder, and then double-click the **virtualNetwork.json** parameters file to open
35. Scroll down to the **outputs** section to view the output called **subnetID**:

```
  "outputs": {
    "subnetID": {
      "type": "string",
      "value": "[variables('vNetSubnetID')]"
    }
  }
```

**Note:** It's a string, and the value is populated from a variable called **vNetSubnetID**.

36. From the **JSON Outline** view, locate and double-click the **variables** section, then locate and click the **vNetSubnetID**variable:

```
"variables": {
    "subnetProdName": "subnet-prod",
    "subnetDevName": "subnet-dev",
    "vNetID": "[resourceId('Microsoft.Network/virtualNetworks',
parameters('vNetName'))]",
    "vNetSubnetID": "[concat(variables('vNetID'), '/subnets/', 'subnet-',
parameters('deploymentType'))]"
  },
```

**Note:** The properties of the **vNetSubnetID** variable are a concatenation of the variable of a **vNetID**, which is another variable that is defined as the **resourceId** of the virtual network. This means the virtual network is building up variables in a very traditional manner and calculating those variables before outputting them.

37. In **Visual Studio**, click the **azureDeploy_5.json** tab
38. Locate the parameter called **deploymentType**:

```
"deploymentType": {
  "type": "string",
  "allowedValues": [ "dev", "prod" ],
  "defaultValue": "dev",
  "metadata": {
    "description": "String to determine which network the VM is deployed
to."
  }
},
```

**Note:** The **deploymentType** parameter defines whether your deployment will be to a development environment or to a production environment. The parameter is simply a string with two allowed values, **dev** and **prod**. The default value is set to **dev**.
**Note:** There are four other shared templates in the nested deployment shared folder that create specific resources, which are self explanatory based on their naming.
The **networkInterface.json, publicIP.json, storageAccount.json, and virtualMachine.json templates** deploy the resource and create an output that can be used in the master template and other shared templates.

PreviousNext

From <https://labondemand.com/CloudClient/0bf6522c-11d0-4764-9f09-68085ae07ec0>

AZ00011: Authoring Deconstructed Templates

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Scenario B - Using Key Vaults

Next, you will deploy a new Azure key vault, add a secret value to that key vault, and use that information to update the deployment parameters of your nested template.

1. In **Visual Studio**, verify the **azureDeploy_5.json** tab is selected; and double-click **parameters** to display all parameters sections
2. In the **parameters** section of the **JSON Outline**, locate the **"vmAdminPassword":** parameter:

```
    "vmAdminPassword": {
  "type": "securestring",
  "metadata": {
    "description": "Password for the local admin account."
  }
},
```

**Note:** This parameter has a type of secure string. You cannot include a secure string in a parameters file. You will use a key vault as a safe, encrypted location to store secure information, and extract and call upon in a deployment scenario.

3. In **Visual Studio**, click the **azureDeploy_5.parameters.json**tab
4. Locate the **"vmAdminPassword":** parameter on line 62 which contains a reference to **keyVault**:

```
"vmAdminPassword": {
  "reference": {
    "keyVault": {
      "id": "/subscriptions/<SUBSCRIPTION_ID>/resourceGroups/[resource group name]/providers/Microsoft.KeyVault/vaults/[resource group name]contosovault"
    },
    "secretName": "vmAdminPassword"
  }
},
```

5. In the jumphost, click the **Start** button, type **Powershell ISE**and select the **Windows PowerShell ISE Desktop App**
6. Click **File > Open**
7. In the **HOL_Lab_Files/5_Authoring_Deconstructed_Templates**folder, double-click **Deploy-KeyVault.ps1**
8. In the **Define Variables** section, replace **[resource group location]** with **South Central US**
9. In the **Define Variables** section, replace **[resource group name]** with **@lab.CloudResourceGroup(786).Name**
10. In the **Define Variables** section, for $keyVAultName replace **$resourceGroupName** with '**lod7332421**'
11. In the **PowerShell** window, paste the following line and press **Enter** to launch the Azure login:

```
  Add-AzureRmAccount
```

12. In the login screen's **Email or phone** field, enter User1-7332421 @cloudplatimmersionlabs.onmicrosoft.com
13. Click **Next**
14. In the **Password** field, enter Bh0De0!!De
15. Click **Sign in**
16. Highlight all the **variables** in the **Define variables** section and click **Run selection**
17. In the **Create Key Vault** section, highlight the **New-AzureRmKeyVault** cmdlet and click **Run selection**
18. In the **Add Password to Key Vault** section, highlight all lines and click **Run selection**
19. If you already logged into the azure portal per instructions on the Resource tab (above), please open that browser to the Azure portal and skip the steps for this instruction
    The next steps will log you into the Azure portal for this lab:
    . [Click here to open the Azure portal]()
    . In the **Email or phone** field, enter User1-7332421 @cloudplatimmersionlabs.onmicrosoft.com and click **Next**
    . In the **Password** field, enter Bh0De0!!De and click **Sign in**
    . In the **Stay signed in?** window, click **No**
    . If a **Welcome to Microsoft Azure** pop-up window appears, click **Maybe Later** to skip the tour
20. In the **All services** search dialog box at the top, type **key vaults**
21. In the **search results**, click on the Key vault name to open the blade
22. Here, you can verify the ***contosovault** key vault has been created and populated for use in the nested deployment template
23. Under SETTINGS click **Scerets** to see the vmAdmindPassword secret you populated into the vault

24. Return to **PowerShell ISE**
25. Click **File > Open**
26. In the **HOL_Lab_Files/5_Authoring_Deconstructed_Templates**folder, double-click **Deploy-NestedSolution.ps1**
27. In the **Define Variables** section, replace **[resource group location]** with **South Central US**
28. In the **Define Variables** section, replace **[resource group name]** with **@lab.CloudResourceGroup(786).Name**
29. In the **Get Subscription ID** section, highlight the command and click **Run selection**
30. Highlight the **subscriptionID** that is returned
31. Right-click the highlighted value and select **copy**
32. In **Visual Studio**, return to the **azureDeploy_5.parameters.json** file
33. In the **vmAdminPassword parameter**, replace **< SUBSCRIPTION_ID >** with the **subscriptionID** value returned in PowerShell
34. On the **vmAdminPassword parameter** line in the first place from the left, replace **[resource group name]** with **@lab.CloudResourceGroup(786).Name**
35. On the **vmAdminPassword parameter** line replace the next **[resource group name]** with **lod7332421**
36. Click **File > Save azureDeploy_5.parameters.json** to save your changes
37. In **Powershell ISE**, return to the **Deploy-NestedSolution.ps1** file
38. In the **Define Variables** section, highlight all the **variables**and click **Run selection**
39. In the **Deploy IaaS Solution** section, highlight all the lines in the **New-AzureRmResourceGroupDeployment** command and click **Run selection**

This script will now deploy the nested templates and created a storage account, virtual network with subnets, network interface, public IP address, and a virtual machine. After the deployment is complete, you will see a deployment report that summarizes the resources created and you will be able to verify the deployment in the Azure portal.
PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/0bf6522c-11d0-4764-9f09-68085ae07ec0>

AZ00011: Authoring Deconstructed Templates

1 Hr 59 Min Remaining

Instructions

Resources

Help

100%

## Conclusion

In this lab, you explored the concept of deconstructed templates and looked at how that is achieved by using output and state sharing. You explored the structure of the nested deployments and enabled your own nested deployments by storing a deployment secret within a key vault, then you rendered deployment of a nested template solution.
PreviousEnd

From <https://labondemand.com/CloudClient/0bf6522c-11d0-4764-9f09-68085ae07ec0>

# AZ00012: Manage, Secure, and Optimize migrated Azure workloads

20 September 2018 09:34

AZ00012: Manage, Secure, and Optimize migrated Azure workloads

1 Hr 30 Min Remaining

Instructions

Resources

Help

100%

## Introduction

In this lab we will look at remotely managing Windows servers in Azure as well as the ones running on-premises; unified security management and advanced threat protection across hybrid cloud workloads; and finally modernizing the application stack with containers.

We will cover the following topics:

```
*Windows Server Admin tool
*Azure Security Center
*Windows server 2016 container
```

Next

From <https://labondemand.com/CloudClient/6f61f08b-73f0-45aa-a4a1-dbb8c2a4d403>

AZ00012: Manage, Secure, and Optimize migrated Azure workloads

1 Hr 30 Min Remaining

Instructions

Resources

Help

100%

## Connect to the hosted environment

In this lab, a hosted environment has been prepared for this lab. Use Remote Desktop to connect.

1. Click here to download the RDP profile needed to connect into the environment. Usually this will download to your **Downloads**folder and be called **jumphost_1440x1000.rdp**.
2. Open the downloaded RDP profile, copy and paste jh7332440.southcentralus.cloudapp.azure.com into the **Computer** field.
   **Note:** If the RDP connection attempts to use your default credentials instead of **LabUser**, click **Show Options** at bottom left and check the box **Always ask for credentials**. This should automatically change the user to **LabUser**. If not, enter **LabUser** and continue to next step.
3. Click **Connect**, on first connect you will need to click the checkbox [ ] Don't ask me again for connections to this computer.
4. For the password, copy and paste De0Ce3@!Dc and click **Connect**.
5. On the next dialog click [ ] Don't ask me again for connection to this computer then click [Yes], this will bypass certificate issues.

PreviousNext: Manage server in azure

From <https://labondemand.com/CloudClient/6f61f08b-73f0-45aa-a4a1-dbb8c2a4d403>

# Manage server in azure

## Remotely manage Windows Server in both on-premises and Azure

The app has been migrated and is running in Azure. Now, let's take a look at how to remotely manage Windows server Azure as well as the ones that are still running on-premises.
**Note:** To walk through these steps we have created a simulation that you can click through. Follow the steps in the guide while navigating the interactive experience.

1. On the Windows desktop, double-click the **Windows Admin Center simulation** shortcut and Maximize the window
2. When prompted, log in by clicking the **Username** field
3. Enter the password by clicking the **Password** field

As you can see here we have a handful of systems listed. Based on the tags applied, you can tell some are still running on-premises, some native to Azure and some are from migrations using Azure Site Recovery.

## Using Tags to view and manage the environment

Here you can see an example Workstation we've created in Azure. The system name starts with **az** suggesting it's an Azure resource, but Windows Admin Center provides a tagging system to allow grouping of systems so you don't have to depend on system names alone.
As a truly hybrid tool, Windows Management Center doesn't split apart systems based on environment thus giving you the ability to view everything in one place. You also have the option of using tags and/or the four built-in categories. This allows you to easily search for your tagged systems when you want to view them outside of the four built-in categories (such as adding **onprem** or cloud-related tags like **azure**). For example, we can add a sample tag and then a search will quickly filter to just that tag.

1. Highlight the row for **azmgmt.corp.net** or click the checkbox next to its name
2. Click **Edit Tags**, the **Edit Connection Tags** window will appear
3. Enter **demotag** into the **Tags to Add** feild by clicking it, then click **+ Add Tags**, then click **Save**
4. In the search field enter **demotag**, Click the search box, The list will now only display those items taged with **demotag**

## Server Management

Managing the system is also simple with just a click of the system name.
Systems just need to be properly configured with appropriate firewall rules; remote management capability; and the primary pre-requisite of having Windows Management Framework 5.1 installed.
Once connected you'll have all the management tools listed on the left for various support

efforts. This list dynamically updates based on the configuration and availability of the remote system. The Overview page provides basic system info and the ability to remotely restart or shut down the system.
All these options provide a modern evolution of in-box management, like you have with Server Manager and MMC, with the added benefit of simplified remote connectivity serving as an excellent complement to the System Center suite.
With tools in place for common system management scenarios, you will also need a hybrid solution for system security. Azure Security Center provides unified security management and advanced threat protection across hybrid cloud workloads.

1. Click-on **azmgmt.corp.net** Server. This will open up an overview of the system and real-time analytics
2. Select **Certificates** and then, select **Expired** in the Certificates window
3. Select **Firewall** and then, select **incoming rules** in the Firewall window
4. Select **Updates** and then, select **update history** in the Updates window
5. Click **Overview** to return to the overview page
   The Windows Server Admin tool consolidates management of systems into 4 common categories: Server Manager, Computer Management, Failover Cluster Management and Hyper-Converged Cluster Management.
6. At the top of the page click the **Server Manager** drop-down and select **Server Manager**
7. At the top of the page click the **Server Manager** drop-down and select **Computer Manager**
8. Click the **Windows Admin Center** button to display all connected systems
9. Click the **X** in the upper right hand corner to close **Windows Admin Center** and end the simulation

PreviousNext

From <https://labondemand.com/CloudClient/6f61f08b-73f0-45aa-a4a1-dbb8c2a4d403>

AZ00012: Manage, Secure, and Optimize migrated Azure workloads

1 Hr 30 Min Remaining

Instructions

Resources

Help

100%

## SecurityCenter

### Start centralized security monitoring of on-premises and Azure sytems

With tools in place for common system management scenarios, you will also need a hybrid solution for system security. Azure Security Center provides unified security management and advanced threat protection across hybrid cloud workloads.
**Note:** To walk through these steps we have created a simulation that you can click through. Follow the steps in the guide while navigating the interactive experience.

1. On the Windows desktop, double-click the **Security Center simulation** shortcut and Maximize the window

2. In the left Favorites menu click **Security Center**

### Steps to On-board a new on-premises system

Security Center also gives you an easy way to view and remediate issues across your hybrid environment. Viewing Endpoint Protection for example provides a break-down of protection on

Azure VM's and on-prem VMs.
You can also easily add more on-prem servers (physical servers or VMs) with instructions from the portal.

1. In the Security Center menu click **Security Solutions**
2. Under **Add data sources** and the **Non-Azure computers**tile, click **Add**
   **Note:** We already have a workspace created called **Migrate-SmartHotelsMigration-180502153603**.
3. Select the **Migrate-SmartHotelsMigration-180502153603**workspace
   Here we have a quick link to download the OMS agent and connectivity issue to add any additional dedicated server or VM. After the agent is installed and configured to connect to the chosen workspace it will begin communicating with Azure and you will see updated recommendations alongside your Azure infrastructure.
4. Close the **Add new non-Azure computers** by clicking the **X**in the upper right

## View Security Health status of Windows Server running On-Premises and Azure using Azure Security Center

Now, let's see all the analytics the Security Center provides to improve the security posture of your resources running in Azure.

1. Under **Resource Security Hygiene**, click **Compute & apps**
2. Click the **VMs and Computers** tile
   Here we can see icons differentiating Azure VMs (represented by the blue computer icon) and on-premises machines (purple server icon). Non-Azure systems from this list are added by installing the Microsoft Management Agent and connecting it to an OMS workspace in the same subscription as Security Center.
   In this list, SHWEBAPP01 and SHWEBAPP02 are two web servers that we migrated to Azure from on-prem. As you can see we are able to continue viewing security status of the on-premises servers we added and maintain that view of systems after we move them into Azure.
3. Click back on the **Overview** tile
4. Click **Endpoint Protection Issues**
   Security Center also gives you an easy way to view and remediate issues across your hybrid environment. Viewing Endpoint Protection for example provides a break-down of protection on Azure VM's and on-prem VMs.
5. Click the **X** in the upper right hand corner to close and end the simulation

PreviousNext: Modernize with containers

From <https://labondemand.com/CloudClient/6f61f08b-73f0-45aa-a4a1-dbb8c2a4d403>

AZ00012: Manage, Secure, and Optimize migrated Azure workloads

1 Hr 30 Min Remaining

Instructions

Resources

Help

100%

# Modernize with containers

Containers are a way to wrap up an application into its own isolated box. For the application in its container, it has no knowledge of any other applications or processes that exist outside of its box. Everything the application depends on to run successfully also lives inside this container. Wherever the box may move, the application will always be satisfied because it is bundled up with everything it needs to run.

Beginning with the release of Windows Server 2016, a certified Docker Engine is included at NO additional cost.
In this scenario we will be migrating the Fabrikamfiber web application to a container on a 2016 Windows server.

## Sign in to the Azure portal

1. On the Windows desktop, click the **Internet Explorer**shortcut to launch **Internet Explorer**
2. **Maximize** the browser window and navigate to **portal.azure.com**
3. In the **Email or phone** field, enter **User1-7332440 @lodsholoutlook.onmicrosoft.com** and click **Next**
4. In the **Password** field, enter **De0Ce3@!Dc** and click **Sign in**
5. In the **Stay signed in?** window, click **No**
6. If a **Welcome to Microsoft Azure** popup window appears, click **Maybe Later** to skip the tour

## Prepare a Docker Server

Azure provides a Windows 2016 image with Containers installed and configured

1. You should be at the main **Dashboard** of the portal
2. In the menu on the left side of the Azure portal, click **Create a resource**
3. In the Search box, type **Windows server docker** and press **Enter** then select **Windows Server 2016 Datacenter - with Containers**
4. On the **Windows Server 2016 Datacenter - with Containers** blade, click **Create**
5. On the **Basics** blade, enter **2016server** under **Name**
   **Note:** If prompted, add numbers or letters to create a unique name
6. Select **Standard SSD** for the **VM disk type**
7. Under **User Name**, enter **demoadmin**
8. Under **Password** and **Confirm password**, enter **Password2017**
9. Under **Subscription** leave the default selection
10. Under **Resource group** change the radio button to **Use existing** and select an entry from the drop-down menu
11. In the **Location** leave the default setting
12. Leave **No** selected under **Already have a Windows Server License?**
13. Click **OK**
14. On the **Choose a size** blade, click **DS1_V2 Standard**
15. Click **Select**
16. On the **Settings** blade under **Select public inbound ports**, select **HTTP, and HTTPS**
17. Leave all other setting as defaults
18. Click **OK**
    **Note:** Due to permissions in this lab you will see a Validation failed error. We've already completed server creation for you
19. Minimize the browser window

## Connect to the Windows 2016 server with containers installed

1. On the Windows desktop, double click the **Remote Desktop**icon
2. In the **Computer** field, enter **2016server**, click connect
3. In the **User Name** filed, enter **labuser**

4. In the **Password** filed, enter **De0Ce3@!Dc** and click **OK** or press **Enter**
5. Click **Yes** to the certificate warning message
6. Minimize or close **Server Manager** after it loads

## Configure Docker

1. On the Windows desktop, right click on the **PowerShell**desktop icon, select run as Administrator
2. Click **Yes** when prompted to allow PowerShell to make changes to your device
3. Navigate to the FabrikamFiber application folder **cd $env:Public\Desktop \FabrikamFiberSupport**

**Note:** The web application files have been pre-copied to the desktop folder **FabrikamFiberSupport**

## Build the docker container

1. From the FabrikamFiber directory type **docker build -t fabrikamfiber .**
   **Note:** we are using the microsoft/asp:latest image. It tags the container with fabrikamfiber name
   **Note:** Download of the microsoft/aspnet Docker image has been done to save time. This can take 20-30 minutes to download and extract
   More information about the microsoft/aspnet Docker image can be found here: https://hub.docker.com/r/microsoft/aspnet/

## Configure the Docker network

1. Remove the existing network, Type **Get-ContainerNetwork | Remove-ContainerNetwork -Force**and hit enter
2. Create the new network, Type **docker network create -d nat --subnet=172.30.0.0/16 ffnet** and hit enter

## Run the Fabrikam Fiber web application container

1. Type **docker run -d --net ffnet --ip 172.30.10.10 --name fabrikamfibersite fabrikamfiber**
2. Wait for the command to complete, takes about a minute.
3. To see that the container is running, type **docker ps**
4. Open a webbroswer and navigate to **http://172.30.10.10**Wait for the website to load

The web application Fabrikam Fiber has now successfully running in a container.
PreviousNext: Conclusion

From <https://labondemand.com/CloudClient/6f61f08b-73f0-45aa-a4a1-dbb8c2a4d403>

AZ00012: Manage, Secure, and Optimize migrated Azure workloads

1 Hr 30 Min Remaining

Instructions

Resources

Help
100%
## Conclusion

You have now completed managing server with Windows Server Admin tool; explored the Azure Security Center and successfully moved an application into a Docker container.
PreviousEnd

From <https://labondemand.com/CloudClient/6f61f08b-73f0-45aa-a4a1-dbb8c2a4d403>

## DT00152: Agile Planning and Portfolio Management with Team Foundation Server 2018

20 September 2018 09:35

# DT00153: Agile Work Item Management with Team Foundation Server 2018

20 September 2018 09:35

# DT00155: Building ASP.NET apps in Azure with SQL Database

20 September 2018 09:34

# DT00161: Embracing Continuous Delivery with Release Management for Team Foundation Server 2018

20 September 2018 09:36

# DT00163: Getting Started with Git using Team Foundation Server 2018

20 September 2018 09:36

# DT00172: Microservice development with ASP.NET Core, Docker, and Azure App Services

20 September 2018 09:37

## OS00130: Intro to PivotCharts, Slicers and Dashboards in Excel

20 September 2018 09:38

# OS00131: Getting Started with Get & Transform in Excel

20 September 2018 09:39

# OS00132: Getting Started with Power Pivot in Excel

20 September 2018 09:39

# OS00134: Connect to the Outlook Calendar with the Microsoft Graph

20 September 2018 09:39

# OS00145: Get notified when data changes through Microsoft Graph Webhooks

20 September 2018 09:40

# OS00147: Synchronize Outlook data with your application

20 September 2018 09:40

## OS00173: Build advanced visualizations and dashboards with Visio Services

20 September 2018 09:41

# OS00174: Introduce yourself to Get & Transform in Excel

20 September 2018 09:45

# OS00175: Get an introduction to analyzing data in Excel

20 September 2018 09:45

# OS00176: Walk through the Excel Data Model

20 September 2018 09:45

# WS00066: Active Directory Deployment and Management Enhancements

20 September 2018 09:47

# WS00124: Identity Services with Active Directory

20 September 2018 09:48

# WS00132: Installing and Managing Nano Server

20 September 2018 09:48

# WS00145: Exploring Virtualization on Windows 10 and Windows Server 2016

20 September 2018 09:48

## WS00146: Building a Storage Infrastructure on Windows Server 2016

20 September 2018 09:49

# WS00149: Implementing a Software Defined Network with Windows Server 2016

20 September 2018 09:49

# WS00153: Enable and secure a remote workforce by joining Windows 10 to Azure Active Directory

20 September 2018 09:49

# WS00154: Migrating your applications and workloads with Azure Migrate and Site Recovery

20 September 2018 09:50