NOTES

# AZ-301T01-A - Designing for Identity and Security

### Welcome to Designing for Identity and Security

Welcome to *Designing for Identity and Security*. This course is part of a series of four courses to help students prepare for Microsoft’s Azure Solutions Architect technical certification exam *AZ-301: Microsoft Azure Architect Design*. These courses are designed for IT professionals and developers with experience and knowledge across various aspects of IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data management, budgeting, and governance.

This course contains the following two modules:

**Module 1** - Managing Security & Identity for Azure Solutions

This module discusses both security and identity within the context of Azure. For security, this module reviews the various options for monitoring security, the options available for securing data and the options for securing application secrets. For identity, this module focuses specifically on Azure Active Directory (Azure AD) and the various features available such as Multi-Factor Authentication (MFA), Managed Service Identity, Azure AD Connect, ADFS and Azure AD B2B/B2C.

**Module 1 Online lab:**

* + This module contains a hands-on lab with a walk-through example of securing Secrets in Azure.

After completing this module, students will be able to:

• Integrate their existing solutions with external identity providers using Azure AD B2B or B2C.

• Design a hybrid identity solution.

• Determine when to use advanced features of Azure AD such as Managed Service Identity, MFA and Privileged Identity Management.

• Secure application secrets using Key Vault.

• Secure application data using SQL Database and Azure Storage features.

**Module 2** - Integrating SaaS Services Available on the Azure Platform

This module introduces multiple SaaS services available in Azure that are available for integration into existing Azure solutions. These services include Cognitive Services, Bot Service, Machine Learning and Media Services.

**Module 2 Online lab:**

* + This module contains a hands-on lab covering the deployment of Service Instances as components of an overall Azure solutions.

After completing this module, students will be able to:

• Detail the various APIs available in Cognitive Services.

• Identify when to use the Face API, Speech API or Language Understanding (LUIS) service.

• Describe the relationship to Bot Framework and Azure Bot Services.

• Create a simple bot using QnA Maker.

• Describe Azure Machine Learning.

• Describe Azure Media Services.

• Discuss Media Services workflows including live streaming, dynamic packaging and static conversion.

• Detail uses of the Computer Vision API.

**Prerequisites**

This course requires that students have the following knowledge and skills:

• Create resources and resource group in Azure.

 • Manage users, groups, and subscriptions in an Azure Active Directory instance.

 • Build an Azure Virtual Machine with related resources.

 • Manage containers and blobs stored in an Azure Storage account.

 • Create App Service Plans and manage apps related to the plan.

 • Configure an Azure Virtual Network and enable S2S and P2S connectivity.

 • Protect networked application components using Network Security Groups.

 • Automate everyday Azure resource tasks using Azure CLI or Azure PowerShell.

 • Deploy an Azure SQL, MySQL, Postgres or Cosmos database instance.

 • Monitor existing Azure solutions using built-in metrics, Application Insights, or Operational Insights.

### Platform Security

In this module you learn how the Azure Platform is built with security in mind, followed by the sharing of responsibility with both Microsoft and you as a customer for security workloads hosted in Azure.

After completing this lesson, you will be able to:

• Understand “shared responsibility” in a cloud model.

• Explain the overall Azure Platform end-to-end security aspects.

• Explain how Azure is offering and handling encryption on different levels.

• Identify how different Azure services and resources are dealing with security, like Azure Networking, Azure Key Vault, Azure SQL, Azure Storage accounts and more.

Azure Security is a “difficult” topic, since it’s the core existence of the platform design and architecture. However, at the same time, a lot of organizations are hesitant from using Public Cloud services like Microsoft Azure, because they don’t think it is providing decent security.  
When an organization starts using Azure, responsibility for securing workloads is shared.

• Microsoft Azure is built with end-to-end security in mind, besides trust. Microsoft gives you a secure foundation, as well as the tooling to control your environment.

• Customers own responsibility of their subscription governance, data, identities, and how to protect those. In IAAS, customer owns more control than in PAAS or SAAS.

Security controls are designed to ensure technology solutions are built and maintained in ways that ensure function and security successfully coexist. This ideal holds strong in Azure where we are constantly vetting and monitoring the implementation of our security controls, as well as watching our service teams continue to innovate new functionality in the cloud environment. With that said, the cloud presents a spectrum of responsibilities based on what types of services and/or features a customer may be consuming. This is unlike more traditional on-premises information systems where most, if not all, security is implemented by the same owner.

Azure is designed for secure multi-tenancy. It’s designed to abstract much of the infrastructure that typically underlies applications (servers, operating systems, Web and database software, and so on) so that customers can focus on building applications—and not on managing resources. The goal is to provide a secure, consistent, scalable set of resources for each customer that they can manage through an Azure subscription. The subscription is associated with a Microsoft account or organizational account.  
   
Technical separation in the Azure datacenter is based on the following components:

• The Azure Fabric Controller (FC) functions as the kernel of the Azure platform, managing resources as needed. The FC provisions, stores, delivers, monitors and commands the VMs and physical servers that make up the Azure customer environment and infrastructure.  
• The Host OS is a configuration-hardened version of Windows Server.

• The Hypervisor is Hyper-V from Windows Server 2012 R2, which has been battle-tested and proven in enterprise environments worldwide.

• The Guest VM OS can be either Windows Server, several distributions of Linux, or an OS image supplied by the customer (much be supported Operating Systems, or starting from the Azure Marketplace images.

From an application & data perspective, Microsoft Azure uses logical isolation to segregate each customer’s data from that of others. This provides the scale and economic benefits of multitenant services while rigorously preventing customers from accessing one another’s data.

**Storage isolation**

* Data is accessible only through claims-based Identity Management & access control with a Storage Access Key (SAK). Shared Access Signature (SAS) tokens can be generated using storage access keys to provide more granular, restricted access. Storage access keys can be reset via the Microsoft Azure Portal or the Storage Management API.
* Storage blocks are hashed by the hypervisor to separate accounts.

**SQL isolation**

* SQL Azure isolates separate account databases.

**Network isolation**

VM switch at the host level blocks inter-tenant communication.

### Securing the Azure Platform

Azure Key Vault helps safeguard cryptographic keys and secrets used by cloud applications and services. By using Key Vault, you can encrypt keys and secrets (such as authentication keys, storage account keys, data encryption keys, .PFX files, and passwords) by using keys that are protected by hardware security modules (HSMs). For added assurance, you can import or generate keys in HSMs. If you choose to do this, Microsoft processes your keys in FIPS 140-2 Level 2 validated HSMs (hardware and firmware).

Key Vault streamlines the key management process and enables you to maintain control of keys that access and encrypt your data. Developers can create keys for development and testing in minutes, and then seamlessly migrate them to production keys. Security administrators can grant (and revoke) permission to keys, as needed.

Anybody with an Azure subscription can create and use key vaults. Although Key Vault benefits developers and security administrators, it could be implemented and managed by an organization’s administrator who manages other Azure services for an organization. For example, this administrator would sign in with an Azure subscription, create a vault for the organization in which to store keys, and then be responsible for operational tasks, such as:

• Create or import a key or secret.

• Revoke or delete a key or secret.

• Authorize users or applications to access the key vault, so they can then manage or use its keys and secrets.

• Configure key usage (for example, sign or encrypt).

• Monitor key usage.

This administrator would then provide developers with URIs to call from their applications, and provide their security administrator with key usage logging information.

For information on securing the Azure platform, visit the Azure Secure Blog: <https://cloudblogs.microsoft.com/microsoftsecure/>

### Azure Active Directory

This section explores Azure Active Directory in the context of advanced identity architectures and solutions.

After completing this lesson, you will be able to:

• Understand Azure Active Directory as a directory service.

• Understanding different topologies to enable Hybrid Identity.

• What is Azure AD Connect, and how to implement and use it.

• Enabling Azure Active Directory Seamless and Single Sign-On.

• What is Azure Active Directory Application Proxy.

• Concepts of Azure Active Directory B2B and B2C.

• Azure Active Directory MFA (multi factor authentication).

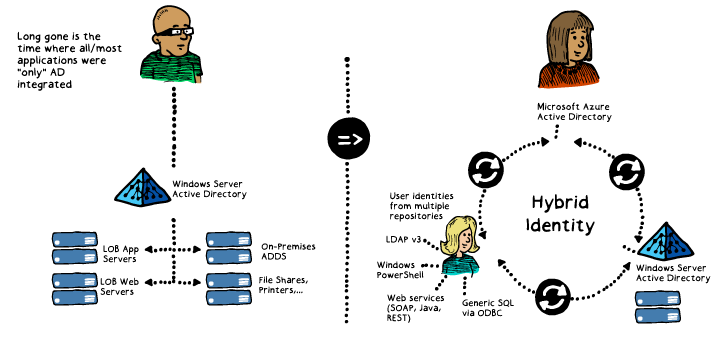
• Using Azure Active Directory for advanced Identity Protection and Privileged Management.

• Use case for Azure Active Directory Domain Services.

**Azure Active Directory**

Azure Active Directory (Azure AD) is Microsoft’s multi-tenant, cloud-based directory and identity management service. Azure AD combines core directory services, advanced identity governance, and application access management.

For IT Admins, Azure AD provides an affordable, easy to use solution to give employees and business partners single sign-on (SSO) access to thousands of cloud SaaS Applications like Office365, Salesforce.com, DropBox, and Concur.



For application developers, Azure AD lets you focus on building your application by making it fast and simple to integrate with a world class identity management solution used by millions of organizations around the world.

Azure AD also includes a full suite of identity management capabilities including multi-factor authentication, device registration, self-service password management, self-service group management, privileged account management, role-based access control, application usage monitoring, rich auditing and security monitoring and alerting. These capabilities can help secure cloud-based applications, streamline IT processes, cut costs and help ensure that corporate compliance goals are met.

Where Active Directory has always been our “trusted” source of identities, long gone is the time where all/most applications were ‘only’ AD integrated. The single user account that could log on to all business applications, as long as they were AD integrated, are not like that anymore in the present world.

Today, a typical business user logs on to 17 applications on average per day, requiring 12 different user accounts/passwords. This is what we call the hybrid identity.

To protect these user accounts from threats, from getting compromised, an end-to-end security model must be put in place.

The “cloud” way of authenticating, is possible using any of the 3 scenarios:

**1. Azure ADConnect using Password Hash Sync**

**2. Azure ADConnect using Federation (ADFS)**

**3. Azure ADConnect using Azure AD Passthrough Authentication Agent**

**Single Sign-On**

Single sign-on, also called identity federation, is a hybrid-based directory integration scenario of Azure Active Directory that you can implement when you want to simplify your user’s ability to seamlessly access cloud services, such as Office 365 or Microsoft Intune, with their existing Active Directory corporate credentials. Without single sign-on, your users would need to maintain separate user names and passwords for your online and on-premises accounts.

An Secure Token Service (STS) enables identity federation, extending the notion of centralized authentication, authorization, and SSO to Web applications and services located virtually anywhere, including perimeter networks, partner networks, and the cloud. When you configure an STS to provide single sign-on access with a Microsoft cloud service, you will be creating a federated trust between your on-premises STS and the federated domain you’ve specified in your Azure AD tenant.

There is a clear benefit to users when you implement single sign-on: it lets them use their corporate credentials to access the cloud service that your company has subscribed to. Users don’t have to sign in again and remember multiple passwords.

### Azure AD Authentication Strategies

Regardless from what authentication mechanism your corporate organization is using, Azure AD Connect is always a required sync tool. Azure AD Connect supports synchronization from multiple Azure AD Forests/Domains, into a single Azure Active Directory environment.

Azure AD Connect can be installed on dedicated VMs, or directly on ADDS Domain Controllers (not recommended, but workable in an SMB environment). The underlying database that is used by AD Connect can be a SQL Server Express, or a full SQL Server 2008 R2 or newer database instance. Azure AD Connect requires an AD Connect Service Account. This account reads/write information from the Azure AD Tenant, as well as requiring an on-premises account in Active Directory, Enterprise Admin level rights, to read/write information back in the on-premises Active Directory.

Azure AD Connect allows for a two-way sync, e.g. password resets (optional – requires P1), account deletions and other strategies for connecting.

### Azure AD B2B & B2C

Both Azure Active Directory (Azure AD) B2B collaboration and Azure AD B2C allow you to work with external users in Azure AD. But how do they compare? Azure AD B2B

**Intended for:**

Organizations that want to be able to authenticate users from a partner organization, regardless of identity provider.

**Identities supported:**

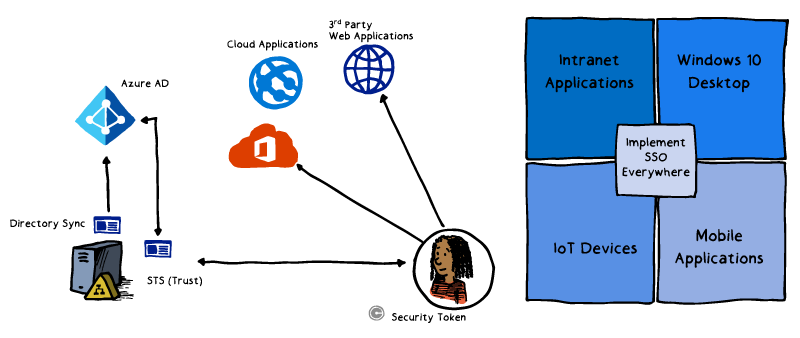
Employees with work or school accounts, partners with work or school accounts, or any email address. Soon to support direct federation.

**Which directory the partner users are in:**

Partner users from the external organization are managed in the same directory as employees, but annotated specially. They can be managed the same way as employees, can be added to the same groups, and so on.

**Single sign-on (SSO):**

Single sign-on to all Azure AD-connected apps is supported. For example, you can provide access to Office 365 or on-premises apps, and to other SaaS apps such as Salesforce or Workday.



**Partner lifecycle:**

Managed by the host/inviting organization.

**Security policy and compliance:**

Managed by the host/inviting organization.

**Branding:**

Host/inviting organization’s brand is used.

#### Azure AD B2C

**Intended for:**

Inviting customers of your mobile and web apps, whether individuals, institutional or organizational customers into your Azure AD.

**Identities supported**:

Consumer users with local application accounts (any email address or user name) or any supported social identity with direct federation.

**Which directory the customer user entities are in:**

In the application directory. Managed separately from the organization’s employee and partner directory (if any).

**Single sign-on (SSO):**

Single sign-on to customer owned apps within the Azure AD B2C tenants is supported. SSO to Office 365 or to other Microsoft and non-Microsoft SaaS apps is not supported.

**Customer lifecycle:**

Self-serve or managed by the application.

**Security policy and compliance:**

Managed by the application.

**Branding:**

Managed by application. Typically tends to be product branded, with the organization fading into the background.

#### Multi-Factor Authentication

Two-step verification is a method of authentication that requires more than one verification method and adds a critical second layer of security to user sign-ins and transactions. It works by requiring any two or more of the following verification methods:

• Something you know (typically a password)

• Something you have (a trusted device that is not easily duplicated, like a phone)

• Something you are (biometrics)

Azure Multi-Factor Authentication (MFA) is Microsoft's two-step verification solution. Azure MFA helps safeguard access to data and applications while meeting user demand for a simple sign-in process. It delivers strong authentication via a range of verification methods, including phone call, text message, or mobile app verification.

### Azure AD Identity Protection

The majority of security breaches take place when attackers gain access to an environment by stealing a user’s identity. Over the years, attackers have become increasingly effective in leveraging third party breaches and using sophisticated phishing attacks. As soon as an attacker gains access to even low privileged user accounts, it is relatively easy for them to gain access to important company resources through lateral movement.

As a consequence of this, you need to:

• Protect all identities regardless of their privilege level.

• Proactively prevent compromised identities from being abused.

Discovering compromised identities is no easy task. Azure Active Directory uses adaptive machine learning algorithms and heuristics to detect anomalies and suspicious incidents that indicate potentially compromised identities. Using this data, Identity Protection generates reports and alerts that enable you to evaluate the detected issues and take appropriate mitigation or remediation actions. Azure Active Directory Identity Protection is a feature of the Azure AD that enables you to:

• Detect potential vulnerabilities affecting your organization’s identities.

• Configure automated responses to detected suspicious actions that are related to your organization’s identities.

• Investigate suspicious incidents and take appropriate action to resolve them.

**Privileged Identity Management (RBAC)**

Securing privileged access is a critical first step to help protect business assets in a modern organization. Privileged accounts are accounts that administer and manage IT systems. Cyber-attackers target these accounts to gain access to an organization’s data and systems. To secure privileged access, you should isolate the accounts and systems from the risk of being exposed to a malicious user.

More users are starting to get privileged access through cloud services. This can include global administrators of Office365, Azure subscription administrators, and users who have administrative access in VMs or on SaaS apps.

Azure AD Privileged Identity Management helps to mitigate the risk of excessive, unnecessary or misused access rights. Azure AD Privileged Identity Management helps your organization:

• See which users are assigned privileged roles to manage Azure resources (Preview), as well as which users are assigned administrative roles in Azure AD.

• Enable on-demand, "just in time" administrative access to Microsoft Online Services like Office 365 and Intune, and to Azure resources (Preview) of subscriptions, resource groups, and individual resources such as Virtual Machines.

• See a history of administrator activation, including what changes administrators made to Azure resources (Preview).

• Get alerts about changes in administrator assignments.  
• Require approval to activate Azure AD privileged admin roles (Preview).

• Review membership of administrative roles and require users to provide a justification for continued membership.

In Azure AD, Azure AD Privileged Identity Management can manage the users assigned to the built-in Azure AD organizational roles, such as Global Administrator. In Azure, Azure AD Privileged Identity Management can manage the users and groups assigned via Azure RBAC roles, including Owner or Contributor.

### Azure AD Domain Services

Azure Infrastructure Services enable you to deploy a wide range of computing solutions in an agile manner. With Azure Virtual Machines, you can deploy nearly instantaneously and you pay only by the minute. Using support for Windows, Linux, SQL Server, Oracle, IBM, SAP, and BizTalk, you can deploy any workload, any language, on nearly any operating system. These benefits enable you to migrate legacy applications deployed on-premises to Azure, to save on operational expenses.

A key aspect of migrating on-premises applications to Azure is handling the identity needs of these applications. Directory-aware applications may rely on LDAP for read or write access to the corporate directory or rely on Windows Integrated Authentication (Kerberos or NTLM authentication) to authenticate end users. Line-of-business (LOB) applications running on Windows Server are typically deployed on domain joined machines, so they can be managed securely using Group Policy. To 'lift-and-shift' on-premises applications to the cloud, these dependencies on the corporate identity infrastructure need to be resolved.

Administrators often turn to one of the following solutions to satisfy the identity needs of their applications deployed in Azure:

• Deploy a site-to-site VPN connection between workloads running in Azure Infrastructure Services and the corporate directory on-premises.

• Extend the corporate AD domain/forest infrastructure by setting up replica domain controllers using Azure virtual machines.

• Deploy a stand-alone domain in Azure using domain controllers deployed as Azure virtual machines.

All these approaches suffer from high cost and administrative overhead. Administrators are required to deploy domain controllers using virtual machines in Azure. Additionally, they need to manage, secure, patch, monitor, backup, and troubleshoot these virtual machines. The reliance on VPN connections to the on-premises directory causes workloads deployed in Azure to be vulnerable to transient network glitches or outages. These network outages in turn result in lower uptime and reduced reliability for these applications.

Azure AD Domain Services provides managed domain services such as domain join, group policy, LDAP, Kerberos/NTLM authentication that are fully compatible with Windows Server Active Directory. You can consume these domain services without the need for you to deploy, manage, and patch domain controllers in the cloud. Azure AD Domain Services integrates with your existing Azure AD tenant, thus making it possible for users to log in using their corporate credentials. Additionally, you can use existing groups and user accounts to secure access to resources, thus ensuring a smoother 'lift-and-shift' of on-premises resources to Azure Infrastructure Services.

Azure AD Domain Services functionality works seamlessly regardless of whether your Azure AD tenant is cloud-only or synced with your on-premises Active Directory.

### Lab Steps

#### Online Lab: Securing Secrets in Azure

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>.

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy Key Vault resources

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Deploy a key vault

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Key Vault** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Key Vault**.
4. On the **Key Vault** blade, click the **Create** button.
5. On the **Create key vault** blade, perform the following tasks:
   * In the **Name** text box, type a globally unique value.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0901-RG**.
   * In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
   * Click **Pricing tier**, on the **Pricing tier** blade, click **A1 Standard**, and then click **Select**.
   * Leave all remaining settings with their default values.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next task.

##### Task 3: Add a secret to a key vault by using the Azure portal

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0901-RG**.
3. On the **AADesignLab0901-RG** blade, click the entry representing the newly created key vault.
4. On the key vault blade, click **Secrets**.
5. On the key vault secrets blade, click the **Generate/Import** button at the top of the pane.
6. On the **Create a secret** blade, perform the following tasks:
   * In the **Upload options** drop-down list, ensure that the **Manual** entry is selected.
   * In the **Name** text-box, type **thirdPartyKey**.
   * In the **Value** text box, enter the value **56d95961e597ed0f04b76e58**.
   * Leave all remaining settings with their default values.
   * Click the **Create** button.

##### Task 4: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 5: Add a secret to a key vault using the CLI

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the Azure key vault you deployed earlier in this exercise:

RESOURCE\_GROUP='AADesignLab0901-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the name of the Azure key vault you created earlier in this exercise:

KEY\_VAULT\_NAME=$(az keyvault list --resource-group $RESOURCE\_GROUP --query "[0].name" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command, and press **Enter** to list secrets in the key vault:

az keyvault secret list --vault-name $KEY\_VAULT\_NAME

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to display the value of the **thirdPartyKey** secret:

az keyvault secret show --vault-name $KEY\_VAULT\_NAME --name thirdPartyKey --query value --output tsv

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to add a new secret to your key vault:

az keyvault secret set --vault-name $KEY\_VAULT\_NAME --name firstPartyKey --value 56f8a55119845511c81de488

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list secrets in the key vault:

az keyvault secret list --vault-name $KEY\_VAULT\_NAME --query "[\*].{Id:id,Created:attributes.created}" --out table

1. Close the **Cloud Shell** pane.

##### Task 6: Add secrets to a key vault by using Azure Resource Manager templates

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template Deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T01\Module\_01\LabFiles\Starter\** folder, select the **secret-template.json** file, and click **Open**. This will load the following content into the template editor pane:

{ "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": { "vaultName": { "type": "string" } }, "variables": { "secretName": "vmPassword" }, "resources": [ { "apiVersion": "2016-10-01", "type": "Microsoft.KeyVault/vaults/secrets", "name": "[concat(parameters('vaultName'), '/', variables('secretName'))]", "properties": { "contentType": "text/plain", "value": "StudentPa$$w.rd" } } ] }

1. Click the **Save** button to persist the template.
2. Back on the **Custom deployment** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
   * In the **Vault Name** text box, type the name of the key vault you created earlier in this exercise.
   * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   * Click the **Purchase** button.
3. Do not wait for the deployment to complete but proceed to the next step.
4. In the upper left corner of the Azure portal, click **Create a resource**.
5. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
6. On the **Everything** blade, in the search results, click **Template Deployment**.
7. On the **Template deployment** blade, click the **Create** button.
8. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
9. On the **Edit template** blade, click **Load file**.
10. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T01\Module\_01\LabFiles\Starter\** folder, select the **storage-template.json** file, and click **Open**. This will load the following content into the template editor pane:

{ "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": { "vaultName": { "type": "string" } }, "variables": { "secretName": "storageConnectionString", "storageName": "[concat('stor', uniqueString(resourceGroup().id))]" }, "resources": [ { "apiVersion": "2017-10-01", "type": "Microsoft.Storage/storageAccounts", "name": "[variables('storageName')]", "location": "[resourceGroup().location]", "kind": "Storage", "sku": { "name": "Standard\_LRS" }, "properties": { } }, { "apiVersion": "2016-10-01", "type": "Microsoft.KeyVault/vaults/secrets", "name": "[concat(parameters('vaultName'), '/', variables('secretName'))]", "dependsOn": [ "[resourceId('Microsoft.Storage/storageAccounts', variables('storageName'))]" ], "properties": { "contentType": "text/plain", "value": "[concat('DefaultEndpointsProtocol=https;AccountName=', variables('storageName'), ';', 'AccountKey=', listKeys(resourceId('Microsoft.Storage/storageAccounts', variables('storageName')), providers('Microsoft.Storage', 'storageAccounts').apiVersions[0]).keys[0].value, ';')]" } } ] }

1. Click the **Save** button to persist the template.
2. Back on the **Custom deployment** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
   * In the **Vault Name** field, type the name of the key vault you created earlier in this exercise.
   * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   * Click the **Purchase** button.
3. Wait for the deployment to complete before you proceed to the next task.

##### Task 7: View key vault secrets

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0901-RG**.
3. On the **AADesignLab0901-RG** blade, click the entry representing the key vault you created earlier in this exercise.
4. On the key vault blade, click **Secrets**.
5. On the key vault secrets blade, review the list of secrets created during this lab.
6. Click the entry representing the **vmPassword** secret.
7. On the **vmPassword** blade, click the entry representing the current version of the secret.
8. On the Secret Version blade, click the **Show secret value** button.
9. Verify that the value of the secret matches the one included in the template you deployed in the previous task.

**Review**: In this exercise, you created a **Key Vault** instance and used several different methods to add secrets to the key vault.

#### Exercise 2: Deploy Azure VM using Key Vault secret

##### Task 1: Retrive the value of the key vault Resource Id parameter

1. At the top of the portal, click the **Cloud Shell** icon to open a new Clould Shell instance.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub virtual network:

RESOURCE\_GROUP='AADesignLab0901-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the resource id of the Azure key vault you created earlier in this exercise:

KEY\_VAULT\_ID=$(az keyvault list --resource-group $RESOURCE\_GROUP --query "[0].id" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the Azure key vault resource id and which takes into account any special character the resource id might include:

KEY\_VAULT\_ID\_REGEX="$(echo $KEY\_VAULT\_ID | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"

##### Task 2: Prepare the Azure Resource Manager deployment template and parameters files

1. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
2. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T01\Module\_01\LabFiles\Starter\** folder, select the **vm-template.json** file, and click **Open**.
3. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
4. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T01\Module\_01\LabFiles\Starter\** folder, select the **vm-template.parameters.json** file, and click **Open**.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **$KEY\_VAULT\_ID** parameter in the **vm-template.parameters.json** parameters file with the value of the **$KEY\_VAULT\_ID** variable:

sed -i.bak1 's/"$KEY\_VAULT\_ID"/"'"$KEY\_VAULT\_ID\_REGEX"'"/' ~/vm-template.parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the placeholder was successfully replaced in the parameters file:

cat ~/vm-template.parameters.json

##### Task 3: Configure a key vault for deployment of Azure Resource Manager templates

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0901-RG**.
3. On the **AADesignLab0901-RG** blade, click the entry representing the key vault you created in the previous exercise.
4. On the key vault blade, click **Access policies**.
5. On the **Access policies** blade, click the **Click to show advanced access policies** link.
6. Select the **Enable access to Azure Resource Manager for template deployment** checkbox.
7. Click the **Save** button at the top of the pane.

##### Task 4: Deploy a Linux VM with the password paramter set by using a key vault secret.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the Azure Resource Manager template with the specified parameters file:

az group deployment create --resource-group $RESOURCE\_GROUP --template-file ~/vm-template.json --parameters @~/vm-template.parameters.json

1. Wait for the deployment to complete before you proceed to the next task.

##### Task 5: Verify the outcome of the deployment

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the newly deployed Azure VM:

RESOURCE\_GROUP='AADesignLab0901-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the name of the Azure key vault containing the secret that stores the value of the password of the local Administrator account:

KEY\_VAULT\_NAME=$(az keyvault list --resource-group $RESOURCE\_GROUP --query "[0].name" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the value of the secret:

az keyvault secret show --vault-name $KEY\_VAULT\_NAME --name vmPassword --query value --output tsv

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the public IP address of the Azure VM you deployed in the previous task:

PUBLIC\_IP=$(az network public-ip list --resource-group $RESOURCE\_GROUP --query "[0].ipAddress" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to connect to the Azure VM via SSH:

ssh Student@$PUBLIC\_IP

1. At the **Cloud Shell** command prompt, when prompted whether you want to continue connecting, type yes and press **Enter**.
2. At the **Cloud Shell** command prompt, when prompted for password, type the value of the secret you retrieved earlier in this task and press **Enter**.
3. Verify that you successfully authenticated.
4. At the **Cloud Shell** command prompt, type exit to log out from the Azure VM.

**Review**: In this exercise, you deployed a Linux VM using a password stored as a key vault secret.

#### Exercise 3: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab09')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab09')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 1 Review Questions

**Shared Access Signatures**  
You are designing a solution for your company. The solution will include an Azure web app.  
The web app must connect to Azure Blob storage.  You need to control access to the storage objects.  
What should you recommend?

Suggested Answer ↓

A shared access signature (SAS) provides you with a way to grant limited access to objects in your storage account to other clients (such as a web app), without exposing your account key. A shared access signature provides delegated access to the Azure blob in the storage account. With a SAS, you can grant clients access to resources in your storage account, without sharing your account keys. This is the key point of using shared access signatures in your applications as a SAS is a secure way to share your storage resources without compromising your account keys.

**Azure Active Directory**  
You are designing a solution for your company. The solution will include an Azure web app.  
Users must be able to sign in to the web app by using their Facebook username and password.  
What should you recommend and why?

Suggested Answer ↓

Azure Active Directory (Azure AD) includes Business-to-Consumer (B2C) features that can be integrated into applications. Azure AD B2C implements OpenID Connect, which supports many different providers, including Twitter, Google, and many others that support the standard. Azure AD B2C protects from denial-of-service and password attacks against your applications and includes user interface customizations to easily integrate into your existing applications.

**AD Connect**  
Your company has an on-premises Active Directory Domain Services infrastructure.  
You need to design a solution that allows the users stored in your on-premises Active Directory to access applications in Azure.   
What should you recommend?

Suggested Answer ↓

Azure AD Connect allows you to integrate your on-premises directories with Azure Active Directory. This allows you to provide a common identity for your users for Office 365, Azure, SaaS and custom applications integrated with Azure AD. With Azure AD Connect installed in a on-premises server and syncing Active Directory identities, you are providing a common identity for accessing both cloud and on-premises resources. Azure AD Connect includes components to monitor the synchronization health and federation of identities.

### Bot Services

This section introduces Bot Services by first detailing its underlying framework, Bot Framework, and then using QnA Maker to create an example bot.

After completing this section you will be able to:

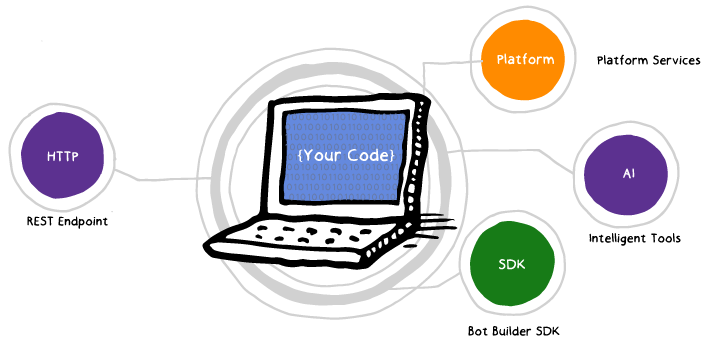
• Describe the relationship to Bot Framework and Azure Bot Services.

• Create a simple bot using QnA Maker.

**Bot Services**

Bot Service provides an integrated environment that is purpose-built for bot development, enabling you to build, connect, test, deploy, and manage intelligent bots, all from one place. Bot Service leverages the Bot Builder SDK with support for .NET and Node.js. You can write a bot, connect, test, deploy, and manage it from your web browser with no separate editor or source control required. For simple bots, you may not need to write code at all. Bot Service accelerates bot development with Five bot templates you can choose from when you create a bot. You can further modify your bot directly in the browser using the Azure editor or in an Integrated Development Environment (IDE), such as Visual Studio and Visual Studio Code.

**Bot Framework**



Below are a few of the key features of Bot Service:

• Multiple language support: Bot Service leverages Bot Builder with support for .NET and Node.js.

• Bot templates: Bot Service templates allow you to quickly create a bot with the code and features you need. Choose from a Basic bot, a Forms bot for collecting user input, a Language understanding bot that leverages LUIS to understand user intent, a QnA bot to handle FAQs, or a Proactive bot that alerts users of events.

• Bring your own dependencies: Bots support NuGet and NPM, so you can use your favorite packages in your bot.

• Flexible development: Code your bot right in the Azure portal or set up continuous integration and deploy your bot through GitHub, Visual Studio Team Services, and other supported development tools. You can also publish from Visual Studio.

• Connect to channels: Bot Service supports several popular channels for connecting your bots and the people that use them. Users can start conversations with your bot on any channel that you've configured your bot to work with, including S**kyp**e, Facebook, Teams, Slack, SMS, and several others.

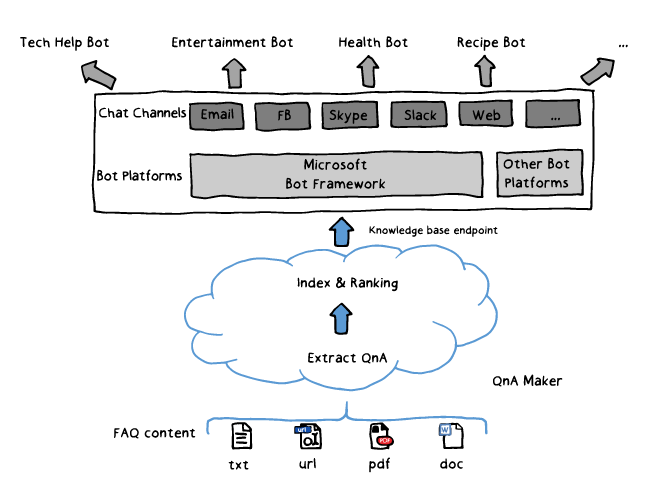
• Tools and services: Test your bot with the Bot Framework Emulator and preview your bot on different channels with the Channel Inspector.

• Open source: The Bot Builder SDK is open-source and available on GitHub.

### QnA Maker

Microsoft QnA Maker is a REST API and web-based service that trains AI to respond to user's questions in a more natural, conversational way. QnA Maker provides a graphical user interface that allows non-developers to train, manage, and use the service for a wide range of solutions.  
QnA Maker extracts a knowledge base from two types of input: FAQ pages and product manuals. The tool supports extraction from FAQ web pages or documents in the question-answer format. The tool can also extra QnA pairs from PDF-format product manuals.

Once extracted, the QnA Maker service creates a knowledge base and bot using the knowledge base. The bot can then be used, via a REST API, in any existing web application or website to answer questions for users. Over time, the knowledge base can be updated, retrained, and republished to meet the morphing needs to a user-facing web application.

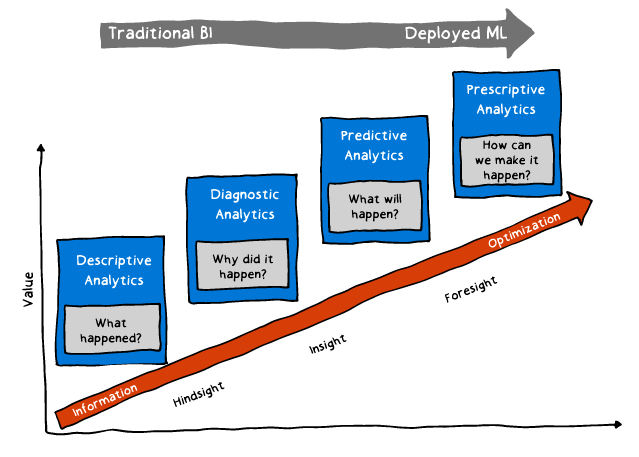


### Machine Learning

This section briefly introduces Azure Machine Learning in the context of traditional BI and prescriptive analytics.

Machine learning is a data science technique that allows computers to use existing data to forecast future behaviors, outcomes, and trends. Using machine learning, computers learn without being explicitly programmed.

Forecasts or predictions from machine learning can make apps and devices smarter. When you shop online, machine learning helps recommend other products you might like based on what you've purchased. When your credit card is swiped, machine learning compares the transaction to a database of transactions and helps detect fraud. When your robot vacuum cleaner vacuums a room, machine learning helps it decide whether the job is done.

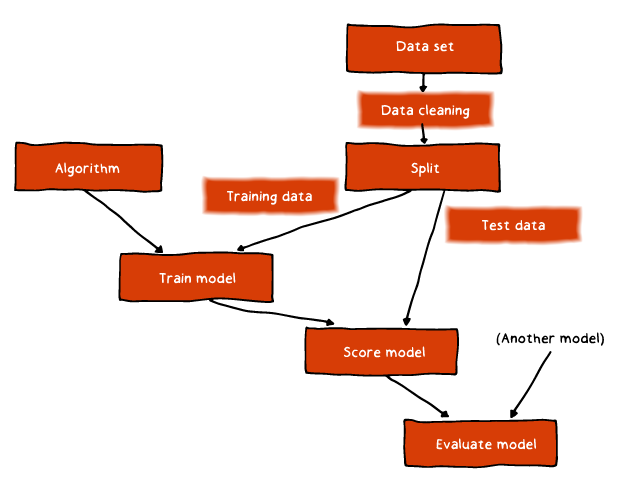


### Azure Machine Learning

Azure Machine learning is an end-to-end data science and analytics solution that’s integrated into Azure. It allows users to develop experiments as well as deploy data and models via the cloud. Its composed of the Azure Machine Learning Workbench, Experimentation service, Model Management Service, Libraries for Apache Spark, and the Visual Studio Code Tools for AI.

Azure Machine learning fully support various open source technologies, such as scikit-learn, TensorFlow, and more. A massive library of open source Python packages is accessible. Also, you can execute experiments in managed environments such as spark clusters or docker containers. Azure Machine Learning built on top of open source technologies. These technologies are Jupyter Notebook, Conda, Python, Docker, Apache Spark, and Kubernetes. It also includes open source technologies from Microsoft itself, such as Cognitive Toolkit.

Azure Machine Learning Workbench is a desktop application supported on both windows and macOS that includes command-line tools. It allows users to help manage learning solutions via data ingestion and preparation, model development, experiment management, and even model deployment in different target environments.  
   
The Azure Machine Learning Experimentation Service helps handle the implementation of machine learning experiments. It also provides project management, roaming, sharing, and git integration to support the Workbench. Azure Machine Learning Experimentation Services allows the implementation of services across a range of environment options such as Local native, Local Docker container, or Scale out Spark cluster in Azure. The Experimentation Service also creates Virtual environments for scripts to provide an isolated space with reproducible results. It documents run history information and visually displays the information so you can quickly select the best model from your experiments.



Azure Machine Learning Model Management Service provides users the ability to deploy predictive models into a range of environments. Information on models, such as the version and lineage, is notated from training runs throughout the deployment. The models themselves are registered, managed, and stored in the cloud.

The Microsoft Machine Learning Library for Apache Spark or MMLSpark is an open-source Spark Package providing data science and Deep Learning tools for Apache Spark. MMLSpark allows users to create robust, analytical, and highly scalable predictive models for large image and text datasets.

Visual Studio Code Tools for AI itself is an extension used with Visual Studio code that allows you to test, build, and deploy AI and Deep Learning solutions. It contains various integration points from Azure Machine learning. Such examples include visualization of run history that displays the performance of training runs, a gallery view allowing you to bootstrap and browse new projects within the Microsoft Cognitive Toolkit and other deep-learning frameworks, as well as an explorer view, to allow users to select targets for your scripts to execute.

Media Services

This section introduces the Cognitive Services Computer Vision API and Azure Media Services as components that can assist with the processing of still-image and video media.

After completing this section you will be able to:

• Describe Azure Media Services.

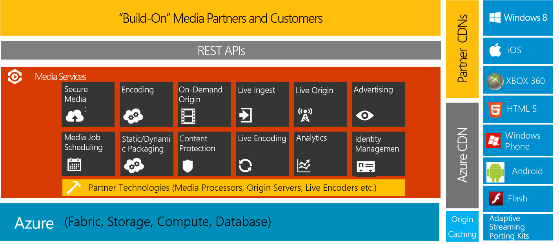
• Discuss Media Services workflows including live streaming, dynamic packaging and static conversion.

• Detail uses of the Computer Vision API.

**Media Services**

Microsoft Azure Media Services is an extensible cloud-based platform that enables developers to build scalable media management and delivery applications. Media Services is based on REST APIs that enable you to securely upload, store, encode, and package video or audio content for both on-demand and live streaming delivery to various clients (for example, TV, PC, and mobile devices).

You can build end-to-end workflows using entirely Media Services. You can also choose to use third-party components for some parts of your workflow. For example, encode using a third-party encoder. Then, upload, protect, package, deliver using Media Services. You can choose to stream your content live or deliver content on-demand.



• Crop photos to be used as thumbnails.

**Image Tagging**

Computer Vision API returns tags based on more than 2000 recognizable objects, living beings, scenery, and actions. When tags are ambiguous or not common knowledge, the API response provides “hints” to clarify the meaning of the tag in context of a known setting. Tags are not organized as a taxonomy and no inheritance hierarchies exist. A collection of content tags forms the foundation for an image 'description' displayed as human readable language formatted in complete sentences.

After uploading an image or specifying an image URL, Computer Vision API's algorithms output tags based on the objects, living beings, and actions identified in the image. Tagging is not limited to the main subject, such as a person in the foreground, but also includes the setting (indoor or outdoor), furniture, tools, plants, animals, accessories, gadgets, etc.

**Description Generation**

Computer Vision API's algorithms analyze the content in an image. This analysis forms the foundation for a 'description' displayed as human-readable language in complete sentences. The description summarizes what is found in the image. Computer Vision API's algorithms generate various descriptions based on the objects identified in the image. The descriptions are each evaluated and a confidence score generated. A list is then returned ordered from highest confidence score to lowest.

**Color Schemes**

The Computer Vision algorithm extracts colors from an image. The colors are analyzed in three different contexts: foreground, background, and whole. They are grouped into twelve 12 dominant accent colors. Those accent colors are black, blue, brown, gray, green, orange, pink, purple, red, teal, white, and yellow. Depending on the colors in an image, simple black and white or accent colors may be returned in hexadecimal color codes.  
The Computer Vision API can also determine color extracted from an image designed to represent the most eye-popping color to users via a mix of dominant colors and saturation and identify this color as the Accent Color.

**Optical Character Recognition (OCR)**

OCR technology detects text content in an image and extracts the identified text into a machine-readable character stream. You can use the result for search and numerous other purposes like medical records, security, and banking. It automatically detects the language. OCR saves time and provides convenience for users by allowing them to take photos of text instead of transcribing the text. If needed, OCR corrects the rotation of the recognized text, in degrees, around the horizontal image axis.

### Cognitive Services

This module introduces multiple SaaS services available in Azure that are available for integration into existing Azure solutions. These services include Cognitive Services, Bot Service, Machine Learning and Media Services.

After completing this module, students will be able to:

• Identify when Cognitive Services, Bot Service or Machine Learning is appropriate for their solution.

• Compare the various features available in Media Services and determine the appropriate features for their solution.

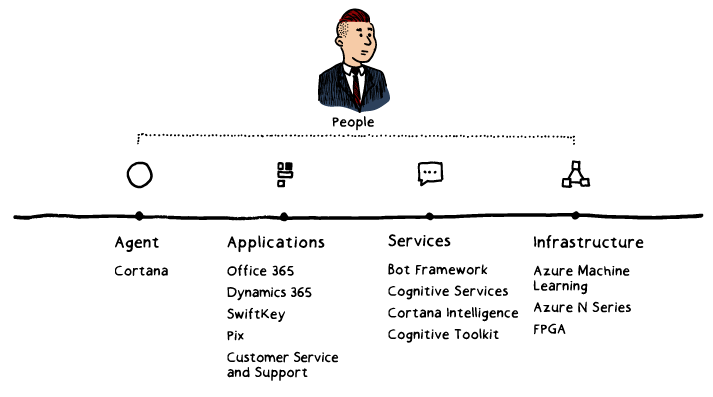
This section introduces the various APIs available in Azure Cognitive Services and specifically reviews Language Understanding LUIS, Face API and Speech API in detailed examples. After completing this section you will be able to:

• Detail the various APIs available in Cognitive Services.

• Identify when to use the Face API, Speech API or Language Understanding (LUIS) service.

**Cognitive Services**

Microsoft Cognitive Services are a set of APIs, SDKs and services available to developers to make their applications more intelligent, engaging and discoverable. Microsoft Cognitive Services expands on Microsoft’s evolving portfolio of machine learning APIs and enables developers to easily add intelligent features – such as emotion and video detection; facial, speech and vision recognition; and speech and language understanding – into their applications.



**Bing APIs**

Cognitive Services, as a suite, also includes various Bing APIs that can be used in your applications:

**• Bing Web Search:** Bing Web Search API provides an experience similar to Bing.com/search by returning search results that Bing determines are relevant to a user's query. The results include Web pages and may also include images, videos, and more.

**• Bing Image Search:** Bing Image Search API provides an experience similar to Bing.com/images by returning images that Bing determines are relevant to a user's query.

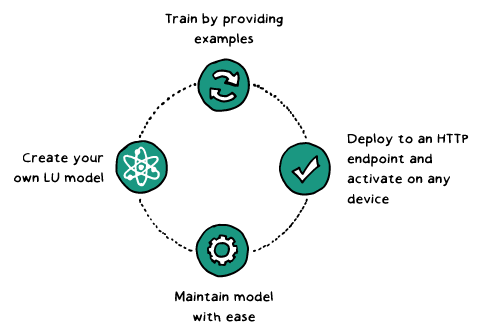
**• Bing Autosuggest:** Bing Autosuggest API lets you send a partial search query term to Bing and get back a list of suggested queries that other users have searched on. For example, as the user enters each character of their search term, you'd call this API and populate the search box's drop-down list with the suggested query strings.

### Intent Detection

Language Understanding (LUIS) allows your application to understand what a person wants in their own words. LUIS uses machine learning to allow developers to build applications that can receive user input in natural language and extract meaning from it. A client application that converses with the user can pass user input to a LUIS app and receive relevant, detailed information back.

A LUIS app is a domain-specific language model designed by you and tailored to your needs. You can start with a prebuilt domain model, build your own, or blend pieces of a prebuilt domain with your own custom information.

A model starts with a list of general user intentions such as “Book Flight” or “Contact Help Desk.” Once the intentions are identified, you supply example phrases called utterances for the intents. Then you label the utterances with any specific details you want LUIS to pull out of the utterance.



Prebuilt domain models include all these pieces for you and are a great way to start using LUIS quickly.

After the model is designed, trained, and published, it is ready to receive and process utterances. The LUIS app receives the utterance as an HTTP request and responds with extracted user intentions. Your client application sends the utterance and receives LUIS's evaluation as a JSON object. Your client app can then take appropriate action.

**Key LUIS Concepts**

• **Intents**: An intent represents actions the user wants to perform. The intent is a purpose or goal expressed in a user's input, such as booking a flight, paying a bill, or finding a news article. You define and name intents that correspond to these actions. A travel app may define an intent named “BookFlight.”

• **Utterances**: An utterance is text input from the user that your app needs to understand. It may be a sentence, like “Book a ticket to Paris”, or a fragment of a sentence, like “Booking” or “Paris flight.” Utterances aren't always well-formed, and there can be many utterance variations for a particular intent.

• **Entities**: An entity represents detailed information that is relevant in the utterance. For example, in the utterance “Book a ticket to Paris.” “Paris” is a location. By recognizing and labeling the entities that are mentioned in the user’s utterance, LUIS helps you choose the specific action to take to answer a user's request.

### Cognitive APIs

Cognitive Services, as a suite, includes a wide variety of APIs. Some example APIs include the following APIs listed below.

**Text Analytics API**

Text Analytics API is a cloud-based service that provides advanced natural language processing over raw text, and includes three main functions: sentiment analysis, key phrase extraction, and language detection.

**Speaker Recognition API**

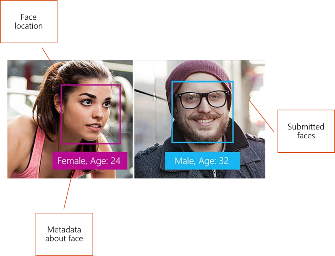
Speaker Recognition API is a cloud-based APIs that provide the most advanced algorithms for speaker verification and speaker identification.

**Content Moderator API**

Content Moderator API tracks, flags, assesses, and filters out offensive and unwanted content that creates risk for applications.

**Face API**

Face API is a cloud-based service that provides the advanced face algorithms with two main functions: face detection with attributes and face recognition. Face API detects up to 64 human faces with high precision face location in an image. And the image can be specified by file in bytes or valid URL. The API returns a face rectangle (left, top, width and height) indicating the face location in the image is returned along with each detected face. Optionally, face detection extracts a series of face related attributes such as pose, gender, age, head pose, facial hair and glasses.  
Face recognition is widely used in many scenarios including security, natural user interface, image content analysis and management, mobile apps, and robotics. Four face recognition functions are provided: face verification, finding similar faces, face grouping, and person identification.



### Lab Steps

#### Online Lab: Deploying Service Instances as Components of Overall Azure Solutions

NOTE: For the most recent version of this online lab, see: [https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign.](https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign)

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy Function App and Cognitive Service using ARM Template

##### Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Deploy Cognitive Service using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template Deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T01\Module\_02\LabFiles\Starter\** folder, select the **cognitive-template.json** file, and click **Open**. This will load the following content into the template editor pane:

{ "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "variables": { "serviceName": "[concat('cgnt', uniqueString(resourceGroup().id))]" }, "resources": [ { "apiVersion": "2017-04-18", "type": "Microsoft.CognitiveServices/accounts", "name": "[variables('serviceName')]", "kind": "TextAnalytics", "location": "[resourceGroup().location]", "sku": { "name": "S1" }, "properties": {} } ], "outputs": { "cognitiveEndpointUrl": { "type": "string", "value": "[reference(variables('serviceName')).endpoint]" }, "cognitiveEndpointKey": { "type": "string", "value": "[listKeys(variables('serviceName'), '2017-04-18').key1]" } } }

1. Click the **Save** button to persist the template.
2. Back on the **Custom deployment** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab1001-RG**.
   * In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
   * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   * Click the **Purchase** button.
3. Wait for the deployment to complete before you proceed to the next step.
4. In the hub menu of the Azure portal, click **Resource groups**.
5. On the **Resource groups** blade, click **AADesignLab1001-RG**.
6. On the **AADesignLab1001-RG** blade, locate the **Deployments** header at the top of the blade and click the below the **Deployments** label, which indicates the number of successful deployments.
7. On the deployments blade, click the name of the most recent deployment.
8. On the **Microsoft.Template- Overview** blade, click **Outputs**.
9. On the **Microsoft.Template - Outputs** blade, identify the values of **COGNITIVEENDPOINTURL** and **COGNITIVEENDPOINTKEY** outputs. Record these values, since you will need them later in the lab.

##### Task 3: Deploy a function app

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Function App** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Function App**.
4. On the **Function App** blade, click the **Create** button.
5. On the next **Function App** blade, perform the following tasks:
   * In the **App name** text box, type a globally unique name.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
   * In the **OS** section, ensure that the **Windows** button is selected.
   * In the **Hosting Plan** drop-down list, ensure that the **Consumption Plan** entry is selected.
   * In the **Runtime Stack** drop-down list, ensure that **.NET** entry is selected.
   * In the **Location** drop-down list, select the Azure region to which you deployed an instance of Cognitive Service in the previous task.
   * In the **Storage** section, ensure that the **Create new** option is selected and accept the default value of the Storage Account name.
   * In the **Application Insights** section, set the extension to **Disabled**.
   * Click the **Create** button.
6. Wait for the provisioning of the function app to complete before you proceed to the next step.
7. In the hub menu of the Azure portal, click **Resource groups**.
8. On the **Resource groups** blade, click **AADesignLab1001-RG**.
9. On the **AADesignLab1001-RG** blade, in the list of resources, click the newly provisioned function app.
10. On the function app blade, click the **Platform features** tab at the top of the blade.
11. On the **Platform features** tab, click the **Application Settings** link in the **GENERAL SETTINGS** section.
12. On the **Application settings** tab, locate the **Application Settings** section. Click the **Add new setting** link and perform the following tasks:
    * In the **Enter a name** text box, type **EndpointUrl**.
    * In the **Enter a value** text box, enter the value of **COGNITIVEENDPOINTURL** you identified earlier.
13. In the **Application Settings** section, click the **Add new setting** link again and pferform the following tasks:
    * In the **Enter a name** text box, type **EndpointKey**.
    * In the **Enter a value** text box, type the value of **COGNITIVEENDPOINTKEY** you identified earlier.
14. Click the **Save** button at the top of the **Application settings** tab.
15. Back on the function app blade, click the **Platform features** tab at the top of the blade.
16. In the **Platform features** tab, click the **Deployment Center** link in the **Code Deployment** section.
17. On the **Deployment Center** blade that appears, click the **External** button and then click **Continue**.
18. Click **App Service Kudu build server** and click **Continue**.
19. Once the **Code** section is displayed, perform the following tasks
    * In the **Repository URL** text box, type [**https://github.com/azure-labs/cognitive-services-function**](https://github.com/azure-labs/cognitive-services-function).
    * In the **Branch** text box, type **master**.
    * In the **Repository Type** section, ensure that the **Git** option is selected.
    * Click the **Continue** button.
20. Click **Finish** and wait for the deployment to complete before you proceed to the next task.**Note**: You will be able to determine that the first deployment has completed by monitoring the **Deployments** tab. This tab updates automatically.

##### Task 4: Test a function app using Cognitive Services

1. Back on the function app blade, click **Functions** to expand the list of functions.**Note**: You may need to click **Functions** twice to refresh the list of functions.
2. Select the **DetermineLanguage** function from the list of functions.
3. In the **run.csx** pane that opens, click **Test** on the right side of the pane.
4. In the **Test** pane, perform the following tasks:
   * In the **Request body** text box, type the following:

{ "text": "I stuffed a shirt or two into my old carpet-bag, tucked it under my arm, and started for Cape Horn and the Pacific." }

* + Click the **Run** button.
  + Review the output in the **Output** section. The output should identify the language as **en** (English).

**Review**: In this exercise, you created a function app that uses Azure Cognitive Services.

#### Exercise 2: Create a Logic App that uses a Function App

##### Task 1: Create a logic app

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Logic App** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Logic App**.
4. On the **Logic App** blade, click the **Create** button.
5. On the **Create logic app** blade, perform the following tasks:
   * In the **Name** text box, enter the value **CognitiveWorkflow**.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
   * In the **Location** drop-down list, select the same Azure region you chose in the previous exercise of this lab.
   * In the **Log Analytics** section, ensure that the **Off** button is selected.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next task.

##### Task 2: Configure logic app steps

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1001-RG**.
3. On the **AADesignLab1001-RG** blade, click the entry representing the logic app you created in the previous task.
4. On the **Logic Apps Designer** blade, scroll down and click the **Blank Logic App** tile in the **Templates** section.
5. On the **Logic Apps Designer** blade, click the **Code view** button at the top of the pane.
6. On the **Logic Apps Designer** blade, review the blank Logic App JSON template:

{ "definition": { "$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2016-06-01/workflowdefinition.json#", "actions": {}, "contentVersion": "1.0.0.0", "outputs": {}, "parameters": {}, "triggers": {} } }

1. Replace the default JSON template with the following template that includes an HTTP trigger (**\allfiles\AZ-301T01\Module\_01\LabFiles\Starter\logic-app.json**):

{ "definition": { "$schema": "https://schema.management.azure.com/providers/Microsoft.Logic/schemas/2016-06-01/workflowdefinition.json#", "actions": {}, "contentVersion": "1.0.0.0", "outputs": {}, "parameters": {}, "triggers": { "manual": { "inputs": { "method": "POST", "schema": { "properties": { "text": { "type": "string" } }, "type": "object" } }, "kind": "Http", "type": "Request" } } } }

1. On the **Logic Apps Designer** blade, click the **Designer** button.**Note**: At this point, you should see a single step in the designer. This is the "trigger" step that begins a workflow.
2. Click the **+ New Step** button in the designer.
3. In the **Choose an action** section, perform the following tasks:
   * In the search text box, type **Azure Functions**.
   * In the search results, select the action named **Choose an Azure function**.
   * In the next set of search results, select the Azure Function instance you created in the previous exercise of this lab.
   * In the final set of search results, select the **DetermineLanguage** function that will be used for the action.
4. In the **DetermineLanguage** step, perform the following tasks:
   * Click the **Show advanced options** link to display all options.
   * In the **Request Body** text box, type **@triggerBody()**.
   * In the **Method** drop-down list, select the **POST** option.
5. Click the **+ New Step** button in the designer. Click the **Add an action** button to open the dialog for creating an action.
6. In the **Choose an action** dialog that displays, perform the following tasks:
   * In the search text box, type **Azure Functions**.
   * In the search results, select the action named **Choose an Azure function**.
   * In the next set of search results, select the Azure Function instance you created in the previous exercise of this lab.
   * In the final set of search results, select the **DetermineKeyPhrases** function that will be used for the action.
7. In the **DetermineKeyPhrases** step, perform the following tasks:
   * Click the **Show advanced options** link to display all options.
   * In the **Request Body** text box, enter the value **@body('DetermineLanguage')**.
   * In the **Method** drop-down list, select the **POST** option.
8. Click the **+ New Step** button in the designer.
9. In the **Choose an action** dialog that displays, perform the following tasks:
   * In the search text box, type **Response**.
   * In the search results, select the **Action** named **Response Request**.
10. In the **Response** step, perform the following tasks:
    * In the **Status Code** text box, ensure that the value **200** is specified.
    * In the **Body** text box, type **@body('DetermineKeyPhrases')**.
11. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.
12. Scroll to the top of the **Logic Apps Designer** area and click the **When a HTTP request is received** step.
13. Copy the value of the **HTTP POST URL** text box. This URL will be used later in this lab.

##### Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployedf resources in this lab
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1001-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 3: Validate Logic App using Python

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to open the interactive **python** terminal:

python

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to import the **requests** library:

import requests

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command (replacing the placeholder <logic app POST Url> with the value of your url recorded earlier in this lab) and press **Enter** to create a variable containing the value of your logic app's url :

url = "<logic app POST Url>"

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to send an HTTP POST request to trigger your logic app workflow:

response = requests.post(url, json={'text': 'Circumambulate the city of a dreamy Sabbath afternoon. Go from Corlears Hook to Coenties Slip, and from thence, by Whitehall, northward.'})

1. At the **Cloud Shell** command prompt at the bottom of the portal, type the following command and press **Enter** to display the output of the Logic App workflow:

print(response.status\_code, response.reason, response.text)

1. Close the **Cloud Shell** pane.

**Review**: In this exercise, you created a logic app that leverages the function app created in the previous exercise of this lab.

#### Exercise 3: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab10')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab10')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 2 Review Questions

**Language Understanding (LUIS)**  
  
You are designing a mobile solution for your company. Users must be able to speak to an app and issue commands such as “Order for Delivery” or “View Menu."  
  
Which Azure service should you recommend? Why? What are the benefits and limitations of the service?

Suggested Answer ↓

Language Understanding (LUIS) is a cloud-based API service that allows you to easily integrate natural language processing into any application. A client application for LUIS is any conversational application that communicates with a user in natural language to complete a task, such as ordering delivery food or viewing a restaurants menu. LUIS allows you to customize the language learning model for your application commands and integrations.

**Azure Machine**  
  
Your company collects data from various retail stores including customer purchase information. You are designing a solution to analyze the data and predict future customer purchases for each store.  
  
What Azure service should you recommend?

Suggested Answer ↓

Azure Machine Learning is an integrated, end-to-end data science and advanced analytics solution. Machine learning is a data science technique that allows computers to use existing data to forecast future behaviors, outcomes, and trends. Using machine learning, computers learn without being explicitly programmed. Azure Machine Learning includes several components, that are open-source friendly, to enable users the ability to develop, deploy and maintain predictive models into a range of environments. Forecasts or predictions from machine learning can make apps and devices smarter.

**Computer Vision API**  
Your company captures images of notes that users write on white boards during meetings. The images are stored in JPEG format. You are designing an app that scans the images.  
The app must extract any handwritten text and save the text to Azure storage.  
  
Which Azure service should you recommend? What additional actions can you perform with the images by using the service?

Suggested Answer ↓

The cloud-based Computer Vision API uses Microsoft Computer Vision algorithms to analyze visual content, such as JPEG pictures and extract handwritten text. The Computer Vision API can also detect faces, categorize and describe an image as well as moderate content contained within images. The Computer Vision API is an image analysis service to provide insights into any image in JPEG, PNG, GIF, or BMP format that is less than 4 megabytes.

# AZ-301T02-A - Designing a Data Platform Solution

### Welcome to Designing a Data Platform Solution

Welcome to *Designing a Data Platform Solution*. This course is part of a series of four courses to help students prepare for Microsoft’s Azure Solutions Architect technical certification exam AZ-301: Microsoft Azure Architect Design. These courses are designed for IT professionals and developers with experience and knowledge across various aspects of IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data management, budgeting, and governance.

This course contains the following three modules:

**Module 1** - Backing Azure Solutions with Azure Storage

This module describes how many Azure services use the Azure Storage service as a backing store for other application solution in Azure. The module dives into critical considerations when using Azure Storage as a supplemental service for an all-up Azure solution.

After completing this module, students will be able to:

•    Determine the ideal pricing option for Azure Storage based on a solution’s requirements.

•    Identify performance thresholds for the Azure Storage service.

•    Determine the type of Storage blobs to use for specific solution components.

•    Use the Azure Files service for SMB operations.

•    Identify solutions that could benefit from the use of StorSimple physical or virtual devices.

**Module 2** - Comparing Database Options in Azure

This module compares the various relational and non-relational data storage options available in Azure. Options are explored as groups such as relational databases (Azure SQL Database, MySQL, and PostgreSQL on Azure), non-relational (Azure Cosmos DB, Storage Tables), streaming (Stream Analytics) and storage (Data Factory, Data Warehouse, Data Lake).

After completing this module, students will be able to:

•    Compare and contrast various database options on Azure.

•    Identify data streaming options for large-scale data ingest.

•    Identify longer-term data storage options.

**Module 3** - Monitoring & Automating Azure Solutions

This module covers the monitoring and automation solutions available after an Azure solution has been architected, designed and possibly deployed. The module reviews services that are used to monitor individual applications, the Azure platform, and networked components. This module also covers automation and backup options to enable business-continuity scenarios for solutions hosted in Azure.

This module contains the hands-on lab entitled Deploying Configuration Management Solutions to Azure.

After completing this module, students will be able to:

•    Compare and contrast monitoring services for applications, the Azure platform, and networking.

•    Design an alert scheme for a solution hosted in Azure.

•    Select the appropriate backup option for infrastructure and data hosted in Azure.

•    Automate the deployment of future resources for backup recovery or scaling purposes.

•    Determine the ideal pricing option for Azure Storage based on a solution’s requirements.

•    Identify performance thresholds for the Azure Storage service.

•    Determine the type of Storage blobs to use for specific solution components.

•    Use the Azure Files service for SMB operations.

•    Identify solutions that could benefit from the use of StorSimple physical or virtual devices.

### Azure Storage

This module describes how many Azure services use the Azure Storage service as a backing store for other application solution in Azure. The module dives into critical considerations when using Azure Storage as a supplemental service for an all-up Azure solution.  
  
After completing this module, students will be able to:

• Determine the ideal pricing option for Azure Storage based on a solution’s requirements.

• Identify performance thresholds for the Azure Storage service.

• Determine the type of Storage blobs to use for specific solution components.

• Use the Azure Files service for SMB operations.

• Identify solutions that could benefit from the use of StorSimple physical or virtual devices.

Azure storage has many options which affect performance and pricing. To be able to select the best storage for your infrastructure and applications requires a knowledge of what to store, how to store it, access is and pay for it. This lesson covers the essential elements of Azure Storage.

After completing this lesson, you will be able to:

• Describe the three Azure storage accounts and features.

• Understand the redundancy options available in Azure Storage.

• Chose which Azure storage account works best for your requirements.

• Understand how to access, share and delegate storage accounts.

**Azure Storage**

Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support the big data scenarios required by scientific, financial analysis, and media applications. You can also store the small amounts of data required for a small business website as billing is calculated by usage, not capacity.  Storage uses an auto-partitioning system that automatically load-balances your data based on traffic. Since storage is elastic and decoupled from your application, you can focus on your workload while relying on Storage's elastic capabilities to scale to meet the demand for your applications.

All Storage services can be accessed using a REST API.  Client libraries are also available for popular languages and platforms such as:

• .NET

• Java/Android

• Node.js

• PHP

• Ruby

• Python

• PowerShell

Microsoft Azure provides a storage subsystem that allows users to create a variety of different types of storage. The storage can be provided in differing levels of performance.

### Storage Services

Storage accounts are subdivided into available storage services all intended for different functions:

• Blobs: VHDs and large blocks of data (images or documents)

• Tables: structured data (a NoSQL store)

• Queues: for message passing in applications and for backlog processing

• Files: Fully managed SMB 3.0 file shares

To store and manage these storage services Azure uses a container called a Storage Account.

This section deals with the storage account; a subsequent lesson will deal with managed Disks for Virtual Machine disk storage without the need for a storage account.

**Azure Storage Accounts**

There are three types of Azure Storage accounts.

| **Type of Account** | **General Purpose Standard** | **General Purpose Premium** | **Blob Storage (hot and cool**  **access tiers)** |
| --- | --- | --- | --- |
| Services Supported | Blob, File, Queue services | Blob service | Blob service |
| Types of Blobs supported | Block blobs, Page blobs and Append blobs | Page blobs | Block blobs and Append  blobs |

There are also three types of entities in Azure Storage accounts:

**Blobs**: Blobs are available in three forms all intended for different purposes:

• Page blobs lend themselves to storing random access files up to 8TB in size, ideal for VHD storage for Virtual Machines.

• Block blobs are for images or other documents and files up to 4TB in size.

• Append blobs are similar in format to block blobs but allow append operations and are ideal for auditing and logging applications such as log or monitoring data from many sources.

**Tables**: “*Massive auto-scaling NoSQL store*.”

Ideal for: User, device and service metadata, structured data. Features include:

• Schema-less entities with strong consistency.

• No limits on number of table rows or table size.

• Dynamic load balancing of table regions.

• Best for Key/value lookups on partition key and row key.

• Entity group transactions for atomic batching.

**Queues**: “*Reliable messaging system at scale for cloud services*.”

Ideal for: Data sharing, Big Data, Backups. Functions include:

• Decouple components and scale them independently.

• Scheduling of asynchronous tasks.

• Building processes/work flows.

Features include:

• No limits on number of queues or messages.

• Message visibility timeout to protect from component issues.

• **Update Message** to checkpoint progress part way through.

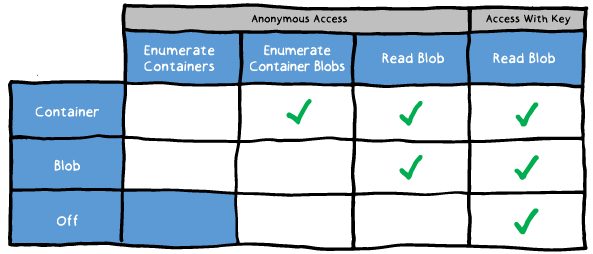
### Storage Account Security

Access to storage accounts is controlled by Storage Account Keys (there are primary and secondary to allow for key refreshes whilst maintaining access). If you have the key, you have access to the account and all the data in the account. Container security is also provided for blob storage at the time of creation; the container can be public, private or read-only.

To provide greater security, Azure provides Azure AD RBAC to allow for management functions on the Storage account and the use of RBAC on the storage account keys to prevent incorrectly applied access to the data itself. There are also Shared Access Signatures and Stored Access Policies to provide more granular access at the container, blob, queue or table level. A shared access policy adds the ability to revoke, expire or extend access. The use of all of these security features is a design decision when building your infrastructure or application.

### Container Security

Typically, only the owner of a storage account can access resources within that account. If your service or application needs to make these resources available to other clients, you have various options available. First, you can make the public access key generally available. This is not typically recommended as this key gives individuals full access to your entire storage account and its management operations. Another, more common option is to manage access to the entire container.



The Public Read Access property controls what data is available anonymously for your container. You can select the following values for the Public Read Access setting:

• **Container:** Blobs in a container can be enumerated. The container metadata is also accessible. Individual blobs within this container and their properties can also be accessed with this setting.

• **Blob:** Only individual blobs and their properties in this container can be accessed. Blobs are not allowed to be enumerated.

• **Off:**With this setting, enumeration of blobs is not allowed. Individual blobs and their properties are also not accessible. You must use your access keys to access any data about this container or its blobs.

### Shared Access Signatures

A shared access signature is a URI that grants restricted access rights to containers, blobs, queues, and tables. You can provide a shared access signature to clients who should not be trusted with your storage account key but to whom you wish to delegate access to certain storage account resources. By distributing a shared access signature URI to these clients, you can grant them access to a resource for a specified period of time, with a specified set of permissions.

A shared access signature can grant any of the following operations to a client that possesses the signature:

• Reading and writing page or block blob content, block lists, properties, and metadata

• Deleting, leasing, and creating a snapshot of a blob

• Listing the blobs within a container

• Adding, removing, updating, and deleting queue messages

• Getting queue metadata, including the message count

• Querying, adding, updating, deleting, and upserting table entities

• Copying to a blob from another blob within the same account

The shared access signature URI query parameters incorporate all of the information necessary to grant controlled access to a storage resource. The URI query parameters specify the time interval over which the shared access signature is valid, the permissions that it grants, the resource that is to be made available, and the signature that the storage services should use to authenticate the request.  
   
For example, you may wish to have your client application access a resource (container: pictures, blob: profile.jpg) at this URI:

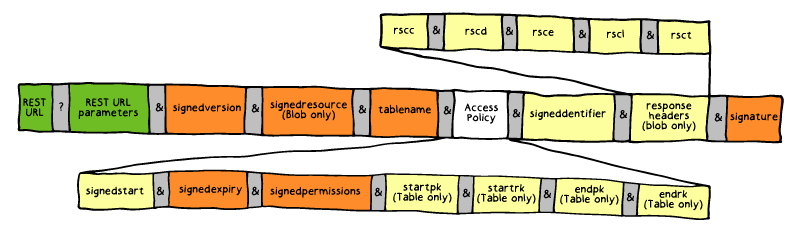
**GET https://[account].blob.core.windows.net/pictures/profile.jpg**

You could give the client application your access key to manage your entire subscription or make the pictures container anonymously accessible. Neither of these is ideal solutions, so you instead chose to generate a shared access signature. You simply append the generated signature to the end of your URI like this:

**GET https://[account].blob.core.windows.net/pictures/profile.jpg?sv=2012-02-12&st=2009-02-09&se=2009-02-10&sr=c&sp=r&si=YWJjZGVmZw%3d%3d&sig=dD80ihBh5jfNpymO5Hg1IdiJIEvHcJpCMiCMnN%2fRnbI%3d**

The signature is using version **2012-02-12** of the storage API; it allows read access is beginning from **02/09/09 to 02/10/09** to the container.

The structure of a SAS URI is the composition of multiple pieces.



Now the resource is temporarily accessible by this particular client.

### Stored Access Policies

Azure SAS also supports server-stored access policies that can be associated with a specific resource such as a table or blob. This feature provides additional control and flexibility compared to application-generated SAS tokens and should be used whenever possible. Settings defined in a server-stored policy can be changed and are reflected in the token without requiring a new token to be issued, but settings defined in the token itself cannot be changed without issuing a new token. This approach also makes it possible to revoke a valid SAS token before it has expired.

A stored access policy provides an additional level of control over shared access signatures on the server side. Establishing a stored access policy serves to group shared access signatures and to provide additional restrictions for signatures that are bound by the policy. You can use a stored access policy to change the start time, expiry time, or permissions for a signature, or to revoke it after it has been issued.

New shared access signatures are then generated from an existing stored access policy. A maximum of five access policies may be set on a container, table, or queue at any given time. To revoke a stored access policy, you can either delete it or rename it by changing the signed identifier. Changing the signed identifier breaks the associations between any existing signatures and the stored access policy. Deleting or renaming the stored access policy immediately effects all of the shared access signatures associated with it.

### Storage Account Replication

The data in your Microsoft Azure storage account is always replicated to ensure durability and high availability.  At a minimum, your data is stored in triplicate.  You may also choose extended replication options for scenarios where you require your data to be replicated across geography.

Options include:

**• Locally redundant storage (LRS):** Locally redundant storage maintains three copies of your data. LRS is replicated three times within a single facility in a single region. LRS protects your data from normal hardware failures, but not from the failure of a single facility. LRS is the minimum amount of replication.

• **Zone-redundant storage (ZRS):** Zone-redundant storage maintains three copies of your data. ZRS is replicated three times across two to three facilities, either within a single region or across two regions, providing higher durability than LRS. ZRS ensures that your data is durable within a single region.

• **Geo-redundant storage (GRS):** Geo-redundant storage is enabled for your storage account by default when you create it. GRS maintains six copies of your data. With GRS, your data is replicated three times within the primary region, and is also replicated three times in a secondary region hundreds of miles away from the primary region, providing the highest level of durability. In the event of a failure at the primary region, Azure Storage will failover to the secondary region. GRS ensures that your data is durable in two separate regions.

• **Read access geo-redundant storage (RA-GRS):** Read access geo-redundant storage replicates your data to a secondary geographic location, and also provides read access to your data in the secondary location. Read-access geo-redundant storage allows you to access your data from either the primary or the secondary location, in the event that one location becomes unavailable. RA-GRS is also used in scenarios where reporting and other read-only functions can easily be distributed to the replica instead of the primary therefore spreading application load across multiple instances.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Replication** | **LRS** | **ZRS** | **GRS** | **RA-GRS** |
| Data stored in multiple datacenters | No | Yes | Yes | Yes |
| Data read from secondary & primary location | No | No | No | Yes |
| No of copies of data stored in separate nodes | 3 | 3 | 6 | 6 |

     **Note**: Geographically distributed replicas receive any replication asynchronously. This means that your replica is eventually consistent and could possibly have older data if you access the replica before the replication operation from the primary is complete.

### Storage Performance and Pricing

Storage Accounts are available in Standard and Premium tiers. Azure Premium Storage delivers low-latency and high-performance disk support for virtual machines (VMs). Premium Storage stores data on solid-state drives (SSDs). Standard VM disks may be migrated to Premium Storage. Using multiple disks gives your applications up to 256 TB of VM storage. Premium Storage provides up to 80,000 I/O operations per second (IOPS) per VM, and a disk throughput of up to 2,000 megabytes per second (MB/s) per VM.

Premium Storage allows the lift-and-shift of demanding applications such as Dynamics AX, Dynamics CRM, SQL Server, Exchange Server and SharePoint farms to Azure. Applications that require consistent high performance and low latency such as SQL Server, Oracle, MongoDB, MySQL, and Redis can run happily in Azure Premium Storage. Premium Storage accounts can only be created with LRS redundancy. Not all Azure VM sizes can take advantage of Premium disk support. Plan carefully your disk and VM sizes to ensure maximum price and efficiency benefits. Currently Premium Storage is available on DS-series, DSv2-series, GS-series, Ls-series, and Fs-series VMs within Azure. Premium disks are always available for use. The storage is allocated on creation and is charged for the whole disk size rather than the data used. This makes premium disks more expensive.

### Blob Storage

The Blob Storage Service provides the infrastructure for Azure to store and manage the Block and Page blobs required to host Azure VM Disks and all other random-access Blobs as well as the block blob documents, images, and files. This lesson provides insight into the managed and unmanaged disk service as well as the blob service within a storage account.

After completing this section, you will be able to:

• Describe managed disks, snapshots, and disk exports.

• Decide whether to deploy VM disks as managed or unmanaged.

• Deploy a managed disk.

• Deploy data disks to a VM for a storage space.

**Blob Storage**

Azure Blob storage was introduced in the previous lesson as part of the overall Azure Storage platform. In this lesson, we cover the Azure Blob service from an Azure IaaS VM Disk perspective. Looking at VM disks in a managed and unmanaged form and the premium versus standard performance tier of both.

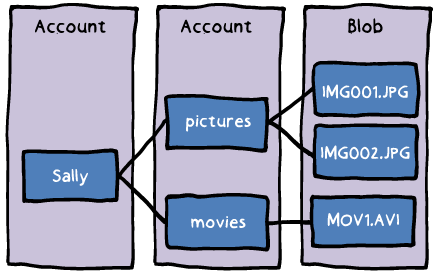
**VM Disks**

Azure IaaS VMs can attach OS and Data Disks. These disks can be premium storage (SSD based) or Standard storage (HDD). The performance of the disks depends upon the VM in which it is attached and the performance tier in which it is created. The management of VM data disks has been a challenging task in the past due to the performance limitations of a storage account. These were limited to a maximum of 20,000 IOPS per storage account. The Managed Disk service has provided a new feature in which Azure handles to underlying storage infrastructure and carries out all the redundancy and high availability tasks as well as ensuring that throughput is not throttled by the 20000 IOPS limit in storage accounts. Managed and un-managed disks have different feature sets and performance.  
   
The choice of whether to use Azure Files, Azure Blobs or Azure Disks is one primarily dictated by the data to be stored and the use of that data.

| **Feature** | **Description** | **Suggested Use** |
| --- | --- | --- |
| Files | SMB 3.0 interface, client libraries and a REST interface access from anywhere to stored files | Application lift and shift using native file system  API, Windows File Shares |
| Blobs | Client libraries, REST interface, for unstructured data stored in Block blobs | Streaming and random access, access application  data from anywhere |
| Disks | Client libraries, REST interface for persistent data accessed from a VHD | Access to data inside a Virtual machine on a VHD,  lift and shift file system API apps that write to  persistent disks |

### Unmanaged Disks

Unmanaged disks are available in Standard and Premium tiers. To use an un-managed disk, it is first necessary to create a Storage account and a container within the blob service of that account.



Once created blobs (including VM disks) can then be stored in the container. Each storage account has a limit of 20000 IOPS throughput. To be able to provide maximum performance on a striped storage space with multiple disks inside a VM it would be necessary to spread those data disks across many storage accounts.

Any Azure VM size can have multiple standard disks attached. A storage account can only store one type of disk. The performance of the storage account is chosen at time of creation. Un-managed standard disks can have disk snapshots (using Azure CLI or PowerShell), and VM images can be created with disks included.

### Managed Disks

Azure Managed Disks simplifies the creation and management of Azure IaaS VM Disks. When using the Managed Disk service, Azure manages the storage accounts in which the VM disks are stored. With un-managed disks, you need to create a storage account, a container and decide on Standard or Premium storage for the account. Then you need to ensure the VM disks are spread efficiently around several storage accounts. With Managed disks, the requirement is merely to choose the storage type and the disk size. Azure does the rest.

Managed disks provide several other benefits. First amongst these is the ability to upgrade a standard disk to a premium disk and downgrade a premium disk to a standard disk. Because there is no performance level for the underlying storage account, the Azure Managed disk architecture allows quick and straightforward performance tier changes. The only requirement is to detach the disk from a VM before it is upgraded or downloaded.

The architecture of Managed disks also provides:

• **Scalable and straightforward VM deployment**—no need to provision additional storage accounts when scaling the number of disks, the ceiling of 20000 IOPS does not exist. In addition, storing custom images on VHD files can now be accomplished with one storage account per region no need for additional storage accounts.

• **Managed Disks allows up to 10000 VM disks per subscription.** Using managed disks in VM Scale Sets allows up to 1000 VMs per scale set using a Marketplace image.

• **Better reliability for Availability Sets**—Managed disks use a different storage scale unit (or stamp) when using Availability Sets, this further ensures that all the disks in a VM are stored in the same scale unit, and more importantly each VM’s disks will be stored in separate scale units ensuring the application relying on those disks remains active.

• **Highly durable and available managed disks are designed for 99.999% availability**. Overall Azure manages a ZERO% annualized failure rate.

**• Granular access control**—Azure AD RBAC allows specific permissions to be assigned to managed disks for one or more users or groups. The granularity allows only read access or to preventing export actions on a disk.

**• Sizing and Pricing**(See: [https://azure.microsoft.com/en-us/pricing/details/managed-disks](https://azure.microsoft.com/en-us/pricing/details/managed-disks/))**:**

--Premium Managed disks are available from P4 which is 32 GB up to P50 which is 4 TB in size.  
--Standard Managed disks are available from S4 which is 32 GB again up to S50 which is 4 TB in size.  
--You are billed for the number of transactions performed on a standard managed disk. Premium disks do not attract transaction charges.

• **Managed Disk Snapshots**—A Managed Snapshot is a full copy of a managed disk which is stored as a standard read-only managed disk. Snapshots allow backups at any time. These are stand-alone objects and can be used to create new disks. You are only billed for the used size of the data on the disk. So, if a 4 TB disk that holds 500 GB of data, the snapshot will be billed for 500 GB.

• **Images**—Managed Disks support creating a managed custom image. This captures in a single image all managed disks attached to a VM.  
• **Images vs. snapshots**—An image is a copy of the VM and all disks attached. A snapshot is a single disk copy. For this reason, a snapshot is not a suitable choice for the scenario with multiple striped data disks. No snapshot co-ordination is available.

• **Managed Disks and Encryption**—By default Managed disks are encrypted by Azure Storage Service Encryption; this provides encryption at rest for disks, snapshots, and images. Azure Disk Encryption is also available at the VM level in Windows this uses BitLocker Drive Encryption. Azure Key Vault integration is included which allows users to bring their own disk encryption keys.

### Deployment Considerations

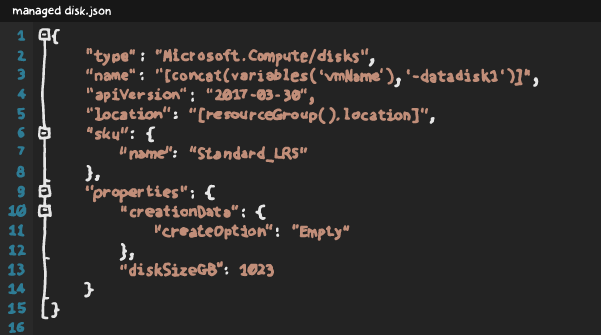
Deployment of Azure VM disks is easy to achieve using any of the standard tools:

**• Azure Portal**

**• PowerShell**

**• Azure CLI**

JSON templates may be used to create and format managed disks as well as un-managed disks.  
The data disks can be added within the VM creation template or defined stand alone as a top-level disk resource; this would then be attached when the VM is created.



### Azure Files

The ability to share files without the need to deploy the underlying server infrastructure provides several benefits when building an Azure based application. This lesson will describe the Azure File Service and the Azure File Sync service to allow the practitioner to choose which method of sharing files across an application or server infrastructure whether hybrid, on-premises or cloud-based.

After completing this lesson, you will be able to:

• Describe Azure Files service features.

• Decide which file sharing option suits and application best.

• Understand the benefits of Azure File Sync.

• Create an Azure File Sync service.

**Azure Files**

File storage offers shared storage for applications using the standard SMB 3.0 protocol. Microsoft Azure virtual machines and cloud services can share file data across application components via mounted shares, and on-premises applications can access file data in a share via the **File storage API.**

Applications running on Azure virtual machines or cloud services can mount a File storage share to access file data, just as a desktop application would mount a typical SMB share. Any number of Azure virtual machines or roles can mount and access the File storage share simultaneously.

Since a File storage share is a standard SMB 3.0 file share, applications running in Azure can access data in the share **via file I/O APIs**. Developers can, therefore, leverage their existing code and skills to migrate existing applications. IT Pros can use PowerShell cmdlets to create, mount, and manage File storage shares as part of the administration of Azure applications. This guide will show examples of both. Typical uses of File storage include:

• Migrating on-premises applications that rely on file shares to run on Azure virtual machines or cloud services, without expensive rewrites.

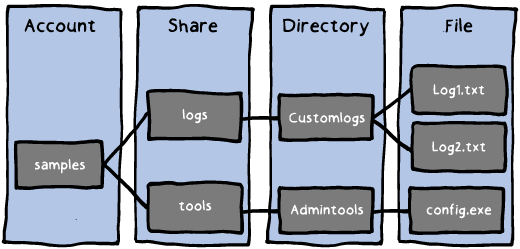
• Storing shared application settings, for example in configuration files.

• Storing diagnostic data such as logs, metrics, and crash dumps in a shared location.

• Storing tools and utilities needed for developing or administering Azure virtual machines or cloud services.

**File Share Service Components**

File storage contains the following components:



**Components of File Share Service**

• **Storage Account**: All access to Azure Storage is done through a storage account.

• **Share**: A File storage share is an SMB 3.0 file share in Azure. All directories and files must be created in a parent share. An account can contain an unlimited number of shares, and a share can store an unlimited number of files, up to the capacity limits of the storage account.

• **Directory**: An optional hierarchy of directories.

• **File**: A file in the share. A file may be up to 1 TB in size.

**• URL format:** Files are addressable using the following URL format:

**https://[account].file.core.windows.net/<share>/<directory/directories>/<file>**

The following example URL could be used to address one of the files in the diagram above:

**HTTP://[account].file.core.windows.net/logs/CustomLogs/Log1.txt**

### Azure File Sync

The Azure File Sync Service is a new service that will allow you to centralize your file shares in Azure Files, whilst maintaining the compatibility of an on-premises file server with all the flexibility and performance benefits that provide. The Azure File Sync service turns your file server into a cache of the Azure-based file share. Any protocol installed on the Windows Server can access the file share, including SMB, NFS and FTPS., The ability to have as many servers as you need spread across the globe turns the service into a useful way of providing all the benefits of a distributed file system without the infrastructure and maintenance requirements.

### Azure File Sync Terminology

When planning an Azure File Sync deployment, there are several components required to deploy the service efficiently.

**• Storage Sync Service:** The Storage Sync Service is the Azure resource created to host the Azure File Sync service. The Storage Sync Service resource is deployed to a resource group and can be created and amended using JSON templates if required. This service is required since the service can sync between multiple storage accounts, hence an additional resource is required to manage this. A subscription can contain multiple storage sync services.

**• Sync Group:** A Sync Group defines and controls the hierarchy and topology of the files to be synced. The sync group will contain Cloud and Server endpoints. Async service can contain multiple sync groups.

**• Registered Server:** Before adding a server endpoint to a sync group, the server must be registered with a storage sync service. A server can only be registered to a single sync service. Async service can host as many registered servers as you need.

**• Azure File Sync Agent:** To register a server, you need to install the Azure File Sync Agent. This is a small downloadable MSI package comprising three components:

**-- FileSyncSvc.exe:** Monitors changes on Server Endpoints, and for initiating sync sessions to Azure.

**-- StorageSync.sys:** A file system filter, which handles tiering files to Azure Files.

**-- PowerShell Management cmdLets:** PowerShell cmdlets for the Microsoft.StorageSync Azure resource provider.

**• Server Endpoint:** Once registered, you can add a server to a Sync group, this then becomes a server endpoint. A server endpoint is synonymous with a folder or a volume on the server that will cache the contents of the Azure File Share. Cloud tiering is configured individually by server endpoint.

**• Cloud Endpoint:** When added to a sync group, an Azure File Share is a cloud endpoint. One Azure File Share can only be a member of one Cloud Endpoint and thereby can only be a member of one Sync Group. Any files that exist in a cloud endpoint or a server endpoint before they are added to the sync group, automatically become merged with all other files in the sync group.

### Azure Files Features

Below is a list of features available for Azure Files:

**• Adding files to the File Share:** Azure File Sync supports adding and removing files directly within the Azure file share. These files will only sync once every 24 hours down to the server endpoints. This is due to the change detection job only being scheduled once every 24 hours.

**• Cloud tiering:** Cloud tiering is a feature of Azure File Sync which can save considerable space on a server endpoint. When a file is tiered, the sync system filter replaces the local file with a pointer to the location if the Azure file share. This is marked as offline in NTFS. When accessed locally the file is downloaded and opened for use. This is Hierarchical Storage Management (HSM).

**• Supported versions of Windows Server:** Currently, Azure File Sync is supported by all GUI editions of Windows Server 2012 R2 and Windows Server 2016 versions of Windows.

**• Access control lists (ACL):** Supported and enforced on files held on Server endpoints. Azure Files do not currently support ACLs.

**• NTFS compression:** Fully supported, and Sparse files are fully supported but are stored in the cloud as full files, and any cloud changes are synced as full files on server endpoints.

**• Failover Clustering:** Supported for File Server for General Use but not for Scale-out file server for application data. Cluster Shared Volumes are not supported. To function correctly, the sync agent must be installed on every node of a cluster.

**• Data Deduplication:** Fully supported for volumes that do not have cloud tiering enabled.

**• Encryption solutions:** Azure File Sync, is known to work with BitLocker Drive Encryption and Azure Rights Management Services. NTFS Encrypted File System does not work with Azure File Sync.

### StorSimple

The use of Active Directory is widespread throughout the on-premises and cloud-based Windows infrastructure world. The advent of Azure AD brings many options for the Azure Architect to choose between. This lesson will examine the benefits of and differences between cloud only and hybrid solutions comprised of on-premises Active Directory Domain Services, Azure AD, and Azure AD Domain Services.

After completing this section you will be able to:

• Describe the StorSimple service and devices.

• Decide when to use StorSimple in your hybrid File storage solution.

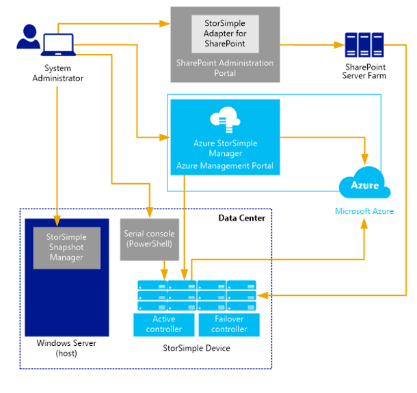
• Understand the StorSimple data tiering process.

**StorSimple**

StorSimple is the combination of a service, device and management tools that can create workflows for migrating data to a cloud storage center or back on premise.

The StorSimple device is an on-premises hybrid storage array that provides primary storage and iSCSI access to data stored on it. It manages communication with cloud storage and helps to ensure the security and confidentiality of all data that is stored on the StorSimple solution. The StorSimple device includes solid state drives (SSDs) and hard disk drives (HDDs), as well as support for clustering and automatic failover. It contains a shared processor, shared storage, and two mirrored controllers.

You can alternatively use StorSimple to create a virtual device that replicates the architecture and capabilities of the actual hybrid storage device. The StorSimple virtual device (also known as the StorSimple Virtual Appliance) runs on a single node in an Azure virtual machine.



StorSimple provides a web-based user interface (the StorSimple Manager service) that enables you to manage data center and cloud storage centrally. You can also use a Windows PowerShell-based, a command-line interface that includes dedicated cmdlets for managing your StorSimple device. Finally, you can interact with StorSimple using a Microsoft Management Console (MMC) snap-in that's used to configure and create consistent, point-in-time backup copies of local and cloud data.

**Features**

**• Transparent integration:**Microsoft Azure StorSimple uses the Internet Small Computer System Interface (iSCSI) protocol to invisibly link data storage facilities. This ensures that data stored in the cloud, in the data center, or on remote servers appears to be stored at a single location.  
 **• Reduced storage costs:** Microsoft Azure StorSimple allocates sufficient local or cloud storage to meet current demands and extends cloud storage only when necessary. It further reduces storage requirements and expense by eliminating redundant versions of the same data (deduplication) and by using compression.

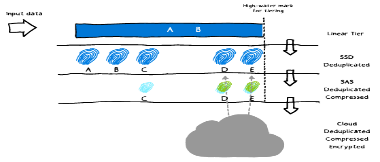
**• Simplified storage management:** Microsoft Azure StorSimple provides system administration tools that you can use to configure and manage data stored on-premises, on a remote server, and in the cloud. Additionally, you can manage backup and restore functions from a Microsoft Management Console (MMC) snap-in. StorSimple provides a separate, optional interface that you can use to extend StorSimple management and data protection services to content stored on SharePoint servers.

**• Improved disaster recovery and compliance:** Microsoft Azure StorSimple does not require extended recovery time. Instead, it restores data as it is needed. This means regular operations can continue with minimal disruption. Additionally, you can configure policies to specify backup schedules and data retention.

**• Data mobility:** Data uploaded to Microsoft Azure cloud services can be accessed from other sites for recovery and migration purposes. Additionally, you can use StorSimple to configure StorSimple virtual devices on virtual machines (VMs) running in Microsoft Azure. The VMs can then use virtual devices to access stored data for test or recovery purposes.

### Data Tiering

StorSimple automatically tiers and classifies your data based on how often you access it.  Data is always being shuffled between tiers as the mechanism learns about your usage patterns. Data that is most active is stored locally, while less active and inactive data is automatically migrated to the cloud.  To enable quick access, StorSimple stores very active data (hot data) on SSDs in the StorSimple device. It stores data that is occasionally used (warm data) on HDDs in the device or on servers at the datacenter. It moves inactive data, backup data, and data retained for archival or compliance purposes to the cloud. StorSimple adjusts and rearranges data and storage assignments as usage patterns change. For example, some information might become less active over time. As it becomes progressively less active, it is migrated from SSD to HDD and then to the cloud. If that same data becomes active again, it is migrated back to the storage device.



### Module 1 Review Questions

**Storage services**  
  
You are designing a solution for a company. The solution will use Azure Storage.  
  
What storage solutions does Azure offer? Explain how these different storage solutions are managed.

Suggested Answer ↓

Azure offers four storage solutions designed for different purposes. These include Blobs, Tables, Queues, and Files. To store and manage services, Azure uses a container called a storage account.

**Azure storage**  
  
You are designing a solution that uses Azure Storage.  
  
What types of Blob disk storage are available? Which type allows quick and straightforward performance tier changes?

Suggested Answer ↓

Blob storage offers managed and unmanaged disks type. There are many benefits to using managed including the ability the quickly change performance tiers.

**Azure storage**  
  
You are designing a solution for an IT security company. The company archives large sets of log files.  
  
You need to design an Azure solution to store the log files.  
  
Which Azure service should you use? What can you do to ensure your files are centralized?

Suggested Answer ↓

Azure Files provides the ability to store and share files without the need to deploy any underlying server infrastructure. The Azure File Sync Service is used to centralize file shares in Azure Files.

### Azure SQL Database

This module discusses the three managed database services in Azure; Azure SQL Database, Azure Database for MySQL, and Azure Database for PostgreSQL.

After completing this section you will be able to:

• Describe the difference between the three services for SQL, MySQL and PostgreSQL.

• Determine when to use advanced features of Azure SQL Database such as Elastic Database and Stretch Database.

**Azure SQL Database**

SQL Database is a relational database as-a-service offering that provides predictable performance and a high degree of compatibility with existing management tools.

**• Predictable Performance**: By using a consistent unit of measurement, such as Database Throughput Units, you can compare the expected service level for each performance tier that is offered in the SQL Database service. Consistent and predictable performance allows you to select a tier that very closely matches your application’s real-world utilization.

**• High Compatibility:** A Tabular Data Stream (TDS) endpoint is provided for each logical server that is created in the SQL Database service. You can use existing SQL client applications and tools with SQL Database by using the TDS protocol.

**• Simple Management:** Additional tools are available in Azure to manage databases that are created by SQL Database. A portal for managing database objects is available in the Azure Management Portal, which you can access by clicking the Manage button. You also can manage SQL Database instances by using the portals, REST API, Windows PowerShell, or the cross-platform command-line interface (Xplat CLI).

**Database Throughput Unit (DTU)**

DTUs are used to describe the capacity for a specific tier and performance level. DTUs are designed to be relative so that you can directly compare the tiers and performance levels. For example, the Basic tier has a single performance level (B) that is rated at 5 DTU. The S2 performance level in the Standard tier is rated at 50 DTU. This means that you can expect ten times the power for a database at the S2 performance level than a database at the B performance level in the Basic tier.

### Azure SQL Database Tiers

Azure SQL Database Tiers

The SQL Database service is offered in several tiers. You can select a tier that closely matches your application’s intended or actual resource needs. The following is a list of SQL Database service tiers with the associated performance characteristics:

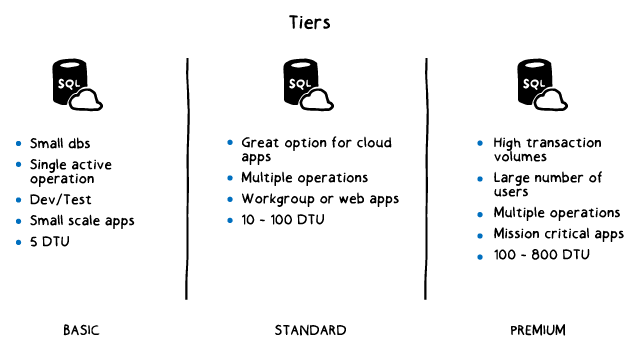
• **Basic**: Ideal for simple databases that requires only a single connection performing a single operation at a time.

• **Standard**: The most common option and is used for databases that require multiple concurrent connections and operations.

• **Premium:** Designed for applications that require large quantities of transactions at volume. These databases support a large quantity of concurrent connections and parallel operations.

These tiers are further separated into performance levels. Performance levels are very specific categories within a service tier that provides a specific level of service. For example, the P1 performance level in the Premium tier offers a maximum database size of 500 gigabyte (GB) and a benchmarked transaction rate of 105 transactions per second.

Every tier has one or more performance levels. In general, the performance levels in the Premium tier are rated higher than the performance levels in the Standard tier, which are again rated higher than the Basic tier. The following chart illustrates this distinction.



### Azure SQL Database Elastic Scale

The new Elastic Scale capabilities simplify the process of scaling out (and in) a cloud application’s data tier by streamlining development and management. Elastic Scale is composed of two main parts:

• An Elastic Scale library for client applications to configure shards and access shards.

• The Elastic Scale features in Azure SQL Database that implements any changes requested by your application.

Elastic Scale implements the database scaling strategy known as sharding. As a developer, you can establish a "contract" that defines a shard key and how shards should be partitioned across a collection of databases. The application, using the SDK, can then automatically direct transactions to the appropriate database (shard), perform queries across multiple shards or modify the service tier for existing shards. Elastic Scale also enables coordinated data movement between shards to split or merge ranges of data among different databases and satisfy common scenarios such as pulling a busy tenant into its own shard. The Split-Merge service is provided through a downloadable package that customers can deploy as an Azure cloud service into their own subscription.

### Third-Party Databases in Azure

Azure provides two additional managed database options for applications running  
on Azure.

• **Azure Database for MySQL:** This is a managed MySQL instance running the MYSQL community version. The instance comes with tools preinstalled like mysql.exe and phpMyAdmin. You can run one or more databases with this instance.

• **Azure Database for PostgreSQL:** This is a managed PostgreSQL instance that can run one or more databases. Both of the managed database services in Azure share a common set of features:

• Built-in high availability with no additional cost.

• Predictable performance, using inclusive pay-as-you-go pricing.

• Scale on the fly within seconds.

• Secured to protect sensitive data at-rest and in-motion.

• Automatic backups and point-in-time-restore for up to 35 days.

• Enterprise-grade security and compliance.

**Other Options**

Using Windows or Linux virtual machines, you can always install and run MySQL in the Azure environment. Clear DB also provides a managed MySQL instance that you can create from the Azure Marketplace.

### Azure Storage

This Module talks about some of the NoSQL services available in Azure to store and perform analysis on massive data sets.

Lesson Objectives

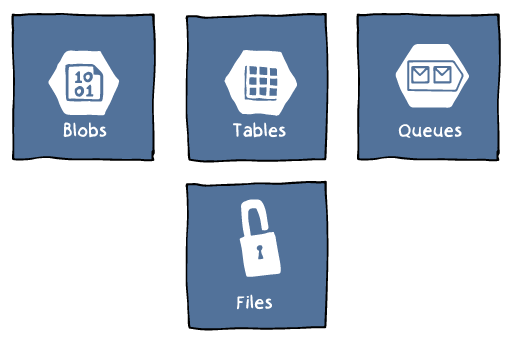
After completing this section you will be able to:

• Identify when to use Azure Storage Tables in a solution.

• Integrate Azure Search with existing data solutions.

**Azure Storage**

Azure Storage is massively scalable, so you can store and process hundreds of terabytes of data to support the big data scenarios required by scientific, financial analysis, and media applications. You can also store the small amounts of data required for a small business website as billing is calculated by usage, not capacity. Storage uses an auto-partitioning system that automatically load-balances your data based on traffic. Since storage is elastic and decoupled from your application, you can focus on your workload while relying on Storage's elastic capabilities to scale to meet demand for your applications.



All Storage services can be accessed using a REST API. Client libraries are also available for popular languages and platforms such as:

• .NET

• Java/Android

• Node.js

• PHP

• Ruby

• Python

• PowerShell

Finally, all resources in Storage can be protected from anonymous access and can be used in the Valet-Key pattern configuration discussed in previous modules.

### Replication

The data in your Microsoft Azure storage account is always replicated to ensure durability and high availability. At a minimum, your data is stored in triplicate. You may also choose extended replication options for scenarios where you require your data to be replicated across geography.

**• Locally redundant storage (LRS):** Locally redundant storage maintains three copies of your data. LRS is replicated three times within a single facility in a single region. LRS protects your data from normal hardware failures, but not from the failure of a single facility. LRS is the minimum amount of replication.

**• Zone-redundant storage (ZRS):** Zone-redundant storage maintains three copies of your data. ZRS is replicated three times across two to three facilities, either within a single region or across two regions, providing higher durability than LRS. ZRS ensures that your data is durable within a single region.

**• Geo-redundant storage (GRS):** Geo-redundant storage is enabled for your storage account by default when you create it. GRS maintains six copies of your data. With GRS, your data is replicated three times within the primary region, and is also replicated three times in a secondary region hundreds of miles away from the primary region, providing the highest level of durability. In the event of a failure at the primary region, Azure Storage will failover to the secondary region. GRS ensures that your data is durable in two separate regions.

**• Read access geo-redundant storage (RA-GRS):** Read access geo-redundant storage replicates your data to a secondary geographic location, and also provides read access to your data in the secondary location. Read-access geo-redundant storage allows you to access your data from either the primary or the secondary location, in the event that one location becomes unavailable. RA-GRS is also used in scenarios where reporting and other read-only functions can easily be distributed to the replica instead of the primary therefore spreading application load across multiple instances.

     **Note:** Geographically distributed replicas receive any replication asynchronously. This means that your replica is eventually consistent and could possibly have older data if you access the replica before the replication operation from the primary is complete.

**Architecture**

The Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency whitepaper that was released at the 2011 Association for Computing Machinery (ACM) Symposium on Operating Systems Principles.

### Azure Storage Tables

The Azure Table storage service stores large amounts of structured data. The service is a NoSQL datastore which accepts authenticated calls from inside and outside the Azure cloud. Azure tables are ideal for storing structured, non-relational data. Common uses of the Table service include:

• Storing TBs of structured data capable of serving web scale applications.

• Storing datasets that don't require complex joins, foreign keys, or stored procedures and can be de-normalized for fast access.  
   
• Quickly querying data using a clustered index.

• Accessing data using the OData protocol and LINQ queries with WCF Data Service .NET Libraries.  
You can use the Table service to store and query huge sets of structured, non-relational data, and your tables will scale as demand increases.

The Table service contains the following components:

**• URL format:** Code addresses tables in an account using this address format:

**http://<storage account>.table.core.windows.net/<table>**

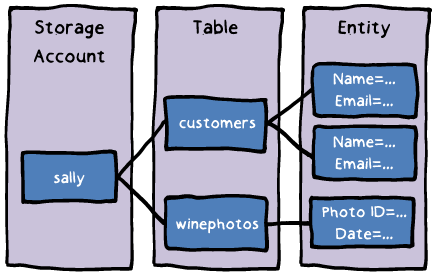
You can address Azure tables directly using this address with the OData protocol.

**• Storage Account:** All access to Azure Storage is done through a storage account.

**• Table:** A table is a collection of entities. Tables don't enforce a schema on entities, which means a single table can contain entities that have different sets of properties. The number of tables that a storage account can contain is limited only by the storage account capacity limit.

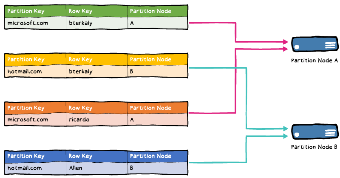
**• Entity:** An entity is a set of properties, similar to a database row. An entity can be up to 1 MB in size.

**• Properties:** A property is a name-value pair. Each entity can include up to 252 properties to store data. Each entity also has 3 system properties that specify a partition key, a row key, and a timestamp. Entities with the same partition key can be queried more quickly, and inserted/updated in atomic operations. An entity's row key is its unique identifier within a partition.



### Table Partitioning

Partitions represent a collection of entities with the same Partition Key values. Partitions are always served from one partition server and each partition server can serve one or more partitions. A partition server has a rate limit of the number of entities it can serve from one partition over time. Specifically, a partition has a scalability target of 500 entities per second. This throughput may be higher during minimal load on the storage node, but it will be throttled down when the node becomes hot or very active.  
   
To better illustrate the concept of partitioning, the following figure illustrates a table that contains a small subset of data for users. It presents a conceptual view of partitioning where the PartitionKey contains different values:



**Storage Table Partioning**

The primary key for an Azure entity consists of the combined PartitionKey and RowKey properties, forming a single clustered index within the table. The PartitionKey and RowKey properties can store up to 1 KB of string values. Empty strings are also permitted; however, null values are not. The clustered index sorts by the PartitionKey in ascending order and then by RowKey in ascending order. The sort order is observed in all query responses.

Because a partition is always served from a single partition server and each partition server can serve one or more partitions, the efficiency of serving entities is correlated with the health of the server. Servers that encounter high traffic for their partitions may not be able to sustain a high throughput. For example, if there are many requests for Partition B, server B may become too hot. To increase the throughput of the server, the storage system load-balances the partitions to other servers. The result is that the traffic is distributed across many other servers. For optimal load balancing of traffic, you should use more partitions, so that the Azure Table service can distribute the partitions to more partition servers.

The PartitionKey values you choose will dictate how a table will be partitioned and the type of queries that can be used. Storage operations, in particular inserts, can also affect your choice of PartitionKey values. The PartitionKey values can range from single values to unique values and also can be composed from multiple values. Entity properties can be composed to form the PartitionKey value. Additionally, the application can compute the value.

### Azure Search

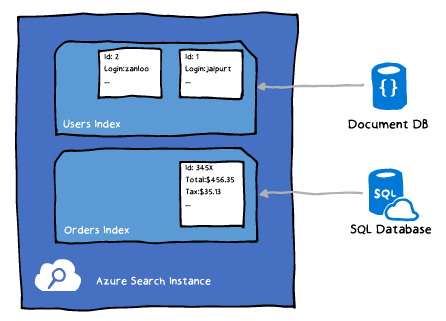
Azure Search is a fully managed cloud service that allows developers to build rich search applications using a .NET SDK or REST APIs. It includes full-text search scoped over your content, plus advanced search behaviors similar to those found in commercial web search engines, such as type-ahead query suggestions based on a partial term input, hit-highlighting, and faceted navigation. Natural language support is built-in, using the linguistic rules that are appropriate to the specified language.

You can scale your service based on the need for increased search or storage capacity. For example, retailers can increase capacity to meet the extra volumes associated with holiday shopping or promotional events. Azure Search is also an API-based service for developers and system integrators who know how to work with web services and HTTP. You can use existing platforms and frameworks since search only requires HTTP requests.

Azure Search is a PaaS service that delegates server and infrastructure management to Microsoft, leaving you with a ready-to-use service that you populate with search data, and then access from your application. Depending on how you configure the service, you'll use either the free service that is shared with other Azure Search subscribers, or the Standard pricing tier that offers dedicated resources used only by your service. Standard search is scalable, with options to meet increased demands for storage or query loads. Azure Search stores your data in an index that can be searched through full text queries. The schema of these indexes can either be created in the Azure Portal, or programmatically using the client library or REST APIs. The schema can also be auto-generated from an existing data source such as SQL Database or Document DB. Once the schema is defined, you can then upload your data to the Azure Search service where it is subsequently indexed.

     NOTE:  Search can index existing data stores including:

* + CosmosDB DocumentDB API
  + Azure SQL Database



### Azure Cosmos DB APIs

This section briefly introduces the Azure Cosmos DB NoSQL database service.

After completing this section you will be able to:

• Describe the general features available in the Azure Cosmos DB service.

• List the specific APIs and models available for Azure Cosmos DB client applications.

**Cosmos DB**

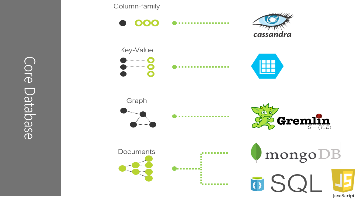
Azure Cosmos DB is Microsoft's new globally distributed, multi-model database service. Azure Cosmos DB offers a turn-key database service that allows you to create a database and distribute the data globally so that the data users access or in datacenters closer to them.  
Azure Cosmos DB APIs. Today, Azure Cosmos DB can be accessed using four different APIs:

**• DocumentDB (SQL) API**

**• MongoDB API**

**• Graph (Gremlin) API**

**• Tables (Key/Value) API**



Over time, Azure Cosmos DB will be expanded to offer new APIs and data models that are relevant to the latest distributed applications.

**Consistency Levels**

When you create a set of data in Azure Cosmos DB, your data is transparently replicated to ensure high availability. To accomplish this, the Azure Cosmos DB service automatically creates partitions, behind the scenes, and distribute your data across these partitions.

In Azure Cosmos DB, a partition is a fixed amount of high-performance storage that contains your data. When your data grows beyond the capacity of a partition, the Azure Cosmos DB service automatically determines the quantity of partitions needed and how to distribute the  
data across those partitions.

Additionally, you can specify a partition key to influence how your data is distributed. A partition key is a JSON path or property that is used by DocumentDB to ensure that related documents are stored in the same parition. This means that documents with the same partition key would be stored within the same partition. This also means that queries within a single partition perform better than queries that cross multiple partitions.

### Consistency Levels

When you create a set of data in Azure Cosmos DB, your data is transparently replicated to ensure high availability. To accomplish this, the Azure Cosmos DB service automatically creates partitions, behind the scenes, and distribute your data across these partitions.

In Azure Cosmos DB, a partition is a fixed amount of high-performance storage that contains your data. When your data grows beyond the capacity of a partition, the Azure Cosmos DB service automatically determines the quantity of partitions needed and how to distribute the  
data across those partitions.

Additionally, you can specify a partition key to influence how your data is distributed. A partition key is a JSON path or property that is used by DocumentDB to ensure that related documents are stored in the same parition. This means that documents with the same partition key would be stored within the same partition. This also means that queries within a single partition perform better than queries that cross multiple partitions.

DocumentDB allows you to specify one of four potential consistency levels per account. A consistency level specified at the database-level is applied automatically to all databases and collections within your account.  
The consistency levels range from very strong consistency where reads are guaranteed to be visible across replicas before a write is fully committed across all replicas to an eventual consistency where writes are readable immediately and replicas are eventually consistent with the primary.

| **Consistency Level** | **Description** |
| --- | --- |
| Strong | When a write operation is performed on your primary database, the write operation is  replicated to the replica instances. The write operation is only committed (and visible)  on the primary after it has been committed and confirmed by ALL replicas. |
| Bounded Stateless | This level is similar to the Strong level with the major difference is that you can configure how  stale documents can be within replicas. Staleness refers to the quantity of time (or version count)  a replica document can be behind the primary document. |
| Session | This level guarantees that all read and write operations are consistent within a user session.  Within the user session, all reads and writes are monotonic and guaranteed to be consistent across  primary and replica instances. |
| Eventual | This level is the loosest consistency and essentially commits any write operation against the  primary immediately. Replica transactions are asynchronously handle and will eventually (over  time) be consistent with the primary. This tier is the most performant as the primary database does  not need to wait for replicas to commit to finalize its transactions. |

https://www.skillpipe.com/api/2/content/8f2a013e-e841-5730-b049-de40d2cf67bb/4/OEBPS/Images/137632-58277.png

### Choosing a Consistency Strategy

There are two main things to consider when thinking about your consistency level. First a consistency level on the strong side of the list will ensure that your versions of documents in your replica do not lag behind the primary. If your application requires all replica documents to exactly match the primary at any point in time, this strategy makes a lot of sense. The downside is that the primary write operation will be a lot slower than usual because that operation must wait for every replica to confirm that the operation has been committed.

A consistency level on the eventual (loose) side will ensure that your database operates at peak efficiency. This occurs because operations against the primary database commit immediately and do not wait for the replicas to confirm that they are committed. This is useful for scenarios where you need the highest tier of performance. The downside here is that there is a potential for any read operations against a replica to be a couple of versions behind the primary and return inconsistent data.

### Data Storage & Integration Options

This section is a brief overview of various Data Storage & Data Integration options available on Azure.

After completing this section you will be able to:

• Describe the SQL Data Warehouse and Data Lake services.

• Describe the Data Factory service.

• Decide when it is appropriate to use SQL Data Warehouse, Data Lake or Data Factory.

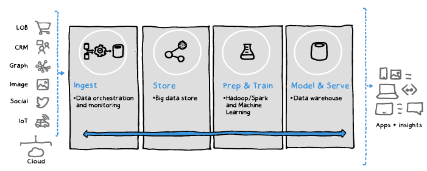
• Incorporate Data Factory into a design that uses either SQL Data Warehouse or Data Factory.

**Data Storage & Integration Options**

SQL Data Warehouse is a cloud-based Enterprise Data Warehouse (EDW) that leverages Massively Parallel Processing (MPP) to quickly run complex queries across petabytes of data. Use SQL Data Warehouse as a key component of a big data solution. Import big data into SQL Data Warehouse with simple PolyBase T-SQL queries, and then use the power of MPP to run high-performance analytics. As you integrate and analyze, the data warehouse will become the single version of truth your business can count on for insights.

In a cloud data solution, data is ingested into big data stores from a variety of sources. Once in a big data store, Hadoop, Spark, and machine learning algorithms prepare and train the data. When the data is ready for complex analysis, SQL Data Warehouse uses PolyBase to query the big data stores. PolyBase uses standard T-SQL queries to bring the data into SQL Data Warehouse.

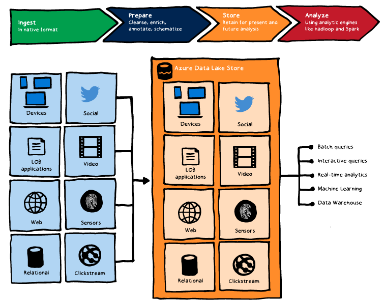
SQL Data Warehouse stores data into relational tables with columnar storage. This format significantly reduces the data storage costs, and improves query performance. Once data is stored in SQL Data Warehouse, you can run analytics at massive scale. Compared to traditional database systems, analysis queries finish in seconds instead of minutes, or hours instead of days.



The analysis results can go to worldwide reporting databases or applications. Business analysts can then gain insights to make well-informed business decisions.

**Azure Data Lake**

Azure Data Lake Store is an enterprise-wide hyper-scale repository for big data analytic workloads. Azure Data Lake enables you to capture data of any size, type, and ingestion speed in one single place for operational and exploratory analytics.  
   
Azure Data Lake Store can be accessed from Hadoop (available with HDInsight cluster) using the WebHDFS-compatible REST APIs. It is specifically designed to enable analytics on the stored data and is tuned for performance for data analytics scenarios. Out of the box, it includes all the enterprise-grade capabilities—security, manageability, scalability, reliability, and availability—essential for real-world enterprise use cases.

Some of the key capabilities of the Azure Data Lake include the following:  
 

Azure Data Lake Analytics is an on-demand analytics job service to simplify big data analytics. You can focus on writing, running, and managing jobs rather than on operating distributed infrastructure. Instead of deploying, configuring, and tuning hardware, you write queries to transform your data and extract valuable insights. The analytics service can handle jobs of any scale instantly by setting the dial for how much power you need. You only pay for your job when it is running, making it cost-effective. The analytics service supports Azure Active Directory letting you manage access and roles, integrated with your on-premises identity system. It also includes U-SQL, a language that unifies the benefits of SQL with the expressive power of user code. U-SQL’s scalable distributed runtime enables you to efficiently analyze data in the store and across SQL Servers in Azure, Azure SQL Database, and Azure SQL Data Warehouse.

### Data Integration

**Azure Data Factory**

Azure Data Factory is a platform created to help refine big data, enormous stores of raw data,  
into actionable business insights. It is a managed cloud service that’s built to handle (ETL)extract-transfer-load, data integration, and hybrid extract-transfer-load projects. You can use it to schedule and create data-driven workflows, other words known as pipelines, that can ingest data from data stores. Azure Data Factory operates in conjuncture with services such as Azure Data Lake Analysts, Azure Machine learning, and Spark to process and transform data.

Azure Data Factory is the service that can help with scenarios of this kind. Azure Data Factory is a cloud-based data integration service. It allows users to create data-driven workflows in the cloud, as well as methods to automate data movement and transformation. You can also issue output data to data stores to consume. By using Azure Data Factory, raw data can be organized into meaningful data lakes to assist in business decisions.

To give an example of the uses of Azure Data Factory imagine a social media company that collects petabytes of member information and logs that are produced by members in the cloud. The company would want to examine to gain insights on member preference, usage behavior, and demographics. The company would also want to discover opportunities to cross-sell or up-sell, generate new features, provide an excellent experience for the members, and drive business growth.

To examine this information the company needs to make use of various reference data. This data includes the member's information, marketing campaign information, and information on the social media platform which is all stored in an on-premises data store. The company could apply data from an additional log data from a cloud store, combining it with the on-premises data store.

When using Azure Data Factory, your data-driven workflows (better known as pipelines) typically performs the following four steps:

Connect & Collect,

Transform & Enrich,

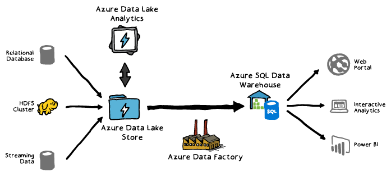
Publish,

and Monitor.

Data Factory connects to all required sources such as file shares, FTP web, databases, and software-as-a-service (SaaS) services. From here you can use Copy Activity within the data pipeline to move data to a centralized data store in the cloud for analysis.

After the raw data has been collected and moved to the centralized data store, Azure Data Factory processes the collected data using compute services such as Spark, Data Lake Analytics, and Machine Learning. The refined data, now in a business-ready consumable form, is loaded to an analytics engine such as Azure SQL database to be published to be used with your business intelligence tools. After building and deploying the data integration pipeline, you can monitor the scheduled activities and pipelines for failure and success rates.

As seen in the graphic below, Azure Data Factory is an option to migrate data from Azure Data Lake to Azure SQL Data Warehouse.



### Data Analysis Options

After completing this section you will be able to:

• Design a solution that uses Azure Analysis Services for end-user ad-hoc data set queries.

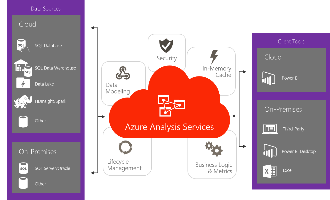
• Design a solution that surfaces “tribal knowledge” using Azure Data Catalog.

• Detail how Azure HDInsight can be integrated into solutions using various Azure Data storage, integration or analysis services.

**Data Analysis Options**

Azure Analysis Services provides enterprise-grade data modeling in the cloud. It is a fully managed platform as a service (PaaS), integrated with Azure data platform services.

With Analysis Services, you can mashup and combine data from multiple sources, define metrics, and secure your data in a single, trusted semantic data model. The data model provides an easier and faster way for your users to browse massive amounts of data with client applications like Power BI, Excel, Reporting Services, third-party, and custom apps.



Azure Analysis Services is compatible with many features already in SQL Server Analysis Services Enterprise Edition. Azure Analysis Services supports tabular models at the 1200 and 1400 compatibility levels. Partitions, row-level security, bi-directional relationships, and translations are all supported. In-memory and DirectQuery modes are also available for fast queries over massive and complex datasets.

For developers, tabular models include the Tabular Object Model (TOM) to describe model objects. TOM is exposed in JSON through the Tabular Model Scripting Language (TMSL) and the AMO data definition language.

### HDInsight

Apache Hadoop was the original open-source framework for distributed processing and analysis of big data sets on clusters. The Hadoop technology stack includes related software and utilities, including Apache Hive, HBase, Spark, Kafka, and many others. To see available Hadoop technology stack components on HDInsight, see Components and versions available with HDInsight. To read more about Hadoop in HDInsight, see the [Azure features page for HDInsight](https://azure.microsoft.com/en-us/services/hdinsight/).

Apache Spark is an open-source parallel processing framework that supports in-memory processing to boost the performance of big-data analytic applications.  
   
Azure HDInsight is the Azure distribution of the Hadoop components from the Hortonworks Data Platform (HDP). Azure HDInsight makes it easy, fast, and cost-effective to process massive amounts of data. You can use the most popular open-source frameworks such as Hadoop, Spark, Hive, LLAP, Kafka, Storm, R, and more to enable a broad range of scenarios such as extract, transform, and load (ETL); data warehousing; machine learning; and IoT.

HDInsight is big data processing on top of Azure Storage.



### Azure Data Catalog

Azure Data Catalog is a fully managed cloud service whose users can discover the data sources they need and understand the data sources they find. At the same time, Data Catalog helps organizations get more value from their existing investments.

With Data Catalog, any user (analyst, data scientist, or developer) can discover, understand, and consume data sources. Data Catalog includes a crowdsourcing model of metadata and annotations. It is a single, central place for all of an organization's users to contribute their knowledge and build a community and culture of data.

Data Catalog provides a cloud-based service into which a data source can be registered. The data remains in its existing location, but a copy of its metadata is added to Data Catalog, along with a reference to the data-source location. The metadata is also indexed to make each data source easily discoverable via search and understandable to the users who discover it.

After a data source has been registered, its metadata can then be enriched, either by the user who registered it or by other users in the enterprise. Any user can annotate a data source by providing descriptions, tags, or other metadata, such as documentation and processes for requesting data source access. This descriptive metadata supplements the structural metadata (such as column names and data types) that's registered from the data source.

Discovering and understanding data sources and their use is the primary purpose of registering the sources. Enterprise users might need data for business intelligence, application development, data science, or any other task where the right data is required. They can use the Data Catalog discovery experience to quickly find data that matches their needs, understand the data to evaluate its fitness for the purpose, and consume the data by opening the data source in their tool of choice.

At the same time, users can contribute to the catalog by tagging, documenting, and annotating data sources that have already been registered. They can also register new data sources, which can then be discovered, understood, and consumed by the community of catalog users.

### Case Study Overview

In this case study, we will look at a customer problem that requires an architectural recommendation.

After this case study, you should:

• Identify customer problems as they are related to networking.

• Design a solution that will meet the customer’s objectives.

• Ensure your designed solution accounts for customer objections.



**Who Is the Customer?**

Founded in 1954, Adventure Works Cycles has grown from a boutique manufacturer of high-quality bicycles and parts into one of the world’s largest makers of premium race and commuter bicycles. The Adventure Works mission has remained the same: to passionately pursue advanced, innovative technologies that help cyclists of all abilities find more enjoyment in the sport. Adventure Works Cycles Company manufactures and sells bicycles, bicycle parts, and bicycle accessories under the Adventure Works Cycles and Tailspin brands worldwide.

During the global recession of 2009, most of the company’s IT infrastructure was located in the company’s Provo, Utah, headquarters, but Adventure Works also had a sizable third-party colocation datacenter, costing US $30,000 to $40,000 a month, and other servers scattered around the United States.

Adventure Works knows that its datacenters are filled with dozens of smaller web servers and databases that run on underutilized hardware, and it has customer data scattered in multiple places. Faced with the prospect of a very large capital expenditure owing to the fact that the vast majority of their servers are now due for a hardware refresh, Adventure Works is looking for other options that eliminate the costly and high risk hardware refresh cycles.

**What Does the Customer Already Have?**

In reviewing all of Adventure Works’ web applications, it was determined that there were three archetypical database backed web applications present:

**• Product Catalog:** Resellers and consumers in North America, Asia and Europe access the product catalog via a website, the data for which is currently stored in SQL Server. Currently, both web app and database are hosted in its Utah datacenter. The database is 50 GB in size, and is not expected to grow past 100 GB.  
   
• **Inventory**: The inventory web application is a mission critical system primarily used by operations running in North America. Currently, both web app and SQL Server database are hosted in its Utah datacenter. This is Adventure Works' largest database at 3 TB in size, but it is not expected to grow beyond 5 TB.

• **Departmental**: These web applications support the regional offices only. Both web app and database are hosted in the Utah data center. While individually these have web applications have low demands and data sizes in the 500 MB-1 GB range, Adventure Works has 100’s of SQL Server databases supporting the numerous web apps, and fully expects new databases to be created to support departmental efforts.

Most of Adventure Works developers are already trained in Microsoft tools as all of Adventure Works' web applications and services are built using .NET. Additionally, they have already deployed Active Directory and are currently using a replicated directory in support of departments within each regional office.

### What Is the Customer's Goal?

Not only are all the servers expensive to acquire and maintain, but scaling the infrastructure takes significant time. “During and after a high-profile race, there’s a great deal of interest in our product, and our web apps receive a lot of hits,” says Hayley Leigh, Manager of Solution Development for Adventure Works Cycles. “It is difficult to scale our web hosting environment fast enough, and consumers and resellers could experience slow response times and even downtime.” Another challenge: Adventure Works conducts weekly server hardware maintenance, which causes downtime for some of its global offices.

Adventure Works wants to move many consumer-facing web apps, enterprise databases, and enterprise web services to Azure. “By using Microsoft global datacenters, we’re able to move infrastructure for key applications and web apps closer to the people who use them,” Leigh says. A big problem for Adventure Works is resellers and consumers in Japan and China have to use applications that run in a Utah datacenter, and because of the distance, encounter performance problems. Adventure Works would like to resolve this without the difficulty, expense and time requirements incurred by setting up infrastructure on the other side of the world using Adventure Works-owned servers.

**What Does the Customer Need?**

In reviewing all of Adventure Works’ web applications, it was determined that there were three archetypical database backed web applications present:

• **Product Catalog**: Resellers and consumers in North America, Asia and Europe access the product catalog via a website, the data for which is currently stored in SQL Server. Currently, both web app and database are hosted in its Utah datacenter. The database is 50 GB in size, and is not expected to grow past 100 GB.

• **Inventory**: The inventory web application is a mission critical system primarily used by operations running in North America. Currently, both web app and SQL Server database are hosted in its Utah datacenter. This is Adventure Works' largest database at 3 TB in size, but it is not expected to grow beyond 5 TB.

• **Departmental**: These web applications support the regional offices only. Both web app and database are hosted in the Utah data center. While individually these have web applications have low demands and data sizes in the 500 MB-1 GB range, Adventure Works has 100’s of SQL Server databases supporting the numerous web apps, and fully expects new databases to be created to support departmental efforts.

Most of Adventure Works developers are already trained in Microsoft tools as all of Adventure Works' web applications and services are built using .NET. Additionally, they have already deployed Active Directory and are currently using a replicated directory in support of departments within each regional office.

### What Things Worry the Customer?

**Scale & Performance**

* I do not want to have to make code changes (or re-deploy) in order to change the scale of a website.
* I hear Azure Web Apps is only useful for web apps with small amounts of traffic; will it really support the heavy traffic we receive?
* We would prefer to avoid performing a database migration (e.g., to another server) in order to scale the throughput of our database.
* We have heard SQL Database does not provide consistent performance, is this true?

**Business Continuity**

* How can we certain our data will survive in the event of a catastrophe in a certain part of the world?
* We need to be able to recover from mistakes made by administrators that accidentally delete production data (we know they happen, we would love an “undo”).
* Do we need to have multiple web server instances for each property to have a high SLA?

**Tool Familiarity**

* Will we need to learn new tools to develop for Azure Web Apps and SQL Database?
* What about diagnosing problems? Are there new tools we need purchase and learn?

**Connectivity**

* Some of our enterprise web services need to access data and other services located on-premises, is this supported?
* How can we ensure we are delivering the lowest latency possible to our website visitors?
* We need to ensure that if we have multiple web servers backing a given website, that no one web server gets all the traffic.

**Management**

* We would prefer not to have to manage patching of web servers and databases.
* With all of our web apps and databases around the world, how do we keep tabs on which is up and which is down and which is struggling?
* We need a simple solution to schedule and automate backup of the website and database.

**Security**

* Is it possible to allow our visitors to use a mix of legacy and modern browsers and still provide for secure transactions?
* What does Azure offer to help us with auditing access to our web servers and databases?
* Our staff is accustomed to accustomed to a single sign-on experience—will this still be possible?

### Case Study Solution

**Preferred Target Audience**

*Hayley Leigh, Manager of Solution Development for Adventure Works Cycles*

The primary audience is the business decision makers and technology decision makers. From the case study scenario, this would include the Manager of Solution Development. Usually we talk to the Infrastructure Managers who report into the CIO’s, or to application sponsors (like a VP LOB, CMO) or to those that represent the Business Unit IT or developers that report into application sponsors.

**Preferred Solution**

Adventure Works Cycles decided that cloud computing could solve just about all these problems. Moving some workloads into a cloud environment—where virtualized compute, storage, and network resources run in public datacenters, are shared by multiple parties, and are delivered over the internet—could reduce costs, improve scalability and business agility, and enable the desktop management team to manage remote computers.

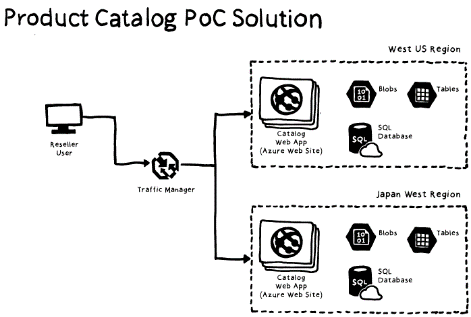
Cowles’s team evaluated cloud offerings from multiple providers and selected Microsoft Azure, the Microsoft cloud platform that provides on-demand compute, storage, content delivery, and networking capabilities from Microsoft datacenters. “The other companies offered infrastructure-as-a-service but not software-as-a-service or platform-as-a-service as Microsoft did,” Cowles says. “We wanted the whole spectrum of cloud options to fit a range of cloud needs. Also, most of our developers are trained in Microsoft tools, so it was much easier for us to connect our in-house systems to Microsoft Azure.”

Adventure Works then moved many consumer-facing web apps, enterprise databases, and enterprise web services to Azure. “By using Microsoft global datacenters, we’re able to move infrastructure for key applications and web apps closer to the people who use them,” Cowles says. For example, resellers and consumers in Japan and China previously had to use applications that ran in a Utah datacenter, and because of the distance, encountered performance problems. In about an hour, Cowles’s team set up the same services in a Microsoft datacenter in Asia and completely eliminated the performance lags. “It would have been extremely difficult, expensive, and time-consuming to set up this infrastructure on the other side of the world using Adventure Works-owned servers,” Cowles says.

### Proof of Concept (PoC)

The PoC scenario would be to address the latency issue faced by resellers around the world that currently rely on to access the Utah datacenter. The PoC would address showing how by using Azure Traffic Manager along with Azure Web Apps in multiple regions, one can ensure that these users are always being routed to the Web app that is “closest” to them in terms of minimizing network latency. The PoC could demonstrate this by provisioning a representative portion of the Product Catalog web application in an Azure Website, along with the storage required for it (e.g., Blob Storage, Table Storage and SQL Database) in the West US region, and then also provisioning the same infrastructure in the Japan West region. Traffic Manager would then be configured to route traffic using the Performance load balancing method.

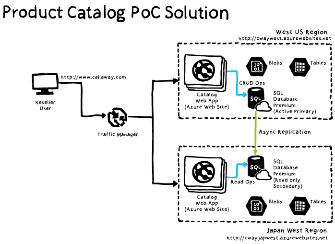
The following diagram illustrates this:



GEO-DISPERSE WEB APPLICATION

To complete the PoC, either Adventure Works staff or resellers located near each region could be asked to visit the PoC reseller web app, measure their experience (e.g., page load times, time to first paint, etc.) and report on the perceived performance (e.g., does it feel more responsive than the current site?).

On the database side, the PoC should be augmented to use the SQL Database Premium tier. The PoC would address showing using SQL Database Premium along with Azure Web Apps in multiple regions, whereby one configures SQL Database Premium to provide Active Geo-Replication across two regions (West US and Japan West as examples). SQL Database Premium could be configured with the primary in the West US Region, and a readable replica available in the Japan West Region.   
The following diagram illustrates this:



 SYNCING DATABASE SOURCES

By approaching it this way, the PoC could explore the benefits of accessing read-only data from the database nearest the Web app, while reserving data modifications operations for the active primary database instance.

To complete the PoC, Adventure Works could simulate a regional failure and switch to using the SQL Database in the Japan West region as the active primary. They could go one step further and stop the Web Apps in the West US region and thereby simulate a complete regional failure of the West US region. By having Traffic Manager in place, all traffic would flow to the Japan West Region, and the active database would be in that region as well.

### Risks & Mitigation

By demonstrating the performance load balancing method, the PoC helps Adventure Works to immediately establish confidence in Azure’s global capabilities to deliver a high performance experience to their customers and resellers around the world. By demonstrating the fail over between regions, the PoC helps Adventure Works to immediately establish confidence in Azure’s global capabilities to deliver a highly available experience to their customers and resellers around the world, even in the face of a regional failure.

By migrating the Product Catalog web app, it can also help to mitigate the risk the components used by the reseller website are incompatible with Azure Web Apps (for example, if one of the components used by the web app application requires an MSI based install).

This PoC is a good opportunity to evaluate Adventure Works’s use of “naked domains” (e.g., a domain with no “www” subdomains) for the product catalog, because Azure Traffic Manager cannot directly resolve requests against naked domains. If naked domains are required, Adventure Works would need to configure DNS forwarding with their DNS provider (if their DNS provider supports it) or leverage another DNS forwarding service to redirect naked domain requests to resource having a subdomain.

By migrating a representative portion of the Product Catalog Web App and its data, it can also help to mitigate the risk that SQL Server features used by the database are incompatible with SQL Database (for example, if they are using CLR stored procedures).

**Success Criteria**

The Product Catalog Web Application (or portion thereof) and supporting data can be successfully migrated to Azure. Users in each region are reporting improved performance with reduced latency. When a failover is simulated, the web app remains functional (albeit users may experience greater latency).

### Implementation

**Inventory Web App**

For the inventory web app, whose database sees large database sizes (3-5 TB), only SQL Server in a VM can scale to that capacity. SQL Database is not a good recommendation here for two reasons:

• SQL Database has a maximum of 500 GB per database

• To grow beyond this using SQL Database would require sharding, which would require re-designing the web application. One of the customer needs is to avoid such re-architecting of the database structure and the making of large changes to the application.

**Departmental Web App**

For the departmental web apps, the database sizes were small (500 MB to 1 GB). In choosing between SQL Database and SQL in a VM, one should consider the ease of migrating brownfield apps. The customer stated having 100’s of such databases, so moving these to SQL Database could prove overly expensive and time consuming on account of any incompatibilities that would need to be worked thru. Therefore, for the existing departmental apps, these databases should be created with SQL Server in a VM.

For new, greenfield, applications that do not have such incompatibilities, the departmental customer is very likely to appreciate the minimal IT involvement required: to quickly provision a SQL Database for their departmental application without IT support, to benefit from the Point-in-Time Restore should a departmental user make a mistake that results in data loss, and to have high availability within a single data center without the extra configuration that setting up a SQL Server cluster or Availability Group would require. Given the small database sizes, most departmental solutions could leverage the cost efficient SQL Database Basic tier.

### Checklist of Preferred Objection Handling

**Scale & Performance**

**I do not want to have to make code changes (or re-deploy) in order to change the scale of a website.**

* With Azure Web Apps, you do not need to make changes or re-deploy in order to change the scale of a Web app.

**I hear Azure Web Apps is only useful for web apps with small amounts of traffic; will it really support the heavy traffic we receive?**

* Azure Web Apps is capable of supporting Web Apps with loads ranging from small amounts of traffic to large amounts of traffic—this is enabled by its ability to scale up the instance size and to scale out the number of instances in order to meet demand.

**We would prefer to avoid performing a database migration (e.g., to another server) in order to scale the throughput of our database.**

* You can move between SQL Database service tiers or performance levels using the Azure Portal or Azure PowerShell without having to migrate the database.

**We have heard SQL Database does not provide consistent performance, is this true?**

* It was true with Web and Business editions, but these are now in the process of being deprecated. The SQL Database Basic, Standard and Premium offerings allow you to purchase a database meeting specific performance criteria.

### Business Continuity

**How can we certain our data will survive in the event of a catastrophe in a certain part of the world?**

* By deploying Web Apps to multiple regions and using Azure Traffic Manager to route between them for performance, you will still get the benefit failing over to another web app in another region should all of the web app endpoints in one region become unavailable. It is worth noting that such a failover may not be to the next “closest” region in such scenarios.
* With Active Geo-Replication, a feature of SQL Database Premium, you can create and maintain up to four readable secondary databases across geographic regions. All transactions applied to the primary database are replicated to each of the secondary databases. The secondary databases can be used for read workloads, database migration, and protection against data loss during application upgrade as a failback option.
* SQL Database Standard provides Standard Geo-Replication which enables you to have a single offline secondary in a pre-set region that is different from the active primary. This secondary is offline in the sense that, while it is synchronized with the primary, it will refuse any other connections until a failover happens and the datacenter hosting the primary becomes unavailable.
* Geo-Restore is a feature available to Basic, Standard and Premium SQL Database. It enables you to request a restore of your database using the latest weekly full backup plus differential backup, to any server in any Azure region. These backups are stored in geographically redundant storage.

**We need to be able to recover from mistakes made by administrators that accidentally delete production data (we know they happen, we would love an “undo”).**

* Microsoft Azure SQL Database creates backups of your data, and gives you the ability to recover your data from unwanted deletions or modifications. With Point in Time Restore on SQL Database Premium, you can restore to a database state as far back as 35 days.

**Do we need to have multiple web server instances for each property to have a high SLA?**

* Unlike other Azure compute services, Web Apps provides a high availability SLA using only a single standard instance.

### Tool Familiarity

**Will we need to learn new tools to develop for Azure Web Apps?**

* No. You can use familiar tools like Visual Studio to develop for Azure Web Apps.
* Visual Studio and SQL Server Management Studio can both be used to manage SQL Database.

**What about diagnosing problems? Are there new tools we need purchase and learn?**

* No. While there are new tool options, such as using the Azure Portal, Server Control Manager (Kudu) or examining logs stored in Blob or Table storage, when it comes to diagnosing problems you can still use familiar tools like Visual Studio.
* SQL Server Management Studio and Dynamic Management Views can be used to diagnose problems with SQL Database and SQL Server in a VM.

### Connectivity

**Some of our enterprise web services need to access data and other services located on-premises, is this supported?**

* Yes. Using the Service Bus Relay, Virtual Networks VPN or ExpressRoute access to data and services located on-premises is made possible.

**How can we ensure we are delivering the lowest latency possible to our website visitors?**

* Use Traffic Manager with the Performance load balancing method and deploy your solution to Web Apps in multiple regions nearest to your web app visitor populations.

**We need to ensure that if we have multiple web servers backing a given website, that no one web server gets all the traffic.**

* Within a datacenter, the Azure load balancer automatically handles round-robin load balancing between instances.
* Outside the datacenter, Traffic manager can be used in the Round-Robin load balancing method to route requests across data centers.

### Management

**We would prefer not to manage patching of web servers and databases.**

* With Azure Web Apps, patching of the underlying virtual machines is performed automatically and is transparent to you.
* With SQL Database, patching of the host OS and the database is handled for you.

**With all of our web apps and databases around the world, how do we keep tabs on which is up and which is down and which is struggling?**

* Azure Web Apps provides support for end point monitoring, which enables you to collect responsiveness metrics of a given Web app from multiple endpoints around the world.
* You can monitor these metrics and also configure the sending of alerts when certain thresholds are exceeded using the Azure Portal.
* You can acquire an overview of Database health for each of your SQL Databases from the Azure Portal.

**We need a simple solution to schedule and automate backup of the website.**

* With Web app Backup, you can create scheduled backups of your Web app. The offline copy is stored in Blob storage.

### Security

**Is it possible to allow our visitors to use a mix of legacy and modern browsers and still provide for secure transactions?**

* Azure Web Apps provides support for both IP Based SSL and SNI SSL. The former is mechanism that should be used to support both legacy and modern browsers.

**What does Azure offer to help us with auditing access to our web servers**?

* Operations logs are available from the Azure portal that can be used to track various management operations. These logs can be retrieved by REST API for collecting them into more permanent storage.

**Our staff is accustomed to accustomed to a single sign-on experience-- will this still be possible?**

* Yes, this is possible using Azure Active Directory.

### Potential Benefits

By integrating Microsoft cloud solutions into its datacenter strategy, Adventure Works Cycles has been able to reduce IT costs by more than $300,000 a year while gaining greater datacenter scalability and datacenter agility. The IT team can respond to server requests in hours instead of months and “turn off” servers when they are no longer needed. With servers running in Microsoft datacenters around the world, Adventure Works can provide better application performance and availability to offices and customers that are located far from the company’s California datacenter.

**• Improve Scalability and Agility**

Adventure Works has gained a level of IT scalability and agility that it never before had.  
Cowles’s team can respond much faster to requests for IT resources. “With Azure, we can respond to business requests in hours versus the months that it took before,” Cowles says. “It’s especially valuable in responding to requests in non-US regions, because it’s difficult to set up infrastructure in Carlsbad and provide remote access to those applications.”

**• Enhance Performance and Availability of Core Business Systems**

Since moving important pieces of its infrastructure to Azure, Adventure Works has enjoyed higher overall availability of critical applications and web properties. “With Azure, we no longer have maintenance-related downtime that negatively affects our business around the world,” Cowles says.  
By migrating its business-to-business website to Microsoft Azure, Adventure Works will eliminate the cost of that on-premises infrastructure and also realize reliability and performance gains. “It’s a lot easier to load-balance virtual machines in Azure than physical servers in our datacenter,” Cowles says. “We can also mitigate the impact of server security updates. By moving the B2B infrastructure to Azure, we can apply updates only to the regional servers that need them without affecting uptime in all of our offices.”

### Lab Steps

#### Online Lab: Deploying Database Instances in Azure

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>.

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy a Cosmos DB database

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Create a Cosmos DB database and collection

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Cosmos DB** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Azure Cosmos DB**.
4. On the **Azure Cosmos DB** blade, click the **Create** button.
5. On the new **Azure Cosmos DB** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * Resource group: ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0701-RG**.
   * In the **Account Name** text box, type a globally unique value.
   * In the **API** drop-down list, select the **Core (SQL)** option.
   * In the **Location** drop-down list, select the Azure region in which you want to deploy resources in this lab.
   * Leave all remaining settings with their default values.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step. **Note**: The deployment should take less than 5 minutes.
7. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance. **Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
8. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
9. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location that you selected earlier in this task.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0701-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
10. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.
11. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the Azure Cosmos DB account you deployed earlier in this task:

RESOURCE\_GROUP='AADesignLab0701-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the CosmosDB account you created earlier in this task:

COSMOSDB\_NAME=$(az cosmosdb list --resource-group $RESOURCE\_GROUP --query "[0].name" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the primary key of the CosmosDB account you created earlier in this task:

PRIMARY\_KEY=$(az cosmosdb list-keys --resource-group $RESOURCE\_GROUP --name $COSMOSDB\_NAME | jq -r '.primaryMasterKey')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the URI of the CosmosDB account you created earlier in this task:

URI="https://$COSMOSDB\_NAME.documents.azure.com:443/"

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new CosmosDB database named **FinancialClubDatabase**:

az cosmosdb database create --url-connection $URI --key $PRIMARY\_KEY --db-name 'FinancialClubDatabase'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a fixed collection named **MemberCollection** in the newly created database:

az cosmosdb collection create --url-connection $URI --key $PRIMARY\_KEY --db-name 'FinancialClubDatabase' --collection-name 'MemberCollection' --throughput 400

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to display the value of the PRIMARY\_KEY variable:

echo $PRIMARY\_KEY

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to display the value of the URI variable:

echo $URI

**Note**: Take a note of these values - you will need them in the next exercise.

##### Task 3: Create and query documents in Cosmos DB

1. On the left side of the Azure Cosmos DB account blade, click **Data Explorer**.
2. In the **Data Explorer** pane, click the **MemberCollection** child node of the **FinancialClubDatabase** node.
3. Click the **New SQL Query** button at the top of the **Data Explorer** pane.
4. In the **Query 1** tab that opened, view the default query:

SELECT \* FROM c

1. Click the **Execute Query** button at the top of the query editor.
2. In the left pane of the Data Explorer, expand the **MemberCollection** node.
3. Click the **Documents** child node within the **MemberCollection** node.
4. In the new **Documents** tab that opened, click the **New Document** button at the top of the tab.
5. In the **Documents** tab, replace the existing document with the following document:

{ "firstName": "Pennington", "lastName": "Oneal", "age": 26, "salary": 90000.00, "company": "Veraq", "isVested": false }

1. Click the **Save** button at the top of the **Documents** tab.
2. In the **Documents** tab, click the **New Document** button at the top of the tab.
3. In the **Documents** tab, replace the existing document with the following document:

{ "firstName": "Suzanne", "lastName": "Oneal", "company": "Veraq" }

1. Click the **Save** button at the top of the **Documents** tab.
2. Switch back to the **Query 1** tab, re-run the default query SELECT \* FROM c by clicking the **Execute Query** button at the top of the query editor, and review the results.
3. In the query editor, replace the default query with the following query:

SELECT c.id, c.firstName, c.lastName, c.isVested, c.company FROM c WHERE IS\_DEFINED(c.isVested)

1. Click the **Execute Query** button at the top of the query editor and review the results.
2. In the query editor, replace the existing query with the following query:

SELECT c.id, c.firstName, c.lastName, c.age FROM c WHERE c.age > 20

1. Click the **Execute Query** button at the top of the query editor and review the results.
2. In the query editor, replace the existing query with the following query:

SELECT VALUE c.id FROM c

1. Click the **Execute Query** button at the top of the query editor and review the results.
2. In the query editor, replace the existing query with the following query:

SELECT VALUE { "badgeNumber": SUBSTRING(c.id, 0, 8), "company": c.company, "fullName": CONCAT(c.firstName, " ", c.lastName) } FROM c

1. Click the **Execute Query** button at the top of the query editor and review the results.

**Review**: In this exercise, you created a new Cosmos DB account, database, and collection, added sample documents to the collection, and run sample queries targeting these documents.

#### Exercise 2: Deploy Application using Cosmos DB

##### Task 1: Deploy API App code using Azure Resource Manager templates and GitHub

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template Deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click the **Load file** link.
7. In the **Open** file dialog that appears, navigate to the **\allfiles\AZ-301T02\Module\_02\LabFiles\Starter\** folder.
8. Select the **api.json** file.
9. Click the **Open** button.
10. Back on the **Edit template** blade, click the **Save** button to persist the template.
11. Back on the **Custom deployment** blade, perform the following tasks:
    * Leave the **Subscription** drop-down list entry set to its default value.
    * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0701-RG**.
    * In the **Terms and Conditions** section, click the **I agree to the terms and conditions stated above** checkbox.
    * Click the **Purchase** button.
12. Wait for the deployment to complete before you proceed to the next task.**Note**: Deployment from source control can take up to 10 minutes.

##### Task 2: Validate API App

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click the entry representing the newly created App Service API app.
4. On the API app blade, click **Application Settings**.
5. On the Application Settings blade, scroll down to the **Application settings** section and perform the following tasks:
   * Set the value of the **CosmosDB:AuthorizationKey** setting to the value of the **PRIMARY KEY** setting of the **Cosmos DB** account you created earlier in this lab.
   * Update the value of the **CosmosDB:EndpointUrl** setting to the value of the **URI** setting of the **Cosmos DB** instance you created earlier in this lab.
   * Click the **Save** button at the top of the pane.
6. On the left-side of the API app blade, click **Overview**.
7. Click the **Restart** button at the top of the blade and, when prompted to confirm, click **Yes**.
8. Click the **Browse** button at the top of the blade. This will open a new browser tab displaying the **Swagger UI** homepage.**Note**: If you click the **Browse** button before the API app has fully restarted, you may not be able to follow the remaining steps in this task. If this happens, refresh your browser until the API app is running again.
9. On the **Swagger UI** homepage, click **GET/Documents**.
10. Click the **Try it out!** button.
11. Review the results of the request.
12. Back on the **Swagger UI** homepage, click **POST/Populate**.
13. In the **Parameters** section, in the **Value** field for the **options** parameter, paste in the following JSON content:

{ "quantity": 50 }

1. In the **Response Messages** section, click the **Try it out!** button.
2. Review the results of the request.
3. Back on the **Swagger UI** homepage, click **GET/Documents**.
4. Locate the **Response Messages** section. Click the **Try it out!** button.
5. Review the results of the request.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

**Review**: In this exercise, you created a new API App that uses the .NET Core DocumentDB SDK to connect to Azure Cosmos DB collection and manage its documents.

#### Exercise 3: Connect Cosmos DB to Azure Search

##### Task 1: Create Azure Search Instance

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Search** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Azure Search**.
4. On the **Azure Search** blade, click the **Create** button.
5. On the **New Search Service** blade, perform the following tasks:
   * In the **URL** text box, enter a globally unique name. Record its value. You will use it later in this lab.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0701-RG**.
   * In the **Location** drop-down list, select the Azure region matching or near the location where you deployed Cosmos DB resource earlier in this labb
   * Click **Pricing tier**.
   * On the **Choose your pricing tier** blade, click **Free** and then click the **Select** button.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.
7. In the hub menu in the Azure portal, click **Resource groups**.
8. On the **Resource groups** blade, click **AADesignLab0701-RG**.
9. On the **AADesignLab0701-RG** blade, click the entry representing the newly created Azure Search instance.
10. On the Search service blade, click **Keys**.
11. In the **Keys** pane, record the value of **PRIMARY ADMIN KEY**. You will use it later in this lab.

##### Task 2: Index Cosmos DB Data in Azure Search

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click the entry representing the Azure Cosmos DB account you created earlier in this lab.
4. On the Azure Cosmos DB account blade, click **Add Azure Search**.
5. On the **Import data** blade, click **Search service** and, on the **Search services** blade, click the newly created Azure Search Service instance.**Note**: You will be presented with two blades: the **Data Source** blade with the **CosmosDB** option already selected and with the **New data source** blade.
6. On the **New data source** blade, perform the following tasks:
   * In the **Name** text box, type **cosmosdata**.
   * In the **Connection string** text box, accept the default entry.
   * In the **Database** drop-down list, select the **FinancialClubDatabase** entry.
   * in the **Collection** drop-down list, select the **MemberCollection** entry.
   * In the **Query** field, enter the following SQL query:SELECT c.id, c.firstName, c.lastName, c.age, c.salary, c.company, c.isVested, c.\_ts FROM c WHERE c.\_ts >= @HighWaterMark ORDER BY c.\_ts
   * Ensure that the **Query results ordered by \_ts** checkbox is selected.
   * Click the **OK** button.
7. On the **Cognitive Search** blade, click the **OK** button.
8. On the **Index** blade, perform the following tasks:
   * In the **Index name** text box, type **memberindex**.
   * In the **Key** drop-down list, ensure that the **id** entry is selected.
   * For the **id** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, and **SORTABLE** checkboxes are selected.
   * For the **firstName** field in the table, ensure that the **RETRIEVABLE**, **SORTABLE**, and **SEARCHABLE** options are selected.
   * For the **lastName** field in the table, ensure that the **RETRIEVABLE**, **SORTABLE**, and **SEARCHABLE** checkboxes are selected.
   * For the **age** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, **SORTABLE**, and **FACETABLE** checkboxes are selected.
   * For the **salary** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, **SORTABLE**, and **FACETABLE** checkboxes are selected.
   * For the **company** field in the table, ensure that the **RETRIEVABLE**, **FACETABLE**, and **SEARCHABLE** checkboxes are selected.
   * For the **isVested** field in the table, ensure that the **RETRIEVABLE**, **FILTERABLE**, **SORTABLE**, **FACETABLE** checkboxes are selected.
   * Click the **OK** button.
9. On the **Create an Indexer** blade, perform the following tasks:
   * In the **Name** text box, type **cosmosmemberindexer**.
   * In the **Schedule** section, select the **Custom** option.
   * In the **Interval (minutes)** text box, type **5**.
   * In the **Start time (UTC)** field, specify the current date and accept the default value of the time entry.
   * Click the **OK** button.
10. Back on the **Import data** blade, click the **OK** button.

##### Task 3: Validate API App

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click the entry representing the App Service API app you created earlier in this lab.
4. On the API app blade, click **Application settings**.
5. On the Application settings blade, scroll down to the **Application settings** section and perform the following tasks:
   * Set the value of the **Search:AccountName** setting to the name of the Azure Search instance you created earlier in this lab.
   * Set the value of the **Search:QueryKey** setting to the value of the **PRIMARY KEY** of the Azure Search instance you created earlier in this lab.
   * Click the **Save** button at the top of the blade.
6. On the API app blade, click **Overview**.
7. Click the **Restart** button at the top of the blade and, when prompted to confirm, click **Yes**.
8. Click the **Browse** button at the top of the blade. This will open a new browser tab displaying the **Swagger UI** homepage.**Note**: If you click the **Browse** button before the API app has fully restarted, you may not be able to follow the remaining steps in this task. If this happens, refresh your browser until the API app is running again.
9. On the **Swagger UI** homepage, click **Cosmos DB API v.1.0.0** at the top of the page and select the **Cosmos DB API v.2.0.0** option from the drop-down list.
10. Click **GET/Documents/search**.
11. In the **Parameters** section, in the **Value** text box of the **query** parameter, type the following text:

Oneal

1. In the **Response Messages** section, click the **Try it out!** button.
2. Review the results of the request.
3. In the **Parameters** section, in the **Value** text box of the **query** parameter, type the following text:

penn\*

1. In the **Response Messages** section, click the **Try it out!** button.
2. Review the results of the request.
3. Close the new browser tab and return to the browser tab displaying the Azure portal.

**Review**: In this exercise, you created an Azure Search instance that uses an indexer to index the documents in Azure Cosmos DB.

#### Exercise 4: Remove lab resources

##### Task 1: Delete the resource group

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0701-RG**.
3. On the **AADesignLab0701-RG** blade, click **Delete resource group**.
4. In the **Are you sure you want to delete "AADesignLab0701-RG"?** pane, in the **TYPE THE RESOURCE GROUP NAME** text box, type **AADesignLab0701-RG** and click **Delete**.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 2 Review Questions

**Azure SQL Database**  
  
You are designing mobile app for your company. You plan to use to Azure SQL Database to support the app.  
  
What are the benefits of using Azure SQL Database? How does Azure SQL Database support scaling? What are some reasons that you would choose an Azure storage solution other than Azure SQL Database?

Suggested Answer ↓

Azure SQL Database offers predictable performance, high compatibility, and simple management. Azure SQL Database provides elastic scaling. This technology uses a strategy known as sharding to simplify scaling.

**Azure storage services**  
  
You are designing mobile app for your company.  
  
The data store that supports the app must store den-ormalized data to optimize read access.  
  
Which Azure service should you use?

Suggested Answer ↓

The Azure Table Storage service stores large amounts of structured data and is great for storing normalized data for quick reads.

**Distributed data storage**  
  
You are designing an Azure solution for your company.  
  
The solution requires a globally distributed database solution that supports multiple database types.  
   
Which Azure storage service should you use? What data store types does it support?

Suggested Answer ↓

Azure Cosmos DB is Microsoft’s new globally distributed, multi-model database service. Azure Cosmos DB currently supports Document DB, MongoDB, Graph, and Tables.

### Azure Network Watcher

This module covers the monitoring and automation solutions available after an Azure solution has been architected, designed and possibly deployed. The module reviews services that are used to monitor individual applications, the Azure platform, and networked components. This module also covers automation and backup options to enable business-continuity scenarios for solutions hosted in Azure.

After completing this module, students will be able to:

• Compare and contrast monitoring services for applications, the Azure platform, and networking.

• Design an alert scheme for a solution hosted in Azure.

• Select the appropriate backup option for infrastructure and data hosted in Azure.

• Automate the deployment of future resources for backup recovery or scaling purposes.  
   
**Monitoring**

This lesson reviews a variety of services available in Azure to monitor your workloads and applications.

After completing this lesson, you will be able to:

• Select between OMS, Network Watcher, Security Center, Azure Monitor, Azure Service Health and Azure Advisors when determining which monitoring solution to use for workloads deployed to Azure.

• Query historical log data using Power BI.

• Integrate Application Insights into a custom software solution hosted on Azure.

**Azure Network Watcher**

Customers can build an end-to-end network in Azure by orchestrating and composing various individual network resources such as VNet, ExpressRoute, Application Gateway, Load balancers, and more. Monitoring is available on each of the network resources. We refer to this monitoring as resource level monitoring.

The end to end network can have complex configurations and interactions between resources, creating complex scenarios that need scenario-based monitoring through Network Watcher.

Network Watcher provides the following features:

• Topology

• Variable Packet Capture

• IP Flow Verify

• Next Hop

• Diagnostics Logging

• Security Group View

• NSG Flow Logging

• VPN Gateway Troubleshooting

• Network Subscription Limits

• Role Based Access Control

• Connectivity

### Network Monitor

Operations performed as part of the configuration of networks are logged. These logs can be viewed in the Azure portal or retrieved using Microsoft tools such as Power BI or third-party tools. Audit logs are available through the portal, PowerShell, CLI, and Rest API.  
Metrics are performance measurements and counters collected over a period of time. Metrics are currently available for Application Gateway. Metrics can be used to trigger alerts based on thresholds.

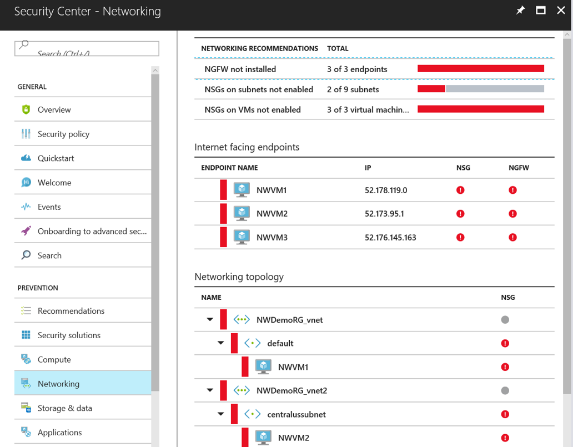
Periodic and spontaneous events are created by network resources and logged in storage accounts, sent to an Event Hub, or Log Analytics. These logs provide insights into the health of a resource. These logs can be viewed in tools such as Power BI and Log Analytics.

The troubleshooting blade, available in the portal, is provided on network resources today to diagnose common problems associated with an individual resource. This blade is available for the following network resources - ExpressRoute, VPN Gateway, Application Gateway, Network Security Logs, Routes, DNS, Load Balancer, and Traffic Manager.

### Azure Security Center

Azure Security Center provides unified security management and advanced threat protection for workloads running in Azure, on-premises, and in other clouds. It delivers visibility and control over hybrid cloud workloads, active defenses that reduce your exposure to threats, and intelligent detection to help you keep pace with rapidly evolving cyberattacks.

The Security Center Overview provides a quick view into the security posture of your Azure and non-Azure workloads, enabling you to discover and assess the security of your workloads and to identify and mitigate risk.



### Azure Monitor & Diagnostics

Azure Monitor is part of Microsoft Azure's overall monitoring solution. Azure Monitor helps you track performance, maintain security, and identify trends. Learn how to audit, create alerts, and archive data with our quickstarts and tutorials.

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. Azure Monitor provides several ways to configure and consume these metrics for monitoring and troubleshooting.  
**What can you do with metrics?**

Metrics are a valuable source of telemetry and enable you to do the following tasks:

• Track the performance of your resource (such as a VM, website, or logic app) by plotting its metrics on a portal chart and pinning that chart to a dashboard.

• Get notified of an issue that impacts the performance of your resource when a metric crosses a certain threshold.

• Configure automated actions, such as autoscaling a resource or firing a runbook when a metric crosses a certain threshold.

• Perform advanced analytics or reporting on performance or usage trends of your resource.

• Archive the performance or health history of your resource for compliance or auditing purposes.

**Metrics**

Azure Monitor enables users to obtain telemetry so that the user can gain visibility into the health and performance of workloads on Azure. Metrics is the essential type of Azure telemetry data that can be emitted by most Azure resources. Though Azure Monitor, a user has several ways to consume and configure these metrics for monitoring and troubleshooting.

Metrics have a set of characteristics you can use to identify it. Metrics become available immediately, meaning there is no need to set up additional diagnostics for metrics, nor opt-in for the data. Metrics also have a frequency of one minute. Users receive all metric values every minute from a resource, which allows for expanded visibility into the health and current state of your resource. Some metrics available can also have name-value pair attributes. These are known as dimensions which enable you to further segment and explore a metric. Moreover, Lastly, metrics allow users to access up to 30 days of history for each metric. By doing so, you can explore recent and monthly trends in the health and performance of your resource.

Users can use metrics to complete multiple tasks. These include tracking the performance of a resource by plotting its metrics on a chart. You can get notified of an issue if the issue impacts the performance of a resource. Since metrics have a frequency of one minute, this allows users to become aware of an issue on a near real-time basis. You can report on performance and usage trends on your resource to perform advanced analytics. After, users can choose to achieve health and performance history of a resource for compliance or auditing purposes.

Users can also choose to configure a metric alert rule that takes an automatic action, or even merely sends out a notification whenever the metric achieves a defined threshold. One of those automated actions is known as autoscale, which allows you to scale out your resource to meet incoming loads or requests. You can also route al metrics Log Analytics or Application Insights to enable instant Analytics. You can also choose to stream this information to an event hub. Doing so allows you to route them to Azure Stream Analytics for near-real-time analysis. You can choose to view all metrics, easily accessing them when you select a resource. You can also choose to achieve metrics to storage if you need to retrain them for longer than the archive period. You can choose o also route the metrics to an Azure Blob storage when you configure the settings for your resource.

### Azure Advisors

Azure Advisor is a personalized cloud consultant that helps you follow best practices to optimize your Azure deployments. It analyzes your resource configuration and usage telemetry. It then recommends solutions to help improve the performance, security, and high availability of your resources while looking for opportunities to reduce your overall Azure spend.

With Advisor, you can:

• Get proactive, actionable, and personalized best practices recommendations.

• Improve the performance, security, and high availability of your resources, as you identify opportunities to reduce your overall Azure spend.

• Get recommendations with proposed actions inline.

The Advisor dashboard displays personalized recommendations for all your subscriptions. You can apply filters to display recommendations for specific subscriptions and resource types. The recommendations are divided into four categories:

**• High Availability:** To ensure and improve the continuity of your business-critical applications.

**• Security:** To detect threats and vulnerabilities that might lead to security breaches.

**• Performance:** To improve the speed of your applications.

**• Cost:** To optimize and reduce your overall Azure spending.

### Azure Service Health

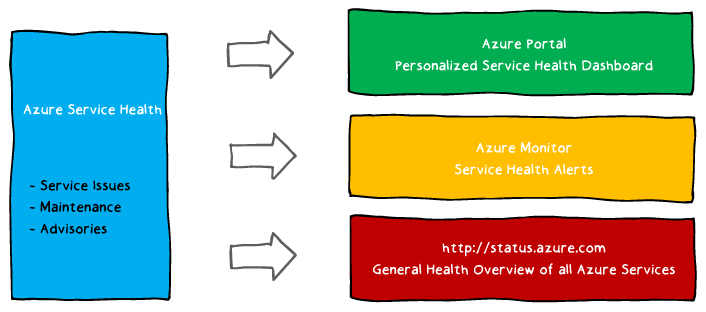
Azure Service Health provides personalized guidance and support when issues in Azure services affect you, and helps you prepare for upcoming planned maintenance. Azure Service Health alerts you and your teams via targeted and flexible notifications.

Service Health tracks three types of health events that may impact your resources:

• Service issues - Problems in the Azure services that affect you right now.

• Planned maintenance - Upcoming maintenance that can affect the availability of your services in the future.

• Health advisories - Changes in Azure services that require your attention. Examples include when Azure features are deprecated or if you exceed a usage quota.



### Operations Management Suite – Log Analytics

Log Analytics is part of Microsoft Azure's overall monitoring solution. Log Analytics monitors cloud and on-premises environments to maintain availability and performance. Get insight across workloads and systems to maintain availability and performance.

Log Analytics is a service in Operations Management Suite (OMS) that monitors your cloud and on-premises environments to maintain their availability and performance. It collects data generated by resources in your cloud and on-premises environments and from other monitoring tools to provide analysis across multiple sources. This article provides a brief discussion of the value that Log Analytics provides, an overview of how it operates, and links to more detailed content so you can dig further.

You can access Log Analytics through the OMS portal or the Azure portal which run in any browser and provide you with access to configuration settings and multiple tools to analyze and act on collected data. From the portal you can leverage log searches where you construct queries to analyze collected data, dashboards which you can customize with graphical views of your most valuable searches, and solutions which provide additional functionality and analysis tools.

### Application Insights

Application Insights is an extensible Application Performance Management (APM) service for web developers building and managing apps on multiple platforms. Learn how to detect and diagnose issues and understand usage for your web apps and services.

Application Insights is an extensible Application Performance Management (APM) service for web developers on multiple platforms. Use it to monitor your live web application. It will automatically detect performance anomalies. It includes powerful analytics tools to help you diagnose issues and to understand what users actually do with your app. It's designed to help you continuously improve performance and usability. It works for apps on a wide variety of platforms including .NET, Node.js and J2EE, hosted on-premises or in the cloud. It integrates with your DevOps process, and has connection points to a variety of development tools. It can monitor and analyze telemetry from mobile apps by integrating with Visual Studio App Center and HockeyApp.

### Power BI

With Azure services and Power BI, you can turn your data processing efforts into analytics and reports that provide real-time insights into your business. Whether your data processing is cloud-based or on-premises, straightforward or complex, single-sourced or massively scaled, warehoused or real-time, Azure and Power BI have the built-in connectivity and integration to bring your business intelligence efforts to life.

Power BI has a multitude of Azure connections available, and the business intelligence solutions you can create with those services are as unique as your business. You can connect as few as one Azure data source, or a handful, then shape and refine your data to build customized reports.

**Azure SQL Database and Power BI**

You can start with a straightforward connection to an Azure SQL Database, and create reports to monitor the progress of your business. Using the Power BI Desktop, you can create reports that identify trends and key performance indicators that move your business forward.  
Do you have more complex data, and all sorts of sources? No problem. With Power BI Desktop and Azure services, connections are just a tap of the Get Data dialog away. Within the same Query you can connect to your Azure SQL Database, your Azure HDInsight data source, and your Azure Blob Storage (or Azure Table Storage), then select only the subsets within each that you need, and refine it from there.

There are all sorts of scenarios where Azure and Power BI can be combined - the possibilities and opportunities are as unique as your business. For more information about Azure services, check out this overview page, which describes Data Analytics Scenarios using Azure, and learn how to transform your data sources into intelligence that drives your business ahead.

### Azure Backup

This lesson covers the many possible ways that Azure Backup can be integrated into an infrastructure solution hosted on Azure or in a hybrid-infrastructure scenario.

After completing this lesson, you will be able to:

• Detail the architectural components for deploying Azure Backup, using different topologies.

• Recognize the key use cases for implementing the different Azure Backup scenarios.

• Understand the built-in Security features that Azure Backup provides.

• Detail the architectural components for deploying Azure Site Recovery, using different topologies.

• Recognize the key use cases for implementing ASR, whether for DR or as a VM migration tool.

**Azure Backup**

Azure Backup is a simple and cost-effective backup as a service (BaaS) solution, that gives you trusted tools on-premises with rich and powerful tools in the cloud. It delivers strong protection for customer data wherever it resides—in your enterprise data center, remote and branch offices, or the public cloud—while being sensitive to the unique requirements these scenarios pose. Azure Backup, in a seamless portal experience with Azure Site Recovery, gives you cost-efficiency and minimal maintenance, consistent tools for offsite backups and operational recovery, and unified application availability and data protection.

### Backup Options

There are three primary options for backing up to Azure Backup with different characteristics:

1. Azure Backup / Restore of On-Premises Files & Folders:

* + Ideal for backing up Files & Folders only
  + Deploy the Azure Backup Agent (Azure Recovery Services Agent) on the VM guests running on-premises Hyper-V/ SCVMM / Vmware / Physical infrastructure.
  + Configure Azure Backup from within the VM guest.
  + Configure the integration with Azure Backup Vault.
  + Run the Backup job from within the VM guest.
  + Files & Folders backup will be stored in Azure Backup Vault, and can be restored from there.

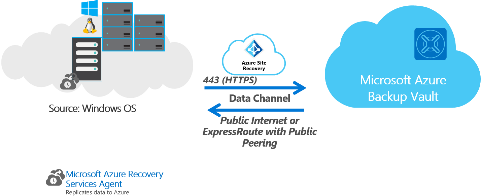
2. Azure Backup / Restore of On-premises running full workloads (OS, Sysvol, and Applications):

* + Better for backing up full system workloads (OS, system state, applications – consistent)
  + Deploy the Azure Backup Server (or System Center DPM 2012 R2 or 2016) on the on-premises Hyper-V / SCVMM / Vmware / Physical infrastructure.
  + Configure Azure Backup Server backup policies, backup storage (2-tier) and deploy agents to your workloads.
  + Configure the integration with Azure Backup Vault.
  + Run the Backup job from within the Azure Backup Server console.
  + VM workloads (system state, OS, applications,…) backup will be stored in Azure Backup Vault, and can be restored from there.

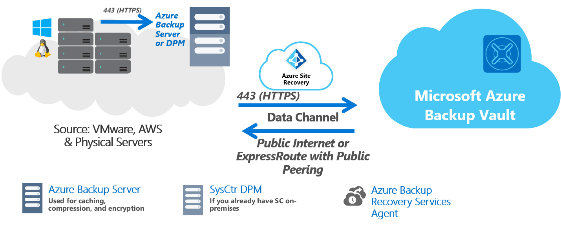
3. Azure VM Backup / Restore to Azure Backup Vault:

* + Best for when you want to backup Azure VMs to Azure Backup Vault
  + Deploy the Azure Backup Extension, or select Azure Backup in the VM configuration.
  + Configure Azure Backup backup policies, in the Azure platform.
  + Configure the integration with Azure Backup Vault.
  + Run the Backup job from within the Azure Platform.
  + Azure VMs will be backed up as full VM snapshots, and can be restored from within the Azure Portal.

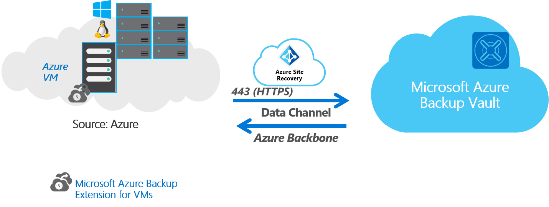
**Azure Backup / Restore of On-Premises Files & Folders**



**Azure Backup and Restore of On-Premises Running Full Workloads (OS, Sysvol, and Applications)**



**Azure VM Backup and Restore to Azure Backup Vault**



### Specialized Backup

**Hybrid Backup Encryption**

Security is an important aspect of any Public Cloud, and particularly in Microsoft Azure.

Azure Backup allows for end-to-end encryption of the backup platform:

1. It starts with a passphrase for the Azure Recovery Services Agent installation.

2. The next layer is the Backup Data itself, which gets encrypted in transit.

3. Once the data is stored in Azure Backup Vault, it gets encrypted at rest as well.

**Azure Backup Monitoring with Log Analytics**

* + Azure Backup monitoring is possible from Log Analytics, part of Azure Operations Management Suite.
  + Out of Log Analytics, one can get a detailed view on the backup statistics, the amount of data that is being consumed, successful and failed jobs and alike.

**Azure Backup Reports with Power BI**

* + Besides Azure Monitoring capabilities in Operations Management Suite and Log Analytics, Azure Backup also allows for reporting integration with Microsoft Power BI.

**Linux Application Consistent Azure Backup**

* + Taking backups of Azure VMs running Linux OS is fully supported, for Azure supported Linux Operating Systems.
  + To allow for application consistent backups, you need to run a pre- and post- backup script. The VM Snapshot will be your VM Backup, which gets stored in the Backup Vault using an incremental update process.

### Site Recovery

Azure Site Recovery is an Azure solution, initially built to provide a datacenter disaster recovery solution for your VM workloads. Whether they were running on-premises on Hyper-V hosts, VMware hosts, running as physical hosts, or as AWS VMs. Next to the core VM disaster recovery aspect of it, Azure Site Recovery is also an ideal tool for performing VM lift & Shift operations of your workloads.

**Why Azure Site Recovery?**

**Azure as your Disaster/Recovery Datacenter Site:**

* Replication-based failover to Azure Virtual Machines
* Near-zero downtime for your application workloads
* Application-consistent failover
* Failover & Failback
* DR for on-premises Hyper-V, VMware and physical Servers (\*), as well as Azure VMs

**Ideal as a Virtual Machine “Lift & Shift” migration tool:**

* Full machine-state replication to an Azure VM
* Perfect for test/dev scenarios
* Zero-data loss during migration

### Azure Automation

Azure Automation provides a way for users to automate the manual, long-running, error-prone, and frequently repeated tasks that are commonly performed in a cloud and enterprise environment. It saves time and increases the reliability of regular administrative tasks and even schedules them to be automatically performed at regular intervals. You can automate processes using runbooks or automate configuration management using Desired State Configuration. This lesson provides a brief overview of Azure Automation and answers some common questions.

After completing this section, you will be able to:

• Understand Azure Automation concepts and architecture.

• Recognize and describe Azure Automation capabilities.

• Interact with machines using Azure Automation and Desired State Configuration.

**Azure Automation**

Microsoft Azure Automation provides a way for users to automate the manual, long-running, error-prone, and frequently repeated tasks that are commonly performed in a cloud and enterprise environment. It saves time and increases the reliability of regular administrative tasks and even schedules them to be automatically performed at regular intervals. You can automate processes using runbooks or automate configuration management using Desired State Configuration.

Azure Automation is a software as a service (SaaS) application that provides a scalable and reliable, multi-tenant environment to automate processes with runbooks and manage configuration changes to Windows and Linux systems using Desired State Configuration (DSC) in Azure, other cloud services, or on-premises. Entities contained within your Automation account, such as runbooks, assets, Run As accounts are isolated from other Automation accounts within your subscription and other subscriptions.

Runbooks that you run in Azure are executed on Automation sandboxes, which are hosted in Azure platform as a service (PaaS) virtual machines. Automation sandboxes provide tenant isolation for all aspects of runbook execution – modules, storage, memory, network communication, job streams, etc. This role is managed by the service and is not accessible from your Azure or Azure Automation account for you to control.  
To automate the deployment and management of resources in your local datacenter or other cloud services, after creating an Automation account, you can designate one or more machines to run the Hybrid Runbook Worker (HRW) role. Each HRW requires the Microsoft Management Agent with a connection to a Log Analytics workspace and an Automation account. Log Analytics is used to bootstrap the installation, maintain the Microsoft Management Agent, and monitor the functionality of the HRW. The delivery of runbooks and the instruction to run them are performed by Azure Automation.

### Automation Flow

An Automation account is separate from the account you use to sign in to the portal to configure and use Azure resources. Automation resources included with an account are the following:

**• Certificates** - contains a certificate used for authentication from a runbook or DSC configuration or add them.

**• Connections** - contains authentication and configuration information required to connect to an external service or application from a runbook or DSC configuration.

**• Credentials** - is a PSCredential object which contains security credentials such as a username and password required to authenticate from a runbook or DSC configuration.

**• Integration modules** - are PowerShell modules included with an Azure Automation account to make use of cmdlets within runbooks and DSC configurations.

**• Schedules** - contains schedules that starts or stops a runbook at a specified time, including recurring frequencies.

**• Variables** - contain values that are available from a runbook or DSC configuration.

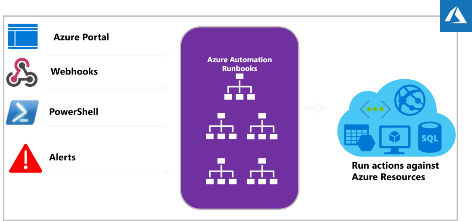
**• DSC Configurations** - are PowerShell scripts that describes how to configure an operating system feature or setting or install an application on a Windows or Linux computer.

**• Runbooks** - are a set of tasks that perform some automated process in Azure Automation based on Windows PowerShell.

When you create an Automation account in the Azure portal, you automatically create two authentication entities:

**• A Run As account**. This account creates a service principal in Azure Active Directory (Azure AD) and a certificate. It also assigns the Contributor role-based access control (RBAC), which manages Resource Manager resources by using runbooks.

**• A Classic Run As account**. This account uploads a management certificate, which is used to manage classic resources by using runbooks.  
Role-based access control is available with Azure Resource Manager to grant permitted actions to an Azure AD user account and Run As account, and authenticate that service principal. Read Role-based access control in Azure Automation article for further information to help develop your model for managing Automation permissions.



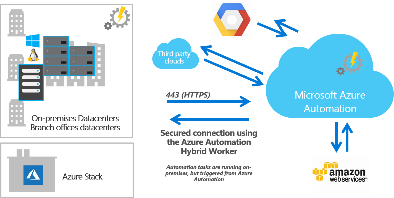
### Cross-Cloud

Although Azure Automation lives in the Azure platform, it allows management and configuration of Azure systems, on-premises running systems, systems in AWS, Google Cloud, or any other 3rd party hosting data center.

The core component of Azure Automation is defined and configured in the Azure Platform (cross Regions). From there, you can establish hybrid automation capabilities, by using the Azure Automation Hybrid Worker.

This is similar to running like an Azure Automation agent on your non-Azure cloud platforms, where Amazon Web Services, Google Cloud, your own Private Cloud datacenter, or any third party hosted datacenter if you want, would be a good example.

Latest supportability added to the Azure Automation feature set, is integration with on-premises running Azure Stack.



### Configuration Management

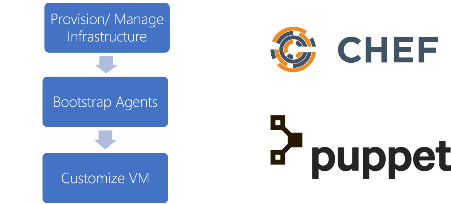
To create and manage Azure virtual machines (VMs) in a consistent manner at scale, some form of automation is typically desired. There are many tools and solutions that allow you to automate the complete Azure infrastructure deployment and management lifecycle. This article introduces some of the infrastructure automation tools that you can use in Azure.

**Chef**

Chef is an automation platform that helps define how your infrastructure is configured, deployed, and managed. Additional components included Chef Habitat for application lifecycle automation rather than the infrastructure, and Chef InSpec that helps automate compliance with security and policy requirements. Chef Clients are installed on target machines, with one or more central Chef Servers that store and manage the configurations.

**Puppet**

Puppet is an enterprise-ready automation platform that handles the application delivery and deployment process. Agents are installed on target machines to allow Puppet Master to run manifests that define the desired configuration of the Azure infrastructure and VMs. Puppet can integrate with other solutions such as Jenkins and GitHub for an improved devops workflow.



### Who Is the Customer?

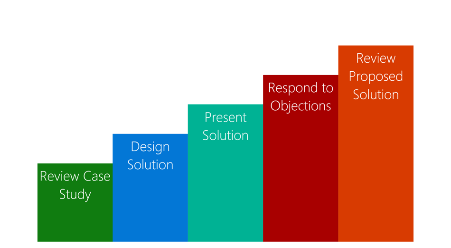
In this case study, we will look at a customer problem that requires an architectural recommendation.

After this case study, you should:

• Identify customer problems as they are related to networking.

• Design a solution that will meet the customer’s objectives.

• Ensure your designed solution accounts for customer objections.



**Who Is the Customer?**

Fabrikam Publishing is a media and publishing company in Seattle, Washington, with approximately 5,000 employees.  
What Does the Customer Already Have?

Fabrikam has a single data center that primarily runs Microsoft server software, including Active Directory Domain Services AD DS) and a number of AD-integrated services, including Exchange 2013, as well multi-tier, internal, AD-integrated IIS-based web applications with SQL Server 2014 as the database platform. The services are consumed by client systems hosted in three buildings located in adjacent areas of the city. Buildings are connected to the data center by using site-to-site VPN with the throughput of 100Mbps. Each building has also an independent connection to the internet.

Server backups are performed by using tape libraries with autoloaders. Tapes are periodically shipped for permanent storage to an offsite location.

If something catastrophic were to happen to the data center, the IT team would have to deploy replacement physical servers by reinstalling the operating system and restoring data from backups in the equipment rooms in small server rooms located in each building.

Fabrikam’s IT staff likes to stay current with the latest offerings from Microsoft so that the department functions as cost-effectively as possible. In order to reduce costs, the IT staff has recently started planning the initiative to virtualize majority of its physical servers using the Hyper-V platform and to deploy System Center 2012 R2 Virtual Machine Manager (SCVMM) for managing the resulting virtualized environment.

**What Is the Customer's Goal?**

“We need to greatly improved disaster, server, and application recovery processes,” says Anthony Ciske, IT Director for Fabrikam. “We’ve had some near-disasters in the past that were a real pain to recover from. We needed a real disaster recovery solution for our critical workloads that was compatible with our budget—and our staffing bandwidth.” The team had explored building a secondary data center and employing commercial disaster recovery solutions in the past, but both turned out to be too expensive for serious consideration.

### What Does the Customer Need?

The ability to perform data-center level recovery for critical workloads that can be executed in the event of a data center failure, with an automated and orderly recovery process so that different tiers of the application start in the correct order and remain in a consistent state without manual intervention.

• The ability to perform failback following restoring on-premises data center functionality that can be executed in the automated and orderly manner.

• The ability to perform multi-tier application and individual server-level recovery of critical workloads.

• Support for server-level and application-level high availability whenever possible.

• Quick testing and validation of recovery processes with minimal interruption to the production environment.

• Minimized capital and operational expenses.

• Optimized authentication for AD-integrated services and applications.

• Centralized management of backups and reduced or eliminated dependency on offsite tape storage.

• The level of security and privacy commensurate with highly sensitive and competitive nature of the business.

**What Things Worry the Customer?**

• Solution must significantly improve their current recovery point/time objectives (which today is a manual process).

• Overall cost of the solution.

• Protecting a diverse environment such as physical servers or other hypervisors.

• The management tools for the solution must be available in the event one of the data centers is unavailable.

• Protect data that is not hosted within a virtual hard disk (VHD/X).

• The protected data must be secure.

• Unsure about which workloads are supported on Azure.

### Case Study Solution

**Preferred Target Audience:** Anthony Ciske, IT Director – Fabrikam Publishing

**Network Administrator** – Fabrikam Publishing

**Application owners** (Exchange, SQL, n-tier applications)

**Preferred Solution**

Fabrikam Publishing decided to complete their deployment of SCVMM and implement two Azure recovery solutions – Azure Site Recovery to provide failover capability of their virtual servers and Azure Backup to replace their existing backup solution and to provide longer term protection of their virtual and physical servers. Azure Site Recovery has been configured to use Microsoft Azure as a disaster recovery site, with protection enabled for virtual machines hosting servers critical from the business continuity standpoint.

For disaster recovery, Fabrikam implemented two separate Azure virtual networks. The first network was for planned/unplanned failover of their server workloads. The second virtual network was configured for testing failover in a non-disruptive manner. As part of the network configuration they implemented site-to-site connectivity between the failover virtual network and the on-premises sites that needed protecting.  
   
Next, they deployed Active Directory in the failover virtual network and configured it to account for the local site to ensure localized authentication in recovery scenarios. In addition, DNS configuration on client systems as well as DNS settings on Azure virtual networks have been modified to ensure that name resolution continues to function during both test and planned/unplanned failover.

The final step for the disaster recovery solution was to address application level recovery on the application servers and high-availability concerns for Exchange 2013, SQL Server 2014, and other Active Directory-integrated applications.

Fabrikam implemented site recovery by configuring cloud recovery settings for the application servers.

Exchange 2013 is not currently supported within Azure IaaS. For an immediate cloud-based solution Exchange online is recommended.

For SQL Server deploying an AlwaysOn availability group on-premises with a replica in the failover network is the best solution. The replica should have async commit configured for replication.

Implementing separate recovery plans allows sequencing the failover of various tiers in a multi-tier application, hence, maintaining application consistency. What changes would you recommend to the existing SQL Server 2012 environment to facilitate high availability and recovery?

To address the data security concerns, encryption was enabled so any at rest data is automatically encrypted by Azure Site Recovery.

The second part of the preferred solution was to implement Azure Backup to remove the reliance on tape storage. Fabrikam created an Azure Backup Vault, downloaded and installed vault credentials on the servers that offload backups to tape and configured the Azure Backup Agent to protect the data in Azure instead of tape backup.

### Potential Benefits

Fabrikam Publishing is using Microsoft Azure Site Recovery to implement their Disaster Recovery and Business Continuity strategy. By leveraging built-in features of Azure Site Recovery, they are able to accomplish their recovery and resiliency objectives efficiently and with a minimal cost, giving them a competitive advantage.

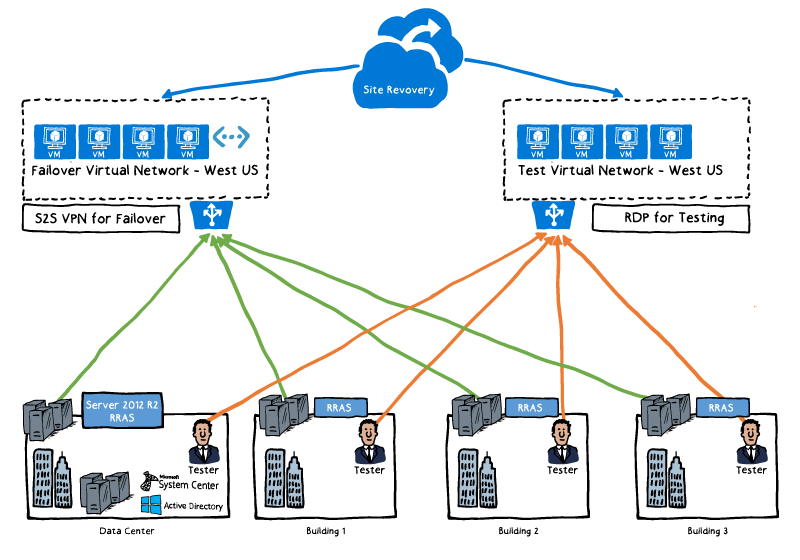
**• Recovery Objectives:** Azure Site Recovery with a recovery site in Azure offers the ability to perform planned and unplanned recovery and carry out testing with minimal disruption to the production environment. It allows for orchestrated, automated and orderly failover and failback processes, so different tiers of the application can start in the correct order and remain in a consistent state without manual intervention.

**• Cost Effective:** Azure Site Recovery eliminates capital expenses associated with implementing a secondary data center and provides predictable operational expenses. Azure Backup allows for centralized management of backups and eliminates the dependency on offsite tape storage.  
   
**• Security and Privacy:** Data replicated to Azure is encrypted during transit and, if desired, it can be encrypted at rest, while residing in the storage account.

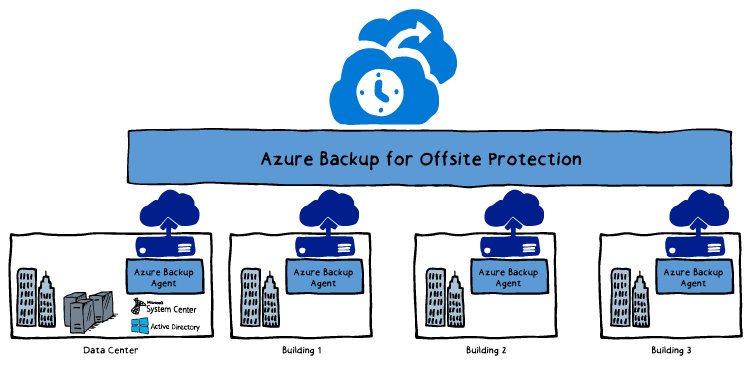
**• Non-Disruptive Testing of their DR Solution:** Site Recovery supports non-disruptive testing. Marquette can validate their disaster recovery solution as their environment changes without disrupting production.

### Architecture Example

Using Azure Site Recovery and multiple virtual networks for failover and non-disruptive failover testing:



ARCHITECTURE NETWORKING EXAMPLE  
Using Azure Backup to protect servers’ on-premises and store the data offsite in Azure:



   
ARCHITECTURE BACKUP EXAMPLE

### Checklist of Potential Benefits

Solution must significantly improve their current recovery point/time objectives (which today is a manual process).

• With Azure Site Recovery the copy frequency can be configured to as low as 30 seconds. You can also configure additional recovery points to automatically be taken (the default is every hour). Additional recovery points contain one or more snapshots that enable you to recover a snapshot of a virtual machine from an earlier point in time.

**Overall cost of the solution**

• Protecting their servers by using Azure as a failover data center is significantly less expensive than implementing a failover solution with secondary data center and hardware.

**Protecting a diverse environment such as physical servers or other hypervisors**

• Currently, the Site Recovery service only allows replication to Azure for virtual machines based on Microsoft Hyper-V. Site Recovery can protect VMWare as well but only for on-premises to on-premises scenarios.

**Protect data that is not hosted within a virtual hard disk (VHD/X)**

• Azure Site Recovery works with Hyper-V based virtual machines and their attached virtual hard disks. In the scenario of restoring to Azure, the source virtual machines can be VHDX. In Azure they will be in the VHD format.

**The management tools for the solution must be available if one of the data centers is unavailable.**

• Azure Site Recovery is hosted in the cloud so in the event that one of the data centers is down the solution is still available to monitoring and to recover the servers.

**Security of protected data**

• Data replicated using Hyper-V Replica/Azure Site Recovery is encrypted while in transit.

• When replicating virtual machines to Azure replicated data can be encrypted even while at rest.

**Supported Workloads**

• Not all workloads are supported (or fully supported) with Hyper-V replica and Site Recovery.

• We are currently working with the various teams for key workloads such as SharePoint, SQL and Exchange to enable full support for these workloads in Hyper-V replica and site recovery.

### Proof of Concept Checklist

**Objectives:**

* Identify service recovery point/time objectives for workloads in proof of concept.
* Identity manual steps needed for a recovery in secondary data center and script those steps as part of a recovery plan to ensure a fully automated recovery.
* Demonstrate that Azure Site Recovery and Hyper-V replication can replicate and recover virtual machines in the secondary data center within the RPO/RTO objectives.
* Enable backup protection to offsite storage.

**Flow/Scope of the proof of concept (a list of 2-3 bullets):**

Identify services (Virtual Machines) to configure for replication.  
Implement two separate Azure virtual networks:

* + One for planned/unplanned failover.
  + The other for testing application and sever-level recovery.
  + Configure network connectivity between the on-premises environment and the failover virtual network.

Create VMM cloud in the data center (if it does not exist).  
Configure Active Directory and DNS.  
Configure Azure Site Recovery with on-premises to Azure:

* + Create an Azure Site Recovery Vault.
  + Install the Provider application on the VMM server.
  + If you don't have a storage account, create one.
  + Install the Microsoft Azure Recovery Services agent on each Hyper-V host located in VMM clouds you want to protect.
  + Configure cloud protection settings for VMM cloud.
  + Configure network mapping to map source VM network to target Azure network.
  + Enable protection for virtual machines located in protected VMM clouds.
  + Failover using the test failover method.

Configure Azure Backup and protect on-premises server backups.

**Conditions of satisfaction / success criteria for the proof of concept.**

* Demonstrate that Azure Site Recovery does indeed fulfill the customer's recovery requirements of protecting the entire data center.

**Resources / Bill of Materials that you would use:**

* Online Documentation
* System Center 2012 R2 and Windows Server 2012 R2
* Azure Site Recovery and Azure Storage
* Virtual Private Networks
* Partner / MCS

### Lab Steps

#### Online Lab: Deploying Configuration Management solutions to Azure

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy compute resources

##### Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab.
   * In the **Resource group** section, select the **Create New** option and then, in the text box, type **AADesignLab1201-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 3: Deploy a Linux VM

1. At the top of the portal, click the **Cloud Shell** icon to open a new Clould Shell instance.
2. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
3. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T02\Module\_03\LabFiles\Starter\** folder, select the **linux-template.json** file, and click **Open**. The file contains the following template:

{ "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": { "userName": { "type": "string", "defaultValue": "Student" }, "password": { "type": "securestring" } }, "variables": { "vmName": "[concat('lvm', uniqueString(resourceGroup().id))]", "nicName": "[concat('nic', uniqueString(resourceGroup().id))]", "publicIPAddressName": "[concat('pip', uniqueString(resourceGroup().id))]", "virtualNetworkName": "[concat('vnt', uniqueString(resourceGroup().id))]", "subnetName": "Linux", "imageReference": { "publisher": "suse", "offer": "opensuse-leap", "sku": "42.3", "version": "latest" } }, "resources": [ { "apiVersion": "2017-06-01", "type": "Microsoft.Network/publicIPAddresses", "name": "[variables('publicIPAddressName')]", "location": "[resourceGroup().location]", "properties": { "publicIPAllocationMethod": "Dynamic" } }, { "apiVersion": "2017-06-01", "type": "Microsoft.Network/virtualNetworks", "name": "[variables('virtualNetworkName')]", "location": "[resourceGroup().location]", "properties": { "addressSpace": { "addressPrefixes": [ "10.0.0.0/16" ] }, "subnets": [ { "name": "[variables('subnetName')]", "properties": { "addressPrefix": "10.0.0.0/24" } } ] } }, { "apiVersion": "2017-10-01", "type": "Microsoft.Network/networkInterfaces", "name": "[variables('nicName')]", "location": "[resourceGroup().location]", "dependsOn": [ "[resourceId('Microsoft.Network/publicIPAddresses/', variables('publicIPAddressName'))]", "[resourceId('Microsoft.Network/virtualNetworks/', variables('virtualNetworkName'))]" ], "properties": { "ipConfigurations": [ { "name": "ipconfig1", "properties": { "privateIPAllocationMethod": "Dynamic", "publicIPAddress": { "id": "[resourceId('Microsoft.Network/publicIPAddresses', variables('publicIPAddressName'))]" }, "subnet": { "id": "[concat(resourceId('Microsoft.Network/virtualNetworks',variables('virtualNetworkName')), '/subnets/', variables('subnetName'))]" } } } ] } }, { "apiVersion": "2017-03-30", "type": "Microsoft.Compute/virtualMachines", "name": "[variables('vmName')]", "location": "[resourceGroup().location]", "dependsOn": [ "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]" ], "properties": { "hardwareProfile": { "vmSize": "Standard\_A1\_v2" }, "osProfile": { "computerName": "[variables('vmName')]", "adminUsername": "[parameters('username')]", "adminPassword": "[parameters('password')]" }, "storageProfile": { "imageReference": "[variables('imageReference')]", "osDisk": { "createOption": "FromImage" } }, "networkProfile": { "networkInterfaces": [ { "id": "[resourceId('Microsoft.Network/networkInterfaces', variables('nicName'))]" } ] } } } ] }

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub virtual network:

RESOURCE\_GROUP='AADesignLab1202-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder <Azure region> with the name of the Azure region to which you intend to deploy resources in this lab):

LOCATION='<Azure region>'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

az group create --name $RESOURCE\_GROUP --location $LOCATION

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the Azure Resource Manager template with the specified parameters file:

az group deployment create --resource-group $RESOURCE\_GROUP --template-file ~/linux-template.json --parameters password=Pa55w.rd1234

1. Do not wait for the deployment to complete before you proceed to the next task.

##### Task 4: Deploy an Azure Automation account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Automation** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Automation**.
4. On the **Automation** blade, click **Create**.
5. On the **Add Automation Account** blade, perform the following tasks:
   * In the **Name** text box, type **LinuxAutomation**.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Create new** option and then, in the text box, type **AADesignLab1203-RG**.
   * In the **Location** drop-down list, select the Azure region matching or near the location where you deployed the Azure VM in the previous task.
   * In the **Create Azure Run As account** section, ensure that **Yes** option is selected.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next task.

**Review**: In this exercise, you created a Linux VM using an Azure Resource Manager template and provisioned an Azure Automation account from the Azure portal.

#### Exercise 2: Configure Azure Automation DSC

##### Task 1: Import Linux PowerShell DSC modules

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1203-RG**.
3. On the **AADesignLab1203-RG** blade, click the newly created Azure Automation account.
4. On the **LinuxAutomation** blade, in the **SHARED RESOURCES** section on the left side of the blade, click **Modules gallery**.
5. On the **LinuxAutomation - Modules gallery** blade, perform the following tasks:
   * In the **Search** text box, type **nx** and press **Enter**.
   * In the search results, click the **nx** module.
6. On the **nx** blade, click the **Import** button at the top of the blade.
7. On the **Import** blade, click the **OK** button.
8. Wait for the import process to finish before you proceed to the next task. A status message on the **nx Module** blade will indicate that the module was successfully imported.**Note**: This process shoudl take about 2 minutes.

##### Task 2: Create Linux DSC Configuration

1. Navigate back to the **LinuxAutomation** blade.
2. Back on the **LinuxAutomation** blade, in the **CONFIGURATION MANAGEMENT** section, click **State configuration (DSC)**.
3. On the **LinuxAutomation - State configuration (DSC)** blade, click the **Configurations** link.
4. On the **LinuxAutomation - State configuration (DSC)** blade, click the **+ Add** button at the top of the pane.
5. On the **Import** blade, perform the following tasks:
   * Next to the **Configuration file** field, click the blue button with a folder icon.
   * In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T02\Module\_02\LabFiles\Starter\** folder.
   * Select the **lampserver.ps1** file.
   * Click the **Open** button to close the dialog and return to the **Import** blade.
   * In the **Name** text box, accept the default entry **lampserver**.
   * In the **Description** text box, type **LAMP Server configuration using PHP and MySQL**.
   * Click the **OK** button.
6. Back in the **DSC configurations** pane, click **Refresh** and then click the newly created **lampserver** configuration.
7. On the **lampserver Configuration** blade, click the **Compile** button at the top of the blade. In the confirmation dialog box, click **Yes** to proceed with compiling the configuration.
8. Wait for the compilation task to finish. To determine the status of the compilation task, review the **STATUS** column of the **Compilation jobs** section of the **lampserver Configuration** blade.**Note**: You may need to close and re-open the blade to see the latest compilation status. This blade does not refresh automatically.

##### Task 3: Onboard Linux VM

1. Navigate back to the **LinuxAutomation - State Configuration (DSC)** blade.
2. Back on the **LinuxAutomation - State Configuration (DSC)** blade, click the **Nodes** link.
3. On the **LinuxAutomation - State configuration (DSC)** blade, click the **+ Add** button at the top of the pane.
4. On the **Virtual Machines** blade, click the entry representing the Linux virtual machine you deployed in the previous exercise.
5. On the virtual machine blade, click **+ Connect**.
6. On the **Registration** blade, perform the following tasks:
   * Leave the **Registration key** setting with its default value.
   * In the **Node configuration name** drop-down list, select the **lampserver.localhost** entry.
   * Leave all remaining settings with their default values.
   * Click the **OK** button.
7. Wait for the connection process to complete before you proceed to the next step.
8. Navigate back to the **LinuxAutomation - State Configuration (DSC)** blade.
9. Back on the **LinuxAutomation - State Configuration (DSC)** blade, click the **Refresh** button.
10. In the list of DSC nodes, verify that the Linxu virtual machine has the **Compliant** status.

**Review**: In this exercise, you created a PowerShell DSC configuration and applied the configuration to a Linux virtual machine.

#### Exercise 3: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab12')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab12')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 3 Review Questions

**Azure Network Watcher**  
You are designing a solution to monitor Azure-based networking resources.  
You need to view the network topology, and relationships between the resources.  
What Azure service should you recommend? What are the benefits and limitations of the service.

Suggested Answer ↓

Azure Network Watcher allows you to view your Azure network topology and the relationships between the resources. As an example, a virtual network contains subnets. Subnets contain resources, such as Azure Virtual Machines (VMs). VMs have one or more network interfaces. Each subnet can have a network security group and a route table associated to it. The topology capability of Azure Network Watcher enables you to view all of these resources in the virtual network and their relationships in a visual diagram.

**Azure Application Insights**  
  
You are designing a solution that includes an Azure web app. The solution must detect and report performance anomalies in real time.  
  
What Azure service should you recommend? What additional information can you collect?

Suggested Answer ↓

Azure Application Insights includes powerful analytics tools to help you diagnose issues and to understand what users do with your app. It is designed to help you continuously improve performance and usability.

Azure Application Insights works for apps on a wide variety of platforms including .NET, Node.js and J2EE. This includes apps that are hosted on-premises or in the cloud.

**Azure Advisor**  
  
A company hires you to audit Azure resources. You must optimize resource usage and reduce costs. You must also examine security, availability, and performance of applications.  
  
What Azure service should you use? What are the limitations of the service?

Suggested Answer ↓

Azure Advisor displays personalized recommendations for all your subscriptions. You can apply filters to display recommendations for specific subscriptions and resource types. The recommendations are across four categories that include high availability, security, performance and cost. Azure Advisor provides proactive, ac

AZ-301T03-A - Designing for Deployment, Migration, and Integration

### Welcome to Designing for Deployment, Migration, and Integration

Welcome to Designing for Deployment, Migration, and Integration. This course is part of a series of four courses to help students prepare for Microsoft’s Azure Solutions Architect technical certification exam AZ-301: Microsoft Azure Architect Design. These courses are designed for IT professionals and developers with experience and knowledge across various aspects of IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data management, budgeting, and governance.  
  
This course contains the following three modules:  
**Module 1** - Deploying Resources with Azure Resource Manager  
This module establishes a basic understanding of Azure Resource Manager and the core concepts of deployments, resources, templates, resource groups, and tags. The module will dive deeply into the automated deployment of resources using ARM templates.  
After completing this module, students will be able to:

•    Create a resource group.  
•    Add resources to a resource group.  
•    Deploy an ARM template to a resource group.  
•    Filter resources using tags.  
•    Author a complex deployment using the Azure Building Blocks tools.

**Module 2** - Creating Managed Server Applications in Azure This module describes how solutions can leverage serverless application hosting services in Azure to  
host web applications, REST APIs, integration workflows and HPC workloads without the requirement to manage specific server resources. The module focuses on App Services-related components such as Web Apps, API Apps, Mobile Apps, Logic Apps, and Functions.  
After completing this module, students will be able to:

•    Select between hosting application code or containers in an App Service instance.  
•    Describe the differences between API, Mobile, and Web Apps.

•    Integrate an API or Logic App with the API Management service.  
•    Design an App Service Plan or multi-region deployment for high performance and scale.

**Module 3** - Authoring Serverless Applications in Azure  
This module describes how solutions can leverage serverless application hosting services in Azure to host web applications, REST APIs, integration workflows and HPC workloads without the requirement to manage specific server resources. The module focuses on App Services-related components such as Web Apps, API Apps, Mobile Apps, Logic Apps, and Functions.  
After completing this module, students will be able to:

•    Select between hosting application code or containers in an App Service instance.  
•    Describe the differences between API, Mobile, and Web Apps.  
•    Integrate an API or Logic App with the API Management service.  
•    Design an App Service Plan or multi-region deployment for high performance and scale.  
•    Create a resource group.  
•    Add resources to a resource group.  
•    Deploy an ARM template to a resource group Integrate an API or Logic App with the API Management service.  
•    Design an App Service Plan or multi-region deployment for high performance and scale.  
•    Integrate an API or Logic App with the API Management service.  
•    Design an App Service Plan or multi-region deployment for high performance and scale.

**Prerequisites**  
This course requires that students have the following knowledge and skills:

• Create resources and resource group in Azure.  
• Manage users, groups, and subscriptions in an Azure Active Directory instance.  
• Build an Azure Virtual Machine with related resources.  
• Manage containers and blobs stored in an Azure Storage account.  
• Create App Service Plans and manage apps related to the plan.  
• Configure an Azure Virtual Network and enable S2S and P2S connectivity.  
• Protect networked application components using Network Security Groups.  
• Automate everyday Azure resource tasks using Azure CLI or Azure PowerShell.  
• Deploy an Azure SQL, MySQL, Postgres or Cosmos database instance.  
• Monitor existing Azure solutions using built-in metrics, Application Insights, or Operational Insights.

### Azure Resource Manager

This module establishes a basic understanding of Azure Resource Manager and the core concepts of deployments, resources, templates, resource groups, and tags. The module will dive deeply into the automated deployment of resources using ARM templates.

After completing this module, students will be able to:

• Create a resource group

• Add resources to a resource group

• Deploy an ARM template to a resource group

• Filter resources using tags

• Author a complex deployment using the Azure Building Blocks tools

**ARM Templates**

Azure has developed a great deal over the last few years, and with new services being released on a regular basis, there was a need to create a way to manage and deploy resources in a componentized and reusable manner. Azure Resource Manager was designed to represent each service in Azure as a resource provider and each service instance in Azure as a modular resource. With Azure Resource Manager, JSON templates were used to deploy collections of resources using Infrastructure-as-Code concepts. Along with Azure Resource Manager, we saw that release of a new Azure Portal ([https://portal.azure.com](https://portal.azure.com/)) that focused on the modular nature of resources. This lesson focuses on Azure Resource Manager (ARM) and how you can use JSON to deploy resources using ARM templates.

After completing this section you will be able to:

• Describe ARM Templates

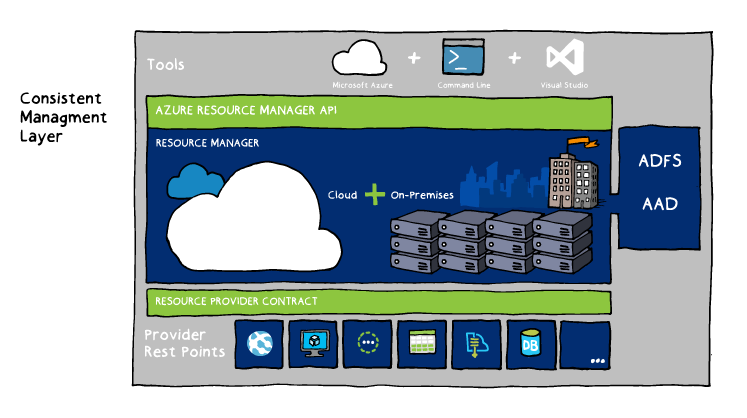
• Understand the JSON format and how to author a JSON template in various ways

• Describe the format of a JSON file and where to obtain example templates for Azure deployments

• Deploy a resource using the Azure Quickstart templates on GitHub

Azure Resource Manager (ARM) is the latest way to conceptualize your Azure resources in your subscription. Service instances are now resources, which are grouped as resource groups. Resource groups provide a common lifecycle for the child resources. They can be created, managed, monitored, or deleted together. The Resource Manager also offers the concept of resource group templates which enable you to define a service unit in advance, and then use the template to create as many resource groups as you need.

Resource groups and resource group templates are ideal for developer operations scenarios where you need to quickly build out development, test, quality assurance, or production environments that are homogenous in nature and can be managed with a shared lifecycle. Developers can quickly delete their environment and create a new environment by using the shared template. The resource groups can be monitored to determine the billing rate or resource usage at a higher level than monitoring individual service instances.



Before using ARM, you need to consider resource providers and ensure your subscription contains registrations for each provider you wish to use. Deploying a complex Virtual Machine template will fail at the first hurdle if you are not registered for the **Microsoft. Compute** provider. This provider controls access to all Virtual Machine creation.

### Azure Resource Manager Objects

When you envision your solution using ARM, you must start by designing and conceptualizing your entire solution considering all components that may compose your solution. Once you have designed your entire solution, you can then identify individual units of functionality and find resources available on Azure that can facilitate the specific functionalities.

You use a template—a resource model of the service—to create a resource group with the resources you specified above. After you author the template, you can manage and deploy that entire resource group as a single logical unit. There are three primary concepts in Resource Manager:

• **Resource**: A resource is merely a single service instance in Azure. Most services in Azure have a direct representation as a resource. For example, a Web App instance is a resource. An App Service Plan is also a resource. Even a SQL Database instance is a resource.

•**Resource Group**: A resource group is a group of resources in the logical sense. For example, a Resource Group composed of a Network Interface Card (NIC), a Virtual Machine compute allocation, a Virtual Network, and a Public IP Address creates what we would logically consider a “Virtual Machine.”

• **Resource Group Template**: Every resource group deployment is completed using a JSON file known as the resource group template. This JSON file declaratively describes a set of resources. The deployment adds the new resources to a new or existing resource group. For example, a template could contain the configuration necessary to create 2 API App instances, a Mobile App instance and a Cosmos DB instance.

### Interacting with Resource Manager

You can use Resource Manager in a variety of different ways including:

• **PowerShell**: There are PowerShell CmdLets already available to allow you to manage your services in the context of resources and resource groups.

•**Cross-Platform Command-Line Interface:** This CLI allows you to manage your Azure resources from many different operating systems.

• **Client Libraries**: There are client libraries already available for various programming frameworks/languages to create resources and resource groups in Azure.

• **Visual Studio**: Visual Studio 2015 ships with a Resource Manager project type that allows you to create a resource group template by either manually modifying JSON (with schema intellisense) or use scaffolding to update your JSON template automatically.

• **Portal template deployment**: In the portal, you can use the Template Deployment option in the Marketplace to deploy a Resource Group from a template.

• **REST API**: All the above options use the REST API to create your resources and resource groups. If you prefer to create resources without a library, you can always use the REST API directly.

     **Reference Link**: <https://docs.microsoft.com/rest/api/resources/>

     **Azure Resource Manager Documentation**: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/>

### ARM Templates

Azure supports the deployment of resources using Azure Resource Manager templates. In an Azure Resource Manager template, you specify the resource—for example, a virtual network, virtual machine, and so on—using a JSON object. Some or all of the properties of the resource can be parameterized so that you can customize your deployment by providing parameter values at deployment time.

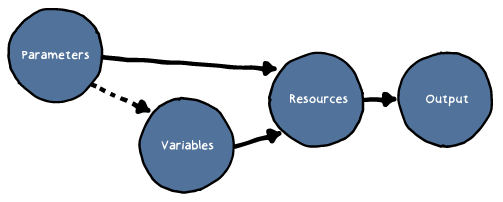
* + ARM Templates are deployed in a few ways. These depend on your aims, the result intended and your chosen method for development.
  + A developer may choose to use Visual Studio to create and deploy ARM templates directly and to manage the lifecycle of the resources through Visual Studio.
  + An administrator may choose to use PowerShell or the Azure Command Line to deploy resources and amend them. An end user without command line or developer skills would choose to use the Azure Portal to deploy resources without realizing a template is involved. This deployment would typically be a less complicated group of resources unless the user chooses to deploy a marketplace offering.

**Advantages Of Using Templates**

Templates are preferred to manually deploying resources for the following:

* + **Ensure idempotency**: If you deploy an identical template to multiple resource groups, they would functionally be the same.
  + **Simplify orchestration**: only need to deploy the template to deploy all of your resources. Normally this would take multiple operations.
  + **Configure multiple resources**: You can configure multiple resources simultaneously and use variables/parameters /functions to create dependencies between resources. For example you can require that a VM is created before a Web App because you need the VM's public IP address for one of the Web App's settings. Additionally, you can require a Storage account is created before a VM so that you can place the VHDs in that storage account.
  + **Parameterize**: You can parameterize input and output values so they can be reused across many different scenarios. Templates can also be nested so you can reuse smaller templates as part of a larger orchestration.

**Template Resources**



### JSON

#### What is JSON?

JavaScript Object Notification (JSON) is a lightweight formatted script designed for data transfer. Azure Resource Manager uses this as the foundation for every resource.

**Empty ARM Template**

{  
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",  
  "contentVersion": "1.0.0.0",  
  "parameters": {  
  },  
  "variables": {  
  },  
  "resources": [  
  ],  
  "outputs": {  
  }  
}

The format of JSON is flexible in that several elements are optional.

The script must have a schema and a content section as well as a resource section. The other three, parameters, variables and output are optional.  
In day to day use, most ARM Template JSON contains all sections and even contains logic as well.

Resources can depend on other resources existing and even their names, locations and content. This resource dependency can lead to very complex nested JSON templates when building a sophisticated resource group, such as a SharePoint farm.

The Azure Quickstart templates on Github are an excellent resource for learning to structure and deploy your templates correctly. If you are just beginning on your JSON journey, then the Automation script element of every object in the Azure Portal allows you to learn the constructs and formatting of JSON.

### Role-Based Access Control Role-Based Access Control (RBAC)

Azure role-based access control (RBAC) leverages existing Azure AD user, groups, and services to create a relationship between those identities and components of your Azure subscription. Using RBAC, you can assign roles to existing Azure AD identities that grants them pre-determined levels of access to an Azure subscription, resource group or individual resource.

This lesson will discuss the best practices and possibilities of using Azure AD RBAC to manage access to your resources, resource groups, and application.

After completing this section, you will be able to:

• Describe Azure AD RBAC features.

• Decide how to provide access to your resources using RBAC.

• Allocate Roles to users and groups to provide access to resources.

• Deploy a custom role to an Azure resource group.

Azure role-based access control allows you to grant appropriate access to Azure AD users, groups, and services, by assigning roles to them on a subscription or resource group or individual resource level. The assigned role defines the level of access that the users, groups, or services have on the Azure resource.

**Roles**

A role is a collection of actions that can be performed on Azure resources. A user or a service is allowed to act on an Azure resource if they have been assigned a role that contains that action. There are built-in roles that include (but is not limited to):

| **ROLE NAME** | **DESCRIPTION** |
| --- | --- |
| Contributor | Contributors can manage everything except access. |
| Owner | Owner can manage everything, including access. |
| Reader | Readers can view everything, but can't make changes. |
| User Access Administrator | Allows you to manage user access to Azure resources. |
| Virtual Machine Contributor | Allows you to manage virtual machines, but not access to them, and  not the virtual network or storage account they are connected to. |

### Role Assignment

A role assignment can be created that associates a security principal to a role. The role is further used to grant access to a resource scope. This decoupling allows you to specify that a specific role has access to a resource in your subscription and add/remove security principals from that role in a loosely connected manner. Roles can be assigned to the following types of Azure AD security principals:

• **Users**: Roles can be assigned to organizational users that are in the  
Azure AD with which the Azure subscription is associated. Roles can also be assigned to external Microsoft accounts that exist in the same directory.

• **Groups**: Roles can be assigned to Azure AD security groups. A user is automatically granted access to a resource if the user becomes a member of a group that has access. The user also automatically loses access to the resource after getting removed from the group. Managing access via groups by assigning roles to groups and adding users to those groups is the best practice, instead of assigning roles directly to users.

• **Service principals**: Service identities are represented as service principals in the directory. They authenticate with Azure AD and securely communicate with one another. Services can be granted access to Azure resources by assigning roles via the Azure module for Windows PowerShell to the Azure AD service principal representing that service.

### Resource Scope

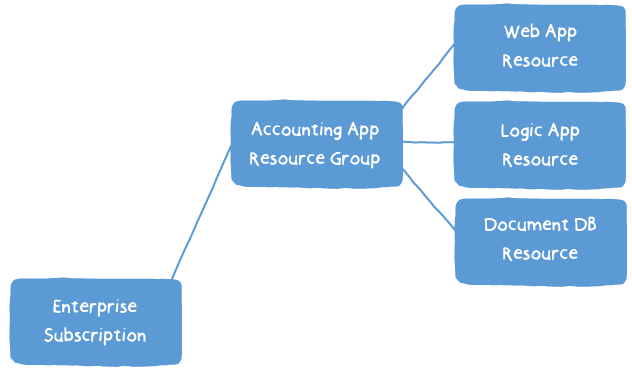
Access does not need to be granted to the entire subscription. Roles can also be assigned to resource groups as well as for individual resources. In Azure RBAC, a resource inherits role assignments from its parent resources. So if a user, group, or service is granted access to only a resource group within a subscription, they will be able to access only that resource group and resources within it, and not the other resources groups within the subscription. As another example, a security group can be added to the Reader role for a resource group, but be added to the Contributor role for a database within that resource group.



### Scoping to Resource Groups

While RBAC can easily be used to grant access to an individual resource, it is preferable to grant access to an entire Resource Group as opposed to individual resources. By scoping to a resource group, you can add/remove and modify resources quickly without having to recreate assignments and scopes. You can also give an individual owner or contributor access to a resource group so that they can create, recreate or destroy resources on their own without requiring involvement from the account administrator.

Let’s explore a resource group for an actual application. In the example below, there is a resource group for an accounting application. this resource group contains multiple resources that are used in the application.



By granting an individual owner or contributor access to the resource group, they can configure resources, create new deployments, add new resources or create automation scripts at will without requiring additional administrator assistance or having access to resources in other resource groups.

**Custom Roles**

If there is no role suitable for your purposes or granular enough to suit your needs, it is possible to create a custom role and apply that. To create a custom role, you must use either Azure PowerShell or the Azure Command-Line Interface.

It is also possible to use the REST API to create roles programmatically. Each Azure AD tenant is limited to 2000 custom roles.

To create a new custom role you run the **New-AzureRmRoleDefinition** cmdlet; you can pass a JSON template to the cmdLet or use **PSRoleDefinitionObject**.  
   
**JSON required for a new Custom Role**

{  
  "Name": "New Role 1",  
  "Id": null,  
  "IsCustom": true,  
  "Description": "Allows for read access to Azure storage and compute resources",  
  "Actions": [  
    "Microsoft.Compute/\*/read",  
    "Microsoft.Storage/\*/read",  
     
  ],  
  "NotActions": [  
  ],  
  "AssignableScopes": [  
    "/subscriptions/c489345-9cd4-44c9-99a7-4gh6575315336g"  
 ]  
}

If you save the JSON to a file such as C:\CustomRole\newrole1.json, you can use the following PowerShell to add the custom role to a subscription.

**Command to add the Custom role to the Subscription in PowerShell**

New-AzureRmRoleDefinition -InputFile "C:\CustomRole\newrole1.json"

### Azure Resource Policies

The use of RBAC controls users access to resources. Resource Policy is a preview service that controls which resources can be created and what for they take, such as naming conventions, locations, and sizes. This lesson will describe the features available and show the ways of defining, assigning and monitoring those policies.

After completing this section, you will be able to:

• Describe Azure Policy service.

• Decide when to apply policy definitions and to what scope.

• Author an Azure Policy definition.

• Assign and monitor an Azure resource policy.

**Azure Resource Policies**

Azure Policy is a new service designed to allow an organization to ensure that resources created in the cloud comply with corporate standards and service level agreements. As an example, preventing users from creating resources in specific locations and of specific types and sizes. If your organization has no need for a Virtual Machine with 32 CPU cores and 500GB of RAM in the more expensive UK South region, then this can be monitored and prevented with Azure Policy.

An Azure policy contains two elements, a policy definition, and a policy assignment. This design allows an organization to create a library of policy definitions to be assigned later. There are many pre-defined policy definitions built into Azure.

### Policy vs. RBAC

There are differences between policy and role-based access control (RABC). RBAC controls user access, permissions, privileges, and actions at different scopes. As an example, you could add a user to the Owner role for a resource group as the desired scope. The Owner role gives you full control of that resource group.  
   
Policy evaluates resource properties for already existing resources and during deployment. As an example, using policies, you can control which type of resource is available for deployment in your organization. Other policies can limit the locations in which you deploy resources or require your resources to follow naming conventions.

Azure policy is a default allow, and explicit deny system. This is not the same as RBAC.  
To be able to create and assign Azure policies, you need to be assigned permissions using RBAC; the contributor role does not contain the necessary permissions.

The permissions required are:

• **Microsoft.Authorization/policydefinitions/write** permission to define a policy.

• **Microsoft.Authorization/policyassignments/write** permission to assign a policy.

### Built-In Policies

Azure has a built-in library of Policies to allow organizations to control their resources. These include:

• Allowed locations

• Allowed resource types

• Allowed storage account SKUs

• Allowed virtual machine SKUs

• Apply tag and default value

• Enforce tag and value

• Not allowed resource types

• Require SQL Server version 12.0

• Require storage account encryption

The inclusion of these built-in policy definitions limits the number a user is required to create to manage their subscription efficiently.

### Policy Definition

Every policy definition contains a minimum of two elements, first the conditions under which it is enforced. Secondly, it has an action that triggers if the conditions are met.

Policy definitions are created using JSON. A policy definition contains elements to define:

**• mode**

**• parameters**

**• display name**

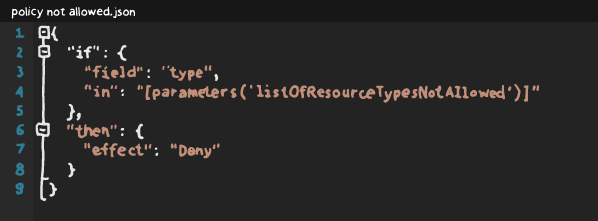
**• description**

**• policy rule**

**• logical evaluation**

**• effect**

The following example shows a policy that defines not allowed resources (This is a built-in policy definition).



During assignment, the list of resource types not allowed parameter is populated and evaluated against all current resources in scope for the assignments. Any non-compliant resources will be identified. When creating new resources, the deployment will fail if they are non-compliant. The above is a straightforward policy definition. The JSON for policy definitions can be much more complicated.

The definition can contain multiple if, then blocks and combines logical operators, conditions, and fields to build out the policy. In addition, the policy can define alternative effects.

The available effects are:

**• Deny**

**• Audit**

**• Append**

**• AuditIfNotExists**

**• DeployIfNotExists**

You can assign any of these policy definitions through the Azure portal, PowerShell, or Azure CLI.

### Policy Assignment

Once a policy definition has been created, this is known as a custom policy definition and can be assigned to take effect over a specific scope. This scope could range from several subscriptions within an enterprise (known as a management group) to a resource group. A scope is the list of all the resource groups, subscriptions, or management groups that the policy definition is assigned to.  
   
If a policy is assigned to a resource group, the same policy applies to all the resources contained in that group. Policy assignments are inherited by all child resources. This can be altered or prevented using the exclusion option at the time of assignment. As an example, a policy assignment could prevent the creation of SQL databases within your subscription. An exclusion could allow this in a single Resource group, which you want to keep as your database Resource Group. RBAC is then used to restrict access to the SQL database resource group to the trusted group of users that have the necessary skills and requirements for these resources.

A policy definition will create parameters to be applied at the time of Policy assignment. This allows for one definition to be re-used across several resource groups, locations, and subscriptions.

The scope, exclusions, and parameters of the Not Allowed resource types are shown in the policy assignments blade when assigning a definition.  
Azure policies can be assigned using the Azure Portal, Azure PowerShell, and the Azure Command Line Interface.

**Initiative Definition**

An initiative definition is a collection of policy definitions that are designed to achieve a design goal. Initiative definitions contain policy definitions and can be assigned as a group to simplify the process of achieving that goal. The definition appears as a single item rather than multiple definitions.

### Policies for Naming Conventions

Azure policies can be used to maintain control on the naming conventions for all resources within your subscriptions.  
A policy definition can be created to enforce naming conventions. These can use wildcards, patterns tags, and multiple patterns to apply restrictions to your Azure resource names.

The example below applies a pattern match asserting that is the resource name does not begin Contoso and have six characters as the rest of the name then it will be non-compliant.

**Example Naming Pattern**

{  
  "if": {  
    "not": {  
      "field": "name",  
      "match": "contoso??????"  
    }  
  },  
  "then": {  
    "effect": "deny"  
  }  
}

The effect of non-compliance is deployment failure on the creation of a resource or listing in the non-compliant reporting if the resource is already in existence.

### Azure Key Vault

When deploying resources using Arm templates and automating that deployment, it is best practice to use a Service Principal, the Azure equivalent of an Active Directory Service Account. This removes the risk whereby an administrator account is stored and used to deploy resources. To facilitate this in a secure manner, Azure Resource Manager can use the Azure Key Vault to store the service principal secrets. This lesson describes the Key Vault and its use in deploying Arm templates securely.

After completing this section you will be able to:

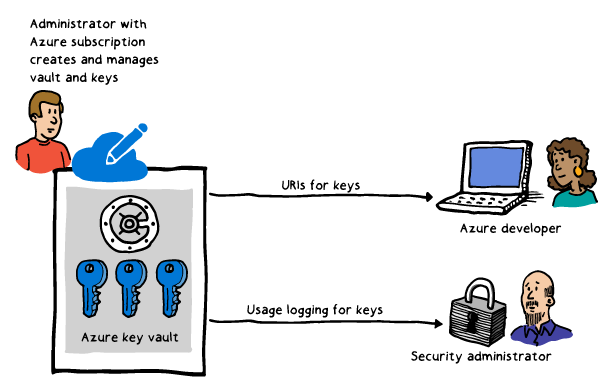
• Describe the Azure Key Vault service.

• Describe the secure deployment of templates using Service Principals and the Key Vault.

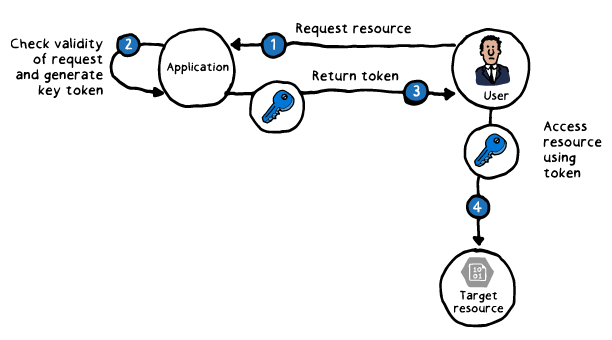
• Create a Service Principal.

• Deploy a resource ARM templates and the Azure Key Vault.

Azure Key Vault is a two-tier service that allows secure storage of cryptographic keys, certificates, and secrets. These can be used by applications, services, and users. The premium tier allows storage of these secrets in a Hardware Security Module, a physical device to contain all your secrets.



Key Vault makes the process of creation, importing, maintaining and recycling secrets much easier. The developer can easily create dev and test keys that can be migrated to production use at deployment. Security administrators can manage all the secrets in the Key Vault with ease and grant and revoke access when required. Key Vault can be used for several scenarios in Azure.



App developers want to use keys for signing and encryption but want all keys outside of the application. App developers do not want to manage keys but wants their customers to bring their own keys and certificates.

Security admins want to ensure that company applications comply with regulatory bodies requirements such as FIPS. Key Vaults can be created in any Azure subscription by any contributor or Owner. Key Vaults can be used to create, manage and import secrets, keys, and certificates for applications, services and users.

### Key Vault Use in ARM Templates

When deploying resources into Azure using ARM Templates, a user credential is often required to allow the resources to be created. Best practices dictate that embedding credentials and passwords inside a template are unwise. Key Svault can be used to pass a secure value as a parameter during deployment. The secure value, in this instance a password can be retrieved from the Key Vault, retrieval is accomplished by referencing both the key vault id and the secret in the parameter file for the template. The benefit is that the value of this secret is never exposed. To further secure the deployment, it is advised to create an Azure Service Principal, which is the equivalent of an Active Directory Service Account. This reduces risk by allowing you to limit the privileges of the Service Principal to only what is needed.

The value is never exposed because you only reference its key vault ID. At deployment time you are not required to enter credentials. There are several steps required to allow a Key Vault to be used during template deployment. First, it is essential to set the **enabledForTemplateDeployment** property to **true** when you create the Key Vault. This setting allows access to the key Vault from Resource manager at deployment time.

Next, you must create a secret using the Azure Portal, PowerShell or Azure CLI. Having created a secret for use by the template you must give the template access to the vault and the secret. This is achieved by ensuring the service principal, user or template has the **Microsoft.KeyVault/vaults/deploy/action** permission for the correct Key Vault. The Contributor built-in role already has this permission. The last step is to reference the secret using a static ID in the template parameter file. Sometimes the Key Vault secret will change from deployment to deployment; this requires the use of a dynamic ID reference which cannot go in the parameter file and calls for a nested template to achieve this.

### Azure Building Blocks

While authoring ARM templates can give you the most flexibility when automating deployments, ARM templates can become very complex in a short amount of time. Additionally, it may be difficult to ensure that Microsoft best practices for Azure infrastructure are reflected in every template authored by your team.

This section introduces the Azure Building Blocks, an open-source tool and repository to help simplify the authoring and deployment of ARM Templates. This tool includes ARM templates that already reflect best practices as prescribed by the Patterns & Practices team at Microsoft.

After completing this section you will be able to:

• Describe the Azure Building Blocks and how they fit into a deployment automation solution.

• Decide when to author an ARM template manually or when to use the Azure Building Blocks.

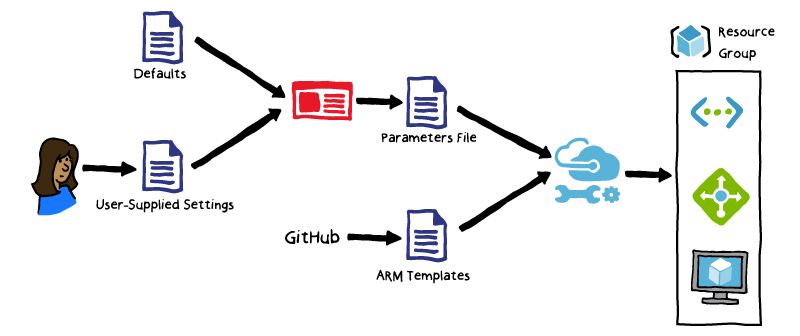
• Author an Azure Building Blocks parameters file.

• Deploy a resource using the Azure Building Blocks command-line tool.

Azure supports the deployment of resources using Azure Resource Manager templates. In an Azure Resource Manager template, you specify the resource—for example, a virtual network, virtual machine, and so on—using a JSON object. Some or all of the properties for the resource can be parameterized so that you can customize your deployment by providing parameter values at deployment time.

If you have spent time developing and maintaining Azure Resource Manager templates, you may have noticed that development of your templates follows a pattern. First you specify the JSON object for a virtual network, then you specify the JSON object for virtual machines that are deployed into the virtual network. Then you specify a JSON object for a network security group to secure the virtual network, and you might specify the JSON object for a load balancer to distribute incoming requests to the virtual machines. Perhaps you have parameterized some of the property values so you can customize the deployment.

While Azure Resource Manager templates are very powerful and allow you to deploy very large and complex architectures to Azure, they require a great deal of knowledge about Azure Resource Manager and the resources themselves. This leads to difficulty maintaining your templates because any modification can lead to unforeseen issues.  
   
The Azure Building Blocks project solves this problem by providing a command line tool and set of Azure Resource Manager templates designed to simplify deployment of Azure resources. You specify settings for Azure resources using the Building Blocks JSON schema, and the command line tool merges these settings with best practice defaults to produce a set of parameter files. The command line tool deploys these parameter files using a set of pre-built Azure Resource Manager templates. Below is an example of Azure Building Blocks workflow:



The Building Blocks JSON schema is designed to be flexible. You can either specify your resources settings in one large file or several small files.

The Template Building Blocks currently support the following resource types:

• Virtual Networks

• Virtual Machines (including load balancers)

• Virtual Machine Extensions

• Route Tables

• Network Security Groups

• Virtual Network Gateways

• Virtual Network Connection

### Deploying Resources Using Building Blocks

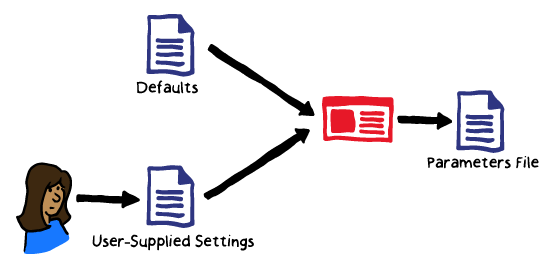
To demonstrate how to use the Azure Building Blocks, we will use the tool to deploy a simple virtual network. This example assumes the **azbb** command line tool and the Azure CLI are both installed on your local machine.

To get started with the building blocks, you must first create a JSON file with a **buildingBlocks** parameter containing an array of resources that you wish to deploy. Each resource within that JSON array can have certain options configured.

For example, a Virtual Network resource would have options such as **addressPrefix**, subnets and name. Any options you do not specify are set using the default options specified in the building blocks repository.

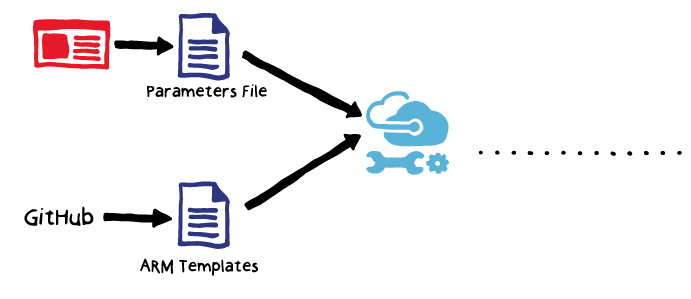
These default options are commonly Microsoft recommended best practices. Once the user-supplied settings file is created, run the **azbb** command line tool to parse your settings and combine it with the default settings from the building blocks repository. The **azbb** command line tool will output an ARM template parameters file that will be used by the tool later in deployment.

We'll begin our architecture with a simple virtual network, where we create a parameters file:

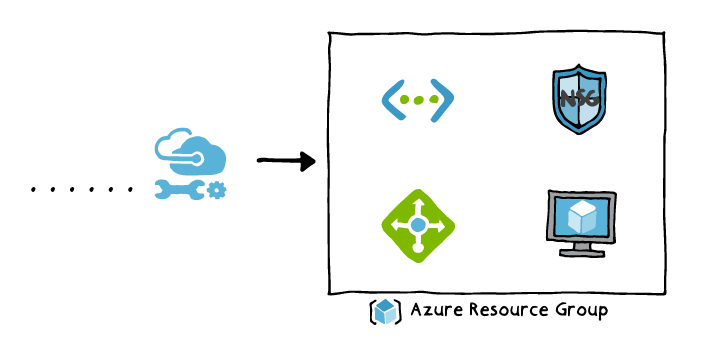


Once the azbb tool has created an ARM template parameters file, the **azbb** tool will use the Azure CLI (version 2.0) to deploy the master ARM template from the building blocks repository using the parameters file that was generated.

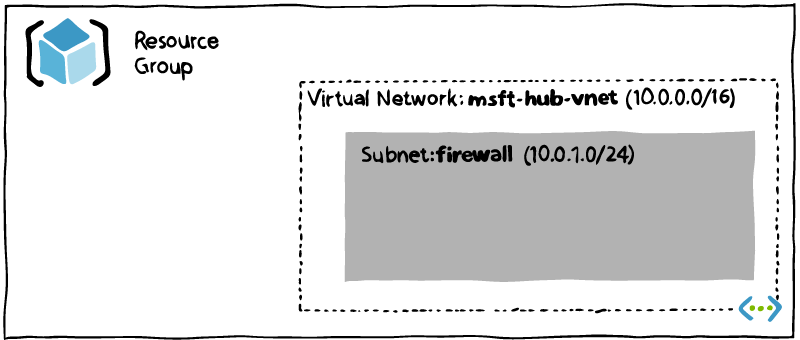
The template and parameters are sent to Azure and a deployment is created.



The Azure deployment will create a collection of resources within an Azure Resource Group. These resource group will have historical deployment information so you can go back and view the template and parameters used by the building blocks command line tool to deploy your resources.



 **SAMPLE RESOURCE GROUP**



The architecture includes the following:

• A VNet named msft-hub-vnet with an address space of **10.0.0.0/16**.

• A subnet within msft-hub-vnet named firewall with an address space of**10.0.1.0/24**.

### Building Block Resource

Every block can be represented as a JSON object. For our example, we will use the following JSON object to represent our simple Virtual Network.

The type property is used by the Azure Building Blocks to identify the type of building block. We're going to deploy a VNet, so we have to specify a VirtualNetwork building block type. This property is required for each of the building blocks.

Every Azure Building block also requires a settings object where the properties for the building block are specified.Let's look at each property for a simple VNet:

• **Name**: In Azure, each resource requires a name to uniquely identify the resource within a resource group. In Azure Building Blocks, you specify a name property for each resource type to provide this unique name. When we deploy this settings file using the command line tool, this is the name that we'll see in the Azure portal user interface. In this settings file we've named the VNet msft-hub-vnet because this in future tutorials this will become the central “hub” VNet for our complex architecture.

• **addressPrefixes**: Next, we specify the address space for our virtual network using the **addressPrefixes** property. The address space is specified using CIDR notation. In our example settings file, we've specified the address space to be 10.0.0.0/16. This means Azure Resource Manager allocates 65536 IP addresses beginning at 10.0.0.0 and ending at 10.0.255.255.

Notice that the field for specifying the virtual network address space is an array. The reason for this is because we can specify multiple address ranges. For example, in addition to 10.0.0.0/16 we could have also specified 11.0.0.0/16 to specify everything between 11.0.0.0 and 11.0.255.255 as well:

"addressPrefixes": [  
"10.0.0.0/16",  
"11.0.0.0/16"  
]

• **subnets**: Now that we have specified the address space for our virtual network, we can begin to create named network segments known as subnets. Subnets are used to manage security, routing, and user access for each subnet independently of the entire VNet. Subnets are also used to segment VMs into back-end pools for load balancers and application gateways.

As you can see from our settings file, we've specified a single subnet named firewall with an address space of 10.0.1.0/24. Note that the subnets property is also an array—we can specify up to 1,000 subnets for each VNet.

### Settings File

To deploy this virtual network, we will need to create a settings file for the azbb command line tool. An empty settings file looks like this:

{  
  "$schema": "https://raw.githubusercontent.com/mspnp/template-building-blocks/master/schemas/buildingBlocks.json",  
  "contentVersion": "1.0.0.0",  
  "parameters" : {  
    "buildingBlocks": {  
      "value": [  
        {}  
      ]  
    }  
  }  
}

In the example above, the **buildingBlocks** array is empty. For our deployment, we will add the simple Virtual Network building block that we discussed earlier in this topic. The settings file for our deployment would look like this:

{  
  "$schema": "https://raw.githubusercontent.com/mspnp/template-building-blocks/master/schemas/buildingBlocks.json",  
  "contentVersion": "1.0.0.0",  
  "parameters": {  
    "buildingBlocks": {  
      "value": [  
        {  
          "type": "VirtualNetwork",  
          "settings": [  
            {  
              "name": "msft-hub-vnet",  
              "addressPrefixes": [  
                "10.0.0.0/16"  
              ],  
              "subnets": [  
                {  
                  "name": "firewall",  
                  "addressPrefix": "10.0.1.0/24"  
                }  
              ]  
            }  
          ]  
        }  
      ]  
    }  
  }  
}

### Deploy a Settings File

To deploy your settings file, you will need to ensure that you are logged in to the Azure CLI. Then, you'll need a few things before you can deploy the VNet using the settings file:

• You need your Azure subscription ID. You can find your subscription ID by using the Azure CLI command **az account list**, or, by going to the Azure Portal and opening the subscriptions blade.

• You'll need to consider the resource group to which the VNet will be deployed. You can deploy to either an existing or new resource group. The Azure Building Blocks command line tool determines if the resource group name you pass with the**-g** option exists or not. If the resource group exists, the command line tool deploys the VNet to the existing resource group. If it doesn't exist, the command line tool creates the resource group for you and then deploys the VNet to the new resource group.

• You'll also need to consider the Azure region where the VNet will be deployed.

Once you have your settings file and the information listed above, you can create a deployment using the following command:

azbb -g <new or existing resource group> -s <subscription ID> -l <region> -p <path to your settings file> --deploy

The command line tool will parse your settings file and deploy it to Azure using Azure Resource Manager. To verify that the VNet was deployed, visit the Azure Portal, click on **Resource Groups** in the left-hand pane to open the **Resource Groups** blade, then click on the name of the resource group you specified above. The blade for that resource group will open, and you should see the **msft-hub-vnet** in the list of resources.

### Lab Steps

#### Online Lab: Getting Started with Azure Resource Manager Templates and Azure Building Blocks

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy core Azure resources by using an Azure Resource Manager Template from the Azure portal

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Deploy an Azure virtual network from the Azure portal by using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T03\Module\_01\Labfiles\Starter\** folder, select the **vnet-simple-template.json** file, and click **Open**. This will load the following content into the template editor pane:

{ "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": { "vnetNamePrefix": { "type": "string", "defaultValue": "vnet-", "metadata": { "description": "Name prefix of the vnet" } }, "vnetIPPrefix": { "type": "string", "defaultValue": "10.2.0.0/16", "metadata": { "description": "IP address prefix of the vnet" } }, "subnetNamePrefix": { "type": "string", "defaultValue": "subnet-", "metadata": { "description": "Name prefix of the subnets" } }, "subnetIPPrefix": { "type": "string", "defaultValue": "10.2.0.0/24", "metadata": { "description": "IP address prefix of the first subnet" } } }, "variables": { "vnetName": "[concat(parameters('vnetNamePrefix'), resourceGroup().name)]", "subnetNameSuffix": "0" }, "resources": [ { "apiVersion": "2018-02-01", "name": "[variables('vnetName')]", "type": "Microsoft.Network/virtualNetworks", "location": "[resourceGroup().location]", "scale": null, "properties": { "addressSpace": { "addressPrefixes": [ "[parameters('vnetIPPrefix')]" ] }, "subnets": [ { "name": "[concat(parameters('subnetNamePrefix'), variables('subnetNameSuffix'))]", "properties": { "addressPrefix": "[parameters('subnetIPPrefix')]" } } ], "virtualNetworkPeerings": [], "enableDdosProtection": false, "enableVmProtection": false }, "dependsOn": [] } ] }

1. Click the **Save** button to persist the template.
2. Back on the **Custom deployment** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Create new** option and, in the text box, type **AADesignLab0201-RG**.
   * In the **Location** drop-down list, select the Azure region to which you want to deploy resources in this lab.
   * Leave the **vnetNamePrefix** text box set to its default value.
   * Leave the **vnetIPPrefix** text box set to its default value.
   * Leave the **subnetNamePrefix** text box set to its default value.
   * Leave the **subnetIPPrefix** text box set to its default value.
   * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   * Click the **Purchase** button.
3. Wait for the deployment to complete before you proceed to the next task.

##### Task 3: View deployment metadata

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group to which you deployed the template in the previous task.
3. With the **Overview** selection active, on the resource group blade, click the **Deployments** link.
4. On the resulting blade, click the latest deployment to view its metadata in a new blade.
5. Within the deployment blade, observe the information displayed in the **Operation details** section.

**Review**: In this exercise, you deployed an Azure virtual network by using an Azure Resource Manager template from the Azure portal

#### Exercise 2: Deploy core Azure resources by using Azure Building Blocks from the Azure Cloud Shell

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployed resources in this lab
   * Resource group: ensure that the **Create new** option is selected and, in the text box, type **AADesignLab0202-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you continue to the next task.

##### Task 2: Install the Azure Building Blocks npm package in Azure Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a local directory to install the Azure Building Blocks npm package:

mkdir ~/.npm-global

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to update the npm configuration to include the new local directory:

npm config set prefix '~/.npm-global'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the ~./bashrc configuration file for editing:

vi ~/.bashrc

1. At the **Cloud Shell** command prompt, in the vi editor interface, scroll down to the bottom of the file (or type **G**), scroll to the right to the right-most character on the last line (or type **$**), type **a** to enter the **INSERT** mode, press **Enter** to start a new line, and then type the following to add the newly created directory to the system path:

export PATH="$HOME/.npm-global/bin:$PATH"

1. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install the Azure Building Blocks npm package:

npm install -g @mspnp/azure-building-blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to exit the shell:

exit

1. In the **Cloud Shell timed out** pane, click **Reconnect**.**Note**: You need to restart Cloud Shell for the installation of the Buliding Blocks npm package to take effect.

##### Task 3: Deploy an Azure virtual network from Cloud Shell by using Azure Building Blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download the GitHub repository containing the Azure Building Blocks templates:

git clone https://github.com/mspnp/template-building-blocks.git

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the content of the Azure Building Block parameter file you will use for this deployment:

cat ./template-building-blocks/scenarios/vnet/vnet-simple.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you created earlier in this exercise:

RESOURCE\_GROUP='AADesignLab0202-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab0201-RG'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy a virtual network by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP -s $SUBSCRIPTION\_ID -l $LOCATION -p ./template-building-blocks/scenarios/vnet/vnet-simple.json --deploy

1. Wait for the deployment to complete before you proceed to the next task.

##### Task 4: View deployment metadata

1. On the left side of the portal, click the **Resource groups** link.
2. On the **Resource groups** blade, click the entry representing the resource group you created earlier in this exercise.
3. With the **Overview** selection active, on the resource group blade, click the **Deployments** link.
4. On the resulting blade, click the latest deployment to view its metadata in a new blade.
5. Within the deployment blade, observe the information displayed in the **Operation details** section.
6. Close the **Cloud Shell** pane.

**Review**: In this exercise, you deployed an Azure virtual network by using an Azure Resource Manager template from the Azure portal

#### Exercise 3: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab02')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab02')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 1 Review Questions

**Manage resources**  
  
You are designing a solution to migrate your on-premises architecture to Microsoft Azure.  
You need to ensure resources are deployed in a componentized and reusable manner.  
  
What Azure service is essential to the deployment of module resources? What is used to manage resources with a shared lifecycle?

Suggested Answer ↓

Azure Resource Manager is designed to represent each service in Azure as a resource provider and each service instance as a modular resource. Resource Groups provide a common lifecycle for child resources. You can add multiple resources to a single resource group. You can manage all resources in a resource group as a single unit.

**Manage resource access**  
You are designing a solution to migrate your on-premises architecture to Microsoft Azure.  
You need to manage access to Azure resources by team members in your company.  
How is access to resources managed in Azure? To which security principals can access be assigned?

Suggested Answer ↓

Azure AD identities grant predetermined levels of access to an Azure subscription, resource group, or individual resource. Role-Based access control is used to assign roles to existing Azure AD identities. You can assign roles to users, groups, and service principals.

**Manage security objects**  
You are designing a solution for Microsoft Azure.  
What should you use to manage certificates and keys? What should you do to secure resource deployments?

Suggested Answer ↓

Azure Key Vault allows the storage of cryptographic keys, certificates, and secrets that can be used by applications, services, and users. You can secure resource deployments by r

### Infrastructure-Backed PaaS

This module describes services that use infrastructure but manage the infrastructure on behalf of the user instead of obfuscating the infrastructure resources. The module focuses on infrastructure-backed PaaS options such as Azure Service Fabric, Container Service, and App Service Environments. The module will explore how to deploy custom workloads to these services such as an HPC batch processing task.

After completing this module, students will be able to:

• Describe the differences between App Service Environments, Service Fabric and Container Service.

• Use Azure Batch to manage an HPC workload.

• Migrate to an Infrastructure-backed PaaS service from another IaaS service or a legacy Cloud Service.

**Infrastructure-Backed Platform-as-a-Service (PaaS)**

The need to deploy highly scalable, isolated applications with secure network access and high memory utilization is one that is growing with increased cloud adoption. This lesson will describe the options within Microsoft Azure to deploy App Service Environments, Azure Service Fabric and Azure Container Service applications. The growth of containerized applications and microservice based applications requires the use of these services to deploy open source solutions to deploy, manage and orchestrate at low cost and high flexibility.

After completing this section, you will be able to:

• Describe App Service Environments and distinguish between v1 and v2.

• Describe Azure Service Fabric and distinguish between Microservices and Containers.

• Describe Azure Container Services and the variants, **Docker**, **Mesosphere**, **Kubernetes**.

• Deploy an App Service Environment with a web app.

**Infrastructure-Backed PaaS**

Every Microsoft Azure PaaS service is already hosted on Azure IaaS. Azure provides several options for architecting and hosting your own PaaS applications at scale in the same secure and isolated infrastructure within the Azure Datacenters. To be able to provide highly scalable applications requiring isolation with secure network access and high memory utilization, there are three services available. App Service Environments provides a dedication scalable home for Azure Web Apps. Azure Service Fabric provides a cluster of VMs to host containers and microservices for those applications. Finally, there is the Azure Container Service this provides open source tools to orchestrate containers based on Docker Swarm, Mesosphere and Kubernetes clusters. The benefit of all these technologies is that Azure lets you concentrate on the application architecture, and code whilst Azure looks after the infrastructure and the orchestration of the containers. In this lesson, we will detail these services and their intended uses.

### App Service Environments

App Services are useful because they separate many of the hosting and management concerns for your web application and allow you to focus on your application's functionality and configuration. There are some scenarios, however, where you require a bit more control.

For example, your organization may require you to make sure that all of the virtual machines hosting your web applications do not allow any outbound requests. This is a typical scenario when implementing a web solution that must be PCI compliant. With App Services, you can't access or modify the configuration of the virtual machines hosting your applications. You do not have any mechanism to implement this requirement.

To implement scenarios where you require more control, you can use the App Service Environment (ASE) service in Azure. ASE allows you to configure network access and isolation for your applications. ASE also allows you to scale using pools of instances far beyond the limits of a regular App Service plan instance. Finally, ASE instances are dedicated to your application alone. You retain much of the convenience of using App Services such as automatic scaling, instance management, and load balancing when using ASE but you gain more control.  
Networking in ASE With ASE, you can configure network access using the same concepts and paradigms that you use in Virtual Machines. Environments are created within a subnet in an Azure Virtual Network. You can use Network Security Groups to restrict network communications to the subnet where the Environment resides. You can also use a protected connection to connect your Virtual Network to corporate resources so that your ASE instance can securely use the resources.

ASE is available in v1 and v2. The ASE v1 is useful since it can take advantage of both classic and Resource Manager Virtual Networks whilst v2 can only use Resource Manager resources. ASE v2 also automates most of the scaling, creation, and maintenance which v1 requires to be carried out manually.

### Azure Service Fabric

Azure Service Fabric provides a distributed systems **DevOps** solution making it simple to package, deploy, and manage scalable and reliable containers and microservices. Developers can use Service Fabric to assist with the developing and managing cloud native applications.  
The use of Service Fabric can avoid complex infrastructure problems and leave operations and development staff to concentrate on implementing mission-critical workloads that are easy to manage, scale and result in a reliable solution. Service Fabric is intended for container-based applications running on enterprise-class, cloud-scale environments.

**Microservice based applications**

Service Fabric creates a cluster of VMs that allow you to run high density, massively scalable applications composed of microservices. These applications can be stateful or stateless microservices running in containers which Azure Service Fabric will assist with provisioning, deploying, monitoring and managing in fast and efficient, automated fashion.

Service Fabric powers many Microsoft services today, including Azure SQL Database, Azure Cosmos DB, Cortana, Microsoft Power BI, Microsoft Intune, Azure Event Hubs, Azure IoT Hub, Dynamics 365, Skype for Business, and many core Azure services. Azure Service Fabric is ideal for hosting cloud-native applications at high scale often growing to hundreds of thousands of machines.

A microservice could be a user profile, a customer’s shopping cart, queues, caches, or any other component of an application. Service Fabric is a platform built to handle microservices by placing each one in a uniquely named container.

Service Fabric provides comprehensive runtime and lifecycle management capabilities to applications that are composed of these microservices. It hosts microservices inside containers that are deployed and activated across the Service Fabric cluster. The use of microservices adds the ability to increase the density of the containers used by a service.

As an example, a single cluster for Azure SQL Database contains hundreds of machines running tens of thousands of containers hosting a total of hundreds of thousands of databases. Each Azure SQL database is a Service Fabric state-full microservice.

**Application lifecycle management**

Azure Service Fabric supports the application lifecycle and CI/CD of cloud applications including containers. Service Fabric empowers operators to use simple workflows to provision, deploy, patch, and monitor applications. This dramatically reduces the workload for the operators. Service Fabric is integrated with **CI/CD** tools such as **Visual Studio Team Services**, **Jenkins**, and **Octopus Deploy**.

### Azure Container Service

Azure Container Service (ACS) provides several alternative ways to simplify the creation, configuration, and management of a cluster of virtual machines that are configured to run your applications in a container. Using open-source orchestration and scheduling tools provides a large community of experience and expertise to assist in deploying and managing your containerized applications. ACS provides three distinct container orchestration technologies:

**• Docker Swarm**

**• Mesosphere DC/OS**

**• Kubernetes**

ACS uses Docker images to provide container portability. The use of the orchestration technologies enables the ability to scale applications to tens of thousands of containers. ACS makes no charges for the service or orchestration since all the tools provided are open source software. Charges are made for resource utilization only. ACS is available for Standard A, D, DS, G, GS, F, and FS series Linux virtual machines. You are only charged for these VMs, Storage and networking resources actually used.

The latest service available is Kubernetes clusters. This is the AKS service. AKS passes the cluster management functions to Azure and removes it from the operator or administrator. Azure manages health, maintenance, and monitoring. AKS also provides easy scaling, automatic version upgrades, and self-healing master nodes. Since Azure provides all the usual management functions, there is no SSH access to the AKS cluster. Access is provided to the Kubernetes API endpoints. This allows any software capable of communicating with Kubernetes endpoints to be used. Examples could include kubectl, helm, or draft. Whilst Azure handles the infrastructure management, Kubernetes itself handles your containerized applications.

The list of Kubernetes management features includes:

**• Automatic binpacking**

**• Self-healing**

**• Horizontal scaling**

**• Service discovery and load balancing**

**• Automated rollouts and rollbacks**

**• Secret and configuration management**

**• Storage orchestration**

**• Batch execution**

### High Performance Computing

#### High Performance Computing (HPC)

Azure provides solutions to allow for massively scalable parallel processing of memory-intensive workloads, such as 3D rendering and tasks that process, transform, and analyze large volumes of data. Azure Batch provides this service for short-term massively scalable infrastructure. For other cloud-based and even hybrid solutions, Azure provides access to both Windows and Linux HPC clusters. This lesson will describe these solutions and their uses. After completing this lesson, you will be able to:

• Describe Azure HPC Pack features.

• Describe the Azure Batch service.

• Understand the use of the HPC pack in Cloud only and hybrid scenarios.

• Deploy an Azure HPC pack for Windows.

**High Performance Computing (HPC)**

Traditionally, complex processing was something saved for universities and major research firms. A combination of cost, complexity, and accessibility served to keep many from pursuing potential gains for their organization from processing large and complex simulations or models. Cloud platforms have democratized hardware so much that massive computing tasks are within reach of hobbyist developers and small to medium-sized enterprises.

High-Performance Computing (HPC) typically describes the aggregation of complex processes across many different machines thereby maximizing the computing power of all of the machines. Through HPC in the cloud, one could create enough compute instances to create a model or perform a calculation and then destroy the instances immediately afterward. Advancements in the HPC field have led to improvements in the way that machines can share memory or communicate with each other in a low latency manner.

### Remote Direct Memory Access

Remote Direct Memory Access, or RDMA, is a technology that provides a low-latency network connection between processing running on two servers, or virtual machines in Azure. This technology is essential for engineering simulations, and other compute applications that are too large to fit in the memory of a single machine. From a developer perspective, RDMA is implemented in a way to make it seem that the machines are “sharing memory.” RDMA is efficient because it copies data from the network adapter directly to memory and avoids wasting CPU cycles.

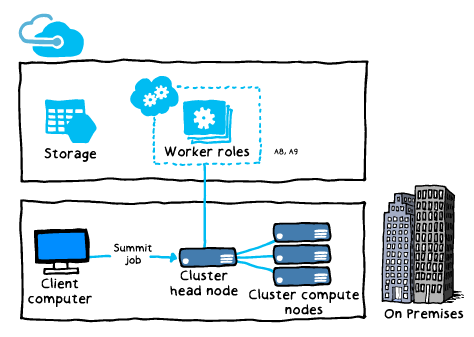
The A8 and A9 VM size in Azure use the InfiniBand network to provide RDMA virtualized through Hyper-V with near “bare metal” performance of less than 3-microsecond latency and greater than 3.5 Gbps bandwidth.

**HPC Pack using Azure Virtual Machines**

HPC Pack is Microsoft’s HPC cluster and job management solution for Windows. The HPC Pack can be installed on a server that functions as the "head node," and the server can be used to manage compute nodes in an HPC cluster.

HPC Pack is not required for you to use the A8, A9, A10, and A11 instances with Windows Server, but it is a recommended tool to create Windows HPC clusters in Azure. In the case of A8 and A9 instances, HPC Pack is the most efficient way to run Windows MPI applications that access the RDMA network in Azure. HPC Pack includes a runtime environment for the Microsoft implementation of the Message Passing Interface for Windows.

HPC Pack can also be used in hybrid scenarios where you want to "burst to Azure" with A8 or A9 instances to obtain more processing power.



**RDMA on Linux Virtual Machines in Azure**

Starting with HPC Pack 2012 R2 Update 2, HPC Pack supports several Linux distributions to run on compute nodes deployed in Azure VMs, managed by a Windows Server head node. With the latest release of HPC Pack, you can deploy a Linux-based cluster that can run MPI applications that access the RDMA network in Azure. Using HPC Pack, you can create a cluster of virtual machines using either Windows or Linux that use the Intel Message Passing Library (MPI) to spread the workload of simulations and computations among compute nodes of many virtual machines.

### Azure Batch

Microsoft Azure Batch is a fully-managed cloud service that provides job scheduling and compute resource management for developers in organizations, independent software vendors, and cloud service providers. Both new and existing high-performance computing (HPC) applications running on workstations and clusters today can be readily enabled to run in Azure at scale, and with no on-premises infrastructure required. Common application workloads include image and video rendering, media transcoding, engineering simulations, Monte Carlo simulations, and software test execution, among others; all highly parallel, computationally intensive workloads that can be broken into individual tasks for execution. With Azure Batch, you can scale from a few VMs, up to tens of thousands of VMs, and run the most massive, most resource-intensive workloads.

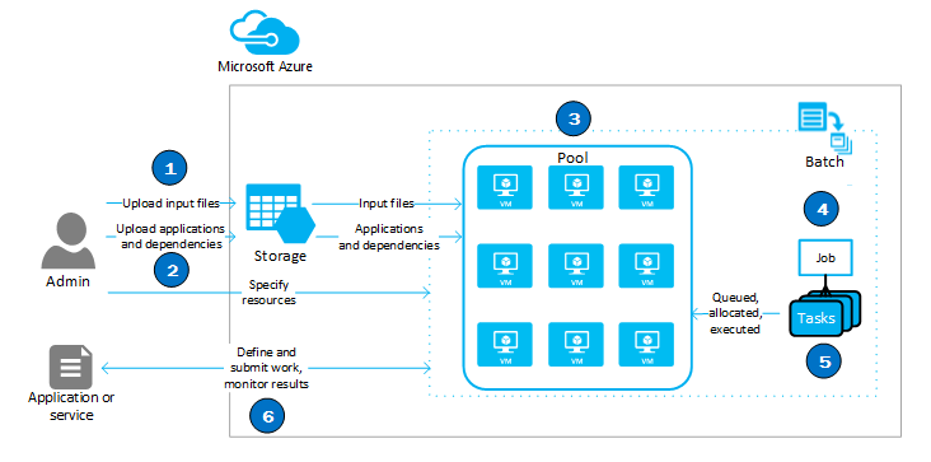
**Azure Batch**

Azure Batch is a service that manages Virtual Machines for large-scale parallel and high-performance computing (HPC) applications. Batch is a Platform as a Service (PaaS) offering that manages the VMs necessary for your compute jobs for you instead of forcing you to manage an HPC cluster or job schedule. Batch provides auto-scaling functionality and job scheduling functionality in addition to managing compute nodes.

Batch computing is a familiar pattern for organizations that process, transform, and analyze large amounts of data, either on a schedule or on-demand. It includes end-of-cycle processing such as a bank’s daily risk reporting or a payroll that must be done on schedule. It also includes large-scale business, science, and engineering applications that typically need the tools and resources of a compute cluster or grid. Applications include traditional HPC applications such as fluid dynamics simulations as well as specialized workloads in fields ranging from digital content creation to financial services to life sciences research. Batch works well with intrinsically parallel (sometimes called “embarrassingly parallel”) applications or workloads, which lend themselves to running as parallel tasks on multiple computers.

**Scaling out Parallel Workloads**

The Batch API can handle scale out of an intrinsically parallel workload such as image rendering on a pool of up to thousands of compute cores. Instead of having to set up a compute cluster or write code to queue and schedule your jobs and move the necessary input and output data, you automate the scheduling of large compute jobs and scale a pool of compute VMs up and down to run them. You can write client apps or front-ends to run jobs and tasks on demand, on a schedule, or as part of a larger workflow managed by tools such as Azure Data Factory.



You can also use the Batch Apps API to wrap an existing application, so it runs as a service on a pool of compute nodes that Batch manages in the background. The application might be one that runs today on client workstations or a compute cluster. You can develop the service to let users offload peak work to the cloud, or run their work entirely in the cloud. The Batch Apps framework handles the movement of input and output files, the splitting of jobs into tasks, job and task processing, and data persistence.

### Stateless Component Workloads

In addition to the formal HPC services above, Azure provides **non-managed solutions** and related services to provide a degree of HPC using IaaS. These include:

• Virtual Machines

• VM scale Sets

• Azure Container Services

• HDInsight

• Machine Learning

And plenty more. The inbuilt massively scalable Azure fabric provides a perfect platform to deploy parallel compute tasks on several services. The use of Azure Resource manager to automate deployment and auto scale based on various criteria enables the developer and architect to be creative with their application solutions. As an example, deploying VMSS to manage an HPC workload by varying the number of VMs based on the queue length would combine auto-scale based on a custom metric with HPC Pack and templated infrastructure deployment. You do not have to rely on the pre defined HPC services.

### On-Premises Lift and Shift

Migrating from on-premises workloads into Azure can be achieved in several ways that depend upon the individual workloads and the level of rewriting that the user is prepared or able to achieve. This lesson covers the Azure Migrate preview service and strategies for migrating from on-premises to Azure IaaS, and PaaS solutions. The ability to convert cloud service roles into Azure Service Fabric microservices is also discussed.

After completing this lesson, you will be able to:

• Describe Azure Migrate.

• Understand the process to migrate from IaaS to PaaS.

• Describe the best way to migrate apps to Azure.

• Use Azure Migrate to plan a single VM migration.

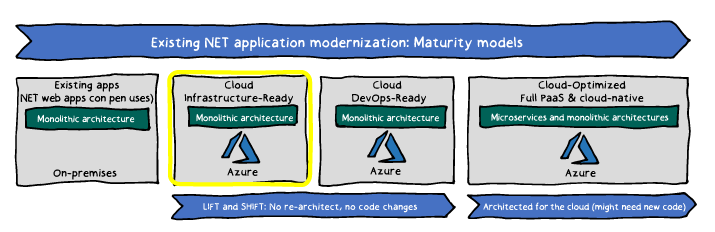
**Migration**

The challenge of migrating workloads and applications to a cloud-based or hybrid platform or even from converting from cloud-based IaaS to a PaaS solution is one that causes much concern when adopting a public cloud strategy. There are several options for beginning the “journey to the cloud.” For moving workloads, the “lift and shift” approach provides all the benefits of pay as you go computing without the potential headaches of rewriting application code or re-architecting the application to fit a specific cloud application pattern.

**On-Premises Lift and Shift**

A new service from Microsoft, the Azure Migrate service provides a discovery and assessment tool. Azure Migrate assesses suitability for migration and ensures that sizing is correct for the performance of the VM. An estimate of the cost of running your VM in Azure is also available; this service is designed for those considering a lift and shift migration or beginning to plan their migration to the cloud. Migrate provides the ability to visualize dependencies of a specific VM or for all VMs in a group. The service currently only provides assessment for VMWare VMs but soon will also include Hyper-V VMs.

The lift and shift approach is an ideal method to begin the migration of workloads to modernization. Commencing with lift and shift, moving through the phases shown below to cloud-optimized taking advantage of the PaaS services and cloud native applications. The latter often requires the applications to be rewritten.



### Migration from Classic IaaS

If there are resources in an Azure subscription based on the classic or Azure Service Manager model, it is now possible to migrate them to an Azure Resource Manager (ARM) deployment. The following resources can be migrated to ARM from ASM:

• Virtual Machines

• Availability Sets

• Cloud Services

• Storage Accounts

• Virtual Networks

• VPN Gateways

• Express Route gateways

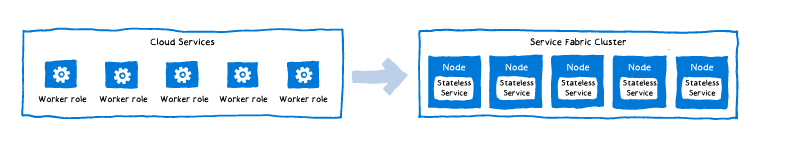
• Network Security Groups

• Route Tables

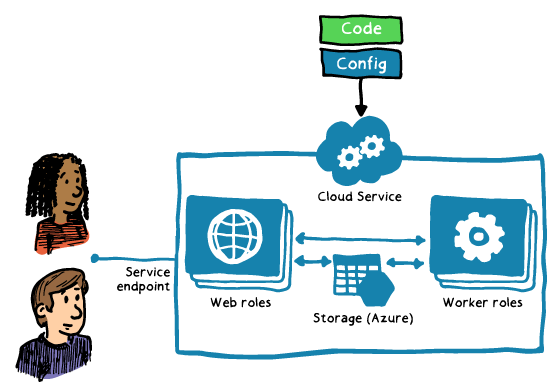
• Reserved IPs

### Migration from Cloud Services

When migrating cloud services to a PaaS solution, it is necessary to consider the difference between VMs, workloads, and applications in each model. A cloud service deploys applications as VMs; code is connected to a VM instance which might be a Web role or a worker role. To scale the application, more VMs are deployed.

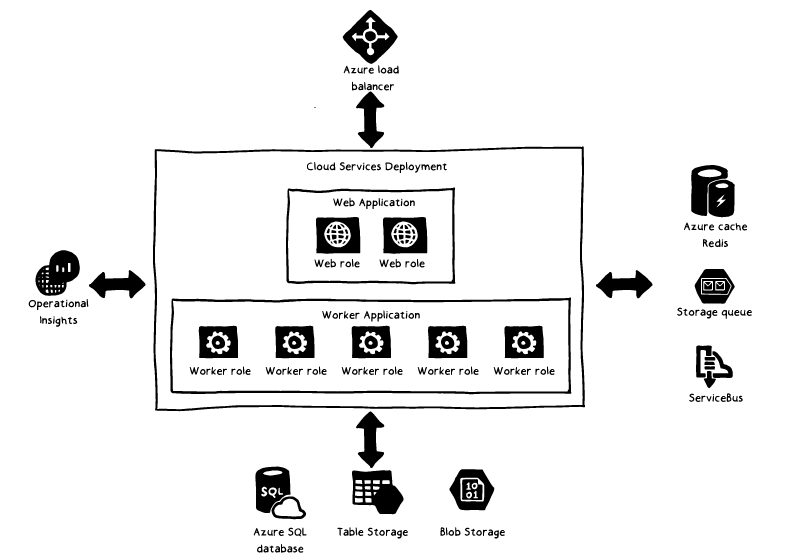


The deployment package contains the web role and worker role definition and specifies the instance count for each role; an instance is a VM hosting that role.



### Deployment Package

Migrating a cloud service to Service Fabric switches to deploying applications to VMs that are already running Service Fabric either on Windows or Linux. The applications or services that are deployed are entirely unrelated to the VM infrastructure. The service fabric application platform hides the VM layer from the application. This also allows multiple applications to be deployed to a service fabric cluster. The architecture of applications running on the two services is fundamentally different. A cloud service application will typically have external dependencies which manage the data and state of an application and the method of communicating between web and worker roles.



**CLOUD SERVICE SOLUTION**

A Service fabric application can also rely on the same external service dependencies. The quickest and easiest way to migrate a Cloud Service application to service fabric is to merely convert the Web roles and worker roles to stateless services whilst keeping the architecture the same. If the aim is to remove the external dependencies and take full advantage of the ability to unify deployment, management and upgrade models, then state-full services would be required which will mean full code and application rewrites. Migration both to the cloud in general as a lift and shift and to modern cloud-native applications is a complicated and time-consuming project.

### Lab Steps

#### Deploying Managed Containerized Workloads to Azure

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesig>n

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Create Azure Kubernetes Service (AKS) cluster

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab.
   * In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0401-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 3: Create an AKS cluster by using Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this task:

RESOURCE\_GROUP='AADesignLab0402-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder <Azure region> with the name of the Azure region to which you intend to deploy resources in this lab):

LOCATION='<Azure region>'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

az group create --name $RESOURCE\_GROUP --location $LOCATION

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new AKS cluster:

az aks create --resource-group $RESOURCE\_GROUP --name aad0402-akscluster --node-count 1 --node-vm-size Standard\_DS1\_v2 --generate-ssh-keys

**Note**: If you receive an error message regarding availability of the VM size which value is represented by the --node-vm-size parameter, review the message and try other suggested VM sizes.**Note**: Alternatively, you can identify VM sizes available in your subscription in a given region by running the following command and reviewing the values in the **Restriction** column (make sure to replace the <region> placeholder with the name of the target region):

Get-AzComputeResourceSku | where {$\_.Locations -icontains "region"} | Where-Object {($\_.ResourceType -ilike "virtualMachines")}.

**Note**: The **Restriction** column will contain the value **NotAvailableForSubscription** for VM sizes that are not available in your subscription.**Note**: As of 2/21/2019, VM Size **Standard\_DS2\_V2** was available in **westeurope**

1. Wait for the deployment to complete before you proceed to the next task.**Note**: This operation can take up to 10 minutes.

##### Task 4: Connect to the AKS cluster.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the credentials to access the AKS cluster:

az aks get-credentials --resource-group $RESOURCE\_GROUP --name aad0402-akscluster

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify connectivity to the AKS cluster:

kubectl get nodes

1. At the **Cloud Shell** command prompt, review the output and verify that the node is reporting the **Ready** status. Rerun the command until the correct status is shown.

**Result**: After you complete this exercise, you should have successfully deployed a new AKS cluster.

#### Exercise 2: Managing an AKS cluster and its containerized workloads.

##### Task 1: Deploy a containerized application to an AKS cluster

1. In the Microsoft Edge window, in the Azure portal, at the **Cloud Shell** prompt, type the following command and press **Enter** in order to deploy the **nginx** image from the Docker Hub:

kubectl run aad0402-akscluster --image=nginx --replicas=1 --port=80

**Note**: Make sure to use lower case letters when typing the name of the deployment.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that a Kubernetes pod has been created:

kubectl get pods

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify the state of the deployment:

kubectl get deployment

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to make the pod available from Internet:

kubectl expose deployment aad0402-akscluster --port=80 --type=LoadBalancer

**Note**: Make sure to use lower case letters when typing the name of the deployment.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify whether the public IP address has been provisioned:

kubectl get service --watch

1. Wait until the value in the **EXTERNAL-IP** column for the **aad0402-akscluster** entry changes from to a public IP address, then press **Ctrl-C** key combination. Note the public IP address in the **EXTERNAL-IP** column for **aad0402-akscluster**.
2. Start Microsoft Edge and browse to the IP address you obtained in the previous step. Verify that Microsoft Edge displays a web page with the **Welcome to nginx!** message.

##### Task 2: Scaling containerized applications and AKS cluster nodes

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to scale the deployment:

kubectl scale --replicas=2 deployment/aad0402-akscluster

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify the outcome of scaling the deployment:

kubectl get pods

**Note**: Review the output of the command and verify that the number of pods increased to 2.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to scale out the number of cluster nodes:

az aks scale --resource-group $RESOURCE\_GROUP --name aad0402-akscluster --node-count 2

1. Wait for the provisioning of the additional node to complete.**Note**: This operation can take up to 10 minutes. If it fails, rerun the az aks scale command.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify the outcome of scaling the cluster:

kubectl get nodes

**Note**: Review the output of the command and verify that the number of nodes increased to 2.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to scale the deployment:

kubectl scale --replicas=10 deployment/aad0402-akscluster

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify the outcome of scaling the deployment:

kubectl get pods

**Note**: Review the output of the command and verify that the number of pods increased to 10.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to review the pods distribution across cluster nodes:

kubectl get pod -o=custom-columns=NODE:.spec.nodeName,POD:.metadata.name

**Note**: Review the output of the command and verify that the pods are distributed across both nodes.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the deployment:

kubectl delete deployment aad0402-akscluster

#### Exercise 3: Autoscaling pods in an AKS cluster

##### Task 1: Deploy a Kubernetes pod by using a .yaml file.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download a sample containerized application:

git clone https://github.com/Azure-Samples/azure-voting-app-redis.git

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to navigate to the location of the downloaded app:

cd azure-voting-app-redis

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list the content of the application **.yaml** file:

cat azure-vote-all-in-one-redis.yaml

1. Review the output of the command and verify that the pod defninition includes requests and limits in the followng format:

resources: requests: cpu: 250m limits: cpu: 500m

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the application based on the **.yaml** file:

kubectl apply -f azure-vote-all-in-one-redis.yaml

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to to verify that a Kubernetes pod has been created:

kubectl get pods

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to identify whether the public IP address for the containerized application has been provisioned:

kubectl get service azure-vote-front --watch

1. Wait until the value in the **EXTERNAL-IP** column for the **azure-vote-front** entry changes from to a public IP address, then press **Ctrl-C** key combination. Note the public IP address in the **EXTERNAL-IP** column for **azure-vote-front**.
2. Start Microsoft Edge and browse to the IP address you obtained in the previous step. Verify that Microsoft Edge displays a web page with the **Azure Voting App** message.

##### Task 2: Autoscale Kubernetes pods.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download a sample containerized application:

git clone https://github.com/kubernetes-incubator/metrics-server.git

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install **Metrics Server**:

kubectl create -f ~/metrics-server/deploy/1.8+/

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to configure autoscaling for the **azure-vote-front** deployment:

kubectl autoscale deployment azure-vote-front --cpu-percent=50 --min=3 --max=10

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the status of autoscaling:

kubectl get hpa

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the pods:

kubectl get pods

**Note**: Verify that the number of replicas incrased to 3. If that is not the case, wait one minute and rerun the two previous steps.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the deployment:

kubectl delete deployment azure-vote-front

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the deployment:

kubectl delete deployment azure-vote-back

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the commands you ran in the previous steps completed successfully:

kubectl get pods

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the AKS cluster:

az aks delete --resource-group AADesignLab0402-RG --name aad0402-akscluster --yes --no-wait

1. Close the **Cloud Shell** pane.

**Review**: In this exercise, you implemented autoscaling of pods in an AKS cluster

#### Exercise 4: Implement DevOps with AKS

##### Task 1: Deploy DevOps with AKS

> \*\*Note\*\*: This solution is based on the DevOps with Containers solution described at https://docs.microsoft.com/en-us/azure/architecture/example-scenario/apps/devops-with-aks.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to generate the SSH key pair that will be used to authenticate when accessing the Linux VMs running the Jenkins instance and Grafana console:

ssh-keygen -t rsa -b 2048

* + When prompted to enter the file in which to save the key, press **Enter** to accept the default value **(~/.ssh/id\_rsa)**.
  + When prompted to enter passphrase, press **Enter** twice.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair:

PUBLIC\_KEY=$(cat ~/.ssh/id\_rsa.pub)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair and which takes into account any special character the public key might include:

PUBLIC\_KEY\_REGEX="$(echo $PUBLIC\_KEY | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"

**Note**: This is necessary because you will use the **sed** utility to insert this string into the Azure Resource Manager template parameters file. Alternatively, you could simply open the file and enter the public key string directly into the file.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use for the deployment:

RESOURCE\_GROUP='AADesignLab0403-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab0402-RG'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

az group create --name $RESOURCE\_GROUP --location $LOCATION

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create an Azure Active Directory service principal for the authentication of services and resources within the sample solution:

SERVICE\_PRINCIPAL=$(az ad sp create-for-rbac --name AADesignLab0403-SP)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the **appId** attribute of the newly created service principal:

APP\_ID=$(echo $SERVICE\_PRINCIPAL | jq .appId | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to retrieve the **password** attribute of the newly created service principal:

PASSWORD=$(echo $SERVICE\_PRINCIPAL | jq .password | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create the parameters file you will use for deployment of the sample solution and open it in the vi interface:

vi ~/parameters.json

1. At the **Cloud Shell** command prompt, in the vi editor interface, add the content of the sample parameters file (**\allfiles\AZ-301T03\Module\_02\Labfiles\Starter\parameters.json**):

{ "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#", "contentVersion": "1.0.0.0", "parameters": { "spClientId": { "value": "$APP\_ID" }, "spClientSecret": { "value": "$PASSWORD" }, "linuxAdminUsername": { "value": "Student" }, "linuxAdminPassword": { "value": "Pa55w.rd1234" }, "linuxSSHPublicKey": { "value": "$PUBLIC\_KEY\_REGEX" } } }

1. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **appId** attibute with the value of the **$APP\_ID** variable in the parameters file:

sed -i.bak1 's/"$APP\_ID"/"'"$APP\_ID"'"/' ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **password** attribute with the value of the **$PASSWORD** variable in the parameters file:

sed -i.bak2 's/"$PASSWORD"/"'"$PASSWORD"'"/' ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **sshPublicKey** parameter with the value of the **$PUBLIC\_KEY\_REGEX** variable in the parameters file:

sed -i.bak3 's/"$PUBLIC\_KEY\_REGEX"/"'"$PUBLIC\_KEY\_REGEX"'"/' ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the placeholders were successfully replaced in the parameters file:

cat ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the sample solution by using its Azure Resource Manager template residing in a GitHub repository:

az group deployment create --resource-group $RESOURCE\_GROUP --template-uri https://raw.githubusercontent.com/mspnp/solution-architectures/master/apps/devops-with-aks/azuredeploy.json --parameters @parameters.json

1. Wait for the deployment to complete before you proceed to the next task.**Note**: The deployment can take up to 15 minutes.

##### Task 2: Review the DevOps with AKS architecture

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the **AADesignLab0403-RG** resource group.
3. On the **AADesignLab0403-RG** resource group blade, review the list of resources and compare them with the information available at <https://docs.microsoft.com/en-us/azure/architecture/example-scenario/apps/devops-with-aks>

**Review**: In this exercise, you deployed Azure VMs running Windows Server 2016 Datacenter and Linux from Cloud Shell by using Azure Building Blocks.

#### Exercise 5: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab04')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab04')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 2 Review Questions

**Infrastructure as a Service (IaaS)**  
You are designing a solution for your company.   
You need to automate management of the infrastructure for the solution.

What Azure architecture type provides automatic management of infrastructure? What Azure services are built on this principle?

Suggested Answer ↓

Azure provides an Infrastructure-Backed Platform-as-a-Service (IaaS) architecture that can be used when automatic management of infrastructure is required. There are several services that Azure offers built on this architecture including Web Apps, Service Fabric Apps, and Container Services.

**High performance computing**  
You are designing a solution for a company. The solution requires high performance and low latency.   
Which Azure services support High Performance Computing?

Suggested Answer ↓

he Microsoft Azure platform has a large number of features that support HPC, including HPC Pack, Azure Batch, and Stateless Component Workloads.

**Migration strategies**  
You develop a solution to migrate your company architecture to Azure Resource Manager (ARM).

You need to move all on-premises and classic Azure resources.  
What types of migration does Azure support? What service does Azure offer for virtual machines (VMs)?

Suggested Answer ↓

Microsoft Azure supports multiple types of migration strategies including on-premises lift and shift, classic (Azure Service Manager) migration to ARM, and cloud to Platform-as-a-Service (PaaS). The Azure Migrate service assesses suitability of workloads and environments for migration, and ensures that VM sizing is optimized.

### Web Apps

This module describes how solutions can leverage serverless application hosting services in Azure to host web applications, REST APIs, integration workflows and HPC workloads without the requirement to manage specific server resources. The module focuses on App Services-related components such as Web Apps, API Apps, Mobile Apps, Logic Apps, and Functions.

After completing this module, students will be able to:

• Select between hosting application code or containers in an App Service instance.

• Describe the differences between API, Mobile, and Web Apps.

• Integrate an API or Logic App with the API Management service.

• Design an App Service Plan or multi-region deployment for high performance and scale.

**Azure Web Apps**

This lesson describes the Web Apps service. In many scenarios, it is preferable to use a quick and easy way to deploy web applications to the cloud rather than to reengineer the web applications as cloud projects. Web Apps allow you to quickly create a new Web App and iterate changes to the Web App in an agile manner.

After completing this section you will be able to:

• Describe the Web Apps service.

• List the different tiers for a Web App.

**Web Apps**

Web Apps is a low friction Platform-as-a-Service (PaaS) offering to host your web applications in the Azure platform. The service is fully managed and you can easily configure advanced features such as AlwaysOn, custom domains, and autoscale by using either portal.

**Flexibility**

You can use a variety of integrated development environments (IDEs) and frameworks, such as .NET, Java, PHP, Node.js, or Python, to develop your web applications that are eventually deployed to Azure Web Apps. You can use Git and Kudu to deploy Node.js or PHP web applications. You also can deploy web applications that are developed in Microsoft Visual Studio to Web Apps by using the File Transfer Protocol (FTP) or the Web Deploy protocol.

**Scalability**

Because Web Apps is a fully managed service implementation, you can focus on developing your application and solving business problems instead of the hosting implementation and hardware scaling or specifics. You can easily scale up a stateless web application by configuring autoscale in the portal. Autoscale creates multiple instances of your Web App that are automatically load balanced so that your application can meet potential spikes in demand.

**Web App Containers**

The Web Apps service is offered in both a Windows and Linux variant. The Linux variant specifically offers the ability to host Docker containers directly using a Web App. The docker containers can be sourced from Docker Hub, Azure Container Registry or GitHub. Containers can be deployed manually, as part of the Web App deployment or deployed in a streamlined continuous integration process using Docker Hub or GitHub.

### API Apps

API and Mobile Apps are specialized version of Web Apps designed to server different purposes in an all-up solution on Azure.

API Apps provide specific support for developing, hosting and securing your custom API’s in the context of App. API Apps can run either custom code or can run dozens of pre-built software to connect to existing popular Software-as-a-Service solutions. API Apps share a lot of the same features and support as Web Apps.

### Mobile Apps

Azure Mobile Apps is a component of Azure App Services offering designed to make it easy to create highly-functional mobile apps using Azure. Mobile Apps brings together a set of Azure services that enable backend capabilities for your apps. Mobile Apps provides the following backend capabilities in Azure to support your apps:

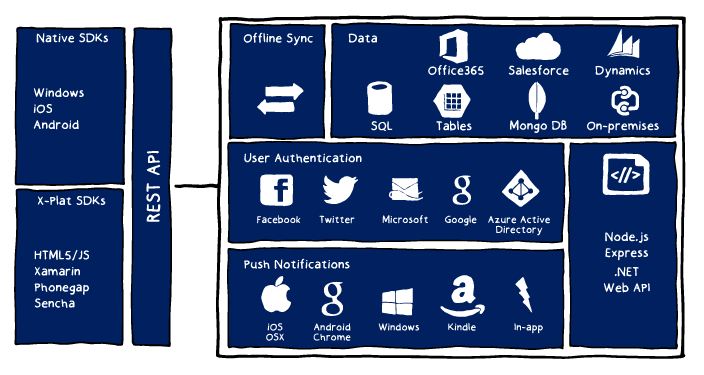
• **Single Sign On:** Select from an ever-growing list of identity providers including Azure Active Directory, Facebook, Google, Twitter, and Microsoft Account, and leverage Mobile Apps to add authentication to your app in minutes.

•**Offline Sync:** Mobile Apps makes it easy for you to build robust and responsive apps that allow employees to work offline when connectivity is not available, and synchronize with your enterprise backend systems when devices comes back online. Offline sync capability is supported on all client platforms and works with any data source including SQL, Table Storage, Mongo, or Document DB, and any SaaS API including Office 365, Salesforce, Dynamics, or on-premises databases.

**• Push Notifications:** Mobile Apps offers a massively scalable mobile push notification engine, Notification Hubs, capable of sending millions of personalized push notifications to dynamic segments of audience using iOS, Android, Windows, or Kindle devices within seconds. You can easily hook Notification Hubs to any existing app backend, whether that backend is hosted on-premises or in the cloud.

**• Auto Scaling:** App Service enables you to quickly scale-up or out to handle any incoming customer load. Manually select the number and size of VMs or set up auto-scaling to scale your mobile app backend based on load or schedule.

Mobile App endpoints are, at their simplest, REST APIs and can be used on a wide variety of platforms and with a wide variety of devices. Client SDKs are available, however, if you like to connect your mobile application to a Mobile App instance for its backend data.



Mobile client SDKs are available for the following platforms:

**• Xamarin Android/IOS**

**• Android Native**

**• IOS Native**

**• Windows Store**

**• Windows Phone**

**• .NET**

**• HTML**

### Serverless Processing

This section briefly introduces Azure Functions and the concept of Serverless processing.

After completing this lesson, you will be able to:

• Explain how serverless compute can potentially save costs.

• Describe the Azure Functions service.

**Serverless Processing**

Azure Functions are the newest kind of App available in App Services. Function Apps are designed to make it faster to process events  
and distribute output to bound services.

     **Note**: Function Apps were implemented using the existing code and base functionality for Azure WebJobs. To this end, you can go to GitHub and view the Azure WebJobs repositories to see the open-source implementation of Function Apps.

Function Apps can be created using one of two models:

• **App Service**: The App Service model allows you to use the already familiar App Service Plan paradigm with Function Apps. In this model, Function Apps exist as simply another kind of app within an App Service Plan.

• **Consumption:** The consumption model allows you to pay for Function Apps based on execution time as opposed to having a dedicated App Service Plan. Function Apps are billed at a rate specific to Gigabyte Seconds after a specific free monthly allocation threshold has been met.  
Function Apps share a lot of functionality with other App types such as Web Apps. You can configure App Settings, Connection Strings, AlwaysOn, Continous Deployment and many other settings that are configurable in a Web App instance.

     **Note**: In order to guarantee that your Function App will respond to requests as quickly as possible, you should enable the AlwaysOn feature for your Function App whenever possible.

### Event-Based Triggers

Azure Functions must be initiated in some way before they can begin running code or processing data. A Trigger is anything that invokes an Azure Function. Azure Functions have triggers for many different scenarios including:

• Creation of Blobs

• Changes to data in Cosmos DB

• External File or Table changes

• HTTP requests

• OneDrive or Excel files

• E-mail messages

• Mobile Apps

• SendGrid e-mails

• Twilio Text Messages

Additionally, Azure Functions can also be triggered by existing messaging services including:

• Service Bus

• Storage Queues

• Logic Apps

• Event Grid

• Event Hubs

• Webhooks

Azure Functions is unique in the sense that all of these triggers can be configured with minimal or no code. The developer team can focus on authoring the application.

### API Management

This section introduces API Management and Logic Apps as integration solutions available on the Azure platform.

After completing this section, you will be able to:

• Describe API Management in the context of a REST API B2B solution.

• Use Logic Apps for sophisticated integration among application components.

**API Management**

Azure API Management helps organizations publish APIs to external, partner and internal developers to unlock the potential of their data and services. Businesses everywhere are looking to extend their operations as a digital platform, creating new channels, finding new customers and driving deeper engagement with existing ones. API Management provides the core competencies to ensure a successful API program through developer engagement, business insights, analytics, security and protection.

To use API Management, administrators create APIs. Each API consists of one or more operations, and each API can be added to one or more products. To use an API, developers subscribe to a product that contains that API, and then they can call the API's operation, subject to any usage policies that may be in effect.

### Logic Apps

Logic Apps are workflows, designed using JSON, that can connect various components of your application together using minimal or no-code. Logic Apps can integrate with existing services using built-in connectors. There are connectors available for popular services such as:

• SQL Server

• OneDrive

• Dropbox

• Twilio

• SendGrid

• Cosmos DB

• Event Grid

• Event Hubs

• Storage Blobs, Queues and Tables

• Mobile Apps

Logic Apps can also integrate with custom APIs that are deployed as API Apps in your subscription. The main components of a Logic App are as follows:

• **Workflow**: The business process described as a series of steps, in an acyclic graph (loops are not supported).

• **Triggers**: The step that starts a new workflow instance.

• **Actions**: A step in a workflow, typically a Connector or a custom API App.

• **Connector**: A special case of API App ready made to integrate a specific service (e.g., Twitter) or data source (e.g., SQL Server). You can think of Logic Apps as being like toy bricks for integration- you select your bricks and snap together an app.  
For example, you could use the Twitter Connector to read tweets of interest and write them to SQL Database using the Microsoft SQL Connector. Additionally, you could use the Azure Service Bus Connector to queue processing that you will run on HDInsight against some big data, and then send the result via an SMS message using the Twilio connector.

All of this is performed by configuring the connectors, declaratively instead of by writing code.



### Best Practices

Azure provides the capability to create scalable and performance aware application using several PaaS Service together. Using a serverless model we can create global scale and high performance web applications. This section outlines how the App Service and related services can work together to create scalable and performance web applications.

After completing this section you will be able to:

• Best Practices in creating performance aware web applications.

• How to scale your web application.

• Understand how multi-region apps can be built in Azure.

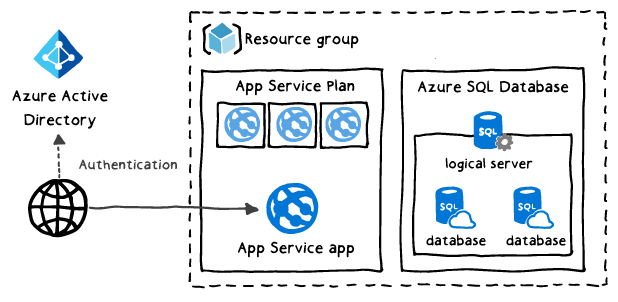
• How to create a business continuity plan.

**Best Practices**

Azure allows you to create state of the art deployments for your web applications. Both Front-end and Back-end. Using the PaaS service App Service, the challenge of creating complex, global high-scalable solutions is simplified. Azure provides a set of services that will simplify the creation of such solutions. These services range from App Service, Traffic Manager, Redis Cache, Queues, Cosmo DB, etc.

### Basic Web Application

In this architecture, we start by using the App Service that can have more than one application running in the same compute. We also have a logic SQL Server to store our data. It’s recommended to use the Standard or Premium tiers in order to support autoscale and SSL. It’s also a recommendation to host both the app services as the SQL database in the same Azure region in order to minimize network latency.



### Scaling

In Azure App Service, when an administrator needs to increase the performance he can do it by changing the number of instances running in the App Service Plan (scale out) or by changing to a higher pricing tier (scale up). Scaling up provides more CPU, memory, disk space and features, like autoscale, deployment slots, etc. Scaling out increases the number of instances running the application. This scaling can be manual, if you’re using the Basic Tier, or automatically, if you’re using the Standard or upper service tiers. Standard tier allows to scale up to 10 instances and if you still need more instances you can go to the Isolated tier where you can scale up to 100 instances. Scaling an application does not require you to change your application code.

In this architecture we had separated our two applications tiers (Web app and an REST API) into different App Service Plans. This way we can scale each tier individually. We had created a Web Job in order to process long-running task in the background, this is also accomplished by using a Message Queue to control the work to be done and already processed.

### Traffic Manager

Using a multi-region architecture provides a higher availability of your applications. If there is an outage in the primary datacenter, using the Traffic Manager service is possible to redirect the traffic to a secondary data center that is available, this way the application will not suffer an outage, reinforcing the overall application uptime. In a multi-region model, we have three main components:

• Active and Standby regions

• Traffic Manager

• SQL Database Geo-replication

In a multi-region model, we have higher global availability of our solutions. This is accomplished by having our solution replicated in two regions and having the Traffic Manager routing incoming traffic to the active region, but in case of failure of this region, this service will failover to the standby region. The SQL Database active geo-replication feature also plays an essential role in this architecture. It creates a replica in the standby region and if the primary database fails or needs to be taken offline, we can failover to one of the up to four replicas.

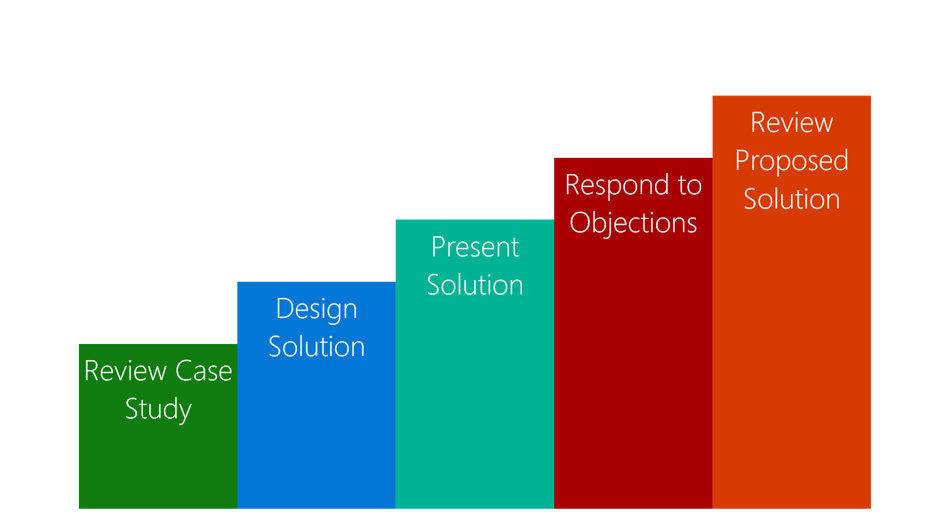
### Case Study Overview

In this case study, we will look at a customer problem that requires an architectural recommendation. After this case study, you should:

• Identify customer problems as they are related to networking.

• Design a solution that will meet the customer’s objectives.

• Ensure your designed solution accounts for customer objections.



**Who Is the Customer?**

With Corporate Headquarters in Phoenix Arizona, the Crazy Taxi Cab Company has quickly risen to be the premier provider of private low-cost transportation in Arizona. Bucking industry norms, Crazy Taxi Cab Co drivers are company employees who work as a team, rather than independent contractors who essentially compete with each other for fares. The founding partners believed this would allow its drivers to focus on providing a great customer experience as opposed to simply “racing to the finish line.” Crazy Taxi has developed a reputation for having fast, reliable, and friendly service, due largely in part to their extensive network of drivers, and their well-executed, albeit radio based dispatching approach.  
Crazy Taxi is drowning in success. While dispatchers are reaching out to drivers to pick up customers who have called in, new callers often find themselves on hold, waiting for their calls to be answered. The executives at Crazy Taxi realize that they need to modernize their operation or risk losing business in the future. Their Chief Operating Officer Christopher Giovanni states that “while we function like a well-oiled machine, we’re still running on 20th century equipment and we are already seeing signs that this is eroding our advantage over the competition…we need to bring our operation into the 21st century and that starts with FastRide.”

**What Does the Customer Already Have?**

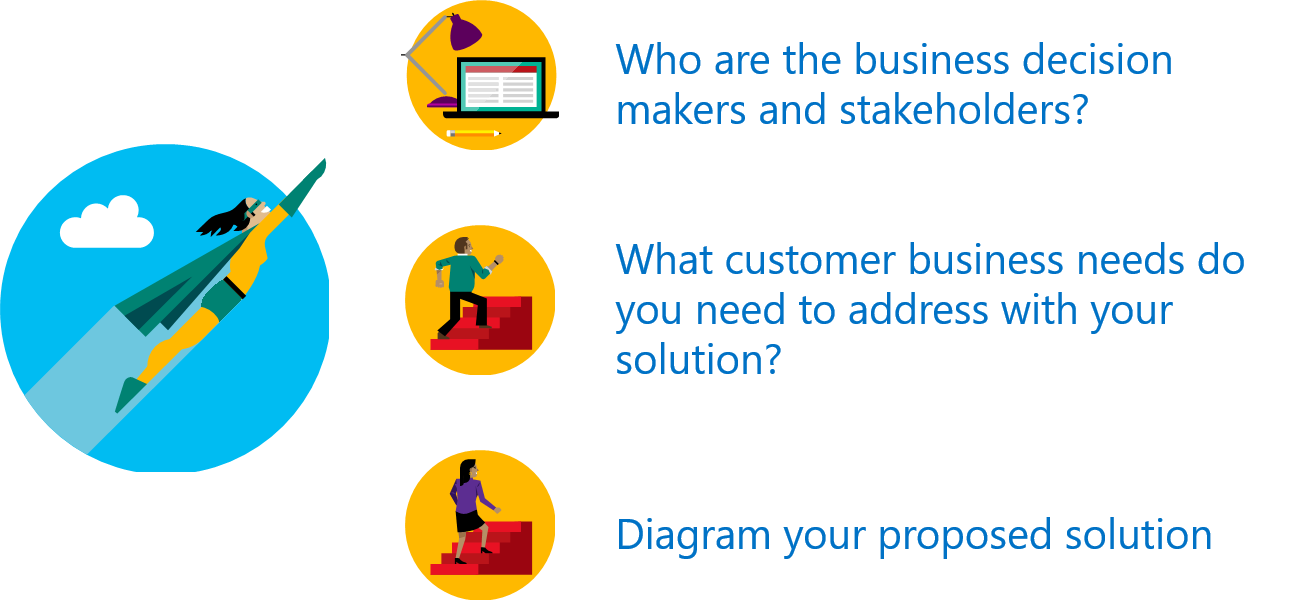
Crazy Taxi does not want to manage a more complex of a system than absolutely necessary. They already have three web and app developers, who have built their marketing pages as well as few proof of concept apps for iOS and Android, and one of them is quite savvy with .NET and brings some backend experience to the table. However, the executives all agree that this one developer cannot deliver the entire backend alone, so they are looking for a solution that provides them a “back end in a box” as they cannot afford to hire any more developers. Additionally, headquarters wants their IT team to have easy visibility into the health of the system, and more importantly that the system can take care of itself. Specifically, they are familiar with the notion of pay-as-you-go billing, and would love a solution that operates at nominal cost during their average daytime load, but on Friday night and Saturday night automatically handles the increased demand, albeit at an additional cost.

**What Is the Customer's Goal?**

FastRide is a software solution that the company has been planning to build that uses the GPS of 4G enabled mobile devices in each Crazy Taxi to monitor the location of each vehicle in the Crazy Taxi fleet. By being able to visualize how their fleet is moving throughout the city in real time, FastRide will allow Crazy Taxi Cab Co. to optimize their driver coverage throughout the city. When a new customer calls in, the telephone dispatcher enters their pickup location and the FastRide system identifies the closest available driver, instead of requiring dispatch to manually poll all of the drivers over the radio. While stationary, a driver accepts a fare using the FastRide app on his mobile device. FastRide will provide the driver with the pick-up details such as location, passenger name and callback phone (if available). Upon arrival the destination, the device will compute the fare automatically based on time in transit and distance driven, all with intent of minimizing driver distraction while driving due to interacting with the device. The fare information should be collected on the device and uploaded to a central database in near real-time. Should the driver enter a pocket without cellular connectivity, the device should store the data offline and sync it as soon as connectivity is restored.

The fare data is associated with each driver, who logs in in thru his or her app using his or her own Microsoft Account, Google, Twitter or Facebook credentials. It is the intent that driver fare data will be used to track the driver’s progress against established company goals, which both the company and driver access in the form of reports. Because these reports may grow to involve more complex computations (such as comparing the driver’s selected route to similar routes taken by other drivers), these data powering the reports will be computed nightly and made available the following day.

While initially notifications would be sent only to drivers about potential fares they can choose to accept, Crazy Taxi is looking forward to when they can offer their patrons a mobile app that can receive notifications about when the driver is in route, ETA updates and any messaging from the driver to the customer. They would like to be certain that the system they build on today has the capacity to meet those increased demands in the future.



**What Does the Customer Need?**

• A back-end as a service that provides both data storage, API support and push notifications.

• A solution that is quick to implement with a small team that has limited back-end development experience and capacity.

• A back-end solution that can be implemented using .NET.

• A solution that is easy to monitor for health and telemetry data.

**What Things Worry the Customer?**

• Doesn’t Azure Mobile Apps only work on Windows devices?

• Our development team doesn’t know node.js. We had heard mention of Mobile Apps, but thought it only supported JavaScript backends.

• Our development team seems to think implementing push notifications using Apple and Android apps directly is easy, but we (as the executives) aren’t so sure. How difficult can it be?

• Can’t we just build all of our backend using Azure Web Apps?

### Case Study Solution

**Preferred Target Audience**

Operations Management Team led by Christopher Giovanni, Chief Operating Officer at Crazy Taxi Cab Co.

**Preferred Solution**

Crazy Taxi Cab Co. liked the idea of a highly scalable, agile solution that they could both execute on and manage with a small IT team with limited back-end development experience in .NET. The company’s Microsoft representative introduced them to Azure Mobile Apps, a cloud based mobile platform that provides backend CRUD data services, offline data sync for mobile devices, custom REST API services, push notifications and services for login via a social identity provider. This was exactly the “back-end in a box” that they were looking for.

Crazy Taxi Cab Co. leveraged the many features of Mobile Apps available to them with the Standard tier in order to minimize their backend development burden and accelerate solution delivery. The implementation of the proposed Azure Mobile Apps solution would create a strategic partnership that will help Crazy Taxi Cab Co. to overcome its challenges with:

• Minimizing system downtime.

• Sending multiple user specific notifications to mobile devices.

• Managing user accounts leveraging social identity providers, such as Microsoft Account, Facebook, and Twitter.

• 24/7 (secure) data accessibility throughout the Crazy Taxi Cab Co. network.

• Scalability in software solutions in an agile marketplace.

As the FastRide system continues to be improved upon, Mobile Apps will help to inject velocity into the development cycle by providing a mobile back end to the application. Mobile Apps offers cross platform compatible components, which gives Crazy Taxi Cab Co. the flexibility to change their mobile platform as the needs of their business dictate. This “back-end as a service” approach will allow Crazy Taxi to focus on building an app that merges the right functionality with a great user experience for each market the operate in.

By utilizing push notifications, Crazy Taxi Cab Co. can optimize their customer pickup messages through the FastRide app. This allows for faster, and more streamlined in-app communication. When a new fare’s pickup address is entered into FastRide by a dispatcher, or the customer facing mobile app, Mobile Apps will enable FastRide to automatically send a notification to the closest available driver—eliminating the need for manual notifications. Since push notifications can be managed, each base will have control over the messages sent to its drivers.

To propose a more complete solution and ensure deployment success, it would be helpful to know:

• Type and operating system of tablets being used.

• Expected product life of current tablet choice.

• Current number of dashboard tablets.

• Projected number of tablets after the planned expansion this year and next year.

• Current average number of fares completed per day, week, month, year.

• Projected average number of fares completed per day, week, month, year after the expansion into new markets.

• Rate of growth across bases (customer and driver).

• Other software products used to operate the company.

Understanding these details and decisions will help identify the current and future software, hardware, and infrastructure needs of Crazy Taxi Cab Co., and to provide solutions that are consistent with their short and long-term business goals.

### High-Level Architecture

Crazy Taxi Cab Co. leveraged the many features of Mobile Apps available to them with the Standard tier in order to minimize their backend development burden and accelerate solution delivery.

**• Authentication:** Drivers login to the FastRide app on their device using their Microsoft, Google, Twitter or Facebook credentials, the federated login process being handled by Mobile Apps in conjunction with the aforementioned identity providers.

**• Notifications**: Once logged in, the app registers with Mobile Apps, associating the driver’s identity with the Notification Hub (associated with the Mobile App). In this way, Crazy Taxi dispatch can send broadcast notifications to all drivers, but still be able to send targeted Fare Alert messages to a particular driver.

By having Mobile Apps in place with Push Notifications, Crazy Taxi Cab Co. is well positioned in the future to light up the ability to deliver a customer-oriented app that deliver push notifications to customers informing them of events relevant to their pickup.

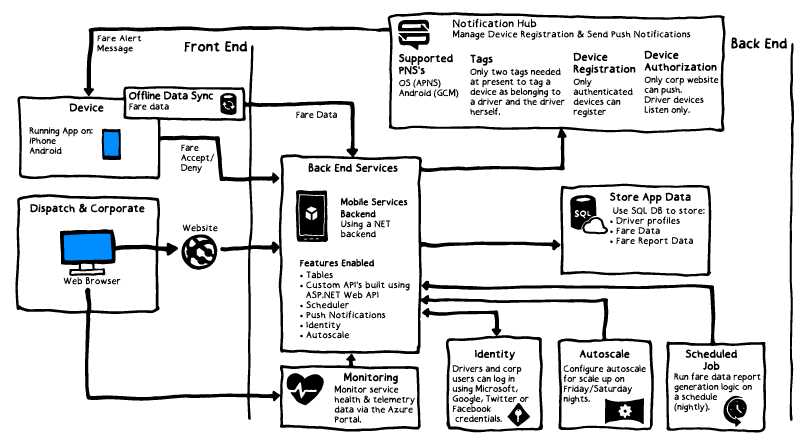
**• Offline Data:** For the driver’s iOS and Android devices, in the construction of the FastRide app, they leveraged the native client Offline Data Sync functionality for synchronizing Fare Data when temporarily disconnected from the cellular network. This Fare Data is stored using Tables in Mobile Apps, which ultimately rests as relational tables in a SQL Database. This SQL Database also stores the driver profiles (that associate social credentials with driver profile information).

• **Back End Custom Services**: When a driver accepts or declines a fare received via a push notification, that message is sent using a Custom REST API hosted by Mobile Apps and built using ASP.NET Web API.

• **Front-End Website:** Dispatch uses a website hosted in Azure Websites to manage the taxi cab dispatch process. The Notification Hub is configured so that only the dispatch website is allowed to send push notifications  
to the drivers (the FastRide app for drivers is Listen only).  
• **Monitoring**: Crazy Taxi corporate IT monitors the health of the solution using the Dashboard for the Mobile App or Website in the Azure Portal. To assist the IT team with visibility into the health of the system, they should configure monitoring endpoints, again using the Azure Portal, on their website and Mobile Apps and enable e-mail alerts should the Response Time and Uptime metrics for those fall below values they deem acceptable.  
•**Scaling Configuration**: They have configured Autoscale on their Mobile App, via the Scale tab in the portal, to double the number of instances it uses on Friday and Saturday night in order to handle the increased load, then return back to their normal instance count for the rest of the week.

• **Backend Jobs**: They have also created a Mobile Apps Scheduled Job that processes the Fare Data on nightly basis to generate the data sets that power the Fare Reports. This data is stored in the same SQL Database that stores all the other data used by the solution.

**Preferred Solution Diagram**



### Checklist of Preferred Objection Handled

**• Doesn’t Azure Mobile Apps only work on Windows devices?**

Azure Mobile Apps provides native clients for iOS, Android, Xamarin, PhoneGap, Sencha and Appcelerator in addition to Windows universal C#, Windows universal JavaScript and Windows Phone. In addition, Azure platform services offer REST APIs that extend the reach to platforms for which there is not a native API, but are capable of making REST style requests.

**• Our development team doesn’t know node.js. We had heard mention of Mobile Apps, but thought it only supported JavaScript backends**.

Mobiles Services supports using .NET for the backend logic, and node.js (or JavaScript logic) does not have to be used anywhere in the backend code.

**• Our development team seems to think implementing push notifications using Apple and Android apps directly is easy, but we (as the executives) aren’t so sure. How difficult can it be?**

While using the Push Notification System of a particular device platform directly is typically made fairly simple by the provider of that platform (e.g., iOS apps have a straightforward API for leveraging Apple’s Push Notification System), this simplicity is only true for the individual device and does not address the complete solution that requires at minimum a backend managing device registrations at scale and sending push notifications cross platforms in a timely fashion. Azure Mobile Apps provides that backend functionality, which can be easily scaled to meet demand.

**• Can’t we just build all of our backend using Azure Web Apps?**

Azure Web Apps is effectively a superset of Mobile Apps and so can be used to implement the backend for Mobile Applications. However, Web Apps do not deliver the services tailored for the mobile application scenario, requiring the developers to write their own logic to integrate with Notification Hubs, SQL Database, Identity services and WebJobs. Additionally, Mobile Apps is prescriptive in the patterns used for developing custom API’s, and so speeds the development of such API’s by allowing the development efforts to focus on the business logic instead of dealing with structural and hosting decisions. These become important factors to consider when taking into account the development team size, capabilities and timeframe.

### Proof Of Concept (POC) Checklist

The primary items a Proof of Concept for this solution could demonstrate include:

• Scalability / Scheduled Autoscale.

• Mobile Apps ease of integration (e.g., the backend in a box).

• Streamlined communication with Push notifications.

• Integration of social identity platforms to aid in customer authentication and profile management.

• Device offline data storage and synchronization.

• Monitoring solution health.

The Proof of Concept will be considered successful if the Crazy Taxi Operations Management Team believes they can realize value in:

• Speeding up the delivery of the overall solution.

• Push notifications to streamline communication and send fare updates to tablet devices.

• Authenticating users via social media platforms and future benefits of successfully leveraging social media integration.

• Minimizing system downtime by keeping app data in the cloud.

• Scalability in a mobile cloud solution.

The personnel resources you would leverage to build the PoC, may include:

• Partner Resources in the Region or MCS to help assist with migration design and implementation.

• Microsoft Azure CAT for level 300 expertise requests with Azure.

### Lab Steps

#### Online Lab - Deploying Serverless Workloads to Azure

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Create Web App

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
   * In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box below, type **AADesignLab0901-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 3: Create an App Service plan

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this exercise:

RESOURCE\_GROUP\_APP='AADesignLab0502-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder <Azure region> with the name of the Azure region to which you intend to deploy resources in this lab):

LOCATION='<Azure region>'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create the resource group:

az group create --name $RESOURCE\_GROUP\_APP --location $LOCATION

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new App Service plan:

az appservice plan create --is-linux --name "AADesignLab0502-$LOCATION" --resource-group $RESOURCE\_GROUP\_APP --location $LOCATION --sku B2

##### Task 4: Create a Web App instance

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view a list of possible runtimes for a Linux-based App Service web app instance:

az webapp list-runtimes --linux

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new variable which value is a randomly generated string that you will use as the name of a new web app:

WEBAPPNAME1=webapp05021$RANDOM$RANDOM

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new web app using a unique name:

az webapp create --name $WEBAPPNAME1 --plan AADesignLab0502-$LOCATION --resource-group $RESOURCE\_GROUP\_APP --runtime "DOTNETCORE|2.1"

**Note**: In case the command fails due to duplicate web app name, re-run the last two steps until the command completes successfully

1. Wait for the deployment to complete before you proceed to the next task.

##### Task 5: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0502-RG**.
3. On the **AADesignLab0502-RG** blade, click the entry representing the Azure web app you created earlier in this exercise.
4. On the web app blade, click the **Browse** button at the top of the blade.
5. Review the default page generated by Azure App Service.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

**Review**: In this exercise, you created a Linux-based App Service Plan that contained a blank web app.

#### Exercise 2: Deploy Web App code

##### Task 1: Deploy code with a Web App Extension using an Azure Resource Manager template and GitHub

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this exercise:

RESOURCE\_GROUP\_APP='AADesignLab0502-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab0502-RG'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new variable which value is a randomly generated string that you will use as the name of a new web app:

WEBAPPNAME2=webapp05022$RANDOM$RANDOM

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new web app using a unique name:

az webapp create --name $WEBAPPNAME2 --plan AADesignLab0502-$LOCATION --resource-group $RESOURCE\_GROUP\_APP --runtime "NODE|9.4"

**Note**: In case the command fails due to duplicate web app name, re-run the last two steps until the command completes successfully

1. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
2. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T03\Module\_03\Labfiles\Starter\** folder, select the **github.json** file, and click **Open**. The file contains the following Azure Resource Manager template:

{ "$schema": "http://schemas.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": { "webAppName": { "type": "string" }, "repositoryUrl": { "type": "string" }, "branch": { "type": "string", "defaultValue": "master" } }, "resources": [ { "apiVersion": "2015-08-01", "type": "Microsoft.Web/sites", "name": "[parameters('webAppName')]", "location": "[resourceGroup().location]", "properties": {}, "resources": [ { "apiVersion": "2015-08-01", "name": "web", "type": "sourcecontrols", "dependsOn": [ "[resourceId('Microsoft.Web/Sites', parameters('webAppName'))]" ], "properties": { "RepoUrl": "[parameters('repositoryUrl')]", "branch": "[parameters('branch')]", "IsManualIntegration": true } } ] } ] }

1. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
2. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T03\Module\_01\Labfiles\Starter\** folder, select the **parameters.json** file, and click **Open**. The file contains the following parameters for the Azure Resource Manager template you uploaded previously:

{ "$schema": "http://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#", "contentVersion": "1.0.0.0", "parameters": { "webAppName": { "value": "$WEBAPPNAME2" }, "repositoryUrl": { "value": "$REPOSITORY\_URL" }, "branch": { "value": "master" } } }

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the GitHub repository hosting the web app code:

REPOSITORY\_URL='https://github.com/Azure-Samples/nodejs-docs-hello-world'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the GitHub repository hosting the web app code and which takes into account any special character the URL might include:

REPOSITORY\_URL\_REGEX="$(echo $REPOSITORY\_URL | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"

**Note**: This is necessary because you will use the **sed** utility to insert this string into the Azure Resource Manager template parameters file. Alternatively, you could simply open the file and enter the URL string directly into the file.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the value of the **webAppName** parameter with the value of the **$WEBAPPNAME2** variable in the parameters file:

sed -i.bak1 's/"$WEBAPPNAME2"/"'"$WEBAPPNAME2"'"/' ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the value of the **repositoryUrl** parameter with the value of the **$REPOSITORY\_URL** variable in the parameters file:

sed -i.bak2 's/"$REPOSITORY\_URL"/"'"$REPOSITORY\_URL\_REGEX"'"/' ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the placeholders were successfully replaced in the parameters file:

cat ~/parameters.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the GitHub-resident web app code by using a local Azure Resource Manager template and a local parameters file:

az group deployment create --resource-group $RESOURCE\_GROUP\_APP --template-file github.json --parameters @parameters.json

1. Wait for the deployment to complete before you proceed to the next task.**Note**: The deployment should take about a minute.

##### Task 2: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0502-RG**.
3. On the **AADesignLab0502-RG** blade, click the entry representing the Azure web app you created in the previous task.
4. On the web app blade, click the **Browse** button at the top of the blade.
5. Review the sample Node.js web application deployed from GitHub.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

##### Task 3: Deploy Code with a Docker Hub container image

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use in this task:

RESOURCE\_GROUP\_CONTAINER='AADesignLab0502-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab0502-RG'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new variable which value is a randomly generated string that you will use as the name of a new web app:

WEBAPPNAME3=webapp05023$RANDOM$RANDOM

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new web app using a unique name:

az webapp create --name $WEBAPPNAME3 --plan AADesignLab0502-$LOCATION --resource-group $RESOURCE\_GROUP\_CONTAINER --deployment-container-image ghost

**Note**: In case the command fails due to duplicate web app name, re-run the last two steps until the command completes successfully

1. Wait for the deployment to complete before you proceed to the next task.**Note**: The deployment should take less than a minute.
2. Close the **Cloud Shell** pane.

##### Task 4: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0502-RG**.
3. On the **AADesignLab0502-RG** blade, click the entry representing the Azure web app you created in the previous task.
4. On the web app blade, click the **Browse** button at the top of the blade.**Note**: If the application does not appear, switch to the web app blade, click **Restart** button at the top of the blade and then click **Browse** again.
5. Review the blog application deployed from Docker Hub.
6. Close the new browser tab and return to the browser tab displaying the Azure portal.

**Review**: In this exercise, you deployed code using an Azure Resource Manager template and a Docker Hub image to App Service web apps.

#### Exercise 3: Deploy a Function App

##### Task 1: Deploy a Function App with code using an Azure Resource Manager template

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click the **Quickstart template** link.
7. In the **Load a quickstart template** pane, in the **Select a template** drop-down list, select the **201-function-app-dedicated-github-deploy** template.
8. Click the **OK** button.
9. Back on the **Edit template** blade, click the **Save** button to persist the template.
10. Back on the **Custom deployment** blade, perform the following tasks:
    * Leave the **Subscription** drop-down list entry set to its default value.
    * In the **Resource group** section, select the **Use existing** option.
    * In the **Resource group** section, ensure that **Create new** option is selected and, in the text box below, type **AADesignLab0503-RG**.
    * In the **App Name** text box, type a unique name for the new Function App.
    * In the **Sku** drop-down list, select the **Basic** option.
    * Leave the **Worker Size** drop-down list set to its default value.
    * Leave the **Storage Account Type** drop-down list set to its default value.
    * Leave the **Repo URL** field set to its default value.
    * Leave the **Branch** text box set to its default value.
    * Leave the **Location** text box set to its default value.
    * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
    * Click the **Purchase** button.
11. Wait for the deployment to complete before you proceed to the next task.**Note**: The deployment should take about a minute.

##### Task 2: View deployment results

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab0503-RG**.
3. On the **AADesignLab0503-RG** blade, click the entry representing the Function App you created in the previous task.
4. On the Function App blade, locate the **Url** entry and click the hyperlink below to see the Function App landing page in a new browser tab.
5. Close the new browser tab and return to the browser tab displaying the Azure portal.

**Review**: In this exercise, you deployed a Function App and code using an Azure Resource Manager template.

#### Exercise 4: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab05')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab05')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 3 Review Questions

**Serverless computing**  
You need to design a serverless framework to host applications.  
 What are the core benefits of Web Apps? What other types of App Services does Azure offer?

Suggested Answer ↓

Web Apps are a fully managed and low friction PaaS offering to host your web application in the Azure platform. The serverless Azure App Services include Web Apps, Mobile Apps, and APIs.

**Serverless computing (Azure Functions)**  
You architect a solution for your company.  
You need to build a serverless processing framework.  
Which Azure service provides serverless processing? What are the two models for using this service?

Suggested Answer ↓

Azure Functions are designed to make it faster to process events and distribute output to services that interact with the functions. You can create Function Apps that use the App Service or Consumption models.

**Integration offerings**  
You are designing a solution that integrates several different applications and services. You plan to implement an Azure Platform-as-a-Service (PaaS) solution.  
What integration solutions does Azure offer?

Suggested Answer ↓

Azure provides API Management and Logic Apps as integration PaaS solutions. Azure API Management helps organizations publish APIs to external partner and internal developers. Logic Apps provide workflows, designed using JOSN, that can connect various components of your appli

# AZ-301T04-A - Designing an Infrastructure Strategy

### Welcome to Designing an Infrastructure Strategy

Welcome to *Designing an Infrastructure Strategy*. This course is part of a series of four courses to help students prepare for Microsoft’s Azure Solutions Architect technical certification exam AZ-301: Microsoft Azure Architect Design. These courses are designed for IT professionals and developers with experience and knowledge across various aspects of IT operations, including networking, virtualization, identity, security, business continuity, disaster recovery, data management, budgeting, and governance.

This course contains the following four modules:

**Module 1** - Application Architecture Patterns in Azure

This module introduces, and reviews common Azure patterns and architectures as prescribed by the Microsoft Patterns & Practices team. Each pattern is grouped into performance, resiliency, and scalability categories and described in the context of similar patterns within the category.

After completing this module, students will be able to:

• Locate and reference the Cloud Design Patterns documentation.

• Locate and reference the Azure Architecture Center.

• Describe various patterns pulled from the Cloud Design Patterns.

**Module 2** - Building Azure IaaS-Based Server Applications (ADSK)

This module identifies workloads that are ideally deployed using Infrastructure-as-a-Service services in Azure. The module focuses on the VM Scale Sets and Virtual Machine services in Azure and how to best deploy workloads to these services using best practices and features such as Availability Sets.

After completing this module, students will be able to:

• Design an availability set for one or more virtual machines.

• Describe the differences between fault and update domains.

• Author a VM Scale Set ARM template.

• Join a virtualized machine to a domain either in Azure or on a hybrid network.

**Module 3** - Networking Azure Application Components

This module describes the various networking and connectivity options available for solutions deployed on Azure. The module explores connectivity options ranging from ad-hoc connections to long-term hybrid connectivity scenarios. The module also discusses some of the performance and security concerns related to balancing workloads across multiple compute instances, connecting on-premise infrastructure to the cloud and creating gateways for on-premise data.  
After completing this module, students will be able to:

• Describe DNS and IP strategies for VNETs in Azure.

• Compare connectivity options for ad-hoc and hybrid connectivity.

• Distribute network traffic across multiple loads using load balancers.

• Design a hybrid connectivity scenario between cloud and on-premise.

**Module 4** - Integrating Azure Solution Components Using Messaging Services

This module describes and compares the integration and messaging services available for solutions hosted on the Azure platform. Messaging services described include Azure Storage Queues, Service Bus Queues, Service Bus Relay, IoT Hubs, Event Hubs, and Notification Hubs. Integration services include Azure Functions and Logic Apps.

After completing this module, students will be able to:

• Compare Storage Queues to Service Bus Queues.

• Identify when to use Azure Functions or Logic Apps for integration components in a solution.

• Describe the differences between IoT Hubs, Event Hubs and Time Series Insights.

#### Prerequisites

This course requires that students have the following knowledge and skills:

• Create resources and resource group in Azure.

• Manage users, groups, and subscriptions in an Azure Active Directory instance.

• Build an Azure Virtual Machine with related resources.

• Manage containers and blobs stored in an Azure Storage account.

• Create App Service Plans and manage apps related to the plan.

• Configure an Azure Virtual Network and enable S2S and P2S connectivity.

• Protect networked application components using Network Security Groups.

• Automate everyday Azure resource tasks using Azure CLI or Azure PowerShell.

• Deploy an Azure SQL, MySQL, Postgres or Cosmos database instance.

• Monitor existing Azure solutions using built-in metrics, Application Insights, or Operational Insights.

### Why Patterns?

This module introduces and reviews common Azure patterns and architectures as prescribed by the Microsoft Patterns & Practices team. Each pattern is grouped into performance, resiliency, and scalability categories and described in the context of similar patterns within the category.

##### Objectives

After completing this module, students will be able to:

• Locate and reference the Cloud Design Patterns documentation.  
• Locate and reference the Azure Architecture Center.  
• Describe various patterns pulled from the Cloud Design Patterns.

This lesson introduces the resources available on Microsoft’s websites that deal with the architecture and design of solutions hosted on the Azure platform.

Many common problem areas in Computer Science have been explored and experienced  
by a wide variety of professionals. As with any discipline, best practices have been conceptualized, proven and prescribed as the most effective or efficient ways to solve common problems. Many professionals rely on these best practices so that they can work more efficiently and focus on obstacles unique to their actual problem space. In software engineering, these best practices are commonly referred to as design patterns.

*In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is [instead] a description or template for how to solve a problem that can be used in many different situations.*

*"Software Design Pattern."* Wikipedia.

There are many different patterns that already exist in the industry. In this course, we will focus on patterns that are Domain-specific to the cloud. While examples of these patterns may be implemented in Azure or using the .NET framework, most cloud patterns are language-agnostic in their design and can be implemented universally across a wide variety of cloud providers, programming frameworks or operating systems.

### Microsoft Patterns & Practices

Microsoft Patterns & Practices is a long-standing group at Microsoft that collects, authors and shares best practice documentation for the benefit of the community. The team’s intention is to provide anyone a stable technical foundation to start from when building elegant solutions built in any language, hosted on any platform. Most solutions from patterns & practices incorporates many different Microsoft products and technologies as part of the overall solution. The team has been responsible for well-known solution libraries like Prism, Unity and Enterprise Library.

**Cloud Design Patterns**

The patterns & practices team at Microsoft has collected twenty-four design patterns that are relevant when designing the architecture of a cloud application. Each pattern includes a brief discussion of the benefits, considerations and implementation of each pattern. The collection of patterns is not meant to be comprehensive and is instead focused on the most popular design patterns for cloud applications.

The guide consists of a collection of web pages each individually focused on a pattern. The pages are broken down into sections describing the problem domain, a high-level technical solution, how these patterns solve a problem, considerations when using the pattern, an example implementing the pattern and when the pattern is suitable.

The patterns included in this guide can be used in many ways including:

• Implement the sample code as a starting point for your own cloud application.

• Use the Context, Problem and Solution sections of a pattern's web page as discussion points during an architectural design session.

• Use the Example section of a pattern's web page to teach colleagues about a design pattern.

• Use the Issues and Considerations section of a pattern's web page to identify common issues in your problem space.

The remainder of this module will focus on a subset of patterns from the Cloud Design Patterns documentation and explain why they are important to understand and how they relate to the decisions you will make as an Azure architect.

Microsoft Patterns & Practices shares much of their documentation, projects and findings today on **GitHub**:

<https://github.com/mspnp>

For example, the Microservices Reference Implementation shares best practices when designing a microservices solution running on Azure using Kubernetes:

<https://github.com/mspnp/microservices-reference-implementation>

### Azure Architecture Centre

The Azure Architecture Centre is a landing page that contains links to documentation written by Microsoft patterns & practices, the Azure product groups and Azure subject-matter experts on architecting solutions for the Azure platform.

The centre contains links to resources such as:

• Reference Azure Architectures

• Cloud Design Patterns

• Best Practices for Cloud Applications

• Azure Building Blocks

• Running SharePoint Server on Azure Guidance

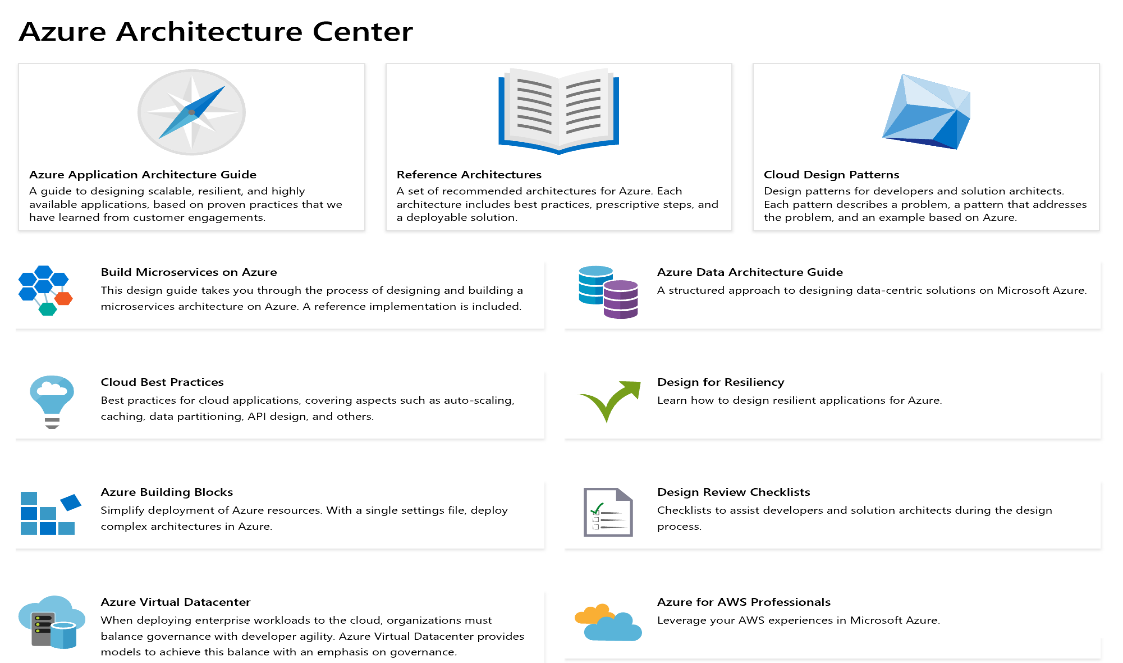
• Performance Antipatterns

• Azure and SQL Server Customer Advisory

• Identity Management for Multitenant Applications

• Azure for AWS Professionals

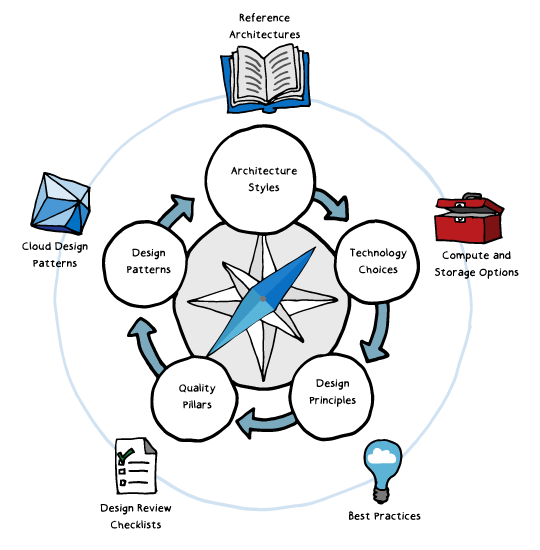
• Microservice Guidance for Azure using Kubernetes and Azure Container Service



Azure Architecture Center: <https://docs.microsoft.com/en-us/azure/architecture/>

##### Architecture Center Guide

This guide presents a structured approach for designing applications on Azure that are scalable, resilient, and highly available. It is based on proven practices that we have learned from customer engagements.

The guide is intended for application architects, developers, and operations teams. It's not a how-to guide for using individual Azure services. After reading this guide, you will understand the architectural patterns and best practices to apply when building on the Azure cloud platform. You can also download an [e-book version of the guide](https://azure.microsoft.com/campaigns/cloud-application-architecture-guide/).

Architecture Center Guide: <https://docs.microsoft.com/azure/architecture/guide/>

### Stateless Applications

#### Partitioning Workloads

This lesson introduces patterns related to the performance of applications and workloads on Azure.

Lesson Objectives

After completing this lesson, you will be able to:

• Describe the Valet Key pattern.

• Describe the Command-Query Responsibility Segregation pattern.

• Describe the Throttling pattern.

A modular application is divided into functional units, also referred to as modules, which can be integrated into a larger application. Each module handles a portion of the application's overall functionality and represents a set of related concerns. Modular applications make it easier to design both current and future iterations of your application. Existing modules can be extended, revised or replaced to iterate changes to your full application.

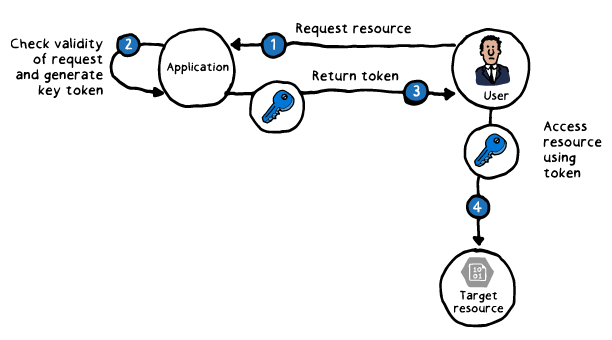
Modules can also be tested, distributed and otherwise verified in isolation. Modular design benefits are well understood by many developers and architects in the software industry.

     Note: What are some of the ideas that you have for modular and scalable applications? These ideas do not necessarily have to be web applications.

### The Valet Key Pattern

Cloud applications might allow some resources to be downloaded by the end users. This can result in a tight dependency on a storage mechanism and apart from being an overhead to the cloud application itself. Consider a scenario where a hosted web service offers an expansive set of functionality, which includes downloading and streaming images to the user. In such a scenario, the application could spend a considerable amount of its processing power and memory just to download the images. Because of specific load patterns the application may limit resources to other tasks to ensure that it handles all of the download requests. This is not an ideal situation and can occur if you have a scenario where your application needs to validate the user before downloading protected images.

The Valet Key pattern introduces a mechanism where your application can validate the user’s request without having to manage the actual download of the media. Both can occur while keeping the media private and locked away from anonymous access.



The pattern specifies that the client needs to send a request to the web service or application if it wants to access a media asset. The application then validates the request. If the request is valid, the application goes to the external storage mechanism, generates a temporary access token, and then provides the URI of the media asset to the client, along with the temporary access token. The application has now processed this request and can now move on to other requests. It is up to the client application or browser to use the URI and temporary access token to download the media asset. Now, the storage service is responsible for scaling up to manage all the requests for media while the application can focus solely on other functionality.

    Reference Link: <https://docs.microsoft.com/azure/architecture/patterns/valet-key>

### Transient Errors

This lesson introduces patterns related to the resiliency of workloads in Azure.

**Lesson Objectives**

After completing this lesson, you will be able to:

• Describe the Circuit Breaker pattern.

• Describe the Retry pattern.

• Describe the Queue-Based Load Leveling pattern.

One of the primary differences between developing applications on-premises, and in the cloud, is the way you design your application to handle transient errors. Transient errors are as errors that occur due to temporary interruptions in the service or due to excess latency. Many of these temporary issues are self-healing and can be resolved by exercising a retry policy.

Retry policies define when and how often a connection attempt should be retried when a temporary failure occurs. Simply retrying in an infinite loop can be just as dangerous as infinite recursion. A break in the circuit must eventually be defined so that the retries are aborted if the error is determined to be of a serious nature and not just a temporary issue.

Transient Fault Handling is a pattern that makes your application more resilient by handling temporary issues in a robust manner. This is done by managing connections and implementing a retry policy. This pattern is already implemented in many common .NET libraries such as Entity Framework and the Azure software development kit (SDK). This pattern is also implemented in the Enterprise Library in such a generic manner that it can be brought into a wide variety of application scenarios.

     **Reference Link**: <https://docs.microsoft.com/aspnet/aspnet/overview/developing-apps-with-windows-azure/building-real-world-cloud-apps-with-windows-azure/transient-fault-handling>

### The Retry Pattern

Applications can experience transient errors when connecting to external services irrespective of whether they are hosted in the cloud or they use external cloud-hosted services. A retry pattern can be used to implement a limited number of retries if the connection error implies that it is a temporary issue. When the initial connection fails, the failure reason is analysed first to determine if the fault is transient or not. If the failure reason or the error code indicates that this request is unlikely to succeed even after multiple retries, then retries are not performed at all. Retries are performed until either the connection succeeds or a retry limit is reached. The limit is in place to avoid a situation where the retries might go on infinitely. The retries are typically performed after a timed delay and the delay might remain constant or increase linearly or exponentially after each retry. If the retry limit is reached, the connection is said to have failed in a non-transient manner.

There are many external libraries that implement the Retry Policy pattern including:

• Entity Framework

• Enterprise Library - Transient Fault Handling Application Block     

 **Reference Link:**<https://docs.microsoft.com/azure/architecture/patterns/retry>

### Queues

Queueing is both a mathematical theory and also a messaging concept in computer science.  
In cloud applications, queues are critical for managing requests between application modules in a manner such that it provides a degree of consistency regardless of the behavior of the modules.

Applications might already have a direct connection to other application modules using direct method invocation, a two-way service, or any other streaming mechanism. If one of the application modules experiences a transient issue, then this connection is severed and it causes an immediate application failure. You can use a third-party queue to persist the requests beyond a temporary failure. Requests can also be audited independent of the primary application as they are stored in the queue mechanism.

### Asynchronous Messaging

Many software curriculums teach that separating your application into smaller modules will make it easier to manage the application in the long term. Modules can be swapped, modified and updated without having to update the entire application. Partitioning your workloads into modules also carries another benefit of allowing each logic center in your application to scale in isolation.

If you have a web application that allows people to upload images for processing, your image processing module can become CPU intensive and easily account for the majority of your CPU time and disk usage. By separating the image processing module out to another distinct server (or set of servers), you can scale this module in isolation without having to modify, scale or change the module that serves the web pages. It then becomes very important to figure out how to communicate between these modules.

Messaging is a key strategy employed in many distributed environments such as the cloud. Messaging enables applications and services to communicate and cooperate, and can help to build scalable and resilient solutions. Messaging supports asynchronous operations, enabling you to decouple a process that consumes a service from the process that implements the service.

**Problem: Handling Variable Quantities of Requests**

An application running in the cloud may be expected to handle a large number of requests. The number of requests could vary significantly over time for many reasons. A sudden burst in user activity or aggregated requests coming from multiple tenants may cause unpredictable workload. At peak hours a system might need to process many hundreds of requests per second, while at other times the number could be very small. Additionally, the nature of the work performed to handle these requests might be highly variable.

Using a single instance of the consumer service might cause that instance to become flooded with requests or the messaging system may be overloaded by an influx of messages coming from the application.  
   
**Solution: Asynchronous Messaging with Variable Quantities of Message Producers and Consumers**

Rather than process each request synchronously, a common technique is for the application to pass them through a messaging system to another service (a consumer service) that handles them asynchronously. This strategy helps to ensure that the business logic in the application is not blocked while the requests are being processed.

A message queue can be used to implement the communication channel between the application and the instances of the consumer service. To handle fluctuating workloads, the system can run multiple instances of the consumer service. The application posts requests in the form of messages to the queue, and the consumer service instances receive messages from the queue and process them. This approach enables the same pool of consumer service instances to handle messages from any instance of the application.

**Example Implementation in Azure**

In Azure, the competing consumers pattern can be easily implemented using either Storage Queues or Service Bus Queues.

1. The IoT device sends an HTTP request to the distributed Azure API App instances with temperature, barometric pressure and humidity information collected throughout the day.

2. The individual application instance that receives the request uses an HTTP request to add a message to the Storage Queue with the weather metadata included as part of the message body.

3. The consumer services use the Azure SDK for Ruby to poll the queue to see if any messages are available.

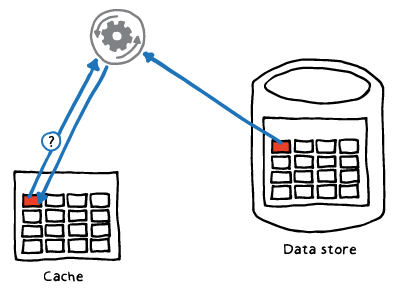
4. One of the consumer services' instances will receive the previously generated message and can now process the message. If the message has been successfully processed, the instance will use the SDK to mark the message as deleted in the queue. If the instance crashes or times out, the queue message will eventually be available to other instances after a visibility timeout period has elapsed.

### Cached Data Consistency

Applications use a cache to optimize repeated access to information held in a data store. However, it is usually impractical to expect that cached data will always be completely consistent with the data in the data store. Applications developers should consider a strategy that helps to ensure that the data in the cache is up to date as far as possible, but can also detect and handle situations that arise when the data in the cache has become stale.

**Solution: Read/Write-Through Caching**

Many commercial caching systems provide read-through and write-through/write-behind operations. In these systems, an application retrieves data by referencing the cache. If the data is not in the cache, it is transparently retrieved from the data store and added to the cache. Any modifications to data held in the cache are automatically written back to the data store as well.



For caches that do not provide this functionality, it is the responsibility of the applications that use the cache to maintain the data in the cache. An application can emulate the functionality of read-through caching by implementing the cache-aside strategy. This strategy effectively loads data into the cache on demand if it's not already available in the cache.

**Example Implementation in Azure**

This pattern can be used with a variety of cache mechanisms. In Azure, the most popular cache mechanism is Azure Redis Cache. Redis Cache is a key-value store that is wildly popular for caching in many web applications.

**SCENARIO:***Your online game web application shows the featured player on the homepage. Since the homepage is accessed often, it is important to cache this value. Azure Redis Cache is used for the cache* *and Document DB is used for the data store. The featured player is typically updated using a separate application where the admins can specify a new featured player.*

1. When the web application loads for the first time it makes a request (GET) to the Redis Cache instance for the name of the featured player using the key: player:featured.

2. The cache will return a nil value indicating that there is not a corresponding value stored in the cache for this key.

3. The web application can then go to the Document DB data store and gets the name of the featured player.

4. The name of the featured player will be stored in the Redis Cache using a request (SET) and the key player:featured.

5. The web application returns the name of the featured player and displays it on the homepage.

6. Subsequent requests for the home page will use the cached version of the value as Redis Cache will successfully return a value for the name of the featured player.

7. If an administrator updates the featured player, the web application will replace the value in both Redis Cache and Document DB to maintain consistency.

### Load Balancing

Load balancing is a computing concept where the application traffic or load is distributed among various endpoints by using algorithms. By using a load balancer, multiple instances of your website can be created and they can behave in a predictable manner. This provides the flexibility to grow or shrink the number of instances in your application without changing the expected behavior.

**Load Balancing Strategy**

There are a couple of things to consider when choosing a load balancer. First, you must decide whether you wish to use a physical or a virtual load balancer. In Azure infrastructure as a service (IaaS), it is possible to use virtual load balancers, which are hosted in virtual machines, if a company requires a very specific load balancer configuration.

After you select a specific load balancer you need to select a load balancing algorithm. You can use various algorithms such as round robin or random choice. For example, round robin selects the next instance for each request based upon a predetermined order that includes all of the instances.

Other configuration options exist for load balancers such as affinity or stickiness. For example, stickiness allows you determine whether a subsequent request from the same client machine should be routed to the same service instance. This might be required in scenarios where your application servers have a concept of state.

### Redis Cache

There are two primary cache mechanisms available in Azure, Azure Cache and Redis Cache. Azure cache is deprecated and only exists to support existing cloud applications. All new applications should use the Redis Cache.

**Azure Managed Cache**

Azure Cache is a managed cache service that is based on the App Fabric platform. You can create the Cache instances by using third-party applications or Windows PowerShell cmdlets. This cache mechanism is typically seen when working with custom cloud service roles.

**Redis Cache**

Redis Cache is an open-source NoSQL storage mechanism that is implemented in the key-value pair pattern common among other NoSQL stores. Redis Cache is unique because it allows complex data structures for its keys.

     **Redis**: <http://go.microsoft.com/fwlink/?LinkID=525523>

Azure Redis Cache is a managed service based on Redis Cache that provides you secure nodes as a service. There are only two tiers for this service currently available:

• **Basic**. Single node

• **Standard**. Two nodes in the Master/Replica configuration. Replication support and Service Level Agreement (SLA) is included Azure Redis Cache provides a high degree of compatibility with existing tools and applications that already integrate with Redis Cache. You can use the Redis Cache documentation that already exists on the open source community for Azure Redis Cache.  
     
     **Reference Link**: <https://docs.microsoft.com/azure/redis-cache/>

### Database Partitioning

Most cloud applications and services store and retrieve data as part of their operations. The design of the data stores that an application uses can have a significant bearing on the performance, throughput, and scalability of a system. One technique that is commonly applied in large-scale systems is to divide the data into separate partitions.

Partitioning refers to the physical separation of data for scale. Modern data stores understand that data may be spread across many different instances as the size of the sum data for the application can be larger than any individual physical store can handle in an efficient manner.

Partitioning can improve performance, availability and scalability. Any data access operation will only occur on a smaller subset (volume) of data which in turn ensures that the operation will be more efficient when compared to a query over the entire superset of data for your application. Individual databases can hit physical limits which are also overcome when portioning data across many different individual databases. Partitioning also spreads your data across multiple nodes which can individually be replicated or scaled. When a node is unavailable or being maintained, the application can use replica nodes.

### Database Partitioning: Scenario

**Problem:** Hosting Large Volumes of Data in a Traditional Single-Instance Store

A data store hosted by a single server may be subject to the following limitations:

• Storage space. A data store for a large-scale cloud application may be expected to contain a huge volume of data that could increase significantly over time. A server typically provides only a finite amount of disk storage, but it may be possible to replace existing disks with larger ones, or add further disks to a machine as data volumes grow. However, the system will eventually reach a hard limit whereby it is not possible to easily increase the storage capacity on a given server.

• Computing resources. A cloud application may be required to support a large number of concurrent users, each of which run queries that retrieve information from the data store. A single server hosting the data store may not be able to provide the necessary computing power to support this load, resulting in extended response times for users and frequent failures as applications attempting to store and retrieve data time out. It may be possible to add memory or upgrade processors, but the system will reach a limit when it is not possible to increase the compute resources any further.

• Network bandwidth. Ultimately, the performance of a data store running on a single server is governed by the rate at which the server can receive requests and send replies. It is possible that the volume of network traffic might exceed the capacity of the network used to connect to the server, resulting in failed requests.

• Geography. It may be necessary to store data generated by specific users in the same region as those users for legal, compliance, or performance reasons, or to reduce latency of data access. If the users are dispersed across different countries or regions, it may not be possible to store the entire data for the application in a single data store.

Scaling vertically by adding more disk capacity, processing power, memory, and network connections may postpone the effects of some of these limitations, but it is likely to be only a temporary solution. A commercial cloud application capable of supporting large numbers of users and high volumes of data must be able to scale almost indefinitely, so vertical scaling is not necessarily the best solution.

**Solution: Partitioning Data Horizontally Across Many Nodes**

Divide the data store into horizontal partitions or shards. Each shard has the same schema, but holds its own distinct subset of the data. A shard is a data store in its own right (it can contain the data for many entities of different types), running on a server acting as a storage node. Sharding physically organizes the data. When an application stores and retrieves data, the sharding logic directs the application to the appropriate shard. This sharding logic may be implemented as part of the data access code in the application, or it could be implemented by the data storage system if it transparently supports sharding. Abstracting the physical location of the data in the sharding logic provides a high level of control over which shards contain which data, and enables data to migrate between shards without reworking the business logic of an application should the data in the shards need to be redistributed later (for example, if the shards become unbalanced). The tradeoff is the additional data access overhead required in determining the location of each data item as it is retrieved.

To ensure optimal performance and scalability, it is important to split the data in a way that is appropriate for the types of queries the application performs. In many cases, it is unlikely that the sharding scheme will exactly match the requirements of every query. For example, in a multi-tenant system an application may need to retrieve tenant data by using the tenant ID, but it may also need to look up this data based on some other attribute such as the tenant’s name or location. To handle these situations, implement a sharding strategy with a shard key that supports the most commonly performed queries.

### Database Partitioning: Example

Microsoft Azure supports Sharding for the Azure SQL Database either by manually creating shards or using the Elastic Scale automated functionality. This example assumes that you are using the Elastic Scale feature.

**SCENARIO:***Your music catalog application uses elastic scale to spread the large amount of data across many partitions. Each partition is an instance of Azure SQL Database. The artist's unique ID number is used as the shard key.*

1. A shard map manager is created and shards are added to the manager. ID ranges for are associated to each shard so that artists are evenly spread across shards (databases). The first three shards are configured in the following manner.

| **Shard Name** | **ID range** |
| --- | --- |
| **DataStore\_I** | [(0-1000)] |
| **DataStore\_II** | [(1001-2000)] |
| **DataStore\_III** | [(2001-3000)] |
| **DataStore\_IV** | [(3001-4000)] |

There are more shards with similar ranges. The application has approximately 40 shards to start.

2. Your application needs a list of albums for an artist. The application uses the shard map manager to get a connection string to the database associated with the shard. The application can now query that database directly for all albums with the artist ID.  
3. One of the artists in a shard is getting very popular. This artist has an ID of 1245 and is affecting the resources of other artists in the same shard. Elastic Scale uses the "Split" feature to create two new shards from the original DataStore\_II shard. The new DataStore\_II shard has a disjoined ID range of [(1001-1244),(1246-2000)]. This means that all IDs from 1001-2000 except 1245 are included in this range. The new DataStore\_XLI shard has an ID range of just [1245].

4. The application already uses the smallest database size for two of its shards, DataStore\_IV and DataStore\_III. Even with this small size, the databases are not fully utilized. Elastic Scale uses the “Merge” feature to create a new shard from these two shards. The new DataStore\_XLII shard has a continuous ID range of [(2001-4000)].

### Module 1 Review Questions

**Design patterns**  
You develop a web application that allows users to sign in and download images and videos.  
You need to ensure the application performs well during peak traffic hours.  
What design pattern should you use for the solution?

Suggested Answer ↓

There are many design patterns that you can use to more effectively plan a solution. The Valet key pattern introduces a mechanism where your application can validate the user’s request without having to manage the actual download of the media. View Microsoft’s patterns and practices resource center on GitHub for more information.

**Design patterns**  
A web application is experiencing transient errors when connecting to an external database.  
What design pattern should you use for the solution?

Suggested Answer ↓

The Retry pattern is used to implement a limited number of retries if an external service connection is the issue. View Microsoft’s [patterns and practices resource center](https://github.com/mspnp) on GitHub for more information.

**Design patterns (applications)**  
You are developing an application that connects with a web app to process information and then displays the results.  
You need to manage requests between the two applications without depending on the behavior of either application.  
What design pattern should you use for the solution?

Suggested Answer ↓

The Queue design pattern is a resiliency pattern You can use

### Azure Availability

This module identifies workloads that are ideally deployed using Infrastructure-as-a-Service services in Azure. The module focuses on the VM Scale Sets and Virtual Machine services in Azure and how to best deploy workloads to these services using best practices and features such as Availability Sets.

After completing this module, students will be able to:

• Design an availability set for one or more virtual machines.

• Describe the differences between fault and update domains.

• Author a VM Scale Set ARM template.

• Join a virtualized machine to a domain either in Azure or on a hybrid network.

**High Availability**

Azure has provided a money-backed Service Level Agreement (SLA) for a long time and with the recent product releases and enhancements now provides several different services and levels of service. This lesson will explain what is available and the best way to ensure the availability of your resources. This section will cover Availability Sets and Availability Zones. After completing this section you will be able to:

• Describe Availability features of Microsoft Azure.

• Understand difference between an Availability Set and an Availability Zone.

• Deploy resources into Availability Sets correctly.

• Decide which applications and resources should not use Availability Sets.

**Azure Availability**

Microsoft Azure provides a Service Level Agreement (SLA) that is backed by a financial service credit payment for infrastructure as a Service (IaaS) Virtual Machines. The SLA depends entirely upon the deployment of the virtual machine and what resources it uses. The aim is to prevent virtual machine reboots. The method Azure uses to ensure that the SLA can be provided is an Availability Set. An availability set ensures that all virtual machines that are added to the set are placed in such a way as to ensure that neither hardware faults or Azure fabric updates that is unplanned and planned maintenance events can bring down all of the virtual machines.

**Application Availability**

An Azure virtual machine can be impacted for one of three reasons:

• Unplanned hardware maintenance event

• An unexpected downtime

• Planned maintenance events

To reduce or remove the impact of downtime related to these events there are several steps to take, these include:

• Place virtual machines in an availability set for redundancy.

• Use managed disks for all VMs placed in an availability set.

• Use Scheduled Events to respond to events.  
• Place each tier of your application in a separate availability set.

• Use a load balancer in combination with availability sets.

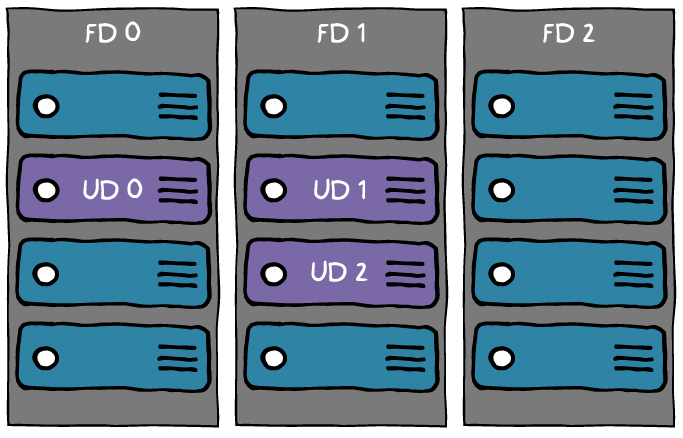
All the above steps provide additional high availability for your application and can be used in varying situations. The fundamental building block is the Availability Set.

### Availability Sets

The availability set is the recommended solution for placing multiple instance VMs, this configuration will allow for at least one VM being available at least 99.95% of the time.

https://www.skillpipe.com/api/2/content/63dcbb0c-612c-5167-8a69-1d12717b4758/4/OEBPS/Images/137130-58089.png     Best Practice: Avoid single instance VMs in an availability set. These are not subject to any SLA unless all the Operating System and Data disks are using Premium storage.

Remember Availability Sets comprise of Update Domains and Fault Domains, as shown in the image below.



**UPDATE AND FAULT DOMAINS**

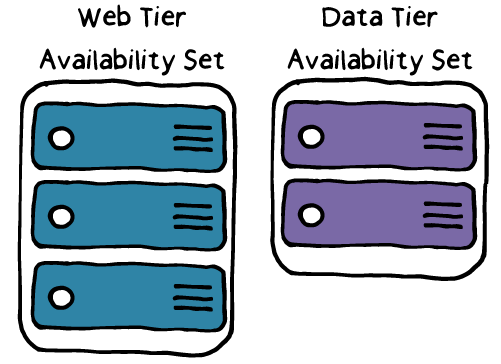
Each machine in the Availability set is placed in an Update Domain and a Fault domain.  
An Availability Set is a logical grouping service that you can create and use to ensure that the VMs you deploy within an Azure datacenter are isolated from each other. Azure places the VMs you deploy within an Availability Set across multiple physical servers, racks and network switches. In the event of an Azure hardware or software failure, only a proportion of the VMs deployed to your Availability set are impacted. The application running on the VMs will remain available. Architecturally availability sets are essential to most cloud-based solutions.

### Multiple Availability Sets

An extension of the availability set model is used logically to place individual tiers of an application into separate Availability Sets.

In this example we have a two-tier VM-based application which contains three load-balanced front-end Web servers and two back-end VMs that host a SQL Server data store. In Azure, it would be best practice to create two availability sets in preparation for the application. AVSet1 for the Web tier and AVSet2 for the data tier. Each time you create a VM for the application, you deploy it to the correct Availability set for the function it will perform.

Azure will then ensure that the VMs you create within the availability set are isolated across multiple physical hardware resources. If the physical hardware that one of your Web Server or Database Server VMs is running on has a problem, you know that the other instances of your Web Server and Database VMs remain running because they are on different hardware.

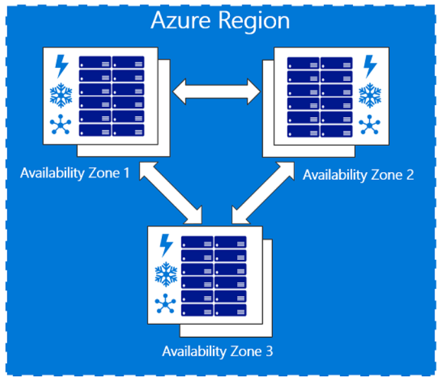


**N-TIER AVAILABILITY SETS**

### Availability Zones

A new preview service further extending highly available VMs is the Availability Zone. Whilst Availability Sets can protect from individual hardware faults and even to rack-based faults, the advent of a datacenter-wide fault would prevent the Availability set from functioning.

To extend the capability further, Microsoft has released a preview service which as an alternative to Availability Sets can provide highly available VMs across datacenter buildings or Zones. In this instance, the Availability Zone feature will allow for a complete data center failure and keep your VM based application running. The title Zone indicates a separate zone or building within a single Azure region.



There is a maximum of three Availability Zones per supported Azure region. Each Zone operates on an entirely isolated power source, cooling system, and network infrastructure.The use of Availability Zones to architect your applications to use VMs replicated in Zones provides an additional level of protection. There are few scenarios where the use of either Availability Zones or Availability Sets would not be used in a production IaaS based infrastructure or application. The use of single VMs or VM’s without availability sets whilst not attracting an SLA would still be suitable for dev and test scenarios.

### Templated Infrastructure

Azure provides a compute resource capable of true Autoscale without pre-provisioning Virtual machines. Azure VM Scale Sets allow for templated deployment of marketplace images and custom images in highly scalable and highly available infrastructure to provide a platform for big compute and big data IaaS based applications. This section provides an overview of Scale Sets and discusses the differences between deployment of multiple VMs and scale sets in your application.

After completing this section you will be able to:

• Describe Azure VM Scale Set features.

• Decide how to deploy VM Scale sets.

• Decide whether VM Scale Sets or standalone VMs provide the features you require.

• Deploy an application to a VM Scale Set using VSTS.

Scalable architecture in cloud platforms is nothing new, but always comes with a trade-off between complexity and control. Complexity is defined here as the difficulty in how to define and deploy the resources you need to control the individual resources once deployed. The tradeoff leads typically to a choice between a PaaS solution and an IaaS solution, neither really providing everything that is required.

Azure VM Scale Sets are an Azure compute resource that provides both a high degree of infrastructure resource control without the need to invest time and energy in managing the attendant networking, storage and compute. In addition, the built-in load balancing allows it to be controlled like IaaS but Scaled like PaaS.

It is usual when building cloud infrastructure to create storage, compute and network resources and create the dependencies between them. Azure VM Scale sets handle this resource creation for you and go one step further by managing the necessary configurations and optimizations when the scale set scales up or down, further reducing the workload required.

### Virtual Machine Scale Sets

An Azure VM Scale Set has several features that make it attractive to the Azure architect. The ability to define a VM Scale Set by JSON template and deploy it using any of the standard deployment methods enables their use in many automated solutions. This extends into continuous deployment scenarios with Visual Studio Team Services.

An Azure VM Scale Set allows a Virtual machine to deploy up to 1000 times in the same subnet in a controlled and automated manner with accurate auto-scaling.

An Azure VM Scale Set also requires no pre-provisioning of the Virtual Machine before adding to the scale set. The network and load balancer are created, configured and managed automatically, including the Network Address Translation (NAT) for access to and from the VM Instances.

These features added to the ease of deployment through the portal, Azure PowerShell or Azure CLI make the Azure VM Scale Set a powerful tool for the Azure cloud architect.



### Virtual Machines vs. Virtual Machine Scale Sets

Azure Virtual Machines and Azure VM Scale Sets have several unique features that allow the architect to choose between their suitability for an application or deployment.  
**Azure VM Scale Sets**

The Scale Set capacity property allows you to deploy more VMs in parallel. This is easier than writing a scripting the orchestration required to deploy individual VMs in parallel. With Azure VM Scale Sets:

• You can use Azure Autoscale to automatically scale a scale set but not individual VMs.

• You can reimage scale set VMs but not individual VMs.

• You can overprovision scale set VMs for increased reliability if a faster deployment time is required. To do this with individual VMs custom code must be written.

• Can take advantage of an upgrade policy. This makes it easy to upgrade all the VMs in your Scale. With individual VMs, this must be orchestrated.

**Azure Virtual Machines**

In contrast, with Azure Virtual Machines:

• You can attach data disks to individual VMs, but attached data disks in VM Scale Sets apply to all instances in the set. You can also attach non-empty data disks to individual VMs but not to VMs in a scale set.

• You can snapshot an individual VM but not a VM in a scale set. You can also capture an image from an individual VM but not from a VM in a scale set.

• You can migrate an individual VM to use managed disks from native disks, this cannot be done in a VM Scale Set.

• You can assign IPv6 public IP addresses to individual network Interface Cards in a VM but cannot do so for VMs in a scale set.

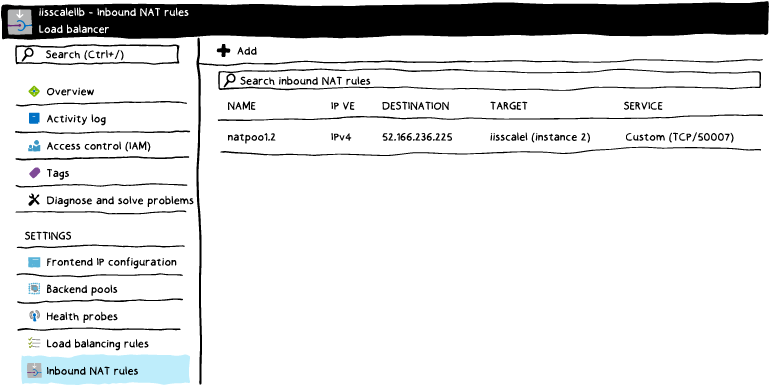
     **Note**: You can, however, assign an IPv6 public IP address to load balancers. It does not matter if the load balancer is in front of a VM or a VM Scale Set.

### Virtual Machine Scale Set Considerations

When working with VM Scale Sets there are design considerations which will affect the  
ease of use and performance of the resultant Scale Set.

**Connecting to a VM Scale Set Instance VM**

If using a Windows VM in the Scale Set, it is possible to connect to a specific VM instance by accessing the Load balancer inbound NAT rules and using the correct IP address and custom port. In the instance below the RDP client would be pointed at 52.166.236.225:50007.  
Once you enter the correct admin user credentials access will be granted to the Virtual Machine instance.



**INBOUND NAT RULES**

     Note: For a Linux VM Scale Set, there is a choice between SSH key or username and password.

### Continuous Delivery in VMSS

Use Continuous delivery to maintain an application in a VMSS with Visual Studio Team Services. Continuous delivery in Visual Studio Team Services simplifies setting up a robust deployment pipeline for your application. By default, the pipeline builds code, and updates VM scale set with the latest version of your application. The update to VM Scale set can be done either by creating an image and using that to create/update VM scale set or by using customs script VM extension to install/update your application on VM scale set. You can quickly add another VM Scale Set to the pipeline to validate your changes before they ever get to production. Need to provision additional Azure resources, run scripts, upgrade your database or run additional validation tests? You can easily extend this deployment automation to handle any other operations your application needs to do during deployment.

Visual Studio Team Services can be used to automate the deployment of code to a VMSS Instance.



When deploying an application to a VM Scale Set, there are usually two ways to achieve the goal:

1. Use of VM extensions to install software to each instance at deployment time.

2. Create a custom image that already contains the OS and application in a single VHD.

Option 2 is also known as an immutable deployment. This method has many advantages:

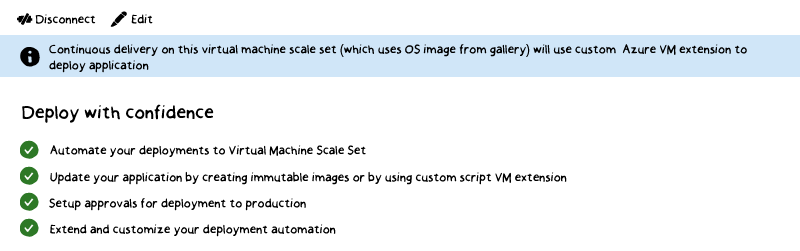
• Predictability

• Easy to Scale

• Easy to roll-back

• Faster to scale (no code to install on each VM as it is deployed)

Having chosen option 2, the benefits can be further enhanced by taking advantage of the Visual Studio Team Services continuous integration toolset and the Continuous Delivery preview service in the VM Scale Set blade.



By enabling and configuring VSTS continuous delivery, the application code and deployment scripts can be deployed from either GitHub or the Visual Studio Team Service repository.

The added benefit is that any new versions of the application can be tested on a similar VM Scale set and then deployed directly into the production instances without any downtime.

### Large VM Scale Sets

Azure VM Scale Sets can handle up to 1000 VMs in each Scale Set. Azure classifies a Scale Set that can scale beyond 100 VMs as a large VM Scale Set. The large VM Scale Set capability is marked by a **singlePlacementGroup= false** property setting. The large VM Scale set has unique requirements as well as changing the way specific aspects of a Virtual Machine deployment behave, such as load balancing and fault domains.

To decide whether your application can make efficient use of large scale sets, consider the following requirements:

• Large scale sets require Azure Managed Disks.

• Scale sets created from Azure Marketplace images can scale up to 1,000 VMs.

• Scale sets created from custom images can scale up to 300 VMs.

• Layer-4 load balancing with scale sets composed of multiple placement groups requires Azure Load Balancer Standard SKU.

• Layer-7 load balancing with the Azure Application Gateway is supported for all scale sets.

• Scale sets are defined with a single subnet – ensure subnet is large enough to handle all potential VM instances.

• Ensure your compute limits are high enough, the requirement for compute cores will prevent a successful deployment if not.

• Fault Domains and Update Domains relate to a single placement group, to maintain high availability ensure there are at least two VM instances in each Fault Domain and Update Domain.

### Domain and IaaS Applications

The use of Active Directory is widespread throughout the on-premises and cloud-based Windows infrastructure world. The advent of Azure AD brings many options for the Azure Architect to choose between. This lesson will examine the benefits of and differences between cloud only and hybrid solutions comprised of on-premises Active Directory Domain Services, Azure AD, and Azure AD Domain Services.

After completing this section you will be able to:

• Describe Azure AD Domain Services and Hybrid AD options.

• Decide when to use Azure AD Domain Services, AD DS in an Azure VM or Hybrid on premises.

• Create an Azure AD Domain Services Managed domain.

**Domain and IaaS Applications**

Authentication and Authorization of users in a cloud or hybrid infrastructure require careful planning and consideration. There are several options when using Azure AD. Azure AD is a multi-tier Identity as a Service offering that provides a complete Identity and Access Management solution for your cloud or hybrid environment. In its basic form, Azure AD is a free service that provides the ability for Single Sign-On into cloud applications. The various tiers basic, premium 1 and Premium 2 each provide additional levels of service such as Multi-Factor Authentication and additional reporting. This lesson covers the various options available to Azure AD administrators to provide domain services to their hybrid or cloud-based networks:

• Azure AD Connect

• Azure AD Domain Services

• Azure AD pass-through Authentication

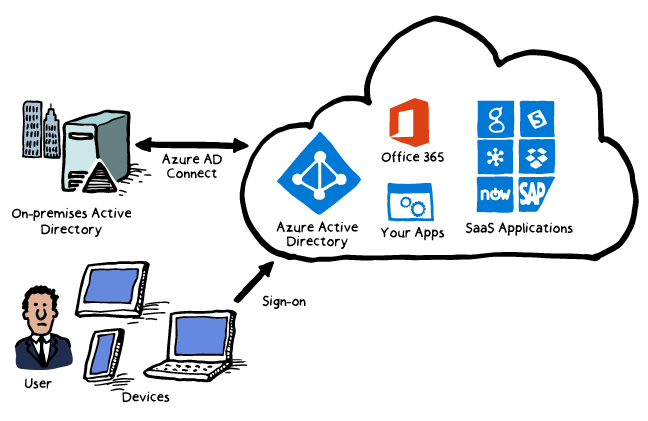
Azure AD also provides services to allow connection with consumers (Azure AD B2C) and business partners (Azure AD B2B) without deploying complex infrastructure or federation.

### Hybrid Connectivity

**Azure AD Connect**

Azure AD Connect provides the ability to integrate your on-premises directories with  
Azure AD. Having deployed AD Connect, you can provide single sign-on to cloud applications such as Office 365, Azure and other SaaS applications.

Why use AD Connect? Having a common identity to access both cloud and on-premises applications and resources enables users to take advantage of a Single identity, one set of credentials to remember and a Single tool, single sign in, easy for administrators to deploy.



**AZURE AD HYBRID**

Azure AD Connect provides the choice of:

• Password Synchronization only, the ability to synchronize users and groups.

• ADFS, to allow on-premises authentication, 3rd party MFA, etc.

• Pass-through authentication, provides on-premises authentication without deploying ADFS.

Whichever architecture you decide to use, if on-premises directories are included, you will need Azure AD Connect to provide the synchronization engine. The Microsoft Identity Manager (MIM) client is installed on-premises and used to configure the users, groups and attributes to be synchronized.

### Azure AD Domain Services

Azure provides the ability to deploy Infrastructure solutions in many ways. To support this Azure AD Domain Services provides managed domain services such as domain join, group policy, LDAP and Kerberos authentication, all of which are entirely Windows Server Active Directory compatible.

Azure AD Domain Services allows you to take advantage of these services without having to deploy a Domain Controller to an IaaS VM in the cloud. Azure AD Domain Services is compatible with both Cloud only tenants and hybrid tenants using Azure AD Connect.

**Cloud Only Tenant**

By enabling Azure AD Domain Services on an Azure AD tenant, Azure creates a highly-available domain service which is connected to the virtual network, and all the Azure AD objects are available within this domain, but all user identities, credentials and groups, including group memberships, are created and managed in Azure AD.

The advantages of this solution are:

• The domain administrator does not need to manage this domain or any domain controllers.

• AD replication for this domain does not require management. All objects are automatically available.

• Azure manages this Domain, so the Azure AD tenant administrator has no domain or enterprise admin privileges.

### Hybrid Cloud Tenant

By enabling Azure AD Domain Services on an Azure AD tenant that is synchronized to an on-premises directory, an additional stand-alone Domain is created by the managed service. All objects from the on-premises domain and the Azure AD tenant are available to the managed service domain. Tenant identities are still created and managed within Azure AD, and on-premises identities are still created and managed on-premises. This solution allows users to sign in to cloud services with their on-premises identities. For this to work with the Azure AD Domain Services, Azure AD Connect must be configured to allow password synchronization; this is required so resources in the cloud connected to the managed domain can use Kerberos to authenticate. The managed domain is a standalone domain and not an extension to the on-premises directory.

The advantages of this solution are:

• The domain administrator does not need to manage this domain or any domain controllers for the managed domain.

• AD replication for this domain does not require management. All objects are automatically available.  
• Azure manages this Domain, so the Azure AD tenant administrator has no domain or enterprise admin privileges. The benefits of the Azure AD Domain Services managed domain offering are:

• **Simple to deploy in a few clicks**—No IaaS infrastructure required to provide authentication to Virtual Machines.

• **Deep integration within your Azure AD tenant.**

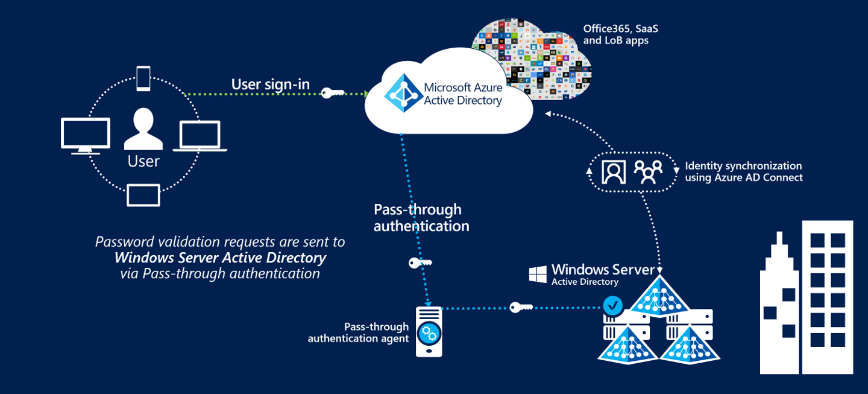
•**Compatible with Windows Server Active Directory**—Not all features available in Windows Server AD are available in Azure AD Domain Services. The following are compatible, LDAP, Kerberos, NTLM, Group Policy, and domain join capabilities.

**• Cost-effective**—No need to pay for Azure IaaS Virtual Machines.

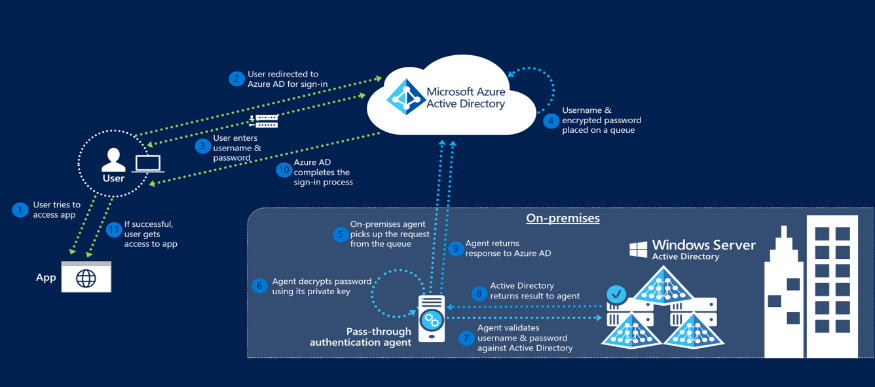
### Azure AD Pass-Through Authentication

Azure AD Connect provides a new service that permits on-premises authentication without the need to deploy ADFS infrastructure. This is a considerable cost and time saver. With Azure AD Connect, you have no need for complicated certificates or trusts.

Azure AD Pass-through Authentication enables users to sign in to both on-premises and cloud applications using the same credentials. When users sign in using Azure AD, this feature validates users' passwords against your on-premises Active Directory, the same was an ADFS based solution would do.



Pass-through authentication:



The highlighted benefits of this solution are:

• A great user experience, the user uses the same passwords to sign into both on-premises and cloud-based applications.

• Users spend less time resolving password-related issues.

• Users can be enabled to use self-service password management from the Azure AD directly.

• Easy to deploy, there is no requirement for on-premises deployments or network configuration. Only a small agent needed on-premises.  
• Secure storage of passwords since on-premises passwords are never stored in the cloud.

• No additional ports or network configuration is required since the agent communicates outbound, so no perimeter network is required.  
• Takes advantage of Azure AD Conditional Access policies, including Multi-Factor Authentication (MFA).

• By installing additional agents, the service can become highly available.

This is a free feature available to all tiers of Azure AD and supports all web-based apps and those supporting modern authentication. Multi-forest environments are supported, although some routing changes may be required.

### Lab Steps

#### Online Lab - Building Azure IaaS-Based Server Applications by using Azure ARM Templates and Azure Building Blocks

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy an Azure VM by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal.

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Create an Azure VM running Windows Server 2016 Datacenter.

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Windows Server 2016** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Windows Server 2016 Datacenter**.
4. On the **Windows Server 2016 Datacenter** blade, click the **Create** button.
5. On the **Basics** tab, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, click **Create new**, in the text box, type **AADesignLab0301-RG**, and click **OK**.
   * In the **Name** text box, enter the value **lab03vm0**.
   * In the **Region** drop-down list, select an Azure region to which you want to deploy resources in this lab.
   * In the **Availability options** drop-down list, select **Availability set**.
   * In the **Availability set** section, click **Create new**, box, enter the value **lab03avset0**, set **Fault domains** to the maximum value, leave **Update domains** with its default value, and click **OK**.
   * Leave the entry in the **Image** drop-down list set to its default value.
   * Ensure that the size is set to **Standard DS1 v2**
   * In the **Username** text box, enter the value **Student**.
   * In the **Password** and **Confirm password** text boxes, enter the value **Pa55w.rd1234**.
   * In the **Public inbound ports** section, select the **Allow selected port** option and, in the **Select inbound ports** drop-down list, select **HTTP**.
   * Leave the **Already have a Windows license?** option set to **No**.
   * Click **Next: Disks >**
6. On the **Disks** tab, perform the following tasks:
   * Ensure that the **OS disk type** dropdown list entry is set to **Premium SSD**
   * Click **Next: Networking >**
7. On the **Networking** tab, perform the following tasks:
   * In the **Virtual network** section, click **Create new**.
   * On the **Create virtual network** blade, specify the following settings and click **OK**:
     + In the **Name** text box, enter the value **lab03vnet0**.
     + In the **Address range** text box, enter the value **10.3.0.0/16**.
     + In the **Subnet name** text box, enter the value **subnet-0**.
     + In the **Subnet address range** text box, enter the value **10.3.0.0/24**, and click **OK**.
   * Leave the **Public IP** entry set to its default value.
   * Leave the **NIC network security group** option set to **Basic**.
   * Leave the **Public inbound ports** option set to **Allow selected ports**
   * Leave the **Select inbound ports** entry set to **HTTP**
   * Leave the **Accelerated networking** entry set to its default value.
   * Click **Next: Management >**
8. On the **Management** tab, perform the following tasks:
   * Leave the **Boot diagnostics** option set to its default value.
   * Leave the **OS guest diagnostics** option set to its default value.
   * Leave the **Diagnostics storage account** entry set to its default value.
   * Leave the **System assigned managed identity** option set to its default value.
   * Leave the **Enable auto-shutdown** option set to its default value.
   * Leave the **Enable backup** option set to its default value.
   * Click the **Review + create** button.
9. On the **Create a virtual machine** blade, review the settings of your new virtual machine and click the **Create** button.
10. Do not wait for the deployment to complete and proceed to the next task.

##### Task 3: View DSC configuration

1. On the Taskbar, click the **File Explorer** icon.
2. In the **File Explorer** window that appears, navigate to the **\allfiles\AZ-301T04\Module\_02\LabFiles\Starter\** folder.
3. Right-click the **IISWebServer.zip** file and select the **Extract All...** option.
4. In the **Extract Compressed (Zipped) Folders** dialog, perform the following tasks:
   * In the **Files will be extracted to this folder:** field, enter the name of the folder into which you want to extract the files.
   * Ensure that the **Show extracted files when complete** checkbox is selected.
   * Click the **Extract** button.
5. In the new **File Explorer** window that appears, right-click the **IISWebServer.ps1** file and select the **Open with Code** option to start the **Visual Studio Code** application.
6. In the **Visual Studio Code** window that appears, review the content of the PowerShell script.
7. At the top of the **Visual Studio Code** window, click the **File** menu and select the **Close Window** option.
8. Close both **File Explorer** windows.
9. Return to the **Microsoft Edge** window with the **Azure Portal** open.

##### Task 4: Create an Azure Storage account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Storage account** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Storage account - blob, file, table, queue**.
4. On the **Storage account - blob, file, table, queue** blade, click the **Create** button.
5. On the **Create storage account** blade, perform the following tasks:
   * In the **Name** text box, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **Deployment model** section, ensure that the **Resource manager** option is selected.
   * In the **Account kind** drop-down list, ensure that the **Storage (general purpose v1)** option is selected.
   * Leave the **Location** entry set to the same Azure region you selected earlier in this exercise.
   * In the **Replication** drop-down list, select the **Locally-redundant storage (LRS)** entry.
   * In the **Performance** section, ensure that the **Standard** option is selected.
   * In the **Secure transfer required** section, ensure that the **Disabled** option is selected.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, ensure that the **Use existing** option is selected and, in the drop-down list below, select the resource group you created earlier in this exercise.
   * Leave the **Configure virtual networks** option set to its default value.
   * Leave the **Hierarchical namespaces** option set to its default value.
   * Click the **Create** button.
6. Wait for the deployment to complete before you proceed to the next task.**Note**: This operation can take about 2 minutes.

##### Task 5: Upload DSC configuration to Azure Storage

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the storage account.
3. On the resource group blade, click the entry representing the newly created storage account.
4. With the **Overview** selection active, on the storage account blade, click **Blobs**.
5. Click the **Container** button at the top of the blade.
6. In the **New container** pane that appears, specify the following settings and click **OK**:
   * In the **Name** text box, enter the value **config**.
   * In the **Public access level** list, select the **Blob (anonymous read access for blobs only)** option.
7. Back on the **Blob service** blade, click the entry representing the new **config** container.
8. On the **config** blade, click the **Upload** button at the top of the blade.
9. In the **Upload blob** pane, perform the following tasks:
   * In the **Files** field, click the blue folder button to the right of the field.
   * In the **Open file** dialog that appears, navigate to the **\allfiles\AZ-301T04\Module\_02\LabFiles\Starter\** folder.
   * Select the **IISWebServer.zip** file.
   * Click the **Open** button to close the dialog box and return to the **Upload blob** popup.
   * Click the **Upload** button.
10. Navigate to the **config** blade and click the entry representing the **IISWebServer.zip** blob.
11. In the **Blob properties** popup that appears, locate and record the value of the **URL** property. This URL will be used later in this lab.

##### Task 6: Deploy an Azure VM by using an Azure Resource Manager template with PowerShell DSC extension from the Aure portal.

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
6. On the **Edit template** blade, click **Load file**.
7. In the **Choose File to Upload** dialog box, navigate to the **\allfiles\AZ-301T04\Module\_02\LabFiles\Starter\** folder, select the **dsc-extension-template.json** file, and click **Open**. This will load the following content into the template editor pane:

{ "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#", "contentVersion": "1.0.0.0", "parameters": { "virtualMachineName": { "type": "string", "defaultValue": "lab03vm0" }, "configurationModuleUrl": { "type": "string" }, "extensionFunction": { "type": "string", "defaultValue": "IISWebServer.ps1\\IISWebServer" } }, "resources": [ { "apiVersion": "2018-06-01", "type": "Microsoft.Compute/virtualMachines/extensions", "name": "[concat(parameters('virtualMachineName'), '/dscExtension')]", "location": "[resourceGroup().location]", "properties": { "publisher": "Microsoft.Powershell", "type": "DSC", "typeHandlerVersion": "2.75", "autoUpgradeMinorVersion": true, "settings": { "ModulesUrl": "[parameters('configurationModuleUrl')]", "ConfigurationFunction": "[parameters('extensionFunction')]", "Properties": { "MachineName": "[parameters('virtualMachineName')]" } }, "protectedSettings": null } } ] }

1. Click the **Save** button to persist the template.
2. Back on the **Custom deployment** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and, in the drop-down list, select the resource group you created earlier in this exercise.
   * Leave the **Location** drop-down list set to its default value.
   * Leave the **Virtual Machine Name** field set to its default value: **lab03vm0**.
   * In the **Configuration Module Url** field, enter the URL value that you recorded in the previous task.
   * Leave the **Extension Function** field set to its default value: **IISWebServer.ps1\IISWebServer**.
   * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
   * Click the **Purchase** button.
3. Wait for the deployment of the DSC configuration to complete before you proceed to the next task.**Note**: DSC configuration deployment can take up to ten minutes.

##### Task 7: Validate that the Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine.
3. On the resource group blade, click the entry representing the **Virtual Machine** you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab.

**Review**: In this exercise, you deployed an **Virtual Machine** from the Azure portal and then used the **PowerShell DSC** extension to apply changes to the virtual machine in an unattended manner.

#### Exercise 2: Deploy an Azure Virtual Machine Scale Set (VMSS) by using Azure Resource Manager templates with PowerShell Desired State Configuration (DSC) extension from the Azure portal.

##### Task 1: View an Azure Resource Manager template.

1. On the Taskbar, click the **File Explorer** icon.
2. In the **File Explorer** window that appears, navigate to the **\allfiles\AZ-301T04\Module\_02\LabFiles\Starter\** folder.
3. Right-click the **vmss-template.json** file and select the **Open with Code** option to start the **Visual Studio Code** application.
4. In the **Visual Studio Code** window that appears, review the content of the JSON file.
5. At the top of the **Visual Studio Code** window, click the **File** menu and select the **Close Window** option.
6. Close the **File Explorer** window.
7. Return to the **Microsoft Edge** window with the **Azure Portal** open.

##### Task 2: Deploy a VMSS using ARM

1. In the hub menu of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Template deployment**.
4. On the **Template deployment** blade, click the **Create** button.
5. On the **Custom deployment** blade, click **Build your own template in the editor**.
6. On the **Edit template** blade, click **Load file**.
7. In the **Open** file dialog that appears, navigate to the **F:\Labfiles\Mod03\Starter** folder.
8. Select the **vmss-template.json** file.
9. Click the **Open** button.
10. Back on the **Edit template** blade, click the **Save** button to persist the template.
11. Back on the **Custom deployment** blade, perform the following tasks:
    * Leave the **Subscription** drop-down list entry set to its default value.
    * In the **Resource group** section, select the **Create new** option and, in the text box, type **AADesignLab0302-RG**.
    * Leave the **Location** entry set to its default value.
    * In the **Admin User Name** text box, enter the value **Student**.
    * In the **Admin Password** text box, enter the value **Pa55w.rd1234**.
    * In the **Instance Count** text box, enter the value **2**.
    * Leave the **Overprovision** text box set to its default value: **true**.
    * In the **Configuration Module Url** text box, enter the URL that you recorded for the uploaded blob in the previous exercise of this lab.
    * In the **Terms and Conditions** section, select the **I agree to the terms and conditions stated above** checkbox.
    * Click the **Purchase** button.
12. Wait for the deployment to complete before you proceed to the next task.

##### Task 3: Validate that VMSS instances are serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine scale set.
3. On the resource group blade, click the resource of the **Public IP address** type.
4. On the Public IP address resource blade, in the **Essentials** section, identify the value of **IP address** entry.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab and return to the browser tab with the **Azure Portal** currently active.

**Review**: In this exercise, you created a Virtual Machine scale set and configured the individual instances using PowerShell DSC.

#### Exercise 3: Deploy Azure VMs running Windows Server 2016 and Linux by using Azure Building Blocks with PowerShell Desired State Configuration (DSC) extension from the Azure Cloud Shell.

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this exercise.
   * In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab0303-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 2: Install the Azure Building Blocks npm package in Azure Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a local directory to install the Azure Building Blocks npm package:

mkdir ~/.npm-global

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to update the npm configuration to include the new local directory:

npm config set prefix '~/.npm-global'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the ~./bashrc configuration file for editing:

vi ~/.bashrc

1. At the **Cloud Shell** command prompt, in the vi editor interface, scroll down to the bottom of the file (or type **G**), scroll to the right to the right-most character on the last line (or type **$**), type **a** to enter the **INSERT** mode, press **Enter** to start a new line, and then type the following to add the newly created directory to the system path:

export PATH="$HOME/.npm-global/bin:$PATH"

1. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install the Azure Building Blocks npm package:

npm install -g @mspnp/azure-building-blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to exit the shell:

exit

1. In the **Cloud Shell timed out** pane, click **Reconnect**.**Note**: You need to restart Cloud Shell for the installation of the Buliding Blocks npm package to take effect.

##### Task 3: Deploy a Windows Server 2016 Azure VM from Cloud Shell by using Azure Building Blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to download the GitHub repository containing the Azure Building Blocks reference architecture files:

git clone https://github.com/mspnp/reference-architectures.git

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the content of the Azure Building Block parameter file you will use for this deployment:

cat ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you created earlier in this exercise:

RESOURCE\_GROUP='AADesignLab0303-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the Building Blocks parameter file:

sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the Building Blocks parameter file:

sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the Building Blocks parameter file:

cat ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy a Windows Server 2016 Azure VM by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP -s $SUBSCRIPTION\_ID -l $LOCATION -p ./reference-architectures/virtual-machines/single-vm/parameters/windows/single-vm.json --deploy

1. Wait for the deployment to complete before you proceed to the next task.

##### Task 4: Validate that the Windows Server 2016 Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the Windows Server 2016 Datacenter virtual machine earlier in this exercise.
3. On the resource group blade, click the entry representing the virtual machine you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Internet Information Services webpage.
7. Close the new browser tab.

##### Task 5: Deploy a Linux Azure VM from Cloud Shell by using Azure Building Blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to view the content of the Azure Building Block parameter file you will use for this deployment:

cat ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to generate the SSH key pair that you will use to authenticate when accessing the Linux VM:

ssh-keygen -t rsa -b 2048

* + When prompted to enter the file in which to save the key, press **Enter** to accept the default value **(~/.ssh/id\_rsa)**.
  + When prompted to enter passphrase, press **Enter** twice.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair:

PUBLIC\_KEY=$(cat ~/.ssh/id\_rsa.pub)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the public key of the newly generated key pair and which takes into account any special character the public key might include:

PUBLIC\_KEY\_REGEX="$(echo $PUBLIC\_KEY | sed -e 's/\\/\\\\/g; s/\//\\\//g; s/&/\\\&/g')"

**Note**: This is necessary because you will use the **sed** utility to insert this string into the Azure Building Blocks parameter file. Alternatively, you could simply open the file and enter the public key string directly into the file.

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group you will use for the deployment:

RESOURCE\_GROUP='AADesignLab0304-RG'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab0301-RG'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the Building Blocks parameter file:

sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **sshPublicKey** parameter with the value of the **$PUBLIC\_KEY\_REGEX** variable in the Building Blocks parameter file:

sed -i.bak2 's/"sshPublicKey": ""/"sshPublicKey": "'"$PUBLIC\_KEY\_REGEX"'"/' ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the Building Blocks parameter file:

cat ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new resource group:

az group create --name $RESOURCE\_GROUP --location $LOCATION

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy a Linux Azure VM by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP -s $SUBSCRIPTION\_ID -l $LOCATION -p ./reference-architectures/virtual-machines/single-vm/parameters/linux/single-vm.json --deploy

1. Wait for the deployment to complete before you proceed to the next task.

##### Task 7: Validate that the Linux Azure VM is serving web content

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click the entry representing the resource group into which you deployed the virtual machine.
3. On the resource group blade, click the entry representing the virtual machine you deployed.
4. On the **Virtual machine** blade, locate the **Public IP address** entry, and identify its value.
5. Open a new Microsoft Edge tab and navigate to the IP address you identified in the previous step.
6. Verify that you are able to access the default Apache2 Ubuntu webpage.
7. Close the new browser tab.
8. Close the **Cloud Shell** pane.

**Review**: In this exercise, you deployed Azure VMs running Windows Server 2016 Datacenter and Linux from Cloud Shell by using Azure Building Blocks.

#### Exercise 4: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab03')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab03')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 2 Review Questions

**Hardware faults**  
You design a cloud solution that uses multiple Azure virtual machines (VMs).  
What can you do to protect from hardware or rack-based faults?  What can you do to protect from datacenter faults?

Suggested Answer ↓

You can create an Availability Set to ensure that all virtual machines which are added to the set are safe from hardware faults or rack-based faults. Availability Zones keep your VMs running during a complete datacenter failure.

**VM Scale sets**  
You design a cloud solution that uses multiple Azure virtual machines (VMs).  
You need to ensure the VMs can auto-scale in parallel during peak workload periods.  
What can you use to auto-scale the VMs? What can you use to maintain the deployment of applications on the VMs?

Suggested Answer ↓

An Azure VM scale set is used to horizontally auto-scale Azure VMs. You can use continuous delivery processes to maintain an application in a scale set by using Visual Studio Team Services.

**Authentication**  
You design a cloud solution that uses on-premises and Azure-based resources.  
You need to design a hybrid solution for authenticating resources.

What Azure service should you use?

Suggested Answer ↓

You can use Azure AD Connect and Active Directory Federation Services (AD FS) to connect Azure and on-premises resources to Azure AD.

### Azure Virtual Network (VNET) Architecture

This module describes the various networking and connectivity options available for solutions deployed on Azure. The module explores connectivity options ranging from ad-hoc connections to long-term hybrid connectivity scenarios. The module also discusses some of the performance and security concerns related to balancing workloads across multiple compute instances, connecting on-premise infrastructure to the cloud and creating gateways for on-premise data.

After completing this module, students will be able to:

• Describe DNS and IP strategies for VNETs in Azure.

• Compare connectivity options for ad-hoc and hybrid connectivity.

• Distribute network traffic across multiple loads using load balancers.

• Design a hybrid connectivity scenario between cloud and on-premise.

**Virtual Networks**

Azure Virtual Networking is the baseline for all Azure networking resources and features. Starting from a high-level topology view and detailed walkthrough of what an Azure VNET is, we drill down into all aspects of Subnet designing, how to integrate networks across Azure regions, as well as between on-premises corporate networks and Azure regions. Next, we describe the different configuration settings within a VNET, like what DNS options are available and how IP-addressing is working for both Public and Private IP’s.

After this section you will be able to:

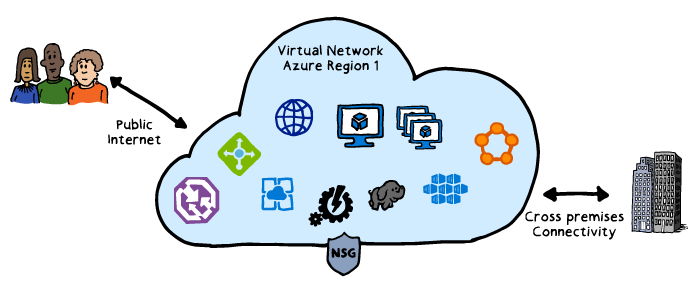
• Understand Azure Virtual Networks.

• Architect multi-region networking across Azure regions as well as between Azure and on-premises networks.

• Understand several Azure VNET configuration options like DNS, IP addressing and alike.

### Multi-Region Virtual Network Architecture

An Azure Virtual Network (or VNET), is the logical unit of multiple or all network resources in an Azure region. On the highest level of the network topology in an Azure Region, you define a Virtual Network. An Azure Region can have one or multiple Virtual Networks defined. Within a Virtual Network, you create one or more subnets. Like in a typical on-premises network, all traffic within a subnet is allowed, but communication across different subnets is blocked by default. Separating your workloads in multiple subnets within a VNET is a best practice and highly recommended.



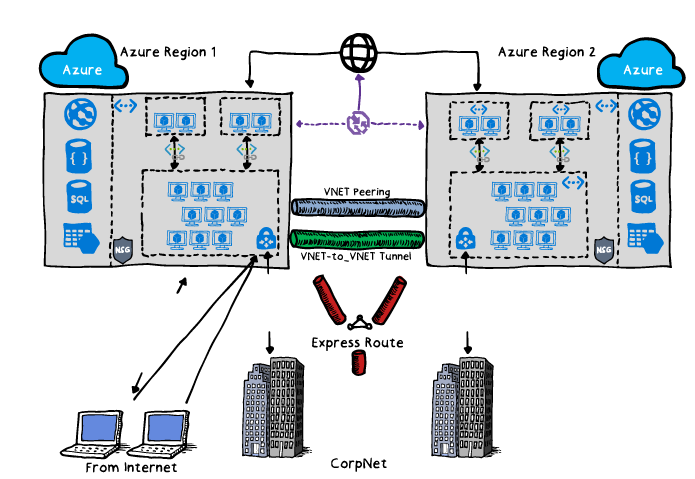
From a high-level perspective, Azure Virtual Networks allow for the following communication flows:

• From the Public Internet, incoming traffic is routed to Public IP addresses of Azure Resources, secured by Network Security Groups. (think of these as software-defined firewalling for now).

• Several Azure Resources are making use of VNETs and subnets for IP addressing. These Azure Resources can be IAAS related (Virtual Machines, VM Scale Sets, Load Balancers, Azure Traffic Manager,) as well as PAAS related (Service Fabric, Azure Container Services, Hadoop, Azure Application Services,).

• From a back-end perspective, Azure allows for a hybrid network integration between multiple Azure Regions, or with on-premises datacenters. In a next module, we will detail the different capabilities and configuration options. We can assume an organization wants to leverage the capabilities of the Azure Public Cloud, by deploying workloads across multiple Azure Regions, or across Azure Regions and on-premises data centers. This scenario is entirely achievable in Azure:

• From the Public Internet, Azure allows for load balancing across multiple Azure Regions by deploying Azure Traffic Manager.



Interconnecting Azure Regions with each other is possible in three different ways:

* Configuring Azure Site-to-Site VPN between both regions
* Configuring Azure ExpressRoute communication tunnels
* A newer capability that allows for interconnecting multiple Azure regions is called Azure VNET Peering

    **Note**: VNET Peering provides typical network communication, where VPN provides tunnel encryption—so it is up to the business requirements to make your decision.

Interconnecting Azure Regions with on-premises datacenters is possible in 2 different ways:

* + Configuring Azure Site-to-Site VPN between Azure and on-premises
  + Configuring Azure ExpressRoute communication tunnels between Azure and on-premises

Network Security Groups are active on Azure VNET layer or on individual NIC layer; however, NSGs cannot span Azure Regions. This means that you need to define the configuration within each VNET in each Azure Region.

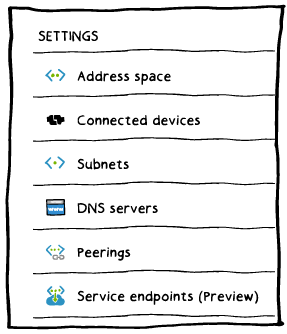
### VNETs & Subnets

**Networking Topology:**

* + Define one or more VNETs within an Azure Region, and configure an address space for each.
  + Define one or more SubNets within a VNET, and configure address space within the VNET range.
  + VNETs and SubNets are using CIDR notation (x.x.x.x/24, x.x.x.x/16,…).
  + Configure Network Security Group settings on VNET level.
  + Attach a NIC to a SubNet.

**SubNet IP Addressing:**

* + IP-address gets allocated to a NIC during provisioning of the NIC
  + First available IP-address in a SubNet range is x.x.x.4
  + Azure SubNets support dynamic (=default) and static IP addressing
  + Subnets can be configured directly in the Azure Portal.

  
SUBNET CONFIGURATION

**Public IP-addressing**

* + Used for all public internet-facing communication
  + Required parameter when creating a VM from the portal

**Private IP-addressing**

* + Used for all inter-VNET communication
  + Used for all communication between an Azure VNET and an on-premises VNET
  + Azure DNS Resolving

**DNS Server settings are configured on VNET level**

**Using Azure DNS is de default configuration setting, but this can be modified**

**Or use your custom DNS configuration:**

* + Azure DNS Appliance (from Azure MarketPlace)
  + Azure VM (e.g. Windows ADDS with DNS)
  + On-premises DNS solution (requires connectivity)

**Public DNS names (available for VMs and App Services) must be unique across Azure regions**

**An example of such Public DNS name is *<host.region.cloudapp.azure.com>***

### Load Balancing Solutions

Azure Load Balancing refers to several different Azure Resources that are available on the Azure Platform, offering application workload load balancing capabilities, much similar to traditional on-premises load balancing solutions. This lesson starts with describing the different flavors, zooming in on the characteristics of each of the available flavors.

After completing this section, you will be able to:

• Understand Azure Load Balancing.

• Recognizing the use case for each of the in-Azure provided Load Balancing solutions.

• Deciding between the different Azure Load Balancing options.

**Load Balancing Solutions**

Azure provides several built-in Azure Load Balancing Solutions:

• Azure Load Balancer

• Azure Application Gateway

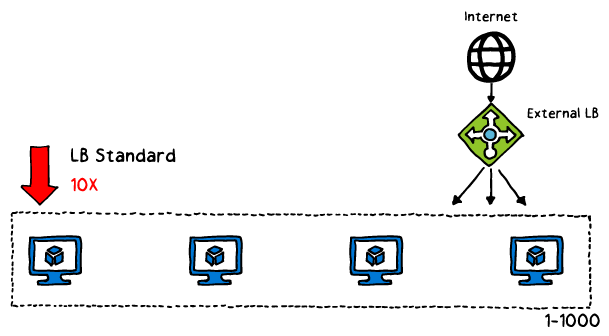
• Azure Marketplace Load Balancing Appliance

• Azure Traffic Manager

**Azure Load Balancer**

Starting with Azure Load Balancer, it is important to point out it can be configured both as an “external load balancer,” or as an “internal load balancer,” where one cannot act as both external and internal at the same time. In the scenario of an Azure external load balancer, the load balancer front-end is configured with a Public-facing IP-address, sending all traffic along to the back-end pool servers, using their internal IP-addresses.

Azure Load Balancers can handle almost any TCP or UDP traffic; use case scenarios include RDS (Remote Desktop Services Farm), Linux SSH server connectivity load balancing, or any other application-specific traffic.



In relation to Azure Virtual Machine Availability Sets, where you deploy multiple instances of the same Virtual Machine, it is thanks to Azure Load Balancer functionality; an Availability Set can grow to 10x the number of instances (100 to 1000 in a single Availability Set).

Pretty similar in functionality is an Azure Internal Load Balancer. The main difference is that this solution doesn’t have a Public-facing IP-address, and all communication is based on internal IP-addressing and IP-routing. A typical use case for this setup is like the diagram shown, where you want to load balance incoming traffic between Azure Subnets or Azure VM Availability Sets. The external Load Balancer takes care of the incoming web traffic to the Web Server AVSet, where any Web Server VM can communicate with the Database Server backend, using the internal Azure Load Balancer option.

Azure Load Balancer existed in a Basic edition, which gave you the following characteristics:

• Up to 100 backend instances

• Single Availability Set

• Basic NAT and Probe health status

• No HA Ports

• Network Security Groups (NSG) are optional

• No cost related to using this Azure Load Balancer solution

While Azure Load Balancer was and still is a viable solution, it also had some technical limitations. Some of these are resolved in the Standard SKU of the Azure Load Balancer, giving you the following characteristics:

• Up to 1000 backend instances

• Availability Sets are not required; providing support for Availability Zones

• Integrated Frontend and Backend health metrics

• Support for HA Ports

• Network Security Groups are required during configuration and deployment

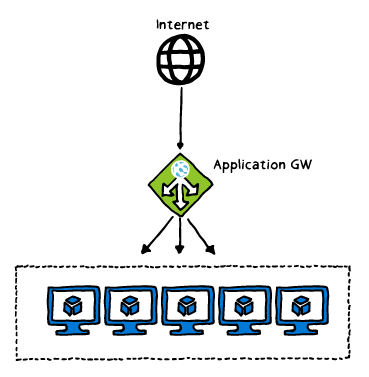
### Azure Application Gateway

Another Azure Load Balancing solution on the platform is Azure Application Gateway. While mostly similar in functionality than the Azure Load Balancer, we like to focus on its specifics:

Application Gateway - Load Balancing, active on Layer 7 of the network stack; this mainly means it is “application intelligent.”

Main features Application Gateway provides, compared to Azure Load Balancer, are:

* HTTP/HTTPS traffic only, no other ports allowed
* SSL Offloading
* Cookie Affinity
* Web Application Firewall (WAF)
* URL Based Routing



In the scenario of URL based Routing, Application Gateway recognized the incoming web request

and based on the URL information; it will redirect traffic to the correct destination. Besides main URL redirection, this can also be used on subheaders. For example, in the given scenario, web requests to the ***http://www.domain2.com/finance***web app will be redirected to WebAV1, where all requests to the ***http://www.domain2.com/sales*** web app, will be redirected to another WebAV2 availability set.

In regard to SSL Termination, Azure App Gateway provides the following capabilities:

• SSL Offloading, by importing the SSL Certificate onto the App Gateway; traffic to the backend servers don’t require HTTPS communication, also that would still be an option.

• HTTP to HTTPS redirect; this means that, whenever a user is connecting to the web app using HTTP, the request will be redirected to HTTPS, forcing SSL Tunneling for this given request. The last feature of Azure Application Gateway we want to discuss here is Web Application Firewall (WAF). Based on industry-standard rules for WAF, CRS 2.2.9 and CRS 3.0, Azure Application Gateway provides protection against several common attacks and threats on application workloads:

• SQL Injection

• Cross-site scripting

• Protocol violations

• Generic attacks

• HTTP rate limiting

• Scanner detection

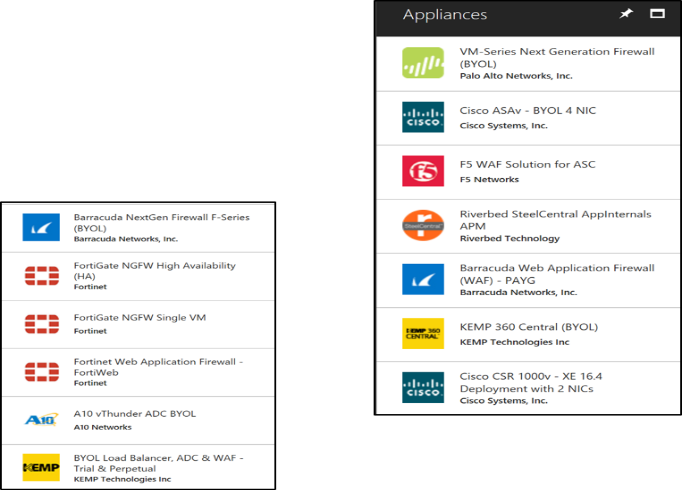
• Session fixation

• LFI/RFI

### Azure Load Balancing Marketplace Appliances

Where both Azure Load Balancer and Azure Application Gateway are an interesting option, directly built into the Azure Platform and offered “as a service,” the capabilities around management, monitoring and control might be too limited for certain organizations. In that scenario, one can deploy a third-party Azure Marketplace appliance.

The “common” vendors from the on-premises world are present in an Azure Appliance alternative. Support is initially provided by Microsoft, backed by SLA’s, and acting as a SPOC.



Most flavors support 2 different licensing models to choose from:

**• BYOL:** Bring Your Own License; this is an ideal candidate if you are removing or downsizing on your on-premises running third-party load balancer. Depending on the specific licensing terms of the vendor, one can reuse the license key on the Azure VM Appliance.

**• Pay-Per-Use:** In this model, the monthly Azure VM consumption cost is based on the VM Size allocation to the Appliance, as well as a monthly licensing fee for the third party load balancing application within the VM.

### Azure Traffic Manager

Microsoft Azure Traffic Manager allows you to control the distribution of user traffic to your specified endpoints, which can include Azure cloud services, websites, and other endpoints. Traffic Manager works by applying an intelligent policy engine to Domain Name System (DNS) queries for the domain names of your internet resources. Your Azure cloud services or websites can be running in different datacenters across the world.

Traffic Manager is very flexible because it allows you to mix various endpoints behind the same DNS name. Traffic Manager can be used in a variety of scenarios but most use cases fall in the following scenarios:

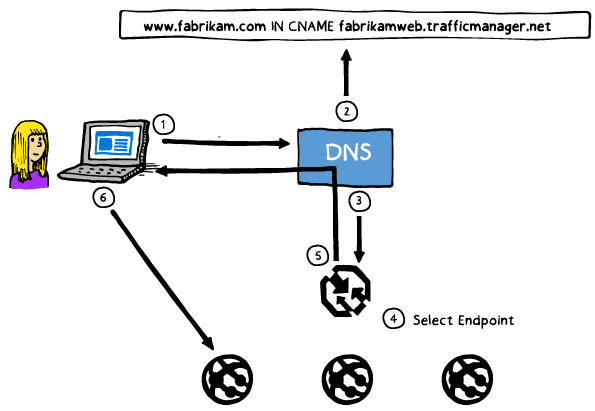
• **Failover**: Traffic Manager can poll to determine if and endpoint is online or offline. The endpoints are then ordered in a priority list. By default, traffic routes to the first endpoint. If the first endpoint is down, traffic routes to the next endpoint (2) in the list. Traffic Manager will route requests to the endpoint that is the highest in the priority list and is still online. Using this method, you can have Traffic Manager route traffic to primary or backup datacenters/services for a simple failover scenario.

• **Geography**: Traffic Manager uses an Internet Latency Table to determine the endpoint that is "closest" to the client that is making a request. Using this method, an application can be hosted in West Europe and West US. A user from Denmark can reasonably expect to be served by the endpoint residing in the West Europe datacenter and should experience lower latency and higher responsiveness.

• **Distribution**: Traffic Manager can distribute traffic in a near-random way to distribute traffic evenly across a set of endpoints. If a specific endpoint is down, the traffic is distributed evenly across the remaining endpoints. The distribution can optionally be weighted so that certain endpoints receive more requests than others. The weighted distribution is especially useful if you want to distribute a small subset of your traffic to a hot disaster recovery site that is using smaller service tiers but keep the majority of your traffic to a primary site that is using larger service tiers.

**Using Traffic Manager with Web Apps**

Traffic Manager can be integrated with Web Apps easily. When you configure a Traffic Manager profile, the settings that you specify provide Traffic Manager with the information needed to determine which endpoint should service the request based on a DNS query. No actual endpoint traffic routes through Traffic Manager.  
The below diagram shows how Traffic Manager directs users to one of a set of endpoints:

  
   
TRAFFIC MANAGER WORKFLOW

1. User traffic to company domain name: The client requests information using the company domain name. The goal is to resolve a DNS name to an IP address. Company domains must be reserved through normal internet domain name registrations that are maintained outside of Traffic Manager. In Figure 1, the example company domain is www.fabrikam.com.

2. Company domain name to Traffic Manager domain name: The DNS resource record for the company domain points to a Traffic Manager domain name maintained in Azure Traffic Manager. This is achieved by using a CNAME resource record that maps the company domain name to the Traffic Manager domain name. In the example, the Traffic Manager domain name is fabrikamweb.trafficmanager.net.

3. Traffic Manager domain name and profile: The Traffic Manager domain name is part of the Traffic Manager profile. The user's DNS server sends a new DNS query for the Traffic Manager domain name (in our example, fabrikamweb.trafficmanager.net), which is received by the Traffic Manager DNS name servers.

4. Traffic Manager profile rules processed: Traffic Manager uses the specified load balancing method and monitoring status to determine which Azure or other endpoint should service the request.

5. Endpoint domain name sent to user: Traffic Manager returns a CNAME record that maps the Traffic Manager domain name to the domain name of the endpoint. The user's DNS server resolves the endpoint domain name to its IP address and sends it to the user.

6. User calls the endpoint: The user calls the returned endpoint directly using its IP address.

### On-Premises to Azure Connectivity

Azure Hybrid Connectivity discusses several options available within the Azure Platform, to establish end-to-end hybrid connectivity. This can be between an on-premises network and one or more Azure regions, or between multiple Azure regions. This lesson discusses some reference architectures, the different solutions available for each challenge, and some overall guidelines and concepts.

After completing this lesson, you will be able to:

• Understand Azure Hybrid Connectivity.

• Architecting Azure VNET Peering and understanding the use cases.

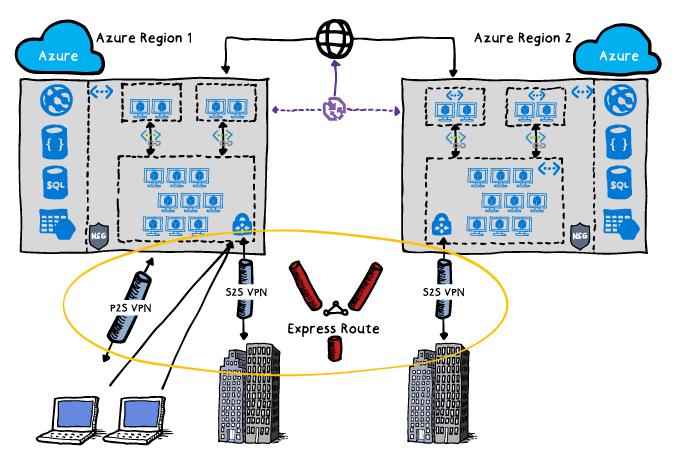
• Understanding the differences between Site-to-Site VPN and Point-to-Site VPN.

• Understand the key capabilities and characteristics of Azure ExpressRoute.

**On-Premises to Azure Connectivity**

There are three primary options to connect your on-premises data center to Azure, described in the following table.

| **Connectivity** | **Benefits** |
| --- | --- |
| ExpressRoute | ·     ExpressRoute as primary cross-premises connectivity  ·     Multiple circuits for redundancy & better routing  ·     ExpressRoute-VPN co-existence for highly available, redundant paths |
| Site-to-Site VPN | ·     S2S VPN over internet for remote branch locations  ·     BGP & active-active configuration for HA and transit |
| Point-to-Site VPN | ·     P2S VPN for mobile users & developers to connect from anywhere with macOS & Windows  ·     AD/radius authentication for enterprise grade security |



### VNET Peering

VNET peering allows you to interconnect 2 Azure Regions with each other, using the Microsoft Backbone (not the public internet). Communication relies on internal IP addressing.

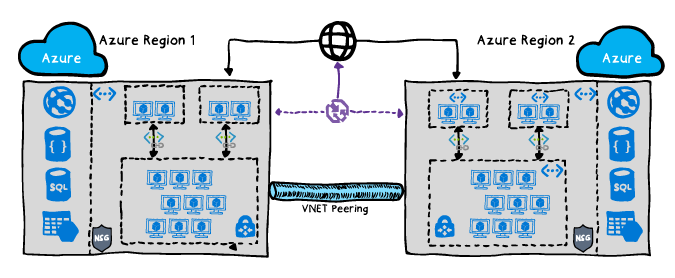
Here are some of the primary features of VNET Peering:

• VNET Peering allows you to interconnect 2 Azure VNET as if they are 1 large VNET.

• VNET Peering is possible within the same Azure region, or across Azure regions (using MS Backbone, no public internet).

• VNET Peering is supported to interconnect an Azure Classic VNET with an ARM VNET (e.g., For migrating workloads).

If VNET peering is not an option, because you might want to encrypt your traffic within the VNET tunnel, one can still deploy a VPN Gateway on both Azure Regional VNETs and creating a Site-to-Site VPN tunnel across those regions.



### Multi-Region VPN Connectivity

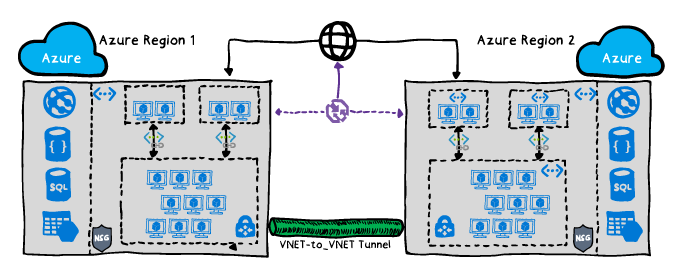
**Forced Tunnelling**

By using Forced Tunnelling, Azure traffic can be rerouted to an on-premises virtual network, to be routed through an existing Site-to-Site VPN or ExpressRoute, into the internal Azure VNET. This is a massive improvement from a security standpoint, as your internal Azure VMs are no longer accessible through the public internet.

**Securing Access to PaaS Services**

A similar concept now exists, to limit network access to PAAS Services in Azure. In a typical scenario, these PAAS services are/were accessible from the public internet. But that’s not what all customers want. Maybe you want your application services endpoints in PAAS to be only accessed from the internal Azure VNETs.

The solution to this is using VNET service endpoints, where you define which PAAS services are no longer accessible through the public internet.



    **Note**: For now, this feature is only available to Azure Storage Accounts, SQL DB Services in PAAS and Web Apps. More PAAS Services will be integrated with this feature soon enough.

### Network Security Groups

This section briefly talks about Network Security Groups and how they can be used to secure connections in a hybrid or external connectivity scenario.

After completing this lesson, you will be able to:

• Determine how and when to use Network Security Groups to secure access to and from a VM.

**Network Security Groups**

Network security groups are different than endpoint-based ACLs. Endpoint ACLs (Access Control Lists) work only on the public port that is exposed through the input endpoint. An NSG works on one or more VM instances and controls all the traffic that is inbound and outbound. You can associate an NSG to a VM, or to a subnet within a VNet. When associated with a VM, the NSG applies to all the traffic that is sent and received by the VM instance. When applied to a subnet within your VNet, it applies to all the traffic that is sent and received by ALL the VM instances in the subnet. A VM or subnet can be associated with only 1 NSG, and each NSG can contain up to 200 rules. You can have 100 NSGs per subscription. on the VM.

**Managing Network Security Groups**

A NSG is a top level object that is associated to your subscription. An NSG contains access control rules that allow or deny traffic to VM instances. The rules of an NSG can be changed at any time, and changes are applied to all associated instances.

A network security group has a Name, is associated to a Region, and has a descriptive label. It contains two types of rules, Inbound and Outbound. The Inbound rules are applied on the incoming packets to a VM and the Outbound rules are applied to the outgoing packets from the VM. The rules are applied at the host where the VM is located. An incoming or outgoing packet has to match an Allow rule for it be permitted, if not it will be dropped.

Rules are processed in the order of priority. For example, a rule with a lower priority number (e.g. 100) is processed before rules with a higher priority numbers (e.g. 200). Once a match is found, no more rules are processed.  
Default Network Security Group Rules

An NSG contains default rules. The default rules cannot be deleted, but because they are assigned the lowest priority, they can be overridden by the rules that you create. The default rules describe the default settings recommended by the platform. While connectivity to the internet is allowed for Outbound direction, it is by default blocked for Inbound direction. There is a default rule to allow Azure’s load balancer (LB) to probe the health of the VM. You can override this rule if the VM or set of VMs under  
the NSG does not participate in the load balanced set.

**Inbound**

| **NAME** | **PRIORITY** | **SOURCE IP** | **SOURCE PORT** | | **DESTINATION IP** | **DESTINATION PORT** | **PROTOCOL** | **ACCESS** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ALLOW VNET INBOUND | 65000 | VIRTUAL\_NETWORK | \* | | VIRTUAL\_NETWORK | \* | \* | ALLOW |
| ALLOW AZURE LOAD BALANCER INBOUND | 65001 | AZURE\_LOADBALANCER | \* | | \* | \* | \* | ALLOW |
| DENY ALL INBOUND | 65500 | \* | \* | | \* | \* | \* | DENY |
|  |  |  |  |  |  |  |  |  |

**Outbound**

| **NAME** | **PRIORITY** | **SOURCE IP** | **SOURCE PORT** | **DESTINATION IP** | **DESTINATION PORT** | **PROTOCOL** | **ACCESS** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ALLOW VNET OUTBOUND | 65000 | VIRTUAL\_NETWORK | \* | VIRTUAL\_NETWORK | \* | \* | ALLOW |
| ALLOW INTERNET OUTBOUND | 65001 | \* | \* | INTERNET | \* | \* | ALLOW |
| DENY ALL OUTBOUND | 65500 | \* | \* | \* | \* | \* | DENY |

### Who Is the Customer?

In this case study, we will look at a customer problem that requires an architectural recommendation.

After this case study, you should:

• Identify customer problems as they are related to networking.

• Design a solution that will meet the customer’s objectives.

• Ensure your designed solution accounts for customer objections.



**Who Is the Customer?**

Fabrikam Residences (*http://fabrikamresidences.com*) is a national real estate services group whose rapid growth was being slowed by an expensive and unresponsive datacenter infrastructure. Fabrikam has two data centers in the United States, but it really doesn’t want to be in the datacenter business. “We are a national real estate firm,” says Craig Jones, Chief Information Officer for Fabrikam. “We want to make investments that support our core business, and buying and managing servers is not our core business. In fact, we have what we call a DOS strategy ‘don’t own stuff.’ We were not an asset-intensive organization in any area but IT, where we had many underutilized assets.”

**What Does the Customer Already Have?**

Fabrikam has about 250 servers in its datacenter in California, and another 110 in its Virginia datacenter, and hundreds of servers scattered across several branch offices throughout the United States. Fabrikam ended up overprovisioning servers each time it deployed an application to ensure that capacity would be there at peak times. This meant that millions of dollars’ worth of hardware and software was sitting idle much of the time.

In addition to the primary data centers, Fabrikam also has several branch offices scattered across the United States that have connectivity to the primary data center through an MPLS based wide area network. Their partner is a Microsoft Azure ExpressRoute partner. To reduce costs, Fabrikam has made the decision to move its West coast datacenter to a colocation site in Silicon Valley and to virtualize the remainder of the servers in its branch offices and Virginia data center into the cloud. Fabrikam’s current virtualization and management solution is based on System Center so a solution that integrates well with these known tools is ideal.

### What Is the Customer's Goal?

**What Is the Customer's Goal?**

Fabrikam Residences would like to eventually migrate the majority of their workloads to Azure. There are several workloads that will be migrated, but the most critical for Fabrikam is their CRM application. The CRM application is a custom web application that runs on IIS 8 and SQL Server 2012 that stores sensitive documents for all of their customer’s transactions. This application needs to perform well at peak time while mitigating the problem of overprovisioned capacity. The centralized nature of the application means that any downtime will block the activity of a significant portion of the company so the solution must be highly available. Due to the sensitive nature of this application security is key so access to the application is restricted to only authorized users from the corporate network including branch offices.

**What Does the Customer Need?**

• Reduce the number of existing on-premises servers through public cloud consolidation to reduce the costs of their current overprovisioned deployments. Servers running in the Virginia data center and remote branch offices will be moved to the closest Azure region. Due the sheer amount of servers being virtualized latency and performance of the network is a big concern.

• Because of the sensitive nature of the data that Fabrikam Real Estate works with ensuring the security and privacy of their infrastructure connects through is critical.

• As part of the migration efforts the CRM application must be deployed in a way that mitigates their current problem of overprovisioning capacity when not needed but able to scale to meet peak demand. The CRM application must be highly available and only accessible from the corporate intranet.

### What Things Worry the Customer?

• We have a national business and we need connectivity that can accommodate connectivity from coast-to-coast.

• Our workloads are very seasonal. I do not want to pay for more resources than I need.

• The data that crosses our network is very confidential. Is Azure

• I need to deploy an intranet-based solution and I have heard that Azure requires an on-premises load-balancer for internal facing workloads.

• I have heard that the public IP address of an Azure deployment can change and break my application.

• My workloads require static IP addresses. I have heard Azure does not support this scenario.

• I have some workloads that require multiple network interfaces on my virtual machines.

• Some deployments require the segmenting of network traffic. Does Azure support this?

### Case Study Solution

**Preferred Target Audience:** Craig Jones, Chief Information Officer for Fabrikam.

The primary audience is the business decision makers and technology decision makers. From the case study scenario, this would include the IT Director, Network Administrator and Security Lead.

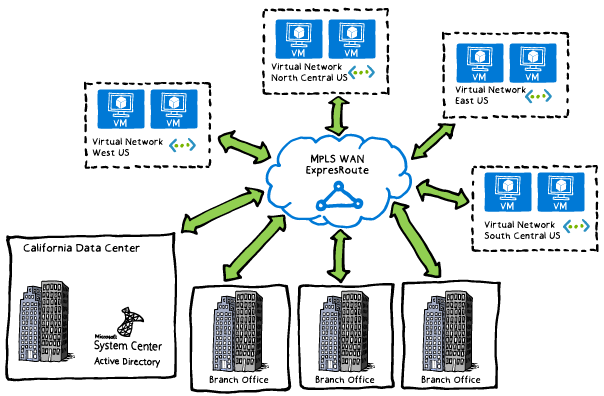
**Preferred Solution**

Fabrikam Residences went with Azure ExpressRoute. They already had a relationship with a Network Service Provider and their MPLS WAN was already in place so it made logical sense to extend their network with Azure. ExpressRoute provides the secure and private connection they need to ensure the privacy of their customer records along with the high speed and low latency connectivity their workloads require. The first step was to configure ExpressRoute by first contacting AT&T and start the onboarding process for ExpressRoute.  
After a circuit was in place with their provider, the next step was to implement Azure Virtual Networks in each of the regions where they would be migrating workloads to the cloud.

They connected the virtual networks across several regions where their branch offices are located to the ExpressRoute circuit using the PowerShell **New-AzureDedicatedCircuitLink** cmdlet.  
The next step was to deploy Active Directory in each of the regions. Each Active Directory DC should be deployed with a static IP into a subnet that does not contain non-static IP based VMs. There should be at least two DCs for redundancy and deployed into an availability set. An Active Directory site should be configured so authentication requests stay local.  
The final step was to architect the solution for their CRM solution to ensure it met the requirements of being secure, highly available, and easily scalable for peak demand. The CRM web servers are deployed into an availability set and auto scale is configured to avoid over provisioning. Access to CRM is through an internally load-balanced endpoint and the SQL Server deployment uses **AlwaysOn** availability groups as well as an internal load balanced IP for the listener.

### Example of a Preferred Solution

**Deploying ExpressRoute with MPLS Network**

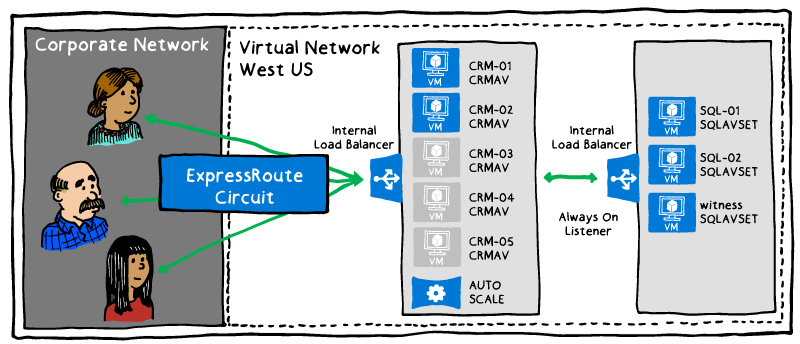


DEPLOYING EXPRESSROUTE

Each region should have at least two domain controllers configured within an availability set. Active Directory sites and site links should be configured to authentication to the local DCs first.  
 

DEPLOYMENT OF ACTIVE DIRECTORY

The CRM webservers are deployed with auto scale enabled and configured using the internal load balancer. SQL Always On is configured using the internal load balancer.



 CUSTOM CRM APPLICATION DEPLOYMENT

### Checklist of Potential Benefits

**Increased Security/Privacy and Network Performance for Enterprise Connectivity**

Azure ExpressRoute provides a dedicated connection between an organizations network to Microsoft Azure through an ExpressRoute partner. Your connection is dedicated bandwidth between your infrastructure and Microsoft Azure. This committed bandwidth and additional control gives you predictable network performance.

Your connection is private. Traffic that flows between your on-premises network and Microsoft Azure does not traverse the public internet and is isolated using industry standard VLANs. ExpressRoute connections support high throughput and low latency for some of the most bandwidth hungry applications and workloads.

For workloads where large amounts of data are leaving the Microsoft Azure data center ExpressRoute can save significant amounts of money due to the included bandwidth and lower bandwidth costs that come with an ExpressRoute circuit.

**Public Peering**

In addition to providing private connectivity between your on-premises network and your Azure virtual networks you can enable public peering which provides private connectivity to a number of Azure public services.

**Cross Region Connectivity**

Azure ExpressRoute makes it simple to connect multiple virtual networks to the same ExpressRoute circuit as long as the virtual networks are on the same continent. This allows you to extend your on-premises network to multiple Azure regions.

**Network Service Provider Model of ExpressRoute Provides a Simple Integration Point**

Fabrikam already has an existing MPLS VPN with a Network Service Provider. With ExpressRoute choosing the Network Service Provider model the provider is responsible for onboarding your network into Azure. They take care of the routing configuration and ensuring everything works. You are still responsible for choosing your service tier and creating the circuit in Azure that they will setup. Bandwidth options for a Network Service Provider range from 10 Mbps all the way up to 1 Gbps. With a network service provider bandwidth is unlimited and not separately charged. When a virtual network in Azure is configured in this manner the virtual network will be accessible to you just like any other site on your wide area network.

### Checklist of Preferred Proof of Concept Potential Flow/Scope

Checklist of Preferred Proof of Concept Potential Flow/Scope

**Objectives:**

* Identify connectivity and latency requirements for proof of concept (regions and site connectivity).
* Configure AD sites/subnets to ensure efficient replication and authentication for the sites.
* Deploy custom CRM intranet based workload securely and without over provisioning.
* Demonstrate that Azure Virtual Machines and Virtual Networks can deliver the connectivity requirements for the solution.
* Address and resolve technical issues involved with connecting and deploying the virtual machines in the proof of concept

**Flow/Scope of the proof of concept (a list of 2-3 bullets):**

* Contact Network Service Provider and sign up for Azure ExpressRoute.
* Provision a circuit in Azure and work with Network Service Provider to connect the new circuit to the existing MPLS network.
* Identify services (virtual machines), regions and on-premises sites to design the network architecture.
* Configure Azure Virtual Networks for connectivity:

--Design subnets for each virtual network to accommodate growth but not overlap any of the on-premises sites or other virtual networks.

--Define the local network sites that the virtual networks will connect to. This may be other virtual networks (connecting across regions) or connecting to one or more of the branch offices.

--Create the dynamic gateways at each virtual network site.

* Connect each virtual network to the ExpressRoute circuit using the **New-AzureDedicatedCircuitLink** PowerShell cmdlet.
* Deploy the virtual machines for Active Directory. There should be two domain controllers in an availability set for each virtual network. The virtual network should reference the IP addresses of both DCs. Sites and site links should be configured per virtual network to ensure Active Directory traffic stays local.
* Deploy the virtual machines for the custom CRM intranet based workload into a virtual network using an internal load balancer configuration for the web servers and another internal load balancer for the SQL Server Always On listener.

**Conditions of satisfaction / success criteria for the PoC:**

* + Demonstrate that Azure Virtual Networks and Virtual Machines can provide the connectivity requirements and capacity for their virtualization.
  + Ensure that AD replication and authentication occurs.
  + Ensure the CRM intranet solution scales without overprovisioning and connectivity is secure.

**Resources / Bill of Materials that you would use:**

* + Azure Virtual Networks and Virtual Machines
  + Azure ExpressRoute and Network Service Provider partner
  + Partner / MCS

### Checklist of Preferred Objection Handled

**We have a national business and we need connectivity that can accommodate coast-to-coast.**

* + Microsoft Azure ExpressRoute can provide connectivity to virtual networks on the same continent. The virtual networks do not even have to reside in the same Azure Subscription.

**I need to deploy an intranet-based solution and I have heard that Azure requires an on-premises load balancer for internal facing workloads.**

* + Microsoft Azure now supports configuring an internal load-balancer using an internal IP address from your virtual network. You can load balance up to 50 virtual machines in a single load-balanced set.

**I have heard that the public IP address of an Azure deployment can change and break my application.**

* + Azure virtual machines now support reserved IP addresses. Reserved IPs allow you to assign an IP address to a virtual machine deployment as the public IP. Even if you shut down all of the virtual machines or delete them and recreate them you can re-use the reserved IP address.

**My workloads require static IP addresses. I have heard Azure does not support this scenario.**

* + Microsoft Azure now supports deploying virtual machines with static IP addresses in virtual networks.

**Network security is critical to our business. Is Azure secure?**

* + Customers often make broad statement that we cannot use public cloud because of the security concerns. We need to make sure we understand their specific concerns. Usually it falls in either of the four buckets:
    - **Trust**—To build trust make sure customer is aware of Microsoft history & experience of delivering cloud services at scale, take them to datacenter tours and take accountability of their success.
    - **Privacy**—Privacy is one of the foundations of Microsoft’s Trustworthy Computing. Microsoft has a longstanding commitment to privacy, which is an integral part of our product and service lifecycle. Share the Microsoft Azure Privacy Statement that describes the specific privacy policy and practices that govern customers’ use of Microsoft Azure.
    - **Compliance**—Microsoft partners with customers to help them address a wide range of international, country/region, and industry-specific regulatory requirements. Microsoft provides Microsoft Azure customers with detailed information about our security and compliance programs, including audit reports and compliance packages, to help customers assess our services against their own legal and regulatory requirements.
    - **Security of Infrastructure and services**—Microsoft Azure runs in geographically dispersed datacenters that comply with key industry standards, such as ISO/IEC 27001:2005, for security and reliability. They are managed, monitored, and administered by Microsoft operations staff (Global foundation Services) that have years of experience in delivering the world’s largest online services with 24 x 7 continuity.

    For more information, visit Microsoft Azure Trust Center and familiarize yourself with Microsoft Azure Security practices. (Links are provided further in this section.)

**AWS supports setting ACLs within subnets of their VPC.**

* + Azure supports setting ACLs on endpoints. As a workaround you can isolate networks in this manner. You can also use the firewall within the guest OS (if Windows you can use Group Policy).

  **Reference Link**: <https://azure.microsoft.com/blog/new-windows-azure-network-security-whitepaper/>

 **Reference Link:**<http://azure.microsoft.com/support/trust-center/>

### Lab Steps

#### Deploying Network Infrastructure for Use in Azure Solutions

NOTE: For the most recent version of this online lab, see: https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Configure the lab environment

##### Task 1: Open the Azure Portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you intend to deploy resources in this lab
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab0901-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 3: Install the Azure Building Blocks npm package in Azure Cloud Shell

1. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to create a local directory to install the Azure Building Blocks npm package:

mkdir ~/.npm-global

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to update the npm configuration to include the new local directory:

npm config set prefix '~/.npm-global'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the ~./bashrc configuration file for editing:

vi ~/.bashrc

1. At the **Cloud Shell** command prompt, in the vi editor interface, scroll down to the bottom of the file (or type **G**), scroll to the right to the right-most character on the last line (or type **$**), type **a** to enter the **INSERT** mode, press **Enter** to start a new line, and then type the following to add the newly created directory to the system path:

`

1. At the **Cloud Shell** command prompt, in the vi editor interface, to save your changes and close the file, press **Esc**, press **:**, type **wq!** and press **Enter**.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to install the Azure Building Blocks npm package:

npm install -g @mspnp/azure-building-blocks

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to exit the shell:

exit

1. In the **Cloud Shell timed out** pane, click **Reconnect**.**Note**: You need to restart Cloud Shell for the installation of the Buliding Blocks npm package to take effect.

##### Task 3: Prepare Building Blocks Hub and Spoke parameter files

1. In the **Cloud Shell** pane, click the **Upload/Download files** icon and, in the drop-down menu, click **Upload**.
2. In the **Open** dialog box, navigate to the **\allfiles\AZ-301T04\Module\_03\LabFiles\Starter\** folder, select the **hub-nva.json** file, and click **Open**.
3. Repeat the previous step to upload to **Cloud Shell** the remaining files in the **\allfiles\AZ-301T04\Module\_03\LabFiles\Starter\** folder, including **hub-vnet.json**, **hub-vnet-peering.json**, **spoke1.json**, and **spoke2.json**.
4. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **hub-vnet.json** Building Blocks parameter file:

sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/hub-vnet.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the **hub-vnet.json** Building Blocks parameter file:

sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/hub-vnet.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **hub-vnet.json** Building Blocks parameter file:

cat ~/hub-vnet.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **hub-nva.json** Building Blocks parameter file:

sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/hub-nva.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the **hub-nva.json** Building Blocks parameter file:

sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/hub-nva.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **hub-nva.json** Building Blocks parameter file:

cat ~/hub-nva.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **spoke1.json** Building Blocks parameter file:

sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/spoke1.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in **spoke1.json** the Building Blocks parameter file:

sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/spoke1.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **spoke1.json** Building Blocks parameter file:

cat ~/spoke1.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminUsername** parameter with the value **Student** in the **spoke2.json** Building Blocks parameter file:

sed -i.bak1 's/"adminUsername": ""/"adminUsername": "Student"/' ~/spoke2.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to replace the placeholder for the **adminPassword** parameter with the value **Pa55w.rd1234** in the **spoke2.json** Building Blocks parameter file:

sed -i.bak2 's/"adminPassword": ""/"adminPassword": "Pa55w.rd1234"/' ~/spoke2.json

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to verify that the parameter values were successfully changed in the **spoke2.json** Building Blocks parameter file:

cat ~/spoke2.json

##### Task 4: Implement the hub component of the Hub and Spoke design

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub virtual network:

RESOURCE\_GROUP\_HUB\_VNET='AADesignLab08-hub-vnet-rg'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment (replace the placeholder <Azure region> with the name of the Azure region to which you intend to deploy resources in this lab):

LOCATION='<Azure region>'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the hub component of the Hub-and-Spoke topology by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP\_HUB\_VNET -s $SUBSCRIPTION\_ID -l $LOCATION -p ~/hub-vnet.json --deploy

1. Do not wait for the deployment to complete but proceed to the next task.

##### Task 5: Implement the spoke components of the Hub and Spoke design

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the same user account account that you used earlier in this lab.
4. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the first spoke virtual network:

RESOURCE\_GROUP\_SPOKE1\_VNET='AADesignLab08-spoke1-vnet-rg'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the first spoke component of the Hub-and-Spoke topology by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP\_SPOKE1\_VNET -s $SUBSCRIPTION\_ID -l $LOCATION -p ~/spoke1.json --deploy

1. Do not wait for the deployment to complete but proceed to the next step.
2. On the Taskbar, click the **Microsoft Edge** icon.
3. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
4. If prompted, authenticate with the same user account account that you used earlier in this lab.
5. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
6. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the second spoke virtual network:

RESOURCE\_GROUP\_SPOKE2\_VNET='AADesignLab08-spoke2-vnet-rg'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the second spoke component of the Hub-and-Spoke topology by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP\_SPOKE2\_VNET -s $SUBSCRIPTION\_ID -l $LOCATION -p ~/spoke2.json --deploy

1. Do not wait for the deployment to complete but proceed to the next task.

##### Task 6: Configure the VNet peering of the Hub and Spoke design

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the same user account account that you used earlier in this lab.
4. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that contains the hub virtual network:

RESOURCE\_GROUP\_HUB\_VNET='AADesignLab08-hub-vnet-rg'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to provision peering of the virtual networks in the Hub-and-Spoke topology by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP\_HUB\_VNET -s $SUBSCRIPTION\_ID -l $LOCATION -p ~/hub-vnet-peering.json --deploy

1. Do not wait for the deployment to complete but proceed to the next task.

##### Task 7: Configure routing of the Hub and Spoke design

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. If prompted, authenticate with the same user account account that you used earlier in this lab.
4. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
5. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of your Azure subscription:

SUBSCRIPTION\_ID=$(az account list --query "[0].id" | tr -d '"')

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the name of the resource group that will contain the hub Network Virtual Appliance (NVA) functioning as a router:

RESOURCE\_GROUP\_HUB\_NVA='AADesignLab08-hub-nva-rg'

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a variable which value designates the Azure region you will use for the deployment:

LOCATION=$(az group list --query "[?name == 'AADesignLab08-hub-vnet-rg'].location" --output tsv)

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to deploy the NVA component of the Hub-and-Spoke topology by using the Azure Building Blocks:

azbb -g $RESOURCE\_GROUP\_HUB\_NVA -s $SUBSCRIPTION\_ID -l $LOCATION -p ~/hub-nva.json --deploy

1. Wait for the deployment to complete before you proceed to the next task.**Note**: The deployment can take about 10 minutes.

#### Exercise 2: Review the Hub-spoke topology

##### Task 1: Examine the peering configuration

1. In the hub menu in the Azure portal, click **All services**.
2. In the **All services** menu, in the **Filter** text box, type **Virtual networks** and press **Enter**.
3. In the list of results, click **Virtual networks**.
4. On the **Virtual networks** blade, click **hub-vnet**.
5. On the **hub-vnet** blade, click **Peerings**.
6. On the **hub-vnet - Peerings** blade, review the list of peerings and their status.
7. Navigate back to the **Virtual Networks** blade and click **spoke1-vnet**.
8. On the **spoke1-vnet** blade, click **Peerings**.
9. On the **spoke1-vnet - Peerings** blade, review the existing peering and its status.
10. Navigate back to the **Virtual Networks** blade and click **spoke2-vnet**.
11. On the **spoke2-vnet** blade, click **Peerings**.
12. On the **spoke2-vnet - Peerings** blade, review the existing peering and its status.

##### Task 2: Examine the routing configuration

1. In the **All services** menu, in the **Filter** text box, type **Route tables** and press **Enter**.
2. In the list of results, click **Route tables**.
3. On the **Route tables** blade, click **hub-dmz-rt**.
4. On the **hub-dmz-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the routes **toSpoke1** and **toSpoke2**.
5. Navigate back to the **Route tables** blade and click **spoke1-rt**.
6. On the **spoke1-rt** blade, click **Peerings**.
7. On the **spoke1-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the route **toSpoke2**.
8. Navigate back to the **Route tables** blade and click **spoke2-rt**.
9. On the **spoke2-rt** blade, click **Peerings**.
10. On the **spoke2-rt** blade, review the list of routes. Note the **NEXT HOP** entry for the route **toSpoke1**.

##### Task 3: Verify connectivity between spokes

1. In the hub menu in the Azure portal, click **All services**.
2. In the **All services** menu, in the **Filter** text box, type **Network Watcher** and press **Enter**.
3. In the list of results, click **Network Watcher**.
4. On the **Network Watcher** blade, in the **NETWORK DIAGNOSTIC TOOLS** section, click **Connection troubleshoot**.
5. On the **Network Watcher - Connection troubleshoot** blade, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** drop-down list, select the **AADesignLab08-spoke1-vnet-rg** entry.
   * In the **Virtual machine** drop-down list, leave the default entry.
   * Leave the **Port** text box blank.
   * Ensure that the **Destination** option is set to **Specify manually**.
   * In the **URI, FQDN, or IPv4** text box, type **10.2.0.68** entry.
   * In the **Port** text, type 3389.
   * Click the **Check** button.
6. Wait until results of the connectivity check are returned and verify that the status is **Reachable**.**Note**: If this is the first time you are using Network Watcher, the check can take up to 5 minutes.

#### Exercise 3: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab08')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab08')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 3 Review Questions

**Virtual networks**  
You are designing a line of business solution for a company. The solution connects to Azure-based resources by using the public Internet.  
You need to recommend a solution to protect network traffic.  
 What should you use? What are the capabilities and limitations of the technology?

Suggested Answer ↓

An Azure Virtual Network (VNET) is the logical unit of multiple or all network resources in an Azure region. From the public Internet, incoming traffic is routed to Public IP addresses of Azure Resources.

**Load balancing**  
A company has an Azure environment that contains a VNET with two subnets.  
You need to recommend a solution to handle all incoming traffic between the subnets.  
What should you recommend?

Suggested Answer ↓

Azure Load Balancers can handle almost any TCP or UDP traffic. An internal load balancer can load balance incoming traffic between Azure Subnets.

**Network types**  
You are designing a solution for a company.  
You need to recommend a solution to connect on-premises resources with resources in Azure.  
 What should you recommend?

Suggested Answer ↓

Azure supports multiple options to connect your on-premises datacenter to Azure including ExpressRoute, Site-to-Site VPN, and Point-to-Site VPN.

### Storage Queues

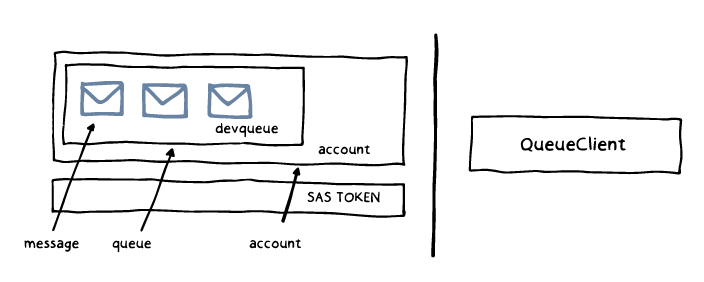
This module describes and compares the integration and messaging services available for solutions hosted on the Azure platform. Messaging services described include Azure Storage Queues, Service Bus Queues, Service Bus Relay, IoT Hubs, Event Hubs, and Notification Hubs. Integration services include Azure Functions and Logic Apps.

After completing this module, students will be able to:

• Compare Storage Queues to Service Bus Queues.

• Identify when to use Azure Functions or Logic Apps for integration components in a solution.

• Describe the differences between IoT Hubs, Event Hubs and Time Series Insights.



**Event Messaging**

This lesson introduces Service Bus and Event Grid as key components of an event messaging architecture hosted in Azure.

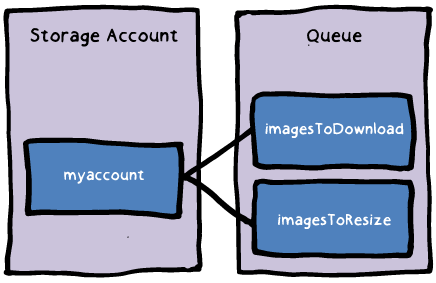
After completing this lesson, you will be able to:

• Describe various Service Bus offerings and compare them to equivalent offerings in other services.

• Identify when to use Event Grid as a messaging component of a solution in Azure.

**Queue Service Components**

The Queue service contains the following components:



**URL** format: Queues are addressable using the following URL format:

http://[account].queue.core.windows.net/<queue>

The following URL addresses one of the queues in the diagram:

http://[account].queue.core.windows.net/imagesToDownload

**Queue**: A queue contains a set of messages. All messages must be in a queue.

**Message**: A message, in any format, of up to 64KB.

### Storage Queue Message Handling

**Storage Queue Message Handling**

Queue messages can be managed in a variety of different ways. The following is a list of actions that you can perform on a queue or it's messages:

**• Create/Delete Queue:** Client SDKs, PowerShell or the REST API can be used to create a new queue instance or remove an  
existing one.

**• Measure Queue Length:** You can get an estimate of the number of messages in the queue. Do to race conditions and technical implementation, this count is not guaranteed and should be treated in your application as an approximate queue length.

**• Insert Message into Queue:** New messages can be added to the queue. These messages have a body that are either a string (in UTF-8 format) or a byte array.

**• Retrieve the Next Message:** A copy of the next available message is retrieved and the message is made invisible for a specific duration. During this time, you can process the message. Other queue consumers will not see this message in the queue when they retrieve messages while it is invisible. After a specific amount of time, the invisibility duration will elapse and the message is available to other queue consumers.

**• Extend Message Lease:** If you need more time to process a retrieved queue message, you can return to the queue and update the invisibility duration for the queue message. This will ensure that the message is not prematurely available to other queue consumers.

**• Peek at the Next Message:** You can retrieve a copy of the next available message in the queue without making the message invisible to other queue consumers.

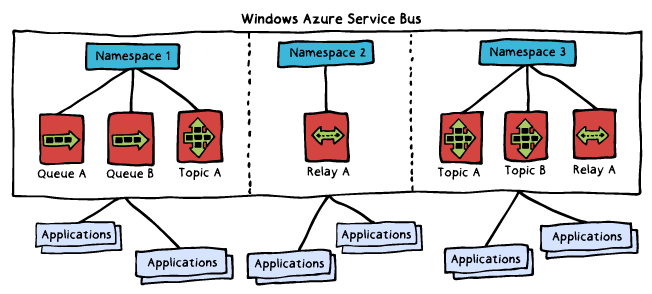
**• Update a Message:** The contents of a retrieved message can be updated in the queue if you need to have a concept of checkpoints or state for queue messages.

**• Delete a Message:** Once you have completed processing a message, you must delete the message from the queue if you do not wish for it to be processed by any more queue consumers. Otherwise the message invisibility will timeout and the message will be processed again by other queue consumers. Because of the workflow, queue messages and their processors/consumers must be designed to be idempotent and should not have side effects from processing the same message multiple times. For example, if a queue message contains data that needs to be stored in a SQL database, your processor should check to see if an existing record exists and then update that record or create a new record. Otherwise, you may end up with false duplicate data from your queue processors processing the same message multiple times.

### Service Bus-managing service (REST, DB)endpoint queueing

Azure Service Bus provides a hosted, secure, and widely available infrastructure for widespread communication, large-scale event distribution, naming, and service publishing. Service Bus provides connectivity options for Windows Communication Foundation (WCF) and other service endpoints—including REST endpoints—that would otherwise be difficult or impossible to reach. Endpoints can be located behind network address translation (NAT) boundaries, or bound to frequently-changing, dynamically-assigned IP addresses, or both.

Service Bus provides both “relayed” and “brokered” messaging capabilities. In the relayed messaging pattern, the relay service supports direct one-way messaging, request/response messaging, and peer-to-peer messaging. Brokered messaging provides durable, asynchronous messaging components such as Queues, Topics, and Subscriptions, with features that support publish-subscribe and temporal decoupling: senders and receivers do not have to be online at the same time; the messaging infrastructure reliably stores messages until the receiving party is ready to receive them.  
Service Bus services are typically partitioned into namespaces as each namespace provide both a service and security boundary.

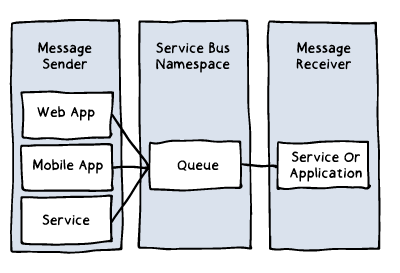


When you create a queue, topic, relay, or Event Hub, you give it a name. Combined with whatever you called your namespace, this name creates a unique identifier for the object. Applications can provide this name to Service Bus, then use that queue, topic, relay, or Event Hub to communicate with one another.

To use any of these objects, Windows applications can use Windows Communication Foundation (WCF). For queues, topics, and Event Hubs Windows applications can also use Service Bus-defined messaging APIs. To make these objects easier to use from non-Windows applications, Microsoft provides SDKs for Java, Node.js, and other languages. You can also access queues, topics, and Event Hubs using REST APIs over HTTP.

### Service Bus Queues-**NUMBER OF NODES /INSTANCES MANAGING MESSAGING** from webapp, service and mobile app

Service Bus Queue is a brokered messaging system, which is similar to the Queue service Azure Storage. By using these queues, application modules that are distributed do not need to communicate directly with each other. These application modules can instead communicate by using the queues. This ensures that there is a separation between message generators and message processors. This separation provides the flexibility of having one or more application component instances that are generating messages and one or more application component instances that are processing the same messages. If an instance encounters an irrecoverable exceptional condition, other instances can continue processing the messages. If the workload for the entire application is increased, new instances can be created to handle the load. These scenarios are common and critical when developing and designing cloud applications. Service Bus queues do implement a familiar first in, first out (FIFO) message delivery strategy. Service Bus queues can also guarantee that a message is received and processed both at least and at most once by the message consumers.

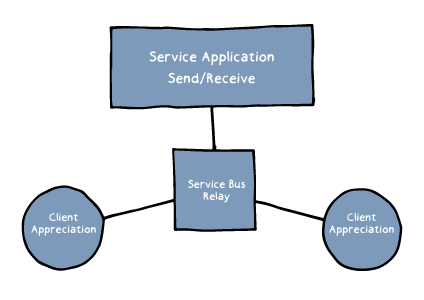


### Service Bus Relay-location/address protection of service NAT

Service Bus includes a relay component that can connect existing services to new client applications without exposing the true location or address of the service. Some of the advantages of using a relay include:

• Services that need to communicate directly with external client applications or devices (For example, mobile devices) typically need to be placed in a special subnet or a virtual network with a unique NAT or firewall configuration. The address for the service endpoint will need to be publicly addressable in order for client devices to connect. In some enterprises, this can be considered dangerous or unacceptable. With Service Bus Relay, the service makes an outbound connection to the relay and bypasses many of the complex network configurations that are necessary for inbound connections.

• Although mobile applications are deployed and updated regularly, end users might not update their applications as regularly as you want them to. If your service needs to be migrated to a new network or moved to a new IP address, this can cause a lapse of connectivity for your mobile applications. Using Service Bus Relay, your mobile applications address a publicly accessible and permanent uniform resource identifier (URI). You are then free to make changes and migrate your service within your organization's infrastructure. The new service instance or location simply needs to connect to the relay for client devices to access it. This enables more mobility for services that are connected to the applications that are already deployed. Service Bus Relay also supports direct peer-to-peer communication. This is negotiated if the relay determines that the connecting application can easily and directly address the service.



### Event Grid-**event routing from source to destination**

Azure Event Grid is a single service designed to managed and route systemic events from any source service in your Azure subscription. Event Grid is designed to eliminate the need to poll existing services and enable applications to run in an event-driven manner where applications are triggered only when needed, and consume compute only when necessary.

Event Grid can be used to publish custom events from your workloads or pre-determined events designed with each Azure service. Event Grid can be used in a variety of ways including:

• Integrate various components of your workloads together using a publish-subscribe mode.

• Enable your solution to listen to events from third-party B2B services of publish events for the third-party services to consume.

• Create serverless compute that is triggered by a specific Azure service event such as the creation of a database or VM.

• Automate the deployment of resources by subscribing to ARM events.

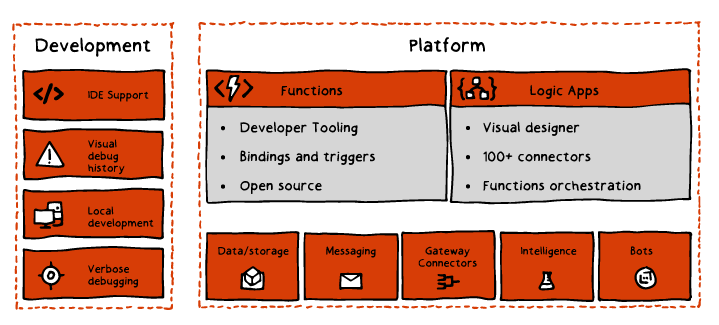
Azure Event Grid allows users to build applications with event-based architectures with ease. You merely select the Azure resource you wish to use, then select the WebHook endpoint or event handler to send the event. Event Grid has integrated support for events coming from Azure services, as well as support for third-party events. You can even use filters to route specific events to different or even multiple endpoints.

An Event Grid instance can use point-and-click to configure events, moving them from your Azure resource to an event handler or endpoint. It also has reliability, allowing the use of a 24-hour retry with exponential backoff so users can ensure events are delivered on time. Event Grid can also build high-volume workloads with support for millions of these events per second, and even custom events can be routed, filtered, and delivered. With all the power of Event Gird, you can rest assured that it is pay per event, meaning that you only pay for what you use with Event Grid.

Event Grid can provide users with several options to improve serverless, integration, and ops automation work. Event Grid can connect data sources and event handlers to provide a serverless application architecture. For example, you can use Event Grid to trigger a function to run image analysis every time a new item becomes added to a blob storage container. Event Grid can also connect an application with other services. As an example, a user can create a storage blob to send your app’s event data to Event grid, while taking advantage of its reliable delivery, direct integration, and advanced routing with Azure. Lastly, it also allows users to speed an automation process. A user can request Event Grid to notify Azure Automation when a virtual machine starts or a SQL Database is spun up as one of many examples. Said events can be used to tag virtual machines, file work items, put metadata into operations tools, or even automatically check to ensure services configurations are correct.

### Serverless Integration

Serverless computing promises agility and power in building the next generation of solutions. See how Azure Logic Apps and Azure Functions provide powerful tools to build serverless applications faster than ever. In video below, you'll see how building an automated email system using only serverless technologies. As the video progresses, it will walk through the steps to integrate the solution with source control and continuous delivery tools.



### Notification Hubs-**this allows SNS notification push**

Notification Hubs is a combination of a service infrastructure and a set of client libraries that allows you to publish push notifications to your mobile applications from any application service component. With Notification Hubs, you can send notifications that are personalized to a specific user, notifications that are distributed to many users across various platforms, and notifications that are filtered to a specific set of users. The Notification Hubs infrastructure abstracts the implementation of the various Platform Notification Systems (PNS) for each mobile platform. By using a single method call, you can send notifications to various device platforms without having to implement a different message structure or communication mechanism for each platform.

You can use notification hubs in a variety of scenarios including:

• Send wide-reaching news notifications to all devices with your mobile application installed.

• Send a notification to a subset of your users that is determined based on a tag, label, or location.

• Send specific notifications to a user for the activities that are related to their specific account.  
   
**Benefits**

Notification hubs eliminate the challenges that are involved in managing push notifications. Notification Hubs use a full multiplatform, scaled-out push notification infrastructure, and considerably reduce the push-specific code that runs in the app. Notification hubs implement all the functionality of a push infrastructure. Devices are only responsible for registering PNS handles, and the backend is responsible for sending platform-independent messages to users or interest groups.

Notification hubs provide a push infrastructure with the following advantages:

**Multiple platforms:**

* Support for all major mobile platforms—Windows, Windows Phone, iOS, and Android.
* No platform-specific protocols. The application only communicates with Notification Hubs.
* Device handle management. Notification Hubs maintains the handle registry and feedback from PNSs.

**Works with any backend -** Works with Cloud or on-premises applications that are written in .NET, PHP, Java, or Node.

**• Scale:** Notification hubs scale to millions of devices without the need of rearchitecting or sharding.

**Rich set of delivery patterns:** Associate devices with tags, representing logical users or interest groups:

* **Broadcast**: Allows for near-simultaneous broadcast to millions of devices with a single Application Programming Interface (API) call.
* **Unicast/Multicast**: Push to tags representing individual users, including all their devices; or a wider group. For example, a user could use the app on separate devices (tablet, phone, etc.) and would require push notifications to either be pushed to all devices or a specific device.
* **Segmentation**: Push to a complex segment that is defined by tag expressions (For example, devices in New York following the Yankees).
* **Personalization:** Each device can have one or more templates to achieve per-device localization and personalization without affecting the backend code.

**Platform Notifications**

At a high level, all platform notification systems follow the same pattern:

1. The client application contacts the PNS to retrieve its handle. The handle type depends on the system. For Windows Notification Service (WNS), it is a URI or notification channel. For Apple Push Notification Service (APNS), it is a token.

2. The client application stores this handle in the app backend for later usage. For WNS, the backend is typically a cloud service. For APNS, the system is called a provider.

3. To send a push notification, the app backend contacts the PNS by using the handle to target an instance of a specific client application.

4. The PNS forwards the notification to the device specified by the handle.

Notification Hubs can be used in flexible ways to register devices and eventually send a message to the devices. Devices can register themselves and receive notifications using the following method:

1. The client device reaches out to the PNS by using the Notification Hubs SDK. It registers a unique PNS handle that is used by the service to send notifications to this device whether the application is running or not.

2. The client device can alternatively send its PNS handle to the application backend to have the application register the device.

3. When the application backend sends a message to the Notification Hubs service, the service handles sending the message to the appropriate target clients by using their registered PNS handles. The application backend simply requests the message is sent and the Notification Hubs service and the PNS handle the actual distribution of messages to client devices.

### Event Hubs **(messaging and images)**

This section covers the two most common real-time message streaming options available in Azure, Event Hubs and IoT Hubs. After completing this section you will be able to:

• Identify the various streaming queue services in Azure.

• Compare and contrast Event Hubs and IoT Hubs.

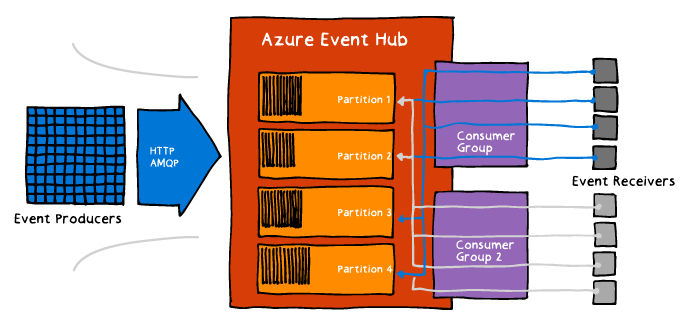
• Architect a real-time streaming solution using Event or IoT Hubs and Azure data processing components.

**Event Hubs**

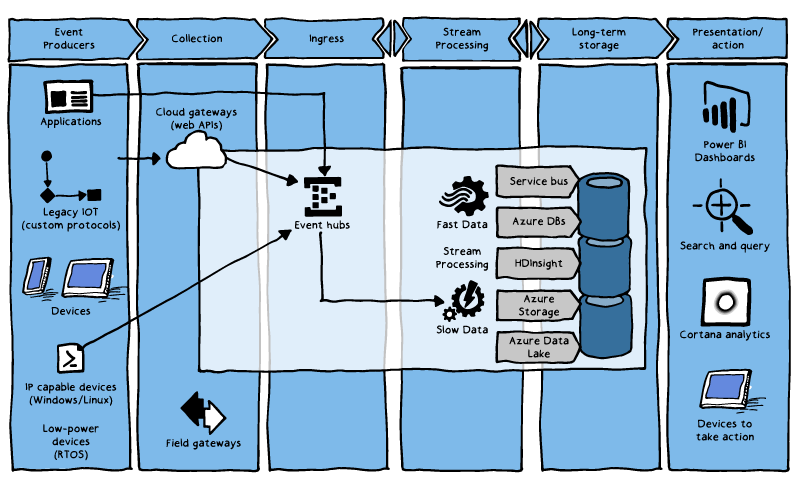
As a scalable data streaming platform with an event ingestion service, Azure Event Hubs can be used to receive and process millions of events per second. It can process telemetry, events, or even information produced by distributed software and devices as well as store said information. Information sent to an event hub can become converted or stored using any real-time analytics provider or batching/storage adapter.

**Azure Event Hubs Features and Role**

Azure Event Hubs provides various features that can assist with storing and processing events. It can provide an entity to send data to event hubs, capable of publishing via AMQP 1.0 or HTTPS. Also, it can capture Event Hubs streaming data to store in an Azure Blob storage account. It can allow you to partition the data, allowing each consumer to only read a specific subset of the event stream, along with the capabilities for each consumer to act independently. Also, they can identify and authenticate the event publisher though Shared Access Signature Token, or SAS Token. Azure Event Hubs use SAS tokens generated from a SAS key to regenerate the hash and authenticate the sender.



Also, Azure Event Hubs also contains Throughput units or Pre-purchased units of capacity. Throughput units can include the capacity of either 1 MB per second or 100 events per second, whichever comes first if its Ingress while maintaining up to 2 MB per second if its egress. These units are billed hourly for a minimum of one hour, and up to a maximum of 20 throughput units per Event Hubs namespace. Azure Event Hubs commonly can be used as a “front line” for an event pipeline of a solution architecture, sometimes known as an event investor. An event investor is a service or component that sits between the event consumers and publishers to separate the production of an event stream that obtained said events. Azure Event Hubs can provide message stream handling, but the capabilities of said service are different from traditional enterprise messaging.  
The following figure depicts this architecture:



### IoT Hubs **(devices)**

Azure IoT hub service is a managed service that can provide reliable and secure bi-directional communication between a multitude of IoT devices with a solution back end. The service can provide a multitude of functions such as providing multiple device-to-cloud and cloud-to-device communication options, a declarative message routing option built in that sends the message to other Azure services, and more.

In addition to its features, Azure IoT Hub provides a way to clarify message routes based on routing rules to control where the hub can send device-to-cloud messages. These rules do not require you to write any code to implement. You can also receive detailed logs for identifying connectivity events. Azure IoT Hub Service and Azure Event Hubs are two separate products despite the similar description. IoT can provide additional features such as Device twins, which can be used to store and research device state information. While IoT provides the device-to-cloud and cloud-to-device communication options mentioned in the previous paragraph, Event hubs can only provide event ingress communication.

When using Azure IoT Hub, it implements a service-assisted communication pattern to act as a mediator between your devices and the solution in the back end. The communication provided to establish bi-directional, reliable communication path between a control system.

To maintain such a communication path, Azure IoT follows a set of principles. It focuses first and foremost on Security, not allowing it to accept any unsolicited network information. It even uses a device dedicated to this, which establish all connections and routes in an outbound direction only. The path between both device and service or even between a device and gateway must be secured at the application protocol layer.  
In an IoT solution, a gateway can typically be either a protocol gateway deployed in a cloud or a field gateway deployed locally with a device. With a field gateway, you can make time-sensitive decisions, run analytics on edge, provide device management services or even enforce security and privacy constraints. With a protocol gateway protocol, translations are carried out, such as MQTT to AMQP.



### Comparing Event Hubs and IoT Hubs

Event Hubs and IoT Hubs are similar services but they each have unique differences detailed in this table.

| **Area** | **IoT Hub** | **Event Hubs** |
| --- | --- | --- |
| Communication patterns | Enables [device-to-cloud communications](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-d2c-guidance) (messaging, file uploads, and reported properties) and [cloud-to-device communications](https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-c2d-guidance) (direct methods, desired properties, messaging). | Only enables event ingress  (usually considered for device-to-cloud  scenarios). |
| Device state information | Device twins can store and query device state information. | No device state information can  be stored. |
| Device protocol support | Supports MQTT, MQTT over WebSockets, AMQP, AMQP over WebSockets, and HTTPS. Additionally, IoT Hub works with the Azure IoT protocol gateway, a customizable protocol gateway implementation to support custom protocols. | Supports AMQP, AMQP over  WebSockets, and HTTPS. |
| Security | Provides per-device identity and revocable access control. See the Security section of the IoT Hub developer guide. | Provides Event Hubs-wide shared access  policies, with limited revocation support through publisher's policies. IoT solutions  are often required to implement a custom solution to support per-device credentials and  anti-spoofing measures. |
| Operations monitoring | Enables IoT solutions to subscribe to a rich set of device identity management and connectivity events such as individual device authentication errors, throttling, and bad format exceptions. These events enable you to quickly identify connectivity problems at the individual device level. | Exposes only aggregate metrics. |
| Scale | Is optimized to support millions of  simultaneously connected devices. | Meters the connections as per Azure  Event Hubs quotas. On the other hand,  Event Hubs enables you to specify  the partition for each message sent. |
| Device SDKs | Provides device SDKs for a large variety of platforms and languages, in addition to direct MQTT, AMQP, and HTTPS APIs. | Is supported on .NET, Java, and C, in  addition to AMQP and HTTPS send  interfaces. |

| **Area** | **IoT Hub** | **Event Hubs** |
| --- | --- | --- |
| File upload | Enables IoT solutions to upload files from devices to the cloud. Includes a file notification endpoint for workflow integration and an operations monitoring category for debugging support. | Not supported. |
| Route messages to multiple endpoints | Up to 10 custom endpoints are supported. Rules determine how messages are routed to custom endpoints. For more information, see Send and receive messages with IoT Hub. | Requires additional code to be  written and hosted for message  dispatching. |

**IoT Remote Monitoring**

You can use an IoT Hub to receive messages from a device and then push them to a Logic App for processing. The Logic App can send notifications for messages that cross specific thresholds and store aggregate data about all messages to a database.

### Time Series Insights

The Azure Time Series Insights service is a product built for visualizing, storing, and querying vast amounts of time series information, such as information generated by IoT devices. If you have plans to store, manage query, or even visualize time series data in the cloud, Azure Time Series Insights may be the product for you.

For those unsure, whether their information could be considered time series a few factors can help determine the difference for you. By definition, time series data shows how an asset or process changes over time. It has a timestamp that is unique to its kind which is most meaningful as an axis. The time series data arrives in time order and can usually be an insert rather than an update to your database. Because time series data obtains and stores every new event as a row, the changes can be measured over a time frame, enabling one to not only look back into the past but also predict the future with the data. You may want to consider using Azure Time Series Insights in various situations, including:

**• Storing and maintaining time series data in a scalable format:** Azure Time Series Insights provides a database with time series data in mind. It handles the work of storing and maintaining events due to its nature as a scalable and fully managed database.

**• Near real-time data visualization:** Azure Time Series Insights can provide an explorer to visualize data streaming into an environment. When you connect an event source, the event data can be viewed, queried, and explored with Azure Time Series Insights.

**• Producing customer applications:** You can build applications that use time series data using the exposed REST Query APIs provided by Azure Time Series Insights.  
 **• Needing a global view of time series data streaming from different locations for multiple sites or assets:** Azure Time Series Insights allows you to connect multiple event sources to the environment. It means that data streaming in different or multiple locations can be viewed together in near real-time. Users of Azure Time Series Insights can use this visibility to share data with business leaders as well as provide better collaboration with domain experts who can apply their expertise to help solve problems.

Azure Time Series also has a multitude of capabilities. It requires no upfront data preparation, making setup a quick process. It can obtain and store millions of sensor events per day with a one-minute latency, giving it the ability to provide near real-time insights. One can even embed Azure Time Series Insights data into existing applications, allowing one to create custom solutions with sharing capabilities for others to explore your insights.

### Time Series Insights

The Azure Time Series Insights service is a product built for visualizing, storing, and querying vast amounts of time series information, such as information generated by IoT devices.

### Lab Steps

#### Online Lab: Deploying Messaging Components to Facilitate Communication Between Azure Resources

NOTE: For the most recent version of this online lab, see: <https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign>

#### Before we start

1. Ensure that you are logged in to your Windows 10 lab virtual machine using the following credentials:
   * Username: **Admin**
   * Password: **Pa55w.rd**
2. Review Taskbar located at the bottom of your Windows 10 desktop. The Taskbar contains the icons for the common applications you will use in the labs:
   * Microsoft Edge
   * File Explorer
   * [Visual Studio Code](https://code.visualstudio.com/)
   * [Microsoft Azure Storage Explorer](https://azure.microsoft.com/features/storage-explorer/)
   * Bash on Ubuntu on Windows
   * Windows PowerShell

**Note**: You can also find shortcuts to these applications in the **Start Menu**.

#### Exercise 1: Deploy a Service Bus namespace

##### Task 1: Open the Azure portal

1. On the Taskbar, click the **Microsoft Edge** icon.
2. In the open browser window, navigate to the **Azure Portal** ([https://portal.azure.com](https://portal.azure.com/)).
3. When prompted, authenticate with the user account account that has the owner role in the Azure subscription you will be using in this lab.

##### Task 2: Create a Service Bus namespace

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Service Bus** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Service Bus**.
4. On the **Service Bus** blade, click the **Create** button.
5. On the **Create namespace** blade, perform the following tasks:
   * In the **Name** text box, enter a globally unique name.
   * In the **Pricing tier** drop-down list, select the **Basic** option.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, ensure that the **Create new** option is selected and then, in the text box, type **AADesignLab1101-RG**.
   * In the **Location** drop-down list, select the Azure region to which you intend to deploy resources in this lab.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.

##### Task 3: Create a Service Bus Queue

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the newly created Service Bus namespace.
4. On the Service Bus namespace blade, in the **ENTITIES** section, click **Queues**.
5. On the Service Bus namespace blade, click the **+ Queue** button.
6. In the **Create queue** pane, perform the following tasks:
   * In the **Name** text box, type **messages**.
   * Leave all remaining settings with their default values.
   * Click the **Create** button.

##### Task 4: Get Service Bus Connection String

1. Back on the Service Bus namespace blade, click **Shared access policies**.
2. On the Service Bus namespace blade, click the **RootManageSharedAccessKey** policy.
3. In the **SAS Policy: RootManageSharedAccessKey** pane, locate and record the value of the **Primary Connection String** field. You will use this value later in this lab.

**Review**: In this exercise, you created a new Service Bus namespace and recorded a connection string to access queues in the namespace.

#### Exercise 2: Create a logic app

##### Task 1: Create an Azure Storage account

1. In the upper left corner of the Azure portal, click **Create a resource**.
2. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Storage Account** and press **Enter**.
3. On the **Everything** blade, in the search results, click **Storage Account - blob, file, table, queue**.
4. On the **Storage Account - blob, file, table, queue** blade, click the **Create** button.
5. On the **Create storage account** blade, perform the following tasks:
   * In the **Name** text box, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **Deployment model** section, ensure that the **Resource manager** option is selected.
   * In the **Account kind** drop-down list, ensure that the **Storage (general purpose v1)** option is selected.
   * Leave the **Location** entry set to the same Azure region you selected earlier in this exercise.
   * In the **Replication** drop-down list, select the **Locally-redundant storage (LRS)** entry.
   * In the **Performance** section, ensure that the **Standard** option is selected.
   * In the **Secure transfer required** section, ensure that the **Disabled** option is selected.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, ensure that the **Use existing** option is selected and, in the drop-down list below, select the resource group you created earlier in this exercise.
   * Leave the **Configure virtual networks** option set to its default value.
   * Leave the **Hierarchical namespaces** option set to its default value.
   * Click the **Create** button.
6. Wait for the provisioning to complete before you proceed to the next step.
7. In the hub menu of the Azure portal, click **Resource groups**.
8. On the **Resource groups** blade, click **AADesignLab1101-RG**.
9. On the **AADesignLab1101-RG** blade, click the newly created Azure Storage account.
10. On the Storage account blade, click the **Blobs** tile.
11. On the Storage account blade, click the **+ Container** button.
12. In the **New container** pane, perform the following tasks:
    * In the **Name** text box, type **messageoutput**.
    * In the **Public access level** drop-down list, select the **Blob (anonymous read access for blobs only)** option.
    * Click the **OK** button.

##### Task 2: Create a logic app

1. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Logic App** and press **Enter**.
2. On the **Everything** blade, in the search results, click **Logic App**.
3. On the **Logic App** blade, click the **Create** button.
4. On the **Create logic app** blade, perform the following tasks:
   * In the **Name** text box, type **ServiceBusWorkflow**.
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1101-RG**.
   * In the **Location** drop-down list, select the same Azure region you chose in the previous task.
   * In the **Log Analytics** section, ensure that the **Off** button is selected.
   * Click the **Create** button.
5. Wait for the provisioning to complete before you proceed to the next task.

##### Task 3: Configure logic app steps.

1. In the hub menu in the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the entry representing the logic app you created in the previous task.
4. On the **Logic Apps Designer** blade, scroll down and click the **Blank Logic App** tile in the **Templates** section.
5. On the **Logic Apps Designer** blade, perform the following tasks:
   * In the **Search connectors and triggers** text box, type **Service Bus**.
   * In the search results, select the trigger named **When a message is received in a queue (auto-complete) - Service Bus**.
   * In the **Connection Name** text box, type **ServiceBusConnection**.
   * In the list of **Service Bus namespaces**, select the namespace you created earlier in this lab.
   * In the list of policies, select the **RootManageSharedAccessKey** policy.
   * Click the **Create** button.
6. In the **When a message is received in a queue (auto-complete)** step, perform the following tasks:
   * In the **Queue name** drop-down list, select the **messages** entry.
   * In the **Interval** text box, type **30**.
   * In the **Frequency** drop-down list, select the **Second** entry.
7. On the **Logic Apps Designer** blade, click the **+ New Step** button.
8. On the **Logic Apps Designer** blade, perform the following tasks:
   * In the **Search connectors and actions** text box, type **Storage blob**.
   * In the search results, select the action named **Create blob - Azure Blob Storage**.
   * In the **Connection Name** text box, type **StorageConnection**.
   * In the list of *Storage accounts*, select the account you created earlier in this lab.
   * Click the **Create** button.
9. In the **Create Blob** step, perform the following tasks:
   * In the **Folder path** text box, type **/messageoutput**.
   * In the **Blob name** text box, type **@concat(triggerBody()?['MessageId'], '.txt')**.
   * In the **Blob content** text box, type **@string(decodeBase64(triggerBody()?['ContentData']))**.
10. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.

##### Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.**Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
2. If this is your first time opening the **Cloud Shell** using your subscription, you will see a wizard to configure **Cloud Shell** for first-time usage. When prompted, in the **Welcome to Azure Cloud Shell** pane, click **Bash (Linux)**.**Note**: If you do not see the configuration options for **Cloud Shell**, this is most likely because you are using an existing subscription with this course's labs. If so, proceed directly to the next task.
3. In the **You have no storage mounted** pane, click **Show advanced settings**, perform the following tasks:
   * Leave the **Subscription** drop-down list entry set to its default value.
   * In the **Cloud Shell region** drop-down list, select the Azure region matching or near the location where you deployed resources in this lab.
   * In the **Resource group** section, select the **Use existing** option and then, in the drop-down list, select **AADesignLab1101-RG**.
   * In the **Storage account** section, ensure that the **Create new** option is selected and then, in the text box below, type a unique name consisting of a combination of between 3 and 24 characters and digits.
   * In the **File share** section, ensure that the **Create new** option is selected and then, in the text box below, type **cloudshell**.
   * Click the **Create storage** button.
4. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

##### Task 4: Validate Logic App using Node.js

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. At the **Cloud Shell** command prompt at the bottom of the portal, type in the following command and press **Enter** to install the **azure** package using NPM:

npm install azure

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to open the interactive node terminal:

node

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to import the **azure** module in Node:

var azure = require('azure');

1. At the **Cloud Shell** command prompt, type in the following command (replacing the placeholder <Service Bus namespace connection string> with the value of your url you recorded earlier in this lab) and press **Enter** to create a new variable for your Service Bus namespace connection string:

var connectionString = '<Service Bus namespace connection string>';

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to create a new client to connect to the Service Bus namespace:

var serviceBusService = azure.createServiceBusService(connectionString);

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to send a message to Service Bus namespace queue using the client.

serviceBusService.sendQueueMessage('messages', { body: 'Hello World' }, function(error) { console.log(error) });

1. In the hub menu of the Azure portal, click **Resource groups**.
2. On the **Resource groups** blade, click **AADesignLab1101-RG**.
3. On the **AADesignLab1101-RG** blade, click the Azure Storage account you created earlier in this lab.
4. On the Storage account blade, click the **Blobs** tile.
5. On the Storage account container blade, click the **messageoutput** container.
6. Note the newly created blob in your container.

**Review**: In this exercise, you created a logic app that is triggered by messages from a queue in a Service Bus namespace.

#### Exercise 3: Remove lab resources

##### Task 1: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open the Cloud Shell pane.
2. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to list all resource groups you created in this lab:

az group list --query "[?starts\_with(name,'AADesignLab11')]".name --output tsv

1. Verify that the output contains only the resource groups you created in this lab. These groups will be deleted in the next task.

##### Task 2: Delete resource groups

1. At the **Cloud Shell** command prompt, type in the following command and press **Enter** to delete the resource groups you created in this lab

az group list --query "[?starts\_with(name,'AADesignLab11')]".name --output tsv | xargs -L1 bash -c 'az group delete --name $0 --no-wait --yes'

1. Close the **Cloud Shell** prompt at the bottom of the portal.

**Review**: In this exercise, you removed the resources used in this lab.

### Module 4 Review Questions

**Azure Service Bus**  
You are designing a mobile solution for your company. Users search for products and create an order by adding the product to a virtual shopping cart.  
You need to recommend a messaging infrastructure to reliably store the orders as messages while the orders are waiting to be processed.  
What Azure service should you recommend?

Suggested Answer ↓

Microsoft Azure Service Bus is a host for queues holding jobs of critical business value, such as ordering products online. It allows for the creation of routes for messages that need to travel between applications and application modules. It is a solid platform for workflow and transaction handling and has robust facilities for dealing with many application fault conditions. A strength of Service Bus is also its function as a bridge between elements of hybrid cloud solutions and systems that include branch-office or work-site systems. Systems that sit “behind the firewall”, are roaming across networks, or are occasionally offline can’t be reached directly via “push” messaging but require messages to be sent to an agreed pickup location from where the designated receiver can obtain them.

**Azure Logic Apps**  
You are designing a solution that will integrate several on-premises and Azure-based services. The solution will include workflows to perform operations in a specific order.  
What Azure service should you recommend?

Suggested Answer ↓

Azure Logic Apps is a serverless solution that allows you to build automated scalable workflows. The workflows can integrate apps and data across cloud services and on-premises systems. Azure Logic Apps include several, built-in connectors to help your apps communicate with other apps and services, control the workflow through your logic apps, and manage or manipulate data. Organizations can also exchange messages through industry-standard protocols, including AS2, X12, and EDIFACT, using Azure Logic Apps enterprise integration features.

**Azure Event Hub**  
Your company has manufacturing facilities worldwide. Each facility has several machines that produce products. The machines generate millions of messages daily to report progress, quality control metrics, and alerts.You need to design a solution to receive and process the messages from the machines.  
What Azure service should you include in the design?

Suggested Answer ↓

Azure Event Hubs is a highly scalable data streaming platform and ingestion service capable of receiving and processing millions of events per second. Event Hubs can process and store events, data, or telemetry produced by distributed software and devices. Data sent to an event hub can be transformed and stored using any real-time analytics provider or batching/storage adapters. Event Hubs sits between event publishers (the manufacturing machinery) and event consumers to decouple the production of an event stream from the consumption of those events. Event Hubs