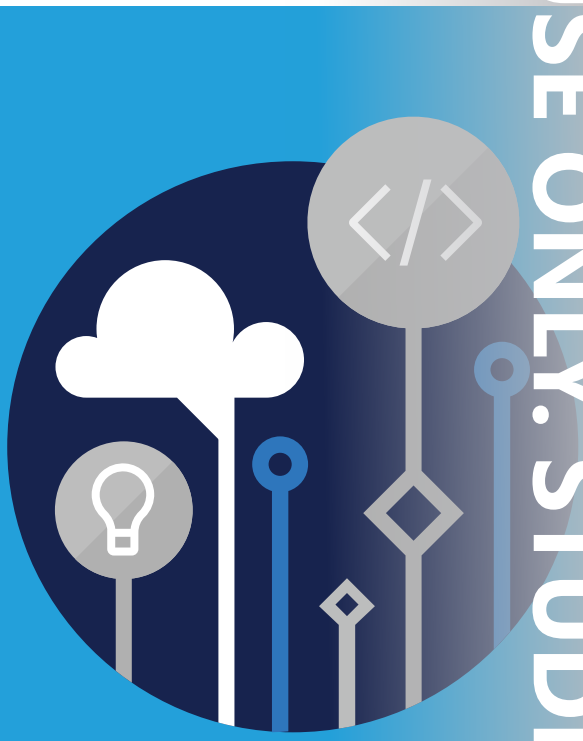
Microsoft

Microsoft Official Course



AZ-301T01

Designing for Identity and Security

AZ-301T01
Designing for Identity and Security

Contents

Module 0 Welcome

Start Here

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.

Module 1 Module Managing Security & Identity for Azure Solutions

Security

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 consum-

ing. 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/**

Identity

Azure Active Directory

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

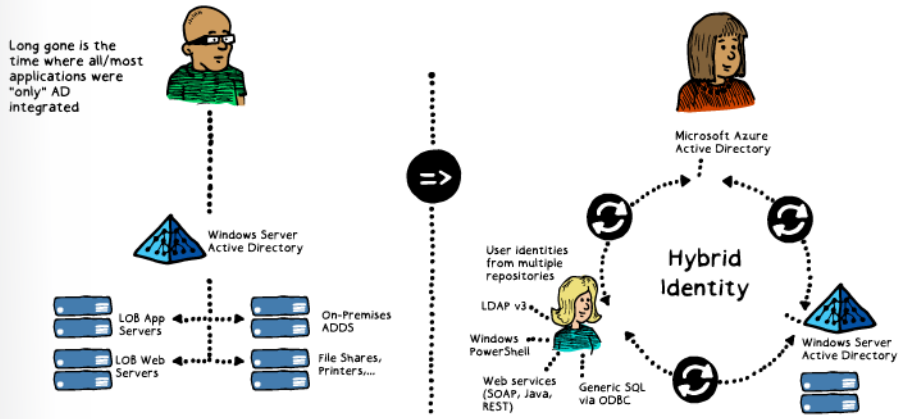
After completing this lesson, you will be able to:

- 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:

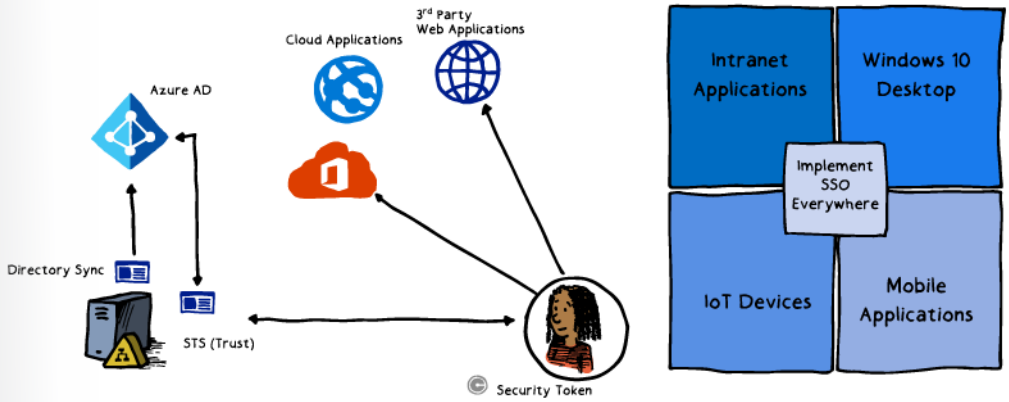
Employees with work or school accounts, partners with work or school accounts, or any email address. Soon to support direct federation.

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

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.

Online Lab - Securing Secrets in Azure

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**[1]
   - **Microsoft Azure Storage Explorer**[2]
   - Bash on Ubuntu on Windows
   - Windows PowerShell
3. **Note**: You can also find shortcuts to these applications in the **Start Menu**.

3

Exercise 1: Deploy Key Vault resources

4

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**).
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.

1 https://code.visualstudio.com/
2 https://azure.microsoft.com/features/storage-explorer/
3 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#exercise-1-deploy-key-vault-resources
4 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-1-open-the-azure-portal

5

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.

6

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.

5 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-2-deploy-a-key-vault

6 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-3-add-a-secret-to-a-key-vault-by-using-the-azure-portal

7

Task 4: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. **Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
3. 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)**.
4. **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.
5. 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.
6. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

8

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'
```

2. 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:

7 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-4-open-cloud-shell

8 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-5-add-a-secret-to-a-key-vault-using-the-cli

```
KEY_VAULT_NAME=$(az keyvault list --resource-group $RESOURCE_GROUP --query
"[0].name" --output tsv)
```

3. 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
```

4. 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
```

5. 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
```

6. 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
```

7. Close the **Cloud Shell** pane.

9

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:

9 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-6-add-secrets-to-a-key-vault-by-using-azure-resource-manager-templates

```
{
    "$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'), '/', varia-
bles('secretName'))]",
            "properties": {
                "contentType": "text/plain",
                "value": "StudentPa$$w.rd"
            }
        }
    ]
}
```

8. Click the **Save** button to persist the template.
9. 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.
10. Do not wait for the deployment to complete but proceed to the next step.
11. In the upper left corner of the Azure portal, click **Create a resource**.
12. At the top of the **New** blade, in the **Search the Marketplace** text box, type **Template Deployment** and press **Enter**.
13. On the **Everything** blade, in the search results, click **Template Deployment**.
14. On the **Template deployment** blade, click the **Create** button.
15. On the **Custom deployment** blade, click the **Build your own template in the editor** link.
16. On the **Edit template** blade, click **Load file**.

17. 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'), '/', varia-
bles('secretName'))]",
            "dependsOn": [
                "[resourceId('Microsoft.Storage/storageAccounts', varia-
bles('storageName'))]"
            ],
            "properties": {
                "contentType": "text/plain",
                "value": "[concat('DefaultEndpointsProtocol=https;Account-
Name=', variables('storageName'), ';', 'AccountKey=', listKeys(re-
sourceId('Microsoft.Storage/storageAccounts', variables('storageName')),
providers('Microsoft.Storage', 'storageAccounts').apiVersions[0]).keys[0].
value, ';')]"
            }
        }
    ]
}
```

18. Click the **Save** button to persist the template.
19. 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.
20. Wait for the deployment to complete before you proceed to the next task.

10

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.

10 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-7-view-key-vault-secrets

11

Exercise 2: Deploy Azure VM using Key Vault secret

12

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'
```

3. 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)
```

4. 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')"
```

13

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**.

11 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#exercise-2-deploy-azure-vm-using-key-vault-secret

12 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-1-retrive-the-value-of-the-key-vault-resource-id-parameter

13 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-2-prepare-the-azure-resource-manager-deployment-template-and-parameters-files

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
```

6. 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
```

14

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.

15

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
```

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

14 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-3-configure-a-key-vault-for-deployment-of-azure-resource-manager-templates

15 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-4-deploy-a-linux-vm-with-the-password-paramter-set-by-using-a-key-vault-secret

16

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'
```

2. 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)
```

3. 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
```

4. 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)
```

5. 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
```

6. At the **Cloud Shell** command prompt, when prompted whether you want to continue connecting, type `yes` and press **Enter**.
7. 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**.
8. Verify that you successfully authenticated.
9. 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.

16 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-5-verify-the-outcome-of-the-deployment

17

Exercise 3: Remove lab resources

18

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
```

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

19

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'
```

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

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

17 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#exercise-3-remove-lab-resources

18 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-1-open-cloud-shell

19 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod01_Securing%20Secrets%20in%20Azure.md#task-2-delete-resource-groups

Review Questions

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-premis-

es resources. Azure AD Connect includes components to monitor the synchronization health and federation of identities.

Module 2 Module Integrating SaaS Services Available on the Azure Platform

Bot Services

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:

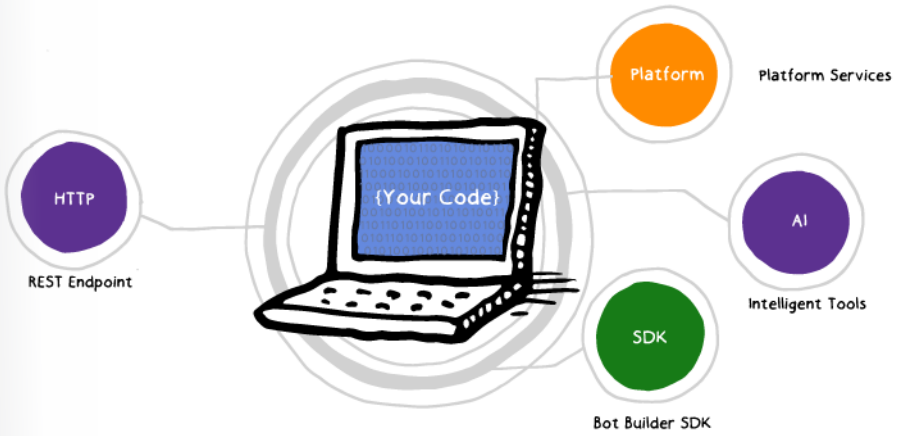
• Describe the relationship to Bot Framework and Azure Bot Services.

• 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 Skype, Facebook, Teams, Slack, SMS, and several others.

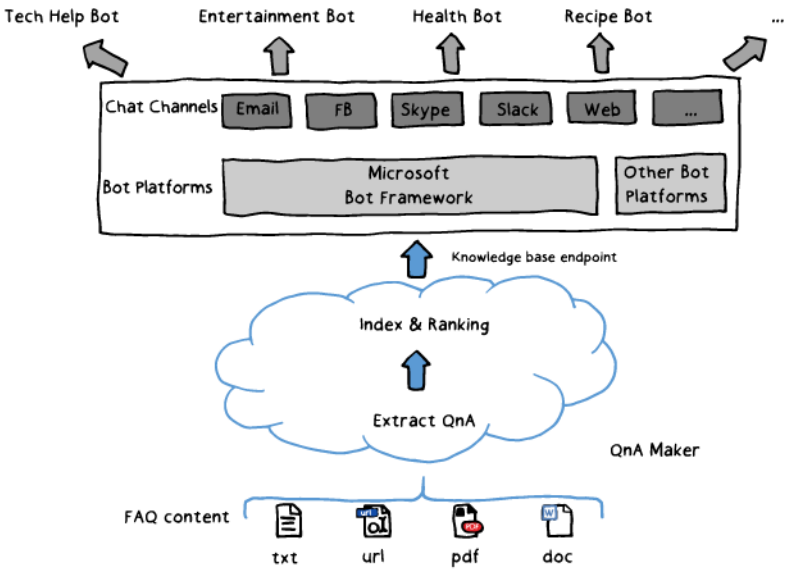
• Tools and services: Test your bot with the Bot Framework Emulator and preview your bot on different channels with the Channel Inspector.

• 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.



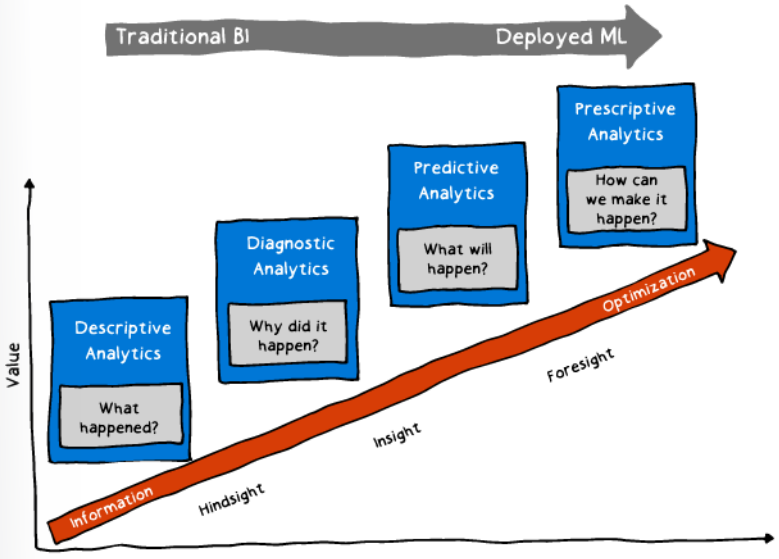
Machine Learning

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.



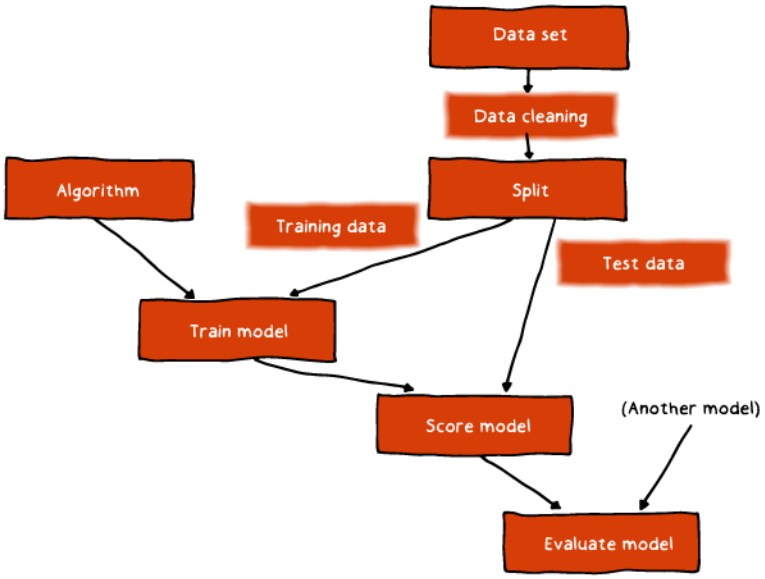
Azure Machine Learning

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 Processing

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.

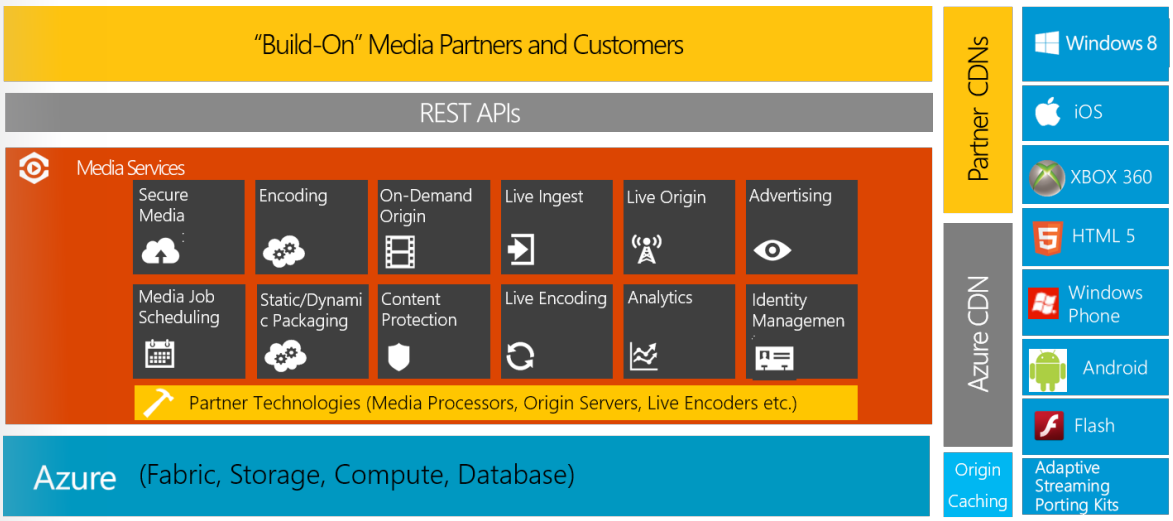
After completing this section you will be able to:

- 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.



Computer Vision API

The cloud-based Computer Vision API provides developers with access to advanced algorithms for processing images and returning information. By uploading an image or specifying an image URL, Microsoft Computer Vision algorithms can analyze visual content in different ways based on inputs and user choices. With the Computer Vision API users can analyze images to:

- Tag images based on content.
- Categorize images.
- Identify the type and quality of images.

- Detect human faces and return their coordinates.
- Recognize domain-specific content.
- Generate descriptions of the content.
- Use optical character recognition to identify printed text found in images.
- Recognize handwritten text.
- Distinguish color schemes.
- Flag adult content.
- 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

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 LUIS, Face API and Speech API in detailed examples.

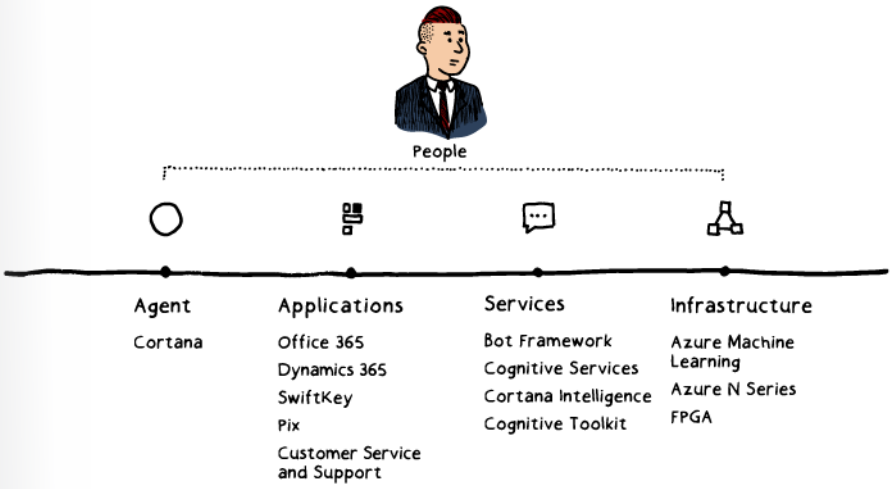
After completing this section you will be able to:

• 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.

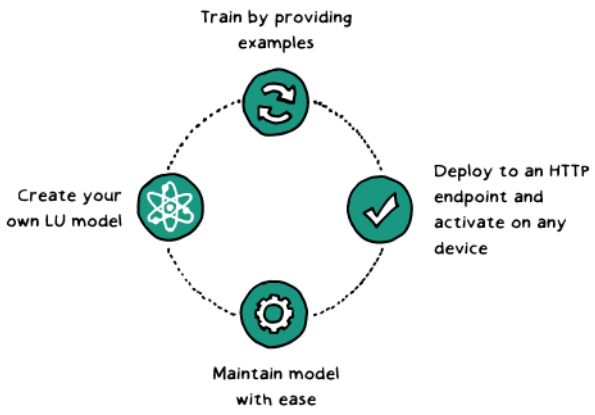
• **Bing Autosuggest:** Bing Autosuggest API lets you send a partial search query term to Bing and get back a list of suggested queries that other users have searched on. For example, as the user enters each character of their search term, you'd call this API and populate the search box's drop-down list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

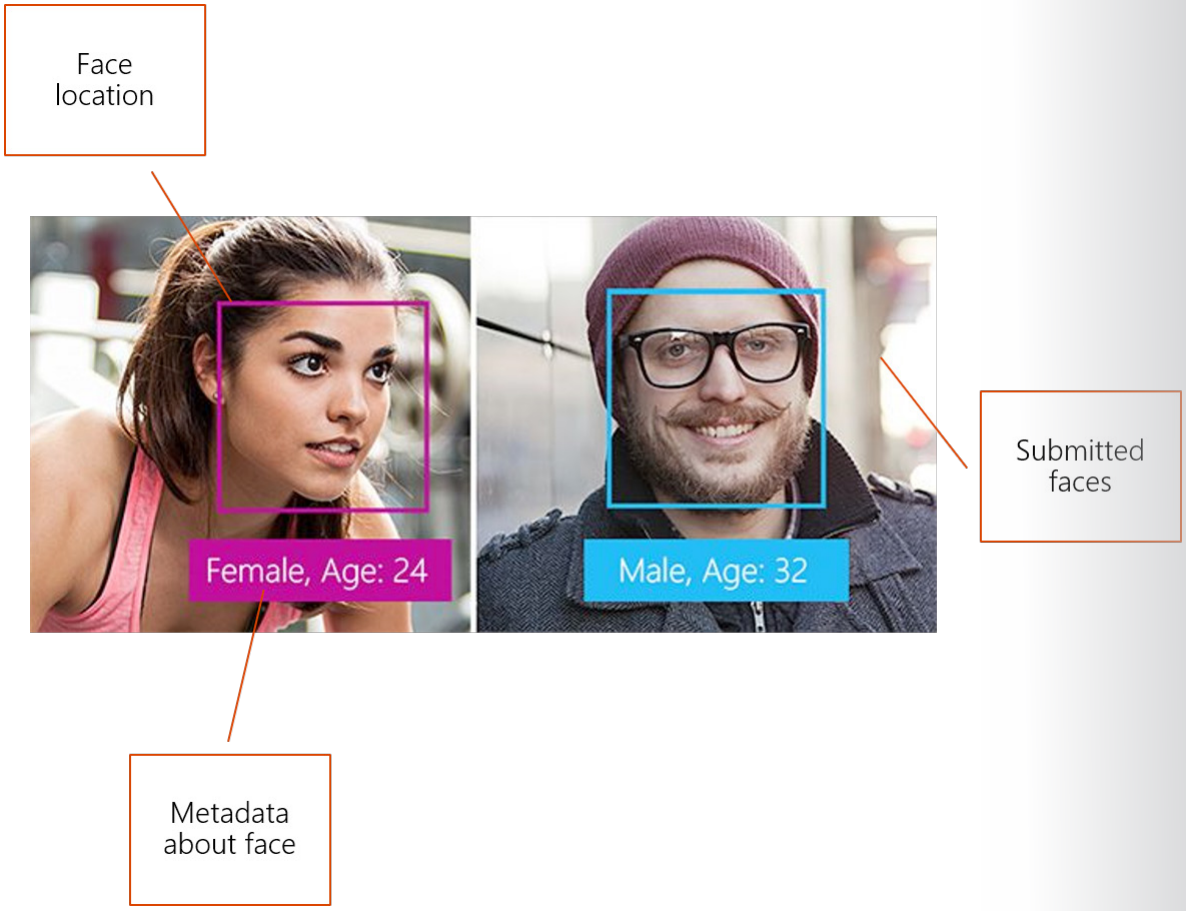
Speaker Recognition API is a cloud-based APIs that provide the most advanced algorithms for speaker verification and speaker identification.

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.



Online Lab - Integrating SaaS Services Available on the Azure Platform

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.**[1]

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**[2]
   - **Microsoft Azure Storage Explorer**[3]
   - Bash on Ubuntu on Windows
   - Windows PowerShell
3. **Note**: You can also find shortcuts to these applications in the **Start Menu**.

1 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign
2 https://code.visualstudio.com/
3 https://azure.microsoft.com/features/storage-explorer/

4

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

5

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**).
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.

6

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": [
```

4 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#exercise-1-deploy-function-app-and-cognitive-service-using-arm-template

5 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-1-open-the-azure-portal

6 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-2-deploy-cognitive-service-using-an-azure-resource-manager-template

```
        {
            "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]"
        }
    }
}
```

8. Click the **Save** button to persist the template.
9. 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.
10. Wait for the deployment to complete before you proceed to the next step.
11. In the hub menu of the Azure portal, click **Resource groups**.
12. On the **Resource groups** blade, click **AADesignLab1001-RG**.
13. 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.
14. On the deployments blade, click the name of the most recent deployment.
15. On the **Microsoft.Template- Overview** blade, click **Outputs**.
16. 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.

7

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.

7 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-3-deploy-a-function-app

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**.
    - 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.
21. **Note**: You will be able to determine that the first deployment has completed by monitoring the **Deployments** tab. This tab updates automatically.

[8]

Task 4: Test a function app using Cognitive Services

1. Back on the function app blade, click **Functions** to expand the list of functions.
2. **Note**: You may need to click **Functions** twice to refresh the list of functions.
3. Select the **DetermineLanguage** function from the list of functions.
4. In the **run.csx** pane that opens, click **Test** on the right side of the pane.
5. 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.

8 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-4-test-a-function-app-using-cognitive-services

9

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

10

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.

11

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.

---

9 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#exercise-2-create-a-logic-app-that-uses-a-function-app

10 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-1-create-a-logic-app

11 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-2-configure-logic-app-steps

6. On the **Logic Apps Designer** blade, review the blank Logic App JSON template:

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

7. 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/Micro-
soft.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"
            }
        }
    }
}
```

8. On the **Logic Apps Designer** blade, click the **Designer** button.
9. **Note**: At this point, you should see a single step in the designer. This is the "trigger" step that begins a workflow.
10. Click the **+ New Step** button in the designer.

11. 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.
12. 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.
13. Click the **+ New Step** button in the designer. Click the **Add an action** button to open the dialog for creating an action.
14. 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.
15. 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.
16. Click the **+ New Step** button in the designer.
17. 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**.
18. 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')**.
19. At the top of the **Logic Apps Designer** blade, click the **Save** button to persist your workflow.
20. Scroll to the top of the **Logic Apps Designer** area and click the **When a HTTP request is received** step.
21. Copy the value of the **HTTP POST URL** text box. This URL will be used later in this lab.

12

Task 2: Open Cloud Shell

1. At the top of the portal, click the **Cloud Shell** icon to open a new shell instance.
2. **Note**: The **Cloud Shell** icon is a symbol that is constructed of the combination of the *greater than* and *underscore* characters.
3. 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)**.
4. **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.
5. 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.
6. Wait for the **Cloud Shell** to finish its first-time setup procedures before you proceed to the next task.

13

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
   ```

2. 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
   ```

12 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-2-open-cloud-shell

13 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-3-validate-logic-app-using-python

3. 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>"
```

4. 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.'})
```

5. 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)
```

6. 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.

14

Exercise 3: Remove lab resources

15

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
```

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

14 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#exercise-3-remove-lab-resources

15 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-1-open-cloud-shell

16

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'
```

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

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

16 https://github.com/MicrosoftLearning/AZ-301-MicrosoftAzureArchitectDesign/blob/master/Instructions/AZ-301T01_Lab_Mod02_Deploying%20Service%20Instances%20as%20Components%20of%20Overall%20Azure%20Solutions.md#task-2-delete-resource-groups

Review Questions

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.