Microsoft Certified:
Azure Administrator Associate

Microsoft Certified: Azure Administrator Associate..... 1
AZ-104: Prerequisites for Azure administrators..... 2
Configure Azure resources with tools..... 2
Use Azure Resource Manager..... 3
Configure resources with Azure Resource Manager templates..... 7
Automate Azure tasks using scripts with PowerShell..... 10
Control Azure services with the CLI..... 12
Deploy Azure infrastructure by using JSON ARM templates..... 13
AZ-104: Manage identities and governance in Azure..... 15
Configure Azure Active Directory..... 15
Configure user and group accounts..... 15
Configure subscriptions..... 16
Configure Azure Policy..... 16
Configure role-based access control..... 16
Create Azure users and groups in Azure Active Directory..... 16
Secure your Azure resources with Azure role-based access control (Azure RBAC)..... 16
Allow users to reset their password with Azure Active Directory self-service password reset.. 16

AZ-104: Prerequisites for Azure administrators

Configure Azure resources with tools

Azure Administrators use tools to interact with the cloud environment and complete such tasks as:

- Deploying dozens or hundreds of resources at a time.
- Configuring individual services using scripts.
- Viewing rich reports across usage, health, costs, and more.

You must select and use a tooling option. Your choices can include the Azure portal, Azure PowerShell, Azure CLI, or Azure Cloud Shell.

Azure Cloud Shell is an interactive, browser-accessible shell for managing Azure resources. Linux users can opt for a Bash experience, while Windows users can opt for PowerShell. Features:

- Times out after 20 minutes without interactive activity.
- Requires a resource group, storage account, and Azure File share.
- Uses the same Azure file share for both Bash and PowerShell.
- Is assigned to one machine per user account.
- Persists $HOME using a 5-GB image held in your file share.
- Permissions are set as a regular Linux user in Bash.

Azure PowerShell is a module that you add to Windows PowerShell or PowerShell Core to enable you to connect to your Azure subscription and manage resources. Azure PowerShell requires PowerShell to function. PowerShell provides services such as the shell window and command parsing. Azure PowerShell adds the Azure-specific commands.

```
New-AzVm `
    -ResourceGroupName "CrmTestingResourceGroup" `
    -Name "CrmUnitTests" `
    -Image "UbuntuLTS"
    ...
```

Azure PowerShell is also available two ways: inside a browser via the Azure Cloud Shell, or with a local installation on Linux, macOS, or the Windows operating system. In both cases, you have two modes from which to choose: you can use it in interactive mode in which you manually issue one command at a time, or in scripting mode where you execute a script that consists of multiple commands.

What is the Az module? Az is the formal name for the Azure PowerShell module containing cmdlets to work with Azure features.

Azure CLI is a command-line program to connect to Azure and execute administrative commands on Azure resources. It runs on Linux, macOS, and Windows, and allows

administrators and developers to execute their commands through a terminal, command-line prompt, or script instead of a web browser. For example, to restart a VM, you would use a command such as the following:

```
az vm restart -g MyResourceGroup -n MyVm
```

You can also use Azure CLI from a browser through Azure Cloud Shell. In both cases, Azure CLI can be used interactively or through scripts.

Commands in the CLI are structured in groups and subgroups. Each group represents a service provided by Azure, and the subgroups divide commands for these services into logical groupings. For example, the storage group contains subgroups including account, blob, share, and queue.

So, how do you find the particular commands you need? One way is to use az find. For example, if you want to find commands that might help you manage a storage blob, you can use the find command:

```
az find blob
```

If you already know the name of the command you want, the --help argument for that command will get you more detailed information on the command, and for a command group, a list of the available subcommands.

Use Azure Resource Manager

Your company is beginning to create resources in Azure. There is no organizational plan for standardizing the effort. There have been several instances where critical resources were inadvertently deleted. It is difficult to determine who owns which resource.

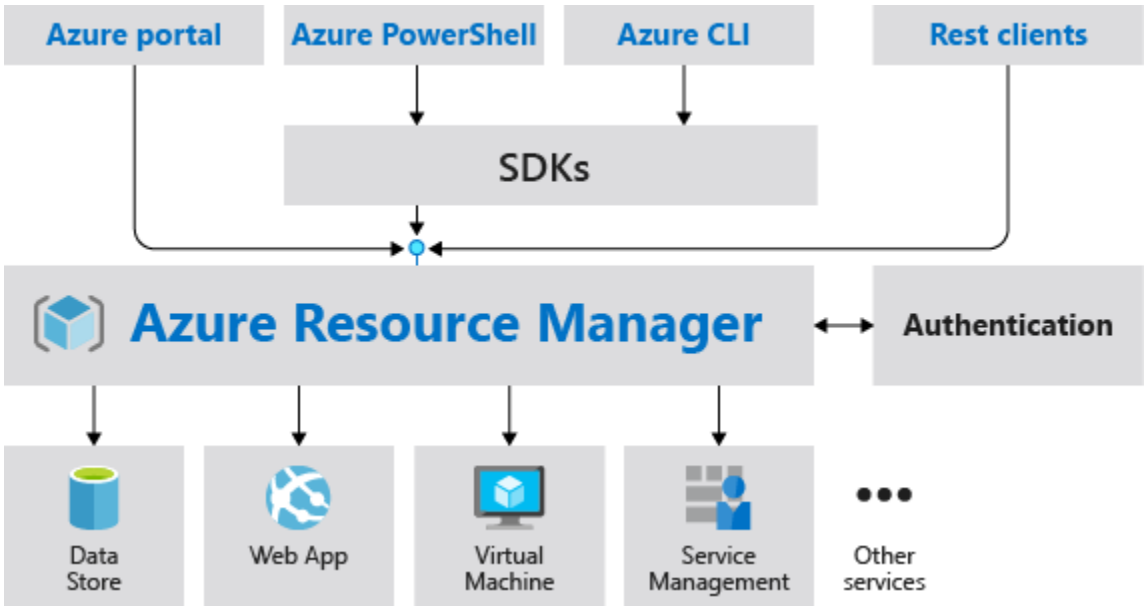
You need to use resource groups to organize the company's Azure resources.

The infrastructure for your application is typically made up of many components – maybe a virtual machine, storage account, and virtual network, or a web app, database, database server, and third-party services. These components are not separate entities, instead they are related and interdependent parts of a single entity. You want to deploy, manage, and monitor them as a group.

You use a template for deployment and that template can work for different environments such as testing, staging, and production. Azure Resource Manager provides security, auditing, and tagging features to help you manage your resources after deployment.

The following image shows how all the tools interact with the same Azure Resource Manager API. The API passes requests to the Azure Resource Manager service, which authenticates and

authorizes the requests. Azure Resource Manager then routes the requests to the appropriate resource providers.



Azure Resource Manager provides several benefits:

- You can deploy, manage, and monitor all the resources for your solution as a group, rather than handling these resources individually.
- You can repeatedly deploy your solution throughout the development lifecycle and have confidence your resources are deployed in a consistent state.
- You can manage your infrastructure through declarative templates rather than scripts.
- You can define the dependencies between resources so they're deployed in the correct order.
- You can apply access control to all services in your resource group because Role-Based Access Control (RBAC) is natively integrated into the management platform.
- You can apply tags to resources to logically organize all the resources in your subscription.
- You can clarify your organization's billing by viewing costs for a group of resources sharing the same tag.

The following suggestions help you take full advantage of Azure Resource Manager when working with your solutions.

- Define and deploy your infrastructure through the declarative syntax in Azure Resource Manager templates, rather than through imperative commands.
- Define all deployment and configuration steps in the template. You should have no manual steps for setting up your solution.
- Run imperative commands to manage your resources, such as to start or stop an app or machine.

- Arrange resources with the same lifecycle in a resource group. Use tags for all other organizing of resources.

resource provider - A service that supplies the resources you can deploy and manage through Resource Manager. Each resource provider offers operations for working with the resources that are deployed. Some common resource providers are Microsoft.Compute, which supplies the virtual machine resource, Microsoft.Storage, which supplies the storage account resource, and Microsoft.Web, which supplies resources related to web apps. The name of a resource type is in the format: {resource-provider}/{resource-type}. For example, the key vault type is Microsoft.KeyVault/vaults.

template - A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group. It also defines the dependencies between the deployed resources. The template can be used to deploy the resources consistently and repeatedly.

declarative syntax - Syntax that lets you state "Here is what I intend to create" without having to write the sequence of programming commands to create it. The Resource Manager template is an example of declarative syntax.

There are a few rules for resource groups.

- Resources can only exist in one resource group.
- Resource Groups cannot be renamed.
- Resource Groups can have resources of many different types (services).
- Resource Groups can have resources from many different regions.
- All the resources in your group should share the same lifecycle. You deploy, update, and delete them together.
- You can move a resource from one resource group to another group. Limitations do apply to moving resources
- A resource can interact with resources in other resource groups; when the two resources are related but don't share the same lifecycle

Resource Manager locks allow organizations to put a structure in place that prevents the accidental deletion of resources in Azure.

- You can associate the lock with a subscription, resource group, or resource.
- Locks are inherited by child resources.

There are two types of resource locks.

- Read-Only locks, which prevent any changes to the resource.
- Delete locks, which prevent deletion.

Sometimes you may need to move resources to either a new subscription or a new resource group in the same subscription. When moving resources, both the source group and the target group are locked during the operation. Write and delete operations are blocked on the resource groups until the move completes. Locks don't mean the resources aren't available.

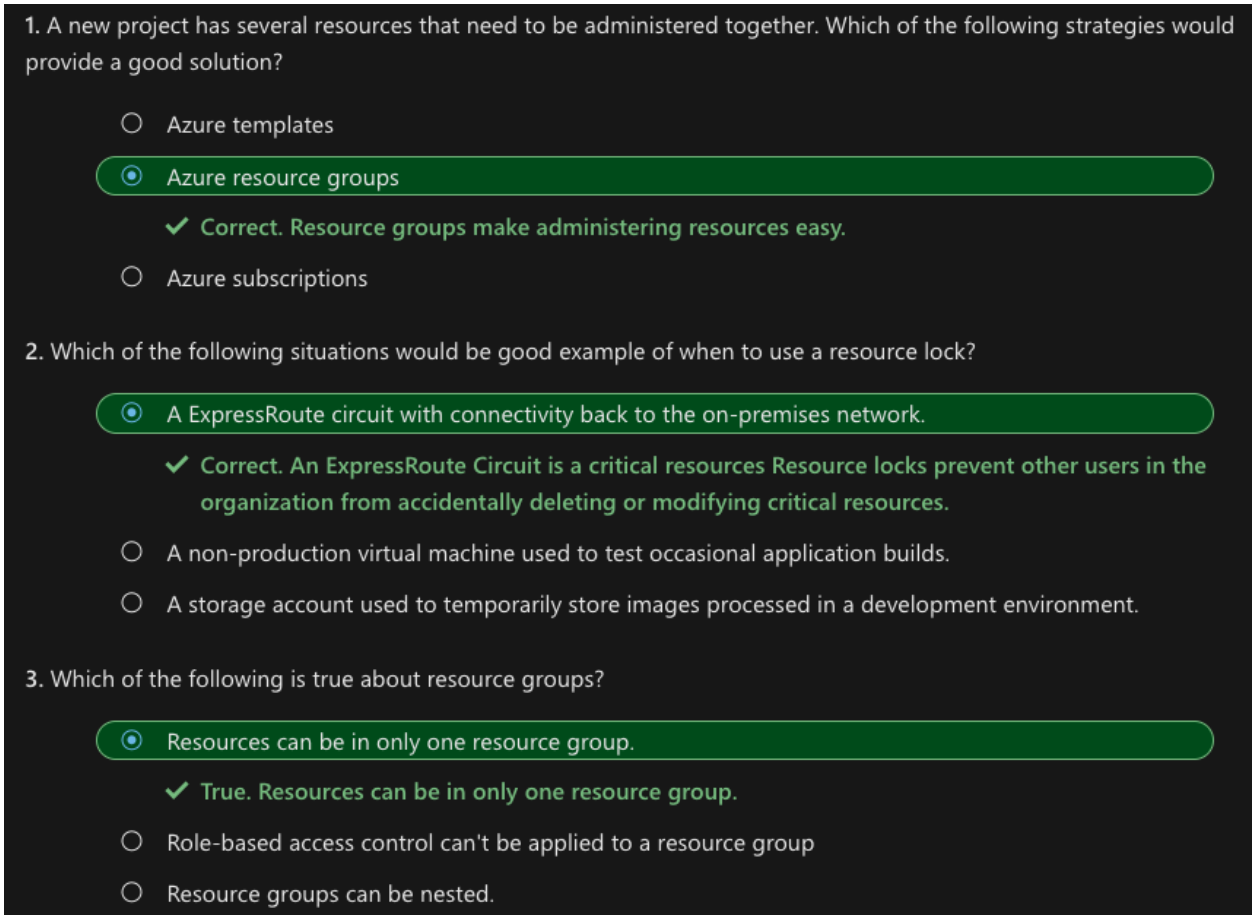
Use caution when deleting a resource group. Deleting a resource group deletes all the resources contained within it. That resource group might contain resources that resources in other resource groups depend on.

Using PowerShell to delete resource groups:

Remove-AzResourceGroup -Name "ContosoRG01"

Azure lets you view resource usage against limits. This is helpful to track current usage, and plan for future use.

- The limits shown are the limits for your subscription.
- When you need to increase a default limit, there is a Request Increase link.
- All resources have a maximum limit listed in Azure limits.
- If you are at the maximum limit, the limit can't be increased.



Configure resources with Azure Resource Manager templates

Your company needs to ensure virtual machine deployments are consistent across the organization. You use Azure Resource Manager templates to deploy resources including virtual machines.

An Azure Resource Manager template precisely defines all the Resource Manager resources in a deployment. You can deploy a Resource Manager template into a resource group as a single operation.

Template benefits

- Templates improve consistency. Resource Manager templates provide a common language for you and others to describe your deployments. Regardless of the tool or SDK that you use to deploy the template, the structure, format, and expressions inside the template remain the same.
- Templates help express complex deployments. Templates enable you to deploy multiple resources in the correct order. For example, you wouldn't want to deploy a virtual machine prior to creating an operating system (OS) disk or network interface. Resource Manager maps out each resource and its dependent resources, and creates dependent resources first. Dependency mapping helps ensure that the deployment is carried out in the correct order.
- Templates reduce manual, error-prone tasks. Manually creating and connecting resources can be time consuming, and it's easy to make mistakes. Resource Manager ensures that the deployment happens the same way every time.
- Templates are code. Templates express your requirements through code. Think of a template as a type of Infrastructure as Code that can be shared, tested, and versioned similar to any other piece of software. Also, because templates are code, you can create a "paper trail" that you can follow. The template code documents the deployment. Most users maintain their templates under some kind of revision control, such as GIT. When you change the template, its revision history also documents how the template (and your deployment) has evolved over time.
- Templates promote reuse. Your template can contain parameters that are filled in when the template runs. A parameter can define a username or password, a domain name, and so on. Template parameters enable you to create multiple versions of your infrastructure, such as staging and production, while still using the exact same template.
- Templates are linkable. You can link Resource Manager templates together to make the templates themselves modular. You can write small templates that each define a piece of a solution, and then combine them to create a complete system.
- Templates simplify orchestration. You only need to deploy the template to deploy all of your resources. Normally this would take multiple operations.

A Resource Manager template can contain sections that are expressed using JSON notation, but aren't related to the JSON language itself:

```
{
                                                    "$schema":
"http://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
   "contentVersion": "",
```

```
    "parameters": {},
    "variables": {},
    "functions": [],
    "resources": [],
    "outputs": {}
}
```

Element name	Required	Description
$schema	Yes	Location of the JSON schema file that describes the version of the template language. Use the URL shown in the preceding example.
contentVersion	Yes	Version of the template (such as 1.0.0.0). You can provide any value for this element. Use this value to document significant changes in your template. This value can be used to make sure that the right template is being used.
parameters	No	Values that are provided when deployment is executed to customize resource deployment.
variables	No	Values that are used as JSON fragments in the template to simplify template language expressions.
functions	No	User-defined functions that are available within the template.
resources	Yes	Resource types that are deployed or updated in a resource group.
outputs	No	Values that are returned after deployment.

In the parameters section of the template, you specify which values you can input when deploying the resources. The available properties for a parameter are:

```
"parameters": {
    "<parameter-name>" : {
        "type" : "<type-of-parameter-value>",
        "defaultValue": "<default-value-of-parameter>",
        "allowedValues": [ "<array-of-allowed-values>" ],
        "minValue": <minimum-value-for-int>,
        "maxValue": <maximum-value-for-int>,
```

```
      "minLength": <minimum-length-for-string-or-array>,
      "maxLength": <maximum-length-for-string-or-array-parameters>,
      "metadata": {
      "description": "<description-of-the parameter>"
      }
    }
}
```

Azure Bicep is a domain-specific language (DSL) that uses declarative syntax to deploy Azure resources. It provides concise syntax, reliable type safety, and support for code reuse.

You can use Bicep instead of JSON to develop your Azure Resource Manager templates (ARM templates). The JSON syntax to create an ARM template can be verbose and require complicated expressions. Bicep syntax reduces that complexity and improves the development experience. Bicep is a transparent abstraction over ARM template JSON and doesn't lose any of the JSON template capabilities.

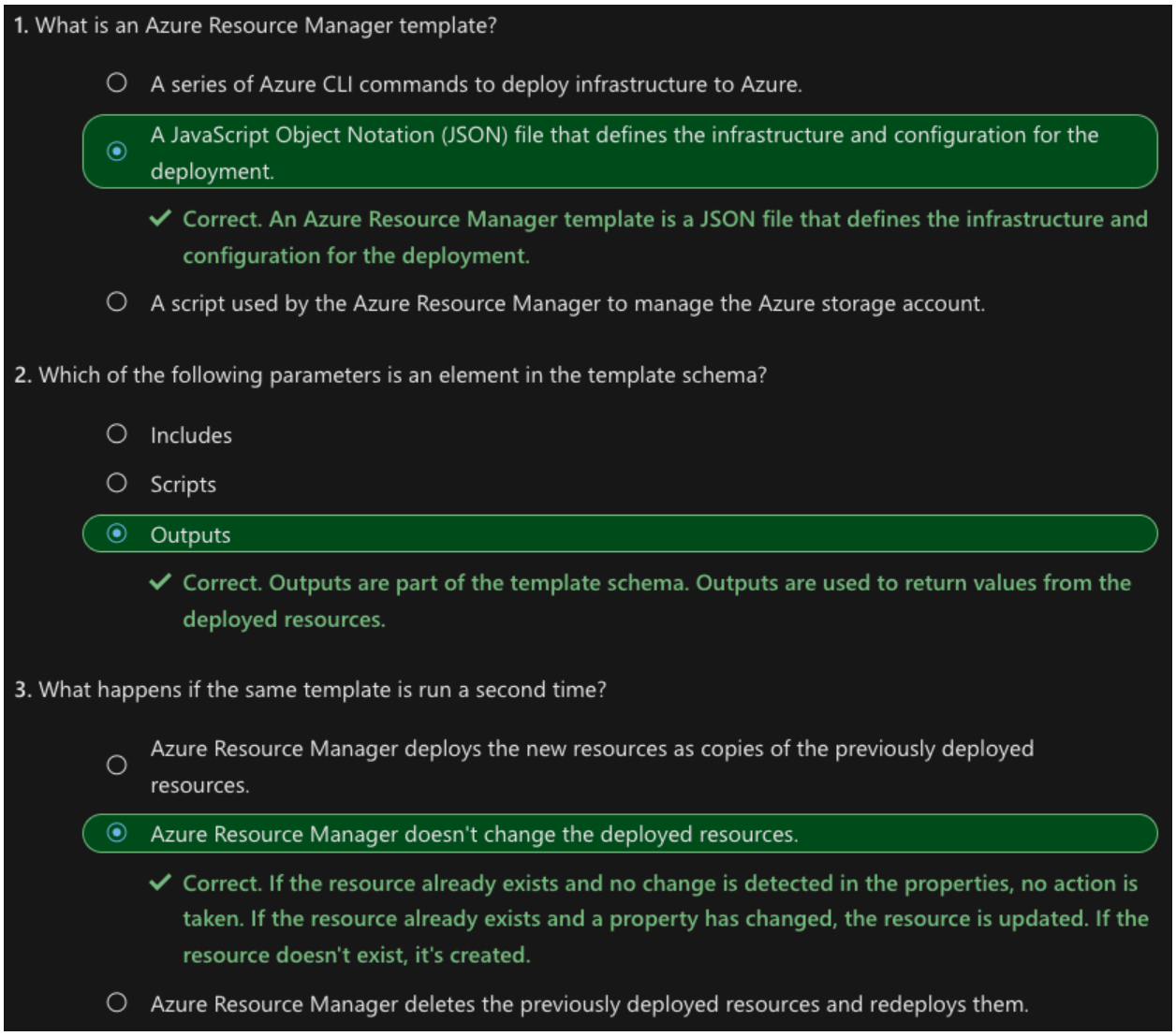
How does Bicep work? When you deploy a resource or series of resources to Azure, the tooling that's built into Bicep converts your Bicep template into a JSON template. This process is known as transpilation. Transpilation is the process of converting source code written in one language into another language.

Bicep provides many improvements over JSON for template authoring, including:

- Simpler syntax: Bicep provides a simpler syntax for writing templates. You can reference parameters and variables directly, without using complicated functions. String interpolation is used in place of concatenation to combine values for names and other items. You can reference the properties of a resource directly by using its symbolic name instead of complex reference statements. These syntax improvements help both with authoring and reading Bicep templates.
- Modules: You can break down complex template deployments into smaller module files and reference them in a main template. These modules provide easier management and greater reusability.
- Automatic dependency management: In most situations, Bicep automatically detects dependencies between your resources. This process removes some of the work involved in template authoring.
- Type validation and IntelliSense: The Bicep extension for Visual Studio Code features rich validation and IntelliSense for all Azure resource type API definitions. This feature helps provide an easier authoring experience.

Azure Quickstart Templates are Azure Resource Manager templates provided by the Azure community.

- The README.md file provides an overview of what the template does.
- The azuredeploy.json file defines the resources that will be deployed.
- The azuredeploy.parameters.json file provides the values the template needs.



Automate Azure tasks using scripts with PowerShell

How to Choose an administrative tool

Automation: Do you need to automate a set of complex or repetitive tasks? Azure PowerShell and the Azure CLI support automation, while Azure portal doesn't.

Learning curve: Do you need to complete a task quickly without learning new commands or syntax? The Azure portal doesn't require you to learn syntax or memorize commands. In Azure PowerShell and the Azure CLI, you must know the detailed syntax for each command you use.

Team skillset: Does your team have existing expertise? For example, your team may have used PowerShell to administer Windows. If so, they'll quickly become comfortable using Azure PowerShell.

Let's look at the two components that make up Azure PowerShell:

- The base PowerShell product This comes in two variants: Windows PowerShell and PowerShell 7.x, which can be installed on Windows, macOS, and Linux.
- The Azure Az PowerShell module This extra module must be installed to add the Azure-specific commands to PowerShell.

The base PowerShell product ships with cmdlets that work with features such as sessions and background jobs. You can add modules to your PowerShell installation to get cmdlets that manipulate other features. For example, there are third-party modules to work with ftp, administer your operating system, access the file system, and so on.

Cmdlets follow a verb-noun naming convention; for example, Get-Process, Format-Table, and Start-Service. There's also a convention for verb choice: "get" to retrieve data, "set" to insert or update data, "format" to format data, "out" to direct output to a destination, and so on.

Cmdlets are shipped in modules. A PowerShell Module is a DLL that includes the code to process each available cmdlet. You load cmdlets into PowerShell by loading the module in which they're contained. You can get a list of loaded modules using the Get-Module command.

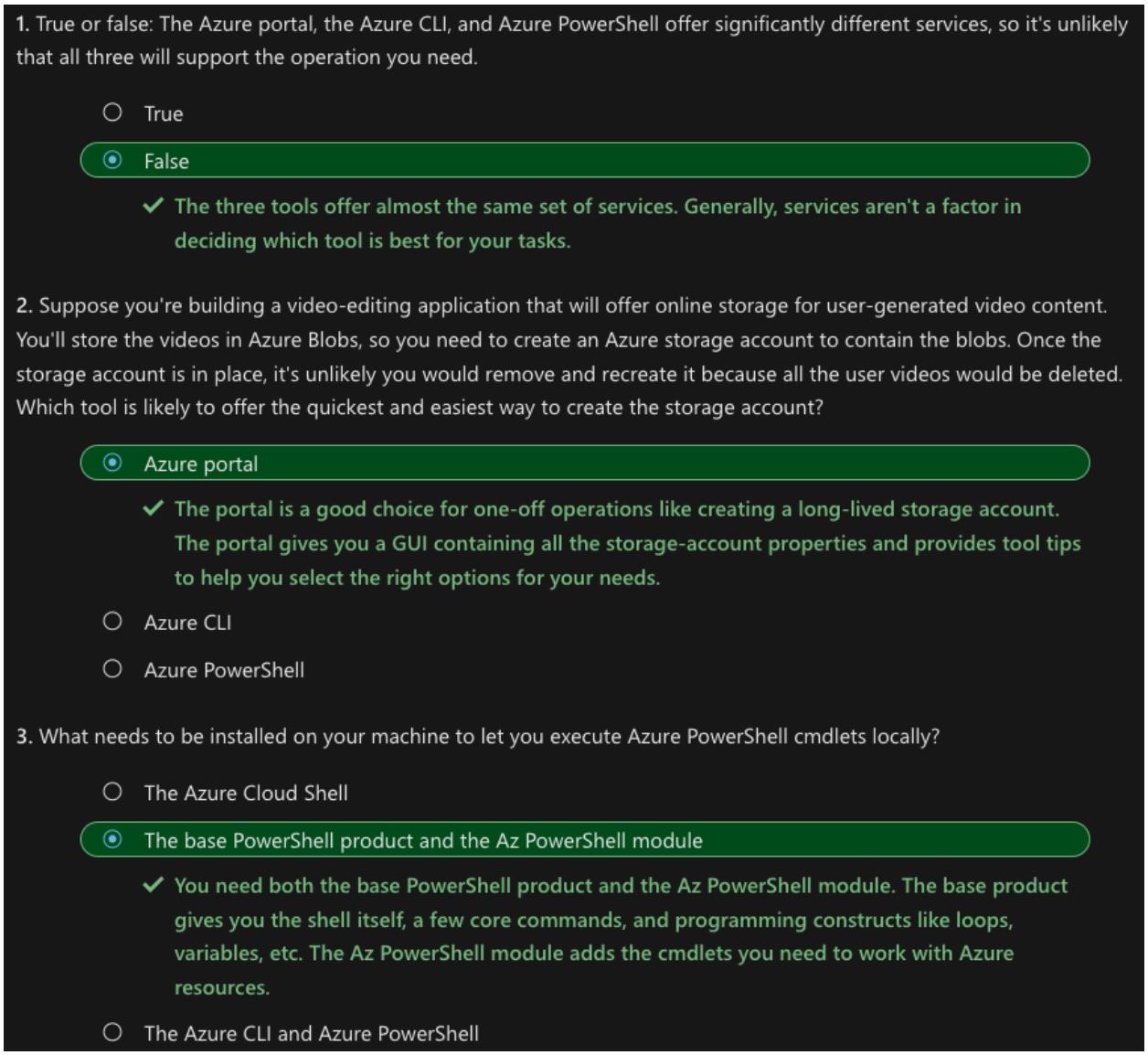
How to create a resource group with Azure PowerShell? There are four steps you need to perform:

- Import the Azure cmdlets.
- Connect to your Azure subscription.
- Create the resource group.
- Verify that creation was successful.

Beginning with PowerShell 3.0, modules are loaded automatically when you use a cmdlet within the module. It's no longer necessary to manually import PowerShell modules unless you've changed the default module autoloading settings.

A PowerShell script is a text file containing commands and control constructs. The commands are invocations of cmdlets. The control constructs are programming features like loops, variables, parameters, comments, etc., supplied by PowerShell.

PowerShell script files have a .ps1 file extension. You can create and save these files with any text editor.



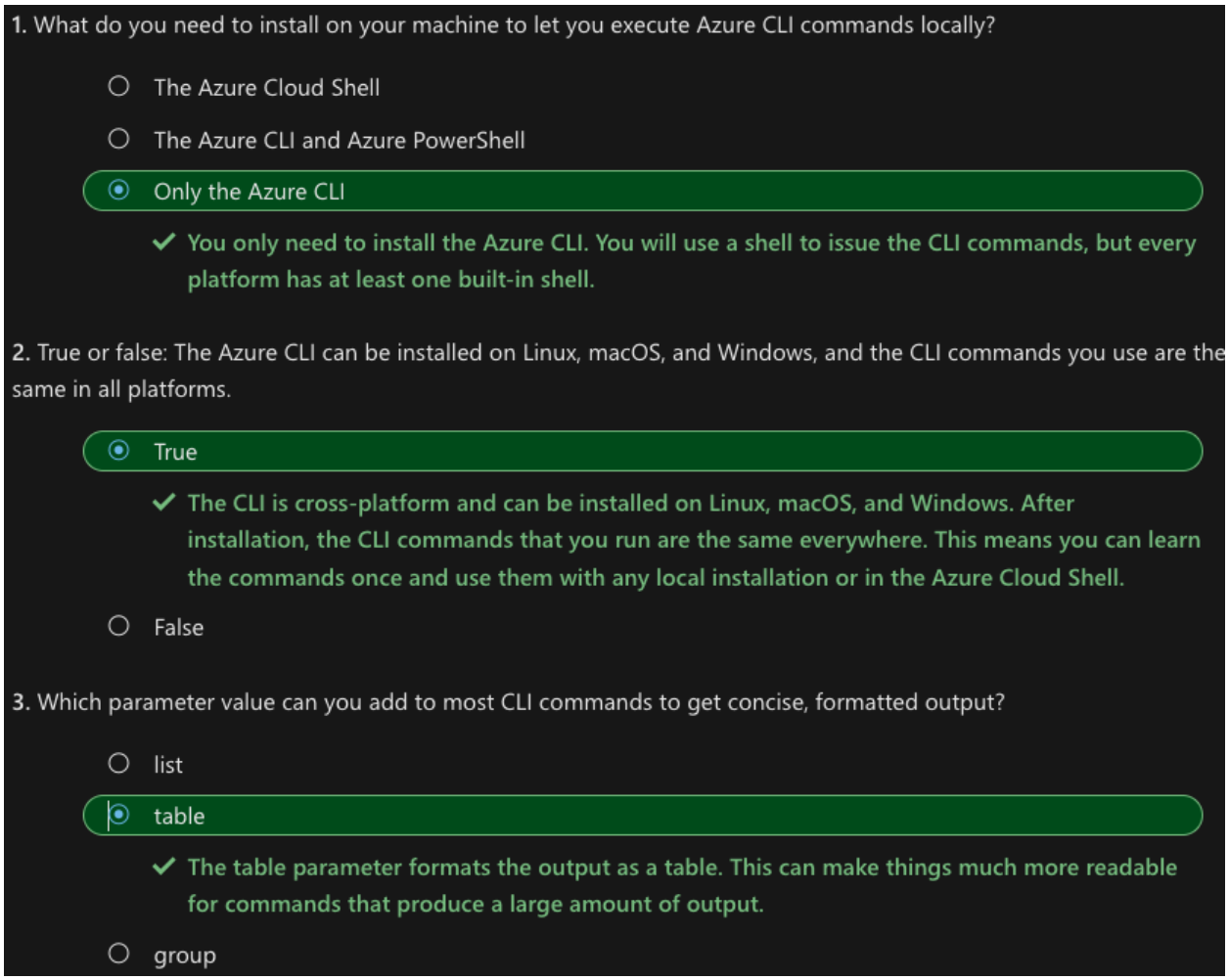
Control Azure services with the CLI

The Azure CLI provides cross-platform command-line tools for managing Azure resources, and you can easily install it locally on Linux, Mac, or Windows computers. You can also use the Azure CLI from a browser through the Azure Cloud Shell.

On both Linux and macOS, you'll use a package manager to install the Azure CLI. The recommended package manager differs by OS and distribution:

Linux: apt-get on Ubuntu, yum on Red Hat, and zypper on OpenSUSE
Mac: Homebrew
The Azure CLI is available in the Microsoft repository, so you'll first need to add that repository to your package manager.

On Windows, you can install the Azure CLI by downloading and running an MSI file.



Deploy Azure infrastructure by using JSON ARM templates

ARM templates are idempotent, which means you can deploy the same template many times and get the same resource types in the same state.

ARM templates allow you to automate deployments and use the practice of infrastructure as code (IaC).

Resource Manager also has built-in validation. It checks the template before starting the deployment to make sure the deployment will succeed.

If your deployments become more complex, you can break your ARM templates into smaller, reusable components. You can link these smaller templates together at deployment time. You can also nest templates inside other templates.

You can also integrate your ARM templates into continuous integration and continuous deployment (CI/CD) tools like Azure Pipelines, which can automate your release pipelines for fast and reliable application and infrastructure updates.

You can deploy an ARM template to Azure in one of the following ways:

- Deploy a local template.
- Deploy a linked template.
- Deploy in a continuous deployment pipeline.

In the parameters section of the template, you specify which values you can input when you deploy the resources. You're limited to 256 parameters in a template. Parameter definitions can use most template functions.

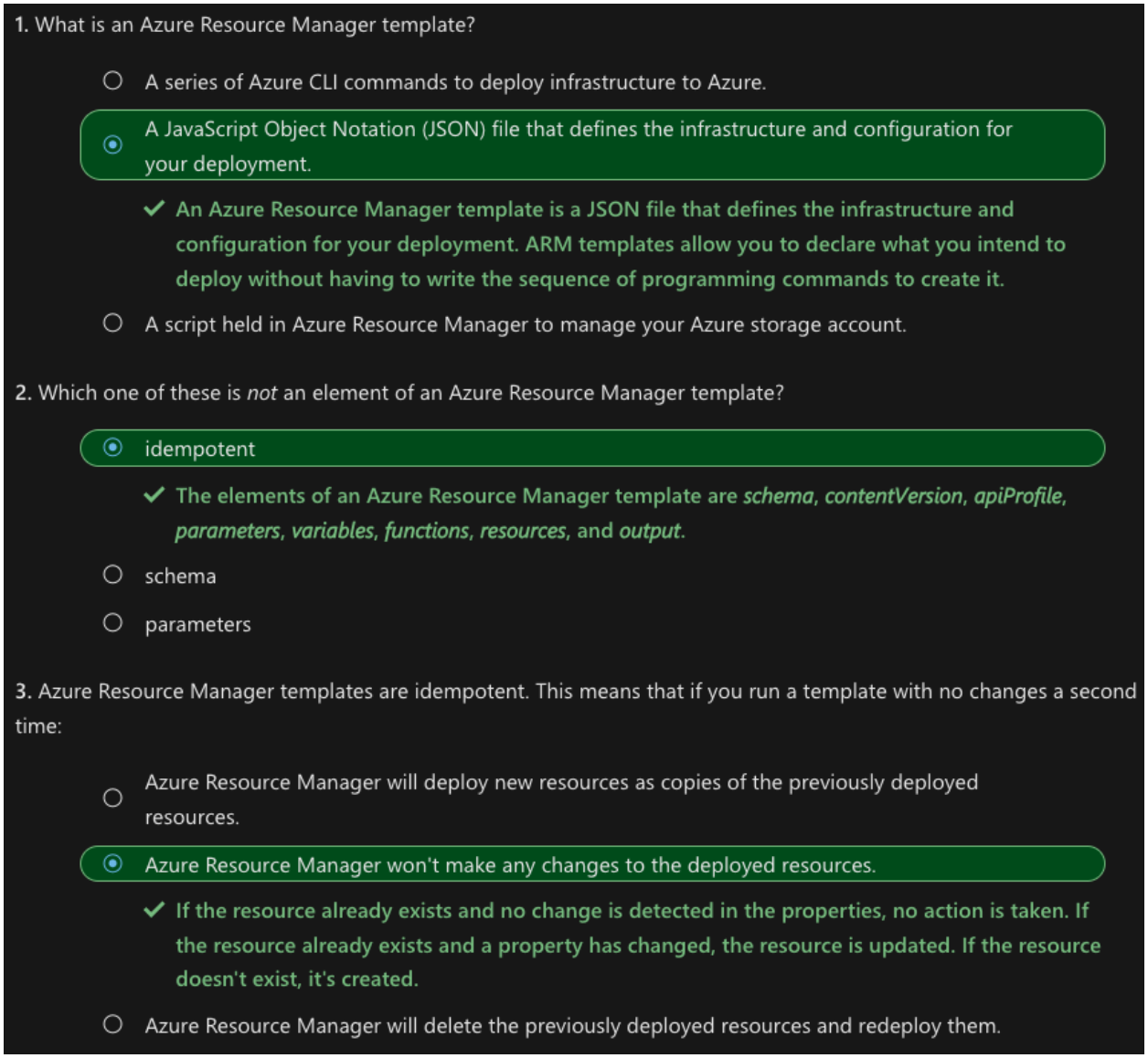
The allowed types of parameters are:

- string
- secureString
- integers
- boolean
- object
- secureObject
- array

For security reasons, never hard code or provide default values for usernames and/or passwords in templates. Always use parameters for usernames and passwords (or secrets). Use secureString for all passwords and secrets.

In the outputs section of your ARM template, you can specify values that will be returned after a successful deployment. Here are the elements that make up the outputs section.

```
"outputs": {
  "<output-name>": {
    "condition": "<boolean-value-whether-to-output-value>",
    "type": "<type-of-output-value>",
    "value": "<output-value-expression>",
    "copy": {
      "count": <number-of-iterations>,
      "input": <values-for-the-variable>
    }
  }
}
```



AZ-104: Manage identities and governance in Azure

Configure Azure Active Directory

In this module, your company is planning to implement Azure Active Directory (Azure AD) and features like Azure AD Join and Self-Service Password Reset. You need to understand how to choose the Azure AD edition that works best for your organization, and explore how to implement required features.

Azure Active Directory (Azure AD) is Microsoft's multi-tenant cloud-based directory and identity management service.

Things to know about Azure AD features

Let's examine some of the prominent features of Azure AD.

Azure AD feature	Description
Single sign-on (SSO) access	Azure AD provides secure single sign-on (SSO) to web apps on the cloud and to on-premises apps. Users can sign in with the same set of credentials to access all their apps.
Ubiquitous device support	Azure AD works with iOS, macOS, Android, and Windows devices, and offers a common experience across the devices. Users can launch apps from a personalized web-based access panel, mobile app, Microsoft 365, or custom company portals by using their existing work credentials.
Secure remote access	Azure AD enables secure remote access for on-premises web apps. Secure access can include multifactor authentication (MFA), conditional access policies, and group-based access management. Users can access on-premises web apps from everywhere, including from the same portal.
Cloud extensibility	Azure AD can extend to the cloud to help you manage a consistent set of users, groups, passwords, and devices across environments.
Sensitive data protection	Azure AD offers unique identity protection capabilities to secure your sensitive data and apps. Admins can monitor for suspicious sign-in activity and potential vulnerabilities in a consolidated view of users and resources in the directory.
Self-service support	Azure AD lets you delegate tasks to company employees that might otherwise be completed by admins with higher access privileges. Providing self-service app access and password management through verification steps can reduce helpdesk calls and enhance security.

The following table describes the main components and concepts of Azure AD and explains how they work together to support service features.

Azure AD concept	Description
Identity	An *identity* is an object that can be authenticated. The identity can be a user with a username and password. Identities can also be applications or other servers that require authentication by using secret keys or certificates. Azure AD is the underlying product that provides the identity service.
Account	An *account* is an identity that has data associated with it. To have an account, you must first have a valid identity. You can't have an account without an identity.
Azure AD account	An *Azure AD account* is an identity that's created through Azure AD or another Microsoft cloud service, such as Microsoft 365. Identities are stored in Azure AD and are accessible to your organization's cloud service subscriptions. The Azure AD account is also called a *work or school account*.
Azure tenant (directory)	An Azure *tenant* is a single dedicated and trusted instance of Azure AD. Each tenant (also called a *directory*) represents a single organization. When your organization signs up for a Microsoft cloud service subscription, a new tenant is automatically created. Because each tenant is a dedicated and trusted instance of Azure AD, you can create multiple tenants or instances.
Azure subscription	An Azure subscription is used to pay for Azure cloud services. Each subscription is joined to a single tenant. You can have multiple subscriptions.

Active Directory Domain Services (AD DS) is the traditional deployment of Windows Server-based Active Directory on a physical or virtual server. Active Directory Domain Services (AD DS) also includes Active Directory Certificate Services (AD CS), Active Directory Lightweight Directory Services (AD LDS), Active Directory Federation Services (AD FS), and Active Directory Rights Management Services (AD RMS).

Although you can deploy and manage AD DS in Azure Virtual Machines, we recommend you use Azure Active Directory, unless your configuration targets IaaS workloads that depend specifically on AD DS.

Things to consider when using Azure AD rather than AD DS:

- Identity solution: AD DS is primarily a directory service, while Azure AD is a full identity solution. Azure AD is designed for internet-based applications that use HTTP and HTTPS communications. The features and capabilities of Azure AD support target strong identity management.

- Communication protocols: Because Azure AD is based on HTTP and HTTPS, it doesn't use Kerberos authentication. Azure AD implements HTTP and HTTPS protocols, such as SAML, WS-Federation, and OpenID Connect for authentication (and OAuth for authorization).

- Federation services: Azure AD includes federation services, and many third-party services like Facebook.

- Flat structure: Azure AD users and groups are created in a flat structure. There are no organizational units (OUs) or group policy objects (GPOs).

- Managed service: Azure AD is a managed service. You manage only users, groups, and policies. If you deploy AD DS with virtual machines by using Azure, you manage many other tasks, including deployment, configuration, virtual machines, patching, and other backend processes.

Azure Active Directory comes in four editions: Free, Microsoft 365 Apps, Premium P1, and Premium P2. The Free edition is included with an Azure subscription. The Premium editions are available through a Microsoft Enterprise Agreement, the Open Volume License Program, and the Cloud Solution Providers program. Azure and Microsoft 365 subscribers can also buy Azure Active Directory Premium P1 and P2 online.

Feature	Free	Microsoft 365 Apps	Premium P1	Premium P2
Directory Objects	500,000	Unlimited	Unlimited	Unlimited
Single Sign-on	Unlimited	Unlimited	Unlimited	Unlimited
Core Identity and Access Management	X	X	X	X
Business-to-business Collaboration	X	X	X	X
Identity and Access Management for Microsoft 365 apps		X	X	X
Premium Features			X	X
Hybrid Identities			X	X
Advanced Group Access Management			X	X
Conditional Access			X	X
Identity Protection				X
Identity Governance				X

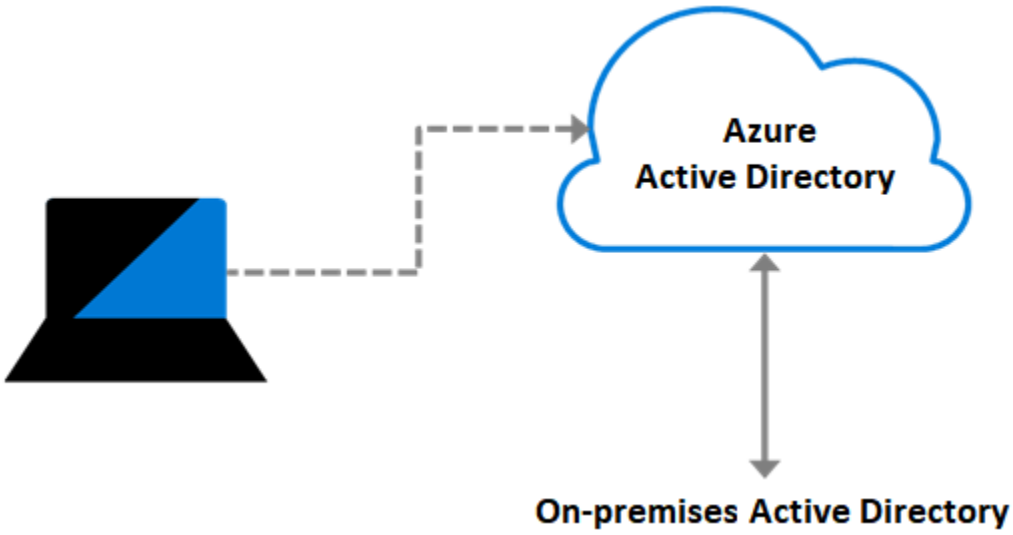
The Free edition provides user and group management, on-premises directory synchronization, and basic reports. Single sign-on access is supported across Azure, Microsoft 365, and many popular SaaS apps.

Azure Active Directory Microsoft 365 Apps is included with Microsoft 365. In addition to the Free features, this edition provides Identity and Access Management for Microsoft 365 apps. The extra support includes branding, MFA, group access management, and self-service password reset for cloud users.

In addition to the Free features, the Premium P1 edition lets your hybrid users access both on-premises and cloud resources. This edition supports advanced administration like dynamic groups, self-service group management, and cloud write-back capabilities. P1 also includes Microsoft Identity Manager, an on-premises identity and access management suite. The extra features in P1 allow self-service password reset for your on-premises users.

In addition to the Free and P1 features, the Premium P2 edition offers Azure AD Identity Protection to help provide risk-based Conditional Access to your apps and critical company data. Privileged Identity Management is included to help discover, restrict, and monitor administrators and their access to resources, and to provide just-in-time access when needed.

Azure Active Directory enables single sign-on (SSO) to devices, applications, and services from anywhere. To support SSO, IT admins must ensure corporate assets are protected, and devices meet standards for security and compliance. The Azure AD join feature works with SSO to provide access to organizational apps and resources, and to simplify Windows deployments of work-owned devices.



Let's look at some of the benefits of using joined devices:

Benefit	Description
Single-Sign-On (SSO)	Joined devices offer SSO access to your Azure-managed SaaS apps and services. Your users won't have extra authentication prompts when they access work resources. The SSO functionality is available even when users aren't connected to the domain network.
Enterprise state roaming	Starting in Windows 10, your users can securely synchronize their user settings and app settings data to joined devices. Enterprise state roaming reduces the time to configure a new device.
Access to Microsoft Store for Business	When your users access Microsoft Store for Business by using an Azure AD account, they can choose from an inventory of applications pre-selected by your organization.
Windows Hello	Provide your users with secure and convenient access to work resources from joined devices.
Restriction of access	Restrict user access to apps from only joined devices that meet your compliance policies.
Seamless access to on-premises resources	Joined devices have seamless access to on-premises resources, when the device has line of sight to the on-premises domain controller.

Things to consider when using joined devices:

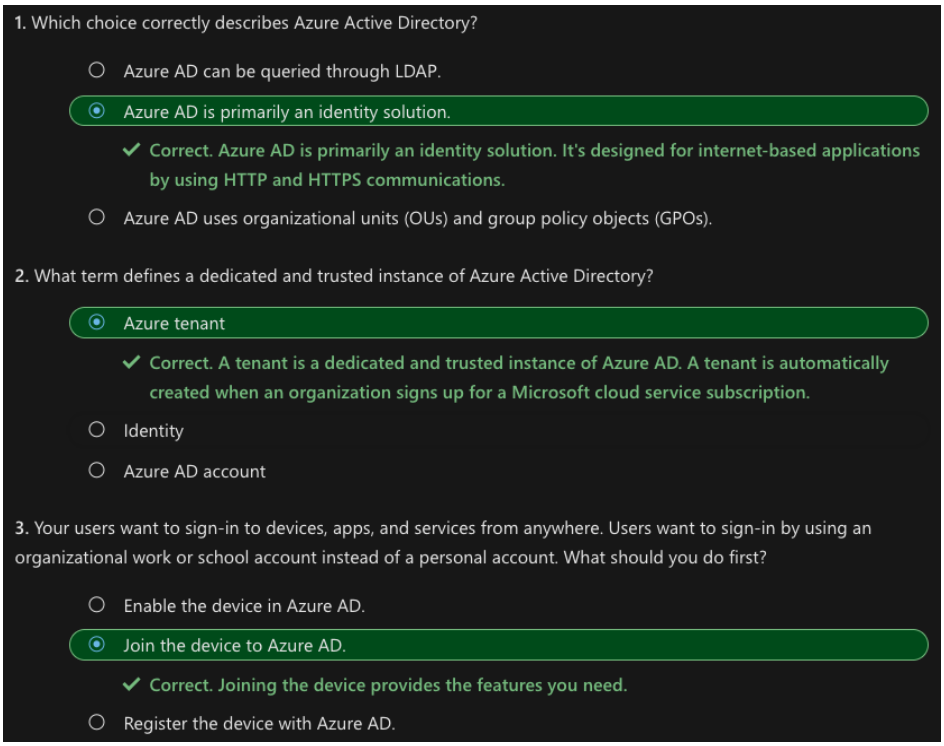
- Consider connection options. Connect your device to Azure AD in one of two ways:
    - Register your device to Azure AD so you can manage the device identity. Azure AD device registration provides the device with an identity that's used to authenticate the device when a user signs into Azure AD. You can use the identity to enable or disable the device.
    - Join your device, which is an extension of registering a device. Joining provides the benefits of registering, and also changes the local state of the device. Changing the local state enables your users to sign into a device by using an organizational work or school account instead of a personal account.
- Consider combining registration with other solutions. Combine registration with a mobile device management (MDM) solution like Microsoft Intune, to provide other device attributes in Azure AD. You can create conditional access rules that enforce access from devices to meet organization standards for security and compliance.
- Consider other implementation scenarios. Although AD Join is intended for organizations that don't have an on-premises Windows Server Active Directory infrastructure, it can be used for other scenarios like branch offices.

The Azure Active Directory self-service password reset (SSPR) feature lets you give users the ability to bypass helpdesk and reset their own passwords.

- SSPR requires an Azure AD account with Global Administrator privileges to manage SSPR options. This account can always reset their own passwords, no matter what options are configured.
- SSPR uses a security group to limit the users who have SSPR privileges.
- All user accounts in your organization must have a valid license to use SSPR.

Things to consider when using SSPR:

- Consider who can reset their passwords. Decide which users in your organization should be enabled to use the feature. In the Azure portal, there are three options for the SSPR feature: None, Selected, and All.
    - The Selected option is useful for creating specific groups who have SSPR enabled. You can create groups for testing or proof of concept before applying the feature to a larger group. When you're ready to deploy SSPR to all user accounts in your Azure AD tenant, you can change the setting.
- Consider your authentication methods. Determine how many authentication methods are required to reset a password, and select the authentication options for users.
    - Your system must require at least one authentication method to reset a password.
    - A strong SSPR plan offers multiple authentication methods for the user. Options include email notification, text message, or a security code sent to the user's mobile or office phone. You can also offer the user a set of security questions.
    - You can require security questions to be registered for the users in your Azure AD tenant.
    - You can configure how many correctly answered security questions are required for a successful password reset.
- Consider combining methods for stronger security.



Configure user and group accounts

Every user who wants access to Azure resources needs an Azure user account. A user account has all the information required to authenticate the user during the sign-in process. Azure Active Directory (Azure AD) supports three types of user accounts. The types indicate where the user is defined (in the cloud or on-premises), and whether the user is internal or external to your Azure AD organization.

The following table describes the user accounts supported in Azure AD.

User account	Description
Cloud identity	A user account with a *cloud identity* is defined only in Azure AD. This type of user account includes administrator accounts and users who are managed as part of your organization. A cloud identity can be for user accounts defined in your Azure AD organization, and also for user accounts defined in an external Azure AD instance. When a cloud identity is removed from the primary directory, the user account is deleted.
Directory-synchronized identity	User accounts that have a *directory-synchronized identity* are defined in an on-premises Active Directory. A synchronization activity occurs via Azure AD Connect to bring these user accounts in to Azure. The source for these accounts is Windows Server Active Directory.
Guest user	*Guest user* accounts are defined outside Azure. Examples include user accounts from other cloud providers, and Microsoft accounts like an Xbox LIVE account. The source for guest user accounts is Invited user. Guest user accounts are useful when external vendors or contractors need access to your Azure resources.

There are several ways to add cloud identity user accounts in Azure Active Directory (Azure AD). A common approach is by using the Azure portal. User accounts can also be added to Azure AD through Microsoft 365 Admin Center, Microsoft Intune admin console, and the Azure CLI.

- A new user account must have a display name and an associated user account name.
- Non-admin users can set some of their own profile data, but they can't change their display name or account name.

Things to consider when managing cloud identity accounts:

- Consider user profile data. Allow users to set their profile information for their accounts, as needed.
- Consider restore options for deleted accounts. Include restore scenarios in your account management plan. Restore operations for a deleted account are available for up to 30 days.
- Consider gathered account data.

Azure Active Directory (Azure AD) supports several bulk operations, including bulk create and delete for user accounts.

- Only Global administrators or User administrators have privileges to create and delete user accounts in the Azure portal.
- To complete bulk create or delete operations, the admin fills out a comma-separated values (CSV) template of the data for the user accounts.
- Bulk operation templates can be downloaded from the Azure AD portal.
- Bulk lists of user accounts can be downloaded.

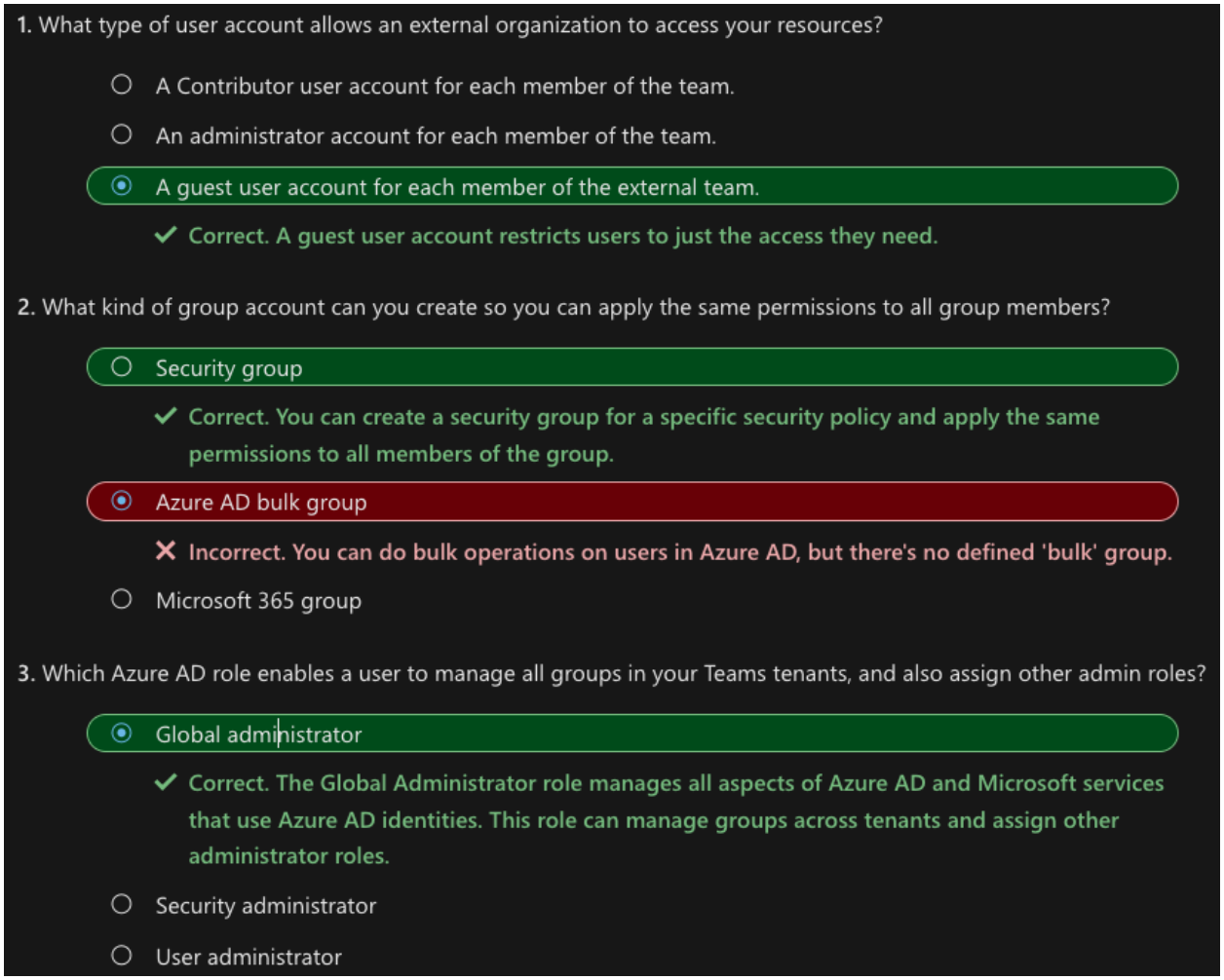
Things to consider when creating user accounts:

- Consider naming conventions.
- Consider using initial passwords. Implement a convention for the initial password of a newly created user. Design a system to notify new users about their passwords in a secure way. You might generate a random password and email it to the new user or their manager.
- Consider strategies for minimizing errors. View and address any errors, by downloading the results file on the Bulk operation results page in the Azure portal. The results file contains the reason for each error. An error might be a user account that's already been created or an account that's duplicated. Generally, it's easier to upload and troubleshoot smaller groups of user accounts.

Azure Active Directory (Azure AD) allows your organization to define two different types of group accounts. Security groups are used to manage member and computer access to shared resources for a group of users. You can create a security group for a specific security policy and apply the same permissions to all members of a group. Microsoft 365 groups provide collaboration opportunities. Group members have access to a shared mailbox, calendar, files, SharePoint site, and more.

- Use security groups to set permissions for all group members at the same time, rather than adding permissions to each member individually.
- Add Microsoft 365 groups to enable group access for guest users outside your Azure AD organization.
- Security groups can be implemented only by an Azure AD administrator.
- Normal users and Azure AD admins can both use Microsoft 365 groups.

Access rights	Description
Assigned	Add specific users as members of a group, where each user can have unique permissions.
Dynamic user	Use dynamic membership rules to automatically add and remove group members. When member attributes change, Azure reviews the dynamic group rules for the directory. If the member attributes meet the rule requirements, the member is added to the group. If the member attributes no longer meet the rule requirements, the member is removed.
Dynamic device	(*Security groups only*) Apply dynamic group rules to automatically add and remove devices in security groups. When device attributes change, Azure reviews the dynamic group rules for the directory. If the device attributes meet the rule requirements, the device is added to the security group. If the device attributes no longer meet the rule requirements, the device is removed.



Configure subscriptions

Access to Azure resources and services is obtained through Azure subscriptions. Payment for services is done through Microsoft Cost Management and Billing.

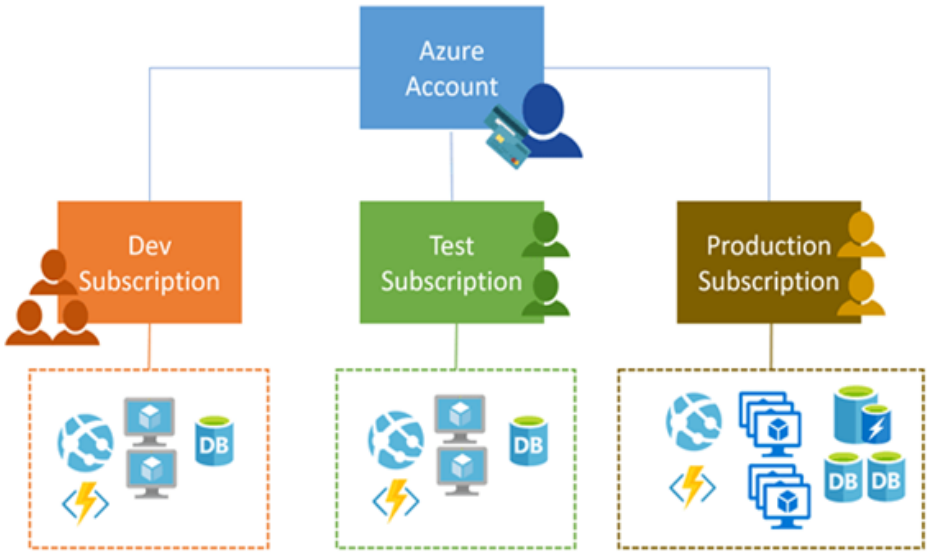
Microsoft Azure is made up of datacenters located around the globe. These datacenters are organized and made available to end users by region. A region is a geographical area on the planet containing at least one, but potentially multiple datacenters. The datacenters are in close proximity and networked together with a low-latency network. A few examples of regions are West US, Canada Central, West Europe, Australia East, and Japan West.

- Azure is generally available in more than 60 regions in 140 countries.
- Azure has more global regions than any other cloud provider.
- Regions provide you with the flexibility and scale needed to bring applications closer to your users.
- Regions preserve data residency and offer comprehensive compliance and resiliency options for customers.

Most Azure regions are paired with another region within the same geography to make a regional pair (or paired regions). Regional pairs help to support always-on availability of Azure resources used by your infrastructure.

Characteristic	Description
Physical isolation	Azure prefers at least 300 miles of separation between datacenters in a regional pair. This principle isn't practical or possible in all geographies. Physical datacenter separation reduces the likelihood of natural disasters, civil unrest, power outages, or physical network outages affecting both regions at once.
Platform-provided replication	Some services like Geo-Redundant Storage provide automatic replication to the paired region.
Region recovery order	During a broad outage, recovery of one region is prioritized out of every pair. Applications that are deployed across paired regions are guaranteed to have one of the regions recovered with priority.
Sequential updates	Planned Azure system updates are rolled out to paired regions sequentially (not at the same time). Rolling updates minimizes downtime, reduces bugs, and logical failures in the rare event of a bad update.
Data residency	Regions reside within the same geography as their enabled set (except for the Brazil South and Singapore regions).

An Azure subscription is a logical unit of Azure services that's linked to an Azure account. An Azure account is an identity in Azure Active Directory (Azure AD) or a directory that's trusted by Azure AD, such as a work or school account. Subscriptions help you organize access to Azure cloud service resources, and help you control how resource usage is reported, billed, and paid.



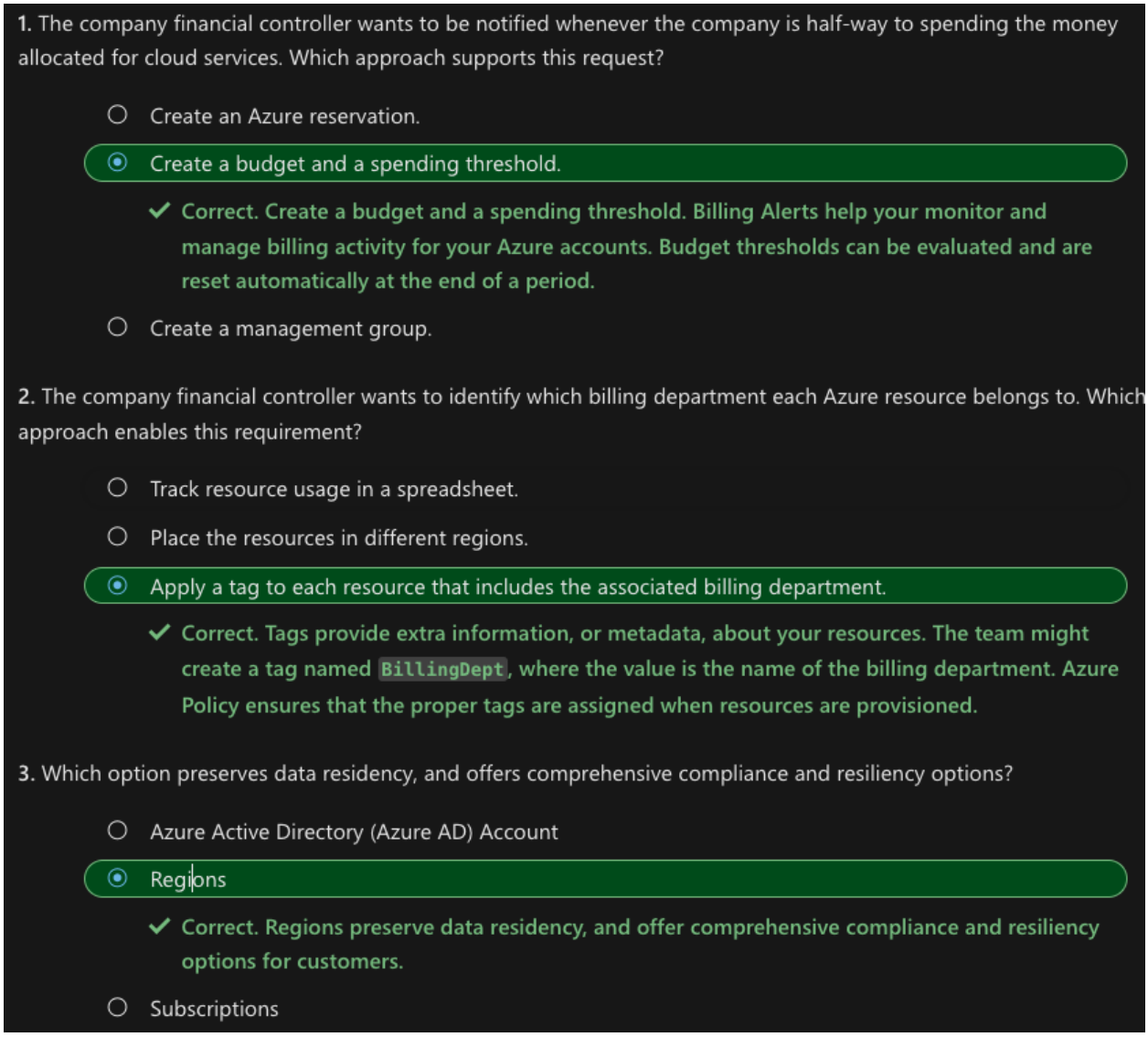
- Every Azure cloud service belongs to a subscription.
- Each subscription can have a different billing and payment configuration.

- Multiple subscriptions can be linked to the same Azure account.
- More than one Azure account can be linked to the same subscription.
- Billing for Azure services is done on a per-subscription basis.
- If your Azure account is the only account associated with a subscription, you're responsible for the billing requirements.
- Programmatic operations for a cloud service might require a subscription ID.

To use Azure, you must have an Azure subscription. There are several ways to procure an Azure subscription. You can obtain an Azure subscription as part of an Enterprise agreement, or through a Microsoft reseller or Microsoft partner. Users can also open a personal free account for a trial subscription. Azure offers free and paid subscription options to meet different needs and requirements. The most common subscriptions are Free, Pay-As-You-Go, Enterprise Agreement, and Student.

You can apply tags to your Azure resources to logically organize them by categories. Tags are useful for sorting, searching, managing, and doing analysis on your resources. Each resource tag consists of a name and a value. You could have the tag name Server and the value Production or Development, and then apply the tag/value pair to your Engineering computer resources.

- The tag name remains constant for all resources that have the tag applied.
- The tag value can be selected from a defined set of values, or unique for a specific resource instance.
- A resource or resource group can have a maximum of 50 tag name/value pairs.
- Tags applied to a resource group aren't inherited by the resources in the resource group.



Configure Azure Policy

Configure role-based access control

Create Azure users and groups in Azure Active Directory

Secure your Azure resources with Azure role-based access control (Azure RBAC)

Allow users to reset their password with Azure Active Directory self-service password reset