

AZ-104 Microsoft Azure Administrator

Course topics

| | |
|---|-------|
| 1. Administer Identity | ...2 |
| 2. Administer Governance and Compliance | ...8 |
| 3. Administer Azure Resources | ...14 |
| 4. Administer Virtual Networking | ...21 |
| 5. Administer Intersite Connectivity | ...28 |
| 6. Administer Network Traffic | ...35 |
| 7. Administer Azure Storage | ...40 |
| 8. Administer Azure Virtual Machines | ...48 |
| 9. Administer PaaS Compute Options | ...54 |
| 10. Administer Data Protection | ...60 |
| 11. Administer Monitoring | ...65 |

01 Administer Identity

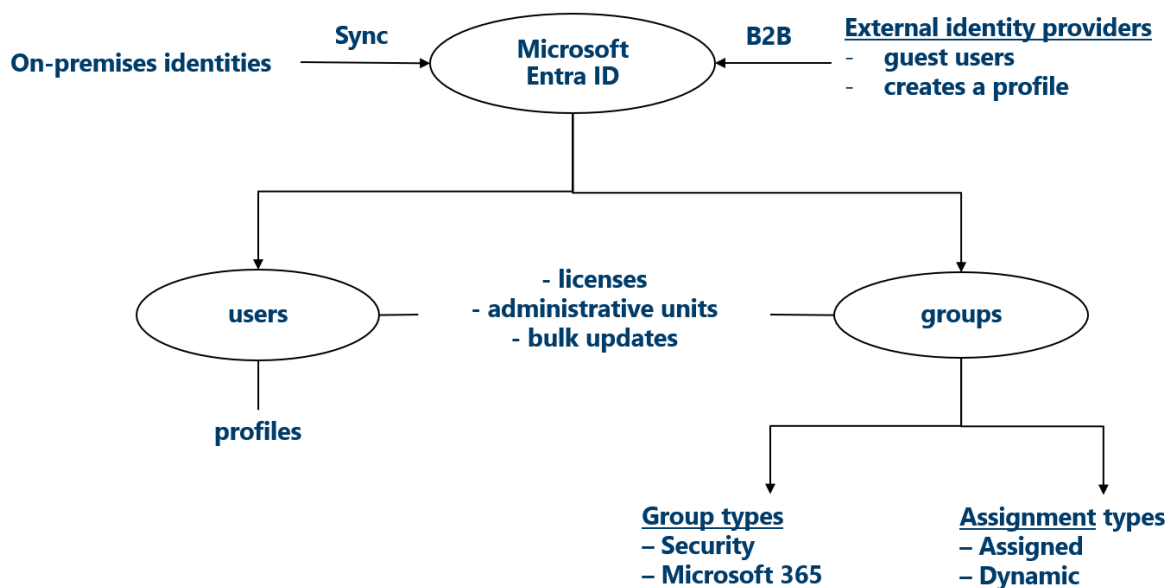
1. Microsoft Entra ID (Formerly Azure Active Directory)

- **Definition:**

Microsoft Entra ID (formerly Azure Active Directory or Azure AD) is a cloud-based identity and access management service. It enables organizations to manage users, groups, and their access to applications, services, and resources.

- **Purpose:**

Provides secure access to Azure resources, SaaS applications (like Microsoft 365), and on-premises applications.



- **Key Features:**

1. **Authentication:**

- Supports different types of sign-ins: password-based, multi-factor authentication (MFA), and passwordless.
- Provides single sign-on (SSO) for seamless access to multiple applications.

2. Authorization:

Enforces access policies based on user identity, group membership, or device compliance.

3. Identity Protection:

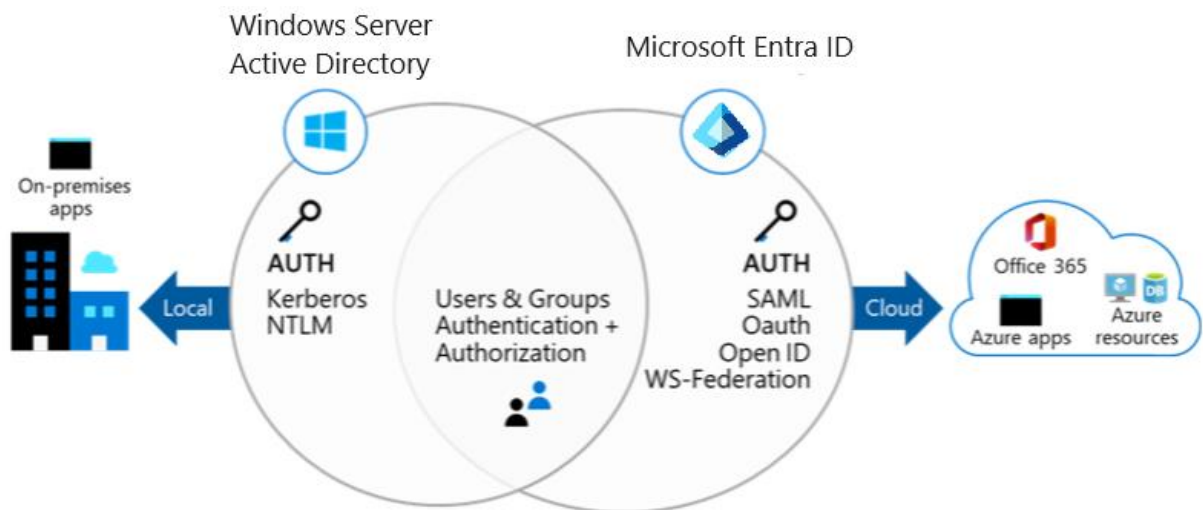
Detects and responds to identity-based threats using tools like risk-based conditional access and real-time risk analysis.

4. Integration with Applications:

- Supports thousands of pre-integrated SaaS applications (e.g., Salesforce, Google Workspace).
- Custom applications can also be integrated using OAuth, SAML, or OpenID Connect protocols.

5. Hybrid Identity:

Integrates on-premises Active Directory with Microsoft Entra ID using Azure AD Connect for hybrid identity scenarios.



- **Benefits:**

- Centralized identity management across cloud and on-premises environments.
- Enhanced security with Conditional Access and MFA.
- Streamlined user experience with SSO.

- **Example Use Case:**

A company uses Microsoft Entra ID to allow employees to log in once (SSO) and access multiple applications, such as Microsoft 365, Dynamics 365, and third-party apps like Dropbox or Zoom.

2. User and Group Accounts

User Accounts:

1. **Definition:**

Represents an individual who requires access to Azure resources or applications.

2. **Types of Users:**

- **Cloud Users:** Accounts created directly in Microsoft Entra ID.
- **Synchronized Users:** Accounts synchronized from on-premises Active Directory using Azure AD Connect.
- **Guest Users:** External users invited via the B2B (Business-to-Business) feature for collaboration.

3. **User Management Tasks:**

- **Add/Remove Users:** Administrators can manually add users or automate via scripts.
- **Password Management:** Manage and reset user passwords. Self-service password reset (SSPR) can be enabled.
- **Assign Roles:** Assign RBAC roles or Entra roles like Global Administrator, User Administrator, etc.

Self-Service Password Reset

Password reset - Authentication methods
mitanic (Default Directory)

Save Discard

Diagnose and solve problems

Manage

- 1 Properties
- 2 **Authentication methods**
- 3 Registration

Notifications

Customization

On-premises integration

Activity

Audit logs

Usage & insights

Troubleshooting + Support

New support request

Number of methods required to reset ⓘ

1 2

Methods available to users

☐ Mobile app notification

☐ Mobile app code

☒ Email

☒ Mobile phone

☐ Office phone

☒ Security questions

Number of questions required to register ⓘ

3 4 5

Number of questions required to reset ⓘ

3 4 5

Select security questions

5 security questions selected




- **Group Accounts:**

1. **Definition:**

Groups are collections of users that simplify access management by assigning permissions or roles to the group instead of individual users.

2. **Types of Groups:**

- **Security Groups:** Used to manage access to Azure resources and applications.
- **Microsoft 365 Groups:** Used for collaboration, including access to shared mailboxes, SharePoint sites, and Teams.

| <input type="text" value="Search groups"/> | | <input type="button" value="+ Add filters"/> | |
|--|--|--|-----------------|
| | Name | ↑↓ Group Type | Membership Type |
| <input type="checkbox"/> |  Managers | Security | Assigned |
| <input type="checkbox"/> |  Virtual Machine Administrators | Security | Assigned |
| <input type="checkbox"/> |  Virtual Network Administrators | Security | Assigned |

3. Dynamic Groups:

Groups whose membership is automatically updated based on predefined rules (e.g., users in a specific department or location).

4. Group Management Tasks:

- Create and delete groups.
- Add or remove members.
- Configure group-based access to resources or applications.

- **Benefits of Groups:**

- Simplifies permissions management.
- Enhances security by reducing the risk of incorrect individual assignments.
- Automates membership using dynamic groups.

- **Example Use Case:**

- A company creates a security group called "Finance Team" and assigns it to access a specific SharePoint site, financial application, and related Azure resources.
- Adding or removing a user from the group automatically updates their access permissions.

How These Concepts Work Together

1. **Microsoft Entra ID** acts as the backbone for managing identities and access in Azure and beyond.
2. **User Accounts** provide the basic building blocks for identity management, representing individuals who need access.

3. **Group Accounts** simplify permissions management by organizing users and assigning access at a group level.

Practical Example:

An organization uses Microsoft Entra ID to manage identities for its employees and external partners. The IT team creates user accounts for employees and places them in groups such as "HR Department" or "IT Support." These groups are then granted access to specific resources or applications, ensuring users only have permissions relevant to their roles.

[Interactive Lab Simulation Lab 01](#) ← *Ctrl+Left Click to start the lab*

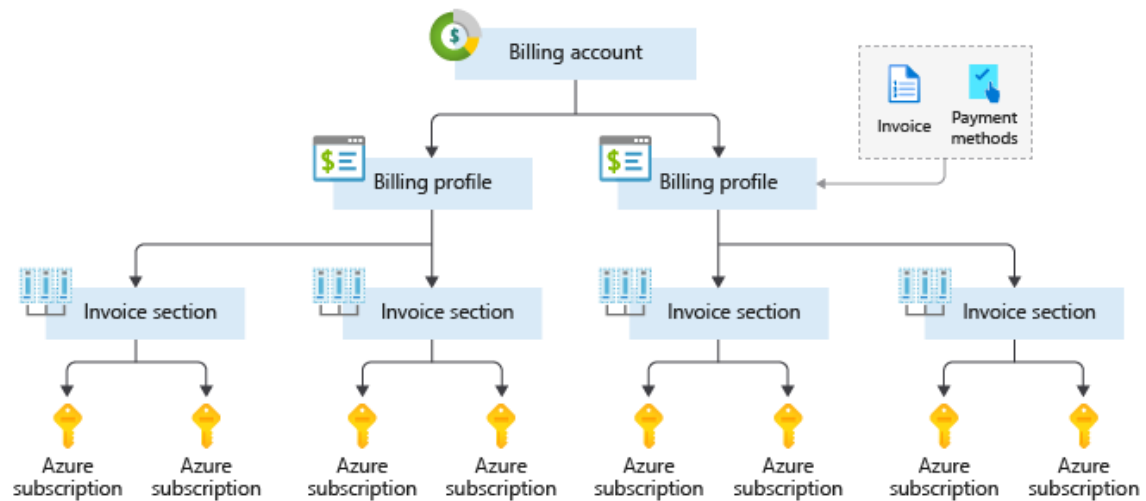


02 Administer Governance and Compliance

1. Subscriptions

- **Definition:**

A subscription in Azure is a logical grouping that acts as a container for Azure resources such as virtual machines, databases, storage accounts, and more. It helps organize resources and manage them effectively.



- **Purpose:**

Subscriptions help in resource management, cost tracking, and ensuring proper governance. They are often used to segment environments (e.g., Development, Test, Production) or departments within an organization.

- **Key Features:**

- **Billing and Costs:** Each subscription is tied to a billing account and provides detailed reports on resource usage and associated costs.
- **Resource Limits:** Subscriptions have specific limits on resources (e.g., number of VMs, storage accounts), which can be increased by contacting Microsoft.
- **Multiple Subscriptions:** Organizations often use multiple subscriptions for better control and cost separation. These subscriptions can be linked to a single Azure account.

- **Example:**

A company might have three subscriptions:

- *Subscription A*: For Development (low-cost environment).
- *Subscription B*: For Testing (medium-cost environment).
- *Subscription C*: For Production (high-availability and high-cost environment).

2. Azure Policy

- **Definition:**

Azure Policy is a governance service that allows you to create, assign, and manage policies to enforce rules and ensure compliance for your Azure resources.

- **Purpose:**

Azure Policy helps maintain control over your environment by enforcing standards and preventing configurations that violate your organizational guidelines.

- **Key Components:**

1. **Policy Definition:**

The rule or guideline to enforce. For example:

- “Restrict resource creation to specific Azure regions.”
- “Require all resources to have a cost center tag.”

2. **Policy Assignment:**

Policies are assigned to scopes like management groups, subscriptions, resource groups, or individual resources.

3. **Compliance Tracking:**

Azure Policy provides dashboards to track how many resources comply with the assigned policies.

- **Policy Examples:**

- Block creating virtual machines in regions outside of "East US" and "West US."
- Ensure all resources are encrypted using specific encryption standards.
- Automatically apply tags to resources (e.g., "Environment: Production").

- **Benefits:**

- Helps prevent configuration drift and maintain compliance.
- Automates resource governance, reducing manual intervention.

3. Role-Based Access Control (RBAC)

- **Definition:**

RBAC is a system that manages who has access to Azure resources and what actions they can perform. It ensures that users only have the permissions they need to do their job.

- **Purpose:**

RBAC provides fine-grained access control, improving security by limiting unnecessary access.

- **Key Elements of RBAC:**

1. **Roles:**

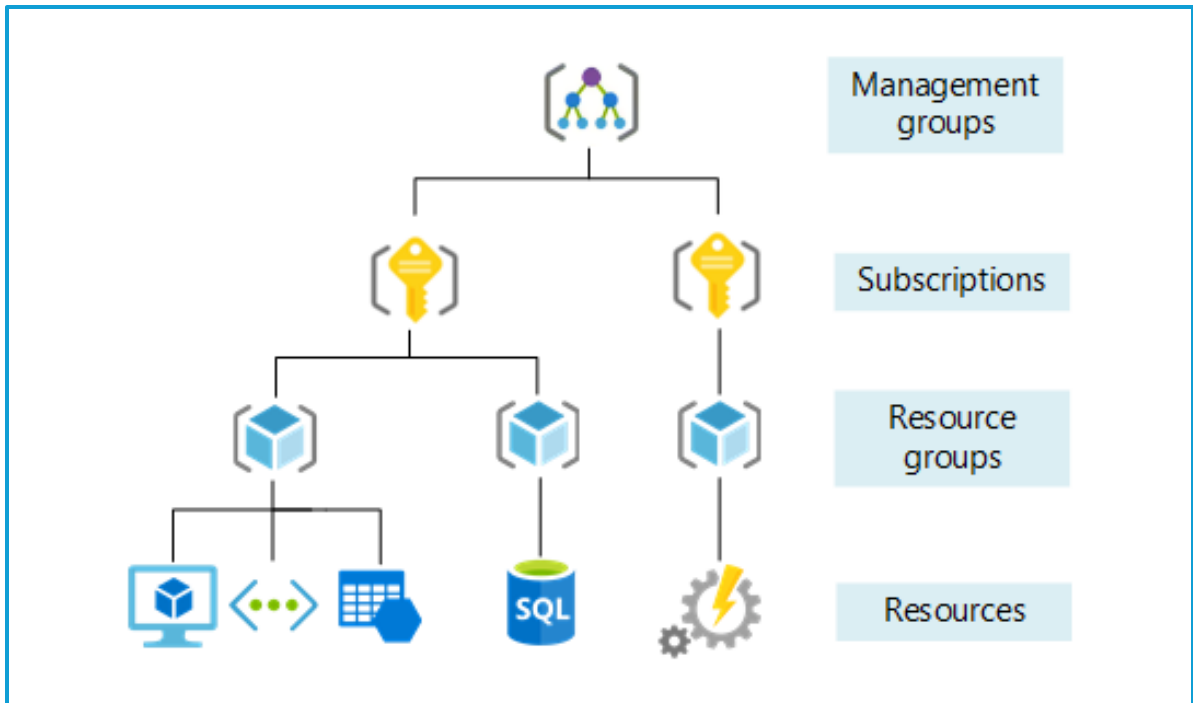
Roles define a set of permissions. Common built-in roles include:

- **Owner:** Full access, including assigning permissions.
- **Contributor:** Manage resources but cannot manage permissions.
- **Reader:** View-only access.

2. **Scope:**

Access can be assigned at different levels:

- **Management Group:** Highest level, applies to multiple subscriptions.
- **Subscription:** Controls access across all resources in a subscription.
- **Resource Group:** Limits access to resources within a specific group.
- **Individual Resource:** Grants access to a single resource like a VM or database.



- **RBAC Example:**

Imagine a company has a subscription with two resource groups:

- *Development Group:* A developer is assigned the "Contributor" role to create and modify resources in this group.
- *Production Group:* The same developer is assigned the "Reader" role to only view resources but not modify them in the production group.

- **Best Practices:**

- Assign roles at the lowest scope necessary to minimize risk.
- Use built-in roles for common scenarios, but create custom roles if unique permissions are needed.
- Regularly review and audit role assignments for compliance.

How These Concepts Work Together

- **Subscriptions** provide the foundation for organizing and managing resources in Azure.
- **Azure Policy** ensures that resources within subscriptions comply with organizational standards and governance rules.

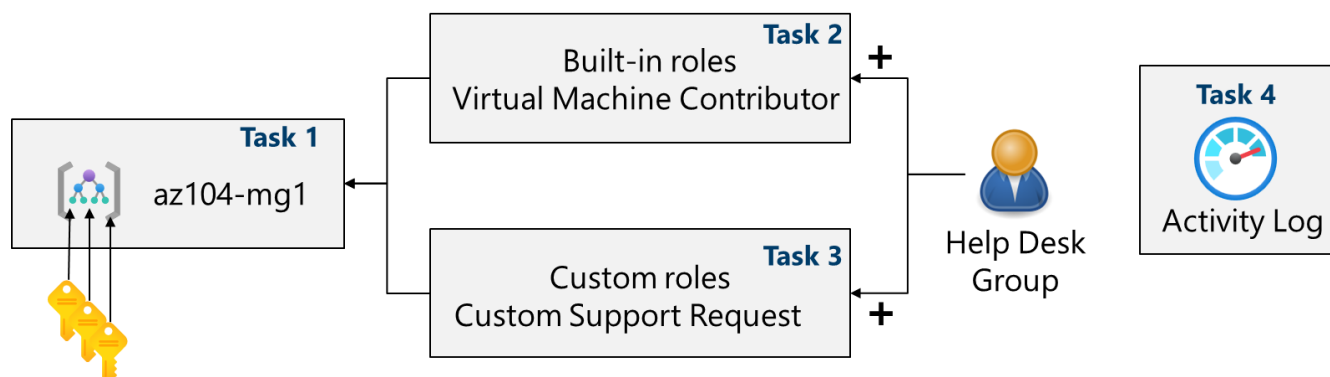
- **RBAC** controls who can access and perform actions on those resources within the subscriptions.

Example:

A company might have multiple subscriptions for different teams. Azure Policies enforce region restrictions and tagging rules across these subscriptions, while RBAC ensures only authorized personnel can manage or access the resources.

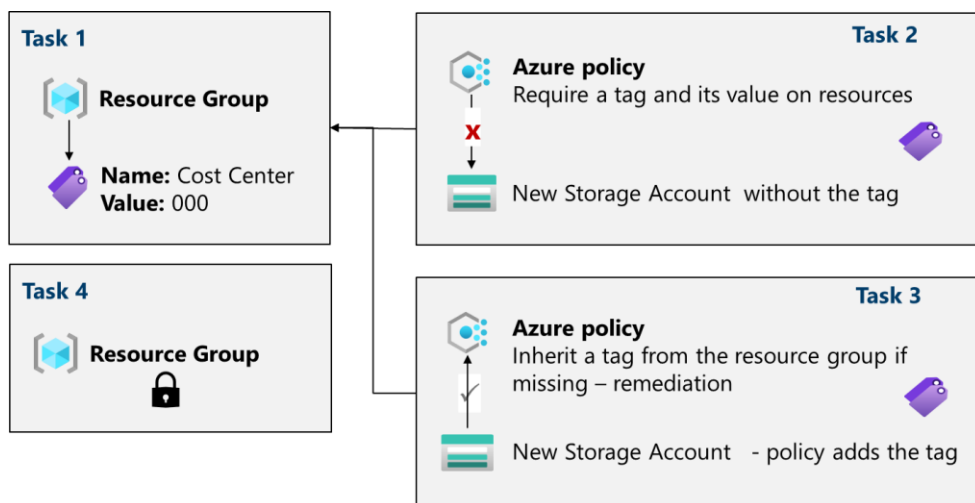
Interactive lab simulations

Lab 02a - Manage Subscriptions and RBAC



1. **Manage access with RBAC.** ⬅️ *Ctrl+Left Click to start the lab*
Assign built-in role to a user and monitor the activity logs.
2. **Manage subscriptions and RBAC.** ⬅️ *Ctrl+Left Click to start the lab*
Implement a management group and create and assign a custom RBAC role.
3. **Open a support request.** ⬅️ *Ctrl+Left Click to start the lab*
Review support plan options, then create and monitor a support request, technical or billing.

Lab 02b - Manage Governance via Azure Policy



1. **Manage resource locks.** ← *Ctrl+Left Click to start the lab*
Add a resource lock and test to confirm.
 2. **Create an Azure policy.** ← *Ctrl+Left Click to start the lab*
Create an Azure policy that restricts the location resources can be located. Create a new resource and ensure the policy is enforced.
 3. **Manage governance via Azure policy.** ← *Ctrl+Left Click to start the lab*
Create and assign tags via the Azure portal. Create an Azure policy that requires tagging. Remediate non-compliant resources.
-

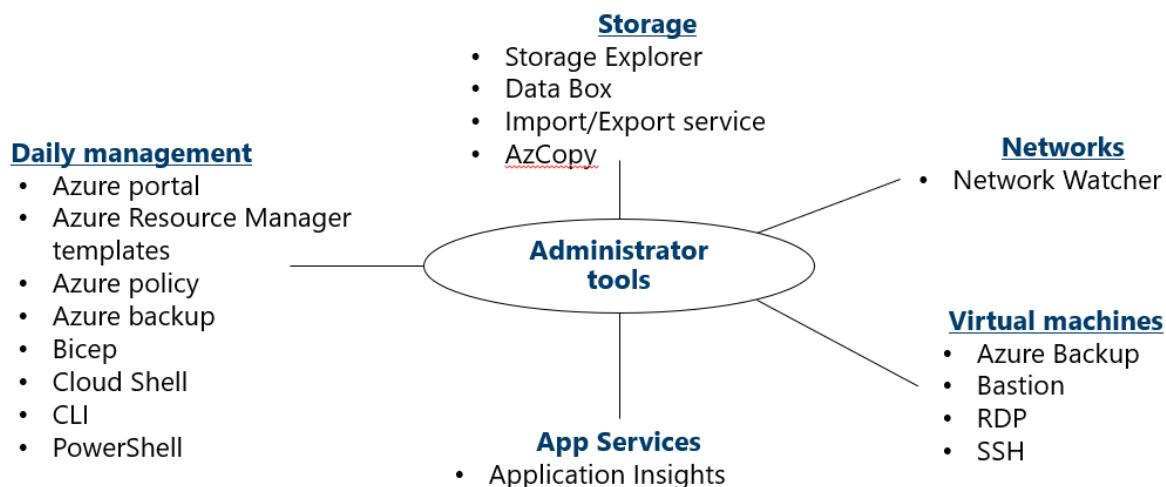
03 Administer Azure Resources

1. Azure Resources with Tools

Azure Resources:

- Azure resources are the core building blocks of Azure services. Examples include virtual machines (VMs), storage accounts, databases, virtual networks, and app services.
- These resources are deployed and managed within Azure subscriptions and grouped logically using **resource groups**.

Managing Resources with Tools



Azure provides several tools to manage and interact with resources:

1. Azure Portal:

- A web-based graphical interface for creating, configuring, and managing Azure resources.
- Features:
 - Drag-and-drop simplicity.
 - Provides visual insights through dashboards and monitoring tools.
 - Easy access to configuration settings for individual resources.

- Use Case:

Ideal for beginners or when performing ad-hoc tasks like deploying a single VM or adjusting resource settings.

2. Azure PowerShell:

- A command-line tool for automating Azure resource management.
- Features:
 - Scriptable for bulk operations.
 - Provides detailed resource configuration via commands (e.g., New-AzVM to create a VM).

- Use Case:

Automating repetitive tasks like deploying multiple resources in different regions.

3. Azure CLI (Command-Line Interface):

- A cross-platform command-line tool for managing Azure resources.
- Features:
 - Works on Linux, macOS, and Windows.
 - Provides commands like `az vm create` for quick resource deployment.

- Use Case:

Preferred for scripting and automation in DevOps environments.

4. Azure Resource Manager (ARM):

- ARM is the management layer in Azure that allows you to deploy, manage, and organize resources through a unified API.
- Tools like the portal, CLI, and PowerShell interact with Azure resources via ARM.

5. Azure Mobile App:

- A mobile application for managing resources on the go.
- Features:
 - Monitor performance.
 - Restart services or check resource health.
- Use Case:

Quick checks or actions during on-call troubleshooting.

6. Azure Monitor:

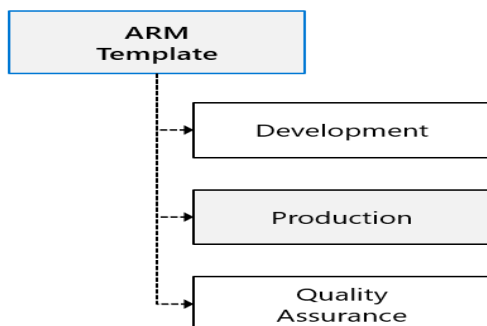
- A service that provides insights into the performance and health of Azure resources.
- Features:
 - Collects metrics and logs.
 - Helps set up alerts to notify administrators of issues.
- Use Case:

Troubleshooting resource performance issues or tracking usage trends.

2. Resources with ARM Templates

What are ARM Templates?

- **Azure Resource Manager (ARM) Templates** are JSON files that define the infrastructure and configuration for Azure resources.
- They provide a **declarative syntax**, meaning you specify what you want to deploy, and Azure handles the actual deployment process.



Key Features of ARM Templates:

1. Declarative Deployment:

- Specify the desired state of resources without detailing step-by-step actions.
- Example: Define that you need a VM with a specific size and OS, and ARM ensures it is created as specified.

2. Reusable Infrastructure Code:

Templates can be reused and shared to ensure consistent deployment across environments (e.g., Development, Testing, Production).

3. Idempotent Operations:

Running the same template multiple times produces the same result, avoiding duplication or misconfiguration.

4. Support for Complex Deployments:

ARM Templates allow you to define dependencies between resources (e.g., a VM depends on a storage account) to ensure proper deployment order.

5. Integration with Other Tools:

ARM Templates can be deployed through Azure CLI, PowerShell, or CI/CD pipelines (e.g., using Azure DevOps).

Structure of an ARM Template:

An ARM Template has the following sections:

- **\$schema**: The location of the JSON schema file for validation.
- **contentVersion**: Version of the template.
- **parameters**: Inputs required for the template, such as resource names or sizes.
- **variables**: Reusable values within the template.
- **resources**: The list of Azure resources to be deployed.
- **outputs**: Values to return after the deployment (e.g., resource names, connection strings).

Deployment Methods for ARM Templates:

1. **Azure Portal:** Upload the template in the "Deploy a Custom Template" section.
2. **Azure CLI:** Use the az deployment group create command to deploy a template.
3. **Azure PowerShell:** Use the New-AzResourceGroupDeployment command for deployment.
4. **CI/CD Pipelines:** Automate deployments through Azure DevOps or GitHub Actions.

Benefits of Using ARM Templates:

- Ensures consistent resource deployments.
- Enables infrastructure as code (IaC), improving collaboration and version control.
- Reduces manual errors during resource creation.
- Simplifies complex deployments by managing dependencies.

Example ARM Template:

```
{
  "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "vmName": {
      "type": "string",
      "defaultValue": "myVM"
    }
  },
  "resources": [
    {
      "type": "Microsoft.Compute/virtualMachines",
      "apiVersion": "2021-03-01",
      "name": "[parameters('vmName')]"
    }
  ]
}
```

```

"location": "East US",

"properties": {
  "hardwareProfile": {
    "vmSize": "Standard_DS1_v2"
  },
  "osProfile": {
    "computerName": "[parameters('vmName')]",
    "adminUsername": "azureuser",
    "adminPassword": "P@ssw0rd!"
  }
}
]
}

```

Use Case for ARM Templates:

A company uses an ARM Template to deploy an entire application environment, including VMs, databases, storage accounts, and networking components. This ensures that every deployment is identical, reducing errors and saving time.

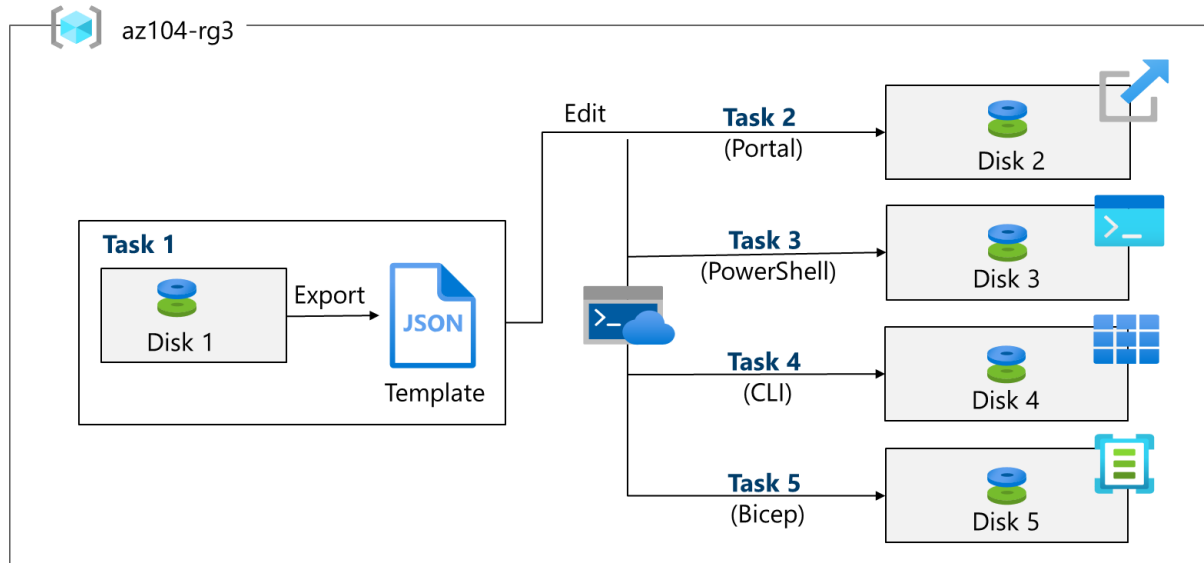
How These Concepts Work Together

1. **Tools like Azure CLI, PowerShell, and the Azure Portal** are used to manage individual resources or trigger ARM Template deployments.
2. **ARM Templates** provide a way to codify and automate resource deployments for consistency and efficiency.

Practical Example:

An IT admin uses the Azure Portal to monitor the status of resources but relies on ARM Templates deployed through Azure CLI to provision infrastructure for new projects.

Interactive lab simulations



1. [Manage Azure resources by using Azure Resource Manager templates.](#) ← *Ctrl+Left*
Click to start the lab
Review, create, and deploy a managed disks with a template.
2. [Create a virtual machine with a template.](#) ← *Ctrl+Left* *Click to start the lab*
Deploy a virtual machine with a QuickStart template.

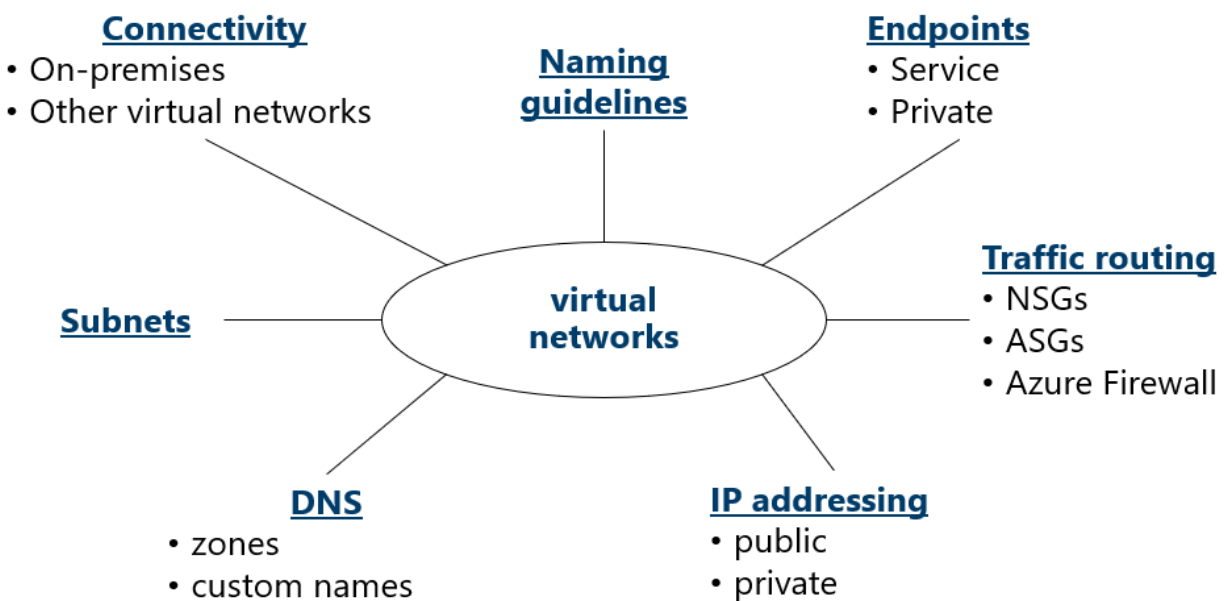
04 Administer Virtual Networking

1. Virtual Networks (VNet)

Definition:

Azure Virtual Network (VNet) is a foundational service in Azure that allows you to create a logically isolated network for hosting your resources, such as virtual machines, app services, and databases.

Virtual networking whiteboard:



Key Features:

1. Isolation and Security:

VNets provide complete network isolation, ensuring no overlap with other networks unless explicitly connected.

2. Customizable IP Addressing:

You can define your own private IP address space using CIDR (e.g., 10.0.0.0/16).

3. Subnets:

A VNet can be divided into smaller segments called subnets, each with its own IP address range. Subnets are used to isolate and organize resources.

4. Connectivity Options:

- **Internet Connectivity:** Resources within a VNet can connect to the internet using public IPs or NAT Gateway.
- **On-Premises Connectivity:** Use VPN Gateway or ExpressRoute to connect VNets to on-premises networks.
- **VNet Peering:** Enables two VNets to communicate privately over Azure's backbone network.

5. Integration with Azure Services:

VNets integrate with many Azure services, such as Azure App Service and Azure Kubernetes Service (AKS).

Use Cases:

- Hosting multi-tier applications by isolating web, application, and database layers into different subnets.
- Extending on-premises networks to the cloud using a hybrid connectivity solution.

Example:

A company creates a VNet with the address space 10.1.0.0/16 and divides it into two subnets:

- Subnet A: 10.1.1.0/24 for web servers.
 - Subnet B: 10.1.2.0/24 for database servers.
- Traffic between the subnets is controlled using Network Security Groups.

2. Network Security Groups (NSG)

Definition:

Network Security Groups are a set of rules that control inbound and outbound traffic to Azure resources in a VNet. They act as virtual firewalls for Azure resources.

Key Features:

1. Rule Types:

- **Inbound Rules:** Control incoming traffic to resources.
- **Outbound Rules:** Control outgoing traffic from resources.

Inbound security rules

| Priority | Name | Port | Protocol | Source | Destination | Action |
|----------|-------------------------------|------|----------|-------------------|----------------|---------|
| 100 | ⚠ RDP_Inbound | 3389 | Any | Any | Any | ✔ Allow |
| 65000 | AllowVnetInBound | Any | Any | VirtualNetwork | VirtualNetwork | ✔ Allow |
| 65001 | AllowAzureLoadBalancerInBound | Any | Any | AzureLoadBalancer | Any | ✔ Allow |
| 65500 | DenyAllInBound | Any | Any | Any | Any | ✖ Deny |

Outbound security rules

| Priority | Name | Port | Protocol | Source | Destination | Action |
|----------|-----------------------|------|----------|----------------|----------------|---------|
| 65000 | AllowVnetOutBound | Any | Any | VirtualNetwork | VirtualNetwork | ✔ Allow |
| 65001 | AllowInternetOutBound | Any | Any | Any | Internet | ✔ Allow |
| 65500 | DenyAllOutBound | Any | Any | Any | Any | ✖ Deny |

2. Rule Components:

- **Priority:** A number that determines the order in which rules are applied (lower numbers have higher priority).
- **Source/Destination:** Specifies the origin or destination of traffic (e.g., IP address, range, or *).
- **Protocol:** Defines the traffic type (e.g., TCP, UDP, or Any).
- **Action:** Either **Allow** or **Deny** the traffic.

3. Associations:

- NSGs can be associated with individual network interfaces (NICs) or subnets.
- When applied at both levels, NIC-level rules override subnet-level rules.

Best Practices:

- Use specific IP ranges to limit access (avoid using * unless necessary).
- Prioritize deny rules over allow rules to prevent accidental exposure.
- Regularly review and update NSG rules to maintain security compliance.

Use Cases:

- Protecting a database subnet by allowing only traffic from the web server subnet on port 1433 (SQL).
- Blocking all outbound traffic except to specific IPs or domains.

Example:

An NSG is created for a web server with the following rules:

- Allow inbound traffic on port 80 and 443 from the internet.
- Deny inbound traffic from all other sources.
- Allow outbound traffic on port 53 for DNS and 443 for HTTPS.

3. Azure DNS**Definition:**

Azure DNS is a cloud-based service for hosting and managing domain name system (DNS) records. It provides reliable and secure name resolution for Azure resources and external domains.

Create DNS zone

×

Basics

Tags

Review + create

A DNS zone is used to host the DNS records for a particular domain. For example, the domain 'contoso.com' may contain a number of DNS records such as 'mail.contoso.com' (for a mail server) and 'www.contoso.com' (for a web site). Azure DNS allows you to host your DNS zone and manage your DNS records, and provides name servers that will respond to DNS queries from end users with the DNS records that you create. [Learn more.](#)

Project details

Subscription *

MSDN Platforms Subscription

▼

Resource group *

rg-dns

▼

[Create new](#)

Instance details

Name *

azureadmininc.org

✓

Resource group location ⓘ

East US

▼

Review + create

Previous

Next : Tags >

[Download a template for automation](#)

Key Features:**1. DNS Zones:**

A DNS zone represents a domain (e.g., contoso.com) and contains records to map names to IP addresses or other endpoints.

2. DNS Record Types:

- **A Record:** Maps a domain name to an IPv4 address.
- **AAAA Record:** Maps a domain name to an IPv6 address.
- **CNAME Record:** Maps an alias name to a canonical domain name.
- **MX Record:** Specifies mail servers for the domain.
- **TXT Record:** Stores arbitrary text data (e.g., for verification or SPF records).

3. Private DNS Zones:

- Enables DNS resolution for private resources within a VNet without exposing them to the public internet.
- Example: Resolve dbserver.internal.contoso.com to a private IP within the VNet.

4. Global Availability:

Azure DNS uses Microsoft's global infrastructure to ensure high availability and low-latency DNS resolution.

Integration with Azure Services:

- Azure DNS can be integrated with VNets for name resolution of private resources.
- It supports automatic DNS record updates for services like Azure App Service or AKS.

Use Cases:

- Host and manage custom domains for public-facing applications.
- Enable seamless internal name resolution for resources in a VNet.

Example:

A company owns the domain example.com and uses Azure DNS to host its DNS zone. The DNS zone contains the following records:

- `www.example.com` → Points to the public IP of a web server using an A record.
- `db.example.com` → Points to a private IP in a VNet using a private DNS zone.

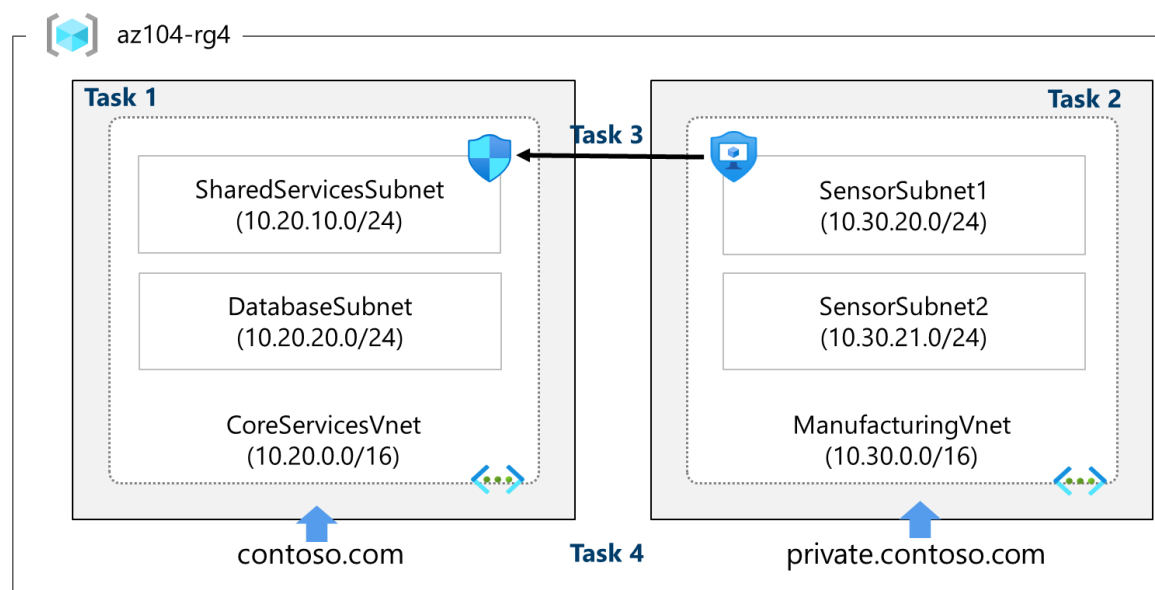
How These Concepts Work Together

1. **Virtual Networks (VNets)** provide the foundation for isolating and connecting Azure resources.
2. **Network Security Groups (NSGs)** secure the resources within a VNet by filtering traffic based on defined rules.
3. **Azure DNS** facilitates name resolution for resources inside and outside of Azure, ensuring they are accessible via user-friendly domain names rather than IP addresses.

Practical Example:

- A company deploys a three-tier web application:
 - VNets are used to isolate the web, app, and database layers.
 - NSGs are applied to restrict access between tiers, only allowing necessary traffic (e.g., HTTP/HTTPS to web servers, SQL to databases).
 - Azure DNS is used to create friendly domain names like `web.example.com` for public access and `db.internal.example.com` for private access within the VNet.

Interactive lab simulations



- 1) [Secure network traffic](#). ⬅ *Ctrl+Left Click to start the lab*

Create a virtual machine, a virtual network, and a network security group. Add network security group rules to allow and disallow traffic.

- 2) [Create a simple virtual network](#). ⬅ *Ctrl+Left Click to start the lab*

Create a virtual network with two virtual machines. Demonstrate the virtual machines can communicate.

- 3) [Design and implement a virtual network in Azure](#). ⬅ *Ctrl+Left Click to start the lab*

Create a resource group and create virtual networks with subnets.

- 4) [Implement virtual networking](#). ⬅ *Ctrl+Left Click to start the lab*

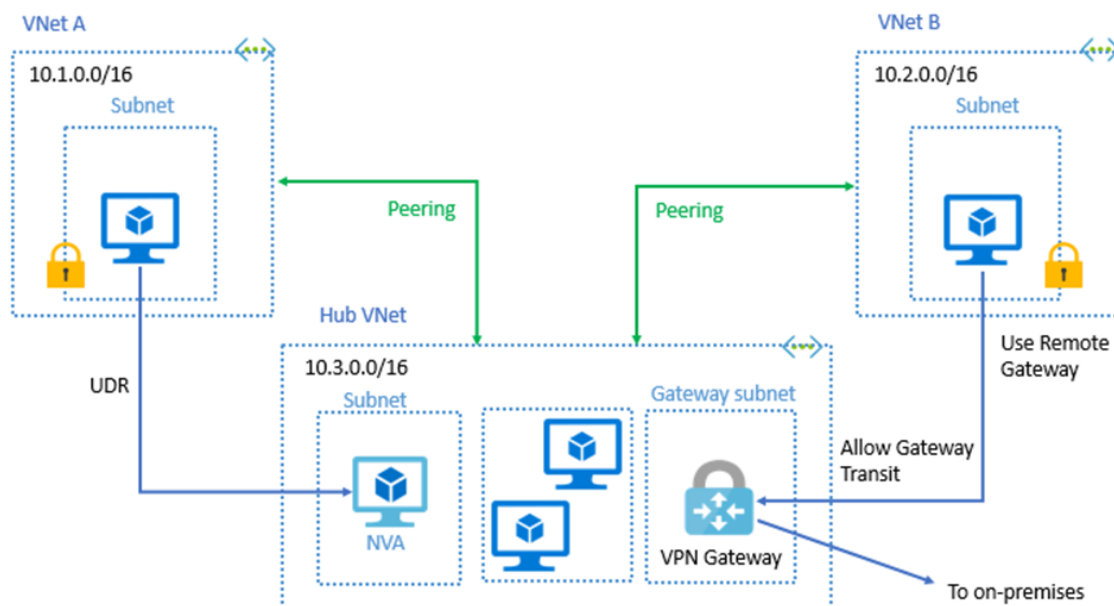
Create and configure a virtual network, deploy virtual machines, configure network security groups, and configure Azure DNS.

05 Administer Intersite Connectivity

1. VNet Peering

Definition:

VNet Peering enables you to connect two or more Azure Virtual Networks (VNets) privately through the Azure backbone network. This allows resources in one VNet to communicate with resources in another VNet as if they were in the same network.



Key Features:

1. Low Latency and High Bandwidth:

Communication between peered VNets uses Azure's backbone, ensuring minimal latency and no performance degradation.

2. Full Connectivity:

Resources in both VNets can communicate bidirectionally, assuming proper permissions and configurations are in place.

3. Support Across Subscriptions and Regions:

- **Intra-Region Peering:** Peering VNets within the same Azure region.
- **Global VNet Peering:** Peering VNets across different Azure regions.

4. No Overlapping Address Spaces:

The IP address ranges of the VNets must not overlap.

Peering Configuration Options:

1. Traffic Forwarding:

Allows one VNet to forward traffic to another (e.g., for hub-and-spoke network architectures).

2. Gateway Sharing:

Enables one VNet to use the VPN Gateway of another peered VNet to connect to on-premises networks.

Limitations:

VNet Peering does not support transitive routing by default (e.g., VNet A connected to VNet B cannot automatically communicate with VNet C if B is also peered to C). This must be explicitly enabled with user-defined routes (UDRs).

Use Cases:

- Connecting development and production VNets for resource sharing.
- Building hub-and-spoke architectures, where a central VNet (hub) provides shared services like firewalls, gateways, or DNS, and other VNets (spokes) connect to the hub.

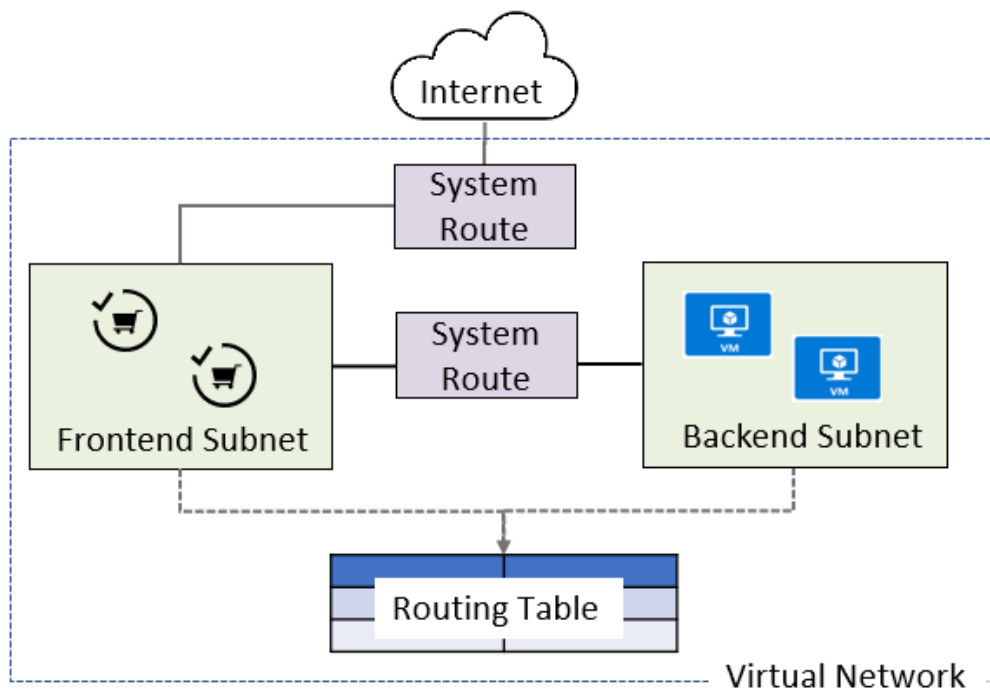
Example:

- **Scenario:** A company has two VNets:
 - VNet A (10.1.0.0/16) in Region 1 for a web application.
 - VNet B (10.2.0.0/16) in Region 2 for a database.
- **Solution:** Configure Global VNet Peering to allow the web application to connect to the database securely without exposing either VNet to the internet.

2. Network Routing and Endpoints

Definition:

Azure manages traffic flow between resources using routing rules and network endpoints. Routing determines how packets travel within VNets, between VNets, or to external destinations (like the internet). Endpoints define the targets for communication.



A. Network Routing

Routing Types in Azure:

1. System Routes:

- Automatically created by Azure to direct traffic within VNets or to external networks. Examples include:
 - Traffic within a VNet.
 - Traffic between peered VNets.
 - Traffic to the internet.

2. User-Defined Routes (UDRs):

- Custom routing rules created by administrators to override system routes.
- Use Cases:
 - Directing traffic through a Network Virtual Appliance (NVA) for inspection.
 - Enabling transitive routing between peered VNets.

3. BGP Routes:

- Routes propagated by on-premises networks via a VPN Gateway or ExpressRoute.
- Use Case:

Ensures seamless integration of Azure and on-premises networks.

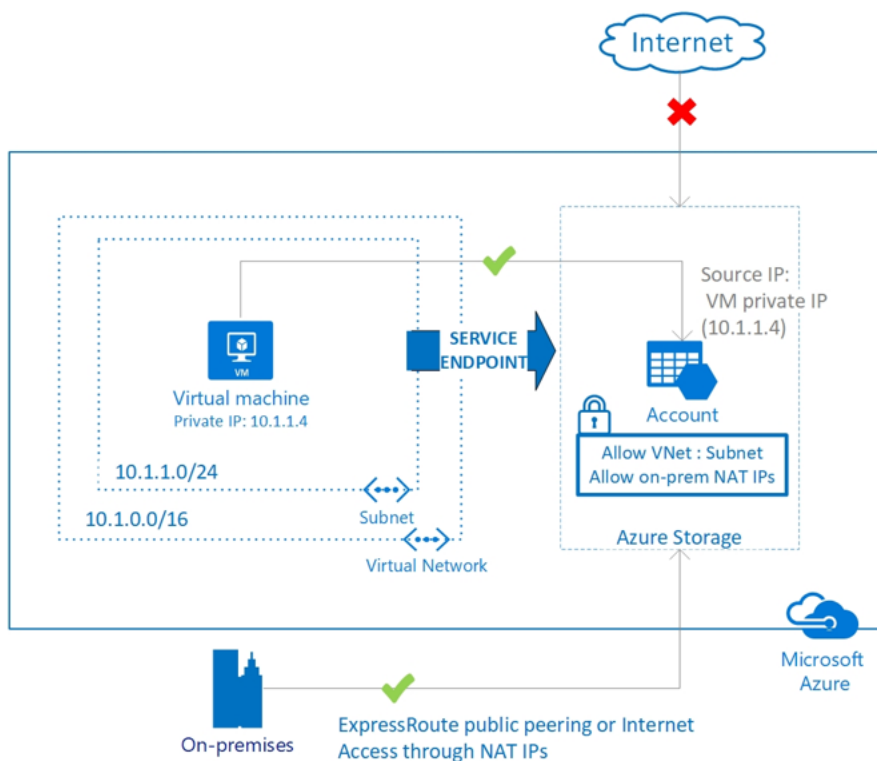
Key Concepts in Routing:

- **Next Hop:** Defines where traffic should go next. Options include:
 - Virtual network.
 - Virtual appliance.
 - Internet.
- **Route Tables:** Collections of routes that can be associated with subnets to define traffic behavior.

Example:

- **Scenario:** An organization wants to inspect all outbound traffic from a subnet through a firewall.
- **Solution:** Create a UDR with a next hop pointing to the firewall's IP address and associate the route table with the subnet.

B. Endpoints



Private Endpoints:

- **Definition:** Private Endpoints allow Azure resources (e.g., Azure Storage, SQL Database) to be accessed privately over a VNet without exposing them to the internet.
- **Key Features:**
 - Provides a private IP address within a VNet for the resource.
 - Secures traffic by eliminating public exposure.
- **Use Cases:**
 - Secure access to sensitive resources like databases or storage accounts.

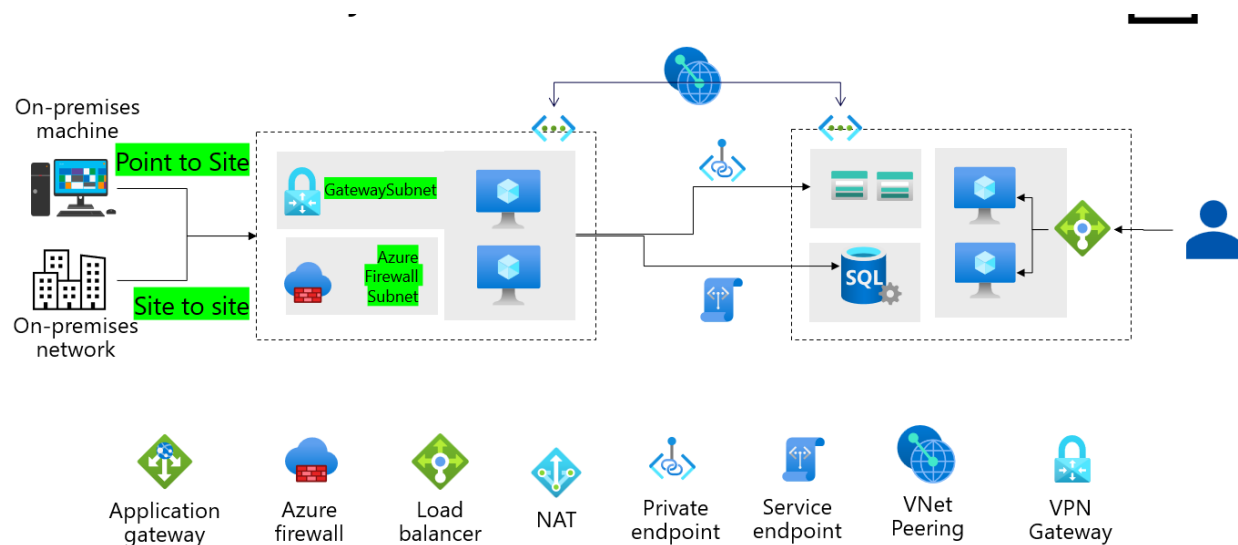
Public Endpoints:

- **Definition:** Public Endpoints expose resources to the internet but can be secured with network controls (e.g., firewalls or NSGs).
- **Use Cases:**
 - Allowing global access to web applications or APIs.

Service Endpoints:

- **Definition:** Service Endpoints allow VNet resources to access Azure services (e.g., Azure SQL, Azure Storage) over the Azure backbone instead of the public internet.
- **Key Features:**
 - Simplifies securing Azure services by restricting access to specific VNets.
- **Use Cases:**
 - Connecting VMs in a VNet to an Azure SQL database without a public IP.

How These Concepts Work Together



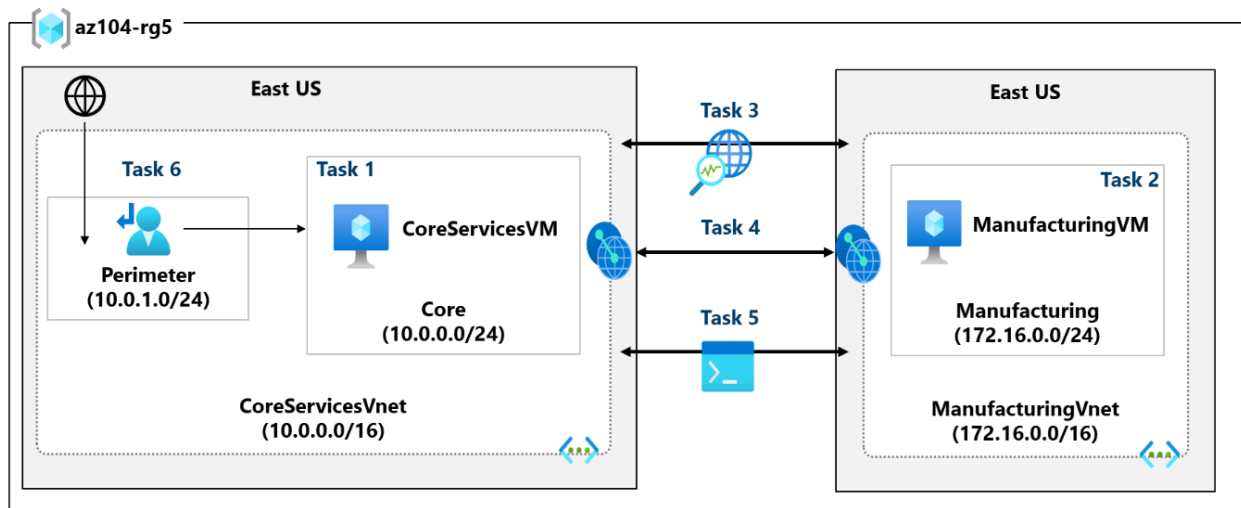
1. **VNet Peering** connects VNets to enable resource communication.
2. **Routing** ensures that traffic flows correctly between VNets, on-premises networks, and the internet.
3. **Endpoints** define secure and efficient access points for Azure services and external networks.

Practical Example:

- A company hosts a three-tier application:
 - **Frontend** in VNet A.
 - **Application layer** in VNet B.
 - **Database** in Azure SQL with a private endpoint.

- Solution: VNet Peering connects VNet A and VNet B. Routing ensures all traffic flows through an NVA for inspection. The database is secured with a private endpoint to prevent public exposure.

Interactive lab simulations



- [Connect two Azure virtual networks using global virtual network peering.](#)

← Ctrl+Left Click to start the lab

Test the connection between two virtual machines in different virtual networks. Create a virtual network peering and retest.

- [Configure monitoring for virtual networks.](#) ← Ctrl+Left Click to start the lab

Understand how to use Azure Network Watcher Connection Monitor, flow logs, NSG diagnostics, and packet capture to monitor connectivity across your Azure IaaS network resources.

- [Implement intersite connectivity.](#) ← Ctrl+Left Click to start the lab

Run a template to create a virtual network infrastructure with several virtual machines. Configure virtual network peerings and test the connections.

06 Administer Network Traffic

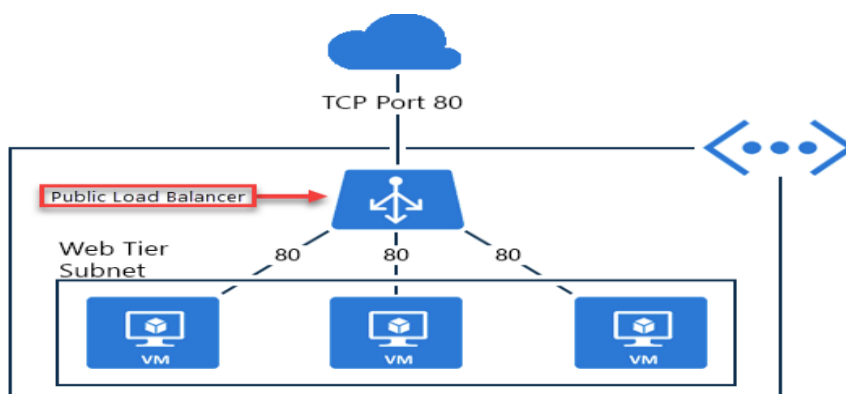
1. Azure Load Balancer

Definition:

Azure Load Balancer is a Layer 4 (TCP/UDP) traffic distribution service that balances incoming traffic across multiple virtual machines (VMs) or instances in Azure to ensure high availability and reliability.

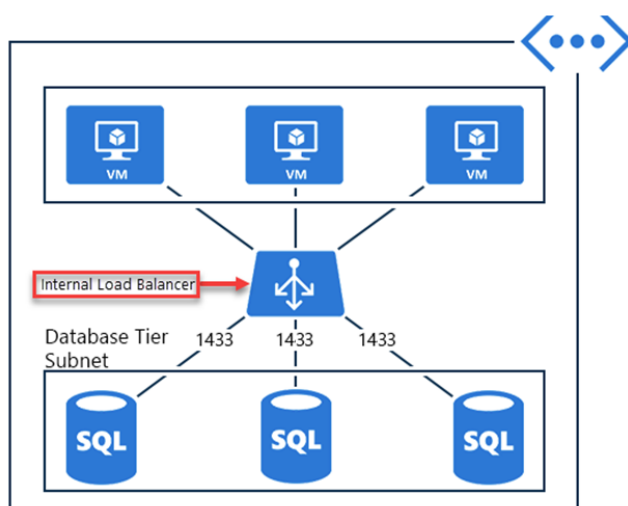
Types of Load Balancers:

1. Public Load Balancer:



- Balances traffic coming from the internet to Azure VMs or services with a public IP address.
- Use Case: Distribute web traffic from internet users to a pool of web servers.

2. Internal Load Balancer:



- Balances traffic within a virtual network (VNet), used for internal applications or services.
- Use Case: Distribute traffic between app servers in the backend tier of a multi-tier application.

Key Features:**1. Health Probes:**

Periodically check the health of backend instances and route traffic only to healthy instances.

2. Outbound Rules:

Manage outbound internet traffic for backend resources using SNAT (Source Network Address Translation).

3. Session Persistence:

Supports maintaining client connections to the same backend server during a session using **Hash-based distribution**.

4. High Availability:

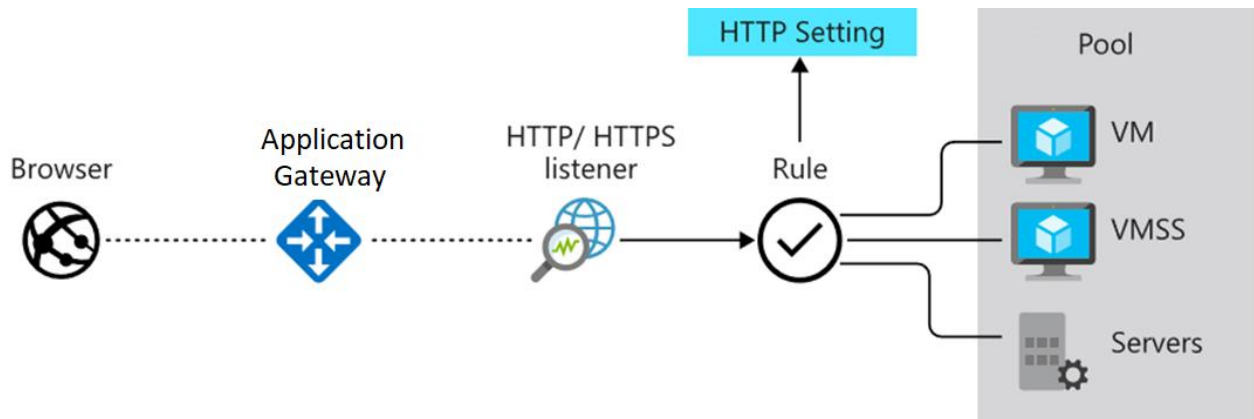
Integrated into Azure's backbone network, providing low latency and high throughput.

Example:

- A public Azure Load Balancer is configured with a frontend IP address to distribute HTTP traffic (port 80) across three VMs in a backend pool. A health probe monitors the status of these VMs on port 80 to ensure only healthy VMs receive traffic.

2. Application Gateway**Definition:**

- Azure Application Gateway is a Layer 7 load balancer designed specifically for web traffic (HTTP/HTTPS). It offers advanced routing and security features to handle complex traffic patterns.



Key Features:

1. URL-Based Routing:

- Routes traffic to different backend pools based on URL paths.
- Example: Requests to `www.contoso.com/images` can route to one backend pool, while requests to `www.contoso.com/videos` route to another.

2. Web Application Firewall (WAF):

- Protects applications from common vulnerabilities such as SQL injection, cross-site scripting (XSS), and more.

3. SSL Termination:

- Decrypts HTTPS traffic at the gateway level, reducing the workload on backend servers.

4. Redirection:

- Supports HTTP to HTTPS redirection for secure communication.

5. Autoscaling:

- Automatically adjusts the number of Application Gateway instances to handle traffic loads efficiently.

6. Session Affinity:

- Sticky sessions allow all requests from a client session to be directed to the same backend instance.

Use Cases:

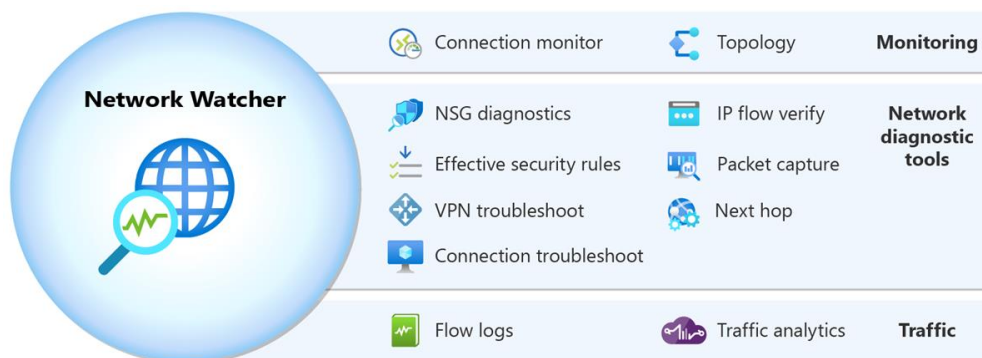
- Hosting multiple websites on the same application gateway.
- Securing applications with a Web Application Firewall.
- Redirecting specific requests to microservices or APIs.

Example:

- A retail company uses Application Gateway to:
 - Route traffic for `www.shop.com/products` to a product catalog backend.
 - Route traffic for `www.shop.com/orders` to an order management backend.
 - Protect both backends with a WAF policy.

3. Network Watcher**Definition:**

Network Watcher is a network monitoring and diagnostic service in Azure that provides tools to monitor, diagnose, and visualize network performance and connectivity.

**Key Features:****1. Connection Troubleshooting:**

- Tests connectivity between a source and destination to identify issues.
- Example: Check if a VM can connect to an Azure SQL database.

2. Packet Capture:

- Captures network traffic at the VM level for analysis. Useful for diagnosing application-level issues or suspicious activity.

3. **Topology Viewer:**

- Provides a visual representation of your network architecture, showing the relationships between VNets, subnets, and resources.

4. **Next Hop:**

- Identifies the next hop for a given route, helping troubleshoot routing issues.

5. **IP Flow Verify:**

- Simulates the flow of traffic between two points to check if NSGs or UDRs are blocking traffic.

6. **Metrics and Logs:**

- Integrates with Azure Monitor to provide insights into network performance and activity.

Use Cases:

- Troubleshooting connectivity issues between VNets, subnets, or resources.
- Verifying security rules (NSGs) and user-defined routes.
- Capturing packets for detailed analysis in security investigations.

Example:

- A company experiences connectivity issues between a web server and a database.
 - Use **Connection Troubleshooting** to test the connection.
 - Use **IP Flow Verify** to ensure no NSG or UDR is blocking traffic.
 - If needed, run a **Packet Capture** to analyze the issue further.

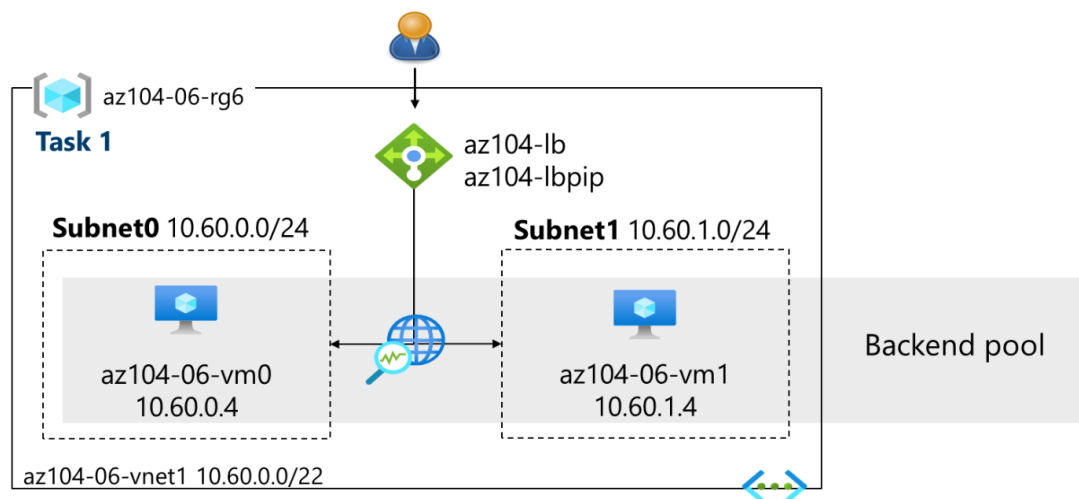
How These Concepts Work Together

1. **Azure Load Balancer** handles high-performance distribution of Layer 4 traffic (e.g., TCP/UDP).
2. **Application Gateway** provides advanced Layer 7 load balancing with features like WAF and URL-based routing for web applications.
3. **Network Watcher** ensures smooth operation of these services by providing monitoring and troubleshooting tools to identify and resolve issues.

Practical Scenario:

- A company deploys a multi-tier web application:
 - A **Public Azure Load Balancer** distributes incoming traffic across multiple application gateways.
 - Each **Application Gateway** routes traffic to backend web servers and protects them with WAF.
 - **Network Watcher** monitors traffic flow, identifies routing problems, and captures network logs for troubleshooting.

Interactive lab simulations



- [Create and configure and Azure load balancer.](#) ← *Ctrl+Left Click to start the lab*

Create a virtual network, backend servers, load balancer, and then test the load balancer.

- [Deploy Azure Application Gateway.](#) ← *Ctrl+Left Click to start the lab*

Create an application gateway, create virtual machines, create the backend pool, and test the gateway.

- [Implement traffic management.](#) ← *Ctrl+Left Click to start the lab*

Implement complete hub and spoke network including virtual machines, virtual networks, peering, load balancer, and application gateway.

07 Administer Azure Storage

1. Storage Accounts

Definition:

- An Azure Storage Account is a container that provides access to various Azure storage services, including blobs, files, queues, and tables. It acts as a gateway to Azure's cloud storage capabilities.

Key Features:

1. Types of Storage Accounts:

- **General Purpose v2 (GPv2):** Supports blobs, files, queues, and tables with all modern features.
- **Blob Storage:** Optimized for unstructured data like images, videos, and backups.
- **FileStorage:** Specifically for Azure Files with high-performance requirements.

2. Replication Options:

- **LRS (Locally Redundant Storage):** Replicates data three times within a single data center.
- **GRS (Geo-Redundant Storage):** Replicates data to a secondary region, providing high availability during regional failures.
- **ZRS (Zone-Redundant Storage):** Replicates data across availability zones within a region.
- **RA-GRS (Read-Access Geo-Redundant Storage):** Allows read-only access to the secondary region.

3. Performance Tiers:

- **Standard Tier:** Cost-effective, suitable for infrequent access.
- **Premium Tier:** High performance for low-latency applications.

4. Access Tiers for Blob Storage:

- **Hot Tier:** Frequent access.
- **Cool Tier:** Infrequent access.
- **Archive Tier:** Long-term storage with low access frequency.

Use Cases:

- Hosting website assets like images and videos.
- Storing backups or logs for long-term retention.
- Supporting scalable cloud-based applications.

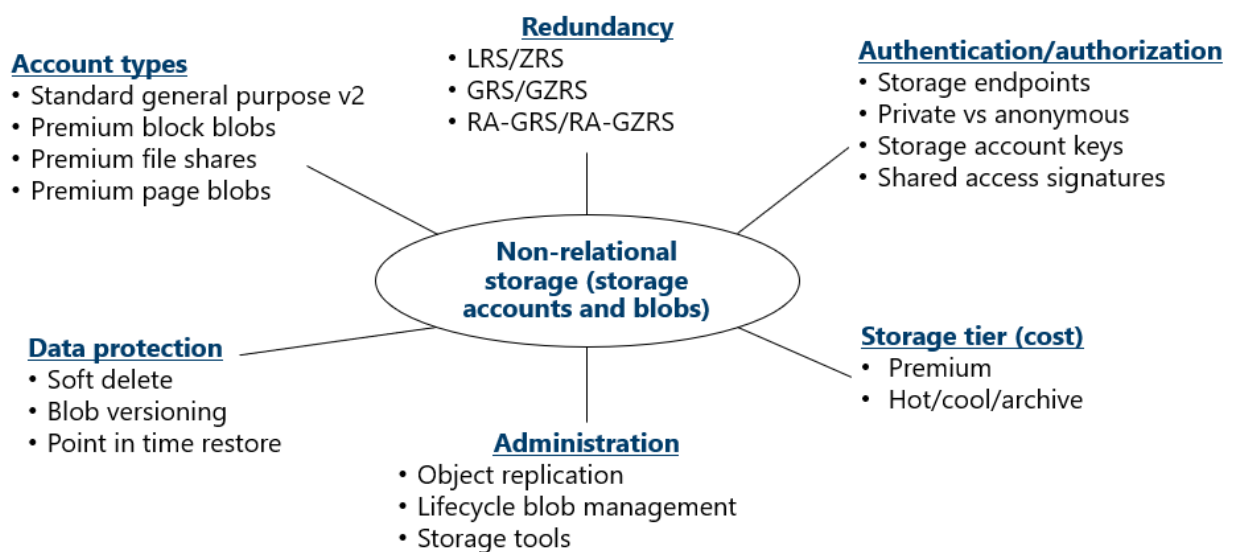
Example:

A retail company creates a GPv2 storage account to store customer images in blob storage and order logs in table storage, ensuring both are securely stored with GRS for redundancy.

2. Blob Storage

Definition:

Azure Blob Storage is a service for storing large amounts of unstructured data, such as text or binary data. It is commonly used for documents, media files, and application data.



Key Features:**1. Blob Types:**

- **Block Blobs:** Used for text and binary data (e.g., files, images, videos).
- **Append Blobs:** Optimized for appending operations, suitable for logging.
- **Page Blobs:** Used for random read/write operations, such as VM disks.

2. Blob Lifecycle Management:

- Automatically transitions blobs between access tiers or deletes them based on rules (e.g., move infrequently accessed blobs to the cool tier).

3. Access Methods:

- Access blobs using REST APIs, SDKs, or tools like Azure Storage Explorer.

4. Blob Versioning:

- Maintains older versions of blobs for recovery or compliance purposes.

5. Immutable Blob Storage:

- Enables Write-Once-Read-Many (WORM) policies to protect against deletion or modification.

Use Cases:

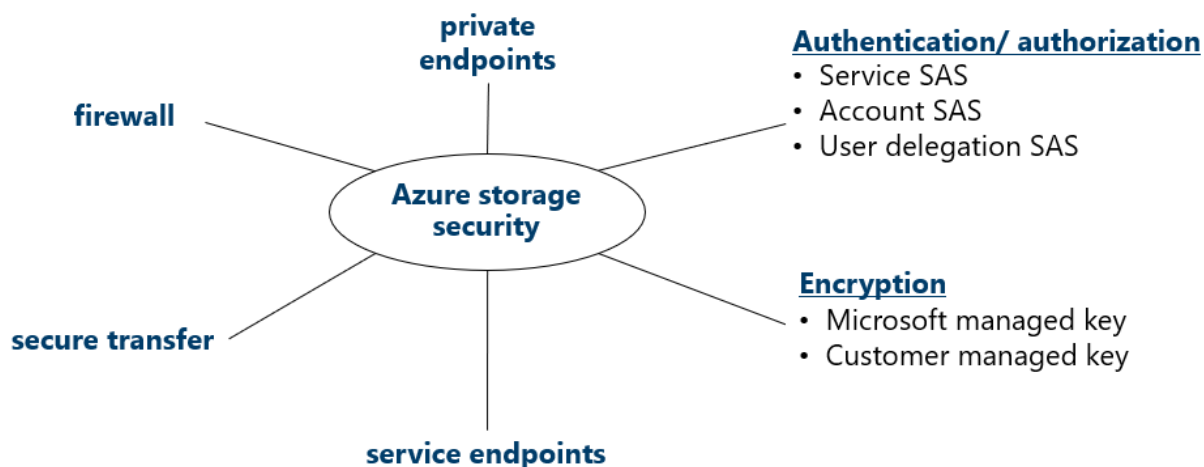
- Hosting large datasets for analytics and machine learning.
- Storing application logs and streaming media files.
- Maintaining data backups and disaster recovery files.

Example:

- A video streaming service stores movie files in block blobs, applies lifecycle policies to archive less popular files, and uses a content delivery network (CDN) for faster access to viewers.

3. Storage Security**Definition:**

Azure Storage Security ensures the confidentiality, integrity, and availability of data in storage accounts through various security features and tools.



Key Features:

1. Encryption:

- **At-Rest Encryption:** All data in Azure Storage is encrypted using Microsoft-managed keys by default. Customers can also use customer-managed keys with Azure Key Vault.
- **In-Transit Encryption:** Data is secured using HTTPS during transfer.

2. Access Control:

- **Shared Access Signatures (SAS):** Provides granular, time-limited access to storage resources without sharing account keys.
- **Role-Based Access Control (RBAC):** Uses Azure Active Directory (AAD) to assign roles to users and applications for resource access.

3. Private Endpoints:

- Enables secure access to storage accounts over a private VNet, eliminating exposure to the public internet.

4. Firewalls and Virtual Networks:

- Restrict access to storage accounts based on IP address or specific VNets.

5. Advanced Threat Protection (ATP):

- Detects unusual activities that may indicate a security threat, such as ransomware or unauthorized access.

Use Cases:

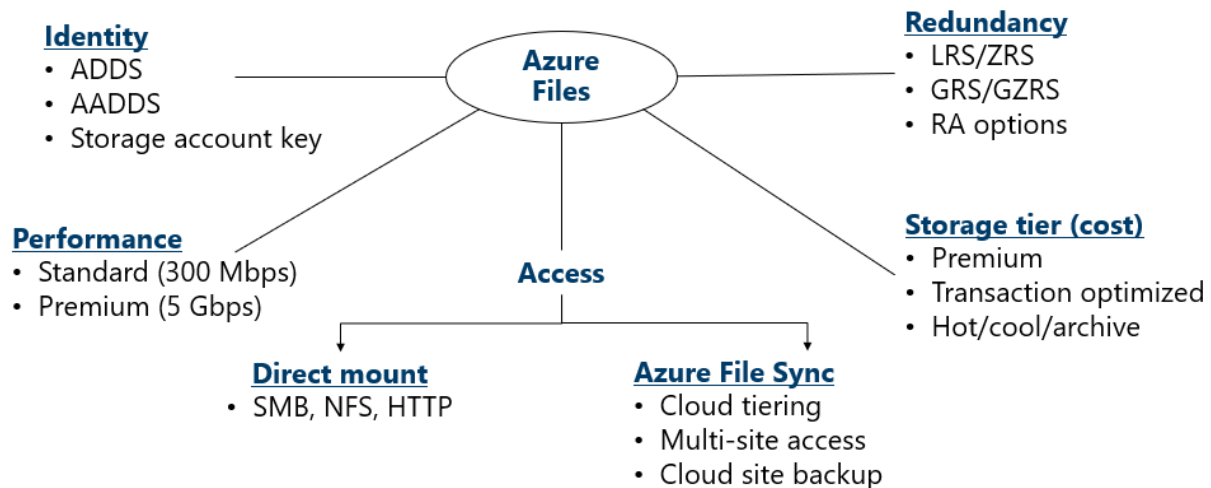
- Ensuring compliance with data protection regulations like GDPR or HIPAA.
- Limiting access to critical resources to specific internal applications.

Example:

- A financial company stores transaction logs in Azure Blob Storage, restricts access via private endpoints, and enables ATP to detect unauthorized access attempts.

4. Azure Files**Definition:**

Azure Files provides fully managed file shares in the cloud that can be accessed via the SMB (Server Message Block) or NFS (Network File System) protocols. These shares can be mounted on VMs or accessed from on-premises systems.

**Key Features:****1. SMB and NFS Support:**

- SMB shares are compatible with Windows, macOS, and Linux.
- NFS shares are used for Unix/Linux-based workloads.

2. Integration with Azure File Sync:

- Synchronizes files between Azure Files and on-premises servers, providing a hybrid solution.

3. Tiering:

- **Transaction Optimized:** For workloads with frequent access.
- **Hot Tier:** For high-performance applications.
- **Cool Tier:** For infrequent access to file data.

4. Backup and Recovery:

- Azure Backup supports Azure Files, providing point-in-time restore for file shares.

5. Secure Access:

- Supports Azure AD-based authentication for SMB shares.

Use Cases:

- Hosting shared files for distributed applications or VMs.
- Replacing on-premises file servers with a cloud-based solution.
- Providing hybrid file storage for applications that require both local and cloud access.

Example:

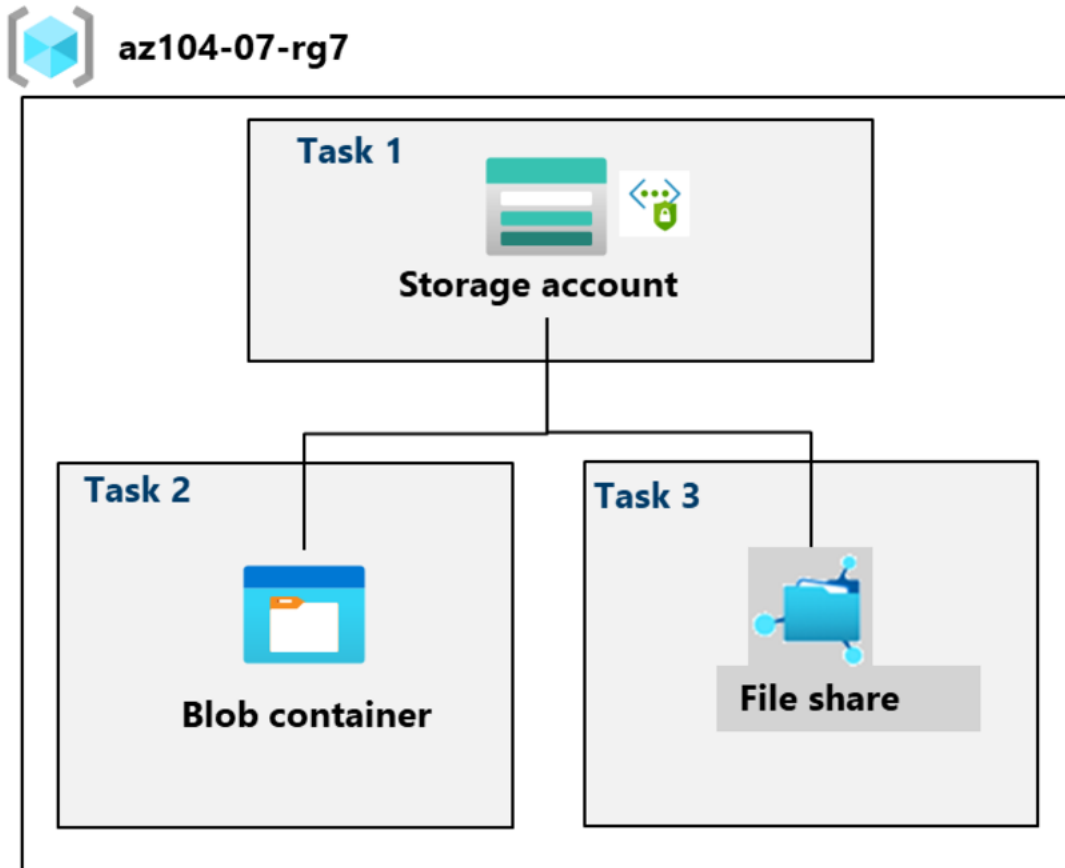
- A software development team stores shared project files on Azure Files, allowing team members to access them from both the office and remote locations via SMB. Azure File Sync ensures copies are cached on an on-premises server for faster local access.

How These Concepts Work Together

1. **Storage Accounts** provide the foundational platform for hosting all types of storage services, including blobs, files, queues, and tables.
2. **Blob Storage** enables scalable and cost-effective storage for unstructured data, supporting scenarios like backups, media hosting, and analytics.
3. **Storage Security** ensures that stored data is protected through encryption, private access, and threat detection mechanisms.
4. **Azure Files** offers a fully managed, cloud-native file sharing solution that integrates seamlessly with existing applications and workflows.

Practical Scenario:

- A hospital stores patient records in Azure Files for shared access across departments, archives older records in Blob Storage using lifecycle policies, and secures all data with encryption, private endpoints, and role-based access.

Interactive lab simulations

- [Create blob storage.](#)

Create a storage account, manage blob storage, and monitor storage activities.

- [Manage Azure storage.](#)

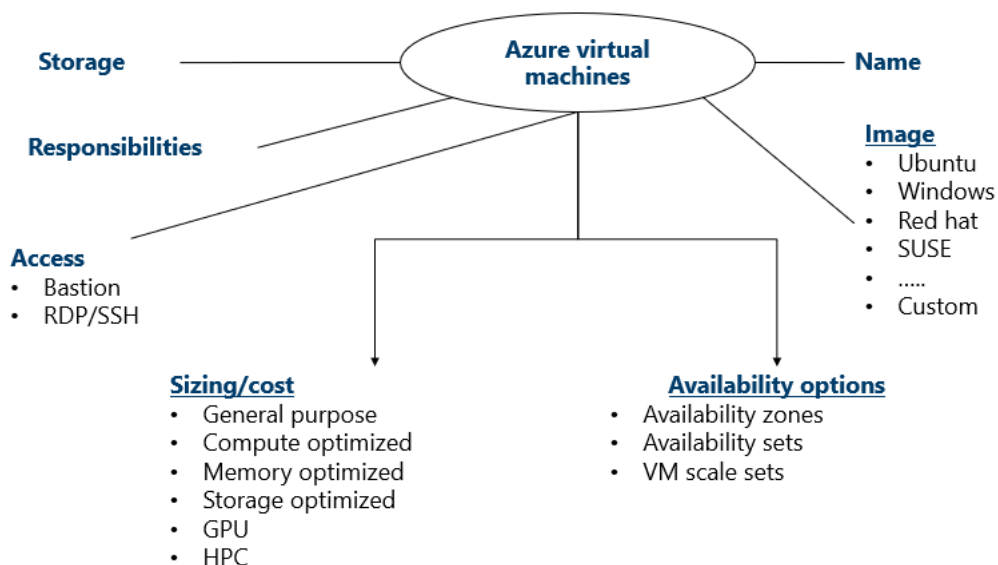
Create a storage account and review the configuration. Manage blob storage containers. Configure storage networking.

08 Administer Azure Virtual Machines

1. Virtual Machines (VMs)

Definition:

Azure Virtual Machines (VMs) provide scalable, on-demand compute resources in the cloud. They emulate a physical computer's functionality and allow you to run various operating systems and applications.



Key Features:

1. VM Sizes:

- Azure offers various VM sizes tailored to specific workloads:
 - **General Purpose:** Balanced CPU-to-memory ratio (e.g., B-series, D-series).
 - **Compute Optimized:** High CPU-to-memory ratio (e.g., F-series).
 - **Memory Optimized:** High memory-to-CPU ratio (e.g., E-series).
 - **Storage Optimized:** High disk throughput and IOPS (e.g., L-series).
 - **GPU VMs:** For graphics-intensive and AI workloads (e.g., NV-series).

2. Operating Systems:

- Supports Windows Server, Linux distributions (e.g., Ubuntu, CentOS), and custom OS images.

3. Disk Types:

- **Standard HDD:** Cost-effective storage for infrequent workloads.
- **Standard SSD:** Moderate performance for general workloads.
- **Premium SSD:** High-performance storage for production environments.
- **Ultra Disk:** Low-latency, high-throughput storage for critical workloads.

4. Networking:

- VMs are connected to virtual networks (VNETs), allowing communication with other Azure resources or on-premises networks.

5. Management Options:

- Use tools like Azure Portal, Azure CLI, PowerShell, and ARM templates for VM deployment and management.

6. Scaling:

- VMs can scale up (more resources per instance) or out (add more instances).

Use Cases:

- Hosting applications, databases, or development environments.
- Running legacy workloads or applications requiring specific OS configurations.
- Supporting scalable compute workloads like batch processing or testing environments.

Example:

- A web hosting company uses Azure VMs running Linux to host customer websites. Each VM is configured with SSD storage and is connected to a virtual network for secure communication.

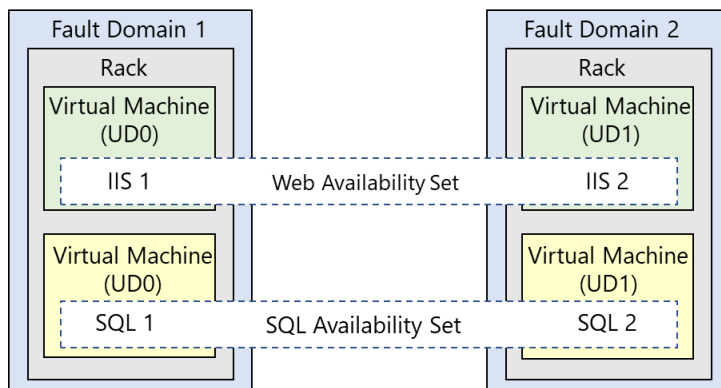
2. Virtual Machine Availability

Definition:

Azure provides features to ensure high availability for Virtual Machines, minimizing downtime and maintaining application reliability.

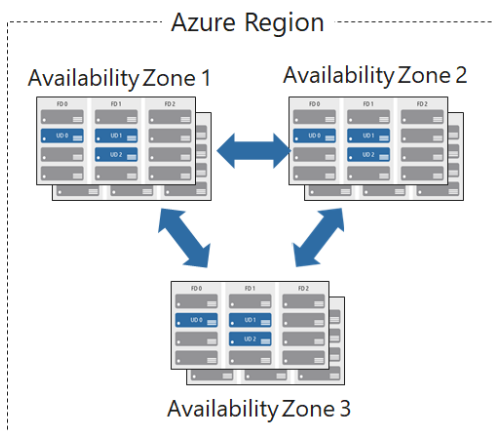
Key Features:

1. Availability Sets:



- Ensures VMs are distributed across multiple fault domains (power/network failures) and update domains (maintenance updates).
- Fault Domain:** A group of VMs sharing the same power source and network switch.
- Update Domain:** A group of VMs that may be rebooted together during maintenance.
- Use Case: Deploy VMs in an availability set to avoid a single point of failure.

2. Availability Zones:



- Physical data center locations within an Azure region.
- VMs placed in different zones are isolated from power, cooling, and network failures.
- Use Case: Use zones for higher fault tolerance compared to availability sets.

3. Azure Load Balancer:

- Distributes traffic across multiple VMs in an availability set or across zones, ensuring that even if one VM fails, the traffic is redirected to healthy instances.

4. VM Scale Sets:

- Automatically deploys and manages a group of identical VMs.
- Enables auto-scaling based on metrics like CPU usage or memory.
- Use Case: Handle fluctuating workloads in a cost-effective manner.

5. Backup and Disaster Recovery:

- Azure Backup: Protects VMs by creating point-in-time snapshots of disks.
- Azure Site Recovery (ASR): Replicates VMs to a secondary region, enabling failover in case of regional outages.

6. Service Level Agreement (SLA):

- Azure offers SLAs to ensure uptime:
 - **99.9%:** Single-instance VMs with premium storage.
 - **99.95%:** VMs in an availability set.
 - **99.99%:** VMs spread across availability zones.

Use Cases:

- Ensuring continuous operation for critical applications like e-commerce websites or enterprise applications.
- Achieving compliance requirements for uptime and disaster recovery.

Example:

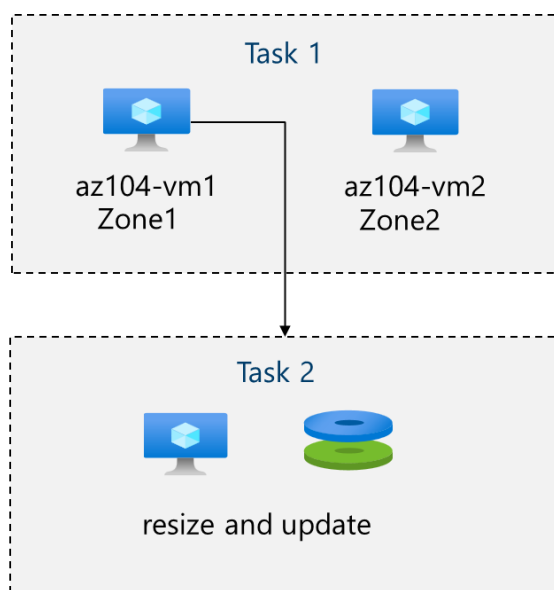
- An online payment gateway deploys its application across three availability zones within a region. A load balancer ensures traffic is routed to active VMs. Azure Backup takes daily snapshots of the VMs, and ASR replicates them to a secondary region for disaster recovery.

How These Concepts Work Together

1. **Azure Virtual Machines** provide the compute resources needed to host and run applications or services.
2. **VM Availability** ensures these VMs remain operational even in the face of failures, maintenance, or high traffic demands.

Practical Scenario:

- A financial institution hosts a critical database on Azure VMs with the following setup:
 - The database application is deployed across an availability set with three VMs to protect against hardware or maintenance failures.
 - An Azure Load Balancer distributes client traffic across the VMs.
 - Azure Backup provides data protection, while ASR replicates the VMs to a secondary region for disaster recovery.

Interactive lab simulations

- [Create a virtual machine in the portal.](#)

Create a virtual machine, connect and install the web server role.

- [Deploy a virtual machine with a template.](#)

Explore the QuickStart gallery and locate a virtual machine template. Deploy the template and verify the deployment.

- [Create a virtual machine with PowerShell.](#)

Use Azure PowerShell to deploy a virtual machine. Review Azure Advisor recommendations.

- [Create a virtual machine with the CLI.](#)

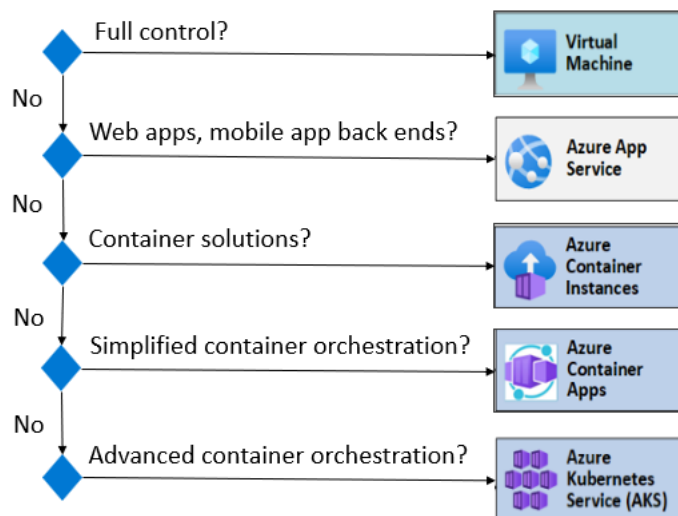
Use the CLI to deploy a virtual machine. Review Azure Advisor recommendations.

9 Administer PaaS Compute Options

1. Azure App Service Plans

Definition:

- Azure App Service Plans define the compute resources and pricing tier that power Azure App Services. These plans dictate the performance, scalability, and cost of your web applications and APIs.



Key Features:

1. Hosting Environment:

- The plan determines where your applications run (shared vs. dedicated environment).
- Shared plans are cost-effective but offer limited performance, while dedicated plans are more powerful and scalable.

2. Pricing Tiers:

- **Free/Shared:** Ideal for testing or small projects; multiple apps share the same resources.
- **Basic:** Dedicated compute with a single instance, suitable for simple production workloads.
- **Standard:** Supports scaling out to multiple instances, custom domains, and SSL.

- **Premium:** Enhanced performance, auto-scaling, and access to advanced features like VNet integration.
- **Isolated:** Runs in a dedicated environment for enhanced security and compliance.

3. Scaling Options:

- **Manual Scaling:** Add or remove instances as needed.
- **Auto-Scaling:** Automatically adjusts the number of instances based on metrics like CPU usage or request count.

4. Custom Domain and SSL:

- Enables the use of custom domain names and HTTPS for secure communication.

Use Cases:

- Hosting e-commerce websites, APIs, or enterprise applications.
- Running high-availability web apps with auto-scaling.

Example:

- A retail company creates a Standard App Service Plan to host its e-commerce platform. They enable auto-scaling to handle peak shopping hours and use a custom domain with SSL for secure transactions.

2. Azure App Services

Definition:

Azure App Services is a fully managed Platform-as-a-Service (PaaS) offering for hosting web applications, RESTful APIs, and mobile backends.



.NET



Node.js



PHP



Java



Python (on Linux)



HTML



Custom Windows/Linux Container

Key Features:**1. Supported Languages:**

- Supports .NET, Java, Node.js, PHP, Python, and Ruby, as well as custom runtimes in containers.

2. Deployment Options:

- Deploy apps using Git, GitHub Actions, Azure DevOps, FTP, or Azure Portal.
- Integration with CI/CD pipelines for streamlined updates.

3. Built-In Features:

- **Monitoring and Diagnostics:** Integration with Application Insights for performance monitoring.
- **Security:** Built-in authentication and authorization using Azure AD, Google, Facebook, etc.
- **Backup and Restore:** Automated backups to ensure data integrity.
- **Staging Slots:** Deploy updates to a staging slot before swapping them into production, ensuring zero-downtime deployments.

4. Global Availability:

- Deploy apps in multiple regions for high availability and lower latency.

5. Serverless Integration:

- Combine with Azure Functions for event-driven, serverless compute options.

Use Cases:

- Hosting scalable APIs or microservices.
- Running web apps with high availability and global reach.

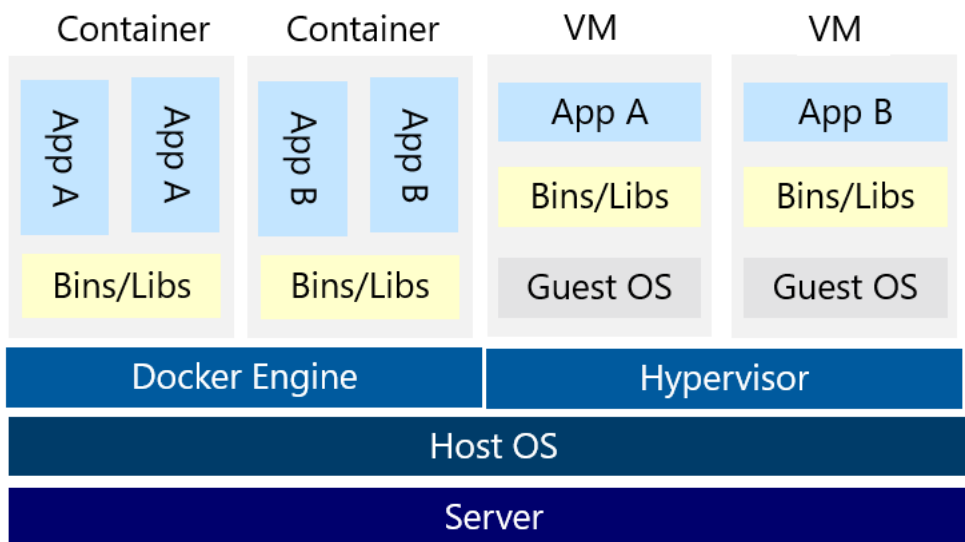
Example:

A startup builds its customer portal using Azure App Services. The app is written in Python, deployed via GitHub, and uses staging slots for seamless updates. Application Insights monitors performance to ensure an optimal user experience.

3. Azure Container Instances (ACI)

Definition:

Azure Container Instances is a serverless container service that enables you to run Docker containers in the cloud without managing underlying infrastructure.



Key Features:

1. Fast Deployment:

- Containers start in seconds, providing a quick way to deploy apps or run tasks.

2. Flexible Sizing:

- Allocate CPU and memory based on your needs, with the ability to handle both lightweight and resource-intensive workloads.

3. Networking Options:

- Expose containers to the public internet with DNS names or keep them private within a virtual network.

4. Persistent Storage:

- Attach Azure File shares to containers for persistent data storage.

5. Event-Driven Architecture:

- Integrate with Azure Logic Apps, Azure Functions, or Azure Event Grid to run containers in response to events.

6. Pricing Model:

- Pay only for the resources consumed while the container runs.

Use Cases:

- Running stateless applications or batch jobs.
- Hosting containerized APIs or lightweight microservices.
- Quickly deploying development and testing environments.

Example:

A data analytics company runs a containerized Python application on ACI to process large datasets on demand. Containers are spun up when a new job is received, and results are saved to an Azure Blob Storage.

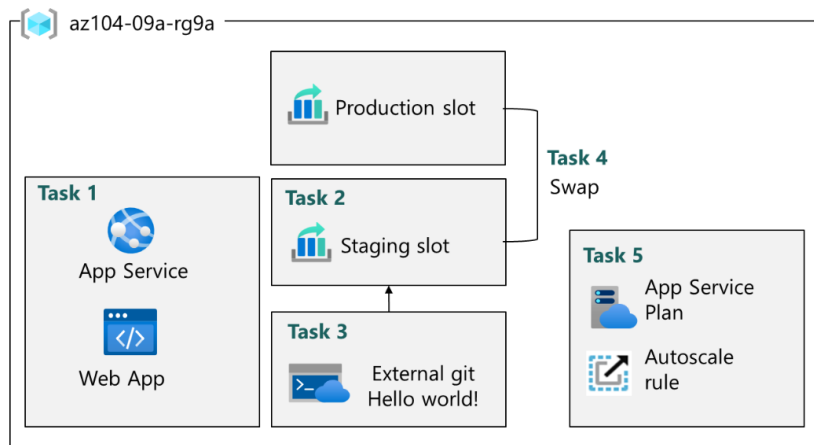
How These Concepts Work Together

1. **Azure App Service Plans** provide the foundation for hosting web apps and APIs, determining the available resources and cost structure.
2. **Azure App Services** utilize the plan to host scalable, fully managed web applications with built-in features like monitoring, security, and global availability.
3. **Azure Container Instances** complement App Services by offering a lightweight, serverless way to run containerized applications or jobs on demand.

Practical Scenario:

- An enterprise builds a customer-facing web application using Azure App Services. The app is hosted on a Premium App Service Plan to handle high traffic and ensure security. Background data processing tasks, like generating reports, are offloaded to containers running on Azure Container Instances.

Interactive lab simulations

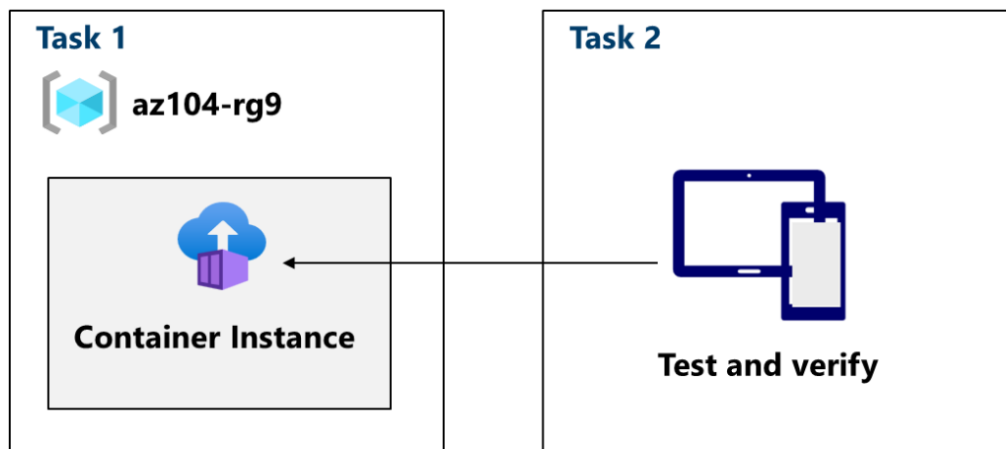


- [Create a web app.](#)

Create a web app that runs a Docker container.

- [Implement Azure web apps.](#)

Create an Azure web app, manage the deployment, and scale the app.



- [Deploy Azure Container Instances.](#)

Create, configure, and deploy a Docker container with Azure Container Instances.

- [Implement Azure Container Instances.](#)

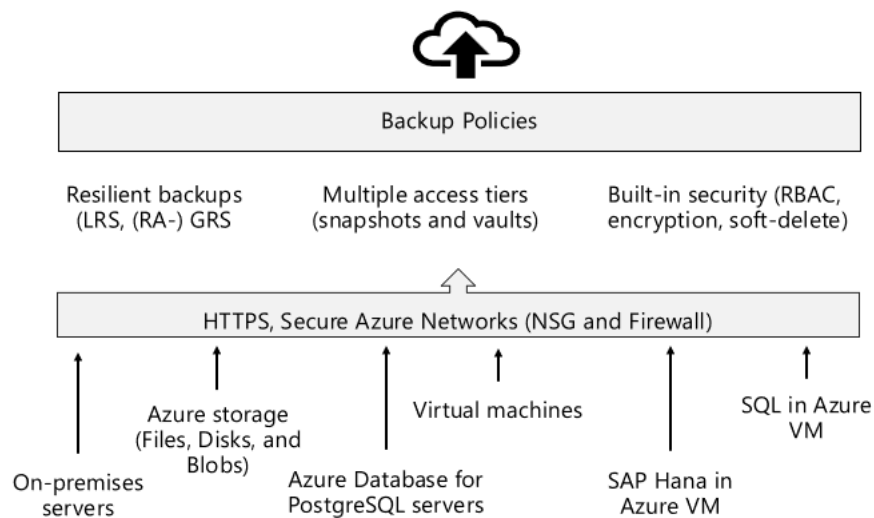
Deploy a Docker image using Azure Container Instances.

10 Administer Data Protection

1. File and Folder Backups

Definition:

Azure provides tools to back up critical files and folders on on-premises servers, virtual machines, and client devices to protect against accidental deletion, corruption, or disasters.



Key Features:

1. Azure Backup:

- A cloud-based solution for backing up files, folders, and system states from Windows servers, Azure VMs, and on-premises systems.

2. Backup Components:

- **Microsoft Azure Recovery Services (MARS) Agent:**

Used for backing up files, folders, and the system state from on-premises machines or Azure VMs to a Recovery Services vault.

- **Azure File Share Backup:**

Protects Azure file shares in storage accounts by creating incremental snapshots.

3. Backup Policies:

- Define the backup schedule (daily, weekly) and retention period (short-term or long-term).
- Example: Backup files daily and retain backups for 30 days.

4. Incremental Backups:

- Only changes since the last backup are saved, reducing storage costs and improving efficiency.

5. Recovery Options:

- Restore individual files or folders from backup points stored in the Recovery Services vault.
- Restore to the original location or an alternate one.

Advantages:

- Centralized management of file backups.
- Encryption at rest and in transit to protect data.
- No need for on-premises backup hardware.

Use Cases:

- Protecting critical documents on an on-premises file server.
- Backing up user home directories stored in Azure file shares.

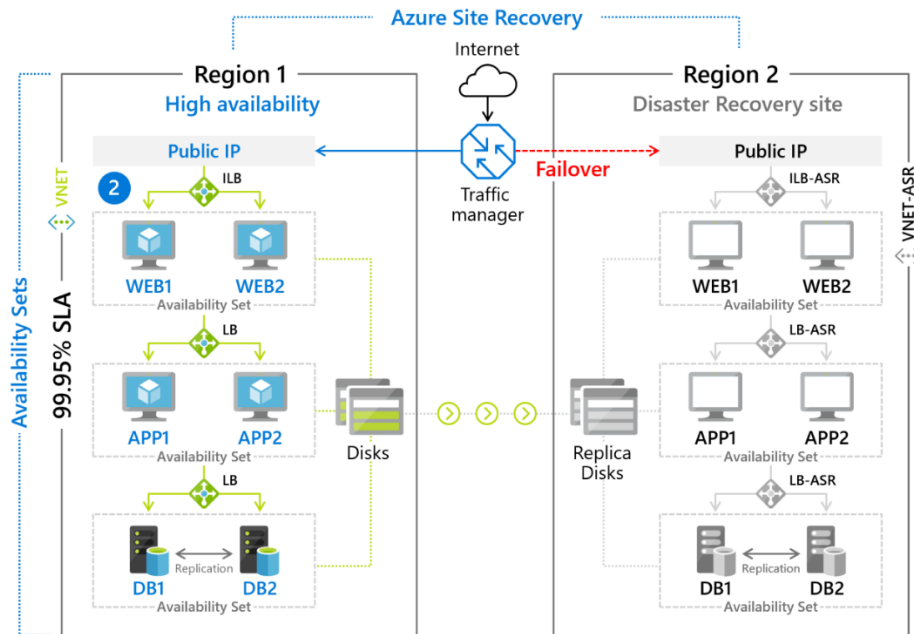
Example:

A law firm uses the MARS agent to back up client records from their on-premises server to Azure. Daily incremental backups ensure that recent changes are always recoverable.

2. Virtual Machine Backups

Definition:

Azure Backup provides native support for backing up Azure Virtual Machines, ensuring data protection and enabling quick recovery in case of failure.



Key Features:

1. Backup Configuration:

- Backups are enabled by associating the VM with a Recovery Services vault.
- Backup policies define the schedule and retention period.

2. Types of Backup:

- **Full Backup:** Captures all VM data during the first backup.
- **Incremental Backup:** Captures only changed data after the initial backup, optimizing storage.

3. Application-Consistent Backups:

- Uses the Volume Shadow Copy Service (VSS) to ensure backups capture application data in a consistent state, minimizing data corruption risks.

4. Granular Recovery:

- Restore the entire VM or specific files from the backup.
- Restore to the original VM, a new VM, or a different region.

5. Cross-Region Restore:

- Enables restoration of backups to a secondary region in case of regional outages.

6. Retention and Compliance:

- Backup retention can be configured for short-term (daily/weekly) or long-term (monthly/yearly).
- Helps meet compliance requirements for data retention policies.

Advantages:

- No additional infrastructure needed for VM backups.
- Automatic backup scheduling and management.
- Data encryption for secure backups.
- Easy integration with disaster recovery strategies.

Use Cases:

- Backing up production VMs running critical applications.
- Enabling cross-region recovery for disaster scenarios.

Example:

- A financial company uses Azure Backup to protect its SQL Server VM hosted in Azure. They schedule daily backups, retain weekly backups for a month, and retain monthly backups for a year to comply with regulatory requirements.

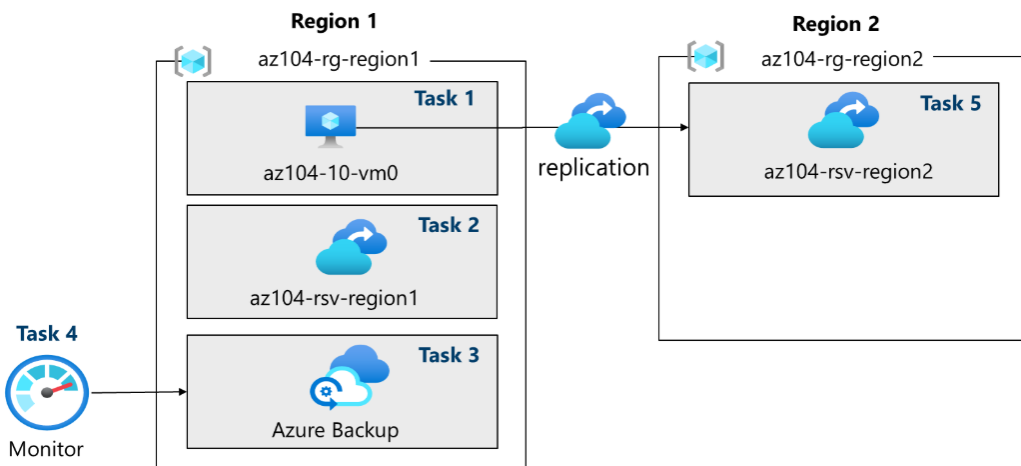
How These Concepts Work Together

1. **File and Folder Backups** ensure protection for individual files and folders stored on servers, workstations, or Azure file shares.
2. **Virtual Machine Backups** provide comprehensive protection for entire VMs, ensuring the recovery of both data and the underlying system in case of failures.

Practical Scenario:

- An organization runs critical workloads on Azure VMs while storing documents on Azure file shares.
 - Azure Backup is configured for daily backups of the VMs, ensuring full system recovery is possible in case of corruption or outages.
 - Azure File Share Backup protects shared files, enabling quick recovery of specific files accidentally deleted by users.

Interactive lab simulations



- **Backup virtual machines and on-premises files..**

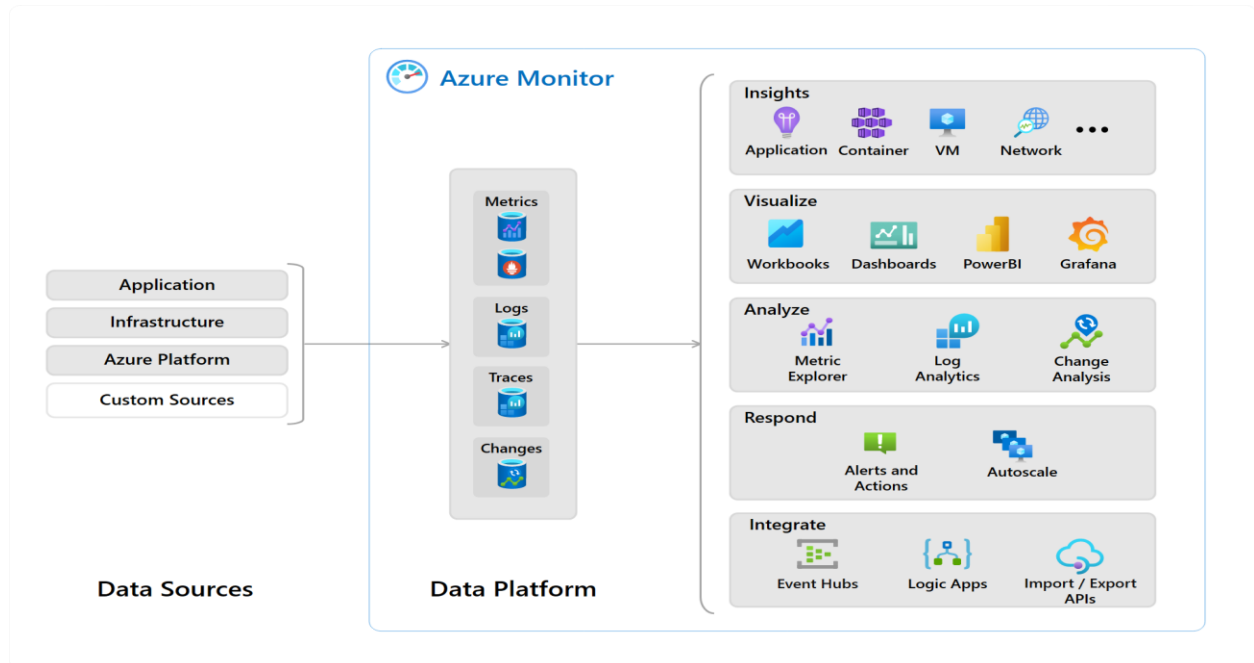
Create a recovery services vault and implement an Azure virtual machine backup. Implement on-premises file and folder backup using the Microsoft Azure Recovery Services agent. On-premises backups are outside the scope of this lab but it might be helpful to view those steps.

11 Administer Monitoring

1. Azure Monitor

Definition:

Azure Monitor is a comprehensive monitoring solution that collects, analyzes, and acts on telemetry data from Azure resources, applications, and infrastructure.



Key Features:

1. Data Sources:

- Collects data from:
 - **Azure Resources:** Metrics and activity logs from Azure services.
 - **Applications:** Logs and performance data from hosted applications.
 - **Guest OS:** Logs from VMs or on-premises machines using agents.
 - **Custom Sources:** External data sources via APIs.

2. Metrics and Logs:

- **Metrics:** Numerical data (e.g., CPU utilization, disk IOPS) collected at regular intervals.
- **Logs:** Detailed, event-based data (e.g., application errors, user activity).

3. Visualizations:

- Create **dashboards** and **workbooks** to visualize monitoring data in real-time.
- Example: Monitor CPU usage across multiple VMs in a graphical format.

4. Insights:

- **Application Insights:** Monitor application performance and diagnose bottlenecks.
- **VM Insights:** Monitor VM performance, disk usage, and dependencies.
- **Container Insights:** Track the health and performance of containerized workloads.

5. Integration with Automation:

- Automate responses to critical issues using Azure Logic Apps, Functions, or runbooks.

Advantages:

- Centralized monitoring across Azure and hybrid environments.
- Real-time alerts and automated issue resolution.
- Helps optimize resource usage and costs by identifying inefficiencies.

Example:

- A company uses Azure Monitor to track the performance of their Azure SQL Database. Dashboards show query execution times, helping the team address slow queries quickly.

2. Azure Alerts

Definition:

- Azure Alerts notify users about critical conditions in their resources, enabling prompt responses to issues.

Key Features:

1. Types of Alerts:

- **Metric Alerts:** Triggered when specific thresholds are breached (e.g., CPU > 80%).

- **Activity Log Alerts:** Triggered by specific events in Azure Activity Logs (e.g., resource deletion).
- **Log Alerts:** Based on queries from Log Analytics (e.g., when a VM goes offline).

2. Alert Rules:

- Define conditions that trigger an alert:
 - **Scope:** Resources to monitor.
 - **Condition:** Metrics or logs to evaluate.
 - **Threshold:** The value that triggers an alert.

3. Action Groups:

- Specify actions to take when an alert is triggered, such as:
 - Sending email/SMS notifications.
 - Triggering webhooks to integrate with third-party tools like Slack or PagerDuty.
 - Executing automated workflows using Azure Logic Apps or Functions.

4. Alert Severity:

- Classify alerts based on their importance:
 - **Sev 0 (Critical):** Immediate action required.
 - **Sev 1-3:** Gradual decrease in urgency.

Advantages:

- Early detection of issues, reducing downtime.
- Automates the notification process, ensuring prompt responses.
- Flexible configuration for different monitoring needs.

Example:

An e-commerce site sets up alerts for high memory usage on their VMs. If memory exceeds 90%, an alert is sent to the operations team and an Azure Function is triggered to scale up resources automatically.

3. Log Analytics

Definition:

Log Analytics is a tool within Azure Monitor that allows you to query and analyze log data collected from Azure resources, applications, and infrastructure.

Key Features:

1. Data Collection:

- Collects logs from:
 - Azure resources (e.g., VMs, App Services).
 - On-premises machines via the Microsoft Monitoring Agent.
 - Custom sources through APIs.

2. Kusto Query Language (KQL):

- Use KQL to write queries that analyze logs.
- Example: Identify failed login attempts on a VM:
 - SecurityEvent
 - | where EventID == 4625
 - | summarize count() by AccountName

3. Visualization and Insights:

- Create **charts**, **tables**, and **dashboards** from query results.
- Example: Display a graph of CPU usage trends across VMs.

4. Workspaces:

- Logs are stored in Log Analytics workspaces, which act as centralized repositories.
- Can integrate multiple Azure resources into one workspace.

5. Integration:

- Works seamlessly with Azure Monitor, Azure Alerts, and external tools like Power BI for advanced analytics.

Advantages:

- Helps diagnose and troubleshoot issues by analyzing historical data.
- Provides detailed insights into performance, security, and operations.
- Scalable and flexible to handle large volumes of log data.

Example:

A financial institution uses Log Analytics to track and investigate security events, such as unauthorized access attempts, across their Azure VMs.

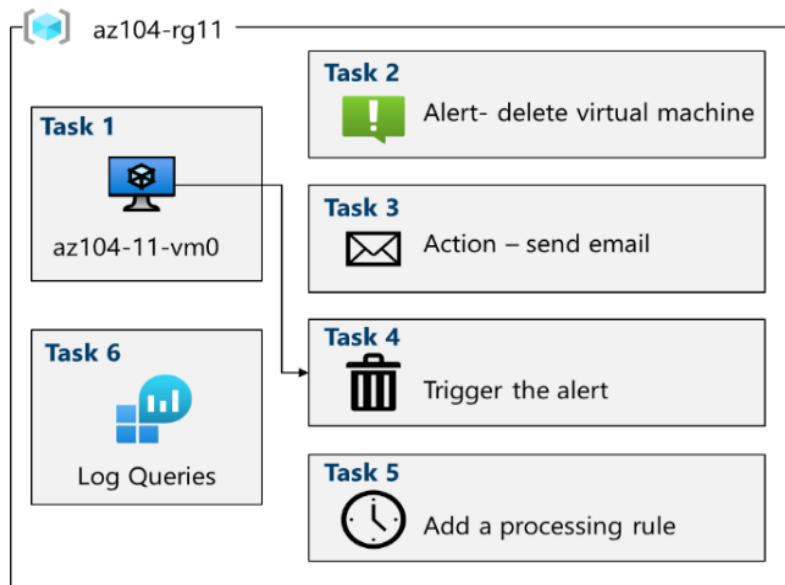
How These Concepts Work Together

1. **Azure Monitor** collects telemetry data (metrics and logs) from resources, applications, and infrastructure, providing a unified monitoring solution.
2. **Azure Alerts** leverage Azure Monitor data to notify users when critical thresholds are breached, enabling prompt responses to issues.
3. **Log Analytics** enhances Azure Monitor by enabling detailed log analysis, custom queries, and historical data insights.

Practical Scenario:

- An organization uses Azure Monitor to oversee the health of their application environment, from VMs to APIs.
 - Azure Alerts notify the team if a VM exceeds 80% CPU usage, and an automated action scales the environment.
 - Log Analytics provides detailed reports on user activity and application errors, helping developers fix issues proactively.

Interactive lab simulations



- [Implement monitoring.](#)

Create a Log Analytics workspace and Azure-automation solutions. Review monitoring and diagnostic settings for virtual machines. Review Azure Monitor and Log Analytics functionality.
