AZ-104 Microsoft Azure Administrator Certification

Deploy and Manage Azure compute resources

❖ **Internet Information Services**

Internet Information Services (IIS) is a web server software created by Microsoft. It allows you to host websites and web applications. Think of it as a platform where you can put your website so that people can access it over the internet.

- **For Windows install Internet Information Services (IIS) for linux install nginx server**

Using IIS on Azure Virtual Machines: When you use Azure Virtual Machines (VMs), you can install IIS to host your websites and web applications. Here's a simple way to understand how to set it up:

Steps to Install IIS on an Azure VM

1. **Create a Windows VM**: First, you need to create a Windows virtual machine in Azure. You can do this through the Azure Portal.
2. **Connect to Your VM**: Once your VM is created, connect to it using Remote Desktop Protocol (RDP).
3. **Install IIS**:
   - After connecting to your VM, open the **Server Manager**.
   - Click on **"Add roles and features"**.
   - Follow the wizard and select **"Web Server (IIS)"**.
   - Complete the installation process.
4. **Deploy Your Website**: Once IIS is installed, you can deploy your website by placing your web files in the **wwwroot** folder (usually located at C:\inetpub\wwwroot).
5. **Access Your Website**: You can now access your website using the public IP address of your VM. Just type the IP address in a web browser.

Benefits of Using IIS on Azure VMs

- **Scalability**: Easily scale your web server up or down based on traffic.
- **Flexibility**: Customize your server environment as needed.
- **Integration**: Seamlessly integrate with other Azure services like Azure Storage and Azure SQL Database.

❖ **Deploying a Linux machine - SSH keys**

SSH keys are a pair of cryptographic keys used to securely connect to a remote server without using a password. There are two keys:

- **Private Key**: Kept on your local machine and should be kept secret.
- **Public Key**: Shared with the server you want to connect to.

Steps to Deploy a Linux Machine with SSH Keys

1. **Generate SSH Keys**:
   - On your local computer, open a terminal and run the command:
     ssh-keygen -t rsa -b 4096 -C "your_email@example.com"

- This will create a pair of keys (private and public). By default, they are saved in the ~/.ssh directory.

2. **Copy the Public Key to the Server**:
   - Use the following command to copy your public key to the server:
     ssh-copy-id username@server_ip_address
   - Replace username with your server's username and server_ip_address with the server's IP address.
3. **Create a Linux VM in Azure**:
   - Go to the Azure Portal and sign in.
   - Click on "Create a resource" and select "Virtual Machine".
   - Choose the Linux operating system you want to use.
   - In the "Authentication type" section, select "SSH public key".
   - Paste your public key (found in ~/.ssh/id_rsa.pub) into the provided field.
4. **Connect to Your Linux VM**:
   - Once the VM is created, you can connect to it using SSH:
     ssh username@server_ip_address
   - Replace username with your VM's username and server_ip_address with the VM's IP address.

Benefits of Using SSH Keys

- **Security**: SSH keys are more secure than passwords because they are harder to crack.
- **Convenience**: Once set up, you can connect to your server without entering a password each time.
- **Automation**: SSH keys are useful for automated scripts and deployments.

❖ **Azure Virtual Machine - Disks**

Azure VM Disks are virtual storage devices used by Azure Virtual Machines to store data, operating systems, and applications. Think of them as the hard drives of your virtual computer in the cloud.

Types of Azure VM Disks

1. **Operating System Disk**: This is where the operating system (like Windows or Linux) is installed. It's similar to the C: drive on a physical computer.
2. **Data Disk**: These are additional disks you can attach to your VM to store data, applications, or anything else you need. They are like extra hard drives you can add to your computer.
3. **Temporary Disk:** Provides short-term storage and is not persistent; data is lost when the VM is stopped or deallocated.

Types of Disk Storage

Azure offers different types of disk storage to suit various needs:

1. **Ultra Disks**: These are the fastest disks, designed for high-performance workloads like databases and transaction-heavy applications.
2. **Premium SSDs**: These are solid-state drives (SSDs) that offer high performance and low latency, suitable for production and performance-sensitive applications.
3. **Standard SSDs**: These SSDs provide a balance between cost and performance, ideal for web servers and lightly used applications.
4. **Standard HDDs**: These are traditional hard disk drives (HDDs) that are cost-effective and suitable for backup or non-critical data.

❖ **What happens when we stop the machine**

Restarting vs. Stopping/Deallocating a VM

When you work with Azure Virtual Machines (VMs), there are two main operations you can perform: restarting and stopping/deallocating. These operations affect the data on your VM's disks differently.

Practical Example

1. **Restarting the VM**:
   - You save files on the OS disk (C: drive), data disk (F: drive), and temporary disk (D: drive).
   - After restarting, all files are still there.
2. **Stopping/Deallocating the VM**:
   - You save files on the OS disk (C: drive), data disk (F: drive), and temporary disk (D: drive).
   - After stopping and starting the VM, files on the OS and data disks are still there, but files on the temporary disk are gone.

❖ **Azure Disks - Server Side Encryption**

Server-side Disk Encryption is a feature in Azure that automatically encrypts the data stored on your virtual machine (VM) disks. This means that your data is turned into a code that only authorized users can read, making it secure even if someone gets hold of the physical disk.

Why Encrypt Your Data?

Even though Azure data centers are very secure, companies often have strict security policies that require all data to be encrypted. This ensures that if someone somehow gets access to the physical disk, they won't be able to read the data without the encryption key.

How Does It Work?

1. **Data Storage**:
   - Your VM's data disks are stored in Azure data centers.
   - Before the data is stored, it is encrypted.
2. **Encryption at Rest**: This means the data is encrypted when it is stored on the disk, not just when it is being transmitted.

Types of Encryption Keys

1. **Platform Managed Keys**: By default, Azure uses its own encryption keys to encrypt your data. These are called Platform Managed Keys.
2. **Customer-Managed Keys**: If you prefer, you can use your own encryption keys. This gives you more control over the encryption process. To use your own keys, you need to set up something called **Disk Encryption Sets**.

❖ **Azure Key Vault Service**

Azure Key Vault is a cloud service that helps you securely store and manage sensitive information, such as passwords, API keys, certificates, and cryptographic keys. Think of it as a highly secure digital safe where you can keep all your important secrets.

Security: It keeps your secrets safe and secure, ensuring that only authorized users and applications can access them.

Centralized Management: You can manage all your secrets in one place, making it easier to control and monitor access.

Example Scenario: Imagine you have a web application that needs to connect to a database. Instead of hardcoding the database password in your application code, you store the password in Azure Key Vault. Your application can then securely retrieve the password from the Key Vault when it needs to connect to the database.

❖ Using Azure Key Vault for Disk Encryption

You want to use your own encryption key to encrypt the disks of your Azure Virtual Machine (VM). This involves a few steps, but I'll explain it simply.

Steps to Use Your Own Encryption Key

1. **Create an Encryption Key in Key Vault**:
   - Go to the **Key Vault** in the Azure Portal.
   - Click on the **Keys** section.
   - Click on the **Generate** button to create a new key.
   - Give the key a name and click **Create**.
2. **Check Current Disk Encryption**:
   - Go to your VM (e.g., appvm) in the Azure Portal.
   - Check the **Disks** section. You'll see that the OS disk is already encrypted with **Platform Managed Keys (PMK)**, which are managed by Azure.
3. **Create a Disk Encryption Set**:
   - Search for **Disk Encryption Set** in the Azure Portal and click **Create**.
   - Choose your subscription and resource group.
   - Give it a name and select the region.
   - Choose **Encryption at rest with a customer-managed key**.
   - Select your Key Vault and the key you created.
   - Click **Review and Create**.
4. **Stop the VM**:
   - Go to your VM and click **Stop** to deallocate it. This is necessary to change the encryption settings.
5. **Change Disk Encryption to Customer-Managed Key**:
   - After the VM is stopped, go to the **Disks** section.
   - Select the disk you want to encrypt.
   - Go to the **Encryption** settings.
   - Change from **Platform Managed Key** to **Customer Managed Key**.
   - Select the Disk Encryption Set you created and click **Save**.
6. **Grant Permissions**:
   - If you get an error about permissions, go to your Disk Encryption Set.
   - Click on the error message to automatically grant the necessary permissions to your Key Vault.
7. **Restart the VM**:
   - Once the encryption settings are updated, start your VM again.

Important Notes

- **Encryption Process**: The data is encrypted transparently. You won't see the encryption process, but your data will be secure.

- **Cannot Revert**: Once you switch to customer-managed keys, you cannot revert back to platform-managed keys.

Summary

By following these steps, you can use your own encryption keys stored in Azure Key Vault to encrypt the disks of your Azure VM. This ensures that your data is secure and meets your company's security policies.

❖ **Azure Disk Encryption**

Azure Disk Encryption is a feature that allows you to encrypt the disks of your Azure Virtual Machines (VMs) using tools available at the operating system (OS) level. This means your data is protected both when it's stored and when it's being used by the OS.

Steps to Use Azure Disk Encryption

1. **Delete Existing Resources**:
   - First, delete all resources except for the Key Vault. This is to start fresh.
2. **Create a New VM**:
   - Go to the Azure Portal and create a new VM.
   - Choose your subscription and resource group.
   - Provide the necessary details and select a Windows Server.
   - Add a new data disk during the VM creation process.
3. **Set Up Disk Encryption**:
   - Once the VM is created, go to the **Disks** section.
   - In the **Additional Settings**, choose to encrypt either the OS disk or both the OS and data disks.
   - Use an encryption key stored in your Key Vault.

Types of Encryption

1. **Server-Side Encryption**: This is done by Azure on the storage units in the data center. It encrypts your data before storing it.
2. **Azure Disk Encryption**: This uses OS-level tools like BitLocker (for Windows) or DM-Crypt (for Linux) to encrypt data. The encryption and decryption happen seamlessly in the background.

Summary: Azure Disk Encryption provides an additional layer of security by encrypting your VM disks using OS-level tools. This ensures that your data is protected both when stored and when being used by the OS. If you encounter any issues, you can use the Azure Monitor service to troubleshoot.

❖ **Data Disks Snapshot**

A **Data Disk Snapshot** is like taking a picture of your data disk at a specific moment in time. This snapshot captures the exact state of the disk, including all the data on it, so you can use it later if needed.

Why Use Snapshots?

- **Backup**: Snapshots are useful for creating backups of your data. If something goes wrong, you can restore your disk to the state it was in when the snapshot was taken.
- **Testing**: You can use snapshots to test changes without affecting the original data. If the changes don't work out, you can revert to the snapshot.
- **Migration**: Snapshots can help you move data from one VM to another.

- **How to Create a Snapshot**

1. **Go to the Azure Portal**: Sign in to the Azure Portal.
2. **Select the Disk**:
   - Navigate to the VM whose disk you want to snapshot.
   - Go to the **Disks** section and select the data disk you want to snapshot.
3. **Create the Snapshot**:
   - Click on **Create Snapshot**.
   - Provide a name for the snapshot and select the resource group and region.
   - Choose the type of snapshot (full or incremental).
4. **Review and Create**: Review your settings and click **Create**.

- **Types of Snapshots**

1. **Full Snapshot**: This captures the entire disk. It's like a complete copy of the disk at that moment.
2. **Incremental Snapshot**: This captures only the changes made since the last snapshot. It's more storage-efficient because it doesn't duplicate unchanged data[12].

❖ **Azure Shared Disks**

Azure Shared Disks is a feature that allows you to attach a single managed disk to multiple virtual machines (VMs) at the same time. This is useful for scenarios where multiple VMs need to access the same data, such as in clustered applications.

Why Use Shared Disks?

- **Clustered Applications**: Shared disks are ideal for applications that require multiple VMs to work together and access the same data.
- **High Availability**: They help ensure that your application remains available even if one VM fails, as other VMs can still access the shared disk.

How It Works

1. **Attach to Multiple VMs**: You can attach a shared disk to multiple VMs. These VMs can read and write to the disk simultaneously.
2. **Cluster Manager**: A cluster manager (like Windows Server Failover Cluster or Pacemaker) is used to handle communication and manage access to the shared disk. This ensures that data is not corrupted when multiple VMs try to write to the disk at the same time.
3. **SCSI Persistent Reservations (SCSI PR)**:This is a standard used to manage access to the shared disk. It ensures that only one VM can write to a specific part of the disk at a time, preventing data corruption.

Example Scenario: Imagine you have a database application that needs to be highly available. You can set up multiple VMs in a cluster, all accessing the same shared disk. If one VM fails, the others can continue to access the data, ensuring your application remains available.

❖ **Custom Script Extensions**

A **Custom Script Extension** is a feature in Azure that allows you to run scripts on your virtual machines (VMs) after they have been deployed. Think of it as a way to automate tasks

like installing software, configuring settings, or performing maintenance without having to manually log into each VM.

Why Use Custom Script Extensions?

- **Automation**: Automate repetitive tasks, saving time and reducing errors.
- **Consistency**: Ensure that all VMs are configured in the same way.
- **Flexibility**: Run any script you need, whether it's for setup, configuration, or maintenance.

How It Works

1. **Create a Script**: Write a script in a language like PowerShell (for Windows) or Bash (for Linux) that performs the tasks you need.
2. **Upload the Script**: Store your script in a location accessible by your VM, such as Azure Storage, GitHub, or an internal file server.
3. **Add the Custom Script Extension**: In the Azure Portal, go to your VM and add the Custom Script Extension. Provide the URL to your script and any parameters it needs.
4. **Run the Script**: The extension will download and run the script on your VM.

Example Scenario: Imagine you have a VM that needs to have a web server installed and configured. Instead of logging into the VM and doing this manually, you can write a script that installs and configures the web server. Then, use the Custom Script Extension to run this script on your VM automatically.

❖ **Virtual Machine - Boot Diagnostics**

Boot Diagnostics is a feature in Azure that helps you troubleshoot issues with your virtual machines (VMs) when they fail to start properly. It provides you with important information about the VM's boot process, such as logs and screenshots, so you can see what went wrong.

Why Use Boot Diagnostics?

- **Troubleshooting**: Helps you identify and fix problems that prevent your VM from booting up.
- **Visibility**: Gives you a clear view of the VM's state during the boot process.

Example Scenario

Imagine your VM fails to start, and you don't know why. By using Boot Diagnostics, you can:

1. **View the Serial Log**: Check the log for error messages that might explain the issue.
2. **See a Screenshot**: Look at the screenshot to see if the VM is stuck on a specific screen, like a boot menu or error message.

How to Enable Boot Diagnostics

1. **For a New VM**: When creating a VM, ensure that Boot Diagnostics is enabled in the **Management** tab.
2. **For an Existing VM**:
    - Go to the VM in the Azure Portal.
    - In the **Help** section, select **Boot Diagnostics**.
    - Enable it and choose the storage account where the logs and screenshots will be stored.

❖ **Redeploying your VM**

Sometimes, your virtual machine (VM) might not start properly. This could be due to an issue with the physical server in the Azure data center where your VM is hosted.

Steps to Redeploy Your VM

1. **Stop and Start the VM**: First, try stopping the VM and then starting it again. This might move your VM to a different physical server or restart it on the same server.
2. **Use Redeploy and Reapply**:
   - If stopping and starting the VM doesn't work, you can use the **Redeploy and Reapply** option.
   - This option is found in the **Help** section of your VM settings in the Azure Portal.

What Does Redeploy Do?

- **Redeploy**: This action moves your VM to a new physical server in the Azure data center. It's like moving your VM to a new computer.
- **Reapply**: This action reapplies the VM's state and settings. It's useful if your VM is in a failed state and you need to reset its configuration.

❖ **Orchestration modes for azure virtual machine scale sets**

Uniform Orchestration Mode: Uniform mode is like having a fleet of identical cars. Every car is the same make, model, and color. This mode is great for scenarios where you need consistency and predictability.

- **Identical VMs**: All VMs in the scale set are identical in terms of configuration and size.
- **Automatic Scaling**: Azure automatically adds or removes VMs based on demand.
- **Single Configuration**: You manage a single configuration for all VMs.
- **Use Case**: Ideal for stateless applications where each instance can handle any request, like web servers.

2. Flexible Orchestration Mode: Flexible mode is like having a fleet of different types of vehicles. You might have cars, trucks, and motorcycles, each serving a different purpose. This mode offers more flexibility and control.

- **Diverse VMs**: VMs can have different configurations and sizes.
- **Manual Scaling**: You have more control over scaling and can manually add or remove VMs.
- **Multiple Configurations**: You can manage different configurations for different VMs.
- **Use Case**: Ideal for stateful applications or microservices where different instances have different roles, like a database server and an application server.

Real-World Example

Imagine you run an online store:

- **Uniform Mode**: You use this mode for your web servers. All servers are identical and handle incoming traffic. Azure automatically scales the number of servers up or down based on traffic.
- **Flexible Mode**: You use this mode for your backend services. You have different VMs for the database, payment processing, and inventory management. Each VM type can be scaled independently based on its specific needs.

❖ **Virtual Machine Scale Sets - Custom Script Extensions**

A **Custom Script Extension** is a tool that allows you to run scripts on your Azure VMs after they are deployed. This is useful for tasks like installing software, configuring settings, or performing any other setup tasks.

Step-by-Step Example

1. **Create a VMSS**:
   - Go to the Azure portal and search for "Virtual machine scale set".
   - Choose an existing resource group or create a new one.
   - Give your scale set a name and select the "Uniform" orchestration mode.
   - Choose an image, like Windows Server 2022 Datacenter.
   - Set a username and password for the VMs.
2. **Configure Networking**:
   - Allow selected ports, such as RDP (port 3389) and HTTP (port 80).
   - Enable the public IP address for remote access.
3. **Set Initial Instance Count**: Set the initial number of VMs to 2.
4. **Deploy the VMSS**: Review your settings and click "Create".
5. **Upload Your Script**:
   - Create a PowerShell script to install Internet Information Services (IIS) and create a default HTML page.
   - Upload this script to an Azure Storage account.
6. **Add Custom Script Extension**:
   - Go to your VMSS resource in the Azure portal.
   - Navigate to "Extensions + Applications" and add a Custom Script Extension.
   - Select your uploaded script from the storage account.
7. **Deploy the Extension**:
   - The script will be deployed to all VMs in the scale set.
   - If the extension is not immediately applied, you may need to manually upgrade the instances.
8. **Verify the Deployment**:
   - Once the script is applied, you can verify by accessing the public IP address of one of the VMs.
   - You should see the default HTML page created by your script.

❖ **Virtual machine images**

A **VM Image** is like a snapshot of a computer's hard drive. It contains everything needed to run a virtual machine, including:

1. **Operating System (OS)**: The main software that manages all the hardware and other software on the computer (e.g., Windows, Linux).
2. **Applications**: Any software programs that are installed on the computer.
3. **Configuration Settings**: All the settings and preferences that make the computer work the way you want.

Why Use VM Images?

1. **Quick Setup**: You can create new VMs quickly without having to install the OS and applications from scratch.

2. **Consistency**: Ensures all VMs created from the same image are identical, which is useful for testing and development.
3. **Backup and Recovery**: VM images can be used to back up a system. If something goes wrong, you can restore the VM to its previous state using the image.

How to Create and Use VM Images

1. **Create a VM**: Start with a VM where you install all the necessary software and configure settings.
2. **Capture an Image**: Once your VM is set up, you can create an image from it. This image will include everything on the VM.
3. **Deploy New VMs**: Use the image to create new VMs. These new VMs will have the same OS, applications, and settings as the original.

Types of VM Images

1. **Specialized Images**:
    - **User-Specific Information**: Retains specific user and machine information from the original VM.
    - **Use Case**: When you want the new VMs to have the same user accounts and settings as the original.
2. **Generalized Images**:
    - **No User-Specific Information**: Removes specific user and machine information before creating the image.
    - **Use Case**: When you want to create a clean, reusable image without any user-specific data.
    - **Generalization Process**: You need to prepare the VM by running a special command to remove user-specific information. After this, the original VM becomes unusable.

Azure Compute Gallery

- **Storage and Sharing**: Store your VM images in the Azure Compute Gallery. This allows you to share images across your organization.
- **Image Versions**: You can have different versions of your images, such as one with version 1 of your application and another with version 2.

❖ Generalized Image and Specialized image

Generalized Images: Think of a generalized image like a **template** for a VM. You prepare it by cleaning it up so it doesn't have any personal data or unique settings. This is done using a tool called sysprep.

- **Example**: Imagine you have a computer set up perfectly for your office work, with all the necessary software installed. You clean it up (remove personal files and settings) and save this clean state as a template. Now, whenever a new employee joins, you can use this template to set up their computer quickly.

Specialized Images: A specialized image is like taking a **snapshot** of your VM exactly as it is, with all its data and settings intact. You don't clean it up; you capture everything just as it is.

- **Example**: Suppose you have a computer running a specific project with all your files, settings, and software configured just the way you need. You take a snapshot of this computer. If something goes wrong, you can use this snapshot to create a new computer that looks exactly like the original one, with all your work and settings preserved.

Summary

- **Generalized Image**: Clean template, reusable, like setting up a new computer for multiple people.
    - **Specialized Image**: Exact copy, includes everything, like taking a snapshot of your current computer.

❖ **Proximity placement groups**

Proximity Placement Groups are a feature in Azure that helps you keep your virtual machines (VMs) physically close to each other in the same data center. This is important for applications that need very low network latency, meaning they need to communicate quickly and efficiently.

Why Use Proximity Placement Groups?

- **Low Latency**: By keeping VMs close together, you reduce the time it takes for data to travel between them.
- **Performance**: This is especially useful for applications that require fast communication between different parts, like databases and web servers.

How to Use Proximity Placement Groups:

1. **Create a Proximity Placement Group**:
    - In the Azure portal, search for "Proximity Placement Groups" and create a new one.
    - Choose your subscription, resource group, and region.
2. **Assign VMs to the Group**:
    - When creating a new VM, select the Proximity Placement Group you created.
    - If you have existing VMs, you can move them into the group, but you might need to stop (deallocate) them first.
3. **Deploy and Manage**:
    - Once your VMs are in the Proximity Placement Group, they will be physically close to each other, ensuring low latency.

Example:

Imagine you have a web application with a front-end server and a back-end database. By placing both servers in a Proximity Placement Group, you ensure that data travels quickly between them, improving the overall performance of your application.

❖ **Azure Web App logging**

Azure Web App logging is a feature that helps you track and record various activities and events happening in your web application. This is useful for debugging, monitoring, and ensuring your app runs smoothly.

Types of Logs:

1. **Application Logs**:

- These logs capture messages generated by your application code. They can include information, warnings, errors, and debug messages.
- You can enable application logging to store these logs in the file system or in Azure Storage blobs.

2. **Web Server Logs**:
- These logs record HTTP requests to your web app. They include details like the request method, resource URI, client IP, and response code.
- Web server logs can be stored in the file system or in Azure Storage blobs.

3. **Detailed Error Messages**:
- These logs capture detailed error pages that are generated when your application encounters errors (HTTP status codes 400 or higher).
- These error pages are saved in the file system for debugging purposes.

4. **Failed Request Tracing**:
- This feature provides detailed tracing information for failed requests. It helps you understand why a request failed and which components were involved.
- The logs include a trace of the IIS components used to process the request and the time taken by each component.

How to Enable Logging:

1. **Application Logging**:
- Go to your Azure Web App in the Azure portal.
- Navigate to "App Service logs" under the "Monitoring" section.
- Turn on "Application Logging (Filesystem)" or "Application Logging (Blob)" depending on where you want to store the logs[1].

2. **Web Server Logging**:
- In the same "App Service logs" section, turn on "Web Server Logging" and choose to store logs in the file system or Azure Storage blobs[1].

3. **Detailed Error Messages and Failed Request Tracing**:
- Enable these options in the "App Service logs" section to capture detailed error messages and failed request traces[1].

Example:

Imagine you have a web application, and users report that they are encountering errors. By enabling logging, you can capture detailed information about these errors and the requests that caused them. This helps you identify and fix issues more quickly.

Azure Web App logging is a powerful tool for maintaining and troubleshooting your web applications, ensuring they run efficiently and reliably.

❖ Azure Web Apps - Deployment Slots

Deployment slots are like different environments for your web app within the same Azure App Service. They allow you to run multiple versions of your web app simultaneously, each with its own unique URL.

Key Features:

1. **Staging Environment**:
- You can create a staging slot to test new changes before making them live.
- This helps ensure that everything works correctly without affecting the live (production) site.

2. **Swapping Slots**:

- Once you're happy with the changes in the staging slot, you can swap it with the production slot.
- This makes the new version live without any downtime.
- If something goes wrong, you can quickly swap back to the previous version.

3. **Multiple Slots**:
- Depending on your App Service Plan, you can have multiple slots (e.g., staging, testing, QA).
- Each slot is a live app with its own configuration and content.

Example:
Imagine you have a live website and you want to add a new feature. Instead of deploying directly to the live site, you:

- Create a staging slot.
- Deploy the new feature to the staging slot.
- Test the feature in the staging slot.
- Swap the staging slot with the production slot once you're sure everything works.

❖ **Auto-Scaling your Web app**

Autoscaling automatically adjusts the number of compute resources (like virtual machines) based on the demand for your web app. This ensures your app performs well even during high traffic periods.

How Autoscaling Works:

1. **App Service Plan**:
- Your web apps run on an App Service Plan, which provides the underlying compute resources.
- The App Service Plan can be scaled up (more powerful machines) or scaled out (more machines).

2. **Scaling Up vs. Scaling Out**:
- **Scale Up**: Increase the power of the existing machines (e.g., from Basic to Standard tier).
- **Scale Out**: Increase the number of machines running your web apps.

3. **Setting Up Autoscaling**:
- Go to your Azure Web App in the Azure portal.
- Navigate to the "Scale out (App Service plan)" section.
- You can manually set the number of instances (machines) or configure autoscaling rules.

Configuring Autoscaling Rules:

1. **Choose a Metric**:
- Autoscaling can be based on various metrics like CPU usage, memory usage, or data in/out.
- For example, you can set a rule to add more instances if CPU usage goes above 70%.

2. **Set the Rules**:
- Define the minimum, maximum, and default number of instances.
- Add a rule specifying the metric and threshold (e.g., if data in is greater than 70 bytes, add one instance).

3. **Save and Monitor**:
- Save the autoscaling configuration.
- Azure will automatically add or remove instances based on the defined rules.

Example:

- **Initial Setup**: You have one machine running your web app.
- **High Traffic**: If the CPU usage goes above 70%, Azure adds more machines to handle the load.
- **Low Traffic**: When the traffic decreases, Azure removes the extra machines to save costs.

❖ **Azure Web App - Virtual Network Integration**

Azure Virtual Network (VNet) integration allows your Azure services, like web apps or functions, to connect securely to other resources within a private network. Here's a simple breakdown:

1. **Virtual Network (VNet)**: Think of it as your own private network in the cloud. It helps keep your resources isolated and secure.
2. **Integration**: This means connecting your Azure services (like a web app) to this private network. This allows the service to access resources (like databases or other services) within the VNet securely.
3. **Benefits**:
   - **Security**: Your services can communicate without going over the public internet.
   - **Isolation**: Keeps your resources separate from others in the cloud.
   - **Control**: You can manage traffic and access rules within your VNet.

For example, if you have a web app that needs to access a database, integrating the web app with a VNet ensures that the communication between them is secure and private[1][2].

Example Scenario

You have a .NET application that you want to run on an Azure web app. This application needs to connect securely to a MySQL database running on a virtual machine (VM) in Azure. Initially, the VM had a public IP address for setup purposes, but now you want it to only have a private IP address for security reasons.

Steps

1. **Remove Public IP from VM**:
2. **Prepare .NET Project**:
3. **Publish .NET Project to Azure Web App**:
4. **Enable VNet Integration**:

Result

Once everything is set up, your Azure web app will be able to securely connect to the MySQL database on the VM using the private IP address. This setup ensures that the database is not exposed to the internet, enhancing security.

❖ **Azure Web App: Custom Domain**

A custom domain is a unique, personalized web address (like www.yourwebsite.com) that you can use instead of the default Azure-provided address (like yourapp.azurewebsites.net).

Why Use a Custom Domain?

- **Branding**: It makes your website look more professional and trustworthy.
- **Memorability**: Easier for users to remember and access your site.

Steps to Set Up a Custom Domain

1. **Buy a Domain**:

- Purchase a domain name from a domain registrar (like GoDaddy, Namecheap, etc.).

2. **Configure DNS Settings**:
- Log in to your domain registrar's website.
- Add DNS records to point your domain to your Azure web app. This usually involves creating an A record or a CNAME record.

3. **Add Custom Domain in Azure**:
- Go to the Azure portal and navigate to your web app.
- In the left menu, select **Custom domains**.
- Click **Add custom domain** and enter your domain name.
- Follow the instructions to verify ownership of the domain (this might involve adding a TXT record to your DNS settings).

4. **Verify and Save**:
- Once Azure verifies that you own the domain, it will link your custom domain to your web app.
- Now, your web app can be accessed using your custom domain.

5. **Done**: Now, when people type www.mywebsite.com, they will be directed to your Azure web app.

❖ **The need of Container**

A container is like a box that packages everything an application needs to run: the code, runtime, system tools, libraries, and settings. This makes it easy to move the application between different environments without worrying about compatibility issues.

Why Use Containers in Azure?

1. **Consistency**: Containers ensure that your application runs the same way everywhere, whether it's on your local machine, in a test environment, or in production on Azure. This eliminates the "it works on my machine" problem[1].
2. **Scalability**: Azure makes it easy to scale your containerized applications up or down based on demand. This means you can handle more users or workloads without manual intervention[2].
3. **Flexibility**: Containers can run almost any application, regardless of the technology stack. This makes it easier to modernize legacy applications or build new ones using microservices[1].

Example: Imagine you have a web application that needs to run consistently across different environments. By packaging it in a container, you can be sure it will work the same way on your development machine, in testing, and in production on Azure. Plus, if you need to handle more traffic, Azure can quickly scale your containers to meet the demand.

Example: You want to install Docker on an Azure VM and run a simple container-based application using an image from Docker Hub.

Steps

1. **Create an Azure VM**:
- Go to the Azure portal and create a new virtual machine (VM) using Ubuntu Linux.
- Choose your subscription, resource group, and give the VM a name.
- Select Ubuntu Server 22.04 as the image.
- Choose a small VM size (like 1 CPU and 1 GB memory).

- Set a username and password for logging into the VM.
- Review and create the VM.

2. **Install Docker on the VM**:
- Once the VM is created, get its public IP address.
- Use a tool like PuTTY to SSH into the VM using the public IP address.
- Run the commands to install Docker:
- Verify Docker installation by running:
  docker --version

3. **Run a Container**:
- Now that Docker is installed, you can run a container. For this example, we'll use the NGINX web server.
- Run the following command to download the NGINX image and start a container:

sudo docker run -d -p 80:80 --name mynginx nginx

This command does the following:
-d: Runs the container in the background.
-p 80:80: Maps port 80 of the VM to port 80 of the container.
--name mynginx: Names the container "mynginx".
nginx: Specifies the NGINX image from Docker Hub.

4. **Access the NGINX Web Server**:
- Open your web browser and go to the public IP address of your VM.
- You should see the NGINX welcome page, indicating that the NGINX server is running inside the Docker container.

❖ **Azure Container Registry**

Azure Container Registry (ACR) is a service provided by Microsoft Azure that allows you to store and manage container images. Think of it as a **library** where you keep all the different versions of your application packaged in containers.
Here's a simple breakdown:

- **Container Images**: These are like blueprints of your applications, including everything needed to run them (code, runtime, libraries, etc.).
- **Registry**: This is the place where you store these container images. ACR is a type of registry.
- **Push and Pull**: You can **push** (upload) your container images to ACR and **pull** (download) them when you need to deploy your application.
  ACR makes it easy to manage and deploy your applications consistently across different environments. It's secure, scalable, and integrates well with other Azure services.

Goal

You want to create a Docker image of your .NET application, publish that image to Azure Container Registry (ACR), and then use it for future deployments.

Steps

1. **Set Up the MySQL Database**:
- Instead of using a VM, you'll use Azure's managed MySQL service.
- Create a new MySQL database in Azure:
  - Go to the Azure portal and search for "MySQL database".

- Choose “Azure Database for MySQL” and create a new flexible server.
- Fill in the details like server name, location, and authentication method.
- Allow public access and add your client IP address.
- Create the database.

2. **Connect to the MySQL Database**:
   - Use MySQL Workbench on your local machine to connect to the Azure MySQL database:
   - Enter the server name, username, and password.
   - Create a new database and table, and insert some records.
3. **Update .NET Application**:
   - Open your .NET project in Visual Studio.
   - Update the connection string in your project to use the Azure MySQL database’s server name, username, and password.
   - Run the project locally to ensure it connects to the MySQL database.
4. **Dockerize the .NET Application**:
   - Create a Dockerfile in your project directory. This file tells Docker how to build your application image.
   - Replace Your Project with the actual name of your project file.
5. **Build and Test the Docker Image**:
   - Open a terminal and navigate to your project directory.
   - Run the following commands to build and test your Docker image:
     docker build -t yourappimage .
     docker run -d -p 8080:80 yourappimage
   - Open a browser and go to http://localhost:8080 to see your application running.
6. **Push the Image to Azure Container Registry (ACR)**:
   - Create an Azure Container Registry in the Azure portal.
   - Log in to your ACR from the terminal:
     az acr login --name yourregistryname
   - Tag your Docker image for ACR:
     docker tag yourappimage yourregistryname.azurecr.io/yourappimage
   - Push the image to ACR:
     docker push yourregistryname.azurecr.io/yourappimage

7. **Prepare Your .NET Application**:
   - Open your .NET project in Visual Studio.
   - Build and publish your application to a local folder.
8. **Create a Dockerfile**:
   - A Dockerfile contains instructions on how to build your Docker image.
   - Replace YourProject with the actual name of your project file.
9. **Copy Files to Azure VM**:
   - Use a tool like WinSCP to copy your published application files and Dockerfile to your Azure VM.
10. **Build the Docker Image on the VM**:
    - SSH into your Azure VM using a tool like PuTTY.
    - Navigate to the directory where you copied your files.
    - Run the following command to build the Docker image:
    - sudo docker build -t yourappimage .

11. **Run the Docker Container**:
- Stop any existing containers that might be using the same port:
- sudo docker stop <container_id>
- Run your application as a Docker container:
- sudo docker run -d -p 80:80 --name yourappcontainer yourappimage
12. **Access Your Application**:
- Open your web browser and go to the public IP address of your VM. You should see your .NET application running.

❖ **Publishing to the Azure Container Registry**

Azure Container Registry (ACR) is a service in Azure that allows you to store and manage your Docker container images in a private registry. Think of it as a secure library where you keep all the images of your applications.

Why Use ACR?

- **Centralized Storage**: Keep all your container images in one place.
- **Security**: Only authorized users can access your images.
- **Integration**: Works seamlessly with other Azure services like Azure Kubernetes Service (AKS).

Steps to Publish a Docker Image to ACR

1. **Create an Azure Container Registry**:
- Go to the Azure portal.
- Search for "Container Registry" and create a new registry.
- Fill in the details like registry name, resource group, and location.
- Click "Review + create" and then "Create".
2. **Log in to ACR**:
- Open a terminal on your local machine.
- Use the Azure CLI to log in to your ACR:
  az acr login --name yourregistryname
3. **Tag Your Docker Image**: Tag your Docker image with the ACR login server name:
   docker tag yourappimage yourregistryname.azurecr.io/yourappimage
4. **Push the Image to ACR**: Push your Docker image to ACR:
- docker push yourregistryname.azurecr.io/yourappimage
5. **Verify the Image in ACR**:
- Go back to the Azure portal.
- Navigate to your container registry and check the "Repositories" section to see your uploaded image.

Example

Imagine you have a Docker image named myappimage that you want to store in ACR. Here's how you do it:

1. **Create ACR**: Set up a new container registry in Azure.
2. **Log in**: Use the Azure CLI to log in to your registry.
3. **Tag Image**: Tag your image with the registry name.
4. **Push Image**: Upload the image to ACR.
5. **Verify**: Check the Azure portal to confirm the image is stored in ACR.

❖ **Azure Container Instance**

Azure Container Instances (ACI) is a service that allows you to run Docker containers in the cloud without managing the underlying infrastructure. It's a quick and easy way to deploy containers without needing to set up virtual machines or orchestrators.

Step-by-Step Procedure to Use Azure Container Instances

1. **Sign in to Azure Portal**: Go to the Azure Portal and sign in with your Azure account. If you don't have an account, you can create a free one.
2. **Create a Resource**:
   - On the Azure Portal homepage, click on "Create a resource".
   - Search for "Container Instances" and select it.
   - Click on "Create".
3. **Configure the Container Instance**:
   - **Subscription**: Choose your Azure subscription.
   - **Resource Group**: Select an existing resource group or create a new one.
   - **Container Name**: Give your container a unique name.
   - **Region**: Choose the region where you want to deploy your container.
   - **Image Source**: Select the container image source (e.g., Docker Hub, Azure Container Registry).
   - **Image**: Enter the name of the container image (e.g., nginx for an NGINX web server).
4. **Set Container Settings**:
   - **Size**: Specify the number of CPU cores and the amount of memory for your container.
   - **Networking**: Choose whether you want a public IP address or a private one within a virtual network.
5. **Review and Create**: Review your settings and click "Create". Azure will deploy your container instance.
6. **Monitor and Manage**: Once deployed, you can monitor the container's performance, view logs, and manage it through the Azure Portal.

❖ **Azure Container Group**

An **Azure Container Group** is a collection of containers that run together on the same host machine. Think of it as a small group of containers that share resources, a network, and a lifecycle. It's similar to a pod in Kubernetes.

Key Features of Azure Container Groups

1. **Shared Resources**: Containers in a group share the same CPU, memory, and storage resources. This means they can communicate with each other quickly and efficiently.
2. **Networking**: All containers in a group share a local network. They can communicate with each other using localhost. The group can also have a single public IP address, making it easy to expose services to the internet.
3. **Lifecycle Management**: Containers in a group start, stop, and scale together. This ensures that all containers in the group are always in sync.
4. **Volume Mounts**: You can mount Azure file shares or other storage volumes to the containers in the group, allowing them to share data.

Example Scenario

Imagine you have a web application that needs a sidecar container for logging. You can create a container group with two containers:

- **Web App Container**: Runs your main web application.
- **Logging Container**: Collects and processes logs from the web app.

❖ **Azure Container Apps**

Azure Container Apps is a service provided by Microsoft Azure that allows you to run your applications in containers without worrying about managing the underlying infrastructure.

What are Containers?

Containers are like lightweight, portable virtual machines that package your application and all its dependencies together. This makes it easy to run your application consistently across different environments.

What are Azure Container Apps?

Azure Container Apps is a **serverless platform** for running these containerized applications. Here's what makes it special:

1. **Serverless**: You don't need to manage servers. Azure handles all the infrastructure for you.
2. **Scalable**: Your applications can automatically scale up or down based on demand. For example, if more users start using your app, it will automatically allocate more resources.
3. **Cost-Effective**: You only pay for the resources you use. If your app isn't being used, you don't pay for idle resources.
4. **Flexible**: You can run different types of applications, such as web apps, APIs, background jobs, and microservices.
5. **Easy Integration**: It integrates well with other Azure services, making it easier to build complex applications.

❖ **Azure Kubernetes**

Kubernetes: Kubernetes, often abbreviated as K8s, is an open-source platform designed to automate deploying, scaling, and operating application containers. Here's a simple way to understand it:

1. **Containers**: Think of containers as small, portable units that package your application and all its dependencies. They ensure your app runs the same way, regardless of where it's deployed.
2. **Orchestration**: Kubernetes helps manage these containers. It automates tasks like starting, stopping, and scaling containers based on demand.
3. **Cluster**: Kubernetes runs your containers in a cluster of machines (nodes). It ensures your application is always running and can handle failures by redistributing the workload.

Azure Kubernetes Service (AKS): Azure Kubernetes Service (AKS) is a managed Kubernetes service provided by Microsoft Azure. Here's what makes it special:

1. **Managed Service**: Azure handles the management of the Kubernetes control plane, so you don't have to worry about the underlying infrastructure.
2. **Integration**: AKS integrates seamlessly with other Azure services, making it easier to build and deploy applications.

3. **Scalability**: AKS can automatically scale your applications based on demand, ensuring they run efficiently.
4. **Security**: Azure provides built-in security features and compliance certifications to protect your applications.

Let's break down the process of deploying an application using Azure Kubernetes Service (AKS) in simple steps:

1. Create a Kubernetes Cluster

Go to Azure Portal: Open the Azure portal and go to "All resources".
Create a Resource: Click on "Create" and search for "Kubernetes service".
Select AKS: Choose "Azure Kubernetes Service" and hit "Create".
Resource Group: Select your existing resource group.
Configuration: Choose the "Dev and Test" preset configuration.
Cluster Name: Give your cluster a name.
Location: Set the location to "North Europe".
Pricing Tier: Leave the pricing tier as "Free".
Scaling Method: Choose "Manual".
Number of Nodes: Set the number of nodes to 1 (for a simple setup).

2. Configure Node Pools and Access

Node Pools: Leave the default settings.
Access: Leave the default settings.

3. Networking and Integrations

Networking: Leave the default settings.
Integrations: Integrate with your Azure Container Registry (ACR) if you have one.

4. Review and Create

Tags: Add any tags if needed.
Review and Create: Click on "Review and Create" and then "Create". The cluster creation might take around 5-10 minutes.

5. Deploy Your Application

Go to Resource: Once the cluster is created, go to the resource.
Workloads: Navigate to "Workloads" in your cluster.
Deployment YAML: Use a YAML file to define your deployment. This file tells the cluster to use the image from your Azure Container Registry.
Service YAML: Use another YAML file to define a service that uses a load balancer to expose your application.

6. Apply YAML Files

Create Deployment: Go to "Workloads", click "Create", and apply the deployment YAML.
Create Service: Go to "Services", click "Create", and apply the service YAML.

7. Access Your Application

External IP: Once the service is up and running, it will have an external IP address.

Access via Browser: Open the external IP address in your browser to access your application.

❖ **Networking option in Azure Kubernetes**

Networking Options in AKS

When you set up a Kubernetes cluster in Azure, you have two main networking options to choose from: **Kubenet** and **Azure CNI**. These options determine how IP addresses are assigned to your containers (pods) and nodes (virtual machines).

1. Kubenet

- **Nodes**: Each node (virtual machine) gets an IP address from the Azure virtual network (VNet).
- **Pods**: Pods get IP addresses from a different, logically separate range, not directly from the VNet.
- **Network Address Translation (NAT)**: Used to allow pods to communicate with other resources in the VNet.
- **Pros**: Uses fewer IP addresses, simpler setup.
- **Cons**: Slightly more complex network traffic routing.

2. Azure CNI (Container Networking Interface)

- **Nodes and Pods**: Both nodes and pods get IP addresses directly from the VNet subnet.
- **Direct Access**: Pods can be accessed directly using their IP addresses.
- **Pros**: Easier to manage network traffic, better performance.
- **Cons**: Requires more IP addresses, more planning needed to avoid IP address exhaustion.

Steps to Configure Networking in AKS

1. **Create a Kubernetes Cluster**:
   - Go to the Azure portal and click on "Create a resource".
   - Search for "Kubernetes service" and select it.
   - Choose your existing resource group and configure other options as needed.
2. **Networking Section**:
   - In the networking section, you will see the options for **Kubenet** and **Azure CNI**.
   - **Kubenet**: Select this if you want a simpler setup with fewer IP addresses.
   - **Azure CNI**: Select this if you need direct IP addresses for your pods and better network performance.
3. **Nodes and Pods**:
   - Nodes are the virtual machines that run your containers.
   - Pods are the containers that run on these nodes.
   - The choice between Kubenet and Azure CNI affects how these nodes and pods get their IP addresses.

Summary

- **Kubenet**: Nodes get IPs from the VNet, pods get IPs from a separate range, uses NAT.
- **Azure CNI**: Both nodes and pods get IPs from the VNet, direct access, requires more IPs.

By understanding these options, you can choose the best networking setup for your AKS cluster based on your needs.

❖ **Azure Kubernetes Persistent Storage**

When you run applications in containers within your Azure Kubernetes cluster, you might need to store data that persists (remains) even if the container restarts. This is called **persistent storage**. Here's a simple guide to setting it up:

1. Understanding Persistent Storage Options

- **Azure Premium Disk**: Similar to attaching disks to Azure virtual machines. These disks are available to a single pod (container running on a node).
- **Azure Files**: Part of Azure storage accounts, allowing you to create file shares that can be accessed by multiple nodes or pods.

2. Example Application: Let's use a simple application that downloads a file from an Azure storage account and stores it locally. This will help demonstrate persistent storage.

3. Setting Up the Azure Storage Account

- Create a Storage Account:
- Create a Container in the Storage Account

4. Preparing the Application

1. **Download the File Locally**:
   - Write a simple console application that downloads the deployment.yaml file from the Azure storage account to your local machine.
   - Update the file path and connection string in your application to match your local setup and storage account details.
2. **Run the Application Locally**: Ensure the application can download the file to your local machine.

5. Containerizing the Application

1. **Create a Dockerfile**:
2. **Build and Push the Docker Image**: Build the Docker image and push it to a container registry (e.g., Azure Container Registry).

6. Deploying to AKS with Persistent Storage

1. **Create a Persistent Volume Claim (PVC)**:
2. **Deploy the Application with PVC**:
3. **Apply the YAML Files**:

Summary

- **Create a Storage Account**: Set up an Azure storage account and upload a file.
- **Prepare the Application**: Ensure the application can download the file locally.
- **Containerize the Application**: Create a Docker image and push it to a registry.
- **Deploy with Persistent Storage**: Use a PVC to provide persistent storage to your application in AKS.

By following these steps, you can ensure your application has persistent storage in AKS, allowing it to store and retrieve data even if the container restarts. If you have any questions or need further details, feel free to ask!

❖ **Azure Kubernetes Persistent Storage - Build Image**

1. Understanding Persistent Storage in Kubernetes: In Kubernetes, persistent storage is used to store data that needs to be retained even if the pod (a group of containers) is deleted or restarted. This is crucial for applications that require data persistence, like databases.
2. Azure Kubernetes Service (AKS): AKS is a managed Kubernetes service provided by Azure. It simplifies deploying, managing, and scaling containerized applications using Kubernetes.
3. Creating Persistent Storage in AKS: To create persistent storage in AKS, you typically follow these steps:
Step 1: Create a Storage Class: A Storage Class in Kubernetes defines the type of storage (like Azure Disks or Azure Blob Storage) and its properties.
Step 2: Create a Persistent Volume (PV): A Persistent Volume is a piece of storage in the cluster. It can be dynamically provisioned using the Storage Class.
Step 3: Create a Persistent Volume Claim (PVC): A Persistent Volume Claim is a request for storage by a user. It can be used to dynamically provision a PV.
Step 4: Use the PVC in a Pod: Finally, you can use the PVC in your pod to mount the persistent storage.

❖ **Azure Kubernetes Persistent Storage - Using Disks**

1. Purpose: You want to deploy a container in your Azure Kubernetes cluster that downloads a file from Azure Storage and saves it locally using persistent storage (Azure Disk).
2. Steps to Achieve This
Step 1: Get Cluster Credentials: First, you need to get the credentials for your Kubernetes cluster to manage it. Use Azure Cloud Shell and run:
Step 2: Identify Node Resource Group: Find the node resource group for your cluster:
Step 3: Create an Azure Disk: Create a new Azure Disk that will be used for persistent storage:
Step 4: Prepare Deployment YAML: Create a deployment.yml file that specifies how to deploy your container and attach the Azure Disk.
Step 5: Deploy the Application: Deploy your application using the deployment.yml file:
Step 6: Verify the Deployment: Check the logs to ensure your application is running and downloading the file:
Step 7: Confirm Persistent Storage: Even if you delete the pod, the data should remain on the Azure Disk. You can verify this by attaching the disk to another VM and checking the contents.
Summary

1. **Get Cluster Credentials**: Access your Kubernetes cluster.
2. **Identify Node Resource Group**: Find the resource group for your nodes.
3. **Create an Azure Disk**: Set up the disk for persistent storage.
4. **Prepare Deployment YAML**: Define how to deploy your container and attach the disk.
5. **Deploy the Application**: Use Kubernetes to deploy your container.
6. **Verify the Deployment**: Check logs to ensure it's working.
7. **Confirm Persistent Storage**: Verify that data persists even after deleting the pod.

This process ensures your application can download and save files persistently using Azure Disks in your Kubernetes cluster.

❖ **Azure Kubernetes Persistent Storage - File shares**

1. Purpose: You want to use Azure File Shares for persistent storage in your Azure Kubernetes Service (AKS) cluster. This allows your application to download a file from Azure Storage and save it to a shared file system.

2. Steps to Use Azure File Shares in AKS

Step 1: Create an Azure File Share

1. Go to your Azure Storage Account.
2. Navigate to **File shares**.
3. Click on **+ File share**.
4. Name the file share (e.g., share) and click **Create**.

Step 2: Create a Kubernetes Secret

You need to create a secret in Kubernetes to store your Azure Storage account name and key. This allows your AKS cluster to access the Azure File Share.

1. Open Azure Cloud Shell.
2. Set the storage account name and key as environment variables:
3. Create the Kubernetes secret:

Step 3: Prepare Deployment YAML

Create a deployment.yml file that specifies how to deploy your container and mount the Azure File Share.

Step 4: Deploy the Application: Deploy your application using the deployment.yml file

Step 5: Verify the Deployment: Check the logs to ensure your application is running and downloading the file

Step 6: Confirm Persistent Storage: Even if you delete the pod, the data should remain on the Azure File Share. You can verify this by checking the contents of the file share in the Azure portal.

Summary

1. **Create an Azure File Share**: Set up the shared storage in your Azure Storage Account.
2. **Create a Kubernetes Secret**: Store your storage account credentials securely.
3. **Prepare Deployment YAML**: Define how to deploy your container and mount the file share.
4. **Deploy the Application**: Use Kubernetes to deploy your container.
5. **Verify the Deployment**: Check logs to ensure it's working.
6. **Confirm Persistent Storage**: Verify that data persists even after deleting the pod.

This process ensures your application can download and save files persistently using Azure File Shares in your Kubernetes cluster.

Configure and manage virtual networking

❖ **Address spaces**

An address space is a range of IP addresses you set for a Virtual Network (VNet) in Azure. Think of it as a big pool of addresses you can use for your network.

Why It's Important

- **Keeps Networks Separate**: Makes sure your network's addresses don't mix with others.
- **Organizes Your Network**: Helps you divide your network into smaller parts called subnets.
- **Allows Growth**: Lets you add more subnets as needed.

How to Define an Address Space: When you create a VNet, you choose an address range using CIDR notation (like 10.0.0.0/16). This gives you a big pool of addresses.

Example

- **Address Space**: 10.0.0.0/16
  - Starts at 10.0.0.0 and goes up to 10.0.255.255.
  - Gives you 65,536 addresses.

Subnets: Inside this big pool, you can create smaller pools called subnets.

Example:

- **VNet Address Space**: 10.0.0.0/16
  - **Subnet 1**: 10.0.1.0/24 (256 addresses)
  - **Subnet 2**: 10.0.2.0/24 (256 addresses)
  - **Subnet 3**: 10.0.3.0/24 (256 addresses)

❖ **CIDR notation**

CIDR (Classless Inter-Domain Routing) notation is a method for allocating IP addresses and IP routing. In Azure, CIDR notation is used to define IP address ranges for virtual networks (VNets), subnets, and other network resources.

How It Works

- **IP Address**: This is the starting point, like 192.168.1.0.
- **Prefix Length**: The number after the slash (/24) tells you how many bits are used for the network part. The rest are for devices (like computers) in that network.

Example

- 192.168.1.0/24:
  - **192.168.1.0** is the network address.
  - **/24** means the first 24 bits are for the network, leaving 8 bits for devices.
  - This gives you 256 addresses, from 192.168.1.0 to 192.168.1.255.

CIDR in Azure

In Azure, CIDR notation helps define the range of IP addresses for networks and subnets.

1. **Virtual Network (VNet)**: When you create a VNet, you give it an address range using CIDR. For example, 10.0.0.0/16 gives a big range that can be split into smaller parts.
2. **Subnets**: Inside a VNet, you can create subnets by dividing the address range. For example, from 10.0.0.0/16, you can make subnets like 10.0.1.0/24 and 10.0.2.0/24.

❖ IP Address in Azure

Types of IP Addresses

1. **Private IP Address**:
   - Used for communication within your Azure network.
   - Example: If you have a virtual machine (VM) in a subnet with the range 10.0.0.0/24, it might get an address like 10.0.0.4.
2. **Public IP Address**:
   - Used for communication from the internet to your Azure resources.
   - Example: If you want to connect to a VM from your laptop over the internet, the VM needs a public IP address.

Network Interface

- Every VM has a network interface, like the network card in your computer.
- This interface gets a private IP address for internal communication.

Subnets and IP Ranges

- A subnet is a range of IP addresses within your virtual network.
- Each VM in a subnet gets a private IP address from that range.

Communication

- **Private Communication**: VMs within the same virtual network can talk to each other using their private IP addresses.
- **Public Communication**: To connect to a VM from the internet, it needs a public IP address.

IP Address Assignment

- **Dynamic IP**: Changes when you restart the VM.
- **Static IP**: Stays the same even if you restart the VM.

Creating a Public IP Address

- You can create a public IP address as a separate resource in Azure.
- Choose between Basic and Standard tiers.
- Decide if you want a dynamic or static IP address.

Availability

- **Standard Tier**: Offers zone redundancy, making your IP address more reliable.

❖ Adding a secondary network interface

Let's simplify the concept of adding a secondary network interface to a virtual machine (VM) in Azure.

What is a Network Interface?

- **Network Interface**: Think of it as a virtual network card for your VM. Just like your computer has a network card to connect to the internet, a VM has a network interface to handle all its network traffic.

Why Add a Secondary Network Interface?

- **Separate Traffic**: You might want to separate different types of traffic. For example, one network interface can handle internet traffic, while another handles internal network traffic.
- **Security**: By separating traffic, you can better manage and secure the data coming in and out of your VM.

Scenario Example

Imagine you have a virtual network with three subnets (A, B, and C):

- **Subnet A**: Contains one VM with two network interfaces.
- **Subnet B and C**: Each contains two VMs with one network interface each.

Use Case

- **Primary Subnet (Subnet A)**: This subnet faces the internet. The VM here has two network interfaces:
  - **First Network Interface**: Handles traffic from the internet.
  - **Second Network Interface**: Passes filtered traffic to other VMs in subnets B and C.

❖ **Network Security Groups (NSGs) - Priority Setting**

Network Security Groups (NSGs) in Azure help control the flow of network traffic to and from your Azure resources. They do this by using rules that either allow or deny traffic based on certain criteria. The priority setting in these rules is crucial for determining which rules are applied first.

How Priority Works

1. **Priority Numbers**:
   - Each rule in an NSG has a priority number.
   - Lower numbers mean higher priority. For example, a rule with priority 100 will be applied before a rule with priority 200.
2. **Rule Evaluation**:
   - When a request comes in, NSG evaluates the rules in order of priority, starting from the lowest number.
   - Once it finds a rule that matches the request, it applies that rule and stops evaluating further rules.

Example Scenario

1. **Initial Rules**:
   - **Rule 1**: Allow HTTP traffic (port 80) with priority 310.
   - **Rule 2**: Deny traffic on ports 75-85 with priority 220.
2. **Impact of Rules**:
   - When a request comes in on port 80, NSG first checks Rule 2 (priority 220).
   - Since port 80 falls within the range 75-85, Rule 2 matches and denies the request.
   - NSG stops evaluating further rules, so Rule 1 (priority 310) is not considered.

❖ **Network Security Groups (NSGs) - Outbound Rules**

Network Security Groups (NSGs) in Azure help control the flow of network traffic to and from your Azure resources. Outbound rules specifically manage the traffic leaving your resources.

What Are Outbound Rules?

- **Outbound Rules**: These rules determine what traffic is allowed to leave your Azure resources, such as virtual machines (VMs).

How Outbound Rules Work

1. **Priority Numbers**:
   - Each outbound rule has a priority number.

- Lower numbers mean higher priority. For example, a rule with priority 100 will be applied before a rule with priority 200.

2. **Rule Evaluation**:
- When traffic tries to leave a resource, NSG evaluates the outbound rules in order of priority, starting from the lowest number.
- Once it finds a rule that matches the traffic, it applies that rule and stops evaluating further rules.

Example Scenario

1. **Default Outbound Rules**: By default, Azure allows all outbound traffic from your resources. This means your VMs can connect to the internet and other Azure services without any restrictions.
2. **Custom Outbound Rules**: You can create custom outbound rules to restrict or allow specific types of traffic.

❖ **Azure load balancer service, Basic Load Balancer - Setup, Basic Load Balancer - Deployment, Basic Load Balancer - Configuration, Basic Load Balancer - NAT rules, Basic Load Balancer - VM Scale Set, Basic Load Balancer - VM Scale Set**

Azure Load Balancer Service: Azure Load Balancer is a service that distributes incoming network traffic across multiple virtual machines (VMs) to ensure high availability and reliability of your applications. It operates at the transport layer (Layer 4) and can handle both inbound and outbound traffic.

Backend Pools of Azure Load Balancer

A **backend pool** in Azure Load Balancer is a group of servers (like virtual machines) that receive the traffic distributed by the load balancer. Think of it as the team of servers that handle the work when users access your application.

Purpose: The backend pool is where the load balancer sends incoming traffic. It ensures that no single server gets overwhelmed by distributing the traffic evenly.

Basic Load Balancer: The Basic Load Balancer is a simple and cost-effective option for small-scale applications or development/testing environments. It provides basic load balancing features without advanced capabilities like zone redundancy.

Setup of Basic Load Balancer

1. **Create a Load Balancer**:
- Go to the Azure portal.
- Search for “Load Balancer” and click “Create”.
- Choose “Basic” as the SKU.
- Fill in the required details like name, region, and resource group.

2. **Create a Backend Pool**:
- A backend pool is a group of VMs that will receive the traffic.
- Add the VMs you want to include in the backend pool.

3. **Create a Health Probe**:
- Health probes check the status of the VMs in the backend pool.

- Configure the probe with settings like protocol (TCP/HTTP), port, and interval.

4. **Create a Load Balancing Rule**:
- Define how traffic is distributed to the VMs.
- Specify the frontend IP, backend pool, protocol, and port.

Deployment of Basic Load Balancer

1. **Deploy VMs**:
- Deploy the VMs that will be part of the backend pool.
- Ensure they are in the same virtual network as the load balancer.
2. **Associate VMs with Backend Pool**:
- Add the deployed VMs to the backend pool of the load balancer.

Configuration of Basic Load Balancer

1. **Frontend IP Configuration**:
- Configure the frontend IP address that will receive the incoming traffic.
- This can be a public or private IP address.
2. **Backend Pool Configuration**:
- Ensure the VMs in the backend pool are correctly configured and healthy.
3. **Health Probe Configuration**:
- Set up health probes to monitor the status of the VMs.
- Adjust settings like interval and timeout as needed.
4. **Load Balancing Rules**:
- Define rules to distribute traffic based on protocol and port.
- Example: A rule to distribute HTTP traffic on port 80.

NAT Rules

Network Address Translation (NAT) rules allow you to map specific ports on the load balancer's frontend IP to ports on the backend VMs. This is useful for scenarios where you need to access individual VMs directly.

1. **Create Inbound NAT Rules**:
- Go to the load balancer settings.
- Add inbound NAT rules to map frontend ports to backend VM ports.
- Example: Map port 8080 on the load balancer to port 80 on a specific VM.

VM Scale Set with Basic Load Balancer

A VM Scale Set allows you to automatically scale the number of VMs based on demand. When combined with a Basic Load Balancer, it ensures that traffic is evenly distributed across the scaled VMs.

1. **Create a VM Scale Set**:
- Go to the Azure portal and create a VM Scale Set.
- Configure the scale set with the desired number of VMs and scaling policies.
2. **Associate with Load Balancer**:
- During the creation of the VM Scale Set, associate it with the Basic Load Balancer.
- Ensure the VMs in the scale set are added to the backend pool.

Example Scenario

1. **Setup**:

- Create a Basic Load Balancer named “MyBasicLB”.
- Deploy three VMs named “VM1”, “VM2”, and “VM3”.

2. **Configuration**:
- Add “VM1”, “VM2”, and “VM3” to the backend pool of “MyBasicLB”.
- Create a health probe to check HTTP status on port 80.
- Define a load balancing rule to distribute HTTP traffic on port 80.

3. **NAT Rules**:
- Create an inbound NAT rule to map port 8080 on “MyBasicLB” to port 80 on “VM1”.

4. **VM Scale Set**:
- Create a VM Scale Set named “MyScaleSet” with a minimum of 2 and a maximum of 5 VMs.
- Associate “MyScaleSet” with “MyBasicLB”.

❖ **Azure Load Balancer: Standard SKU**

Azure Load Balancer - Standard SKU

The Standard SKU of Azure Load Balancer is designed for production environments and offers advanced features, higher availability, and better performance compared to the Basic SKU. In the context of Azure Load Balancer, **SKU** stands for **Stock Keeping Unit**. It represents different versions or types of the load balancer with varying features and capabilities. Let’s dive into the details.

Key Features of Standard SKU

1. **High Availability and Scalability**:
- **Zone Redundancy**: Ensures high availability by distributing traffic across multiple availability zones. This means if one zone goes down, the load balancer can still route traffic to healthy instances in other zones.
- **Larger Backend Pools**: Supports larger backend pools, allowing you to scale your applications more effectively.

2. **Enhanced Security**:
- **Integration with Azure Private Link**: Allows secure access to services over a private network, reducing exposure to the public internet.
- **Advanced Network Security**: Provides more granular control over network traffic with support for Network Security Groups (NSGs) and user-defined routes.

3. **Performance and Reliability**:
- **Higher Throughput**: Can handle more significant amounts of traffic, making it suitable for high-traffic applications.
- **Consistent Performance**: Offers better and more consistent performance compared to the Basic SKU.

4. **Global Reach**: **Global Load Balancing**: Supports global load balancing scenarios, allowing you to distribute traffic across multiple regions for better performance and reliability.

❖ **Standard Load Balancer - Outbound Connectivity, Multiple Backend Pools, and Multiple Frontend IP Addresses**

The Standard SKU of Azure Load Balancer is designed for more demanding applications and offers advanced features. Let's break down the concepts of outbound connectivity, multiple backend pools, and multiple frontend IP addresses in simple terms.

Outbound Connectivity: Outbound connectivity means your virtual machines (VMs) can connect to the internet or other external services.

1. **Default Outbound Access**:
   - By default, VMs can connect to the internet using a public IP address provided by the load balancer.
2. **Outbound Rules**:
   - You can create rules to control how VMs connect to the internet.
   - Example: Allow all VMs in a group to access the internet on specific ports like 80 (HTTP) and 443 (HTTPS).
3. **SNAT (Source Network Address Translation)**:
   - Allows multiple VMs to share a single public IP address for outbound connections.
   - Helps save public IP addresses and simplifies management.

Multiple Backend Pools: Multiple backend pools let you group different sets of VMs that will receive traffic from the load balancer.

1. **Creating Backend Pools**:
   - You can create several backend pools within one load balancer.
   - Each pool can have different VMs or VM scale sets.
2. **Use Cases**:
   - **Separate Services**: Group different services or applications into different pools.
   - **Scaling**: Scale each pool independently based on demand.

Multiple Frontend IP Addresses

Multiple frontend IP addresses allow you to assign more than one IP address to the load balancer.

Example Scenario

1. **Setup**: Create a Standard Load Balancer named "MyStandardLB". Deploy six VMs named "VM1" to "VM6".
2. **Configuration**:
   - **Backend Pools**: Create two backend pools: "BackendPool1" (VM1, VM2, VM3) and "BackendPool2" (VM4, VM5, VM6).
   - **Frontend IP Addresses**: Create two frontend IP configurations: "FrontendIP1" (public IP) and "FrontendIP2" (private IP).
   - **Outbound Rules**: Create an outbound rule to allow all VMs in "BackendPool1" to access the internet on ports 80 and 443.
3. **Load Balancing Rules**: Define rules to distribute traffic from "FrontendIP1" to "BackendPool1" and from "FrontendIP2" to "BackendPool2".

- **Azure Application Gateway, Azure Application Gateway - Components, Azure Application Gateway - URL Routing, Azure Application Gateway - Multiple Sites, Azure Application Gateway - Multiple Sites – Gateway**

Azure Application Gateway is a service that helps manage web traffic to your applications. It works at the application layer (Layer 7) and provides advanced features like routing traffic based on URLs, SSL termination, and more.

- **Components of Azure Application Gateway**

1. **Frontend IP Configuration**:
   - This is the IP address that users connect to.
   - It can be a public or private IP address.
2. **Listeners**:
   - These listen for incoming traffic on the frontend IP.
   - You can set them to listen on specific ports (like 80 for HTTP or 443 for HTTPS).
3. **Backend Pools**:
   - Groups of servers (like VMs or web apps) that receive the traffic.
   - These are the servers where your application is running.
4. **HTTP Settings**:
   - Define how traffic is sent to the backend servers.
   - Includes settings like which port to use and whether to keep sessions sticky (cookie-based affinity).
5. **Rules**:
   - Determine how traffic is routed from the frontend to the backend pools.
   - Can include rules based on URL paths or hostnames.
6. **Health Probes**:
   - Check if the backend servers are healthy and can handle traffic.
   - Only sends traffic to healthy servers.

- **URL Routing:** URL routing allows you to direct traffic based on the URL path. This is useful for sending different types of requests to different backend servers.

1. **Path-Based Routing**:
   - Routes traffic based on the URL path.
   - Example: Requests to /images go to one set of servers, while requests to /videos go to another.
2. **Host-Based Routing**:
   - Routes traffic based on the hostname in the URL.
   - Example: Requests to www.example.com go to one set of servers, while requests to api.example.com go to another.

- **Multiple Sites:** You can host multiple websites on a single Application Gateway. This is useful for managing multiple applications with one gateway.

1. **Multiple Site Hosting**:
   - Host different websites on the same Application Gateway.
   - Each site can have its own listener, rules, and backend pools.
2. **Configuration**:
   - Create multiple listeners, each with its own IP and port.
   - Define rules for each listener to route traffic to the correct backend pool.

- **Multiple Sites – Gateway:** When hosting multiple sites, the Application Gateway can route traffic to different backend pools based on the site being accessed.

1. **Listeners for Multiple Sites**:
   - Create separate listeners for each site.
   - Each listener can have its own IP and port.
2. **Routing Rules**:
   - Define routing rules for each listener.
   - Use URL path-based or host-based routing to direct traffic to the correct backend pool.

- **Example Scenario**

1. **Setup**:
   - Create an Application Gateway named "MyAppGateway".
   - Configure a public IP for the gateway.
2. **Components**:
   - **Listeners**: Create listeners for www.example.com and api.example.com.
   - **Backend Pools**: Create backend pools for the main website and the API.
   - **HTTP Settings**: Define how traffic is forwarded to the backend pools.
   - **Health Probes**: Set up health probes to monitor the backend servers.
3. **URL Routing**:
   - **Path-Based Routing**: Route traffic to /images to one backend pool and /videos to another.
   - **Host-Based Routing**: Route traffic to www.example.com to the main website backend pool and api.example.com to the API backend pool.
4. **Multiple Sites**:
   - Configure multiple listeners and routing rules to host both www.example.com and api.example.com on the same Application Gateway.

❖ **Azure Bastion**

Azure Bastion is a service that lets you securely connect to your virtual machines (VMs) in Azure without exposing them to the internet. You can use it to access your VMs directly from the Azure portal using RDP (Remote Desktop Protocol) or SSH (Secure Shell).

Key Features

1. **Secure Access**:
   - Connect to your VMs without needing a public IP address.
   - All connections are made securely through the Azure portal.
2. **Easy to Use**:
   - No need for extra software or complex setups.
   - Access your VMs directly from your web browser.
3. **Managed by Azure**:
   - Azure takes care of the maintenance and scaling.
   - You don't need to worry about managing the service.
4. **Enhanced Security**:
   - Keeps your VMs off the public internet, reducing the risk of attacks.
   - Uses Azure's security features to ensure safe access.

How Azure Bastion Works

1. **Deployment**:

- You set up Azure Bastion in the same virtual network (VNet) as your VMs.
- It creates a secure connection point within your VNet.

2. **Accessing VMs**:
- Go to the Azure portal and find the VM you want to connect to.
- Click the "Connect" button and choose "Bastion".
- Enter your login details to connect securely to the VM.

❖ **Explain Point-to-Site VPN, Point-to-Site VPN - Certificates for authentication, Point-to-Site VPN - Establishing the connection**

A Point-to-Site (P2S) VPN allows you to securely connect your individual computer to an Azure Virtual Network (VNet). This is useful for remote workers who need to access resources in Azure from anywhere.

Certificates for Authentication

1. **Generate Certificates**:
- **Root Certificate**: This authenticates VPN clients. Create a self-signed root certificate using tools like PowerShell.
- **Client Certificates**: These are derived from the root certificate and installed on client machines.
  - Export the root certificate's public key and upload it to the VPN gateway.
  - Generate client certificates signed by the root certificate and install them on client machines.

2. **Upload Root Certificate to Azure**:
- Go to the Point-to-site configuration of your Virtual Network Gateway.
- Upload the public key of the root certificate.

Example Scenario

1. **Setup**:
- Create a Virtual Network Gateway named "MyVNetGateway".
- Configure Point-to-site settings with an address pool of 172.16.0.0/24 and SSTP tunnel type.

2. **Certificates**:
- Generate a root certificate named "RootCert".
- Create client certificates for each remote worker.
- Upload the root certificate's public key to Azure.

3. **Connection**:
- Download the VPN client configuration package.
- Install the VPN client and client certificates on remote workers' computers.
- Remote workers use the VPN client to connect securely to the Azure VNet.

❖ **Azure Virtual WAN, Virtual WAN - Virtual Hub, zure Virtual WAN Hub - Azure virtual networks, Azure Virtual WAN Hub - Point-to-Site connections, Azure Virtual WAN - VPN Sites**

Azure Virtual WAN is a networking service that provides a unified and simplified way to connect your on-premises networks, branch offices, and Azure VNets. Azure Virtual WAN is a service that helps you connect different networks, like your office networks and Azure Virtual Networks (VNets), in a simple and unified way. It acts like a central hub that connects everything together.

- **Virtual WAN - Virtual Hub:** A **Virtual Hub** is the central point in Azure Virtual WAN where all your network connections come together. Think of it as the main station where all the trains (networks) meet.

1. **Components of a Virtual Hub:**
   - Hub VNet: The virtual network linked to the hub.
   - Hub Gateway: Manages connections to on-premises networks and other VNets.
   - Hub Route Table: Defines how traffic is routed within the hub.

- **Connecting Azure Virtual Networks to the Virtual Hub:** You can connect multiple Azure VNets to a Virtual Hub, allowing them to communicate with each other easily.
  Benefits:
  Centralized Management: Manage all your VNet connections from one place.
  Improved Connectivity: VNets connected to the hub can talk to each other efficiently.

- **Point-to-Site Connections:** Point-to-Site (P2S) connections allow individual devices (like laptops) to connect securely to the Virtual Hub, providing remote access to Azure resources.
- **VPN Sites:** VPN Sites represent your on-premises locations (like your office) that connect to the Azure Virtual WAN using Site-to-Site VPN connections.

Example Scenario

1. **Setup**:
   - Create a Virtual WAN named “MyVirtualWAN”.
   - Create a Virtual Hub named “MyVirtualHub” within the Virtual WAN.
2. **Connecting VNets**: Connect VNets “VNet1” and “VNet2” to “MyVirtualHub”.
3. **Point-to-Site Connections**: Configure Point-to-Site connections for remote workers to connect to “MyVirtualHub”.
4. **VPN Sites**:
   - Create a VPN site named “MyOnPremSite” representing your on-premises network.
   - Connect “MyOnPremSite” to “MyVirtualHub” using a Site-to-Site VPN connection.

❖ **User Defined Routes, User Defined Routes - Route Table, User Defined Routes - Enable forwarding**

User Defined Routes (UDRs) in Azure allow you to control the routing of network traffic within your virtual network (VNet). By creating custom routes, you can direct traffic to specific destinations, such as network virtual appliances (NVAs) or on-premises networks.

- **User Defined Routes - Route Table:** A **Route Table** is a collection of routes that define how traffic should be directed within your VNet. You can associate a route table with one or more subnets in your VNet. A **Route Table** is a set of rules that tells your network where to send traffic.
- **Enable Forwarding: Enable IP Forwarding** allows a VM to act like a router and forward traffic to other destinations.

Example Scenario

1. **Setup**:
   - Create a route table named "MyRouteTable".
   - Add a route to direct traffic destined for 10.0.1.0/24 to a virtual appliance with IP 10.0.0.4.
2. **Associating the Route Table**:
   - Associate "MyRouteTable" with the subnet SubnetA in your VNet.
3. **Enable IP Forwarding**:
   - Enable IP forwarding on the VM with IP 10.0.0.4.
   - Configure the VM to forward traffic.

❖ **AZure Network Watcher, Network Watcher - Connection Troubleshoot, Network Watcher - Connection Monitor, Network Watcher - IP Flow Verify, Network Watcher - Next hop, Network Watcher - NSG Diagnostic**

Azure Network Watcher: Azure Network Watcher is like a toolkit for checking and fixing network issues in your Azure environment. It helps you see how your network is performing and find any problems.

- **Connection Troubleshoot: Connection Troubleshoot** is like a doctor for your network connections. Imagine you have a virtual machine (VM) in Azure, and it's having trouble connecting to a website. Connection Troubleshoot checks the path from your VM to the website and tells you where the problem is, like if there's a roadblock or a slow route. **Example:** Your VM can't reach www.example.com. Connection Troubleshoot will check the path and tell you if there's a firewall blocking it or if the route is too slow.

- **Connection Monitor: Connection Monitor** is like a security camera for your network connections. It keeps an eye on the connections between your Azure VMs and other machines, whether they're in Azure or on-premises. It checks if the connections are working well and logs any issues.

Example: You want to ensure your Azure VM can always connect to your on-premises server. Connection Monitor will continuously check this connection and alert you if there's a problem.

- **IP Flow Verify: IP Flow Verify** is like a traffic cop for your network. It checks if a specific type of network traffic is allowed or blocked based on your security rules.
  Example: You want to know if traffic from your VM to a specific IP address is allowed. IP Flow Verify will tell you if it's allowed or blocked and which rule is responsible.

- **Next Hop: Next Hop** is like a GPS for your network traffic. It tells you the next stop for your data packets on their way to a destination.
  Example: You want to know the next hop for traffic from your VM to 192.168.1.1. Next Hop will tell you the next device or route the traffic will take.

- **NSG Diagnostic: NSG Diagnostic** is like a detective for your network security rules. It helps you understand why certain traffic is allowed or blocked by simulating the traffic flow and showing you the rules in action.
  Example: You want to know why traffic from your VM to a specific IP is blocked. NSG Diagnostic will simulate the traffic and show you the rule that's blocking it.

❖ **Azure Public DNS, Azure Private DNS, Azure Private DNS - Peered networks**

- **Azure Public DNS: Azure Public DNS** is a service that provides DNS hosting for your domains. It allows you to manage DNS records using the same credentials, APIs, tools, and billing as your other Azure services. This service is used to resolve domain names to IP addresses for resources that are accessible over the internet.
  Example: If you own the domain example.com, you can use Azure Public DNS to manage DNS records for this domain. When someone types www.example.com in their browser, Azure Public DNS resolves this name to the IP address of your web server.

- **Azure Private DNS: Azure Private DNS** provides a secure and reliable DNS service for your virtual networks. It manages and resolves domain names within your virtual network without needing a custom DNS solution. This service allows you to use custom domain names instead of the default Azure-provided names.
  Example: If you have a virtual network with several VMs, you can create a private DNS zone like internal.example.com. This way, VMs can communicate using names like vm1.internal.example.com instead of IP addresses.

- **Azure Private DNS - Peered Networks:** When you have multiple virtual networks that are peered (connected), **Azure Private DNS** can be used to manage DNS resolution across these networks. This means that VMs in different peered virtual networks can resolve each other's domain names.

Example: Suppose you have two virtual networks, VNet1 and VNet2, which are peered. You can link both VNets to the same private DNS zone. This allows a VM in VNet1 to resolve the domain name of a VM in VNet2 using the private DNS zone.

❖ **What is Bicep in azure?**

- Azure Bicep is a language designed to make it easier to deploy and manage resources in Azure. Here's a simple breakdown:
- Purpose: Bicep is used to define the infrastructure you want to create in Azure, like virtual machines, storage accounts, and databases.
- Declarative Syntax: Instead of writing complex scripts, you describe what you want, and Bicep takes care of the rest. For example, you can specify that you need a storage account, and Bicep will handle the details.
- Simpler than JSON: Bicep is easier to read and write compared to JSON, which was previously used for Azure Resource Manager (ARM) templates. This makes it more accessible, even if you're not a programming expert1.
- Consistency: By using Bicep, you ensure that your infrastructure is deployed in a consistent manner every time. This reduces errors and makes it easier to manage your resources1.
- Integration with Tools: Bicep integrates well with tools like Visual Studio Code, providing features like syntax highlighting and error checking to help you write your templates
-

❖ **What are Azure Resource Manager templates**

Azure Resource Manager (ARM) templates are JSON files that define the infrastructure and configuration for your Azure projects. They allow you to deploy, manage, and organize Azure resources in a consistent and repeatable manner. Here are the key points:

Key Features of ARM Templates

1. **Declarative Syntax**: You describe the desired state of your infrastructure without specifying the sequence of commands to achieve it.
2. **Idempotent**: You can deploy the same template multiple times and get the same result, ensuring consistency.
3. **Infrastructure as Code**: ARM templates enable you to manage your infrastructure using code, which can be versioned and stored in source control.
4. **Modular and Reusable**: You can create modular templates and reuse them across different projects.
5. **Integration**: ARM templates integrate with other Azure services like Azure Policy and Azure DevOps for compliance and CI/CD pipelines

Benefits

- **Consistency**: Ensures that your infrastructure is deployed in a consistent manner.
- **Automation**: Automates the deployment process, reducing manual errors.
- **Scalability**: Easily scale your infrastructure by modifying the template and redeploying.

❖ **What is the Azure Monitor Service**

Azure Monitor is a comprehensive monitoring solution for collecting, analyzing, and responding to data from your cloud and on-premises environments. Here are the key points:

- **Data Collection**: It gathers metrics and logs from various sources, including applications, virtual machines, databases, and more.
- **Analysis and Visualization:** Provides tools to analyze and visualize the collected data, helping you understand the performance and health of your systems.
- **Alerts and Automation:** Allows you to set up alerts and automate responses to specific conditions, ensuring proactive management of your infrastructure.
- **Integration:** Works seamlessly with other Azure services and third-party tools for a unified monitoring experience

❖ **What is a Log Analytics Workspace**

A Log Analytics Workspace in Azure is a central repository where you can collect, store, and analyze log data from various sources.

Key Features

1. **Data Collection:** It gathers log data from Azure resources, on-premises systems, and other cloud environments.
2. **Centralized Management:** Acts as a single place to manage and query all your log data.
3. **Integration**: Works with Azure Monitor, Microsoft Sentinel, and other Azure services for enhanced monitoring and security.
4. **Data Retention:** Allows you to retain log data for analysis over different periods, supporting both short-term and long-term retention[12].

Example Use Cases

- **Monitoring:** Track the performance and health of your applications and infrastructure.
- **Security:** Detect and respond to security threats using integrated tools like Microsoft Sentinel.
- **Compliance:** Maintain logs for auditing and compliance purposes.

How It Works

- **Log Tables:** Data is stored in tables within the workspace, which you can query using Kusto Query Language (KQL).
- **Custom Queries:** Write custom queries to analyze data and create visualizations or alerts based on specific conditions.

❖ **Azure VM Insights**

Azure VM Insights is a feature of Azure Monitor that provides comprehensive monitoring for your virtual machines (VMs) and virtual machine scale sets. Here are the key aspects:

Key Features

1. **Performance Monitoring**: Collects and analyzes performance data from your VMs, including CPU, memory, disk, and network usage.
2. **Dependency Mapping:** Automatically discovers and maps the dependencies between your VMs and other resources, helping you understand how your applications are interconnected[1].
3. **Predefined Workbooks:** Offers a set of predefined workbooks that visualize performance trends and other important metrics over time[1].
4. **Multi-Environment Support:** Supports monitoring for Azure VMs, on-premises VMs, and VMs in other cloud environments through Azure Arc[2].
5. **Alerts and Automation:** Allows you to set up alerts based on specific conditions and automate responses to maintain the health and performance of your VMs[3].

How to Access VM Insights

- **Azure Portal**: You can access VM Insights from the Azure portal by selecting your virtual machine and navigating to the "Insights" section.
- **Aggregated View**: Use Azure Monitor to get an aggregated view of multiple VMs, making it easier to manage large environments[1].

Example Use Cases

- **Performance Troubleshooting**: Identify and resolve performance bottlenecks in your VMs.
- **Application Dependency Mapping**: Visualize and manage the dependencies between different components of your applications.
- **Proactive Monitoring**: Set up alerts to get notified about potential issues before they impact your users.

❖ **What is the Azure Backup feature**

Azure Backup is a cloud-based service that provides simple, secure, and cost-effective solutions to back up your data and recover it from the Microsoft Azure cloud. Here are the key features:

Key Features

1. **Data Protection**: Azure Backup protects your data by backing it up to the cloud, ensuring that you can recover it in case of accidental deletion, corruption, or disasters.
2. **Scalability**: It leverages the power and scale of Azure to provide high availability without the need for complex on-premises backup solutions[1].
3. **Security**: Data is encrypted both in transit and at rest, ensuring that your backups are secure. Azure Backup also supports features like multi-user authorization and immutability to prevent unauthorized access and tampering[2].
4. **Centralized Management**: You can manage and monitor your backups through a centralized interface, making it easier to keep track of your backup status and health[1].
5. **Cost-Effective**: Azure Backup offers a pay-as-you-go model, allowing you to scale your backup storage as needed without upfront costs[3].

Supported Scenarios

- **On-Premises**: Back up files, folders, system state, and on-premises VMs (Hyper-V and VMware).
- **Azure VMs**: Back up entire Windows/Linux VMs or specific files and folders.
- **Azure Services**: Back up Azure Managed Disks, Azure Files shares, SQL Server in Azure VMs, SAP HANA databases, and more[1].

Example Use Cases

- **Disaster Recovery**: Ensure business continuity by recovering critical data after a disaster.
- **Compliance**: Meet regulatory requirements by retaining backups for specified periods.
- **Operational Backup**: Regularly back up operational data to prevent data loss.

❖ **Azure Backup reports**

Azure Backup Reports provide detailed insights into your backup environment, helping you monitor, analyze, and optimize your backup operations. Here are the key aspects:

Key Features

1. Data Collection: Azure Backup Reports collect data from various sources, including Azure VMs, SQL in Azure VMs, SAP HANA in Azure VMs, and more[1].
2. Visualization: Uses Azure Monitor Logs and Azure Workbooks to visualize data, making it easier to understand trends and performance[1].

3. Customizable Reports: You can customize reports to focus on specific metrics or time periods, and even export them to Excel for further analysis[1].
4. Centralized Management: View reports across multiple vaults, subscriptions, and regions from a single interface[1].
5. Alerts and Notifications: Set up alerts for critical backup incidents and receive notifications via email or other channels[2].

How to Configure Azure Backup Reports

1. Sign in to the Azure Portal: Navigate to the Azure portal and go to the Backup Center.
2. Select Reports: In the Backup Center, select the Reports tab.
3. Configure Reports: Follow the prompts to configure your reports, including selecting the data sources and setting up any necessary permissions[1].

Example Use Cases

- Auditing: Track and audit backup and restore operations to ensure compliance.
- Storage Forecasting: Allocate and forecast cloud storage consumption to optimize costs.
- Performance Monitoring: Identify key trends and performance issues at different levels of granularity

❖ **Azure Site Recovery**

Azure Site Recovery (ASR) is a powerful service designed to ensure business continuity and disaster recovery.

Key Points of Azure Site Recovery

1. **Business Continuity:** ASR helps keep critical applications running even if the primary server or data center goes down. This is crucial for businesses that rely on applications to generate revenue, like e-commerce sites.
2. **Replication:** ASR continuously replicates data from your primary server to a secondary server or data center. This ensures that if the primary server fails, the secondary server can take over with the most recent data.
3. **Disaster Recovery:** In case of an unplanned outage (like a server crash) or a planned outage (like maintenance), ASR can switch operations to the secondary server seamlessly.
4. **Versatility**: ASR works with various environments, including:

- Physical servers
- Virtual machines (VMs) on Hyper-V or VMware
- Azure VMs

5. **Ease of Use**: Instead of buying and managing complex software for data replication, businesses can use ASR to handle this process efficiently.

How It Works

- **Primary and Secondary Data Centers:** Businesses typically have a primary data center where their main servers are located and a secondary data center as a backup.
- **Continuous Data Replication:** ASR ensures that any changes made on the primary server are continuously replicated to the secondary server.
- **Failover and Failback:** If the primary server goes down, ASR can failover to the secondary server. Once the primary server is back online, you can failback to it.

Benefits

- **Minimized Downtime**: Ensures that applications remain available, minimizing downtime and potential revenue loss.
- **Simplified Management:** Reduces the complexity of managing disaster recovery solutions.
- **Cost-Effective**: Eliminates the need for additional software and infrastructure for disaster recovery.

❖ **Azure AD Premium licensing**

Azure Active Directory (Azure AD) Premium is available in two main versions: Premium P1 and Premium P2. These versions offer advanced features for identity and access management, enhancing security and productivity for organizations.

Azure AD Premium P1

- **Features:** Includes advanced administration features, dynamic groups, self-service password reset, and conditional access policies.
- **Use Case:** Suitable for organizations that need enhanced identity management and security features.
- **Pricing:** Typically costs around $6 per user per month[1].

Azure AD Premium P2

- **Features:** Includes all P1 features plus additional capabilities like Identity Protection, Privileged Identity Management (PIM), and access reviews.
- **Use Case:** Ideal for organizations requiring advanced security and governance features.
- **Pricing:** Typically costs around $9 per user per month[1]**.**

Licensing Options

- **Standalone:** Both P1 and P2 can be purchased as standalone licenses.
- **Bundles:** They are also included in bundles like Microsoft 365 E3 (P1) and Microsoft 365 E5 (P2)[2].

❖ **Management Groups:**

- **Purpose:** Management groups are used to logically manage multiple Azure subscriptions, especially for companies with different departments or environments[1].
- **Structure:** There is a default root group called the tenant root group[23]. Under this, you can create management groups and add subscriptions to them.
- **Advantages:** Role-based access control and Azure policies can be assigned at the management group level, trickling down to all subscriptions and resources within the group.
- **Implementation:** Creating a management group is simple, requiring just a group ID and name[4]. Existing subscriptions can be added to these groups, and role assignments can be managed at this level.

❖ **Azure Policy Service**

Azure Policy is a governance tool that helps you enforce organizational standards and assess compliance at scale.

Key Features

1. **Enforce Standards:** Ensure that resources across your Azure subscriptions comply with your organization's standards.
2. **Built-in Policies:** Utilize a wide range of pre-built policies for common governance scenarios.
3. **Custom Policies:** Create custom policies tailored to your specific needs.
4. **Compliance Assessment:** Continuously monitor and assess the compliance of your resources.

Example Use Cases

- **Location Restrictions**: Ensure that resources are only deployed in specific regions**.**
- **Tag Enforcement:** Require that certain tags are applied to resources for better management and billing.
- **Security Policies:** Enforce security configurations, such as requiring encryption for storage accounts.

How It Works

1. **Policy Definition:** Define what you want to enforce or audit. This can be done using JSON.

2. **Assignment:** Assign the policy to a scope, such as a subscription, resource group, or management group.
3. **Evaluation:** Azure Policy continuously evaluates resources to ensure they comply with the assigned policies.
4. **Remediation:** If resources are found to be non-compliant, you can take corrective actions, either manually or automatically.

❖ **Data Redundancy Options in Azure Storage Accounts**

When creating an Azure Storage account, you have several redundancy options to ensure your data is safe, available, and durable.

1. Locally-Redundant Storage (LRS): 3 copies

- Description: LRS keeps three copies of your data within a single data center in the same region.
- Use Case: Suitable for protecting against local hardware failures, such as a server rack or drive failure.
- Example: If you have a storage account in the Central US region, LRS will store three copies of your data within that data center.

2. Zone-Redundant Storage (ZRS): 3 Copies

- Description: ZRS replicates your data across three different availability zones within the same region.
- Use Case: Protects against data center-level failures, ensuring data availability even if one data center goes down.
- Example: Your data is stored in three separate data centers within the same region, each with independent power, cooling, and networking.

3. Geo-Redundant Storage (GRS): 6 copies

- Description: GRS replicates your data to a secondary region, hundreds of miles away from the primary location, in addition to keeping three copies in the primary region.
- Use Case: Provides protection against regional outages, ensuring data availability even if an entire region goes down.
- Example: If your primary region is Central US, GRS will also store your data in a secondary region, such as East US.

4. Read-Access Geo-Redundant Storage (RA-GRS):

- Description: RA-GRS offers the same redundancy as GRS but also provides read access to the data in the secondary region.
- Use Case: Ideal for applications that require high availability and read access to data even during a regional outage.

- Example: Your data is available for read operations in both the primary and secondary regions.

5. **Geo-zone-redundant storage (GZRS): 6 copies**

- in Azure is a robust storage option designed to ensure high availability and durability of your data. Here's a brief overview:
- Replication Across Zones: GZRS replicates your data synchronously across three Azure availability zones within the primary region. Each zone is a separate physical location with independent power, cooling, and networking1.
- Secondary Region Replication: In addition to the primary region replication, GZRS also asynchronously copies your data to a secondary region. Within this secondary region, your data is further replicated synchronously three times2.
- Durability and Availability: This setup provides a high level of durability (at least 99.9999999999% over a year) and ensures that your data remains accessible even in the event of a regional outage1.
- Read Access: You can also configure your storage account for read-access geo-zone-redundant storage (RA-GZRS), which allows read access to the data in the secondary region during an outage in the primary region

6.

Summary

- LRS: Protects against local hardware failures.
- ZRS: Protects against data center-level failures.
- GRS: Protects against regional outages.
- RA-GRS: Provides read access to data in secondary regions during regional outages.