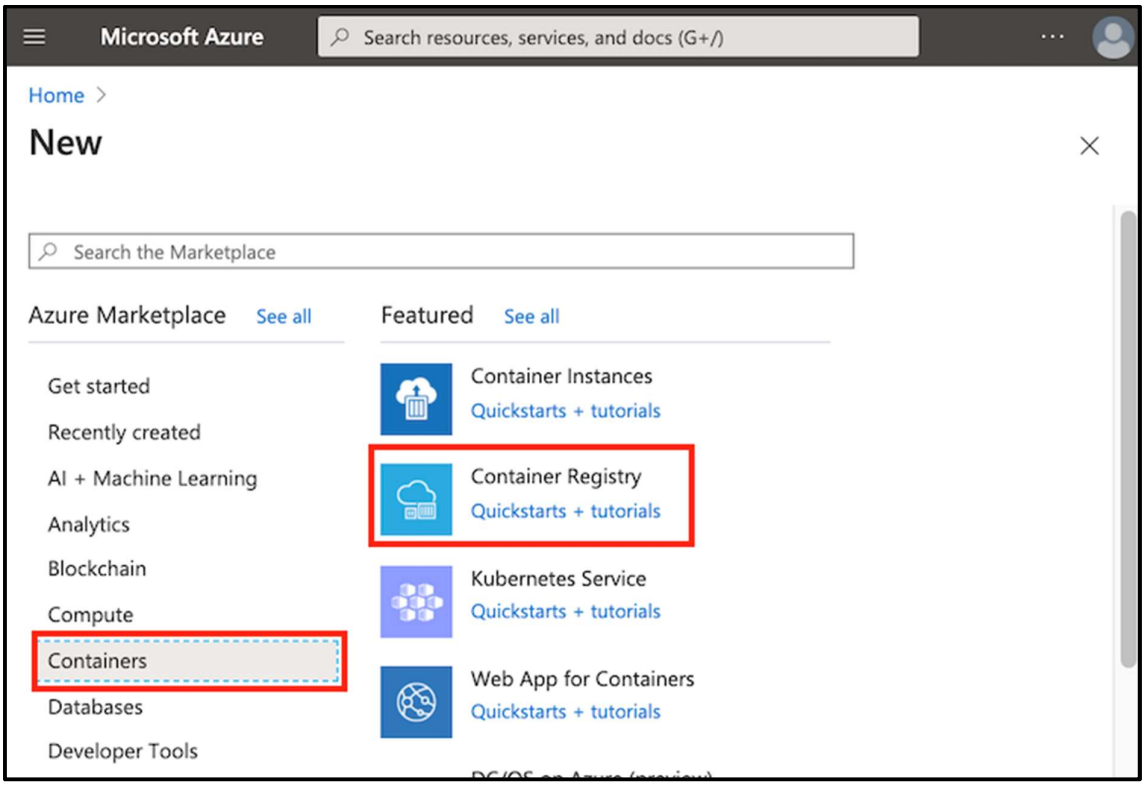
Azure Container Registry

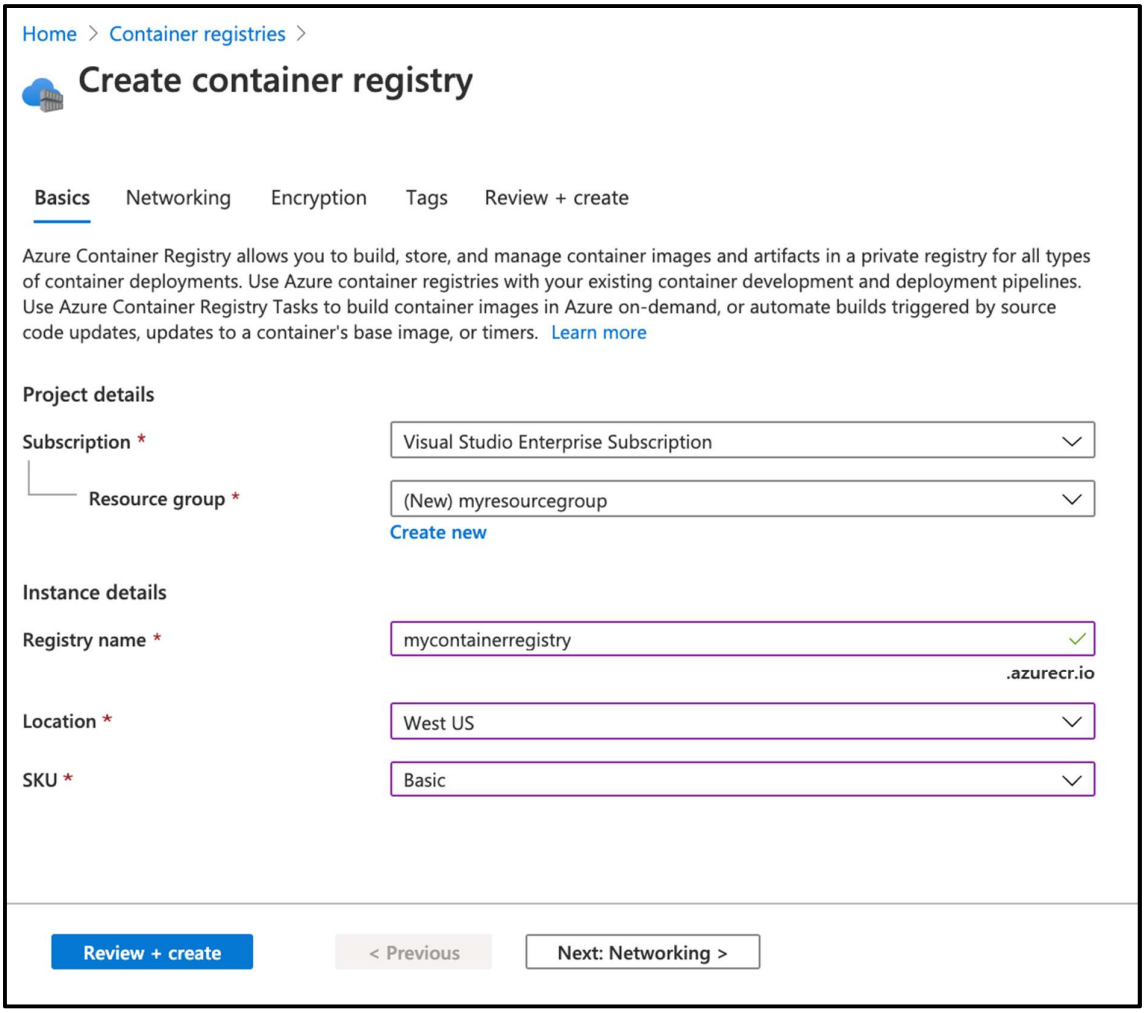
- Azure Container Registry is a managed registry.
- Azure Container Registry allows you to build, store, and manage container images and artifacts in a private registry for all types of container deployments.
- Azure Container Registry Tasks can be used to build container images in Azure and automate builds triggered by source code updates, updates to a container's base image, or timers.
- **Key features of Azure Container Registry:**
  - ✓ **Registry service tiers** - Create one or more container registries. Registries are available in three tiers: Basic, Standard, and Premium.
  - ✓ **Security and access** - You log in to a registry using the Azure CLI or the standard docker login command. Azure Container Registry transfers container images over HTTPS, and supports TLS to secure client connections.
  - ✓ **Supported images and artifacts** - Images in Azure container registries can be Windows and Linux images. Standard Docker commands are used to push images into a repository, or pull an image from a repository.
  - ✓ **Automated image builds** - Use Azure Container Registry Tasks (ACR Tasks) to automate building, testing, pushing, and deploying images in Azure.

Lab - Azure container registry

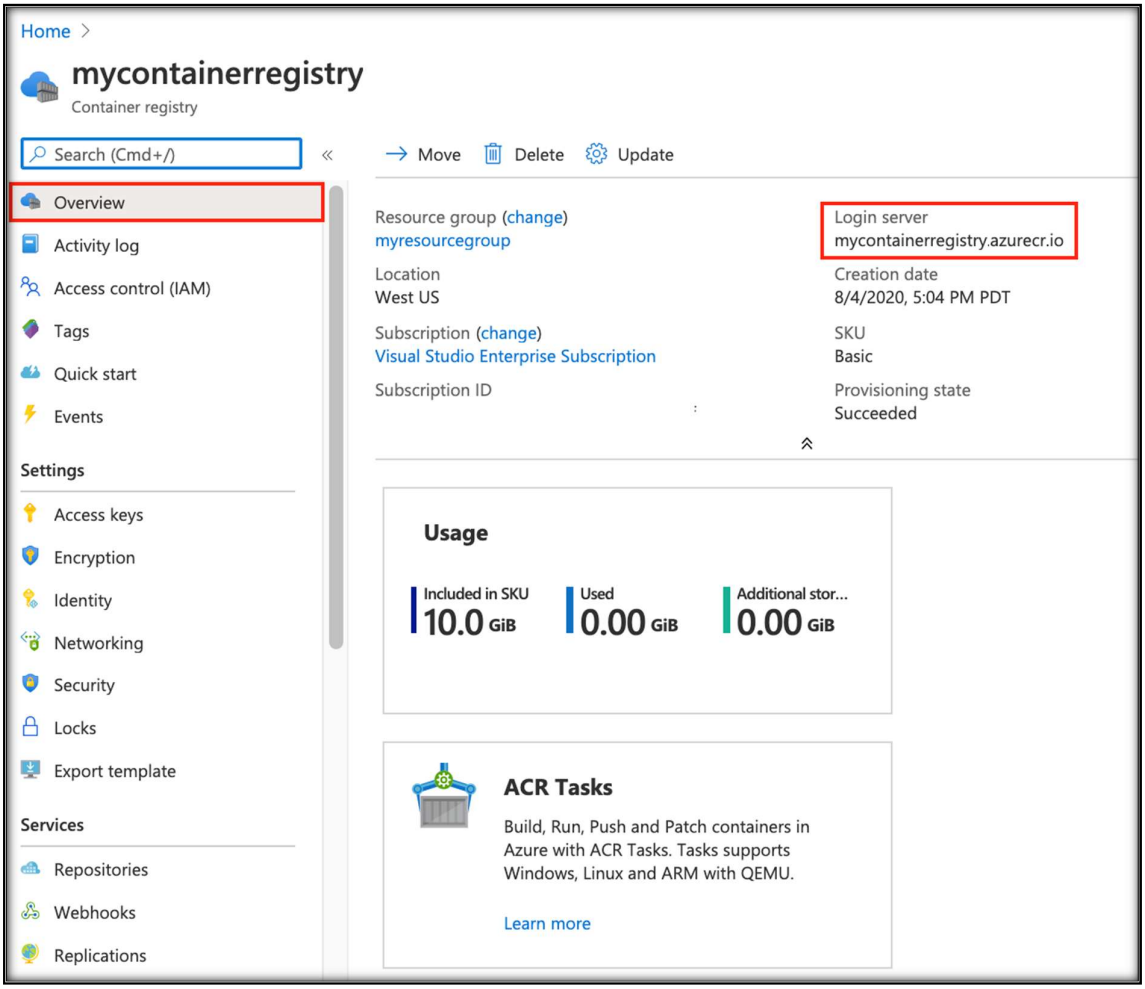
- Select **Create a resource** > **Containers** > **Container Registry**.



- In the **Basics** tab, enter values for **Resource group** and **Registry name**. The registry name must be unique within Azure, and contain 5-50 alphanumeric characters. For this quickstart create a new resource group in the West US location named myResourceGroup, and for **SKU**, select 'Basic'.



- Accept default values for the remaining settings. Then select **Review + create**. After reviewing the settings, select **Create**.
- When the **Deployment succeeded** message appears, select the container registry in the portal.



- Take note of the registry name and the value of the **Login server**, which is a fully qualified name ending with azurecr.io in the Azure cloud. You use these values when you push and pull images with Docker.

Publishing images to Azure container Registry

- To push an image to an Azure Container registry, you must first have an image.
- If you don't yet have any local container images, run the following docker pull command to pull an existing public image. For this example, pull the hello-world image from Microsoft Container Registry.

  docker pull mcr.microsoft.com/hello-world
- Before you can push an image to your registry, you must tag it with the fully qualified name of your registry login server.

  Syntax:

  docker tag mcr.microsoft.com/hello-world <login-server>/hello-world:v1

  Example:

  docker tag mcr.microsoft.com/hello-world mycontainerregistry.azurecr.io/hello-world:v1

- Finally, use docker push to push the image to the registry instance. Replace <login-server> with the login server name of your registry instance. This example creates the **hello-world** repository, containing the hello-world:v1 image.

  docker push <login-server>/hello-world:v1
- After pushing the image to your container registry, remove the hello-world:v1 image from your local Docker environment.

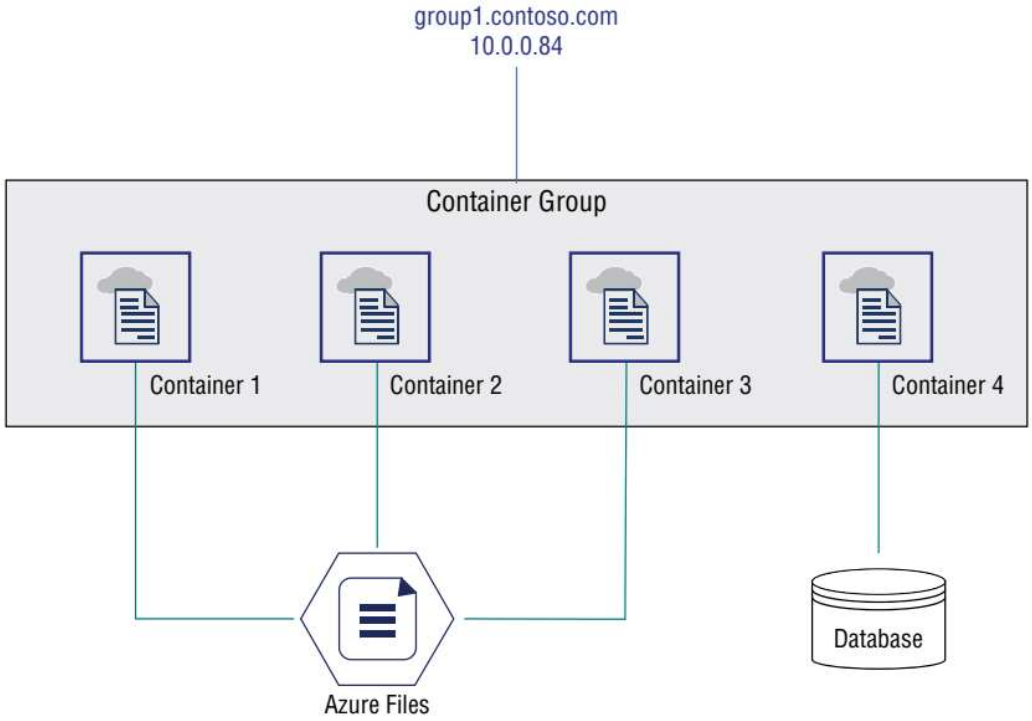
  docker rmi <login-server>/hello-world:v1

Azure Container Instances

- Azure Container Instances is a service that enables a developer to deploy containers on the Microsoft Azure public cloud without having to provision or manage any underlying infrastructure.
- Azure Container Instances (ACI) is the Azure service that gives you the ability to create and deploy containerized applications.
- ACI supports both Windows and Linux containers. Containers offer a potentially significant cost savings because you are only paying for consumption of CPU and memory resources used by the container, rather than paying for a VM instance.
- In addition, containers can easily scale out to accommodate demand changes for the containerized app.
- ACI supports both Linux and Windows containers -- eliminates the need for a developer to provision virtual machines, or implement a container orchestration platform, such as Kubernetes, to deploy and run containers.
- Instead, with Azure Container Instances (ACI), an organization can spin up a new container via the Azure portal or command-line interface (CLI), and Microsoft automatically provisions and scales the underlying compute resources. A
- ACI also supports standard Docker images a developer can pull from a container registry, such as Docker Hub or Azure Container Registry.
- According to Microsoft, ACI reduces management overhead, so a developer can deploy a container on Azure within seconds.
- Key features of the ACI service include:
  - ✓ Public IP connectivity: A developer can expose containers to the internet with a fully qualified domain name (FQDN) and an IP address.
  - ✓ Customization: A developer can specify the number of CPU cores and memory he or she wants for a container instance.
  - ✓ Persistent storage: Container instances are stateless by default, but an organization can choose to mount an Azure file share to a container to enable persistent storage.
  - ✓ Container groups: A developer can schedule multiple containers to deploy as a group that shares the same host machine, storage, network and other resources. This feature is beneficial when a developer wants to split one functional task among several container images.

Azure Container Group

- Although ACI supports only single container instances for Windows as of this writing, ACI supports container groups for Linux.
- A container group is a collection of containers that run on the same host machine and share the same operating system, lifecycle, local network, resources, and storage. The group shares a single IP address and DNS name.



- The following diagram shows an example of a container group that includes multiple containers:

- This example container group:
    - Is scheduled on a single host machine.
    - Is assigned a DNS name label.
    - Exposes a single public IP address, with one exposed port.
    - Consists of two containers. One container listens on port 80, while the other listens on port 5000.
    - Includes two Azure file shares as volume mounts, and each container mounts one of the shares locally.
- **Note:** Multi-container groups currently support only Linux containers. For Windows containers, Azure Container Instances only supports deployment of a single container instance.
- Deployment: Here are two common ways to deploy a multi-container group: use a Resource Manager template or a YAML file. A Resource Manager template is recommended when you need to deploy additional Azure service resources when you

deploy the container instances. Due to the YAML format's more concise nature, a YAML file is recommended when your deployment includes only container instances.

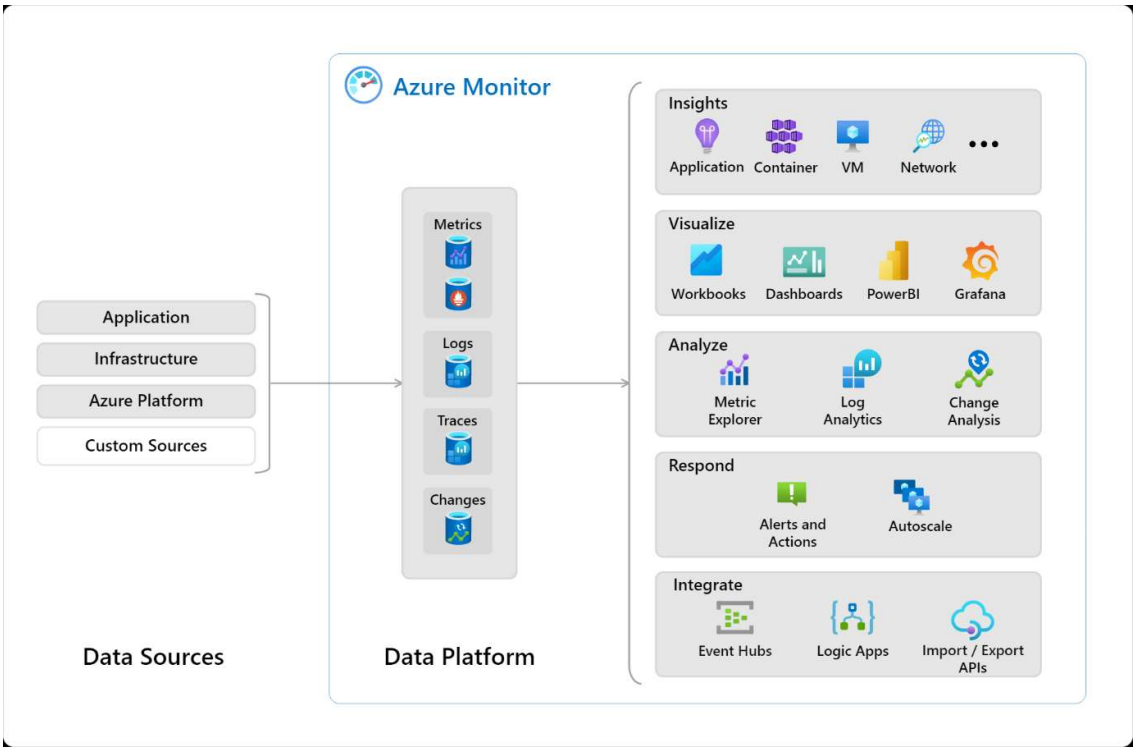
- Resource allocation: Azure Container Instances allocates resources such as CPUs, memory, and optionally GPUs (preview) to a multi-container group by adding the resource requests of the instances in the group.
- Resource usage by container instances: Each container instance in a group is allocated the resources specified in its resource request. However, the maximum resources used by a container instance in a group could be different if you configure its optional resource limit property. The resource limit of a container instance must be greater than or equal to the mandatory resource request property.

Azure Kubernates Service

- Azure Kubernetes Service Managing a few container instances is relatively easy, but as the number of containers increases, deployment and management become much more complex.
- That is where Azure Kubernetes Service (AKS) comes into play.
- AKS is a container orchestration service that monitors container health, provides container scalability, and enables resource sharing among containers in a Kubernetes cluster.
- Each of the containers in the Kubernetes cluster is called a node.
- AKS simplifies deployment because once you've defined a container image, you can use AKS to easily deploy as many instances of that image as needed within a cluster, as well as deploy multiple clusters.

Azure Monitor

- Azure Monitor Service Azure monitor service helps to monitor apps and infrastructure in cloud.
- It delivers a comprehensive solution for collecting, analyzing, and acting on telemetry from your cloud and on-premises environments.
- This information helps you understand how your applications are performing and proactively identify issues that affect them and the resources they depend on.
- Azure Monitor helps you maximize the availability and performance of your applications and services.
- A few examples of what you can do with Azure Monitor include:
    - Detect and diagnose issues across applications.
    - Correlate infrastructure issues with VM insights and Container insights.
    - Drill into your monitoring data with Log Analytics for troubleshooting and deep diagnostics.
    - Create visualizations with Azure dashboards and workbooks.
    - Collect data from monitored resources by using Azure Monitor Metrics.

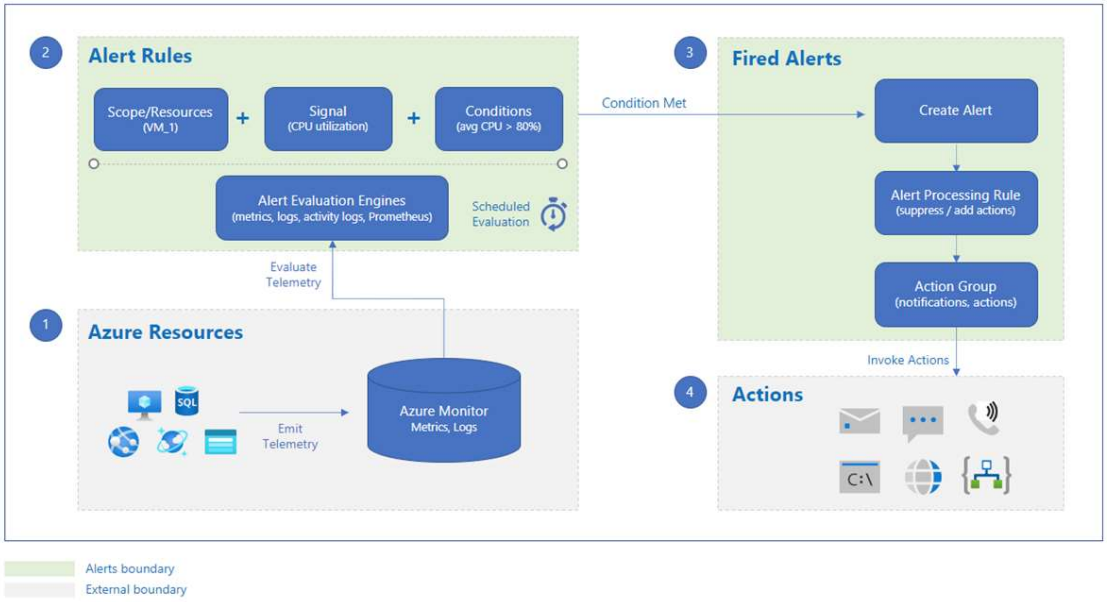


The above diagram gives a high-level view of Azure Monitor.

- The stores for the data platform are at the centre of the diagram. Azure Monitor stores these fundamental types of data: metrics, logs, traces, and changes.
- The sources of monitoring data that populate these data stores are on the left.
- The different functions that Azure Monitor performs with this collected data are on the right. This includes such actions as analysis, alerting.

Azure Monitor Alerts

- Alerts help you detect and address issues before users notice them by proactively notifying you when Azure Monitor data indicates that there may be a problem with your infrastructure or application.
- You can alert on any metric or log data source in the Azure Monitor data platform.
- This diagram shows you how alerts work:



- An **alert rule** monitors your telemetry and captures a signal that indicates that something is happening on the specified resource. The alert rule captures the signal and checks to see if the signal meets the criteria of the condition. If the conditions are met, an alert is triggered, which initiates the associated action group and updates the state of the alert. An alert rule combines:
    - The resource(s) to be monitored
    - The signal or telemetry from the resource
    - Conditions
- If you're monitoring more than one resource, the condition is evaluated separately for each of the resources and alerts are fired for each resource separately.
- Once an alert is triggered, the alert is made up of:
    - **Alert processing rules** allow you to apply processing on fired alerts. Alert processing rules modify the fired alerts as they're being fired. You can use alert processing rules to add or suppress action groups, apply filters or have the rule processed on a pre-defined schedule.
    - **Action groups** can trigger notifications or an automated workflow to let users know that an alert has been triggered. Action groups can include:
        - Notification methods such as email, SMS, and push notifications.
        - Automation Runbooks
        - Azure functions
        - ITSM incidents
        - Logic Apps
        - Secure webhooks
        - Webhooks
        - Event hubs
    - **Alert conditions** are set by the system. When an alert fires, the alert's monitor condition is set to 'fired', and when the underlying condition that caused the alert to fire clears, the monitor condition is set to 'resolved'.
    - The **user response** is set by the user and doesn't change until the user changes it.

Log Analytics Workspace

- A Log Analytics workspace is a unique environment for log data from Azure Monitor.
- Each workspace has its own data repository and configuration but might combine data from multiple services.
- When you collect logs and data, the information is stored in a workspace.
- A workspace has a unique workspace ID and resource ID.
- The workspace name must be unique for a given resource group. You can use a single workspace for all your data collection.
- You can also create multiple workspaces based on requirements such as:
    - The geographic location of the data.
    - Access rights that define which users can access data.
    - Configuration settings like pricing tiers and data retention.
- You need a Log Analytics workspace if you collect data from:
    - Azure resources in your subscription.
    - On-premises computers monitored by System Centre Operations Manager.
    - Device collections from Configuration Manager.
    - Diagnostics or log data from Azure Storage.

Azure Cloud Security

- Protect data, apps, and infrastructure quickly with built-in security services in Azure that include unparalleled security intelligence to help identify rapidly evolving threats early.

VM Security Best Practices

- Use Azure Secure Score in Azure Security Center as your guide.
- Isolate management ports on virtual machines from the Internet and open them only when required
- Use complexity for passwords and user account names
- Keep the operating system patched
- Keep third-party applications current and patched
- Actively monitor for threats
- Azure Backup Service.
- Protect VMs by using authentication and access control.
- Use multiple VMs for better availability.
- Protect against malware.
- Manage your VM updates.
- Manage your VM security posture.
- Monitor VM performance.
- Encrypt your virtual hard disk files.
- Restrict direct internet connectivity.

Networking Security Best Practices

- Use strong network controls
- Logically segment subnets
- Adopt a Zero Trust approach
- Control routing behavior
- Use virtual network appliances
- Deploy perimeter networks for security zones
- Avoid exposure to the internet with dedicated WAN links
- Optimize uptime and performance
- Disable RDP/SSH Access to virtual machines
- Secure your critical Azure service resources to only your virtual networks

Database Security Best Practices

Database security is the processes, tools, and controls that secure and protect databases against accidental and intentional threats. The objective of database security is to secure sensitive data and maintain the confidentiality, availability, and integrity of the database. In addition to protecting the data within the database, database security protects the database management system and associated applications, systems, physical and virtual servers, and network infrastructure.

- **Database hardening:** Securing or "hardening" a database server combines physical, network, and operating system security to address vulnerabilities and make it more difficult for hackers to access the system.
- **Comprehensive data encryption:** Always encrypted data (at retst and in transit)
- **Advanced threat protection analyses logs** to detect unusual behaviour and potentially harmful attempts to access or exploit databases. Alerts are created for suspicious activities
- **Separate authentication accounts:** As a best practise, users and applications should use separate accounts to authenticate. This limits the permissions granted to users and applications and reduces the risks of malicious activity
- **Principle of least privilege:** The information security principle of least privilege asserts that users and applications should be granted access only to the data and operations they require to perform their jobs.
- **A Zero Trust security model** validates identities and device compliance for every access request to protect people, devices, apps, and data wherever they're located.

Zero Trust Security

- Zero Trust is a new security model that assumes breach and verifies each request as though it originated from an uncontrolled network.
- Zero Trust security model, which is based on these guiding principles:

    - **Verify explicitly** - Always authenticate and authorize based on all available data points.
    - **Use least privilege access** - Limit user access with Just-In-Time and Just-Enough-Access (JIT/JEA), risk-based adaptive policies, and data protection.

- **Assume breach** - Minimize blast radius and segment access. Verify end-to-end encryption and use analytics to get visibility, drive threat detection, and improve defenses.

Azure Key Vault

- Azure Key Vault is a cloud service for securely storing and accessing secrets.
- A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, or cryptographic keys.
- Key Vault service supports two types of containers: vaults and managed hardware security module(HSM) pools. Vaults support storing software and HSM-backed keys, secrets, and certificates. Managed HSM pools only support HSM-backed keys.
- To do any operations with Key Vault, you first need to authenticate to it.
- Azure Key Vault enforces Transport Layer Security (TLS) protocol to protect data when it's traveling between Azure Key vault and clients.
- Azure Key Vault is one of several key management solutions in Azure, and helps solve the following problems:
    - **Secrets Management** - Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets
    - **Key Management** - Azure Key Vault can be used as a Key Management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data.
    - **Certificate Management** - Azure Key Vault lets you easily provision, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and your internal connected resources.
    - Why use Azure Key Vault?
        - Centralize application secrets: Centralizing storage of application secrets in Azure Key Vault allows you to control their distribution. Key Vault greatly reduces the chances that secrets may be accidentally leaked.
        - Securely store secrets and keys: Access to a key vault requires proper authentication and authorization before a caller (user or application) can get access.
        - Monitor access and use: Once you've created a couple of Key Vaults, you'll want to monitor how and when your keys and secrets are being accessed. You can monitor activity by enabling logging for your vaults.
        - Simplified administration of application secrets
        - Integrate with other Azure services