DAY 2 DOCUMENTATION

AZURE ARM (AZURE RESOURCE MANAGER):

WHY AZURE ARM?

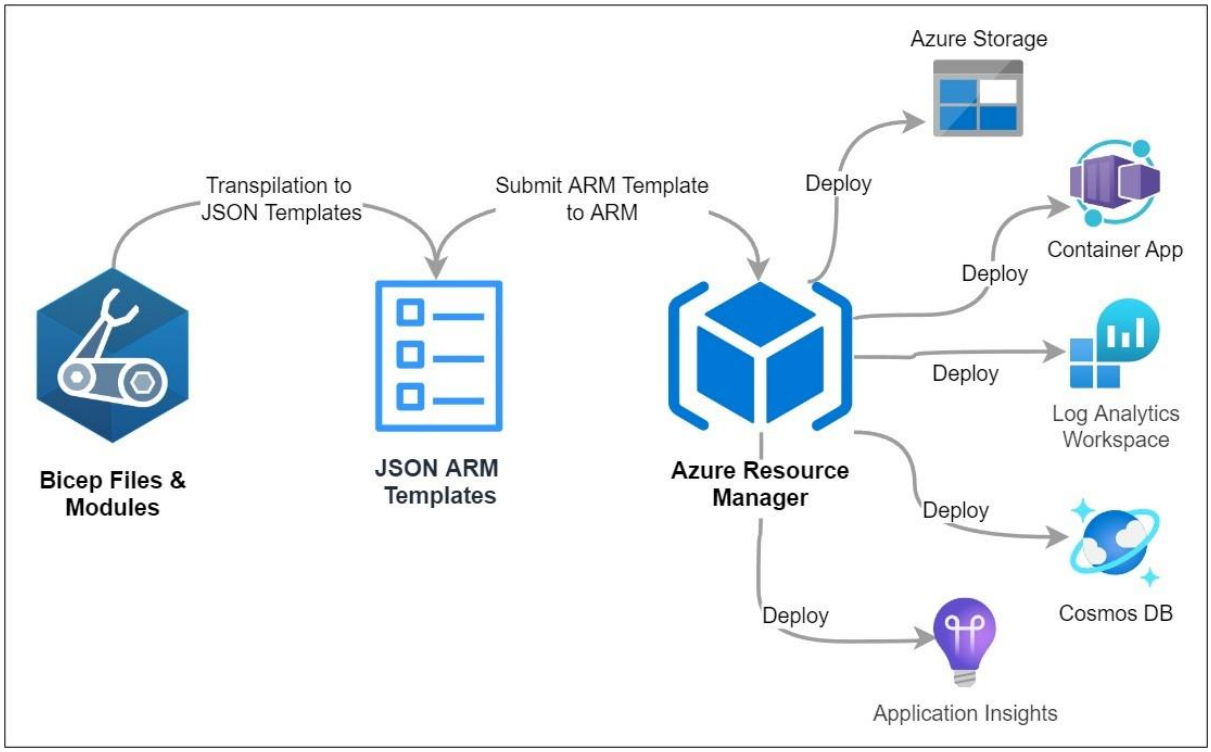
Azure Resource Manager (ARM) is an essential part of Microsoft Azure, offering a range of benefits for managing cloud resources efficiently. Here are some key reasons why Azure ARM is important:

1. Infrastructure as Code (IaC): ARM allows you to define your infrastructure using templates (JSON or YAML), enabling version control and automation in deployment processes.
2. Resource Grouping: You can organize related resources into resource groups, making it easier to manage, monitor, and apply access control.
3. Deployment Consistency: With ARM templates, you can ensure consistent deployment across different environments (development, testing, production) by reusing the same templates.
4. Access Control: ARM integrates with Azure's Role-Based Access Control (RBAC), allowing you to assign permissions at various scopes (subscription, resource group, or individual resource), enhancing security.
5. Tagging: Resources can be tagged for better organization and cost management, helping in tracking usage and expenses.
6. Tracking Changes: ARM provides a history of changes made to resources, which aids in auditing and troubleshooting.
7. Dependency Management: ARM handles dependencies between resources automatically during deployments, ensuring that resources are created or modified in the correct order.
8. Multi-Region Deployment: ARM enables the deployment of resources across multiple Azure regions with ease, promoting scalability and redundancy.

WHAT IS AZURE ARM?

Azure Resource Manager (ARM) is the deployment and management service for Microsoft Azure. It provides a comprehensive management layer that enables users to create, update, and delete resources in their Azure accounts.

Azure Resource Manager is a vital component of Azure's infrastructure, offering tools and frameworks for efficient resource management, deployment, security, and organization



What is difference between ARM and ARM templates?

The terms "ARM" (Azure Resource Manager) and "ARM templates" refer to related but distinct concepts in Azure. Here's a breakdown of their differences:

Azure Resource Manager (ARM)

- Definition: ARM is the service that allows for the deployment and management of Azure resources.
- Functionality: It provides a management layer for handling requests through APIs, managing resource groups, applying role-based access control (RBAC), and maintaining overall resource configuration.
- Role: ARM orchestrates the operations and ensures resources interact correctly according to defined dependencies.

ARM Templates

- Definition: ARM templates are JSON (JavaScript Object Notation) files or Bicep files that define the infrastructure and configuration of Azure resources that you want to deploy.
- Purpose: They specify what resources are needed (e.g., virtual machines, storage accounts) and their configurations in a declarative manner without describing how to create them.

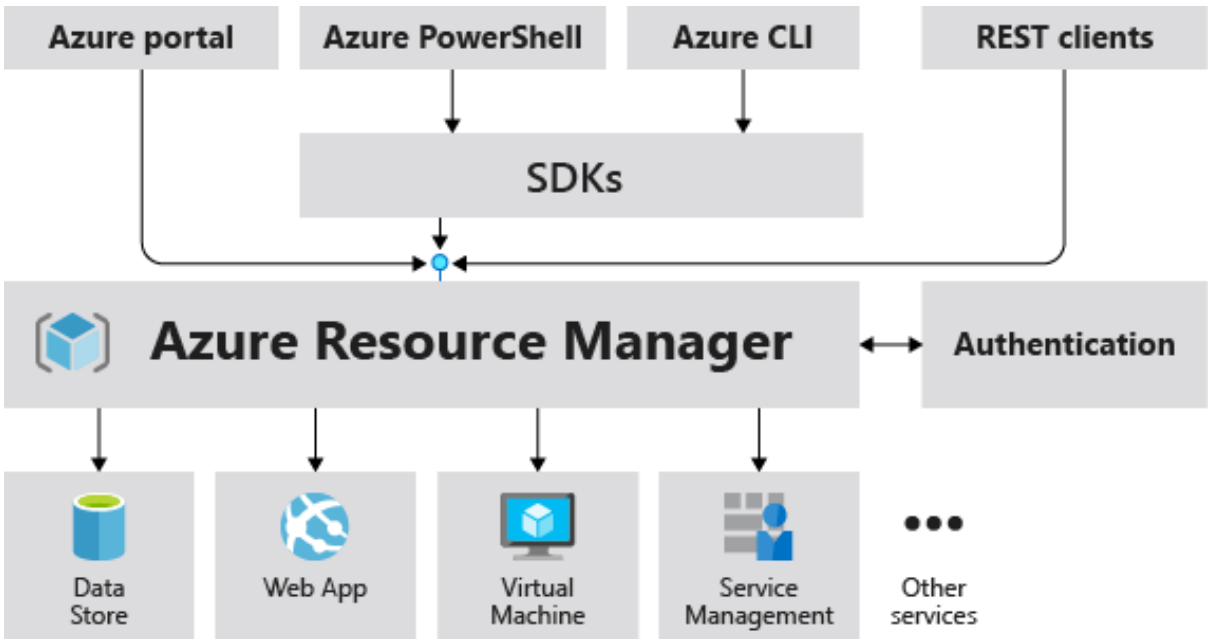
- Reusability: Templates can be reused across environments, enabling consistency and automating deployments.

Key Differences

1. Nature:
    - ARM is a service for managing Azure resources.
    - ARM templates are specific files that define what resources to create and their configurations.
2. Function:
    - ARM handles the lifecycle of Azure resources, including deployment, monitoring, and scaling.
    - ARM templates serve as blueprints for deployments, detailing the resources and their properties.

3. Syntax:
    - ARM operates behind the scenes, processing requests and managing resources based on the defined configurations.
    - ARM templates are written in JSON or Bicep syntax, outlining the desired state of the infrastructure.

In summary, Azure Resource Manager is the overarching service for resource management, while ARM templates are the specific definitions that describe the resources you wish to deploy and manage within that framework.



1. $schema
    - Purpose: Defines the location of the JSON schema file that will validate the structure of the template. It ensures the template conforms to a specific version of deployment schema in Azure.
    - Example:

https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#

2. contentVersion
    - Purpose: Indicates the version number of the template. This is useful for tracking changes and ensures compatibility with the Azure API.
    - Example: "1.0.0.0"
3. parameters
    - Purpose: A section for defining configurable parameters. This template has an empty parameters object, meaning it does not require any external inputs for deployment.
    - Example: {}
4. functions
    - Purpose: Allows you to define reusable functions within the template. This template does not use any custom functions, so the array is empty.
    - Example: []
5. variables
    - Purpose: This section is for defining variables that can be reused throughout the template. The template has no defined variables, indicated by an empty object.
    - Example: {}
6. resources
    - Purpose: Defines the resources that will be created or managed. This section is crucial as it specifies the actual storage account to be deployed.

    Example API Version:

    - 2023-01-01 is the version of the API that will be used for the storage account operations, ensuring access to the latest features and capabilities.
7. outputs

- Purpose: This section is for defining outputs returned after the deployment completes, such as resource IDs or other useful information. This template does not define any outputs, indicated by an empty object.
- Example: {}

Connection of ARM

The connection of this ARM template to Azure Resource Manager involves several key processes:

1. Template Deployment: When deployed, the ARM template is submitted to Azure Resource Manager, which parses and validates the JSON structure against the defined schema.
2. Resource Creation: ARM interprets the resources defined in the template and interacts with the Azure backend to create the specified storage account based on the configurations provided.
3. Dependency Handling: ARM manages any dependencies when multiple resources are defined in a template, ensuring they are created in the correct order. In this case, since only a single resource is defined, this is less applicable.
4. Resource Management: After creation, ARM continues to manage the lifecycle of the storage account, facilitating updates, scaling, or deletions as needed.
5. Monitoring and Logging: Once deployed, ARM logs the deployment operations, allowing for auditing, history tracking, and monitoring of the storage account's usage and health.

Summary

This ARM template serves as a declarative way to create a storage account in Azure. It provides a clear structure using key fields such as $schema, contentVersion, and the resources array, which specifies the properties and configuration of the storage account. The connection to Azure Resource Manager ensures that the template is processed, validated, and deployed correctly, allowing for effective management of the Azure resource throughout its lifecycle.

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {},
    "functions": [],
    "variables": {},
    "resources": [
        {
            "name": "rutwiksg",
            "type": "Microsoft.Storage/storageAccounts",
            "apiVersion": "2023-04-01",
            "tags": {
                "displayName": "rutwiksg"
            },
            "location": "[resourceGroup().location]",
            "kind": "StorageV2",
            "sku": {
                "name": "Standard_LRS",
                "tier": "standard"
            }
        }
    ],
    "outputs": {}
}
```