Azure Resource Manager (ARM) is a management framework used in Microsoft Azure to deploy, manage, and organize Azure resources. ARM provides a way to create and manage resources consistently and predictably.

deploymentTemplate.json and subscriptiondeploymentTemplate.json refer to Azure Resource Manager (ARM) templates, which are JSON files used to define and deploy Azure resources. The key difference between them lies in their scope of deployment:

* deploymentTemplate.json (Resource Group Scope):

This is the most common type of ARM template. It is designed to deploy resources within a specific resource group. When you deploy this template, you specify the resource group where the resources defined in the template will be created or updated. Examples include deploying virtual machines, storage accounts, or web apps into an existing resource group.

* subscriptiondeploymentTemplate.json (Subscription Scope):

This type of ARM template is used to deploy resources or configure settings at the subscription level, rather than within a specific resource group. This is useful for tasks that affect the entire subscription, such as:

* + Creating new resource groups.
  + Assigning Azure Policies at the subscription level.
  + Creating management groups.
  + Deploying resources that are inherently subscription-scoped (e.g., Azure Blueprints definitions).

In essence, deploymentTemplate.json targets resources within a resource group, while subscriptiondeploymentTemplate.json targets resources or configurations at the subscription level.

{

"$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",

"contentVersion": "1.0.0",

"parameters":{},

"variables":{},

"functions":[],

"resources": [],

"outputs":{}

}

<https://k21academy.com/microsoft-azure/solution-architect/arm-templates/#format>

<https://learn.microsoft.com/en-us/training/paths/bicep-azure-pipelines/>

<https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/file>

@<decorator>(<argument>)

metadata <metadata-name> = ANY

targetScope = '<scope>'

@<decorator>(<argument>)

type <user-defined-data-type-name> = <type-expression>

@<decorator>(<argument>)

func <user-defined-function-name> (<argument-name> <data-type>, <argument-name> <data-type>, ...) <function-data-type> => <expression>

@<decorator>(<argument>)

param <parameter-name> <parameter-data-type> = <default-value>

@<decorator>(<argument>)

var <variable-name> = <variable-value>

@<decorator>(<argument>)

resource <resource-symbolic-name> '<resource-type>@<api-version>' = {

<resource-properties>

}

@<decorator>(<argument>)

module <module-symbolic-name> '<path-to-file>' = {

name: '<linked-deployment-name>'

params: {

<parameter-names-and-values>

}

}

@<decorator>(<argument>)

output <output-name> <output-data-type> = <output-value>