ARM Templates - Ejercicio

para éste ejercicio se debe revisar la documentación oficial de Microsoft acerca de las ARM Templates:

https://docs.microsoft.com/en-us/azure/templates/

en éste repositorio de GitHub pueden encontrar todas las ARM templates disponibles para desplegar recursos en Azure, sin embargo a menudo son complejas de entender porque contienen dependencias que pueden resultar innecesarias para el proceso de entendimiento y aprendizaje, lo que recomiendo es que empiecen a construir sus templates paso a paso de acuerdo a lo que vayan a desplegar.

https://github.com/Azure/azure-quickstart-templates

A continuación encuentran un pequeño esquema que pretende mostrar la estructura básica de un ARM Template

Schema

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

Content Version

"contentVersion": "1.0.0.0",

Parameters

Object

    "parameters": {

        "Name": {

            "metadata": {

                "description": "some text"

            },

            "type": "string, Object, Array…, etc"

        },

.

.

.

}

Variables

Object

    "variables": {

        "Name": value, (some functions are applicable)

.

.

.

}

Resources

Array of Objects

    "resources": [

        {

            "res1":

                Properties depends on the template,

        },

.

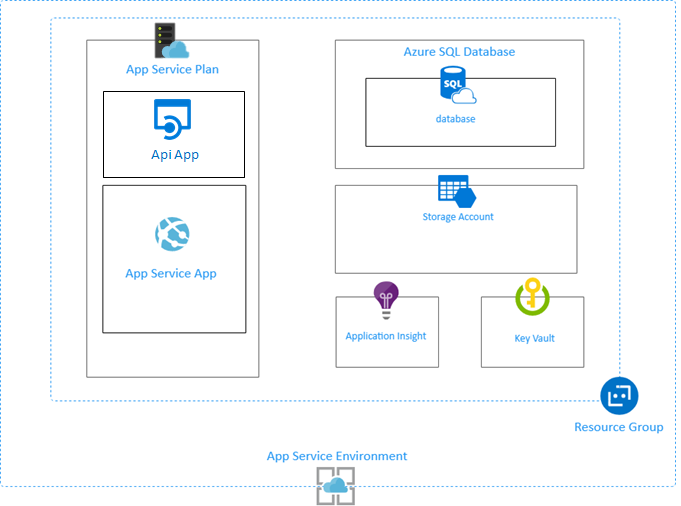
.

.

]

La primera parte de éste ejercicio está comprendida por la creación de una serie de templates para desplegar distintos tipos de recursos:

1. App Service Plan
2. API App: .Net Core 3.1
3. WEB App: con soporte para NodeJS
4. KeyVault
5. Storage Account
6. Sql Server con al menos una BD



**Condiciones adicionales**

* El App service plan debe ser usado por el API y el Web App.
* El API App debe tener configurado los siguientes connection strings:

BD\_ConnString

Storage\_Conn

Kv\_Conn

* El KeyVault debe tener habilitada la característica de Soft Deletion y debe tener implementado un lock que impida que el recurso sea eliminado.
* La contraseña del usuario admin de la BD debe estar almacenada como un secreto en el KeyVault

**Recomendaciones:**

* Para cada recurso debe existir un archivo json con la definición y un archivo json de parámetros.
* Para testear los templates es más rápido hacerlo haciendo uso de la función de powershell: **Test-AzResourceGroupDeployment**, (<https://docs.microsoft.com/en-us/powershell/module/az.resources/test-azresourcegroupdeployment?view=azps-5.5.0>), pueden agregar el switch -debug para ver un log completo de la validación.
* Crear un CI build con tareas de validación para sus templates, la tarea que se debe usar para esto es: Resource Group Deployment task en modo de validación

**Nota:** Para el desarrollo de éste ejercicio se debe crear un nuevo repositorio que se llame Infrastructure\_Repo y para la manipulación de los templates se recomienda el uso de Visual Studio Code, a continuación hay una pequeña guía de como puede ser configurado para que el ambiente les sea de ayuda: <https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/quickstart-create-templates-use-visual-studio-code?tabs=CLI>

A continuación podrán encontrar algunas funciones que pueden ser de ayuda:

| **Type** | **Usage** | **Sample Methods** | **Use When** |
| --- | --- | --- | --- |
| Resourcegroup | "[resourcegroup().xxxxxxx]" | * Location * ID * Name | Dynamically inserting information about the target resource group into the template |
| Subscription | "[subscription().xxxxxxxx]" | * Tenantid * subscriptionID * DisplayName | Insert information about the subscription |
| ResourceID | "[resourceId([subscriptionId], [resourceGroupName], resourceType, resourceName1, [resourceName2]...)]" | * None | Get the full resource ID of an Azure resource |
| String Functions | "[toLower(stringToChange)]”  "[toUpper(stringToChange)]“  "[first(string)]” | * None | Set the string to all lowercase letters |
| Reference | "[reference(resourceName or resourceIdentifier, [apiVersion], ['Full'])]" | * identity.tenantId * identity.principalId | Initiates an Azure REST API call that returns information about an Azure resource. Can be used to get values from keyvault |
| List | "[list{Value}(resourceName or resourceIdentifier, apiVersion, functionValues)]" | * None | Lists information values from a targeted resource. Use for dynamically getting keys to storage accounts and service bus |
| Concatenate | “[concat (arg1, arg2, arg3, ...)]” | * None | Combines parameters or variables into one string |