Imagen que contiene agua, hombre, carretera, sostener

Descripción generada automáticamente

**Lab Details**

1. In this lab, you will learn how to Publish Azure Static Web Apps using an ARM Template.
2. Duration: **60 minutes**

**Prerequisites**

* **GitHub Account**: If you already have a GitHub account then login into your account. Otherwise, create a new GitHub account.

**Introduction**

**What is an ARM template?**

* An ARM template is a JSON file that defines one or more resources to be deployed to Azure.
* ARM templates use a declarative syntax, meaning you define the desired end state of your resources rather than the specific steps required to create them.
* ARM templates can be used to deploy a wide variety of Azure resources, including virtual machines, storage accounts, virtual networks, and more.
* They can also be used to deploy resources from the Azure Marketplace, such as SaaS applications and IaaS offerings.
* ARM templates are an important part of the Azure DevOps process, as they can be used to automate the deployment of resources as part of a CI/CD pipeline.
* They can be used in conjunction with Azure Automation to automate the deployment and management of resources on a scheduled basis.

**What is an Azure Web App?**

* Azure App Service is a collection of four services, which are built to help you host and run web applications.
* The four services are:
  + Web Apps
  + Mobile Apps
  + API Apps
  + Logic Apps
* Although they look different, in the end, they all operate in very similar ways.
* Web Apps are the most commonly used.

**Architecture Diagram**

**Diagrama

Descripción generada automáticamente**

**Task Details**

1. Sign in to Azure Portal
2. Create a GitHub personal Access Token
3. Create a GitHub repository
4. Explore the ARM Template
5. Deploy the ARM template
6. Delete the Resources

# ****Lab Steps****

## ****Task 1: Sign in to Azure Portal****

1. Go to the Azure portal by using URL [https://portal.azure.com](https://portal.azure.com/).
2. Sign in with your given **username** and **password** on Azure portal.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

## ****Task 2: Create a GitHub Personal Access Token****

In this task, we will generate a GitHub Personal Access Token with the specified scopes and note for authentication and authorization purposes.

1. Log into your GitHub account. On the homepage of your GitHub account, in the upper right corner click on your profile and select settings option.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Scroll down to the bottom and from the left menu select **Developer Settings**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Now select **Personal Access Token**.
2. Then, from the drop down list select **Tokens (classic)**.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Click on **Generate New Token**.
2. Then, click on **Generate New Token (classic)**.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Provide a name to this token in the note field. Let's say myfirstwebapp.

Escala de tiempo

Descripción generada automáticamente con confianza media

1. In the select scopes section, mark repo, workflow and write:packages.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Scroll down to the bottom of the page and select the Generate token.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

1. Copy the provided token value and keep it handy for future reference.

Interfaz de usuario gráfica, Aplicación, Escala de tiempo

Descripción generada automáticamente

ghp\_6vjs43yUq6tKs7kqnUl0ihdn6Dpkuf1Wpaed

## ****Task 3: Create a GitHub Repository****

In this task, we will create a GitHub repository named **'myfirstwebapp'** from a template and save the repository URL for future reference.??

1. Click on the following link to create a new repository from template.

[**Template**](https://github.com/staticwebdev/vanilla-basic/generate)

1. Enter Repository Name as myfirstwebapp.
2. Now select Create Repository from template.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. Now, copy your Repository URL and paste it in your notepad for future reference.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

https://github.com/adriangtajamar/myfirstwebapp.git

## ****Task 4: Explore the ARM template****

In this task, we will download the ARM template files, review their content, and update the parameters in **azuredeploy.parameters.json** file with the repository URL and GitHub PAT token.

1. On the **Whizlabs LMS,** click on the file **azuredeploy.json** and **azuredeploy.parameters.json**present under **Lab Resources** to **download** the **ARM template**, which will further be used to **deploy** the **internal load balancer**.
2. It is advisable to go through the template once before starting with the next step.
   * **azuredeploy.json**

{

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

    "contentVersion": "1.0.0.0",

    "parameters": {

        "name": {

            "type": "string"

        },

        "location": {

            "type": "string"

        },

        "sku": {

            "type": "string"

        },

        "skucode": {

            "type": "string"

        },

        "repositoryUrl": {

            "type": "string"

        },

        "branch": {

            "type": "string"

        },

        "repositoryToken": {

            "type": "securestring"

        },

        "appLocation": {

            "type": "string"

        },

        "apiLocation": {

            "type": "string"

        },

        "appArtifactLocation": {

            "type": "string"

        },

        "resourceTags": {

            "type": "object"

        },

        "appSettings": {

            "type": "object"

        }

    },

    "resources": [

        {

            "apiVersion": "2021-01-15",

            "name": "[parameters('name')]",

            "type": "Microsoft.Web/staticSites",

            "location": "[parameters('location')]",

            "tags": "[parameters('resourceTags')]",

            "properties": {

                "repositoryUrl": "[parameters('repositoryUrl')]",

                "branch": "[parameters('branch')]",

                "repositoryToken": "[parameters('repositoryToken')]",

                "buildProperties": {

                    "appLocation": "[parameters('appLocation')]",

                    "apiLocation": "[parameters('apiLocation')]",

                    "appArtifactLocation": "[parameters('appArtifactLocation')]"

                }

            },

            "sku": {

                "Tier": "[parameters('sku')]",

                "Name": "[parameters('skuCode')]"

            },

            "resources": [

                {

                    "apiVersion": "2021-01-15",

                    "name": "appsettings",

                    "type": "config",

                    "location": "[parameters('location')]",

                    "properties": "[parameters('appSettings')]",

                    "dependsOn": [

                        "[resourceId('Microsoft.Web/staticSites', parameters('name'))]"

                    ]

                }

            ]

        }

    ]

}

### Understanding Azuredeploy.json script

1. Let’s divide the script into various groups for better understanding.
   * {"$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",...  "type": "object"}}- This part of the ARM script defines a template with a specified schema and version. It includes parameters such as name, location, SKU, repository details, and application settings, each with a specified data type. Users deploying the template must provide values for these parameters to customize the deployment in Azure.
   * "resources": [... "appArtifactLocation": "[parameters('appArtifactLocation')]"}},- This section of the ARM script defines a static site resource with specified properties for a web application deployment. It uses parameters for dynamic values such as name, location, and tags. The properties include details such as repository URL, branch, and build properties like app location, API location, and app artifact location, allowing customization during deployment in Microsoft Azure.
   * "sku": {

                "Tier": "[parameters('sku')]",

                "Name": "[parameters('skuCode')]"

            },- This script section configures the SKU (Stock Keeping Unit) for a resource, using parameters for the tier and name. The tier and name values are dynamic and provided during the deployment in Azure.

* + "resources": [{ "apiVersion": "2021-01-15",]- In this script segment, a configuration resource for app settings is defined, specifying its type, location, and properties. The resource has a dependency on a static site, ensuring proper sequencing in deployment, with the app settings provided through parameters during deployment in the specified Azure location.
  + **azuredeploy.parameters.json**

{

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

    "contentVersion": "1.0.0.0",

    "parameters": {

        "name": {

            "value": "myfirstswadeployment"

        },

        "location": {

            "value": "Central US"

        },

        "sku": {

            "value": "Free"

        },

        "skucode": {

            "value": "Free"

        },

        "repositoryUrl": {

            "value": "https://github.com/adriangtajamar/myfirstwebapp.git"

        },

        "branch": {

            "value": "main"

        },

        "repositoryToken": {

            "value": "ghp\_6vjs43yUq6tKs7kqnUl0ihdn6Dpkuf1Wpaed"

        },

        "appLocation": {

            "value": "/"

        },

        "apiLocation": {

            "value": ""

        },

        "appArtifactLocation": {

            "value": "src"

        },

        "resourceTags": {

            "value": {

                "Environment": "Development",

                "Project": "Testing SWA with ARM",

                "ApplicationName": "myfirstswadeployment"

            }

        },

        "appSettings": {

            "value": {

                "MY\_APP\_SETTING1": "value 1",

                "MY\_APP\_SETTING2": "value 2"

            }

        }

    }

}

### **Understanding Azuredeploy.Parameters.Json script**

* **"$schema":"https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#", "contentVersion": "1.0.0.0",-** This part of the script specifies the JSON schema for deployment parameters and defines the content version. It indicates the structure and version of the deployment parameters used in conjunction with the Azure Resource Manager template.
* **"parameters": {..."MY\_APP\_SETTING2": "value 2}-** These deployment parameters provide specific values for customization during deployment. They include details such as the resource name, location, GitHub repository information, app settings, and resource tags, enabling a tailored deployment of a Static Web App with Azure Resource Manager. The values specified will be used to replace corresponding parameters in the ARM template during deployment.

3. In the **azuredeploy.parameters.json** template update the following parameters :

* **repositoryUrl**: Provide the URL to your Static Web App GitHub repository which you copied earlier. Like the example shown in next image

  
Logotipo

Descripción generada automáticamente con confianza baja

* **repositoryToken**: provide the GitHub PAT Token which you have saved in the previous task. Like the example shown in next image

Texto

Descripción generada automáticamente  


**Note:**Make sure to **Save** the changes in the JSON file.

### **Do you Know?**

Publishing Azure Static Web Apps using an ARM template empowers developers to automate and streamline the deployment process, leveraging infrastructure as code for effortless management of static web applications.

## ****Task 5: Deploy the ARM template****

In this task, we will deploy the ARM template using Azure Cloud Shell by uploading the **azuredeploy.json** and **azuredeploy.parameters.json** files, and executing the deployment command with the specified resource group and files.

* + 1. On the Azure Portal, Click on the Azure Cloud Shell icon to start the Azure CLI.
    2. Now on the cloud Shell tab click on Bash.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + 1. Select show advanced settings.
    2. Now, leave all the settings to **default** and Enter **globally unique Storage account** name and **File Share** name. Then click on **Create Storage**.
    3. In the toolbar of Cloud Shell pane, select the **Upload/Download files** icon, select **Upload** From the dropdown and upload **azuredeploy.json** file into the Cloud Shell home directory.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

* + 1. When the template is uploaded successfully, you will see the **complete** message popup below at the bottom right of CLI.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

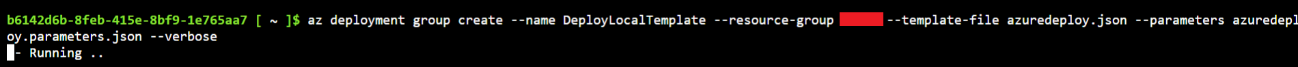
* + 1. **Repeat the step 5** to once again to upload the **azuredeploy.parameters.json** file which you have already downloaded from the **Lab Resources**.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* + 1. Now **deploy** the resources using the **given command**. This step may take a few minutes in order for the resources to get deployed successfully.

az deployment group create --name DeployLocalTemplate --resource-group rggithub --template-file azuredeploy.json --parameters azuredeploy.parameters.json --verbose

**NOTE**: Replace<resourcegroupname> with the resource group provided to you on Whizlabs LMS.

Texto

Descripción generada automáticamente con confianza media

1. Post deployment you will see the output as shown below and at the bottom end of the whole output you can see the **“provisioningState” : “Succeeded”**. It means that your ARM template is **deployed successfully.**

Texto

Descripción generada automáticamente

## ****Task 6: Delete the Resources****

In this task, we will delete all the resources.

1. In the search box at the top of the Azure portal, enter **Resource groups**. Select **Resource groups** from the search results.
2. Click on the name of the **Resource groups**.
3. Select all the Resources in that **Resource groups.**
4. Go to three dots to the right and then click **Delete** button.
5. Now type **delete** in the box present at the bottom.
6. Click on **Delete** to confirm deletion of resources.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

# ****Completion and Conclusions****

1. You have successfully logged into Azure portal.
2. You have successfully created a GitHub personal access token
3. You have successfully created a GitHub repository
4. You have successfully explored and understood the ARM template
5. You have successfully deployed the ARM template
6. You have successfully deleted the resources.