Deploy the DocFx Documentation Website to an Azure Website Automatically

In the article Using DocFx and Companion Tools to generate a Documentation website the process is described to generate content of a documentation website using DocFx. This document describes how to setup an Azure Website to host the content and automate the deployment to it using a pipeline in Azure DevOps.

The QuickStart sample that is provided for a quick setup of DocFx generation also contains the files explained in this document. Especially the **.pipelines** and **infrastructure** folders.

The following steps can be followed when using the Quick Start folder. In the **infrastructure** folder you can find the Terraform files to create the website in an Azure environment. Out of the box, the script will create a website where the documentation content can be deployed to.

1. Install Terraform

You can use tools like Chocolatey to install Terraform:

```
choco install terraform
```

2. Set the Proper Variables

> **Note:** Make sure you modify the value of the **app_name**, **rg_name** and **rg_location** variables. The *app_name* value is appended by **azurewebsites.net** and must be unique. Otherwise the script will fail that it cannot create the website.

In the Quick Start, authentication is disabled. If you want that enabled, make sure you have create an *Application* in the Azure AD and have the *client ID*. This client id must be set as the value of the **client_id** variable in *variables.tf*. In the *main.tf* make sure you uncomment the authentication settings in the *app-service*. For more information see Configure Azure AD authentication - Azure App Service.

If you want to set a custom domain for your documentation website with an SSL certificate you have to do some extra steps. You have to create a *Key Vault* and store the certificate there. Next step is to uncomment and set the values in *variables.tf*. You also have to uncomment the necessary steps in *main.tf*. All is indicated by comment-boxes. For more information see Add a TLS/SSL certificate in Azure App Service.

Some extra information on SSL certificate, custom domain and Azure App Service can be found in the following paragraphs. If you are familiar with that or don't need it, go ahead and continue with Step 3.

SSL Certificate

To secure a website with a custom domain name and a certificate, you can find the steps to take in the article Add a TLS/SSL certificate in Azure App Service. That article also contains a description of ways to obtain a certificate and the requirements for a certificate. Usually you'll get a certificate from the customers IT department. If you want to start with a development certificate to test the process, you can create one yourself. You can do that in PowerShell with the script below. Replace:

- **[YOUR DOMAIN]** with the domain you would like to register, e.g. `docs.somewhere.com`
- **[PASSWORD]** with a password of the certificate. It's required for uploading a certificate in the Key Vault to have a password. You'll need this password in that step.
- **[FILENAME]** for the output file name of the certificate. You can even insert the path here where it should be store on your machine.

You can store this script in a PowerShell script file (ps1 extension).

```
$cert = New-SelfSignedCertificate -CertStoreLocation
cert:\currentuser\my -Subject "cn=[YOUR DOMAIN]" -DnsName "
[YOUR DOMAIN]"
$pwd = ConvertTo-SecureString -String '[PASSWORD]' -Force -
AsPlainText
$path = 'cert:\currentuser\my\' + $cert.thumbprint
Export-PfxCertificate -cert $path -FilePath [FILENAME].pfx -
Password $pwd
```

The certificate needs to be stored in the common Key Vault. Go to `Settings > Certificates` in the left menu of the Key Vault and click `Generate/Import`. Provide these details:

- Method of Certificate Creation: `Import`
- Certificate name: e.g. `ssl-certificate`
- Upload Certificate File: select the file on disc for this.
- Password: this is the [PASSWORD] we reference earlier.

Custom Domain Registration

To use a custom domain a few things need to be done. The process in the Azure portal is described in the article Tutorial: Map an existing custom DNS name to Azure App Service. An important part is described under the header Get a domain verification ID. This ID needs to be registered with the DNS description as a TXT record.

Important to know is that this `Custom Domain Verification ID` is the same for all web resources in the same Azure subscription. See this StackOverflow issue. This means that this ID needs to be registered only once for one Azure Subscription. And this enables (re)creation of an App Service with the custom domain though script.

Add Get-permissions for Microsoft Azure App Service

The Azure App Service needs to access the Key Vault to get the certificate. This is needed for the first run, but also when the certificate is renewed in the Key Vault. For this purpose the Azure App Service accesses the Key Vault with the App Service resource provided identity. This identity can be found with the service principal name **abfa0a7c-a6b6-4736-8310-5855508787cd** or **Microsoft Azure App Service** and is of type **Application**. This ID is the same for all Azure subscriptions. It needs to have Get-permissions on secrets and certificates. For more information see this article Import a certificate from Key Vault.

Add the Custom Domain and SSL Certificate to the App Service

Once we have the SSL certificate and there is a complete DNS registration as described, we can uncomment the code in the Terraform script from the Quick Start folder to attach this to the App Service. In this script you need to reference the certificate in the common Key Vault and use it in the custom hostname binding. The custom hostname is assigned in the script as well.

The settings `ssl_state` needs to be `SniEnabled` if you're using an SSL certificate. Now the creation of the authenticated website with a custom domain is automated.

3. Deploy Azure Resources from Your Local Machine

Open up a command prompt. For the commands to be executed, you need to have a connection to your Azure subscription. This can be done using Azure Cli. Type this command:

```
az login
```

This will use the web browser to login to your account. You can check the connected subscription with this command:

```
az account show
```

If you have to change to another subscription, use this command where you replace *[id]* with the id of the subscription to select:

```
az account set --subscription [id]
```

Once this is done run this command to initialize:

```
terraform init
```

Now you can run the command to plan what the script will do. You run this command every time changes are made to the terraform scripts:

```
terraform plan
```

Inspect the result shown. If that is what you expect, apply these changes with this command:

```
terraform apply
```

When asked for approval, type "yes" and ENTER. You can also add the *-auto-approve* flag to the apply command.

The deployment using Terraform is not included in the pipeline from the Quick Start folder as described in the next step, as that asks for more configuration. But of course that can always be added.

4. Deploy the Website from a Pipeline

The best way to create the resources and deploy to it, is to do this automatically in a pipeline. For this purpose the **.pipelines/documentation.yml** pipeline is provided. This pipeline is built for an Azure DevOps environment. Create a pipeline and reference this YAML file.

> **Note:** the Quick Start folder contains a web.config that is needed for deployment to IIS or Azure App Service. This enables the use of the json file for search requests. If you don't have this in place, the search of text will never return anything and result in 404's under the hood.

You have to create a Service Connection in your DevOps environment to connect to the Azure Subscription you want to deploy to.

> **Note:** set the variables **AzureConnectionName** to the name of the Service Connection and the **AzureAppServiceName** to the name you determined in the *infrastructure/variables.tf*.

In the Quick Start folder the pipeline uses `master` as trigger, which means that any push being done to master triggers the pipeline. You will probably change this to another branch.

Last update: August 22, 2024