Azure scope structure

10 September 2023 20:59

To ease our compliance and access management let's start off by creating two different environment:

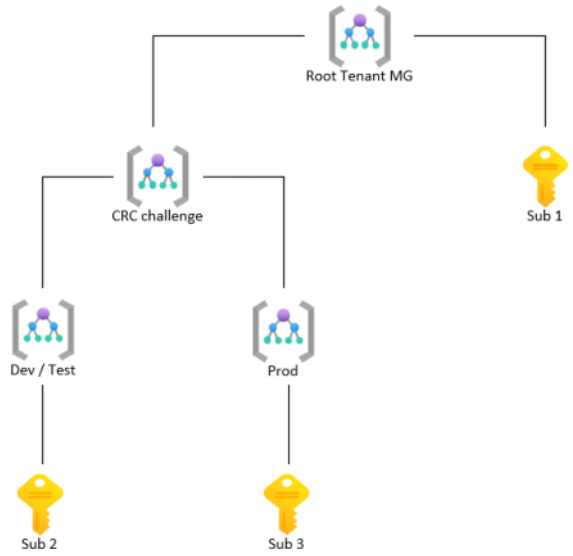
- Development / Testing (QA)
- Production (live, user-facing)

We will be organizing this structure using Management Groups.

In real circumstances, Management Groups are used as a way to compartmentalize subscriptions, we end up with a separate billi ng for each environment.

Governance can be applied at the MG level and inherited to the child resources.

Structure:



We create the Pay-As-You Go subscriptions through the azure portal.

In VS code let's use Azure CLI to push the changes:

```
# global admin azure login
az login

# create parent management group
az account management-group create --name CRC_challenge

# create child management groups
az account management-group create --name Dev_test --parent CRC_challenge
az account management-group create --name Prod --parent CRC_challenge

# list management groups
az account management-group list

# list all subscriptions (first re-auth the CLI with az login!)
az account list

# associate subscriptions to their parent management groups
az account management-group subscription add --name Dev_test --subscription "Subscription 2 - Dev_test"
az account management-group subscription add --name Prod --subscription "Subscription 3 - Prod"
```

Reflected changes:

Name ↑	Type
Tenant Root Group	Management group
Subscription 1 - Root	Subscription
CRC_challenge	Management group
Dev_test	Management group
Subscription 2 - Dev_test	Subscription
Prod	Management group
Subscription 3 - Prod	Subscription

Git: Setting up Version Control

10 September 2023 23:54

We download an html5/css3 template for the website

We first set up source control on Git:

1. Download and install Git Bash, terminal can then be launched within VS code like so

2. Manually create the resume repo and its readme.md file on Git

3. https://docs.github.com/en/authentication/connecting-to-github-with-ssh/generating-a-new-ssh-key-and-adding-it-to-the-ssh-agent
   Generate the ssh key for the repository

```
yoan.famel@pc4578945 MINGW64 ~
$ ssh-keygen -t ed8641686 -C "famel.yoan@protonmail.com"
Generating public/private ed25519 key pair.

Enter file in which to save the key (/c/Users/yoan.famel/.ssh/id_ed8641686):
Created directory '/c/Users/yoan.famel/.ssh'.

Enter passphrase (empty for no passphrase):
Enter same passphrase again:

Your identification has been saved in /c/Users/yoan.famel/.ssh/id_ed8641686
Your public key has been saved in /c/Users/yoan.famel/.ssh/id_ed8641686.pub

The key fingerprint is:
SHA256:464UiO0P8461HfstF68528ubdcndkcivbb948684 famel.yoan@protonmail.com
The key's randomart image is:
+--[ED25519 256]--+
|      ..***++. ..|
|       =.o+   ...|
|      5548 . ..oo|
|       *.   ..+.+|
|       .S E .o * |
|     ..78. o  * .|
|      +.. .  . + |
|     **=.     .  |
|    oo===.       |
+----[SHA256]-----+
```

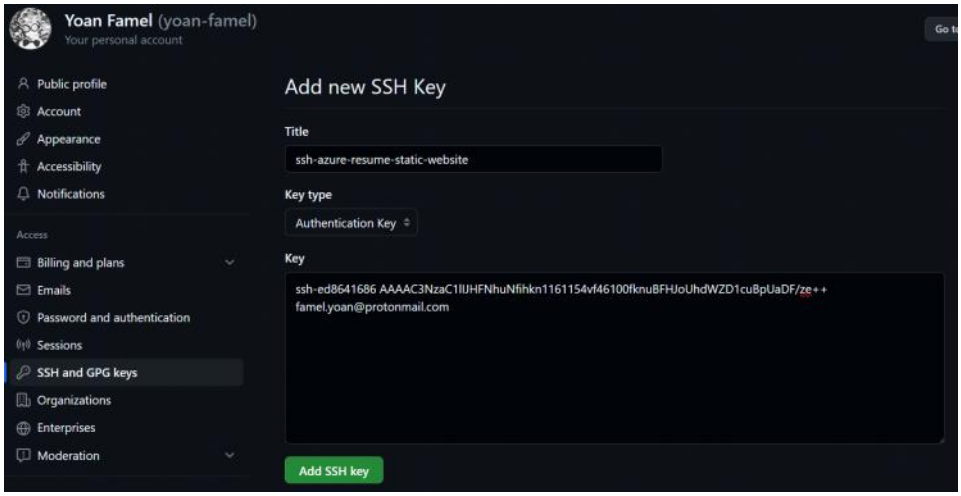
4. We start the Git ssh-agent and add our private key

```
yoan.famel@pc4578945 MINGW64 ~
$ eval "$(ssh-agent -s)"
Agent pid 678

yoan.famel@pc4578945 MINGW64 ~
$ ssh-add ~/.ssh/id_ed8641686

Enter passphrase for /c/Users/yoan.famel/.ssh/id_ed8641686:
Identity added: /c/Users/yoan.famel/.ssh/id_ed8641686 (famel.yoan@protonmail.com)
```

5. Let's now add the public key to Git

```
# copies the public key to clipboard
clip < ~/.ssh/id_ed8641686.pub
```

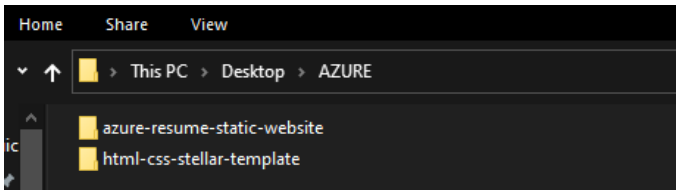
   On the web interface we manually add the previously copied public key;



6. Now we clone the remote repo to our local folder

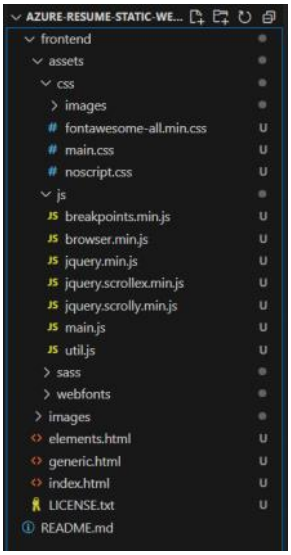
```
yoan.famel@pc4578945 MINGW64 ~
$ cd Desktop/AZURE/

yoan.famel@pc4578945 MINGW64 ~/Desktop/AZURE
$ git clone git@github.com:yoan-famel/azure-resume-static-website.git
Cloning into 'azure-resume-static-website'…

The authenticity of host 'github.com (180.82.181.3)' can't be established.
ED25519 key fingerprint is SHA256:+643513435dmjkmidHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes

Warning: Permanently added 'github.com' (ED8641686) to the list of known hosts.
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Notice we placed our html5 template in the same root folder.
All its content is then copied to a frontend root folder and moved to the local repository.



7. Now we can start reviewing the code structure in VS code, though we will leave the template as is for now and focus on uploading the static website to Azure storage, we will later come back to it and implement the JavaScript counter

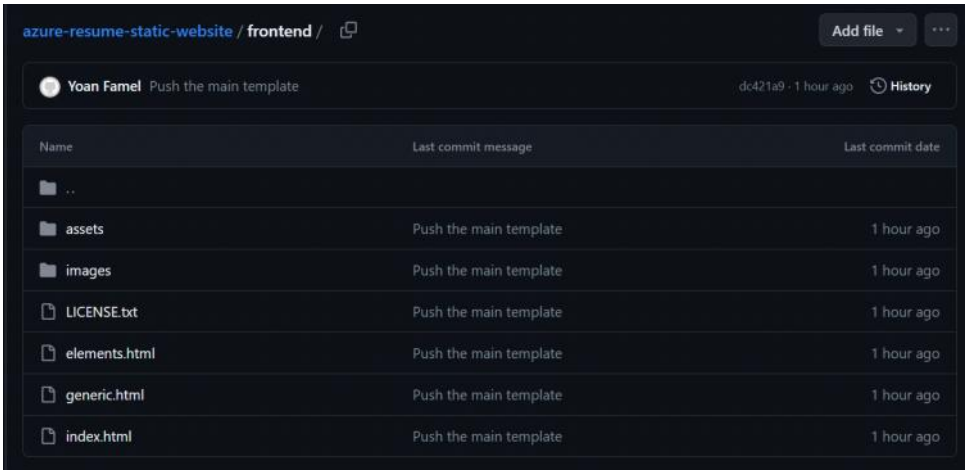
```
yoan.famel@pc4578945 MINGW64 ~/Desktop/AZURE
$ cd azure-resume-static-website/

yoan.famel@pc4578945 MINGW64 ~/Desktop/AZURE/azure-resume-static-website (main)
$ code .
```



Let's push the changes to Git before moving on:

```
# stage all the latest changes for our next commit
git add -A

# creates a snapshot of our current repo state
git commit -m "push front end template"

# push changes to the remote repo on Git
git push
```

Reflected changes on Git:

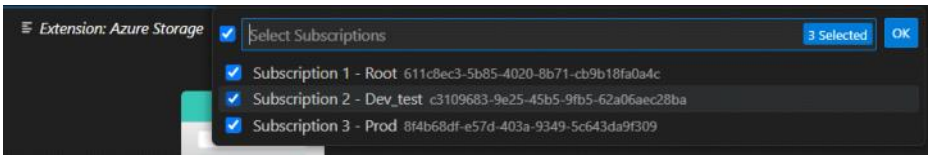


Azure storage: Enabling Static Web Hosting

11 September 2023 13:02

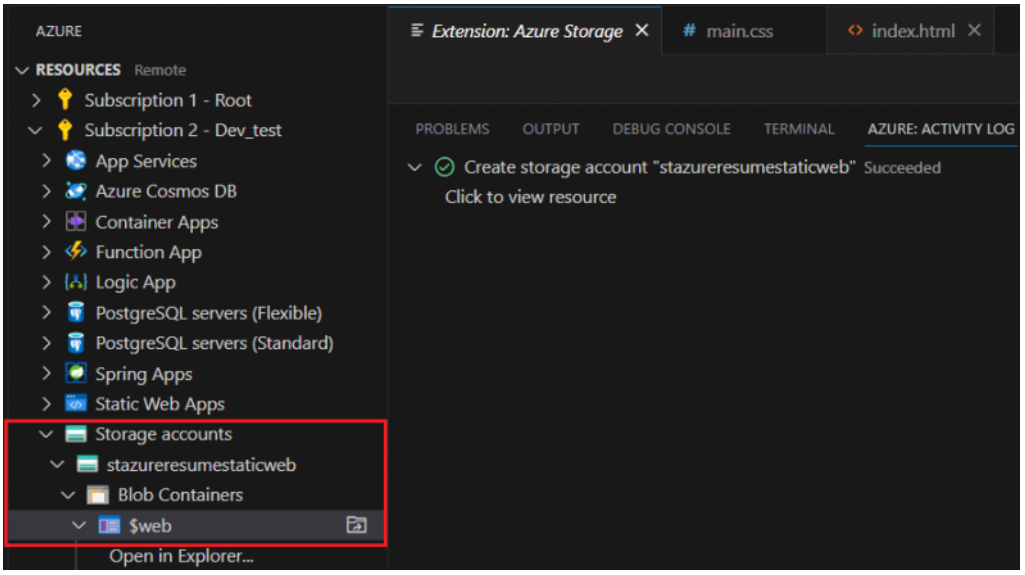
1. Let's add the module for Azure Storage to VS code first, that way we can create our storage account without going to the portal

2. In this case we select the Dev_test subscription we previously created since we are still in the development phase. We will leverage the prod subscription later once our CI/CD pipeline is up and running

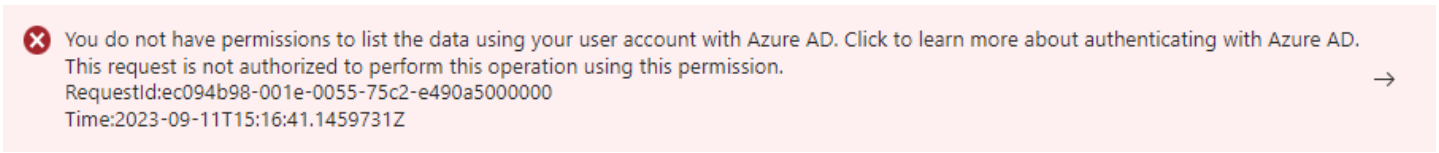


3. The Standard_LRS (HDD, locally redundant) storage is created and associated to the dev subscription.

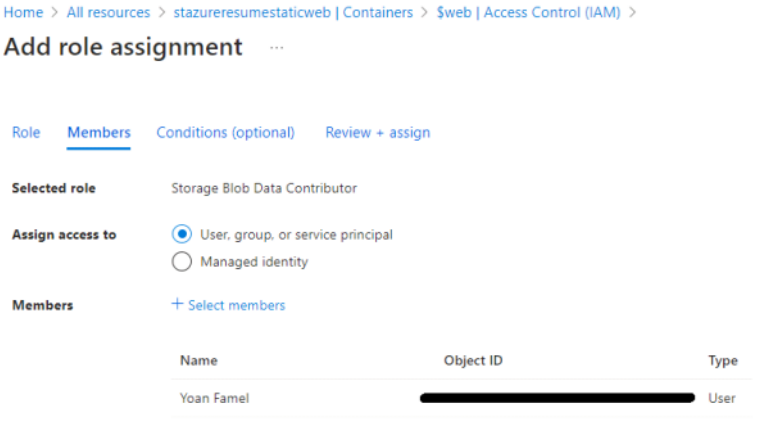
   A few basic naming conventions to respect while creating our resources in Azure:
   https://learn.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming



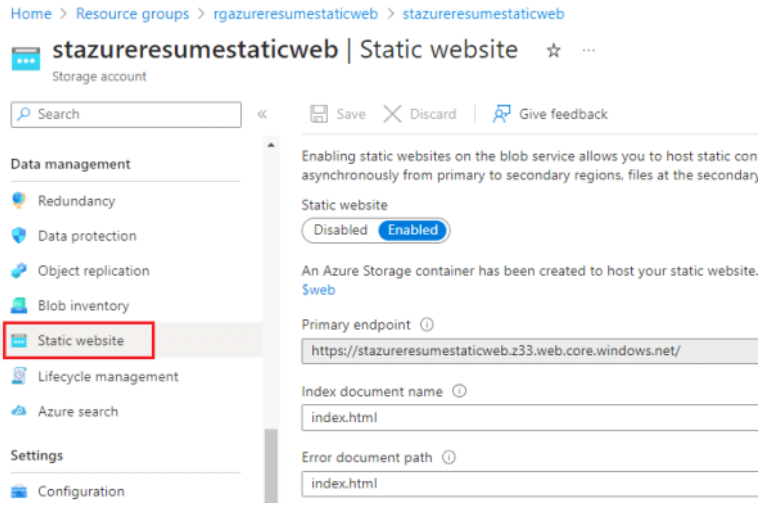
4. By navigating to the blob container we notice that we should first grant RBAC to manage and list its data



   In this case as a global admin I am able to change my permissions. Let's move to the IAM blade of the storage account, this way all child resources inherit the same permissions in case of future blob containers being added to the structure:



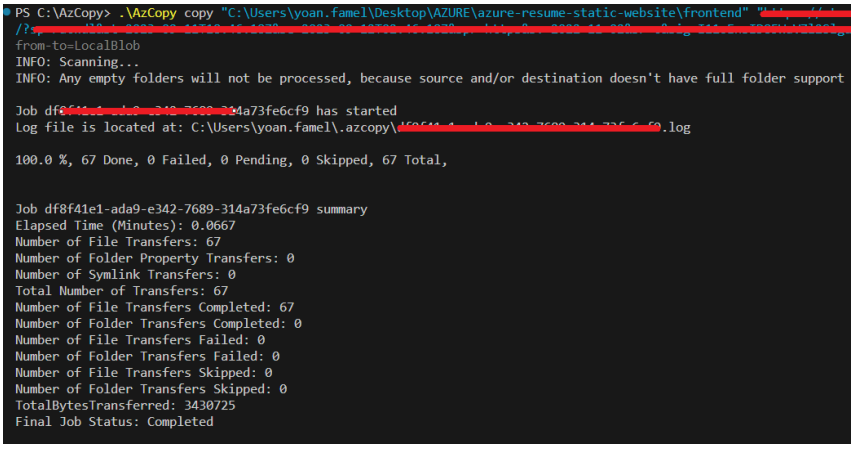
5. Now, after I made sure no other changes needed to be pushed to Git I moved on to importing the files to Azure storage, but first let's double check it is indeed set as static website:



There are several ways of uploading data to azure storage and I decided to practice the Az-Copy utility with a Shared Access Signature to then realize I had too many problems with this method.

I figured the $web blob container was the issue and created a test container with a newly generated SAS.

Which worked:



Issues found along the way:

- the SAS token should be placed after the container name and a forward slash
- The $web container should be escaped with \ since the $ sign conflicts the command

After trying Az-Copy countless times to no avail I used VS code and imported the frontend folder to the $web Container that way.


Deployment complete. The primary web endpoint is
https://
.web.core.windows.net/
Source: Azure Storage (Extension)
Browse to website
Ln 9, Col 49
Spaces: 3
UTF-8
CRLF
Log

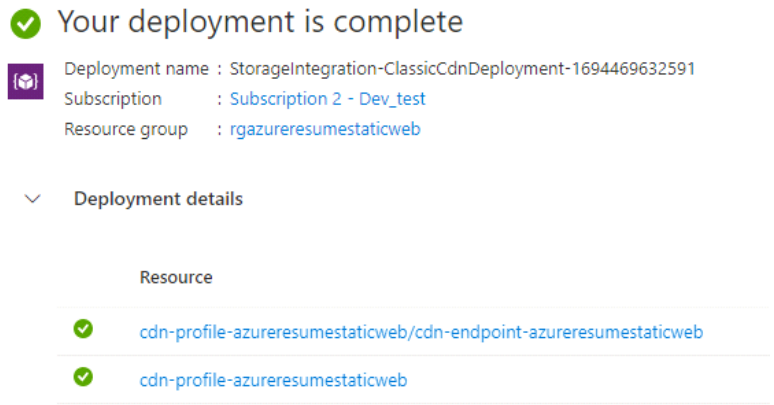
Mapping my custom domain to CDN and enabling TLS

11 September 2023 22:27

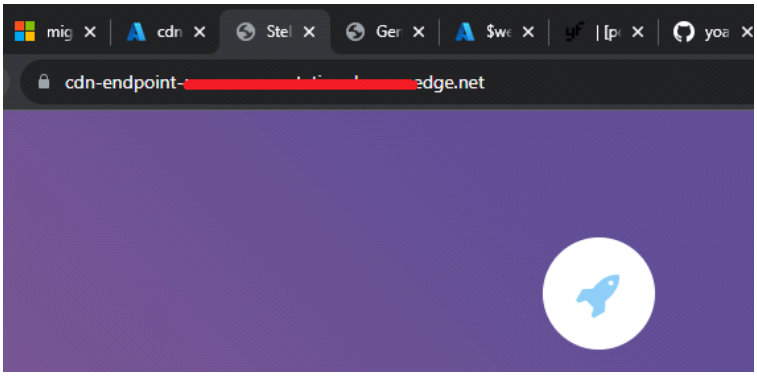
1. We can start implementing Azure Content Delivery Network

That way the website can be made accessible to people around the world.
It caches and serves content from servers strategically placed worldwide, reducing the distance data has to travel. Making it more available.

This will ensure that my site's resources are quickly available, no matter where the users are located.



After a little while our website can be accessed through Azure CDN

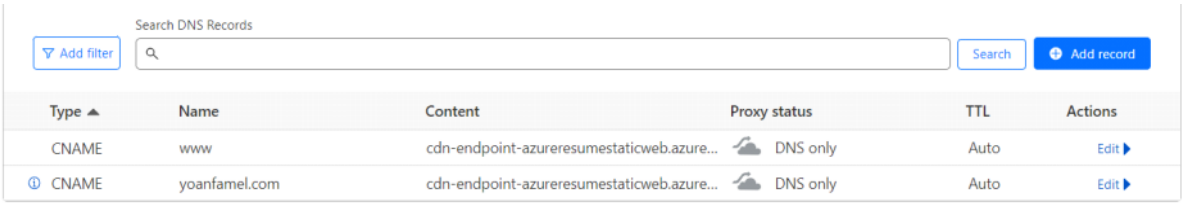


Now that I got the custom domain off of CloudFlare I can associate it to my CDN endpoint.

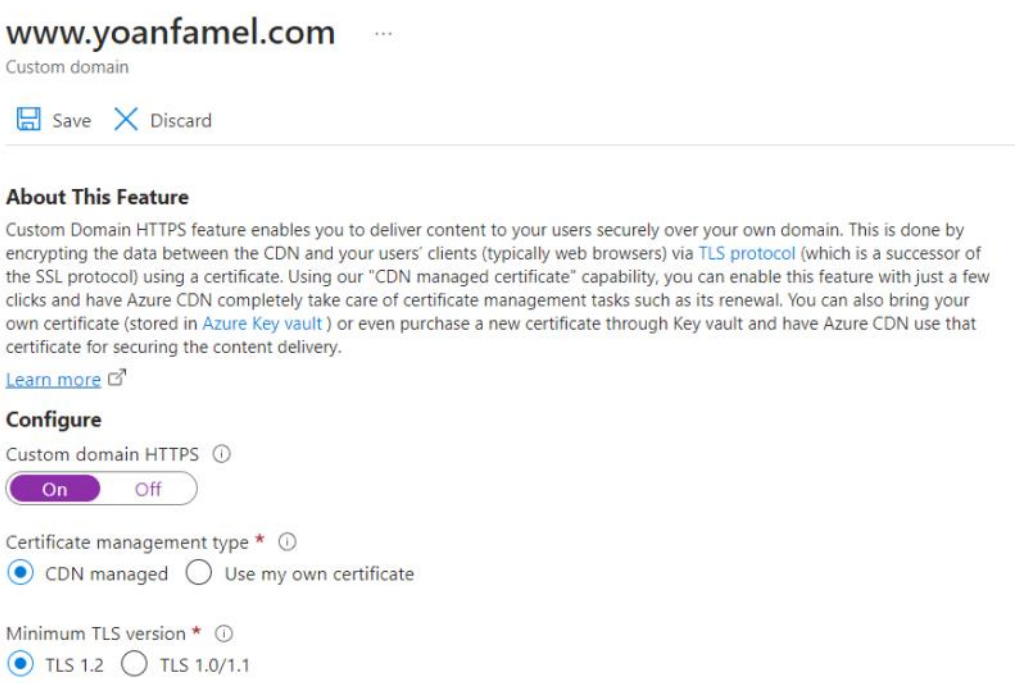
Let's create a CNAME record so that my source domain yoanfamel.com maps to my destination domain, in this case the CDN endpoint.

All traffic coming from the custom domain will be routed to the endpoint.

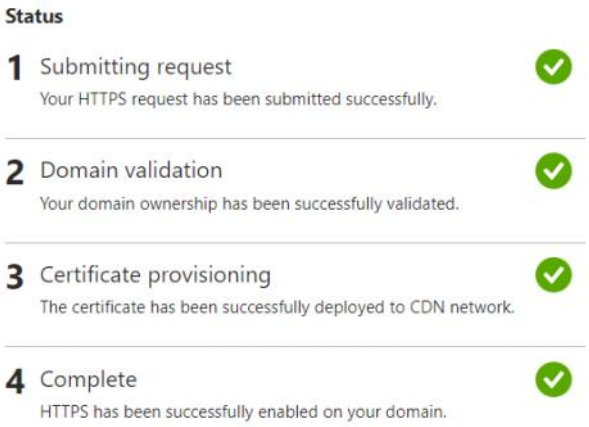
I added a few records on CloudFlare,



and another for the root. Then I associated the custom domain to the CDN endpoint and enabled HTTPS.



We wait for certificate validation:



Now to avoid this scenario:

we need to enforce https on users with CDN rules engine:

Select the action to apply to the requests that satisfy the match condition:

- Select Add action, and then select URL redirect.
- For Type, select Found (302).
- For Protocol, select HTTPS.
- Leave all other fields blank to use incoming values.

All we have to do now is refresh the page and observe the redirection when using http.

GitHub Actions: Setting up a CI/CD pipeline

12 September 2023 02:21

Purge CDN endpoint cache on new code Integration

12 September 2023 17:02

Back End API

10 September 2023 23:54

Back End

10 September 2023 23:54