Migrer un site d'AWS vers bunny.net

Migration des sites web d'AWS S3 et Cloudshare vers Bunny.net

Mon Makefile génère tout le contenu du site à partir de fichiers de balisage restructuredText dans un répertoire local de mon ordinateur portable. Pour le site hébergé sur Amazon AWS, j'utilisais trois scripts Bash simples pour publier ce contenu. Un premier script, «deploy.aws», synchronise tout le contenu avec un compartiment S3. Un deuxième script, «invalidate.sh», invalide les fichiers modifiés dans le cache Cloudshare. Enfin, un troisième script, «wait.sh», bloque l'exécution jusqu'à la fin de l'invalidation.

Pour effectuer la migration d'AWS vers Bunny.net, il me suffit d'implémenter des versions alternatives des scripts de déploiement. L'ensemble de mon cycle de maintenance de blog reste inchangé.

Configuration d'un compte bunny.net et des ressources pour le site

- J'ai d'abord créé un compte bunny.net et je m'y suis connecté.
- J'ai créé et nommé (par exemple, blog-example-com) un "storage zone" standard.
- J'ai créé et nommé (par exemple, blog-example-com-zone) un "pull zone" (sous l'onglet CDN) associée au nom du "storage zone". J'ai également ajouté un nom d'hôte au "pull zone". Ce nom doit correspondre au nom de domaine de mon site (par exemple, blog.example.com).
- Avec mon service de noms de domaine actuel (dans mon cas, Squarespace), je dois remplacer l'ancien enregistrement CNAME qui pointe vers mon ancien compartiment AWS S3 par un enregistrement comme celui-ci:

  blog.example.com CNAME log-example-com-zone.b-cdn.net

Script de déploiement

À chaque modification, suppression ou ajout de contenu lors de la maintenance de mon blog, je dois publier ces modifications au "storage zone" de bunny.net et purger le "pull zone" pour diffuser les modifications sur le CDN.

- Pour cela, j'ai créé un script de déploiement nommé **deploy.bunny**.
- Pour que ce script fonctionne, je dois configurer des identifiants sur mon ordinateur portable. J'ai donc écrit un autre script court, **login.bunny**, qui s'en charge. Il suffit de l'exécuter une seule fois. Ensuite, les identifiants seront stockés dans $HOME/.bunny/credentials.

deploy.bunny

```
#!/usr/bin/env bash
#
#
set -euo pipefail

MYPNAME=$(basename ${0})
ARG1NAME=${1:-NOARG}

# --- Configuration ---
STNAME="blog-bernatchez-net"
LOCDIR="$(pwd)/bucket"
PULLZONENAME="blog-bernatchez-net-zone"
# --- End Configuration ---
echo MYPNAME $MYPNAME
echo LOCDIR $LOCDIR
echo STNAME $STNAME
echo PULLZONEID $PULLZONEID
```

```
echo PULLZONENAME $PULLZONENAME


# 1. Sync up to bunny zone
echo ${MYPNAME}: npx --yes bunny-transfer@latest sync ${LOCDIR} ${STNAME}
npx --yes bunny-transfer@latest sync ${LOCDIR} ${STNAME}
#  use move instead of sync if you dont want --delete

# 2. purge invalidates all cached objects
echo ${MYPNAME}: npx --yes bunny-transfer@latest purge $PULLZONENAME
npx --yes bunny-transfer@latest purge $PULLZONENAME

echo ${MYPNAME}: "Deployment complete."
```

deploy.login

```
#!/usr/bin/env bash
#
#
set -euo pipefail

MYPNAME=$(basename ${0})
ARG1NAME=${1:-NOARG}

# --- Configuration ---
STNAME="blog-bernatchez-net"
PULLZONENAME="blog-bernatchez-net-zone"
APIKEY="xxxxxxxx-xxxx-etc"
# --- End Configuration ---
echo MYPNAME $MYPNAME
echo STNAME $STNAME
echo PULLZONENAME $PULLZONENAME
echo APIKEY $APIKEY

# login
echo $MYPNAME: Login creates the file "~/.bunny/credentials"
echo $MYPNAME: Until you do that, your other bunny commands wont work
echo $MYPNAME: npx --yes bunny-transfer@latest login default $APIKEY
npx --yes bunny-transfer@latest login default $APIKEY

# verify
echo $MYPNAME: This verifies we have got it right
echo $MYPNAME: npx --yes bunny-transfer@latest who-am-i
npx --yes bunny-transfer@latest who-am-i
```

Dernière touche

Pour intégrer le script **deploy.bunny** à mon flux de travail, j'ai ajouté un appel à celui-ci à la fin de la cible «pubhtml» de mon fichier Makefile. Ainsi, il est exécuté chaque fois que je régénère le contenu pour publication.