Migrar un sitio de AWS a Bunny.net

Migración de sitios web de AWS S3 y CloudShare a Bunny.net

Mi Makefile genera todo el contenido del sitio a partir de archivos de marcado restructuredText en un directorio local de mi portátil. Para el sitio alojado en AWS, tenía tres sencillos scripts BASH para publicar el contenido. Uno, "deploy.aws", sincroniza todo el contenido con un bucket de S3. Otro, "invalidate.sh", invalida los archivos modificados en la caché de CloudShare, y otro, "wait.sh", bloquea la operación hasta que finaliza la invalidación.

Para migrar de AWS a Bunny.net, solo necesito implementar versiones alternativas de los scripts. Mi ciclo de trabajo de mantenimiento del blog se mantendrá igual.

Configuración de la cuenta y los recursos de Bunny.net para dar servicio al sitio

- Primero, obtuve una cuenta de bunny.net e inicié sesión.
- Creé y nombré (por ejemplo, blog-example-com) un "standard storage zone".
- Creé y nombré (por ejemplo, blog-example-com-zone) un "pull zone" (en la pestaña CDN) y la asocié con el nombre del "standard storage zone". También agregué un nombre de host al "pull zone" Este debe corresponder al nombre de dominio de mi sitio web (por ejemplo, blog.example.com).
- Usando mi servicio DNS actual (en mi caso, Squarespace), necesito reemplazar el registro CNAME antiguo que apunta a mi antiguo bucket de AWS S3 con algo como esto:

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

Scripts de despliegue

Cada vez que modifico, elimino o añado contenido durante el mantenimiento de mi blog, necesito publicar esos cambios hasta mi "storage zone" de bunny.net y vaciar el "pull zone" para causar la redistribution de mi contenido al CDN.

- Para eso, necesité crear un nuevo script de despliegue llamado **deploy.bunny**.
- Para que mi script funcione, debo configurar las credenciales en mi portátil, así que escribí otro script corto que hace eso, llamado **login.bunny**. Solo es necesario ejecutarlo una sola vez. Después, las credenciales se guardarán en $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
```

Toque final

Para integrar el script deploy.bunny en mi flujo de trabajo, añadí una llamada al mismo al final del objetivo "pubhtml" de mi archivo Makefile. De esta forma, se ejecuta cada vez que regenero el contenido para su publicación.