Compte-Rendu

Objet	Date 30/04/2020	Référence B1B/UF-3
Présentation premier projet réseau, infrastructure et systèmes d'information Streamify	Rédacteur Ianis BORDREZ Laurent MARECHAL Akram ABDOUL-WAHAB	
Destinataire Léo GODEFROY John DREYFUS Responsable filière B1 Informatique		

I. Préparation

A. Choix du projet

Tout d'abord, concernant le choix du projet, nous étions plutôt d'accord concernant la création d'un site web. Ceci est dut au fait que le concept de la création d'un site web était assez flou pour nous. Nous étions donc intéressé de savoir comment se déroule les différentes étapes de ce processus.

De plus, nous serons sûrement amené à créer un site web à un moment ou à un autre durant notre carrière, il est donc utile de s'exercer et comprendre les mécanismes de base en amont.

Dans un second temps, nous avons longuement hésité pour le thème général du site. En effet, plusieurs thèmes ont été proposés au sein du groupe, comme par exemple: un forum de jeu, un site de streaming d'animés, un site de streaming de music, un site de streaming de film… Nous avons finalement opté pour un site de streaming de musique qui serait donc hébergé sur un serveur web.

Enfin, ce projet serait notamment intéressant pour la simple raison que le groupe apprécie écouter de la musique fréquemment. De ce fait, pour nous si nous voulons stocker des musiques en ligne sans l'aide d'autre outils.

B. Choix des outils

1. Software
    - NGINX: Système est un logiciel libre de serveur Web ainsi qu'un proxy inverse (reverse proxy). C'est un outil simple à mettre en place et d'utilisation mais aussi assez puissant pour l'ampleur de notre projet. C'est donc le choix optimal dans notre situation.

    - Flask: Flask est un framework open-source de développement web en Python. Son but principal est d'être léger, afin de garder la souplesse de la programmation Python, associé à un système de templates. Pour notre projet ce framework était largement suffisant pour les tâches à accomplir.

    - BorgBackup: BorgBackup (ou Borg) est un programme de déduplication de sauvegarde. Optionnellement, il supporte la compression et l'encryption authentifiée. C'était notre premier outil de Backup qui nous a permit de mettre en place une automatisation des Backups rapide et simplement même s'il y a eu quelque soucis (voir partie Troubleshoot)

2. Langage

    - Python: Langage est un langage de programmation interprété, multi-paradigme et multiplateformes lequel nous avons pas mal pratiqué durant l'année. De plus une multitude de framework utiles sont disponible dans ce langage. Nous avons donc choisis de l'utiliser

    - HTML : L'HTML5, est le langage de balisage conçu pour représenter les pages web. C'est un langage permettant d'écrire de l'hypertexte, d'où son nom. Il nous a permis de construire la base de nos page web, il est donc essentiel.

    - CSS:Les CSS de l'anglais Cascading Style Sheets, forment un langage informatique qui décrit la présentation des documents HTML et XML. Il nous a donc permit de styliser nos pages.

    - Bash: Le Bash est un shell, c'est-à-dire un interpréteur de commandes. Nous avons pu automatiser certaine de nos actions grâce à celui-ci.

    - Javascript: JavaScript est un langage de programmation de scripts principalement employé dans les pages web interactives mais aussi pour les serveurs avec l'utilisation de Node.js. Dans notre cas nous l'avons seulement utilisé dans son côté interaction avec les pages web.

II. Conception

A. Préparation de l'environnement

Tout d'abord vous devrez louer un serveur sur le site que vous trouver le plus adapté pour vous. Lorsque vous avez établir une connexion au terminal, vous pouvez suivre les commandes suivantes pour mettre en place le serveur.

Mise en place du serveur:

- sudo apt install python3-pip nginx
- sudo /etc/init.d/nginx start
- sudo rm /etc/nginx/sites-enabled/default
- sudo touch /etc/nginx/sites-available/flask_settings
- sudo ln -s /etc/nginx/sites-available/flask_settings /etc/nginx/sites-enabled/flask_settings
- nano /etc/nginx/sites-enabled/flask_settings

```
server {
        location / {
            proxy_pass http://127.0.0.1:8000;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
        }
}
```

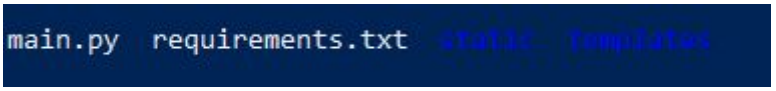
- sudo /etc/nginx/init.d/nginx restart
- pip3 install virtualenv flask gunicorn flask_monitoringdashboard
- cd /home/ianis/
- mkdir streamify
- cd streamify

- python3 -m venv env
- source env/bin/activate
- export FLASK_APP=main.py
- flask run
- gunicorn main :app

Ouverture des ports :

- port 80 (http)
- port 5000 (flask)
- port 19999 (netdata)

L'arborescence du dossier est notamment important dans flask pour qu'il puisse reconnaître les différents type de fichier et leur utilité.



Dans static nous avons les musiques que nous voulons afficher et dans templates les pages webs.
De plus, nous pouvons remarquer la présence d'un fichier main.py qui représente le coeur de l'application flask et qui nous permet de servir les différentes pages. Pour le faire marcher, il faut compléter le fichier de cette manière:

```
from flask import Flask, render_template,request
from pathlib import Path

import os
import flask_monitoringdashboard as dashboard

app = Flask(__name__, static_folder="static")
dashboard.bind(app)

@app.route("/upload", methods=["POST"])
def upload():
    try:
        file = request.files["inputFile"]
        if not file.filename.lower().endswith((".mp3",".wav")):
                return "Mauvais format"
        elif Path(f"static/music/{file.filename}").is_file():
                return "Cette musique existe déjà"
```

```
        file.save(f"static/music/{file.filename}")
        return f"Musique envoyée {file.filename}"
    except Exception as e:
        return f"Erreur lors de l'envoi : {e}"

@app.route("/")
def index():
        return render_template("index.html")

@app.route("/music")
def music():
    musicList = []
    files = os.listdir("static/music")
    for f in files:
        musicList.append(f)
    return render_template("music.html", musicList=musicList)
```

Installation de BorgBackup :

```
~$ borg init --encryption=repokey /home/ianis/borg

~/borg$ borg create ~/borg/::Monday ~/Streamify/static/music/

~$ borg list ~/borg/::Monday

~/borg_test$ borg extract ~/borg/::Monday

~$ cd ~/borg_test/

~/borg_test$ touch backup.sh

~$ sudo nano backup.sh

#!/bin/sh

export BORG_REPO=/home/ianis/borg/

export BORG_PASSPHRASE='borg_streamify'

info() { printf "\n%s %s\n\n" "$( date )" "$*" >&2; }

trap 'echo $( date ) Backup interrupted >&2; exit 2' INT TERM

info "Starting backup"

borg create                         \
```

```
    --verbose                       \
    --filter AME                    \
    --list                          \
    --stats                         \
    --show-rc                       \
    --compression lz4               \
    --exclude-caches                \
                                    \
    ::'{hostname}-{now}'            \
     ~/Streamify/static/music/      \
backup_exit=$?
info "Pruning repository"
borg prune                          \
    --list                          \
    --prefix '{hostname}-'          \
    --show-rc                       \
    --keep-daily    7               \
prune_exit=$?
global_exit=$(( backup_exit > prune_exit ? backup_exit : prune_exit ))
if [ ${global_exit} -eq 0 ]; then
    info "Backup and Prune finished successfully"
elif [ ${global_exit} -eq 1 ]; then
    info "Backup and/or Prune finished with warnings"
else
    info "Backup and/or Prune finished with errors"
fi
exit ${global_exit}
```

~$ ./backup.sh

B. Utilisation

Comment utiliser notre projet ?

Connectez vous à http://34.65.67.122/

Vous pouvez ainsi envoyer vos musiques sur le serveur en cliquant sur “parcourir”. L’extension du fichier

vous pouvez cliquer sur le bouton “music” pour accéder à toutes les musiques envoyés par tous les utilisateurs.

Vous pouvez aussi écrire dans la barre de recherche pour trouver instantanément la musique que vous cherchez.

Lors de la mise en place des différents outils. Nous avons rencontrés différent problèmes auxquels nous avons dû trouver des solutions que ce soit par nos propre moyen ou alors en nous aidant de documentation en ligne:

C. Troubleshoot

1. BorgBackup

Problème:

Comme dit précédemment est un outil permettant de faire des sauvegardes de données. Pour rendre nos sauvegardes automatique il a fallu confectionner le script bash visible ci-dessus. Ce script permet d’effectuer un backup avec la commande “borg create”. Pour le rendre automatique, nous avons dû le mettre dans la crontab. Cependant lors de l'exécution du script par la crontab, borg n’avait plus la permission d'interagir avec les données pour cause de manque de droit. Ce problème était causé par le fait que la crontab lancé le script en tant que root mélangeant ainsi les droits dans le dépôt.

Solution:

Pour régler ce soucis nous avons choisis de supprimer le dépôt créer et d'en créer un nous mais aussi de changer la crontab pour que l'utilisateur "ianis" lance le script comme montré ci-dessous:

```
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
57 16   * * *   ianis   /home/ianis/backup.sh
#
```

2. Netdata

Netdata est un outil open source permettant de visualiser et de contrôler des mesures en temps réel, optimisé pour accumuler tous type de données, telles que les utilisations CPU, l'activité du disque, etc… Pour utiliser cet outil nous avons lancer un script fournis qui permet l'installation automatique. Une fois installé, nous devons ajouter un fichier dans "/etc/nginx/sites-enabled". Pour cela nous décidons de créer le fichier "netdata" avec le contenu ci dessous :

```
upstream netdata {
    server 127.0.0.1:19999;
    keepalive 64;
}

server {
   listen 80;

   location = /netdata {
        return 301 /netdata/;
   }

   location ~ /netdata/(?<ndpath>.*) {
        proxy_redirect off;
        proxy_set_header Host $host;

        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_http_version 1.1;
        proxy_pass_request_headers on;
        proxy_set_header Connection "keep-alive";
        proxy_store off;
        proxy_pass http://netdata/$ndpath$is_args$args;

       # gzip on;
       # gzip_proxied any;
       # gzip_types *;
    }
}
```

Nous avons un autre fichier pour activer flask sur nginx. Le problème est qu'il nous était impossible d'accéder à flask ou netdata si les deux "applications" étaient activés.

Solution temporaire:

Nous avons donc choisis de ne plus utiliser netdata à cause des conflits. Pour le monitoring du site, nous utilisons dès à présent le système de monitoring intégré à Flask.

D. Améliorations

Pour ce projet, différent aspect aurait pu être améliorer.
Voici quelques points sur lesquels des améliorations sont possibles:

- HTML/CSS: Le côté visuel du site n'a pas été une priorité lors de la création de ce projet. De ce fait, visuellement le site pourrait être améliorer de multiples façons.

- Sécurité: On note que, la vérification des fichiers lors de l'envoi se fait par la vérification de l'extension du fichier. Or, celle-ci ne représente pas forcément la nature réelle du fichier en question. Pour une question de sécurité, il serait très préférable de vérifier les premiers caractère composant les données du fichier que l'on peut récupérer avec la commande "cat" par exemple, ce qui nous donnerait donc la vrai nature du fichier.

- Monitoring: Pour le monitoring, nous aurions préféré utiliser Netdata cependant pour des raisons mentionnées ci-dessus, nous n'avons pas pu l'utiliser. On notera que cet outils est beaucoup plus adapté à un contexte serveur que le dashboard intégré à Flask.

- DDOS: On soulignera aussi le fait que le serveur de google cloud platform de base ne fournit pas de solution gratuite contre le DDOS. De ce fait, une amélioration en terme de sécurité et stabilité du serveur aurait pu être fait.

III. Conclusion

Pour conclure, nous pouvons affirmer que cette expérience était très enrichissante que ce soit dans l'apprentissage de nouveaux outils, la programmation, l'organisation d'un réseau et les différentes étapes de la mise en place d'un site web. Nous avons notamment pu améliorer notre travail d'équipe et l'organisation des différentes tâches, apprendre à trouver des solutions face au problèmes que l'on rencontre de nos propre moyens.

Plus tard, nous aimerions participer à la création d'un site plus abouties et poussé qui ressemblerait davantage à un réel projet professionnel.