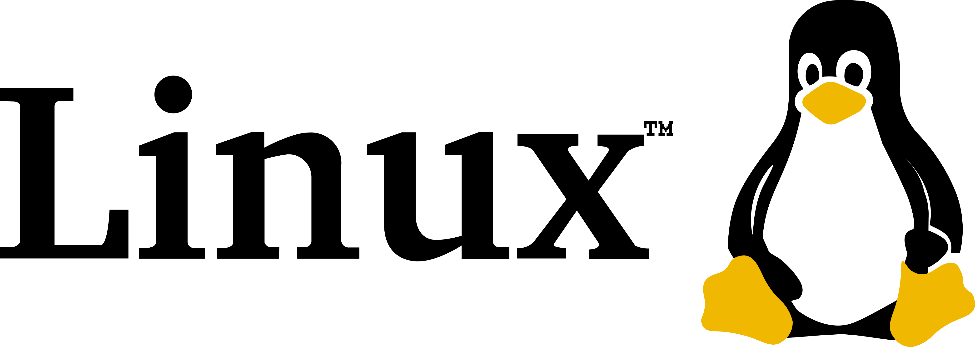
# **DOCUMENTATION TECHNIQUE**

## **Projet :** Coursero – Plateforme sécurisée de dépôt et correction automatique de code



### Sommaire

### **Introduction**

#### Objectif du projet

Le projet *Coursero* vise à fournir une **plateforme sécurisée de dépôt de code** étudiants, dotée d’un système de **correction automatique**, de **notation dynamique** et d’une interface de consultation intuitive. Le projet permet aux étudiants de :

* S’authentifier de manière sécurisée (via token JWT)
* Déposer leurs fichiers .c ou .py dans une interface web
* Obtenir automatiquement la correction + une note
* Voir leurs anciennes soumissions

Du côté système :

* Un script (worker.sh) traite en arrière-plan les soumissions
* Le code est exécuté de manière sécurisée (avec timeout, inputs isolés)
* L’infrastructure est segmentée pour maximum d’isolation

#### Contexte

Le projet simule une situation proche d’un environnement universitaire réel, où les étudiants soumettent des exercices codés qui sont corrigés de façon automatique, offrant à la fois un gain de temps et un cadre éducatif plus moderne et évolutif.

### **Architecture et Infrastructure**

#### Schéma général de l’infrastructure

**SRV-WEB (srv-web01 et srv-web02) :**

* Héberge le site Coursero
* Gère les requêtes utilisateurs : Authentification, affichage des soumissions, poses de fichiers…
* Utilise Apache (HTTP/HTTPS) : Peut être distribué via HAProxy
* Communique avec :
  + Srv-app : pour traitement/correction
  + Srv-bdd : pour lire/écrire MySQL

**SRV-BDD :**

* Stocke toutes les données :
  + Etudiants (étudiants)
  + Soumissions (submissions)
  + Résultats
* Permet les accès via MySQL : via un utilisateur sécurisé dédié (webuser)

**SRV-APP :**

* Exécute des scripts Python/Bash : traitement des fichiers déposés (script worker.sh)
* Gère les corrections automatiques : Lit la BDD submissions / exécute le code étudiant / compare les résultats
* Contient le worker.sh : Script qui s’exécute manuellement ou via cron
* Fait proxy/load balancing (HAProxy) : Répartit la charge entre srv-web01 / web02
* Contient les modèles de tests : Répertoire /templates/ avec config.json, input/output\_X.txt
* Connexions sortantes vers :
  + Srv-bdd : pour SQL
  + Srv-web : via HAProxy

#### Spécifications des serveurs

| **Serveur** | **Rôle** | **IP** | **Services** |
| --- | --- | --- | --- |
| **SRV-WEB** | Hébergement du site | 192.168.146.101 et .102/24 | Serveur web apache (HTTPS/HTTP) |
| **SRV-APP** | Correction automatique | 192.168.146.105/24 | Script worker.sh (correction de soumissions) |
| **SRV-BDD** | Serveur base de données | 192.168.146.103 et .104 | DHCP, DNS, iptables, NAT, Routage, VLANs. |

### **Configuration Réseau**

#### Ports et Protocoles principaux

| **Service** | **Port** | **Protocole** | **Description** |
| --- | --- | --- | --- |
| HTTP | 80 | TCP | Accès web (redirigé vers 443) |
| HTTPS | 443 | TCP | Accès web sécurisé |
| DNS | 53 | TCP/UDP | Service DNS |

#### Pare-feu et Réseaux

| **Zone** | **Plage IP** | **Passerelle** | **Masque** | **VLAN** |
| --- | --- | --- | --- | --- |
| VLAN-WEB | 192.168.10.0/24 | 192.168.10.254 | 255.255.255.0 | 10 |
| VLAN-APP | 192.168.20.0/24 | 192.168.20.254 | 255.255.255.0 | 20 |
| VLAN-BDD | 192.168.30.0/24 | 192.168.30.254 | 255.255.255.0 | 30 |

### **Configurations Technique**

#### Choix des technologies

**OS :**

**Debian 11** sur toutes les machines → stabilité, compatibilité

**Web & Auth :**

**PHP + Apache2** : technologie simple, largement déployée sur Debian

**Token JWT** : JSON Web Token pour authentification sans session serveur (meilleure sécurité, REST-like)

**bcrypt (password\_hash)** : pour hacher les mots de passe étudiants

**Bootstrap/CSS pur** : interface sobre, responsive, sans dépendance externe

**Base de données :**

**MariaDB** (MySQL) hébergeant :

* etudiants : id, email, password
* submissions : note, statut, fichier, etc.
* Comptes sécurisés webuser isolés avec mot de passe

**Scripts backend :**

worker.sh :

* écrit en **bash**
* corrige automatiquement les soumissions en\_attente
* compare les fichiers output
* met à jour la base
* Exécuté via **cron** ou **manuellement**

**Sécurité & réseau :**

* **VLANs 802.1Q** séparant Web / App / DB
* Machine **fw-coursero** faisant :
* **Routage inter-VLAN**
* **NAT vers Internet**
* **DHCP (en option)**
* **Pare-feu (iptables ou ufw)**
* **iptables NAT masquerading** pour restaurer Internet
* Accès MySQL restreint **au seul srv-app**

**Fonctionnement plateforme**

**Étudiant :**

1. Se connecte via login.php
2. Reçoit un JWT s’il est authentifié
3. Est redirigé vers dashboard.php?token=...
4. Peut :

* envoyer un fichier (upload.php)
* voir ses notes (soumissions.php)
* modifier son mot de passe (change\_password.php)

**Admin :**

* Les scripts corrigent automatiquement
* Les soumissions sont notées, évaluées et historisées
* Possibilité d'ajouter des tests dans /templates/exoX/

#### Tableau des Machines Virtuelles

### **SRV-WEB01 / 02**

Installation d’apache sur les équipements : Apt-get install apache2

La plateforme doit être accessible en HTTPS, il est donc important de sécuriser la connexion. Nous utiliserons un certificat auto-signé :

* Activation du module ssl d’apache : a2enmod ssl
* Création d’un répertoire pour les certificats : mkdir -p /etc/apache2/ssl
* Génération d’un certificat auto-signé : sudo openssl req -x509 -nodes -days 365 -newkey
* rsa:2048 -keyout /etc/apache2/ssl/apache.key -out /etc/apache2/ssl/apache.crt

(valable 365 jours)

Une image contenant texte, capture d’écran, Police, noir et blanc

Le contenu généré par l’IA peut être incorrect.

Activation du site HTTPS :

Apache fournit un fichier de configuration par défaut pour SSL sité dans /etc/apache2/sites-available/default-ssl.conf  
Modification de ce fichier pour pointer vers nos certificats :

SSLCertificateFile /etc/apache2/ssl/apache.crt

SSLCertificateKeyFile /etc/apache2/ssl/apache.key

Définition du ServerName : srv-web01

Activation de la config ssl : a2ensite default-ssl

Cela créer un lien symbolique dans /etc/apache2/sites-enabled/ pointant vers le fichier modifié de sites-available  
test de la config apache : apache2ctl configtest

Redémarrage apache2 : systemctl reload apache2

Attribution des IPs statiques pour les deux équipements en modifiant le fichier /etc/network/interfaces

### **SRV-APP**

Installation HAProxy : apt update && apt install haproxy -y

Configuration HAProxy pour répartir la charge :

Edition du fichier nano /etc/haproxy/haproxy.cfg

Répertorier les serveurs web et gérer la répartition de charge :

global

log /dev/log local0

maxconn 2000

daemon

defaults

log global

mode http

timeout connect 5s

timeout client 50s

timeout server 50s

option httplog

option forwardfor

option redispatch

frontend http\_front

bind \*:80

use\_backend web\_servers

frontend https\_front

bind \*:443 ssl crt /etc/haproxy/haproxy.pem

use\_backend web\_servers

backend web\_servers

balance roundrobin

option httpchk HEAD / HTTP/1.1\r\nHost:localhost

server srv-web01 192.168.146.101:80 check

server srv-web02 192.168.146.102:80 check

Explication :

 **frontend http\_front** : Capture les requêtes sur le port **80** et les redirige vers les serveurs backend.

 **frontend https\_front** : Capture les requêtes **HTTPS** sur le port **443**. (Nous ajouterons le certificat SSL ensuite).

 **backend web\_servers** : Répartition **roundrobin** entre srv-web01 et srv-web02

Génération d’un certificat autosigné pour les tests pour HTTPS :

HAProxy gère HTTPS dcp :

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/haproxy/haproxy.key -out /etc/haproxy/haproxy.crt

Combinaison de la clé et du certificat :

sudo cat /etc/haproxy/haproxy.crt /etc/haproxy/haproxy.key > /etc/haproxy/haproxy.pem

Redélarrer HAProxy : sudo systemctl restart haproxy

sudo systemctl enable haproxy

Vérif : Testez maintenant l'accès depuis un autre poste du réseau vers **srv-app (192.168.146.105)** :

curl -I http://192.168.146.105

Si tout est bon, HAProxy répartira automatiquement le trafic entre srv-web01 et srv-web02.

Partage du répertoire /var/www/html entre les serveurs web :

Sur srv-app :

sudo apt install nfs-kernel-server -y

creation du repertoire /var/www/html : sudo mkdir -p /var/www/html

configuration du serveur nfs :

ajout de la config dans /etc/exports : nano /etc/exports

/var/www/html 192.168.146.101(rw,sync,no\_root\_squash)

/var/www/html 192.168.146.102(rw,sync,no\_root\_squash)

Ces lignes signifient que srv-web01 et srv-web02 pourront lire/écrire dans /var/www/html

Rechargement de la config NFS :

sudo exportfs -a

sudo systemctl restart nfs-kernel-server

sudo systemctl enable nfs-kernel-server

puis installation du client nfs sur les serveurs web : sudo apt install nfs-common -y

montage du dossier /var/www/html partagé depuis srv-app : sudo mount -t nfs 192.168.146.105:/var/www/html /var/www/html

on rends ce montage permanent en ajoutant dans /etc/fstab de srv-web01 et 02 :

192.168.146.105:/var/www/html /var/www/html nfs defaults 0 0

Rechargement fstab : sudo mount -a

**Test du Partage NFS**

Sur **srv-web01** :

sudo touch /var/www/html/test-srv-web01.txt

Sur **srv-web02** :

ls /var/www/html/

Le fichier test-srv-web01.txt **doit être visible sur srv-web02** car ils partagent maintenant le même répertoire.

Génération d’un certificat autosigné pour les tests pour HTTPS :

HAProxy gère HTTPS dcp :

sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-keyout /etc/haproxy/haproxy.key -out /etc/haproxy/haproxy.crt

Combinaison de la clé et du certificat :

sudo cat /etc/haproxy/haproxy.crt /etc/haproxy/haproxy.key > /etc/haproxy/haproxy.pem

Redélarrer HAProxy : sudo systemctl restart haproxy

sudo systemctl enable haproxy

Vérif : Testez maintenant l'accès depuis un autre poste du réseau vers **srv-app (192.168.146.105)** :

curl -I http://192.168.146.105

Si tout est bon, HAProxy répartira automatiquement le trafic entre srv-web01 et srv-web02.

Partage du répertoire /var/www/html entre les serveurs web :

Sur srv-app :

sudo apt install nfs-kernel-server -y

creation du repertoire /var/www/html : sudo mkdir -p /var/www/html

configuration du serveur nfs :

ajout de la config dans /etc/exports : nano /etc/exports

/var/www/html 192.168.146.101(rw,sync,no\_root\_squash)

/var/www/html 192.168.146.102(rw,sync,no\_root\_squash)

Ces lignes signifient que srv-web01 et srv-web02 pourront lire/écrire dans /var/www/html

Rechargement de la config NFS :

sudo exportfs -a

sudo systemctl restart nfs-kernel-server

sudo systemctl enable nfs-kernel-server

puis installation du client nfs sur les serveurs web : sudo apt install nfs-common -y

montage du dossier /var/www/html partagé depuis srv-app : sudo mount -t nfs 192.168.146.105:/var/www/html /var/www/html

on rends ce montage permanent en ajoutant dans /etc/fstab de srv-web01 et 02 :

192.168.146.105:/var/www/html /var/www/html nfs defaults 0 0

Rechargement fstab : sudo mount -a

**Test du Partage NFS**

Sur **srv-web01** :

sudo touch /var/www/html/test-srv-web01.txt

Sur **srv-web02** :

ls /var/www/html/

Le fichier test-srv-web01.txt **doit être visible sur srv-web02** car ils partagent maintenant le même répertoire.

Pour que le script fonctionne installation du binaire gcc pour compiler les fichier .c en utilisant gcc

Et installation de python3 pour exécuter les scripts python

### **SRV-BDD01 / 02**

Choix techno pour le site : mariaDB + PHP

Apt install mariadb-server -y

Vérifie que MariaDB démarre bien :

sudo systemctl status mariadb

**Sécurisation initiale** Exécute la commande pour ajouter un mot de passe root et désactiver les accès publics :

sudo mysql\_secure\_installation

Sur srv-db01 – configuration (master)

Edition du fichier : sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf

Sous [mysqld], ajout des lignes :

bind-address = 0.0.0.0

server-id = 1

log\_bin = /var/log/mysql/mysql-bin.log

binlog\_do\_db = corrections

🔹 **Explications** :

* server-id = 1 → identifiant unique du serveur dans le cluster
* log\_bin → Activation du **log des transactions**
* binlog\_do\_db = corrections → Indique que seule cette base sera répliquée

redémarrage mariadb : sudo systemctl restart mariadb

Création user de réplication : sudo mysql -u root -p

CREATE USER 'repli'@'%' IDENTIFIED BY 'password123';

GRANT REPLICATION SLAVE ON \*.\* TO 'repli'@'%';

FLUSH PRIVILEGES;

Configuration srv-db02 (slave)

Edition fichier : sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf

Puis ajout des lignes :

bind-address = 0.0.0.0

server-id = 2

relay\_log = /var/log/mysql/mysql-relay-bin.log

redémarrage mariadb : sudo systemctl restart mariadb

connection à mysql (sudo mysql -u root -p), puis config de la réplication :

CHANGE MASTER TO

MASTER\_HOST='192.168.146.101',

MASTER\_USER='repli',

MASTER\_PASSWORD='password123',

MASTER\_LOG\_FILE='mysql-bin.000001',

MASTER\_LOG\_POS=4;

START SLAVE;

Vérif de l’état de la réplication : SHOW SLAVE STATUS\G;

Si "Slave\_IO\_Running: Yes" et "Slave\_SQL\_Running: Yes" = tout fonctionne

Connexion du site web à la BDD :

Création base de données :

Sur srv-db01 (et répliqué automatiquement sur srv-db02):

CREATE DATABASE corrections;

CREATE USER 'webuser'@'%' IDENTIFIED BY 'webpassword';

GRANT ALL PRIVILEGES ON corrections.\* TO 'webuser'@'%';

FLUSH PRIVILEGES;

Déploiement de la première version du site :

Création fichier /var/www/html/index.php sur srv-app :

<?php

$servername = "192.168.146.101"; // Adresse du serveur DB principal

$username = "webuser";

$password = "webpassword";

$dbname = "corrections";

// Connexion à la base de données

$conn = new mysqli($servername, $username, $password, $dbname);

if ($conn->connect\_error) {

die("Échec de la connexion : " . $conn->connect\_error);

}

echo "Connexion réussie à la base de données !";

// Ferme la connexion

$conn->close();

?>

Test de l’accès : http://192.168.146.105

Création de la base de données et la table étudiants :

Sur srv-db01 : sudo mysql -u root -p

Puis :

CREATE DATABASE corrections;

USE corrections;

CREATE TABLE etudiants (

id INT PRIMARY KEY AUTO\_INCREMENT,

email VARCHAR(100) UNIQUE NOT NULL,

password VARCHAR(255) NOT NULL

);

**Explication** :

* id : Clé primaire auto-incrémentée pour chaque étudiant.
* email : L'adresse email est unique pour chaque étudiant.
* password : Le mot de passe est stocké de façon sécurisée.

Ajouter un étudiant test dans la base :

INSERT INTO etudiants (email, password)

VALUES ('etudiant@example.com', PASSWORD('password123'));

Tests : [**http://192.168.146.105/login.php**](http://192.168.146.105/login.php)

**Puis** [**etudiant@example.com**](mailto:etudiant@example.com) **et password123**

**SRV-WORKER**

**Active ufw et bloque tout sauf vers la base de données :**

**sudo apt install ufw -y**

**sudo ufw default deny outgoing**

**sudo ufw default deny incoming**

**# Autoriser uniquement l'accès vers la base de données**

**sudo ufw allow out to 192.168.30.103 port 3306 proto tcp**

**# (Optionnel) autoriser SSH entrant (si besoin)**

**sudo ufw allow in ssh**

**sudo ufw enable**