Projet GLPI et Site Web

Epreuve E5 : Support et mise à disposition de services informatiques

COUSIN Victor

Introduction

Contexte du projet

Dans le cadre d’un des projets en BTS SIO option SISR, j'ai réalisé le déploiement complet d'une infrastructure de ticketing accessible via le web sur machine virtuelle. Ce projet s'inscrit dans le contexte de Tierslieux86, un espace collaboratif qui souhaite permettre à ses utilisateurs de signaler facilement des incidents informatiques via un formulaire web simple, sans avoir besoin d'accéder à l'interface d'administration GLPI.

Objectifs du projet

Objectifs techniques

Les objectifs techniques du projet sont multiples :

- Déployer un serveur web Apache sur Ubuntu Server 24.04 LTS
- Créer un site vitrine avec une page de contact permettant de signaler des incidents
- Installer et configurer GLPI qui est un logiciel open source de gestion de tickets
- Connecter le formulaire web à GLPI via son API REST pour créer automatiquement des tickets.

L’environnement technique se compose d’une machine virtuelle Ubuntu Server 24.04 LTS sur VMware Workstation. Son IP sera « 192.168.85.128 ». Et hébergera un serveur web : Apache 2.4 ainsi qu’une base de données MariaDB, le logiciel de ticketing sera GLPI 10.0.10. Et un PC hôte Windows qui aura pour IP « 192.168.85.1 ».

Présentation de l’infrastructure

L'ensemble du projet est hébergé sur une unique machine virtuelle VMware en mode NAT. Le PC hôte Windows communique avec la VM via le réseau virtuel VMnet8. La résolution des noms de domaine locaux est assurée par les fichiers hosts des deux machines.

Composant	Détail
Hyperviseur	VMware Workstation
OS de la VM	Ubuntu Server 24.04 LTS
IP de la VM	192.168.85.128/24
Passerelle VMware NAT	192.168.85.2
Interface réseau	Ens33
Serveur web	Apache 2.4.58
Base de données	Maria DB 10
Langage serveur	PHP 8.3
Outil de ticketing	GLPI 10.0.10

Schéma réseau et flux de communication Le schéma ci-dessous illustre le flux complet d'une demande : l'utilisateur remplit le formulaire sur le site vitrine, le script PHP contacte l'API GLPI en interne, et un ticket est créé automatiquement dans la base de données.

Deux noms de domaine locaux seront utilisés, définis dans les fichiers hosts :
http:/ /glpi.local qui est l’accès à l’interface d'administration GLPI
http:/ /www.monsite.local qui est l’accès au site vitrine avec formulaire de contact

INSTALLATION D'APACHE, PHP ET MARIADB

Installation des paquets

Dans un premier temps le système est mis à jour . Ensuite, Apache, PHP avec toutes ses extensions requises par GLPI, et MariaDB sont installés en une seule commande comme ceci :

```
sudo apt update && sudo apt upgrade -y

sudo apt install apache2 -y

sudo apt install php php-curl php-json php-mysqli php-mbstring \
php-xml php-gd php-intl php-zip php-bz2 php-ldap \
libapache2-mod-php -y

sudo apt install mariadb-server -y
```

Les diverses extensions PHP installées sont nécessaires au bon fonctionnement de GLPI :

Extension PHP	Rôle
Php-mysqli	Communication avec MariaDB / MySQL
Php-curl	Requêtes HTTP (API REST)
Php-xml / php-mbstring	Traitement des données XML et chaînes multi-octets
Php-gd	Génération d'images (graphiques GLPI)
Php-int	Internationalisation et formatage des dates
Php-zip / php-bz2	Gestion des archives (plugins GLPI)
Php-ldap	Authentification LDAP / Active Directory

Vérification des services

Les services Apache et MariaDB sont activés pour démarrer automatiquement au lancement du serveur :

```
sudo systemctl enable apache2 mariadb
sudo systemctl start apache2 mariadb

# Vérifier que tout soit actif

sudo systemctl status apache2
sudo systemctl status mariadb
```

```
root@pc-admin:~$ sudo systemctl status apache2
[sudo] password for bob:
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: enabled)
     Active: active (running) since Fri 2026-02-27 09:42:39 UTC; 58s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 1079 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 1259 (apache2)
      Tasks: 6 (limit: 4545)
     Memory: 22.3M (peak: 22.5M)
        CPU: 349ms
     CGroup: /system.slice/apache2.service
             ├─1259 /usr/sbin/apache2 -k start
             ├─1376 /usr/sbin/apache2 -k start
             ├─1378 /usr/sbin/apache2 -k start
             ├─1379 /usr/sbin/apache2 -k start
             ├─1384 /usr/sbin/apache2 -k start
             └─1386 /usr/sbin/apache2 -k start

févr. 27 09:42:38 pc-admin systemd[1]: Starting apache2.service - The ApacheHTTPServer...
févr. 27 09:42:39 pc-admin apachectl[117]: AH00558: apache2: Could not reliably determine the s
févr. 27 09:42:39 pc-admin systemd[1]: Started apache2.service - The Apache HTTP Server.
lines 1-20/20 (END)
root@pc-admin:~$ sudo systemctl status mariadb
● mariadb.service - MariaDB 10.11.14 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: enabled)
     Active: active (running) since Fri 2026-02-27 09:42:42 UTC; 1min 3s ago
```

Pour confirmer qu'Apache réponde correctement, accédez à « http://192.168.85.128 » depuis le navigateur du PC hôte. La page par défaut d'Apache s'affichera, confirmant que le serveur web est opérationnel.

Sécurisation de MariaDB

Après l'installation, MariaDB est sécurisé avec le script fourni. Ce script supprime les utilisateurs anonymes, désactive la connexion root distante et supprime la base de test :

```
sudo mysql_secure_installation
# Répondre : n (pas de plugin mot de passe)
# Puis : y, y, y, y pour sécuriser l'ensemble
```

Création de la base et de l'utilisateur GLPI

Une base de données dédiée est créée pour GLPI, ainsi qu'un utilisateur MariaDB avec les droits nécessaires uniquement sur cette base (principe du moindre privilège) :

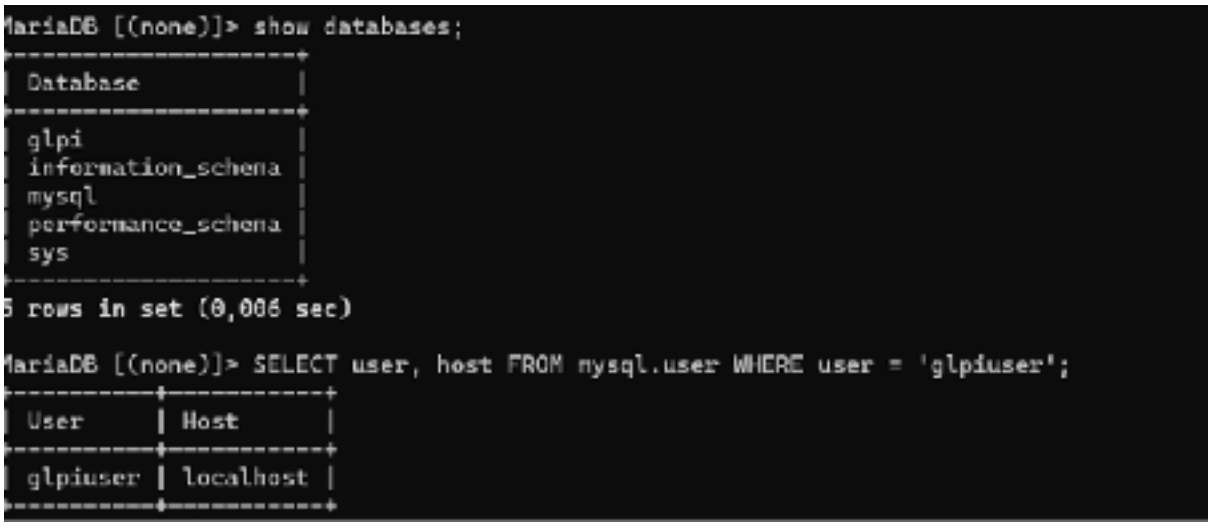
```
sudo mysql -u root -p

-- Création de la base de données

CREATE DATABASE glpi CHARACTER SET utf8 COLLATE

utf8_unicode_ci;

-- Création de l'utilisateur dédié

CREATE USER 'glpiuser'@'localhost' IDENTIFIED BY

'MotDePasse123!';

-- Attribution des droits uniquement sur la base glpi

GRANT ALL PRIVILEGES ON glpi.* TO 'glpiuser'@'localhost';

-- Rechargement des droits

FLUSH PRIVILEGES;

EXIT;
```

Grâce à cela l'utilisateur 'glpiuser' n'a accès qu'à la base 'glpi' et uniquement depuis localhost. Il n'a aucun droit sur les autres bases de données du serveur.

Vérification de la base

Vérifiez que la base et l'utilisateur ont bien été créés :

```
sudo mysql -u root -p
SHOW DATABASES;
-- Doit afficher 'glpi' dans la liste
SELECT user, host FROM mysql.user WHERE user = 'glpiuser';
-- Doit afficher glpiuser | localhost
exit;
```



Téléchargement et déploiement

GLPI 10.0.10 est téléchargé depuis le dépôt officiel GitHub, extrait et déplacé dans le répertoire web d'Apache. Les droits sont ensuite attribués à l'utilisateur www-data (utilisateur Apache) :

```
cd /tmp
wget https:/ /github.com/glpi-
project/glpi/releases/download/10.0.10/glpi-10.0.10.tgz

tar -xzf glpi-10.0.10.tgz

sudo mv glpi /var/www/glpi

# Donner les droits à Apache
sudo chown -R www-data:www-data /var/www/glpi
sudo chmod -R 755 /var/www/glpi
```

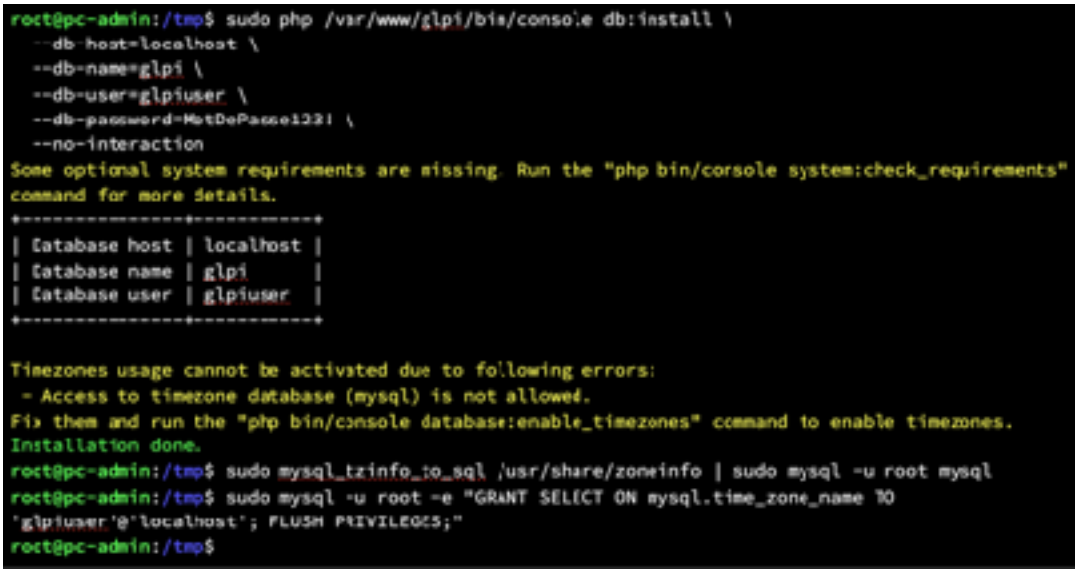
La structure du dossier GLPI après déploiement sera ainsi :

```
/var/www/glpi/
├── bin/ ← outils console (install, migration...)
├── config/ ← fichiers de configuration
├── files/ ← données uploadées, cache
├── public/ ← dossier web public (DocumentRoot Apache)
│   ├── index.php ← point d'entrée de l'application
│   └── lib/ ← librairies front-end
└── src/ ← code source PHP de GLPI
```

Installation via la console CLI

Contrairement aux versions antérieures, GLPI 10 ne fournit plus de page install.php accessible via navigateur. L'installation de la base de données se fera donc obligatoirement via la console PHP :

```
sudo php /var/www/glpi/bin/console db:install \
--db-host=localhost \
--db-name=glpi \
--db-user=glpiuser \
--db-password=MotDePasse123! \
--no-interaction
```



Une fois l'installation terminée, les timezones sont configurées afin d'éviter des erreurs dans GLPI :

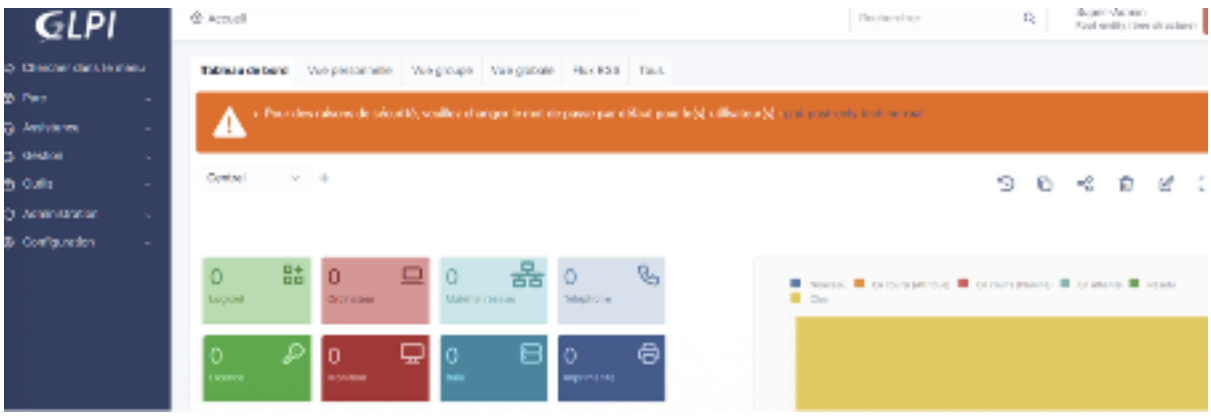
```
sudo mysql_tzinfo_to_sql /usr/share/zoneinfo | sudo mysql -u root mysql

sudo mysql -u root -e "GRANT SELECT ON mysql.time_zone_name TO
'glpiuser'@'localhost'; FLUSH PRIVILEGES;"
```

Première connexion à GLPI

Après l'installation, GLPI est accessible via le navigateur. Les identifiants par défaut seront modifiés immédiatement pour des raisons de sécurité :

Compte	Login	Mot de passe par défaut	Action
Administrateur	glpi	glpi	Changer immédiatement
Technicien	tech	tech	Changer ou désactiver
Utilisateur	normal	normal	Changer ou désactiver
Post-only	post-only	postonly	Changer ou désactiver

Connectez-vous ensuite à www.glpi.local :



Configuration Apache / Virtualhosts / DNS

VirtualHost pour le site vitrine

Exécutez « sudo nano /etc/apache2/sites-available/site.conf » :

```
<VirtualHost *:80>
  ServerName www.monsite.local
  DocumentRoot /var/www/site
<Directory /var/www/site>
AllowOverride All
Require all granted
</Directory>
ErrorLog ${APACHE_LOG_DIR}/site_error.log
CustomLog ${APACHE_LOG_DIR}/site_access.log combined
</VirtualHost>
```

Activation des deux VirtualHosts et du module rewrite, puis rechargement d'Apache

```
sudo a2ensite glpi.conf
sudo a2ensite site.conf
sudo a2enmod rewrite
sudo a2dissite 000-default.conf # désactiver le site par défaut
sudo systemctl reload apache2
```

Vérification de la configuration :

```
sudo apache2ctl -S
# Doit afficher les deux VirtualHosts actifs
```

```
port 80 namevhost glpi.local (/etc/apache2/sites-enabled/glpi.conf:1)
port 80 namevhost www.monsite.local (/etc/apache2/sites-enabled/site.conf:1)
```

Fichier .htaccess pour GLPI

GLPI 10 nécessite un fichier .htaccess dans son dossier public pour que la réécriture des URLs fonctionne correctement. Sans ce fichier, toutes les pages de GLPI retournent une erreur 404, accédez à « sudo nano /var/www/glpi/public/.htaccess » :

```
RewriteEngine ON
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ index.php [QSA,L]
```

Ce fichier indique à Apache de rediriger toutes les requêtes vers index.php, qui se charge ensuite de router vers la bonne page GLPI. C'est le pattern Front Controller, très répandu dans les frameworks PHP.

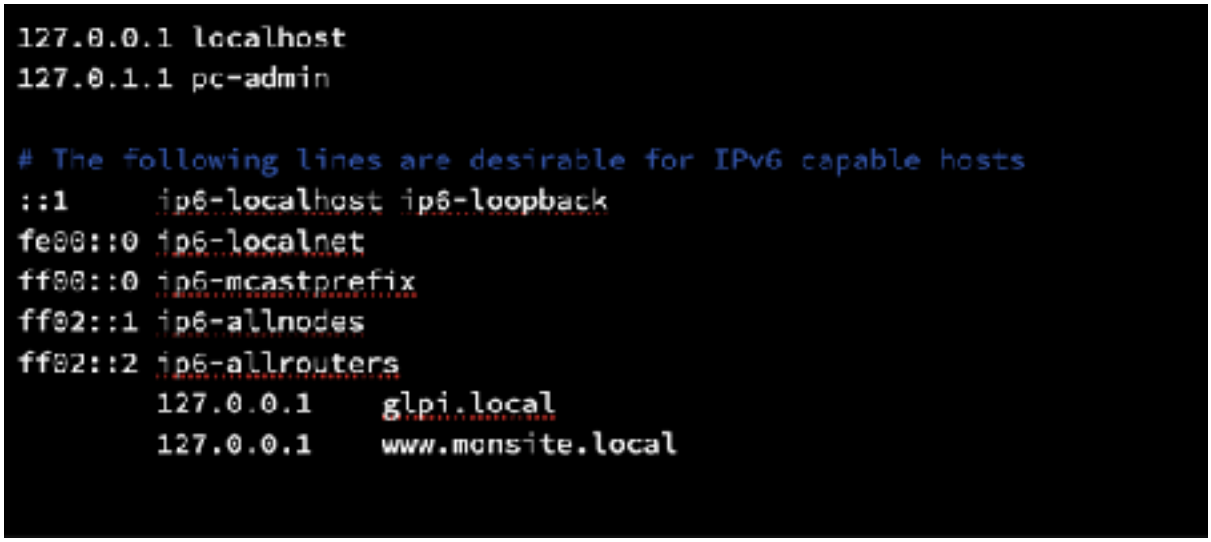
Configuration DNS local (fichiers hosts)

En l'absence d'un serveur DNS dédié, la résolution des noms est configurée manuellement dans les fichiers hosts des deux machines. Configurez donc sur le PC hôte Windows « RewriteRule ^(.*)$ index.php [QSA,L] ».

Ce fichier indique à Apache de rediriger toutes les requêtes vers index.php, qui se charge ensuite de router vers la bonne page GLPI. C'est le pattern Front Controller, très répandu dans les frameworks PHP.

Sur la VM Ubuntu, le fichier /etc/hosts de la VM est aussi modifié pour que les scripts PHP puissent résoudre glpi.local en interne (car la VM ne lit pas le fichier hosts Windows). Dans « sudo nano /etc/hosts » :

Ajouter ces lignes :
127.0.0.1 glpi.local
127.0.0.1 www.monsite.local



Création du site web et des pages HTML

Le site vitrine est hébergé dans « /var/www/site/ ». Il est composé de trois fichiers, dont la page d'accueil, la page de contact avec le formulaire et le script PHP qui traite le formulaire.

Page d'accueil / index.html

La page d'accueil présente le portail de signalement et propose un lien vers la page de contact :

sudo mkdir /var/www/site
sudo nano /var/www/site/index.html

Contenu de la page :

<!DOCTYPE html>
<html lang="fr">

```
<head>
   <meta charset="UTF-8">
   <title>Portail Support Informatique</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         max-width: 800px;
         margin: 60px auto;
         background: #f5f5f5;
      }
      .card {
         background: white;
         padding: 40px;
         border-radius: 8px;
         box-shadow: 0 2px 8px rgba(0,0,0,0.1);
      }
      h1 { color: #1F4E79; }
      .btn {
         display: inline-block;
         background: #2E75B6;
         color: white;
         padding: 12px 28px;
         text-decoration: none;
         border-radius: 6px;
         font-size: 16px;
      }
   </style>
</head>
<body>
   <div class="card">
      <h1>Portail Support Informatique</h1>
      <p>Bienvenue sur le portail de signalement des incidents.</p>
      <p>Utilisez le formulaire ci-dessous pour nous signaler
         tout problème informatique rencontré.</p>
      <a href="contact.html" class="btn">Signaler un problème</a>
   </div>
</body>
</html>
```

La page aura alors ce résultat :

Page de contact / contact.html

La page de contact contient le formulaire HTML. Les champs collectent les informations nécessaires pour créer un ticket GLPI pertinent. Le formulaire envoie les données en méthode POST vers le script PHP. La page est accessible via « sudo nano /var/www/site/contact.html » et voici son contenu :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <title>Signaler un problème</title>
    <style>
        body { font-family: Arial, sans-serif; max-width: 600px; margin: 60px auto; }
        .card { background: white; padding: 40px; border-radius: 8px;
            box-shadow: 0 2px 8px rgba(0,0,0,0.1); }
        h1 { color: #1F4E79; }
        label { display: block; margin-top: 15px; font-weight: bold;
            color: #333; }
        input, select, textarea {
            width: 100%; padding: 10px; margin-top: 5px;
            border: 1px solid #ccc; border-radius: 4px;
            box-sizing: border-box; font-size: 14px;
        }
        textarea { height: 120px; resize: vertical; }
        button { margin-top: 20px; background: #2E75B6; color: white;
            padding: 12px 30px; border: none; border-radius: 6px;
            font-size: 16px; cursor: pointer; width: 100%;
        }
        button:hover { background: #1F4E79; }
    </style>
</head>
<body>

<div class="card">
    <h1>Signaler un probleme</h1>
    <form action="submit_ticket.php" method="POST">

        <label>Votre nom :</label>
        <input type="text" name="nom" required
            placeholder="Ex: Jean Dupont">

        <label>Votre email :</label>
        <input type="email" name="email" required
            placeholder="Ex: jean.dupont@entreprise.fr">

        <label>Niveau d urgence :</label>
        <select name="urgence">
            <option value="2">Basse - Pas bloquant</option>
            <option value="3" selected>Moyenne - Genant</option>
            <option value="4">Haute - Bloquant partiellement</option>
            <option value="5">Tres haute - Completement bloque</option>
        </select>

        <label>Description du probleme :</label>
        <textarea name="description" required
            placeholder="Decrivez le probleme rencontre..."></textarea>

        <button type="submit">Envoyer le ticket</button>
    </form>
</div>
```

```
</body>
</html>
```

Voici le rendu final :

Configuration de l’API REST GLPI

L'API REST de GLPI permet à des applications externes (comme notre formulaire PHP) de créer des tickets, consulter le parc informatique, ou gérer des utilisateurs. Elle utilise le protocole HTTP avec des tokens d’authentification.

Activation de l'API REST

Dans l’interface GLPi, rendez-vous dans « Configuration », puis « Général » puis « Onglet API ». Activez alors « l’ API REST » (sélectionnez oui), puis activez la connexion avec les tokens externes (sélectionnez oui) puis cliquez sur « Sauvegarder ».

Création du client API et génération du token applicatif

Un client API représente l'application qui va utiliser l'API. Sur la même page, rendez-vous dans « Configuration » puis « API », cliquez sur « Ajouter un client API ».

Paramètre	Valeur
Nom	Monsite (nom du client)
Adresse IP début	Vide (pas de restriction)
Adresse IP fin	Vide (pas de restriction)
App-Token (Jeton d'application	Généré automatiquement (à noter)

Le App-Token identifie l'application cliente. Il est généré une seule fois et doit être conservé. Dans ce projet, il est intégré dans le script PHP submit_ticket.php.

Génération du token utilisateur

Le token utilisateur permet d'authentifier un compte GLPI spécifique via l'API. Il est généré dans « Administration » puis « Utilisateurs » sélectionnez ensuite l'utilisateur glpi puis l'onglet « Préférences », section « Token API » et cliquez sur « Régénérer ».

Le User-Token identifie l'utilisateur GLPI qui crée les tickets via l'API. Dans ce projet, c'est le compte administrateur glpi qui est utilisé.

Token	Rôle	Utilisation
App-Token	Identifie l'application cliente	Header HTTP : App-Token
User-Token	Identifie l'utilisateur GLPI	Header HTTP : Authorization: user_token

---

Fonctionnement du script

Le script submit_ticket.php est le cœur de l'intégration.
Lorsqu'un utilisateur soumet le formulaire, le script effectuera les opérations suivantes, dans un premier temps il y aura récupération et nettoyage des données du formulaire pour la sécurité. Ensuite l'ouverture d'une session API GLPI via les deux tokens aura lieu. Ensuite il y aura création du ticket via une requête HTTP POST vers l'API, puis fermeture de la session API. Alors l'affichage du résultat à l'utilisateur se fera (succès ou erreur).

Code source et explication

Le fichier « sudo nano /var/www/site/submit_ticket.php » est alors complété par :

```
<?php
// ===== CONFIGURATION API =====
define('GLPI_URL', 'http://glpi.local/apirest.php');
define('APP_TOKEN', 'votre_app_token_ici');
define('USER_TOKEN', 'votre_user_token_ici');

// ===== RÉCUPÉRATION DES DONNÉES DU FORMULAIRE =====
// htmlspecialchars() protège contre les injections XSS
$nom   = htmlspecialchars($_POST['nom'] ?? '');
$email  = htmlspecialchars($_POST['email'] ?? '');
$urgence  = intval($_POST['urgence'] ?? 3); // intval() pour sécuriser
$description = htmlspecialchars($_POST['description'] ?? '');

if (empty($nom) || empty($description)) {
    die('Erreur : champs obligatoires manquants.');
}

// ===== ÉTAPE 1 : OUVERTURE DE SESSION API GLPI =====
$ch = curl_init(GLPI_URL . '/initSession');
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        'Content-Type: application/json',
```

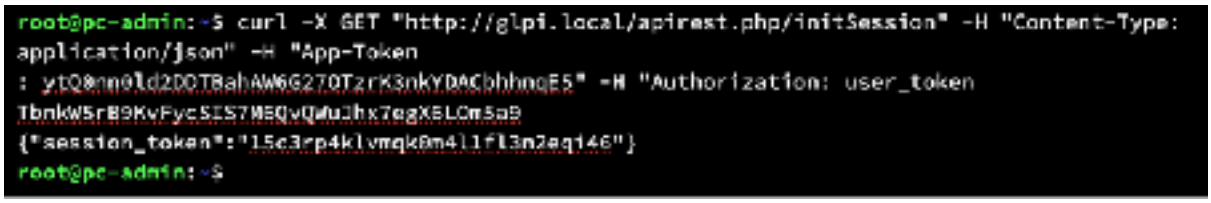
```
        'App-Token: ' . APP_TOKEN,
        'Authorization: user_token ' . USER_TOKEN,
    ],
]);
$session = json_decode(curl_exec($ch), true);
curl_close($ch);
$session_token = $session['session_token'];




// ===== ÉTAPE 2 : CRÉATION DU TICKET =====
$ticket = [
    'input' => [
        'name'  => "Incident signalé par $nom",
        'content' => "Nom : $nom\nEmail : $email\n\nDescription :\n$description",
        'type' => 1,    // 1 = Incident, 2 = Demande de service
        'status' => 1,  // 1 = Nouveau
        'urgency' => $urgence, // 1=très basse à 5=très haute
        'impact' => 3,  // 3 = moyen
        'priority' => $urgence, // priorité = urgence
    ],
];

$ch = curl_init(GLPI_URL . '/Ticket');
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_CUSTOMREQUEST => 'POST',
    CURLOPT_POSTFIELDS  => json_encode($ticket),
    CURLOPT_HTTPHEADER  => [
        'Content-Type: application/json',
        'App-Token: ' . APP_TOKEN,
        'Session-Token: ' . $session_token,
    ],
]);
$result = json_decode(curl_exec($ch), true);
curl_close($ch);

// ===== ÉTAPE 3 : FERMETURE DE SESSION =====
$ch = curl_init(GLPI_URL . '/killSession');
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
    CURLOPT_HTTPHEADER => [
        'App-Token: ' . APP_TOKEN,
        'Session-Token: ' . $session_token,
    ],
]);
curl_exec($ch);
curl_close($ch);

// ===== AFFICHAGE DU RÉSULTAT =====
if (isset($result['id'])) {
    echo '<h2>Ticket créé avec succès !</h2>';
    echo '<p>Numéro : <strong>#' . $result['id'] . '</strong></p>';
    echo '<p>Vous serez contacté à : ' . $email . '</p>';
    echo '<a href="index.html">Retour à l\'accueil</a>';
} else {
    echo '<h2>Erreur lors de la création du ticket</h2>';
    echo '<pre>' . print_r($result, true) . '</pre>';
}
?>
```

Test de l'API en ligne de commande

Avant d'intégrer les tokens dans le script PHP, l'API a été testée directement en ligne de commande avec « curl » pour vérifier le bon fonctionnement de l'authentification :

```
curl -X GET "http://glpi.local/apirest.php/initSession" \
-H "Content-Type: application/json" \
-H "App-Token: votre_app_token" \
-H "Authorization: user_token votre_user_token"
```

Une réponse valide retourne un session_token JSON, confirmant que l'API fonctionne et que les tokens sont corrects :

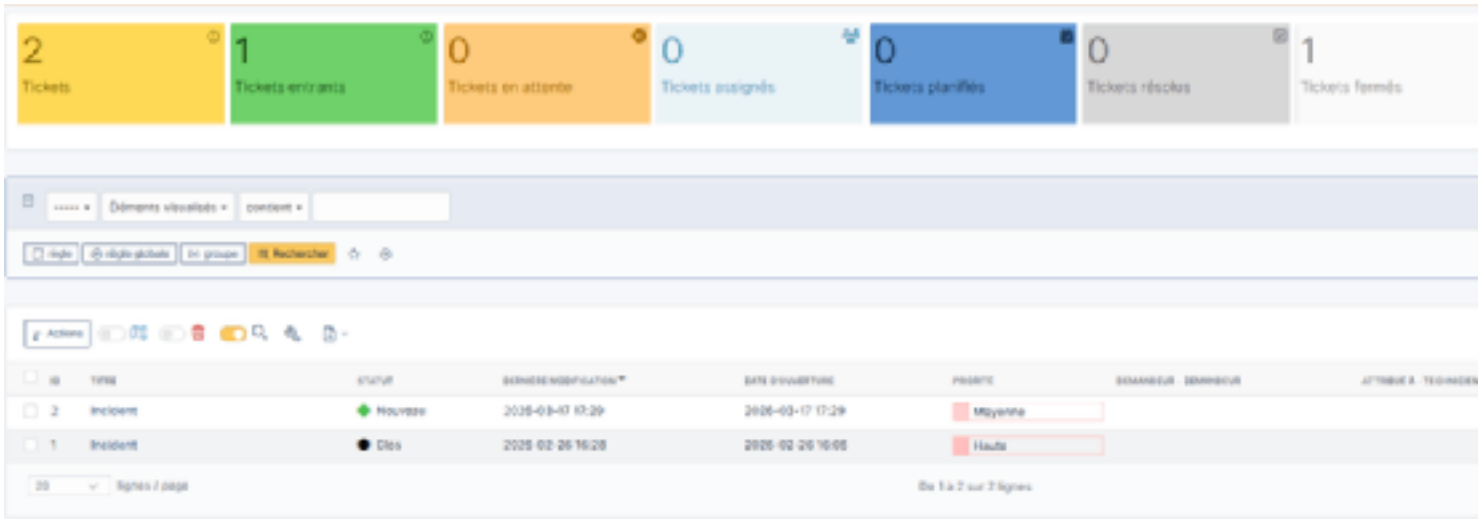


Scénario de test complet

Le test de bout en bout consiste à simuler le parcours complet d'un utilisateur, depuis le formulaire web jusqu'à l'apparition du ticket dans GLPI. Il faudra dans un premier temps accéder au site depuis un navigateur via « http://www.monsite.local, afin que la page d'accueil du portail s'affiche. Ensuite il faudra cliquer sur « Signaler un problème » sur la page et alors le formulaire de contact s'affichera. Il faudra ensuite remplir et envoyer le formulaire. Puis une fois la connexion à GLPI faite via « http://glpi.local » lors de la connexion à GLPI le ticket apparaîtra alors dans la liste.

Vérification du ticket dans GLPI

Dans l'interface GLPI, sous « Assistance » puis « Tickets », le ticket créé par le formulaire est visible avec toutes les informations saisies, comme le nom de l'utilisateur, le mail, le niveau d'urgence ainsi que la description du problème.



DIFFICULTÉS RENCONTRÉES

Au cours de ce projet, j'ai été confronté à plusieurs problèmes techniques qui ont nécessité une analyse approfondie avant de trouver la bonne solution.

Fichier install.php absent dans GLPI 10

Lorsque j'ai tenté d'accéder à « http://glpi.local/install/install.php » pour lancer l'installation, je suis tombé sur une erreur 404. J'ai d'abord pensé à un problème de configuration Apache, mais en réalité GLPI 10 a tout simplement supprimé l'installateur web au profit d'une installation en ligne de commande. Il a fallu passer par la console PHP avec php bin/console db:install. Ce problème m'a appris qu'il faut systématiquement consulter les notes de version lors d'un changement de version majeure, car les méthodes d'installation peuvent changer du tout au tout.

Erreur 404 sur toutes les pages GLPI

Même après l'installation réussie, toutes les pages de GLPI renvoyaient une erreur « 404 Not Found ». Le problème venait du fichier .htaccess qui était absent du dossier /var/www/glpi/public. GLPI 10 s'appuie sur un pattern Front Controller où toutes les requêtes doivent transiter par « index.php », et sans le .htaccess contenant les règles RewriteEngine, Apache ne savait pas rediriger correctement les URLs. La création manuelle de ce fichier avec les bonnes règles de réécriture a résolu le problème.

Erreur ERROR_WRONG_APP_TOKEN_PARAMETER

Lors des premiers tests de l'API REST, celle-ci renvoyait systématiquement l'erreur ERROR_WRONG_APP_TOKEN_PARAMETER. Après investigation, j'ai découvert que le client API dans GLPI avait été configuré avec une restriction d'IP limitée à 192.168.85.128 uniquement. Or, le script PHP s'exécutait directement sur la VM elle-même et n'utilisait donc pas cette adresse en source. La solution a consisté à supprimer la restriction d'IP dans les paramètres du client API GLPI en laissant le champ vide, ce qui autorise les appels depuis n'importe quelle origine.

Conclusion

Ce projet m'a permis de déployer de A à Z une infrastructure complète et fonctionnelle, en partant de la création de la machine virtuelle jusqu'à la validation du flux de création automatique de tickets. Chaque étape a nécessité de comprendre le rôle de chaque composant et leurs interactions. Les principaux apprentissages de ce projet sont :

- La configuration d'Apache avec plusieurs VirtualHosts pour héberger plusieurs sites sur un serveur
- L'utilisation d'une API REST pour faire communiquer deux applications (formulaire web et GLPI)
- L'importance du fichier /etc/hosts pour la résolution DNS dans un environnement sans serveur DNS
- La différence entre les versions de GLPI et la nécessité de consulter la documentation officielle
- La résolution méthodique de problèmes techniques à partir des logs et des tests progressifs

Résultat final

L'objectif est pleinement atteint car un utilisateur qui accède à http://www.monsite.local, peut remplir le formulaire puis son incident est automatiquement enregistré dans GLPI sous forme d’un ticket, sans aucune action manuelle de la part de l'équipe informatique.

Les différentes améliorations possibles

- Mise en place d'un certificat SSL/TLS (HTTPS) avec Let's Encrypt pour chiffrer les communications
- - Déploiement d'un vrai serveur DNS (Bind9) pour remplacer les fichiers hosts
- - Configuration d'un serveur DHCP dédié (isc-dhcp-server)
- - Envoi d'un email de confirmation automatique lors de la création d'un ticket

SOURCES / DOCUMENTATION :

Ubuntu : https://help.ubuntu.com/

GLPI : https://help.glpi-project.org/documentation/fr

Bind9 : https://doc.ubuntu-fr.org/bind9

API REST : https://siocours.lycees.nouvelle-aquitaine.pro/doku.php/si7/configuration/glpi_gestionapirest