B3.2 -TD17-Continuité

# **REPRISE PRÉCÉDENTE**

## **Manipulations**

## 

* **Il a fallu changer le paramètre BIND\_ADRESS de 127.0.0.1 à 0.0.0.0 : expliquer l’intérêt.**

L’intérêt de changer le 127.0.0.1 en 0.0.0.0 permet la connexion sur toutes les cartes de serveur de base de données.

* **quelle adresse doit-on mettre dans le GRANT … TO <utilisateur>@<adresse>…**

L’adresse que nous devons mettre est celle du serveur WEB

* **Quelle adresse doit-on mettre dans la commande mysqli\_connect (<adresse Serveur>, ….)**

as le serveur de BDD car le serveur WEB communique avec les utilisateurs donc avec l’extérieur.

* **Séparer les parties Web et BDD permet de démultiplier les serveurs Web au contact des utilisateurs. Quelle sera la difficulté en termes de cohérence des données si on a d’innombrables serveurs de bases de données ?**

S' il y a d'innombrables serveurs de base de données, les données recueillies ne seront pas exploitables car il n’y aura pas de cohérence entre les tables.

* **Si le serveur BDD tombe en panne, plus aucun usage du site n’est possible. Imaginer comment on pourra rendre la BDD résistante aux pannes.**

L’utilisation d’un serveur BDD miroir qui fonctionne en simultané.

* **Expliquer pourquoi les utilisateurs pourront consulter les mêmes informations (produits, commandes, etc) quel que soit le serveur sur lequel ils tomberont**

Les utilisateurs pourront consulter les mêmes informations quel que soit le serveur car les serveurs travaillent en simultanés et miroir. C’est pour cela

* **Expliquer pourquoi il est nécessaire que toutes ces adresses soient associées au même FQDN (fnac.com)**

Il est nécessaire que toutes les adresses soient associées au même FQDN pour quand un utilisateur arrive sur le site, qu’il soit associé à un serveur aléatoirement afin qu’il ne soit pas surchargé.

* **Expliquer l’intérêt d’un point de vue performance.**

L’intérêt d’un point de vue performance est que cela permet d’éviter au serveur d'être saturé de requête donc d’avoir une réponse des serveurs toujours rapides.

# **CONTINUITÉ**

## **Compréhension**

1. **Expliquer pourquoi la solution de répartition du TD16 n’assure pas de *continuité***La solution de répartition du TD16 n’assure pas la continuité car on ne l’avait fait que sur un seul serveur.

1. **Expliquer pourquoi la sauvegarde reste indispensable malgré la mise en place de techniques de tolérance de panne**  
   Au cas ou destruction des serveurs ils faut pouvoir récupérer les données sauvegarder .

1. **Le principe de *failover* est de la tolérance dite « *active/passive* » : expliquer**  
    Failover permet la tolérance au panne, en effet un serveur marche en continue pendant que le deuxième est inactif, il sera actif seulement si le premier tombe en panne
2. **Le *load balancing* est une tolérance dite *« active/active* » : expliquer pourquoi.**  
   Le load balancing est une tolérance dite “active/active” car les serveurs fonctionnent tous les deux en même temps.

# **LOAD BALANCING DU SERVEUR WEB**

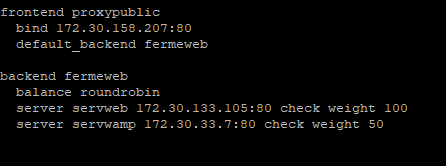
## **Contexte**

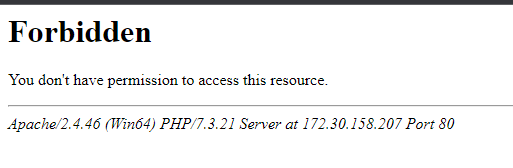
## mot de passe remotusr : mpremot

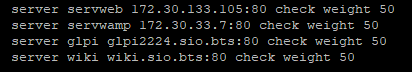
## **Optimisation**

1. 







1. 
2. Cela fonctionne bien si on enlève le wiki.

## **Analyse**

1. **Expliquer en quoi cette solution améliore grandement la performance d’une application web**

Cela améliore grandement la performance d’une application web car cela évite de surcharger un serveur et de répartir les requêtes entre les serveur

1. **Expliquer pourquoi il est nécessaire d’ajouter l’option check pour assurer de la continuité de service (pourquoi sans cela, le système n’assure pas la continuité?)**

Il est nécessaire de rajouter l’option check pour assurer la continuité du service car grâce à la commande check, le proxy va vérifier l’état du serveur avant l’envoi de la requête cela permet de ne pas afficher une page d’un serveur qui ne fonctionne pas mais seulement celle d’un serveur en état de marche

1. **Rappeler l’intérêt d’utiliser le poids lors de la répartition**

Des serveurs sont plus puissants que d’autre donc il est important de bien équilibrer la réception de requêtes, un serveur plus puissant pourra recevoir plus de requêtes

1. **Expliquer les manipulations qui seront à entreprendre sur le serveur BDD pour qu’il autorise les connexions à la base depuis les 4 serveurs**  
   Les manipulations qui seront à entreprendre sur le serveur BDD pour qu’il autorise les connexions à la base depuis les 4 serveurs sont :   
   - Faire un GRANT avec l’ip des différents serveurs qui communique avec le serveur BDD  
   - Rajoutez la commande mysql\_connect pour que les pages communiquent avec la base de données.

## 

# **Failover**

### **Mise en cluster**

### 

* **expliquer le principe Actif/passif**

Le principe d’Actif/Passif :   
Prenons l’exemple de deux serveurs WEB, le serveur 1 fonctionne et le serveur 2 lui n’est pas actif. Si le serveur 1 tombe en panne alors le serveur 2 va s’activer pour prendre le relais.

* ***indiquer à quel type de disponibilité (répartition de charge ou load balancing / tolérance de panne ou failover) cela répond***

failover

## 