REPUBLIQUE DU SENEGAL

Un peuple-un but-une foi

Ministère de l'Enseignement Supérieur et de la Recherche
****

Université Catholique de l'Afrique de l'Ouest

Institut Supérieur de Gestion Saint Michel

MEMOIRE DE FIN DE FORMATION
EN VUE DE L'OBTENTION DU MASTER
EN INFORMATIQUE DE GESTION

Sujet :

ETUDE ET IMPLEMENTATION D'UNE REPLICATION SYMETRIQUE SYNCHRONE DE BASES DE DONNEES DANS DES ENVIRONEMENTS PHYSIQUE ET VIRTUEL.

Présenté et soutenu par :

David Olivier Nkogo Ondo

Sous la direction de :

M Edouard Ngor SARR

Enseignant-chercheur à UCAO

ED2DS Université de Thiès

Année académique 2015-2016

SOMMAIRE

DEDICACE

Je dédie ce travail :

A mon père :

Mr Ondo Nkogo Marcelin en signe de reconnaissance de l'immense bien que vous m'avez fait concernant mon éducation qui aboutit aujourd'hui à la réalisation de cette étude. Recevez à travers ce travail, toute ma gratitude et mes profonds sentiments. Que Dieu le tout puissant soit à vos côtés et vous accorde une meilleure santé (amen).

A ma mère :

Mme Ondo Nathalie pour m'avoir donnée la vie et la joie de vivre. Ta bonne éducation, tes conseils et tes bénédiction n'ont jamais fait défaut, que Dieu le tout puissant soit à tes côtés (amen).

REMERCIEMENTS

La réalisation de ce projet a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

Je voudrais tout d'abord adresser toute ma gratitude à monsieur Edouard Ngor Sarr pour son encadrement, sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion dans la rédaction de ce mémoire.

Je remercie le directeur général de l'université catholique de l'Afrique de l'ouest le Dr Jean Marie SENE pour sa disponibilité, sa rigueur scientifique et son sens d'écoute et d'échange.

Je remercie M Remy BASSE, chef du département informatique à l'Université Catholique de l'Afrique de l'Ouest. Sans qui rien n'aurai été possible. Son ouverture d'esprit et sa justesse de jugement m'ont permis d'enrichir mes connaissances dans le domaine informatique. Merci pour tout professeur

Dans le même ordre d'idée, je remercie très vivement tout le corps professoral de l'université catholique de l'Afrique de l'ouest qui nous a fait bénéficier d'une formation pluridisciplinaire de très haut niveau et très adaptée aux réalités du domaine informatique.

Je voudrais exprimer ma reconnaissance envers les amis et collègues qui m'ont apporté leur support moral et intellectuel tout au long de ma démarche. Un grand merci à Danielle pour la mise en forme de mon rapport, ceci a grandement facilité mon travail. Enfin, je tiens à témoigner toute ma gratitude à Billy, pour sa confiance et son support inestimables.

ABREVIATIONS OU SIGLES

- SGBD : Système de gestion de base de données
- RAM : Mémoire à accès aléatoire (Random Access Memory)
- NOSQL : Not only SQL
- OLAP : OnLine Analytiquecal Processing
- OLTP : On Line Transaction Processing
- RWA : Read One Write All
- RWA-A : Read OneWrite All Available
- V2P : Validation a deux Phases
- BSD : Berkley Software Distribution
- IO : Input/Output
- ACID : Atomicité Cohérence Isolation Durabilité
- NAS : Network Attached Storage
- DRDB : Distributed Replicated Block Device
- WAL : Wats Acces Line

RESUME

Lors de l'utilisation d'une application de base de données, les problèmes majeurs rencontrés par cette dernière sont la disponibilité des données, le temps de réponse et la résistance de l'application à d'éventuelles pannes. Et l'une des solutions à ses problématiques est la réplication des bases de données. La réplication met en jeu au moins deux SGBD et consiste en un processus de propagation d'éventuelles modifications réalisées sur la base de données et se déroule généralement en trois étapes : La détection des modifications dans la base de données, leur stockage dans une table, et leur propagation dans les différentes bases esclaves. La réplication est là pour améliorer les performances, l'équilibrage de charge, avoir un meilleur temps de réponse, augmenter la disponibilité des données et améliorer la tolérance aux pannes. La mise en œuvre de la réplication des bases de données n'est pas une mince affaire c'est pour ça qu'il ne faut l'appliquer que dans des scénarios ou elle sera vraiment utile comme : le partage des données à travers des postes distants connectées via un réseau étendu (WAN), l'amélioration de l'accessibilité aux données du serveur et la sauvegarde des données. Comme points forts de la réplication nous pouvons citer une performance améliorée et une probabilité de panne plus faible. Au niveau des points faibles nous avons la gestion des mises à jour surcoût : échange de messages inter-sites. A cet effet plusieurs solutions de réplication s'offre à nous.

- **La réplication mono maître / multi maître :** Le maître peut exécuter des requêtes de type lecture ou écriture, l'esclave ne peut exécuter que des requêtes de type lecture et des requêtes de réplication. On parle alors de mono maître pour désigner un système à un seul maître et réplication multi maître pour des systèmes avec plus d'un maître.
- **La réplication symétrique / asymétrique :** Selon le sens de propagation de la réplication, on parle de réplication bidirectionnelle ou symétrique si la réplication se passe dans le sens de l'esclave vers le maître et inversement. La réplication est unidirectionnelle si la réplication se passe seulement dans le sens du maître vers l'esclave.
- **La réplication synchrone, aussi appelée réplication en temps réel :** La synchronisation est effectuée en temps réel puisque chaque requête est déployée sur l'ensemble des bases de données avant sa validation sur le serveur ou la requête est exécutée. Ce type de réplication assure un haut degré des données mais requiert une disponibilité permanente des serveurs et de la bande passante, et nécessite de gérer des transactions multi-sites coûteuses en ressources.
- **La réplication synchrone asymétrique :** Elle utilise un site primaire qui propage les mises à jour en temps réel vers un ou plusieurs sites secondaires, la table répliquée est immédiatement mise à jour pour chaque modification par utilisation de trigger sur la table maître.

- **La réplication synchrone symétrique :** Lors de la réplication synchrone symétrique il n'y a pas de table maîtresse. L'utilisation de trigger sur chaque table doit différencier une mise à jour client à répercuter d'une mise à jour par réplication.
- **La mise à jour asynchrone :** La réplication asynchrone stocke les opérations intervenues sur une base de données dans une queue locale pour les propager plus tard à l'aide d'un processus de synchronisation. Ce type de réplication est plus flexible que la réplication synchrone.
- **La réplication asynchrone symétrique :** Elle propage les mises à jour en temps différé via une file persistante. Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, l'horloge par exemple.
- **La réplication asynchrone symétrique :** Dans ce cas, la mise à jour des tables répliquées est différée, cette technique risque de provoquer des incohérences de données.

La réplication des bases de données est un concept très intéressant permettant de régler beaucoup de problèmes dans le domaine du stockage des données. Son utilité est indiscutable et nécessaire à beaucoup de système.

Mots clés : Base de données, réplication, disponibilité.

ABSTRACT

When using a database application, the main problems encountered by the application are: the data's availability, the response time and the ability to withstand possible failures.

One of the solution to these problems is the data's replication. Replication involves at least two DBMSs (SGBD) and consists of a process of propagating any changes made to the database and generally takes place in three steps: The detection of changes in the database, their storage in a table, and their propagation in the different slave bases. Replication is there to improve performance, load balancing, to allow a better response time, increase the data's availability and improve failure's tolerance. The implementation of database replication is not a small matter, which is why it should be applied only in scenarios where it will be really useful as: the data sharing through remote workstations connected over a wide area network (WAN), the improvement of accessibility to server's datas and the backing-up of datas. As great advantages of the replication, we have an improved performance, a lower probability of failures. As weak sides, we have the management of the updates' costs: messages exchange between sites. Thus, several replication solutions are available to us.

- **Mono master/multi master Replication:** The master can execute read or write requests, while the slave can only execute read requests and replication requests. This is called a mono master to designate a single-master system and multi-master replication for systems with more than one master.

- **Symmetric / asymmetric replication:** Depending on the propagation's direction of the replication, we speak of bidirectional or symmetric replication if the replication occurs in the direction of the slave to the master and inversely. Replication is unidirectional if it occurs only in the direction of the master to the slave.

- **Synchronous update, also known as real-time replication:** Synchronization is carried out in real time since each request is deployed on all the databases before its validation on the server where the request is executed. This type of replication assures a high degree of data but requires a permanent availability of the servers and bandwidth, and also requires managing the multi-site transactions which are very costly.

- **Synchronous asymmetric replication:** It uses a primary site that propagates real-time updates to one or more secondary sites, the replicated table is immediately updated for each change by using triggers on the master table.

- **Symmetric synchronous replication:** During symmetric synchronous replication there is no master table. The use of triggers on each table should differentiate a client update to be reflected from an update by replication.

- **The asynchronous update:** Asynchronous replication stores operations occurred on a database in a local queue to propagate them later using a synchronization process. This type of replication is more flexible than synchronous replication.

- **Symmetric asynchronous replication:** It propagates time-delayed updates via a persistent queue. The updates will be executed later, from an external trigger, the clock for example.

- **Symmetric asynchronous replication:** In this case, the replicated tables ‘update is delayed. This technique may cause data inconsistencies.

Replication of databases is a very interesting concept allowing to solve many problems in the field of data storage. Its utility is indisputable and necessary for many systems.

Keywords: database, replication, availability.

LISTE DES FIGURES

LISTE DES IMAGES

LISTE DES TABLEAUX

INTRODUCTION GENERALE

INTRODUCTION GENERALE

Mes études en Master 2 Informatique de Gestion cette année 2015-2016 à UCAO Saint Michel ont été l'opportunité pour moi d'appréhender plusieurs concepts informatiques allant des réseaux, à la programmation web en passant par la gestion des bases de données avec oracle et MySQL. La réplication synchrone symétrique est aussi appelée réplication en temps réel car la synchronisation est effectuée en temps réel. Chaque requête est déployée sur l'ensemble des bases de données avant la validation de la requête sur le serveur où la requête est exécutée. Tout ce mécanisme s'effectue en continue d'une base de donnée à une autre, de serveurs distants entre entreprises. Chaque base est maitre il n'y a pas de base esclave, toutes travaillent à un même niveau de réplication de données en recevant les modifications et les transactions à la chaine de base en base, de serveur à serveur.

Dans une application avec base de données, nous avons le modèle client-serveur. Le client se connecte à un gestionnaire de base de données (SGBD) pour envoyer des demandes. Ces demandes sont de deux natures différentes ; lecture (de requêtes), mise à jour (transaction). Au niveau des transactions nous avons des modifications de données sur le serveur, mais cela reste des demandes de courte durée. Mais dans le cas d'une lecture nous n'avons pas de modifications de données mais les traitements peuvent être longs et porter sur une grande masse de données. Nous pouvons donc dans ce cas faire un constat : dans le cas d'un site ou application web un nombre important de données ou de requêtes peut obstruer partiellement ou complètement le serveur. À cet effet il existe plusieurs solutions pour contrer ce problème. D'où la réplication des données.

L'objectif principal de la réplication est de faciliter l'accès aux données en augmentant la disponibilité. Soit parce que les données sont copiées sur différents sites permettant de répartir des requêtes, soit parce qu'un site peut prendre la relève lorsque le serveur principal s'écroule. A cet effet, l'objet de mon travail consistera à faire l'étude et l'implémentation d'une réplication symétrique et synchrone dans un environnement physique et virtuel : sous le SGBD MySQL.

Grace à une réplication symétrique et synchrone certaines entreprises qui ne sont pas très avancées dans le domaine informatique se verront très ravies d'avoir une gestion de leurs données au sein de leurs entreprises permettant de mieux gérer l'accès aux données. Mon étude fut réalisé avec pour support les cours et exercices vus et pratiqués en classe. Elle s'est étendue aussi sur des recherches effectuées sur le Net.

Pour mettre la réplication en œuvre, nous allons utiliser comme SGBD, MySQL-5.6.17 avec xampp 5.6 et comme support de test nous aurons deux ordinateurs physiques (maitres) pour montrer comment la réplication se passe en temps réel sur les différentes machines en interconnexion. Nous utiliserons Virtualbox comme émulateur c'est-à-dire pour créer nos machines virtuelles sous Debian 8 et Windows 8.1.

En vue de rendre compte de manière fidèle de l'objet de mon étude, aux cours de mon analyse je présenterai une partie théorique qui portera sur l'explication de différents concepts et processus utilisés et dans la deuxième partie une analyse pratique pour montrer comment ça marche.

Première partie :

Etude détaillée sur la réplication des Données

Chapitre 1 : Objectifs de la réplication

Section 1. Définition des différents concepts

1. La réplication de données

Dans une base de données distribuée, la réplication de données elle, permet d'augmenter la fiabilité et la disponibilité des données ainsi que les performances d'accès au fichier. En général, l'objet de réplication est une table relationnelle, un document ou un fichier. La réplication consiste alors à placer plusieurs copies de l'objet sur différents nœuds ou sites du réseau. Cela fournit une grande disponibilité des données. Si un site devient non opérationnel à la suite d'une panne par exemple, une autre copie est toujours accessible sur un autre site. La réplication permet aussi d'améliorer les performances d'accès en augmentant la localité de référence. Lorsque le coût de communication est un facteur dominant, le placement d'une copie sur le site ou elle est le plus souvent sollicitée favorise les accès locaux en évitant les accès réseaux.

La réplication repose aussi sur un ensemble de technologies qui permet de copier et de distribuer des données et des objets de base de données d'une base de données vers une autre puis de synchroniser ces bases de données afin de préserver leurs cohérences. Avec la réplication, vous pouvez distribuer des données en différents emplacements et à des utilisateurs distants ou mobiles sur des réseaux locaux et étendus, des connexions d'accès à distance, des connexions sans fil, et internet.

Les avantages apportés par la réplication peuvent être comparés avec la complexité et les coûts supplémentaires de maintenance des copies qui doivent en théorie rester identiques à tout moment. La mise à jour d'une copie doit être répercutée automatiquement sur toutes ses répliques. Le problème est amplifié par la présence de pannes de nœuds (ou de sites) ou réseaux. Le compromis recherché entre performance d'accès en consultation et en mise à jour des données rend difficile le choix du niveau de réplication. Celui-ci est très dépendant de la charge de travail demandée par les applications. Le problème de la réplication de données reste donc un vaste thème de recherche et les solutions doivent être adaptées au contexte afin d'offrir un bon compromis entre des objectifs conflictuels tels que disponibilité, cohérence, performance.

1.1. Site primaire (ou maître ou référence)

Sur un site primaire, ou maitre nous avons des accès en lecture et écriture au niveau application. Les transactions de mise à jour sont de plusieurs ordres notamment :

SELECT, UPDATE, INSERT, DELETE.

1.2. Site secondaire (ou esclave ou cible)

Au niveau d'un site secondaire, ou esclave, les accès au niveau application se font en lecture seule. Les requêtes sont sous plusieurs ordres de select. Le SGBD ou le réplicateur gère la mise à jour.

1.3. La propagation.

La propagation des données répercute la mise à jour d'une donnée de référence, sur une donnée cible, du serveur vers le client, l'envoi des modifications apportées aux données des clients vers le serveur et éventuellement le suivi des modifications et des erreurs.

2. Propriété de la réplication

2.1. Ecriture en mémoire Ram et fichier journal (log), et réplication asynchrone.

Deux techniques sont utilisées pour limiter le temps d'attente, toutes deux affectant un peu la sécurité des opérations : l'écriture en mémoire RAM et la réplication asynchrone. La première technique est très classique et elle est utilisée par tous les SGBD du monde. Au lieu d'effectuer des écritures répétées sur le disque sans ordre prédéfini (accès dits « aléatoires ») qui imposent à chaque fois un déplacement de la tête de lecture et donc une latence de quelques millisecondes, on écrit séquentiellement dans un fichier de journalisation (log) et on place également la donnée en mémoire RAM.

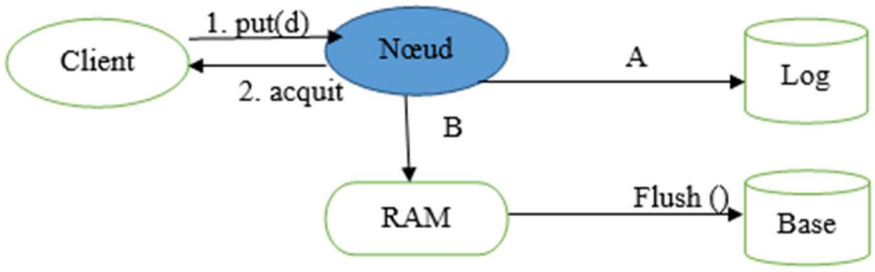


Figure *1*: Procédé d'écriture en RAM [w16]

A terme, le contenu de la mémoire RAM, marqué comme contenant des données modifiées, sera écrit sur le disque dans les fichiers de la base de données (opération de flush()). La séquence est utilisée dans la figure ci-dessus. Cela permet de grouper les opérations d'écritures et donc de revenir à des entrées et des sorties séquentielles sur le disque, aussi bien dans le fichier journal que dans la base principale. En cas de panne avant l'opération de flush (), les données modifiées n'ont pas été écrites dans la base, mais le journal (log) est quant à lui préservé. La reprise sur panne consiste à ré-effectuer les opérations enregistré dans le log.

Autre scénario d'une réplication avec deux copies. Quand le client reçoit finalement l'acquittement, il peut être sûr que trois copies ce sont effectivement enregistrées de manière durable dans le système. Cela nécessite une chaîne comprenant 8 messages, tout obstacle le long du chemin (un serveur temporairement surchargé par exemple) risquant d'allonger considérablement le temps d'attente. Nous parlons dans ce cas d'une réplication avec écritures synchrones.

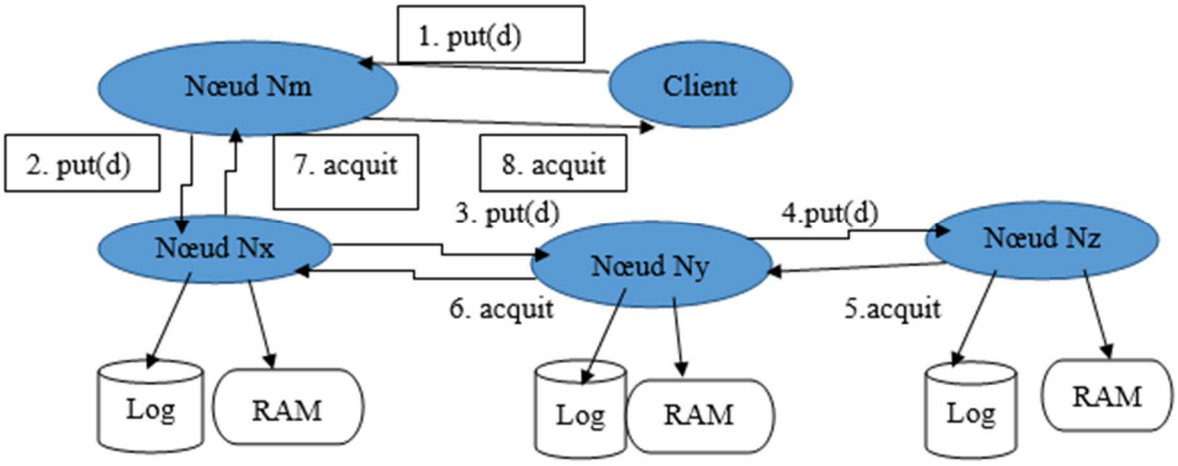


Figure *2*: Réplication avec écritures synchrones [w16]

La seconde technique pour limiter le temps d'écriture est le recours à des écritures asynchrones. Contrairement au scénario de la figure ci-dessus, le serveur Nx qui reçoit la requête va effectuer l'écriture et envoyer immédiatement l'acquittement au client, lui rendant ainsi la main et permettant la poursuite de son exécution. Après l'acquittement, Nx commence l'envoi des messages pour la réplication, là encore en mode asynchrone : figure ci-dessous

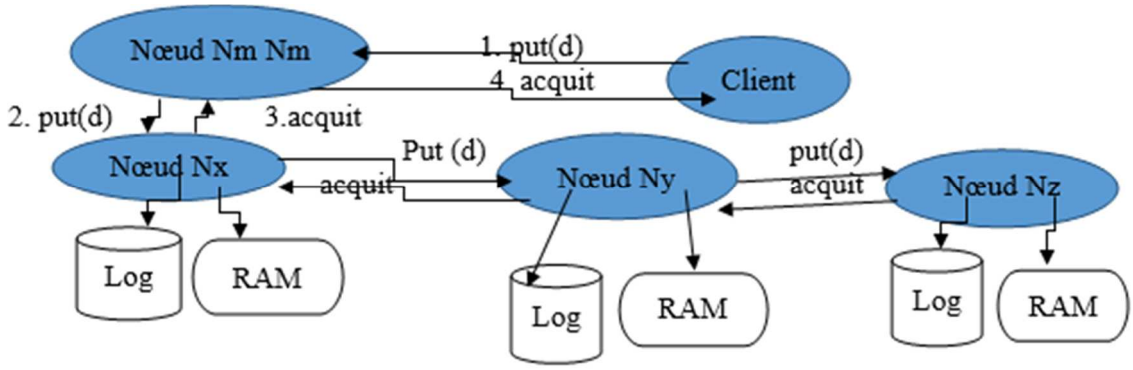


Figure *3*: réplication avec écritures asynchrones [w3]

Dans ce scénario, beaucoup plus rapide pour le client, deux phénomènes apparaissent :

- Le client reçoit un acquittement alors que la réplication n'est pas complète ; il n'y a donc pas à ce stade de garantie complète de sécurité.
- Le client poursuit son exécution alors que toutes les copies ne sont pas encore mises à jour ; il se peut alors qu'une lecture renvoie une des versions antérieures**.**

Il y a donc un risque pour l'incohérence des données. C'est un problème sérieux, caractéristique des systèmes en général, du NoSQL en particulier.

2.2. Cohérence des données.

La cohérence est la capacité d'un système de gestion de données à refléter fidèlement les opérations d'une application. Un système est cohérent si toute opération (validée) est

immédiatement visible et permanente. Si je fais une écriture suivie d'une lecture, je dois constater les modifications effectuées ; si je refais une lecture en peu plus tard, ces modifications doivent toujours être présentes. La cohérence dans les systèmes répartis dépend de deux facteurs : la topologie du système (maître-esclave ou multi-nœud) et le caractère asynchrone ou non des écritures. Nous avons trois combinaisons qui sont possibles en pratique : illustration ci-dessous

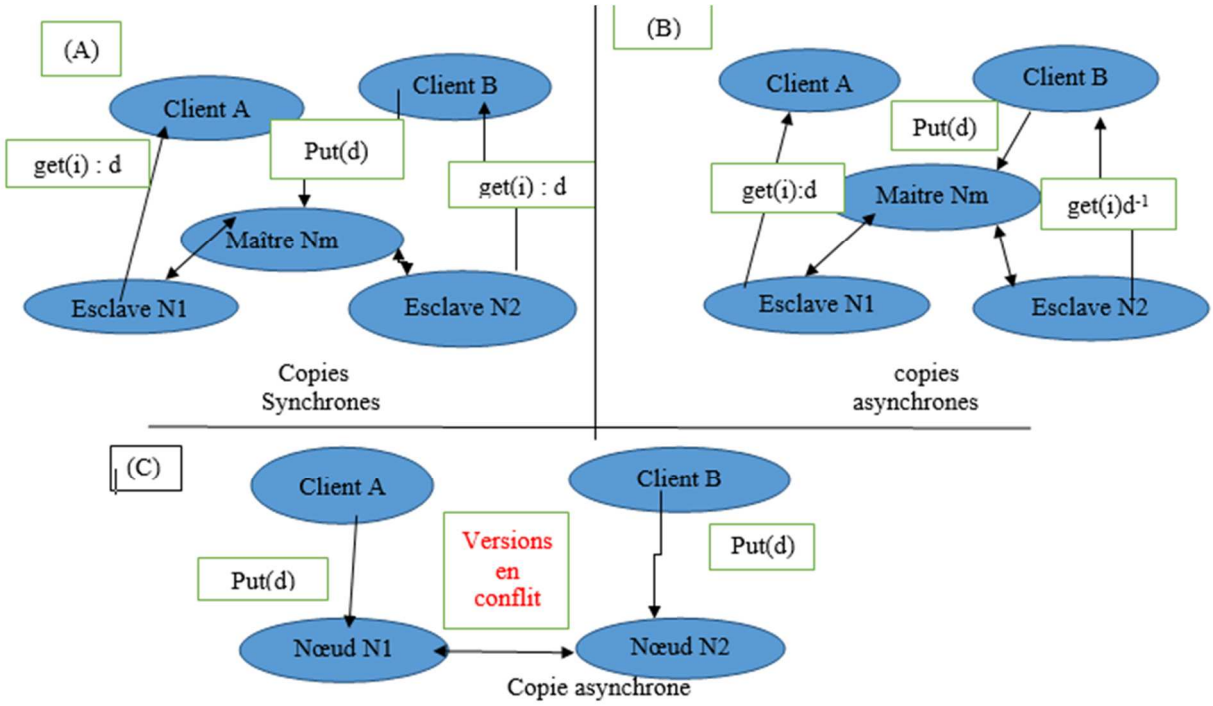


Figure 4: Topologie maître-esclave ou multi-nœud [w16]

Premier cas (A en haut à gauche) : la topologie est de type maître-esclave, et les écritures synchrones. Toutes les écritures se font par une requête adressée au nœud maître qui se charge de les distribuer aux nœuds esclave. L'acquittement n'est envoyé au client que quand toutes les copies sont en phase. Ce cas assure la cohérence forte, car toute lecture du document, quel que soit le nœud sur lequel elle est effectuée, renvoie la même version, celle qui vient d'être mise à jour. Cette cohérence se fait au prix de l'attente que la synchronisation soit complète, et ce à chaque écriture. Dans le second cas (B), la topologie est toujours de type maître esclave, mais les écritures sont asynchrones. La cohérence n'est plus forte : il est possible d'écrire en s'adressant au nœud maître, et de lire sur un nœud esclave. Si la lecture s'effectue avant la synchronisation, il est possible que la version du document retournée soit non pas **d** mais $\mathbf{d^{-1}}$, celle qui précède la mise à jour.

L'application client est alors confrontée à une situation, rare mais perturbante, d'une écriture sans effet apparent, au moins immédiat. C'est le mode d'opération le plus courant des systèmes NoSQL, qui autorisent donc un décalage potentiel entre l'écriture et la lecture. La garantie est cependant apportée que ce décalage est temporaire et que toutes les versions vont être synchronisées « à terme » (délai non précisé). On parle donc de cohérence à terme

(eventualconsistency en anglais). Enfin, le dernier cas (C) correspond à une topologie multi nœuds, en mode asynchrone. Les écritures peuvent se faire sur n'importe quel nœud, ce qui améliore l'extensibilité du système. L'inconvénient est que deux écritures concurrentes du même document peuvent s'effectuer en parallèle sur deux nœuds distincts. Au moment où la synchronisation s'effectue, le système va découvrir au mieux que les deux versions sont en conflit. Le conflit est reporté à l'application qui doit effectuer une réconciliation (il n'existe pas de mode automatique de réconciliation).

En résumé, trois niveaux de cohérence peuvent se rencontrer dans les systèmes NoSQL :

- Cohérence forte : toutes les copies sont toujours en phase, le prix à payer étant un délai pour chaque écriture.
- Cohérence faible : les copies ne sont pas forcément en phase, et rien ne garantit qu'elles le soient ; cette situation, trop problématique, a été abandonnée
- Cohérence à terme : c'est le niveau de cohérence typique des systèmes NoSQL : les copies ne sont pas immédiatement en phase, mais le système garantit qu'elles le seront « à terme ».

Dans la cohérence à terme, il existe un risque, faible mais réel, de constater de temps en temps un décalage entre une écriture et une lecture. Il s'agit d'un marqueur typique des systèmes NoSQL par rapport aux systèmes relationnels, résultant d'un choix de conception privilégiant l'efficacité à la fiabilité stricte.

Section 2. Classification des solutions de réplication

1. Mono maitre/ multi maitre

Mono maitre. Cette solution consiste à répartir la charge entre plusieurs esclaves pour améliorer les performances. Dans cet environnement, toutes les écritures et mises à jour doivent avoir lieu sur le serveur maître. La lecture, cependant peut prendre place sur un ou plusieurs esclaves. Ce modèle peut améliorer la performance des écritures (puisque le maître est dédié aux mises à jour), tout en augmentant la vitesse de lecture à travers un nombre croissant d'esclaves.

Au niveau sécurité des données, les données sont répliquées vers l'esclave et l'esclave peut interrompre le processus de réplication, il est possible d'exécuter des services de sauvegarde sur l'esclave sans corrompre les données de base correspondantes. L'analyse des données directes peuvent être créées sur le maître, tandis que l'analyse des informations peut avoir lieu sur l'esclave sans affecter les performances du maître. Concernant la distribution des données longue distance, si une succursale souhaite travailler avec une copie de vos données principales, vous pouvez utiliser la réplication pour créer une copie locale des données pour leur utilisation sans avoir besoin d'un accès permanent au maître

Nous pourrons aussi noté que la réplication avec MySQL est basée sur le fait que le serveur maître va garder la trace de toutes les évolutions de vos bases (modifications, effacements, etc.) dans un fichier de log binaire. Les esclaves pourront se connecter au maître

pour lire les requêtes stockées dans ce fichier de log, puis les exécuter. Il est très important de comprendre que le fichier de log binaire est simplement un enregistrement des modifications depuis un point fixe dans le temps (le moment ou vous activez le log binaire). Les esclaves devront par conséquent avoir les mêmes données que notre MySQL maître avant l'activation des logs binaires. Si nos esclaves n'ont pas le même jeu de donnée, alors notre réplication sera corrompue et échouera.

L'avantage d'une telle approche est qu'elle facilite la gestion de la cohérence globale du système (cohérence mutuelle) puisque toutes les mises à jour sont effectuées sur une seule copie. Cependant, cet avantage est contradictoire avec le passage à l'échelle et avec la disponibilité qui nécessitent que plusieurs copies soient utilisées en même temps pour faire face à une charge applicative d'écritures très importante et variable.

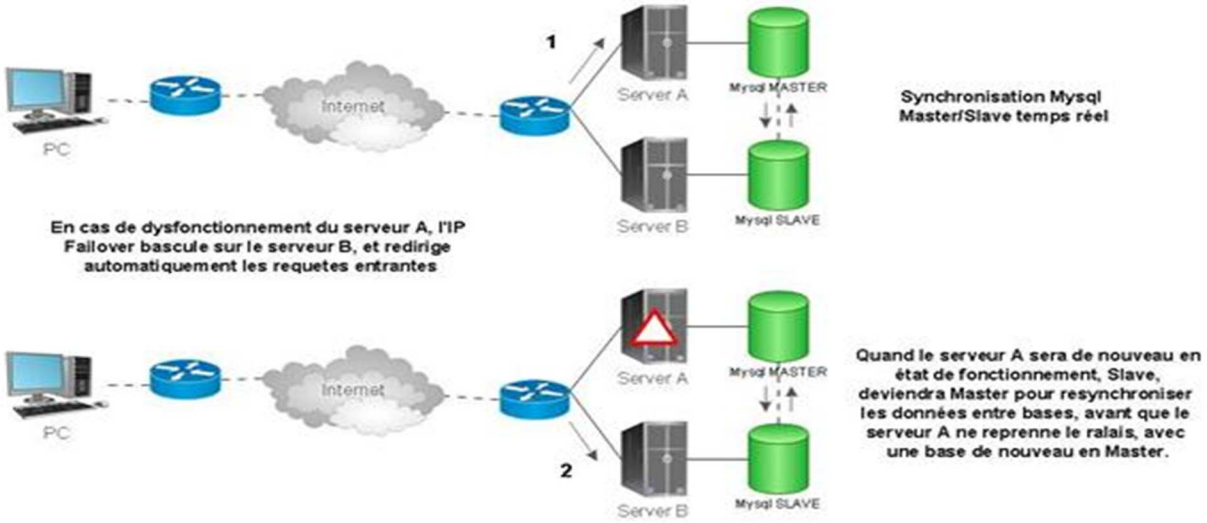


Figure 5: Mono maître [w12]

Une approche multi maître, dans laquelle les mises à jour peuvent être exécutées sur n'importe quelle réplique, permet d'améliorer aussi bien les performances des transactions de lecture seule que d'écriture. L'inconvénient de cette approche est qu'elle requiert des mécanismes de contrôle de concurrence distribués plus complexes pour garantir la cohérence mutuelle. La configuration mono maître profite plus aux applications de types OLAP avec lesquelles les données de production doivent être séparées des données d'analyse. Par contre les applications OLTP tirent plus de bénéfices de l'approche multi maître et particulièrement quand le nombre de mises à jour est très important et provient de diverses sources (utilisateurs). Il existe beaucoup de produits commerciaux qui offrent des solutions de réplication mono maître ou multi maître

Pour les architecture mono maître, nous pouvons citer de manière non exhaustive Microsoft SQL Server replication, Oracle Streams, Sybase replication Server, MySQL replication, IBM DB2 DataPropagator, GoldenGate TDM platform, et Veritas Volume Replicator. Alors que les exemples de la solution multi maître incluent Continuent uni/Cluster, XkotoGridscale, MySQL Cluster, DB2 Integrated Cluster.

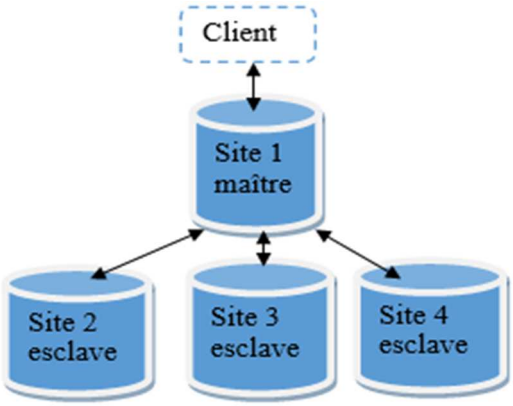


Figure *6*: Mono maître [w22]

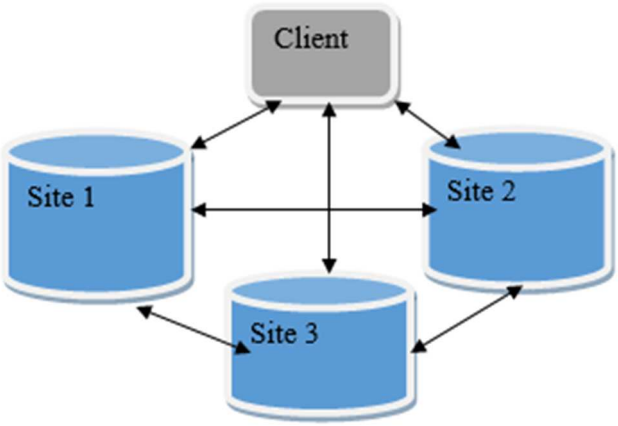


Figure *7*: Multi maître [w22]

2. Synchrone /asynchrone

Il existe une grande différence entre une réplication synchrone et une réplication asynchrone. Le choix détermine s'il y'a perte de données sur panne et failover (gère le basculement automatique d'un pare-feu à un autre. Combinez les serveurs de plusieurs générations dans le même pool de serveurs virtualisés pour étendre les fonctionnalités de reprise après incident, d'équilibrage de charge et de reprise après sinistre) d'une application critique. La réplication synchrone est essentielle pour le failover d'applications transactionnelles. Avec une réplication synchrone, toutes les données comités sur le disque du serveur primaire se retrouvent sur le disque du serveur secondaire. Avec une réplication asynchrone, des donnés comités sur le disque du serveur primaire peuvent être perdue en cas de panne. Il existe aussi une solution alternative appelée réplication semi-synchrone avec les données comités sur le serveur secondaire mais pas forcément sur son disque.

2.1. Synchrone.

Les mises à jour sont effectuées en une seule transaction. Il y a une coordination forte entre les nœuds (tous valident la transaction ou aucun). La réplication synchrone est aussi appelée « réplication en temps réel ». La synchronisation est effectuée en temps réel puisque chaque requête est déployée sur l'ensemble des bases de données avant la validation de la requête sur le serveur ou la requête est exécutée. Ce type de réplication assure un haut degré d'intégrité

des données mais requiert une disponibilité permanente des serveurs et de la bande passante. Ce type de réplication, fortement dépendant des pannes du système, nécessite de gérer des transactions multi-sites couteuses en ressources. La réponse au client est renvoyée après que les serveurs se soient coordonnés. Comme dans un système centralisé, avant d'accéder à une donnée, on acquiert un verrou exclusif dessus. Avec le protocole RWA (Read One Write All), on améliore la disponibilité des lectures en ne demandant le verrou que sur une copie pour les lectures et sur toutes les copies pour les écritures. Par la suite, le protocole est adapté aux pannes avec sa variante ROWA-A (Read OneWrite All Available), ou seules les copies disponibles sont mises à jour. Dès qu'une copie redevient disponible, elle doit d'abord être synchronisée avant d'être de nouveau utilisable.

D'autres protocoles synchrones comme le protocole Non-Dsjointconflictclasses and Optimistic multicast (NODO) se basent sur la division des données en classes de conflits. On détermine ainsi que deux transactions accédant à la même classe de conflit ont une grande probabilité de conflit. A l'inverse, deux transactions qui n'accèdent pas aux mêmes classes peuvent s'exécuter en parallèle. Les classes de conflits doivent être connues à l'avance et chaque classe de conflit à un seul nœud maître. Pour exécuter une transaction, le protocole diffuse deux messages.

Le premier message permet d'exécuter la transaction sur une copie ombre (une copie de la dernière version cohérente de la base), le deuxième message permet de valider la transaction ou alors de l'annuler pour la réordonnancer. Ainsi, au moins une des copies secondaires aura la dernière valeur. Les autres solutions combinent l'approche ROW/ROWAA. Ces protocoles utilisent un protocole de validation de type Validation à deux phases (V2P) pour valider la transaction. Le protocole s'assure alors que tous les nœuds font le même choix : validation ou abandon de la transaction. Ces protocoles assurent une cohérence sur toutes les copies.

2.2. Asynchrone.

Les mises à jour se font dans des transactions séparées. Le serveur initiateur renvoie la réponse au client immédiatement après la mise à jour sur sa base de données. Il propage ensuite la mise à jour aux autres nœuds. Ce modèle a la particularité d'envoyer la réponse au client avant la coordination des copies et avant même la diffusion de la mise à jour sur les autres copies. Cette propriété impose une contrainte forte, une réconciliation doit être effectuée pour assurer la cohérence de l'ensemble des copies. La réplication asynchrone stocke les opérations intervenues sur une base de données dans une queue locale pour les propager plus tard à l'aide d'un processus de synchronisation. Ce type de réplication est plus flexible que la réplication synchrone. Il permet en effet de définir les intervalles de synchronisation, ce qui permet à un site de fonctionner même si un site de réplication n'est pas accessible. L'utilisateur peut ainsi déclarer que la synchronisation sera effectuée la nuit, à une heure de faible affluence. Si le site distant est victime d'une panne, l'absence de synchronisation n'empêche pas la consistance de la base maître et esclave. Les opérations sur les données sont plus rapides, puisqu'une requête n'est pas immédiatement déployée. Le trafic sur le réseau est de ce fait plus compact. Par contre, le planning de réplication est dans ce cas plus complexe, puisqu'il s'agit de gérer les conflits émanant d'un éventuel accès en écriture sur une base esclave entre deux mises à jour.

Cependant les algorithmes de réplication asynchrone sont souvent déterministes : ils appliquent les mises à jour en étant sûrs qu'elles ne donneront pas d'incohérences. On peut classer les algorithmes de réplication synchrone en deux grandes familles : la réplication asynchrone optimiste et la réplication asynchrone pessimiste. Avec la réplication asynchrone optimiste, les transactions sont exécutées en parallèle sans contrôle de cohérence. En cas de conflit, il faut réconcilier les copies pour que les données convergent vers la même valeur : soit en ré-exécutant les transactions et en les réordonnant, soit en annulant certaines transactions. Cette technique est efficace dans les cas où les conflits entre les transactions sont rares et lorsque les transactions acceptent une certaine incohérence des données. L'un des avantages de la réplication asynchrone optimiste est que les transactions sont exécutées localement, un nœud du système peut donc exécuter une transaction en étant déconnecté du reste du système, la réconciliation se faisant à la reconnexion du nœud. De plus, en l'absence de contrôle de cohérence, les transactions sont exécutées sans les délais supplémentaires introduits par la réplication pessimiste synchrone ou asynchrone. Un désavantage de cette technique est que la réconciliation s'effectue après avoir renvoyé la réponse au client. Ainsi, le client peut recevoir une réponse incohérente et le nœud peut se trouver dans un état incohérent jusqu'à la réconciliation.

3. La réplication asynchrone pessimiste

Tout comme la réplication synchrone garantit une cohérence forte, un bon encodage de données ne peut pas être garanti dans tous les cas. Pour contourner ce problème, la solution consiste à restreindre le placement des copies primaires et secondaires sur les nœuds. Une autre solution hybride ou la réplication synchrone est utilisée lorsqu'il y a des cycles dans le nœud. Bien que toutes ces solutions couvrent un large environnent de configurations, dans les applications réelles, la restriction sur le placement et l'approche centralisée des mises à jour sont trop restrictives et limitent le type de transactions autorisées à être exécutées.

4. La réplication asynchrone multi-maître.

Pour les serveurs qui ne sont pas connectés en permanence, comme les ordinateurs portables ou les serveurs distants, conserver la cohérence des données entre les serveurs est un challenge. L'utilisation de la réplication asynchrone multi-maître permet à chaque serveur de fonctionner indépendamment. Il communique alors périodiquement avec les autres serveurs pour identifier les transactions conflictuelles. La gestion des conflits est alors confiée aux utilisateurs ou à un système de règles de résolution. Bucardo est un exemple de ce type de réplication. Bucardo est asynchrone, un système de réplication qui permet des opérations à la fois multi-maître et multi-esclaves. Il a été développé par Jon Jensen et Greg Sabino Mullane de End point Corporation, et est maintenant utilisé dans de nombreuses autres organisations. Bucardo est un logiciel libre et open source distribué sous licence BSD.

5. La réplication synchrone multi-maître.

Dans les réplications synchrones multi-maitres, tous les serveurs acceptent les requêtes en écriture. Les données sont transmises du serveur d'origine à tous les autres serveurs avant toute validation de transaction. Une activité importante en écriture peut être la cause d'un verrouillage excessif et conduire à un effondrement des performances. Dans les faits, les performances en écriture sont souvent plus importantes que celles d'un simple serveur. Tous les serveurs acceptent les requêtes en lecture. Certaines implantations utilisent les disques

partagés pour réduire la surcharge de communication. Les performances de la réplication synchrone multi-maître sont meilleures lorsque les opérations de lecture représentent l'essentiel de la charge, alors que son grand avantage est l'acceptation des requêtes d'écriture par tous les serveurs. Il n'est pas nécessaire de répartir la charge entre les serveurs maîtres et esclaves, parce que les modifications de données sont envoyées d'un serveur à l'autre. Les fonctions non déterministiques, comme random() (fonctions aléatoires), ne posent aucun problème.

6. La réplication semi-synchrone.

Avec une réplication semi-synchrone au niveau fichier, l'asynchronisme n'est pas réalisé sur la machine primaire mais sur la machine secondaire. Dans cette solution, l'acquittement des deux machines est toujours attendu avant d'envoyer l'acquittement à l'application ou au cache système. Mais sur la secondaire, il y a deux options asynchrone ou synchrone.

Dans le cas asynchrone, la secondaire envoie l'acquittement à la primaire dès réception de l'IO puis écrit sur disque. Dans le cas synchrone, la secondaire écrit l'IO sur disque puis envoie l'acquittement à la primaire. Mais nous devons faire attention, le mode synchrone sur le secondaire est nécessaire si l'on considère une double panne électrique simultanée des deux serveurs avec impossibilité de redémarrer l'ex serveur primaire et obligation de redémarrer sur le secondaire.

Chapitre 2 : Les types de réplications

La réplication a principalement pour but de disposer d'un deuxième serveur, copie du premier, ou travailler au cas où le premier tomberait en panne. L'idée est que l'activité ne doit pas être impactée par la perte d'un serveur. Mais ce n'est pas la seule raison pour laquelle un administrateur veut la réplication. Le serveur répliqué peut aussi servir à y déporter une partie du travail, dans le but d'alléger la charge du serveur principal. Dans ce cadre, plusieurs types de réplications vont apparaître, chacune permettant d'apporter une solution à un type de problème.

Section 1. Réplication asymétrique (maître et esclave) avec propagation asynchrone

C'est le mode de réplication le plus simple. Un serveur est en lecture écriture, il est généralement appelé serveur maître. Le ou les autres serveurs ne sont pas disponibles en écriture. Ils peuvent cependant être disponibles en lecture. Tout enregistrement fait sur le maître n'est pas immédiatement reporté sur l'esclave. Il existe généralement un petit délai avant application sur l'esclave. Cela sous-entend que, si le serveur maître est tombé en panne avant d'avoir eu le temps de transférer les dernières transactions au serveur esclave, il manquera les données de ces transactions sur l'esclave. Il faut donc être prêt à accepter une certaine perte, généralement mineure. De plus, si l'esclave sert à répartir la charge, il est possible qu'une lecture du maître et qu'une lecture de l'esclave ne donnent pas le même résultat

- Ecriture sur le maître
  Dans la réplication asymétrique, seul le maître accepte des écritures, et le ou les esclave(s) ne sont accessibles qu'en lecture
- Mise en attente
  Dans la réplication asynchrone, il existe un processus extérieur au SGBD qui gère la réplication des changements.
- Réplication des changements sur l'esclave
  Les seules « écritures » acceptées par le ou les esclave(s) sont la réplication des changements effectués par les utilisateurs sur le maître.

La mise à jour de la ou des tables répliquées est différée (asynchrone), elle est réalisée par un programmateur de tâches, possédant une horloge. Des points de synchronisation sont utilisés pour propager les changements.

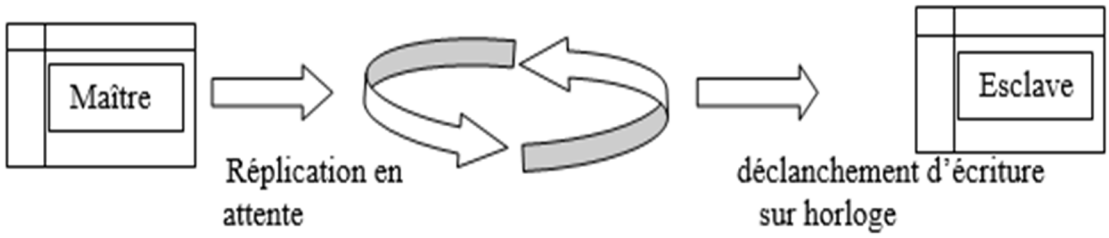


Figure 8: Réplication asymétrique avec propagation asynchrone

Section 2. Réplication asymétrique (maître et esclave) avec propagation synchrone

Ce mode de réplication n'est pas trop complexe, mais lent. Il n'y a qu'un seul maître mais chaque modification réalisée sur le maître doit être enregistrée sur l'esclave avant de redonner la main à l'utilisateur. Ce qui est source de lenteurs (deux systèmes doivent avoir enregistré l'information au lieu d'un seul). C'est la meilleure solution s'il est inconcevable de perdre des données suite à l'arrêt inopiné du maître. Par contre, cela ne résout pas complètement le problème de la répartition de charge. En effet, cette solution garantit seulement que la donnée est enregistrée sur les esclaves, pas qu'elle soit visible. Donc, encore une fois, si l'esclave sert à répartir la charge, il est possible qu'une lecture du maître et qu'une lecture de l'esclave ne donnent pas le même résultat, même si le risque est minimisé par rapport aux solutions.

- Ecriture sur le maître

  Dans la réplication asymétrique, seul le maître accepte des écritures, le ou les esclaves ne sont accessibles qu'en lecture.
- Recopie instantanée sur l'esclave

  Dans la réplication synchrone, il n'y pas de processus extérieur qui propage les changements. Dans ce cas, on utilise un mécanisme dit Two Phase Commit ou « Commit en deux phases », qui assure qu'une transaction est comité sur tous les nœuds dans la même transaction. Les propriétés ACID sont dans ce cas respectées.

La copie est instantanément mise à jour à chaque modification de la table « maître ». Si la copie échoue c'est toute la transaction qui est annulée, et elle n'est appliquée sur aucun des nœuds.

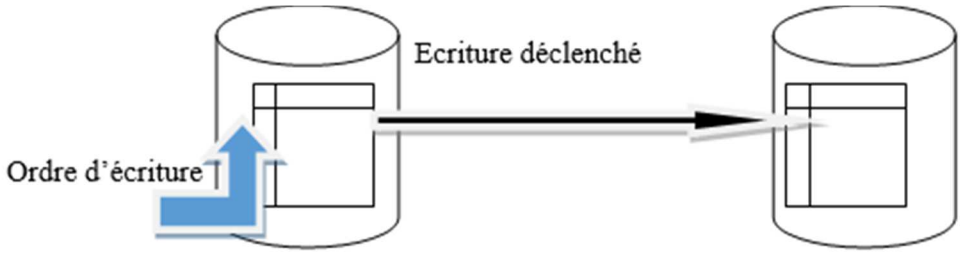


Figure 9: Réplication asymétrique avec propagation synchrone [w3]

Section 3. Réplication symétrique avec propagation synchrone

La réplication symétrique synchrone gère des accès concurrentiels aux écritures, toutes les copies sont accessibles à la fois en lecture et en écriture. Chaque copie possède ses propres triggers qui lors de la réception d'une requête de mise à jour déclenchent le déploiement de cette dernière vers toutes les autres copies. A ce moment entre en jeu le protocole de validation atomique qui doit s'assurer que la transaction est bien validée sur toutes les copies ou sur aucune. Dans ce cas, il n'y a pas de table maître, mais chaque table possède ses triggers, déclenchés lors d'une modification. Il est alors nécessaire de définir des priorités et de gérer les blocages des tables en attendant qu'une modification soit entièrement propagée. D'autre part, les triggers doivent différencier les mises à jour issues d'une réplication des mises à jour de requête directes.

- Ecritures concurrentielles sur deux « maîtres »

  Dans la réplication symétrique, tous les maîtres sont accessibles aux utilisateurs, aussi bien en lecture, qu'en écriture.
- Gestion des verrous et de la concurrence

  Dans la réplication synchrone, il n'y a pas de processus extérieur qui propage les changements. Dans ce cas, on utilise un mécanisme dit de Two Phase Commit ou « Commit en deux phases », qui assure qu'une transaction est comité sur tous les nœuds dans la même transaction. Les propriétés ACID sont dans ce cas respectées.

  Dans le cas particulier de la réplication synchrone symétrique, il faut en plus gérer les éventuels conflits qui peuvent survenir quand deux transactions concurrentes opèrent sur le même ensemble de tuples. On résout ces cas particuliers avec des algorithmes plus ou moins complexes.

Les deux tables peuvent être modifiées, et les mises à jour sont propagées directement dans l'autre table. Il est à noter que la réplication fait partie de la transaction, ce qui ne ralentit que très peu le système.

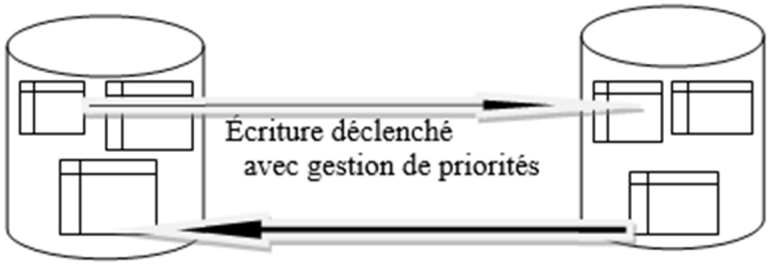


Figure *10*: Réplication symétrique avec propagation synchrone [w3]

NB : la réplication symétrique ou Peer-to-peer (update everywhere) avec propagation asynchrone

Ce mode est plus complexe. Les serveurs sont tous en lecture et écriture. Il faut donc pouvoir gérer les conflits causés par la mise à jour des mêmes objets sur plusieurs serveurs en même temps. Cela rends beaucoup plus difficile le respect de la norme ACID. Le principe de la réplication symétrique asynchrone est basé sur trois éléments :

- Ecritures sur le maître : les écritures initiées par les utilisateurs sont orientés vers un site unique (le site maître), les esclaves n'acceptent que les requêtes de lecture. Dans la réplication symétrique, tous les maîtres sont accessibles aux utilisateurs, aussi bien en lecture, qu'en écriture.
- Mises en attente

  Chaque écriture effectuée sur le site maître est sauvegardée dans une file locale ou dans un journal. C'est un processus extérieur au SGBD qui scrute en permanence la file (ou le journal) afin de détecter les modifications apportées au site maître. Ce dernier se charge par la suite de propager ces changements vers les copies secondaires (esclaves) via un déclencheur tel que l'horloge.
- Propagation des changements sur l'esclave

  Les seules écritures admises par les esclaves sont celles effectués par les utilisateurs sur le maître.

Une panne qui intervient sur le site maître avant qu'il n'ait eu le temps d'enregistrer la requête dans une file d'attente ou un journal, entraîne la perte de la mise à jour. Deux « maître » répliquent les données de l'un sur l'autre, via un programmateur (voire deux programmateurs). Ce mode de réplication ne respecte généralement pas les propriétés ACID, car si une copie échoue alors que la transaction a déjà été validée, on peut alors se trouver dans une situation ou les données sont incohérentes entre les serveurs.

Figure *11*: Réplication symétrique avec propagation asynchrone

1. Etude comparative des types de réplication :

Fonctionnalité	Bascule par disque partagés	Réplication par système de fichiers	Secours semi-automatique	Réplication maître/esclave basé sur les triggers	Middleware de réplication sur instructions	Réplication asynchrone multi-maîtres	Réplication synchrone multi-maîtres
Exemple d'implémentation	NAS	DRBD	PITR	Slony	Pgpool-II	Bucardo	
Méthode de communication	Disque partagé	Blocs disque	WAL	Lignes de tables	SQL	Lignes de tables	Lignes de tables et verrous de ligne
Ne requiert aucun matériel spécial		Oui	Oui	Oui	Oui	Oui	Oui
Autorise plusieurs serveurs maîtres					Oui	Oui	Oui
Pas de surcharge sur le serveur maître	Oui		Oui		Oui		
Pas d'attente entre serveurs	Oui		Oui	Oui		Oui	
Pas de perte de données en cas de panne du maître	Oui	Oui			Oui		Oui
Les esclaves acceptent les requêtes en lecture seule			Hot only	Oui	Oui	Oui	Oui
Granularité de niveau table				Oui		Oui	Oui
Ne nécessite pas de résolution de conflit	Oui	Oui	Oui	Oui			Oui

Tableau *1*Comparatif des différentes fonctionnalités au niveau des solutions de réplications [w19]

Chapitre 3 : Etat de l'art et étude comparative des outils de réplication Synchrone Symétrique

Section 1. Etat de l'art de la réplication synchrone symétrique sous MySQL

Pour pouvoir faire de la réplication synchrone symétrique sous MySQL nous disposons de plusieurs outils, permettant de gérer et de monitorer nos instances MySQL.

- **MySQL Administrator (gratuit)**

MySQL Administrator est un logiciel édité et distribué par la société MySQL AB : il permet d'administrer, de gérer et de monitorer de multiples instances MySQL (un serveur physique peut héberger plusieurs instances MySQL).

Image 1: MySQL Administrator [w30]

Fonctionnalités :

- Gestion des utilisateurs et des privilèges,
- Monitoring de la santé de votre serveur : espace disque, nombre de requêtes, taille des index, etc.
- Gestion du schéma des bases de données,
- Système de sauvegarde/restauration rapide,
- Informations clés et détails sur le serveur,
- Etat des connexions,
- Informations sur l'état de la réplication,
- Visualisation des logs.

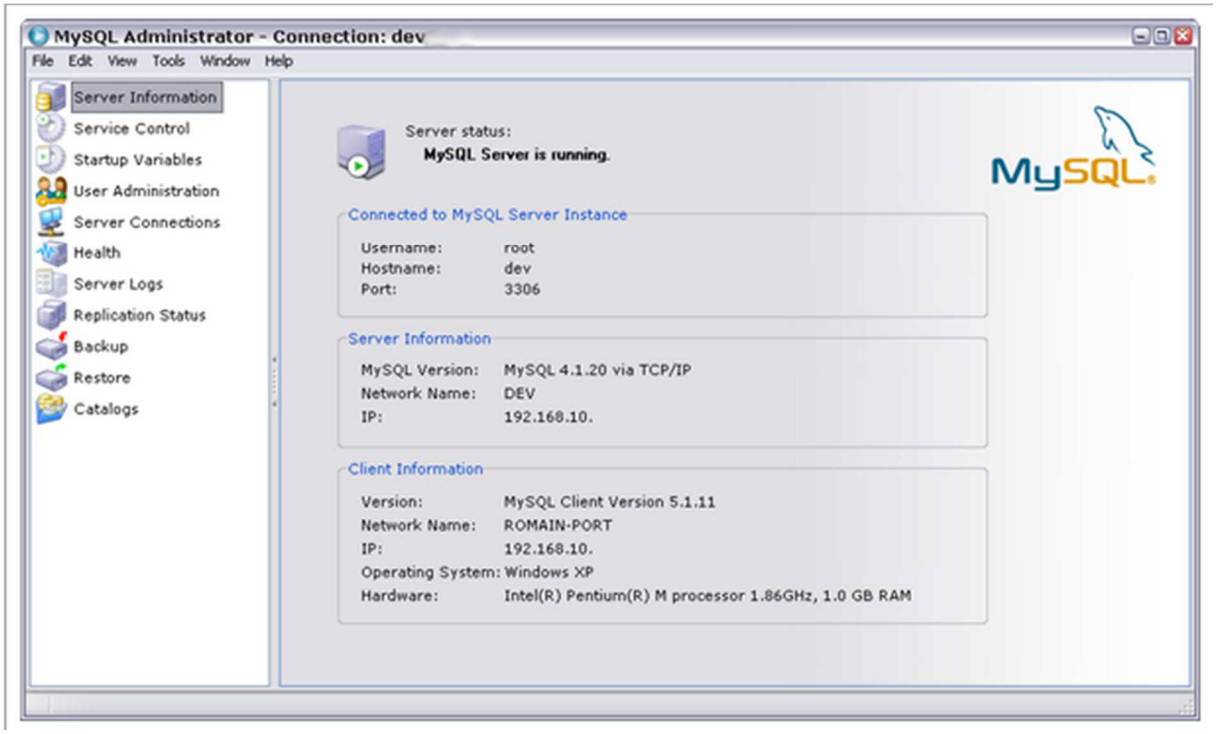


Image *2*: Information sur le serveur [w30]

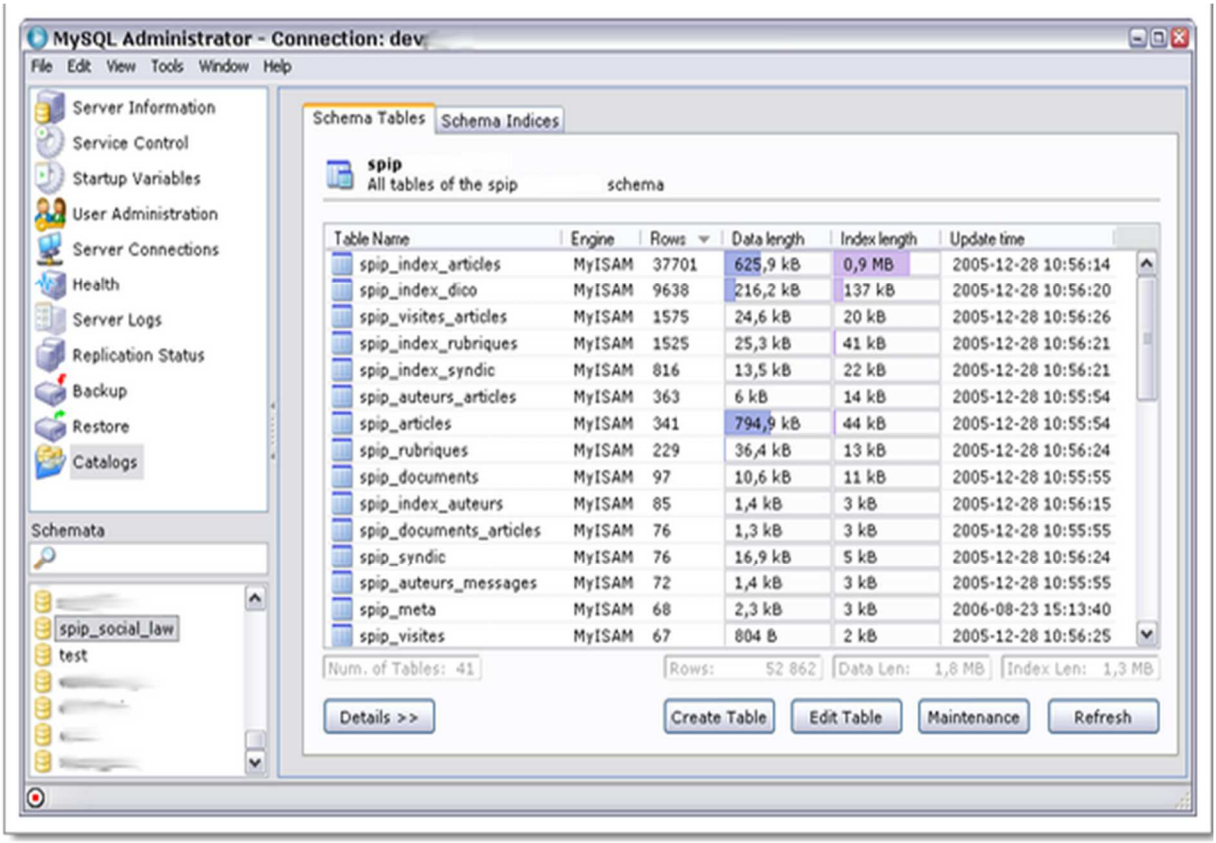


Image *3*: Détails sur une table [30]

Il est intéressant de noter que MySQL Administrator fait partie de MySQL Tools, une suite de 4 outils édités par MySQL AB :

- MySQL Administrator : Logiciel vous permettant d'administrer et de monitorer vos serveurs MySQL,
- MySQL Migration Toolkit : Framework doublé d'un assistant, migration Toolkit vous permet de migrer d'une base propriétaire (MS SQL, Oracle, etc.) vers MySQL (voir image ci-dessous),
- MySQL Query Browser : Outil de création (mode texte, ou par drag and drop), d'exécution et d'optimisation de requêtes MySQL,
- MySQL Workbench : Outil de modélisation pour MySQL vous permettant également de faire du reserve-engineering.

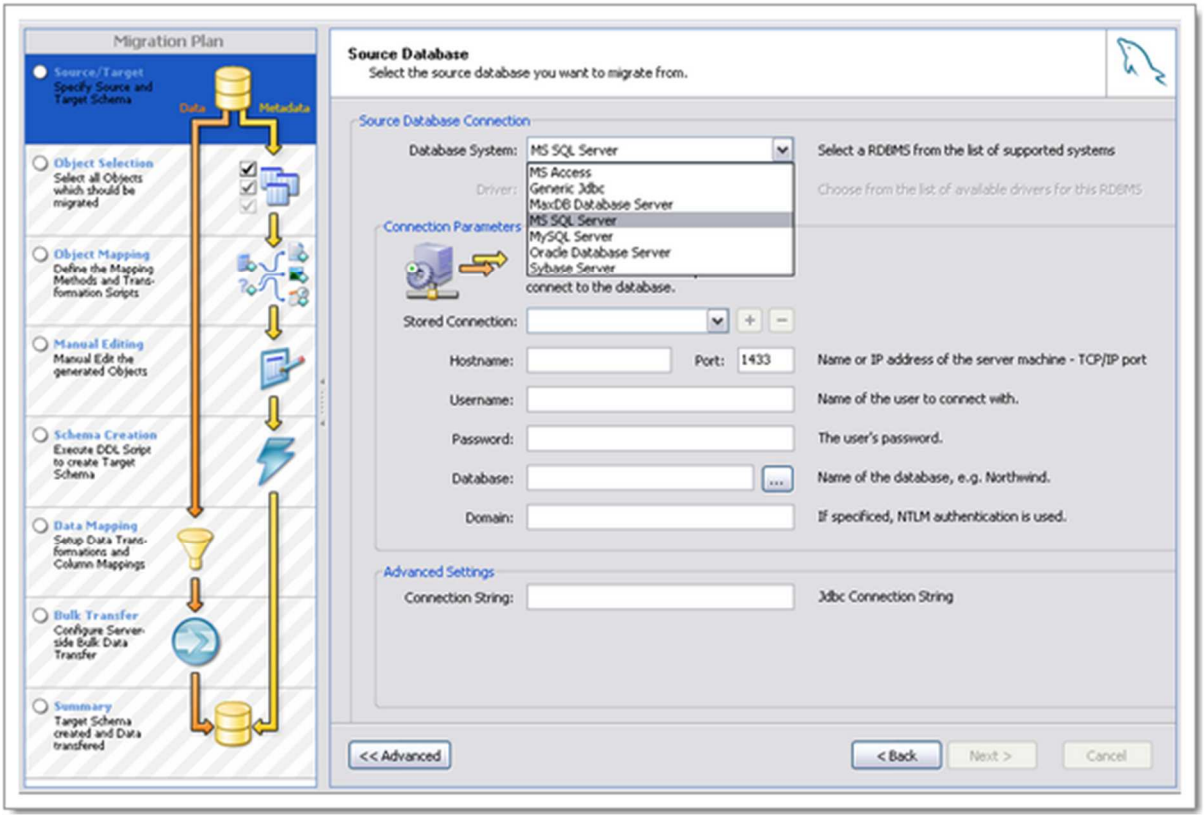


Image 4: Migration Toolkit : Assistant pour migrer vos bases propriétaires MySQL [w30]

7. Navicat for MySQL (payant)

Navicat for MySQL. Est également un logiciel de gestion de vos bases de données MySQL. Très complet, il apporte des fonctionnalités très appréciables pour l'import de données, ou encore la synchronisation (données et/ou structure) entre deux bases de données. Il vous faudra entre 95$ et 170$ pour acquérir une licence.

Fonctionnalités :

- Plusieurs modes de connexion : SSL, tunnel SSH, tunnel http,
- Gestion des utilisateurs et des privilèges,

- Gestion des bases de données et de leurs structures,
- Synchronisation de données et de structure entre deux bases,
- Assistant de sauvegarde/restauration : possibilité de programmer des sauvegardes,
- Complétion de code,
- Procédures stockées,
- Import et export de données (à partir d'Access, d'Excel, XML, PDF, et TXT),

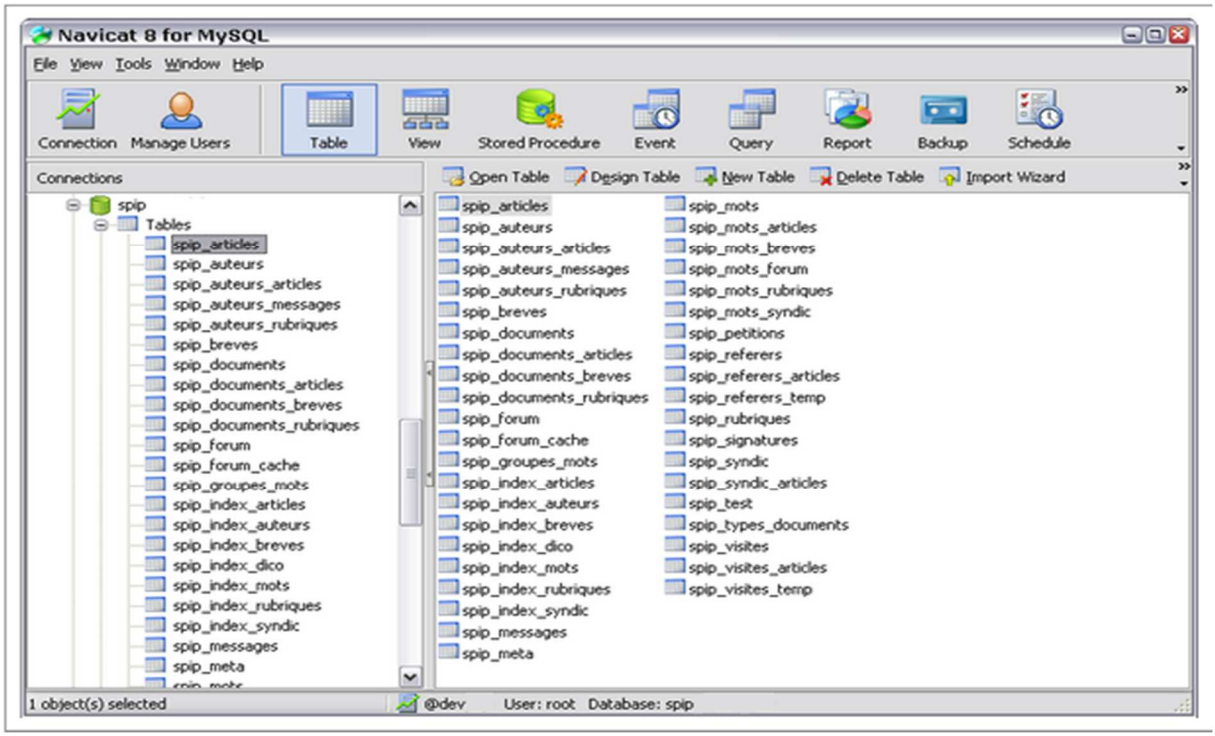


Image 5: Vue d'une base de données [w30]

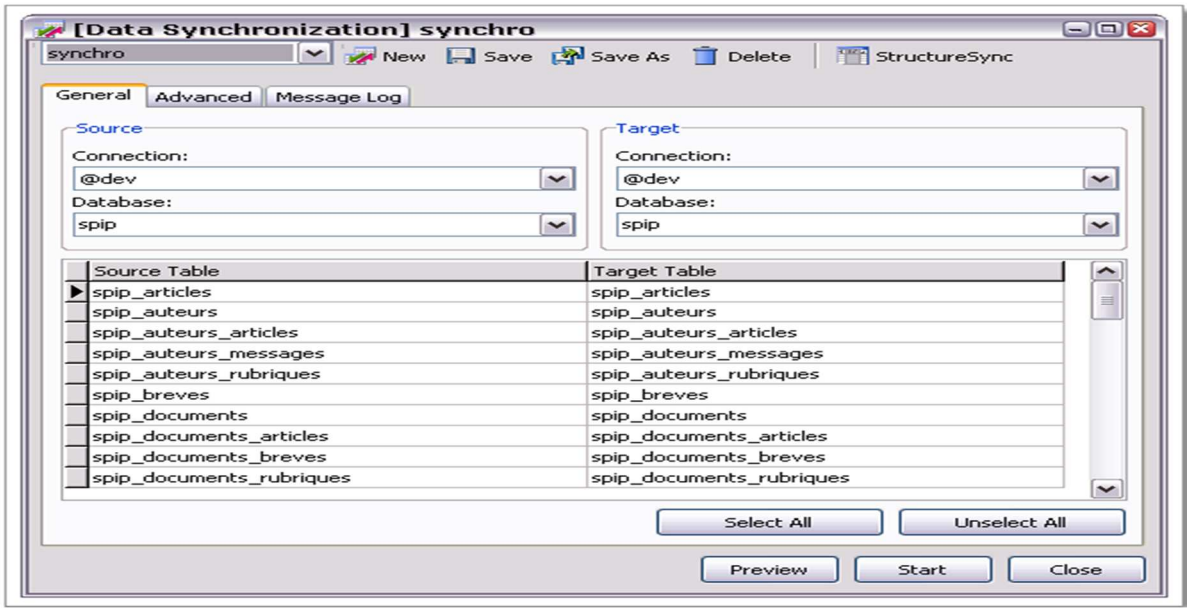


Image 6: Procédure de synchronisation entre deux bases de données [w30]

8. Spotlight on MySQL (gratuit)

Spotlight on MySQL est un outil un peu particulier, dans la mesure où il est dédié au monitoring, dont les prérequis sont assez stricts. Il ne fonctionne en effet que pour MySQL 5, et uniquement avec le moteur de stockage InnoDB. Cependant, pour les personnes concernées, c'est l'outil idéal pour identifier les problèmes de performance et les goulots d'étranglement.

Image 7: Vision globale et temps réel du comportement de votre MySQL [w30]

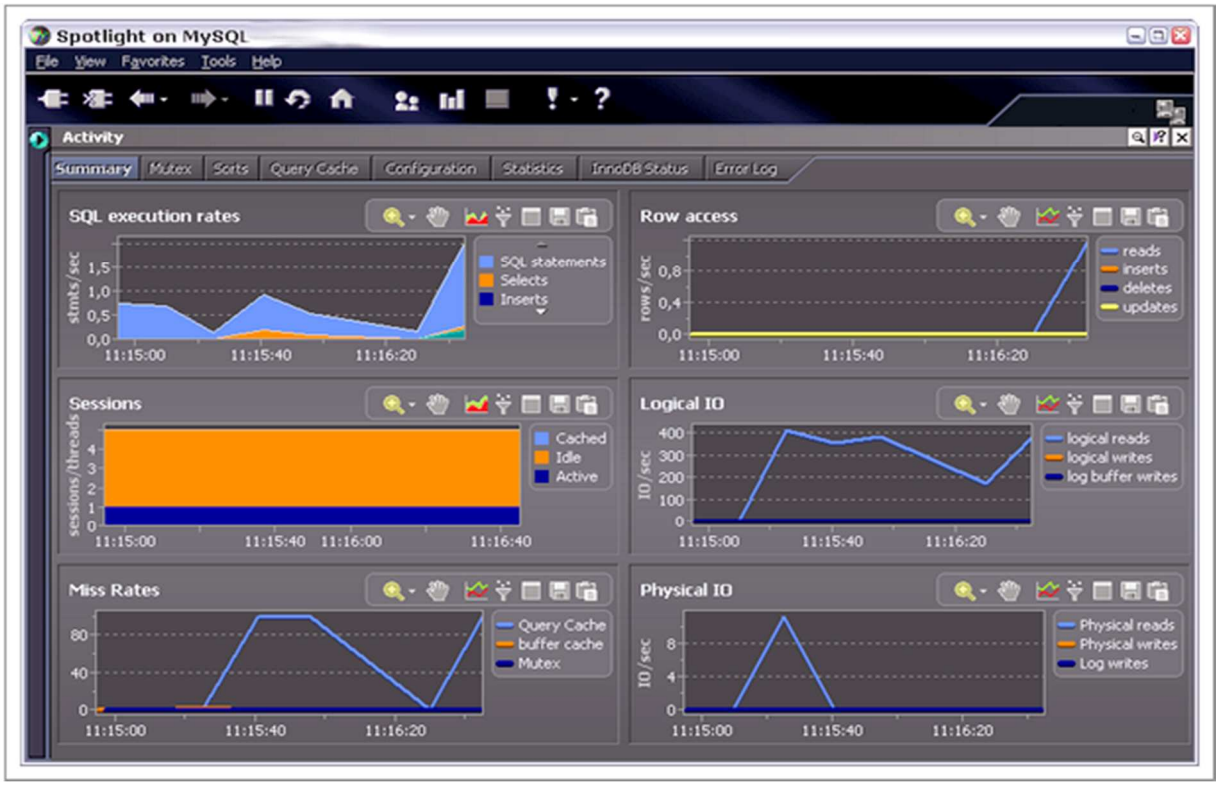


Image 8: Graphique détaillés de monitoring [w30]

Son éditeur, Quest software, propose également d'autres outils (Toad, Data Analysis, etc.) pour d'autres types de bases de données (Oracle, DB2, Sybase, SQL Server).

9. PhpMyAdmin (gratuit)

PhpMyAdmin est l'outil graphique de gestion MySQL le plus utilisé au monde. Développé en PHP, il permet d'accéder à une gestion complète de vos bases grâce à un simple navigateur.

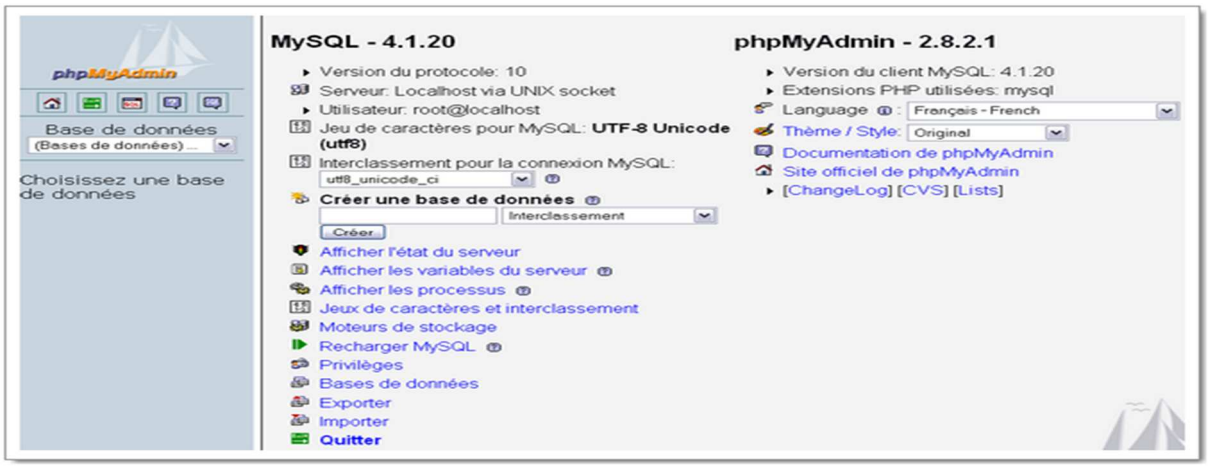


Image 9: PhpMyAdmin [w30]

10. VerticalSlave

VerticalSlave est un outil de monitoring de votre réplication MySQL. VerticalSlave est aussi un outil léger pour administrer vos bases de données répliquées et non-répliquées. Ses principales fonctionnalités sont :

- Vérification de la configuration de l'outil
- Vérification de l'état de la réplication
- Vérification de l'état de la réplication avec auto-réparation des tables en erreur
- Réinitialisation de la réplication
- Dump des bases non-répliquées
- Magasin des rapports des actions passées
- Envoi d'un rapport raccourci par mail

Prérequis

- MySQL 5.5 ou ultérieur
- PHP 5.3 ou ultérieur
- Serveur web Apache et hôte virtuel configuré
- Chaîne de certification SSL
- Environnement de réplication configuré. Les bases répliquées doivent être déclarées avec l'option replicate-do-db
- Connaissances en réplication MySQL .

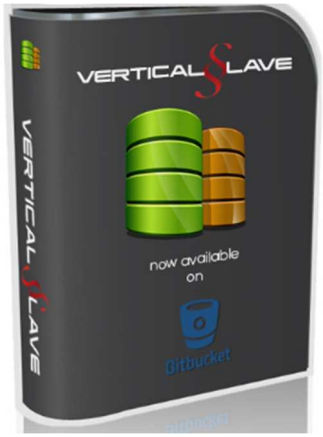


Image *10*: VerticalSlave [w29]

11. Xampp

C'un ensemble de logiciels permettant de mettre en place facilement un serveur web et un serveur FTP. Il s'agit d'une distribution de logiciels libres (X Apache MySQL Per PHP) offrant une bonne souplesse d'utilisation, réputé pour son installation simple et rapide.

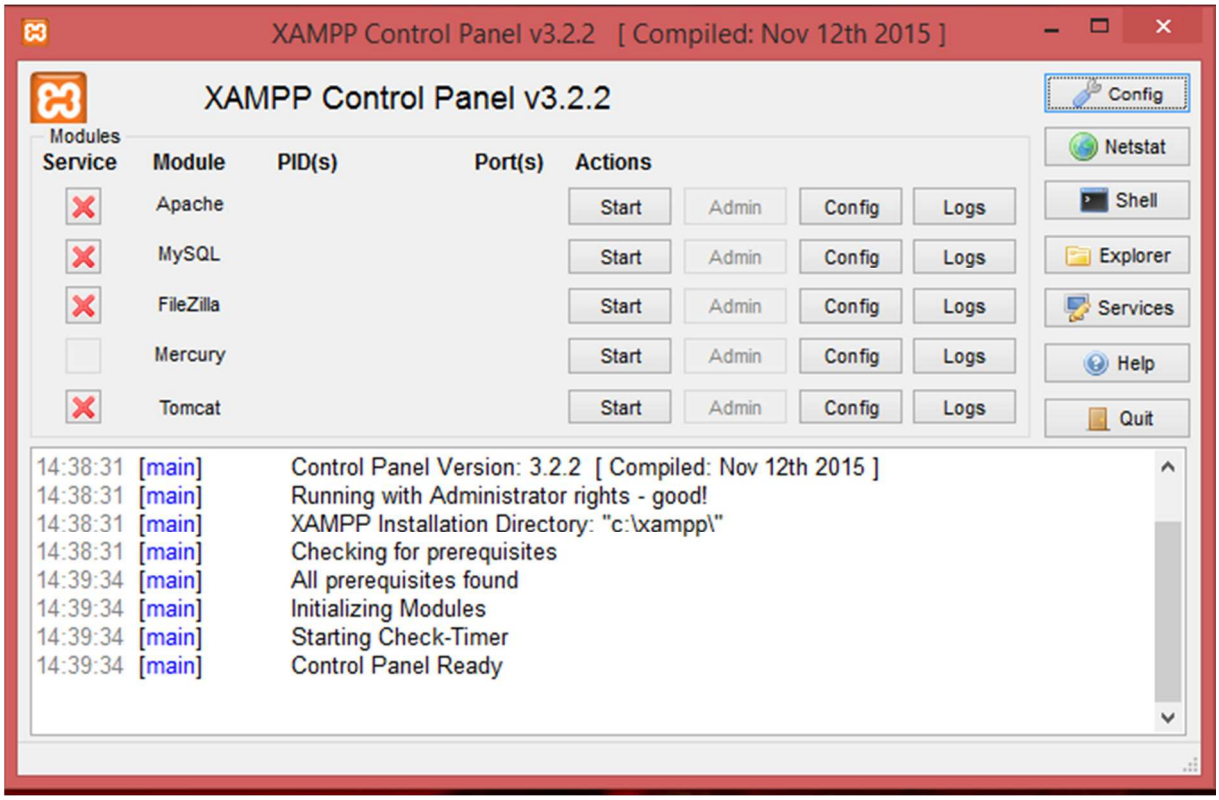


Image 11: Xampp

Section 2. Tableau comparatif des outils et choix de la technologie

Outils / Fonctionnalités	MySQL Administrator	Navicat for MySQL	Spotlight on MySQL	PhpMyAdmin	Vertical Slave
Gestion des utilisateurs et privilège	Oui	Oui		Oui	Oui
Monitoring	Oui		oui		Oui
Gestion du schéma des bases de données	Oui	Oui	Oui	Oui	Oui
Information clés et détails sur le serveur	Oui			Oui	Oui
Information sur la réplication	Oui	Oui	Oui	Oui	Oui
Etat des connexions	Oui	Oui	Oui	Oui	Oui
Système de sauvegarde/restauration rapide	Oui	Oui		Oui	Oui
Visualisation des logs	Oui	Oui		Oui	Oui
Plusieurs modes de connexion : SSL, SSH, http		Oui			
Synchronisation de données et de structure entre deux bases		Oui	Oui	Oui	oui

Complétion de code		Oui			
Procédures stockée	Oui	Oui	Oui	Oui	Oui
Import et export des données (à partir d'Access, d'Excel, XML, PDF, et TXT)		Oui		Oui	Oui
Vérification de l'état de la réplication avec autoréparation des tables en erreur					Oui
Réinitialisation de la réplication					Oui
Dump des basses non-répliquées					Oui

Tableau *2* Tableau comparatif des outils et choix de la technologie

Choix de l'outil

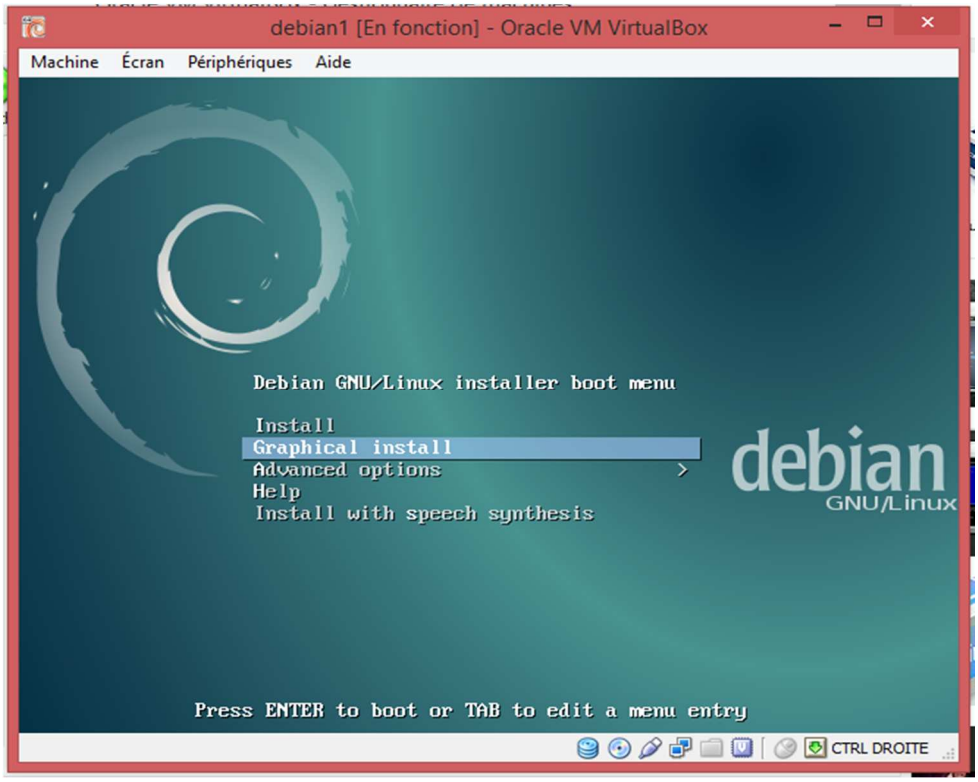
Pour le choix de l'outil nous allons utiliser Xamp sous Windows et MySQL Server sous Debian 8. Notre choix est motivé par la bonne souplesse d'utilisation de Xamp mais aussi la stabilité de Debian.

Deuxième partie : Implémentation de la réplication symétrique et synchrone sous MySQL

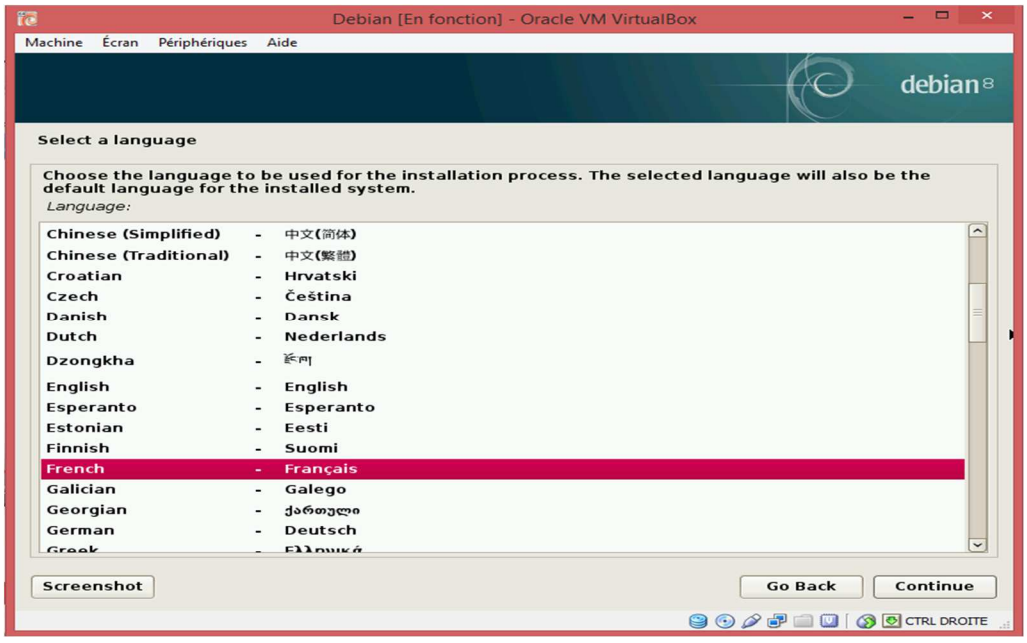
Chapitre 1 : Installation des solutions et outils nécessaires

Section 1. Debian en virtuelle sous VirtualBox

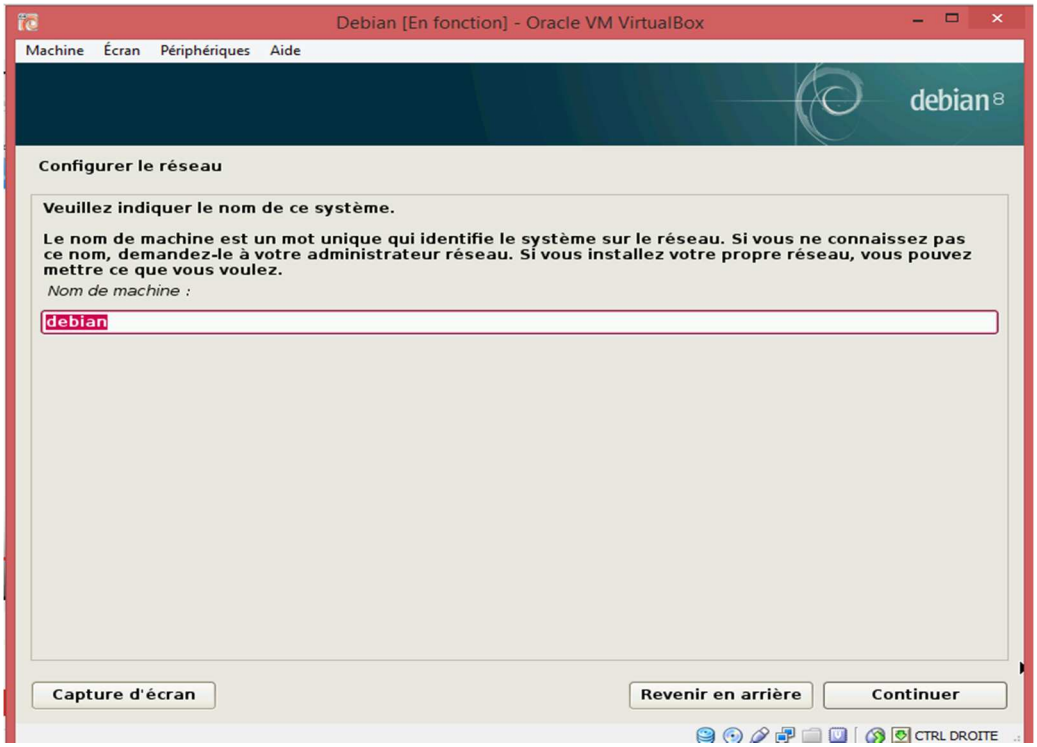
Démarrage de l'assistant d'installation choisir le mode d'installation (graphicall) puis cliquer sur la touche entrée



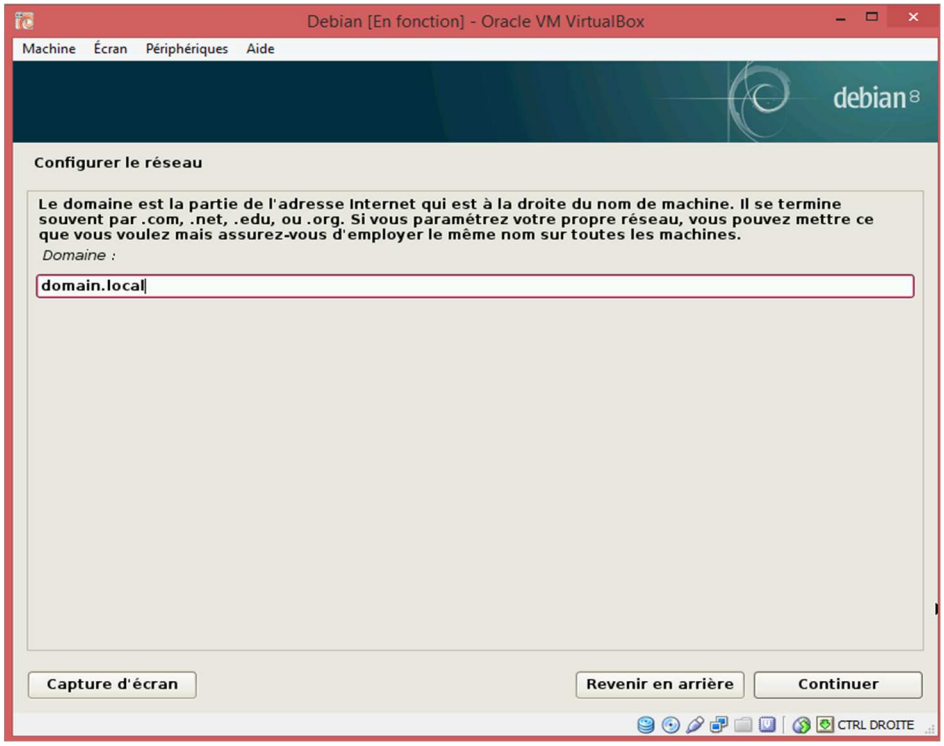
Ici, nous devons choisir comme langue bien évidemment le français ainsi que pour le clavier. Pour le choix du pays nous sélectionnons France et cliquons sur continuer.



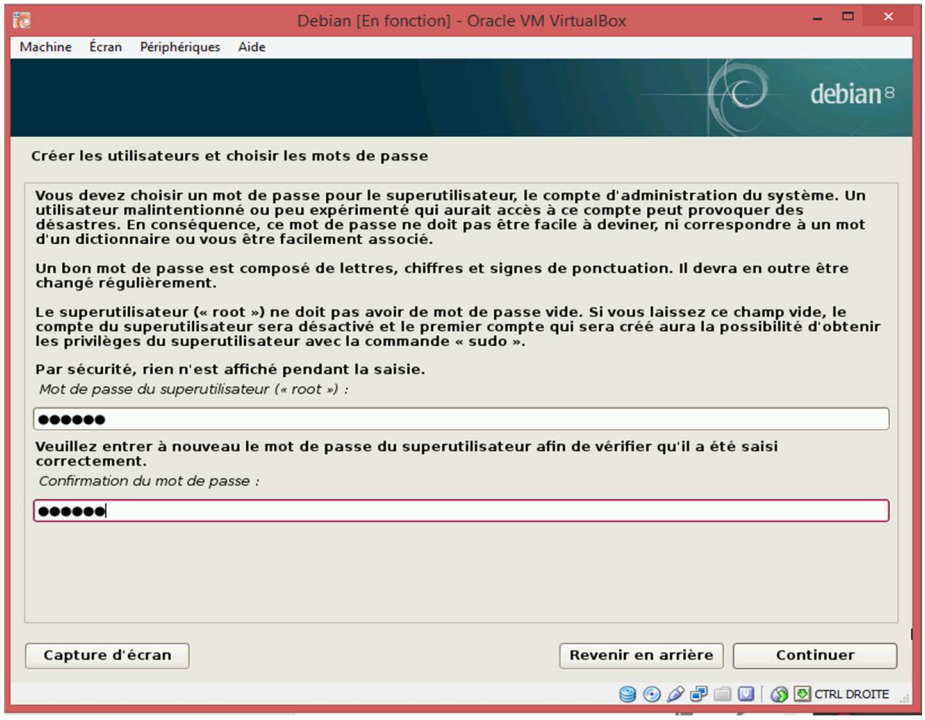
Pour le nom de la machine nous prendrons celui proposé par défaut mais nous pouvons choisir n'importe quel nom. Juste pour un test nous allons laisser le debian.



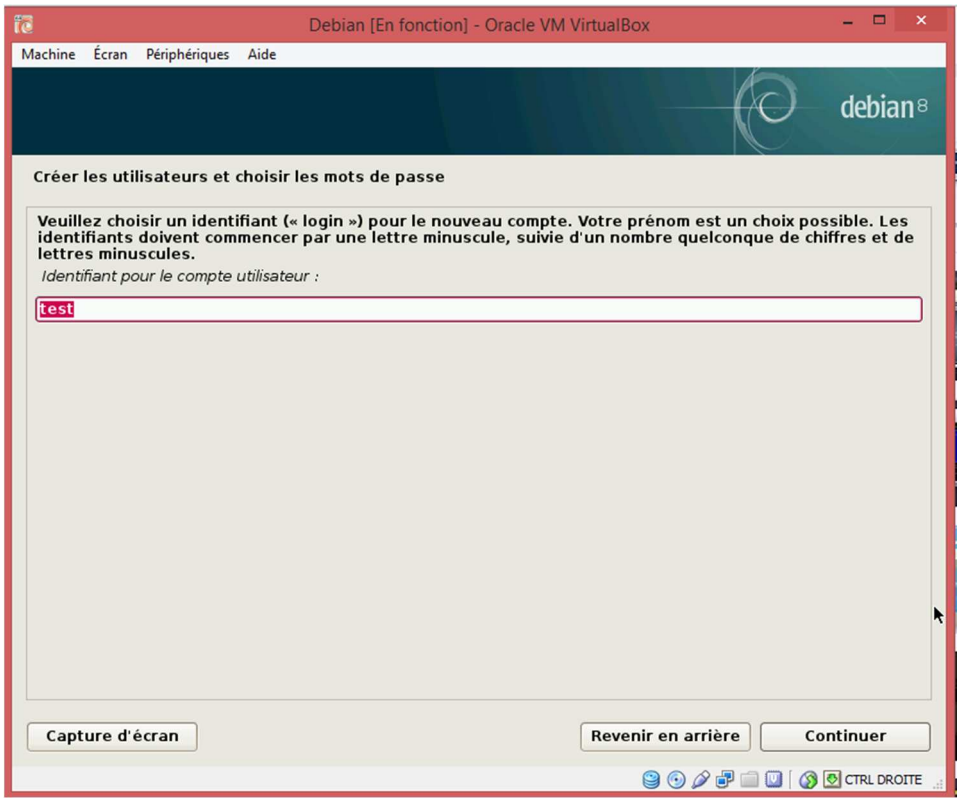
Le nom de domaine : Si vous avez un nom de domaine vous le mentionnez sinon vous pouvez choisir (domain.local).



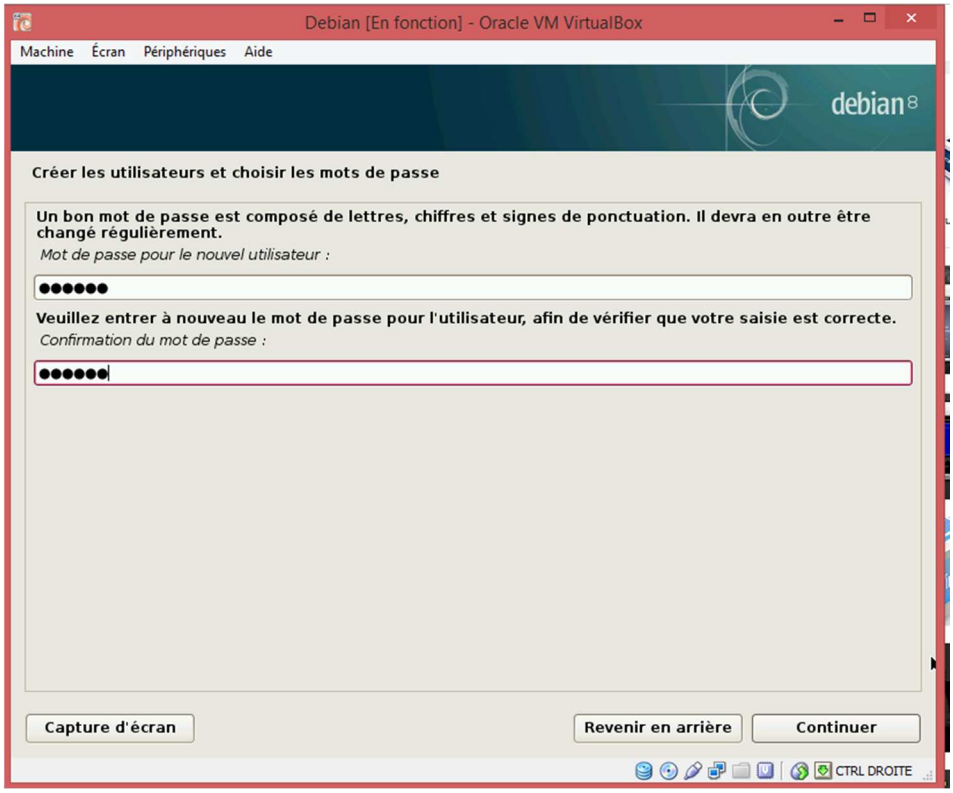
Ici nous allons définir le mot de passe du super utilisateur



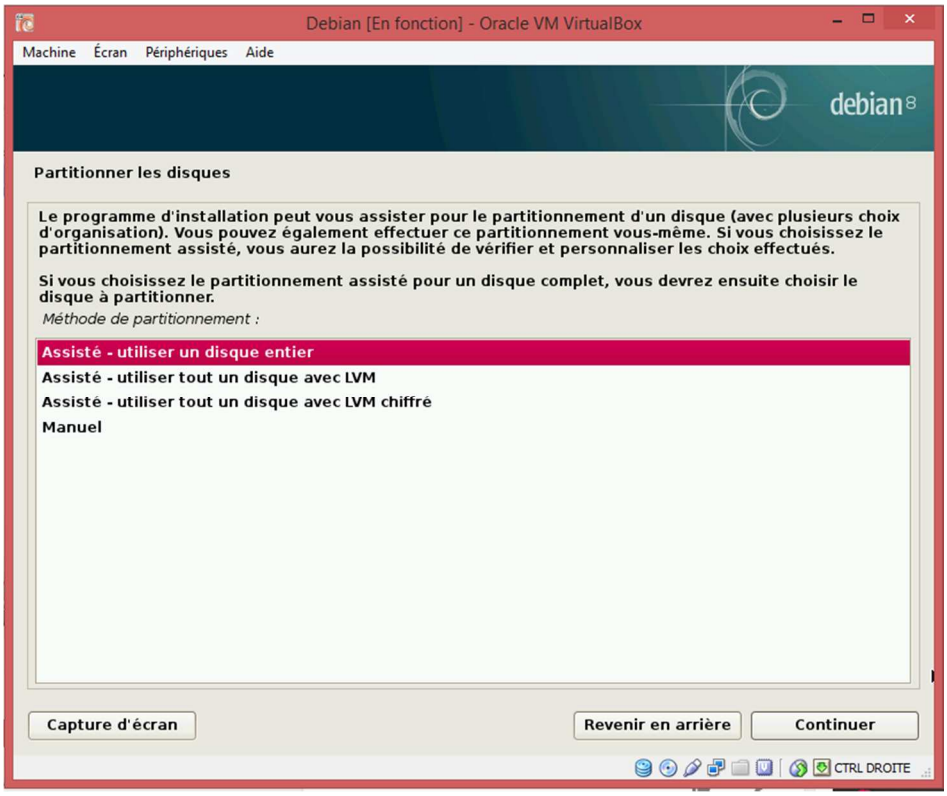
Création de l'utilisateur



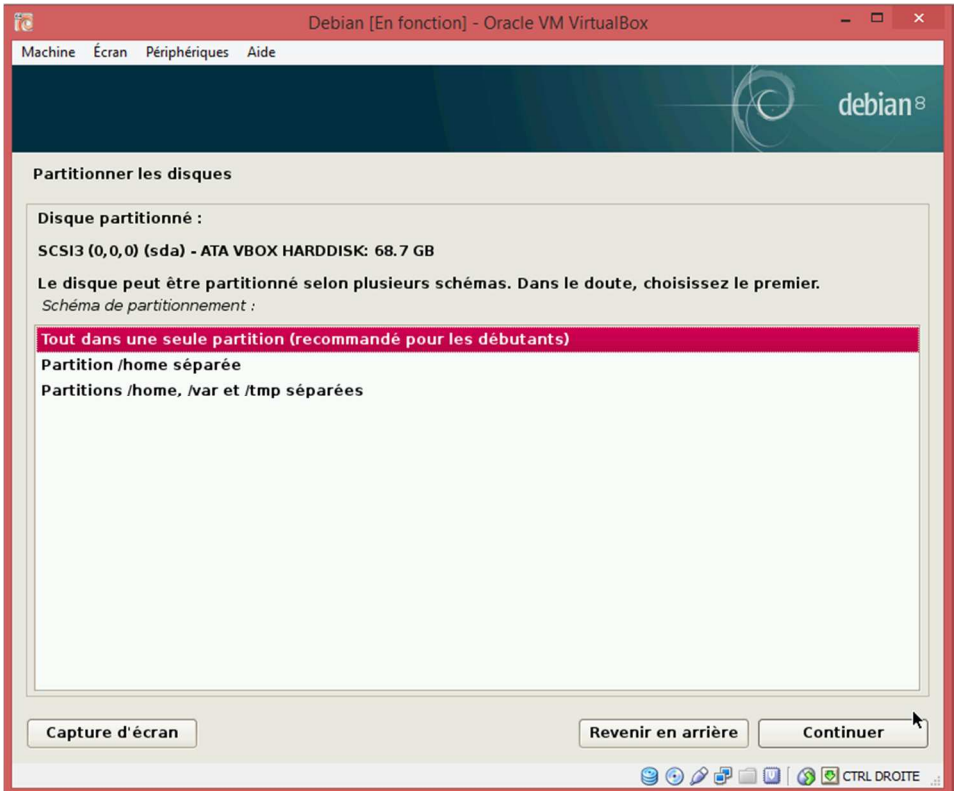
Choix des mots de passe



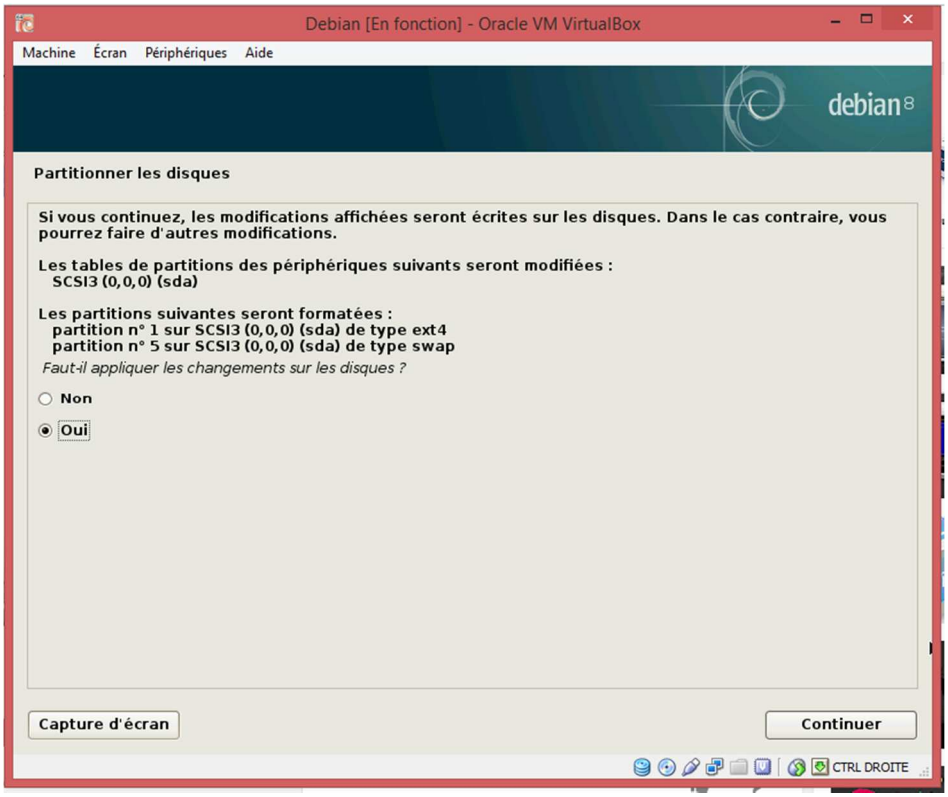
Nous devons choisir comment géré nos partitions, nous sélectionnons assister-utiliser un disque entier.



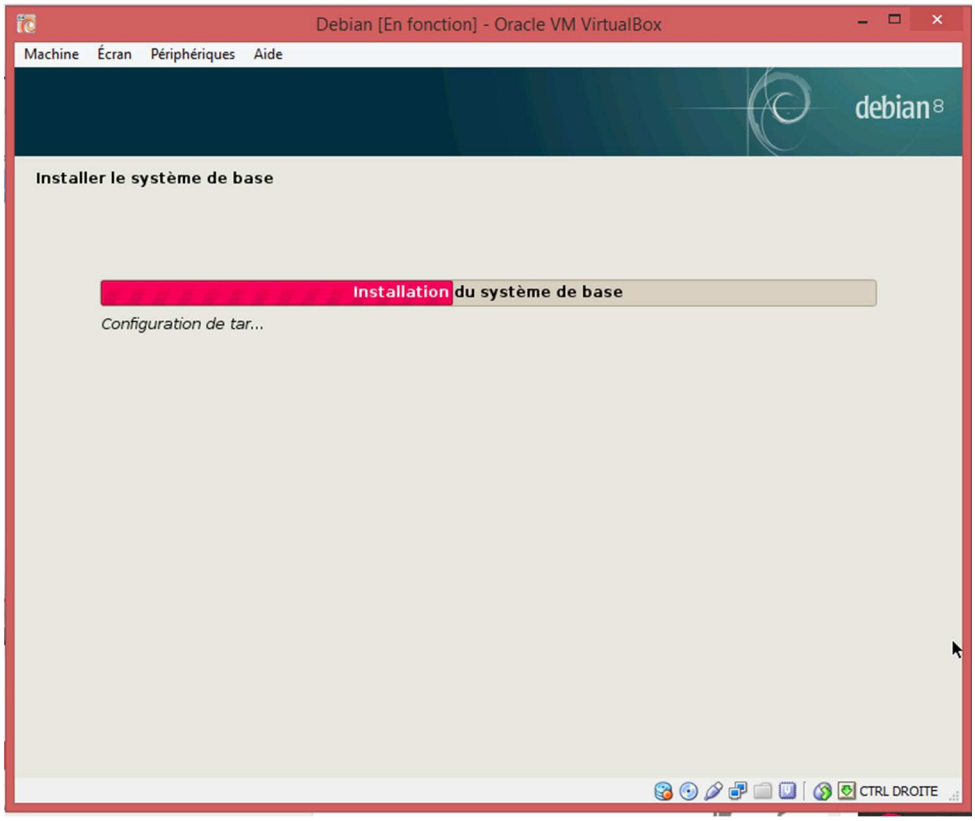
Ici nous confirmons que nous voulons tout mettre dans une seule partition



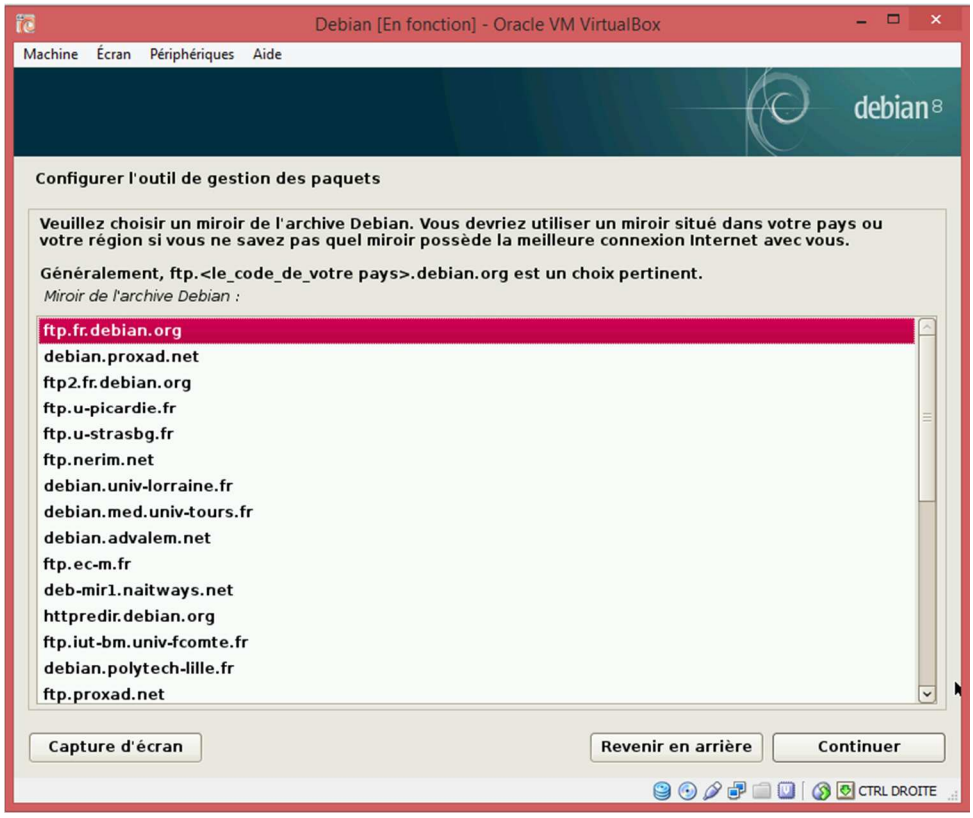
Là le système nous demande si nous voulons vraiment le faire sur un seul disque après avoir opté pour oui, nous continuons.



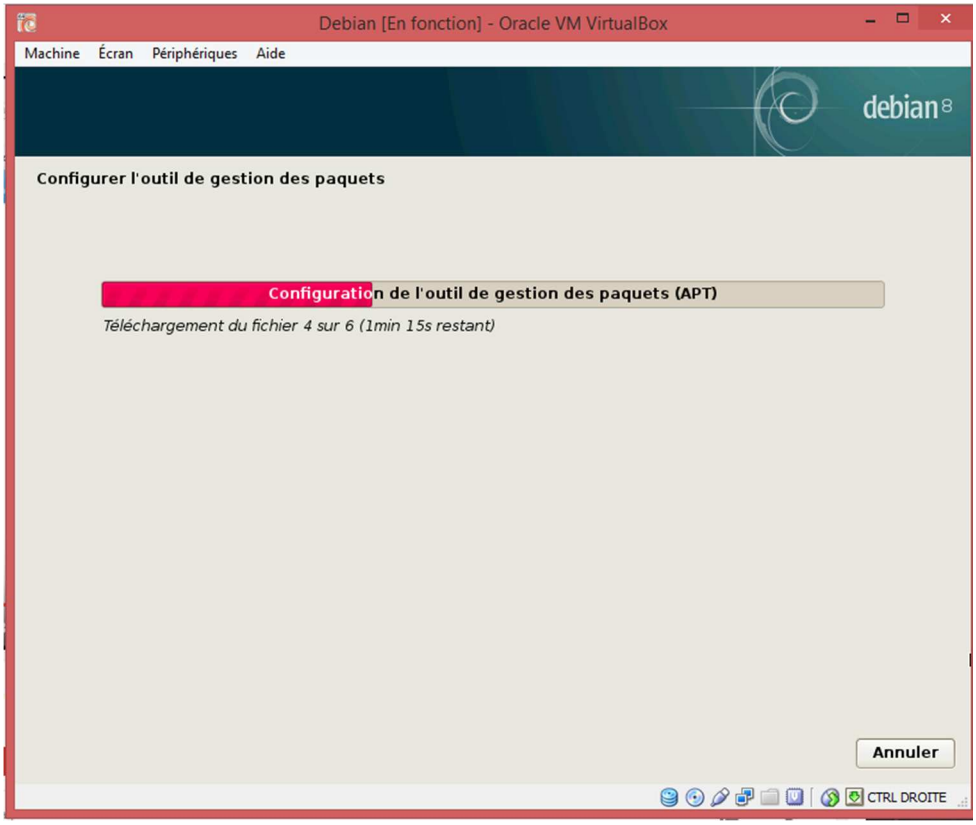
Installation du système de base



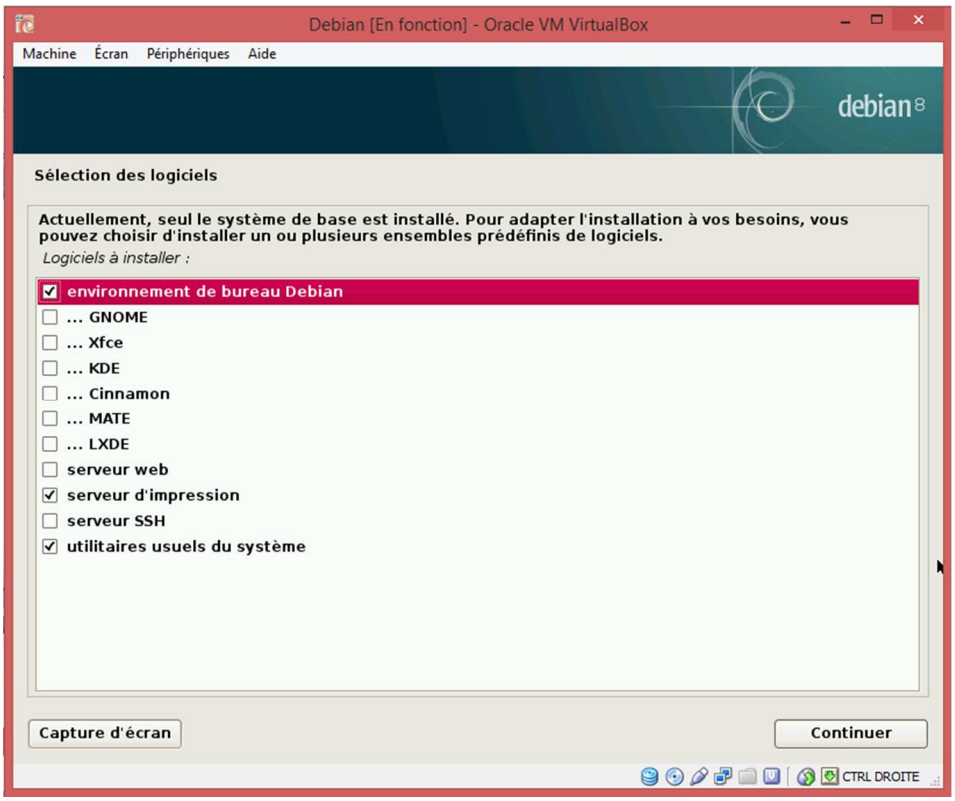
Choix du serveur



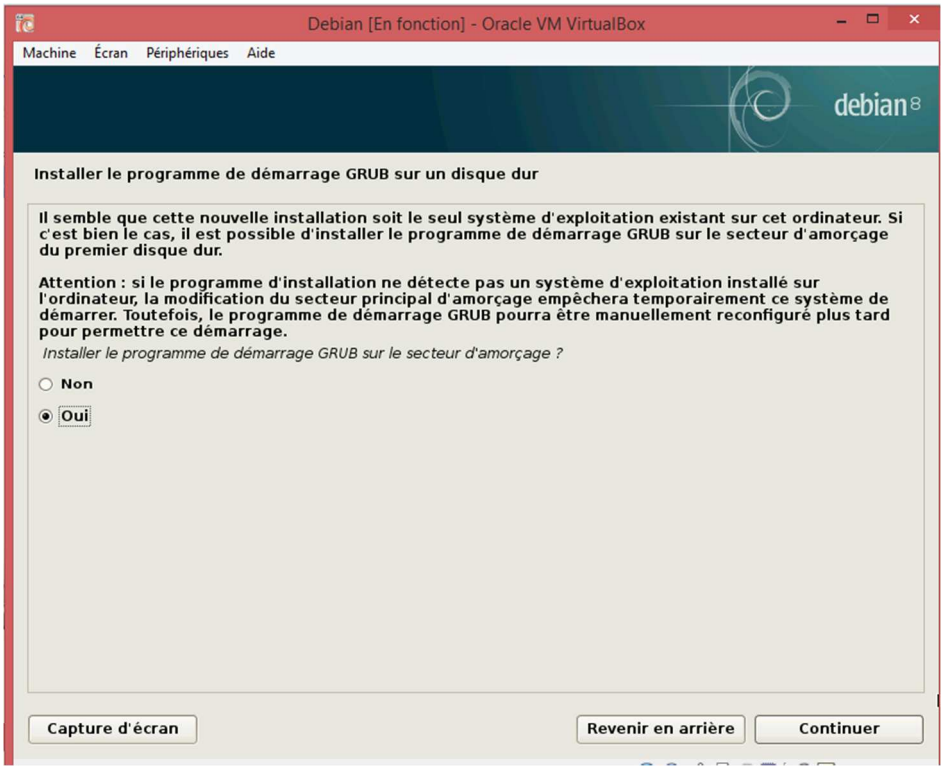
Configuration de l'outil de gestion des paquets



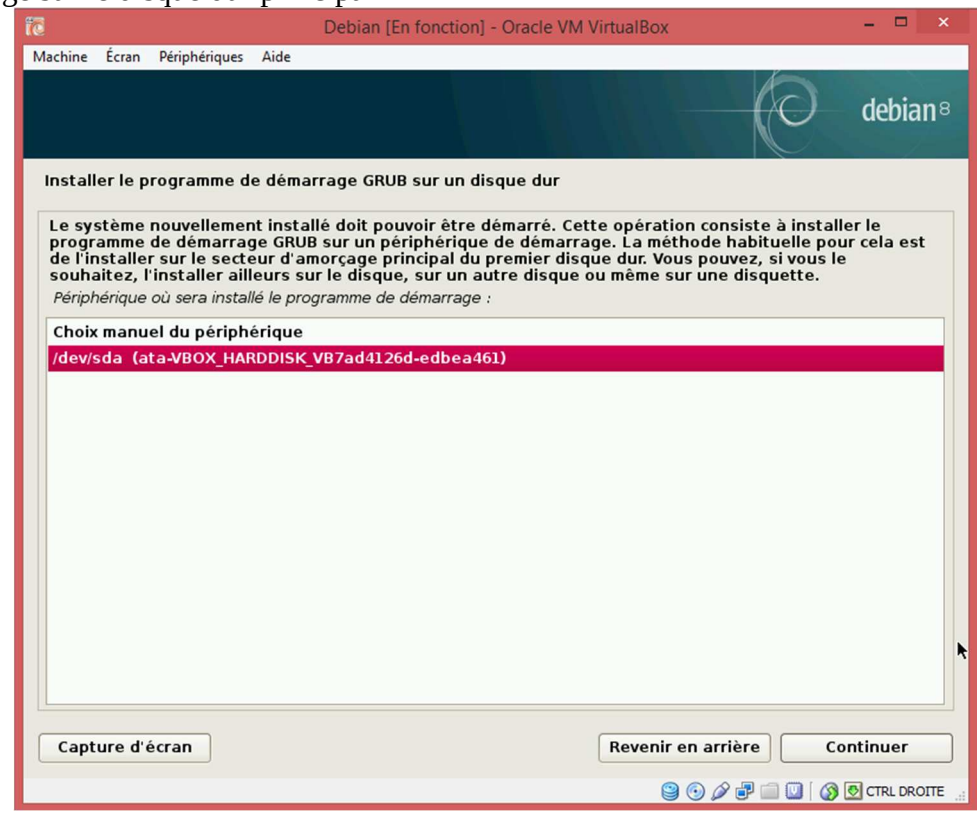
Au niveau de la sélection des outils il y a certaines options sont présélectionnées par défaut ; ensuite nous continuions.



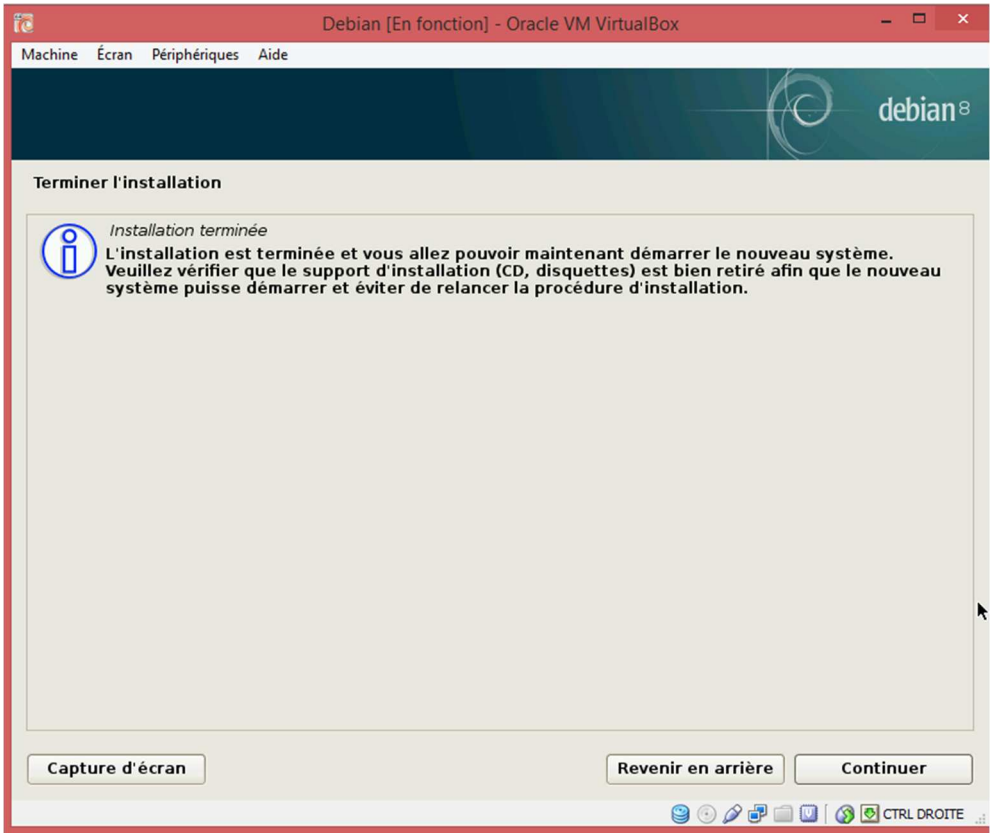
Installation du système de démarrage GRUB



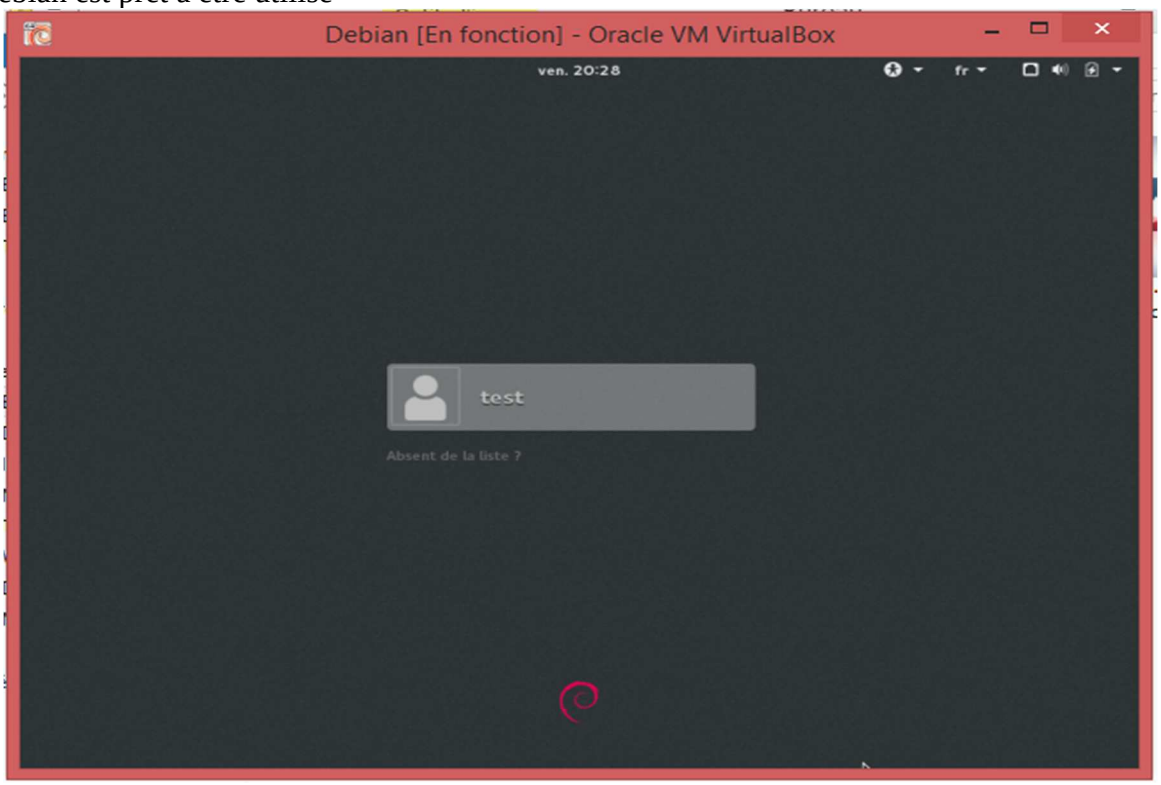
Démarrage sur le disque dur principal



Nous terminons l'installation.

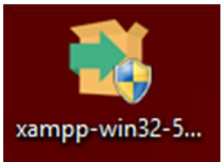


Debian est prêt à être utilisé

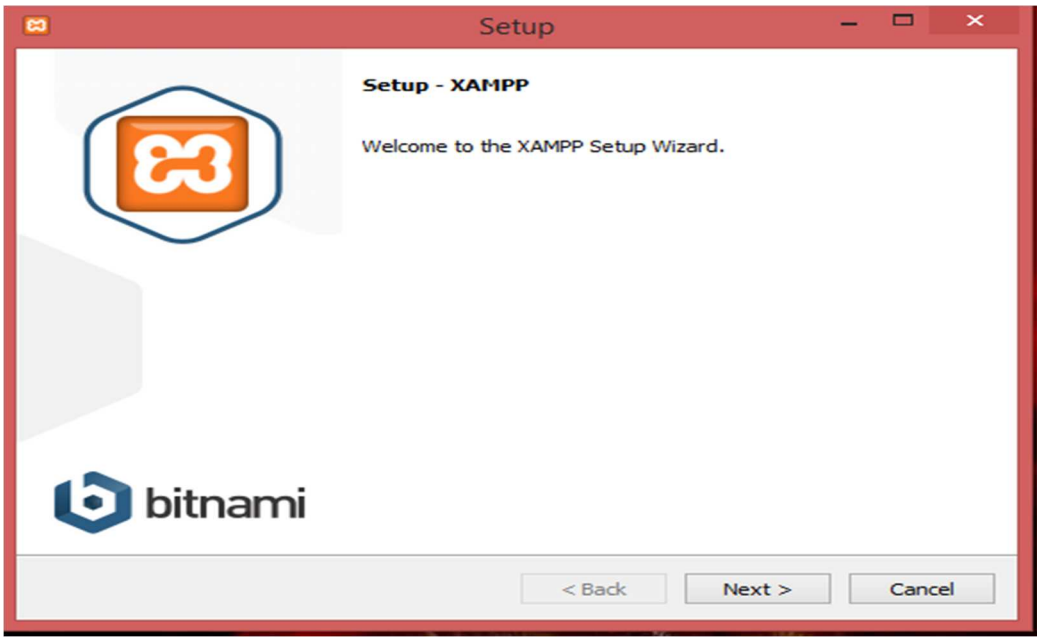


Section 2. Xampp sous Windows

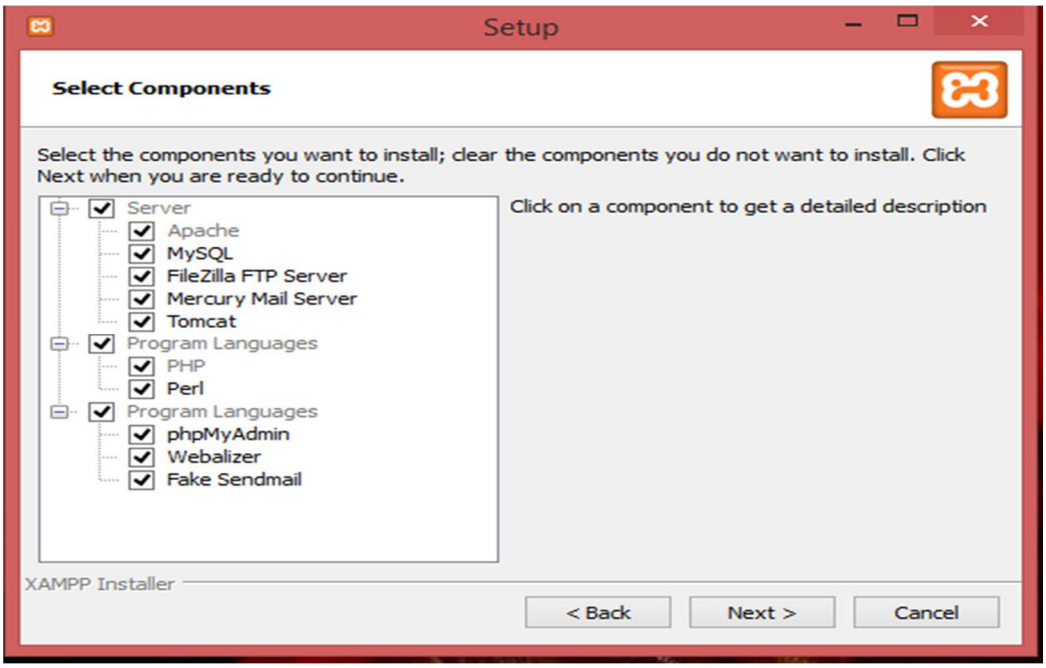
Xampp est une distribution d'apache qui contient également MySQL, PHP et Perl (et bien d'autres). L'avantage de XAMPP est que tout est bien ficelé pour faciliter l'installation et l'utilisation. Double-cliquer sur l'icône du programme pour le lancer.



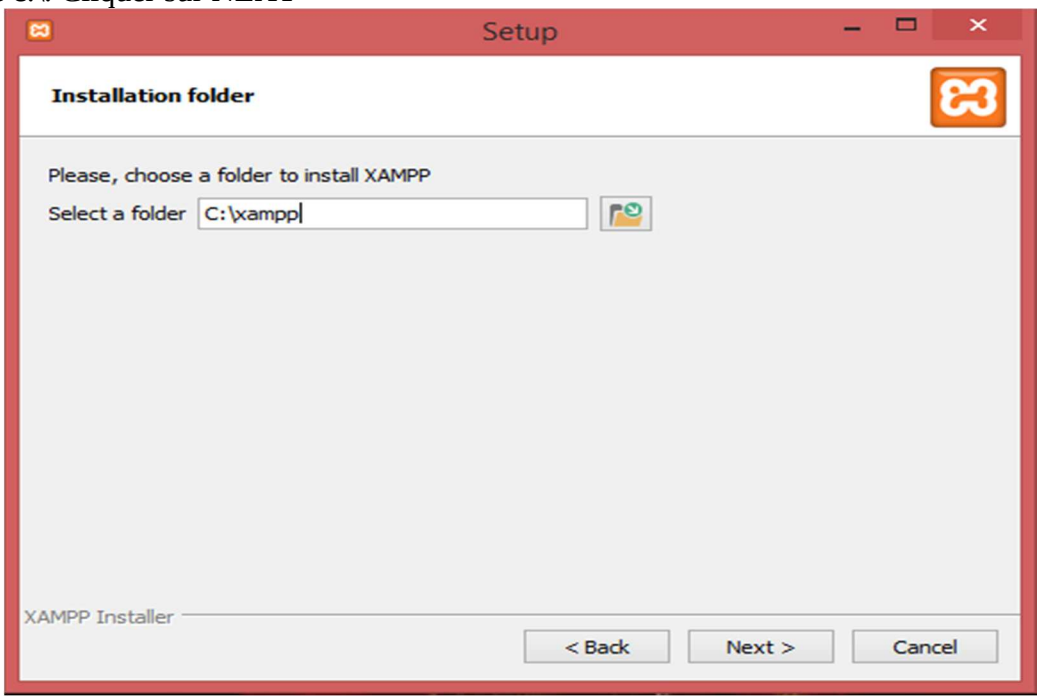
L'écran suivant est un écran d'accueil classique. Cliquer sur le bouton « NEXT »



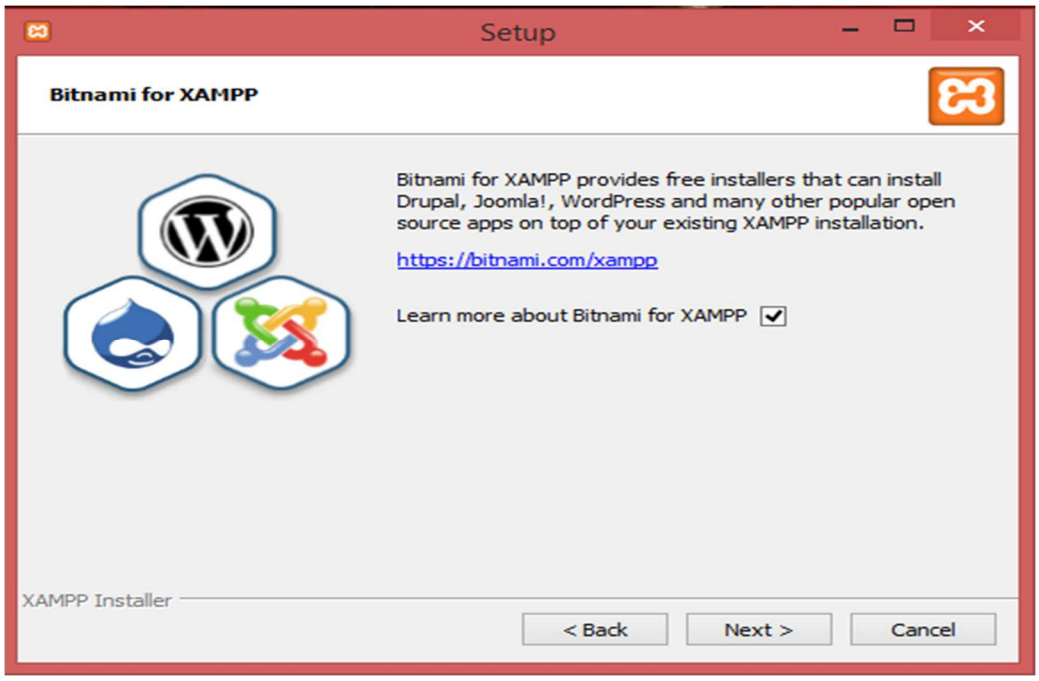
Liste des différents composants : cliquer sur NEXT



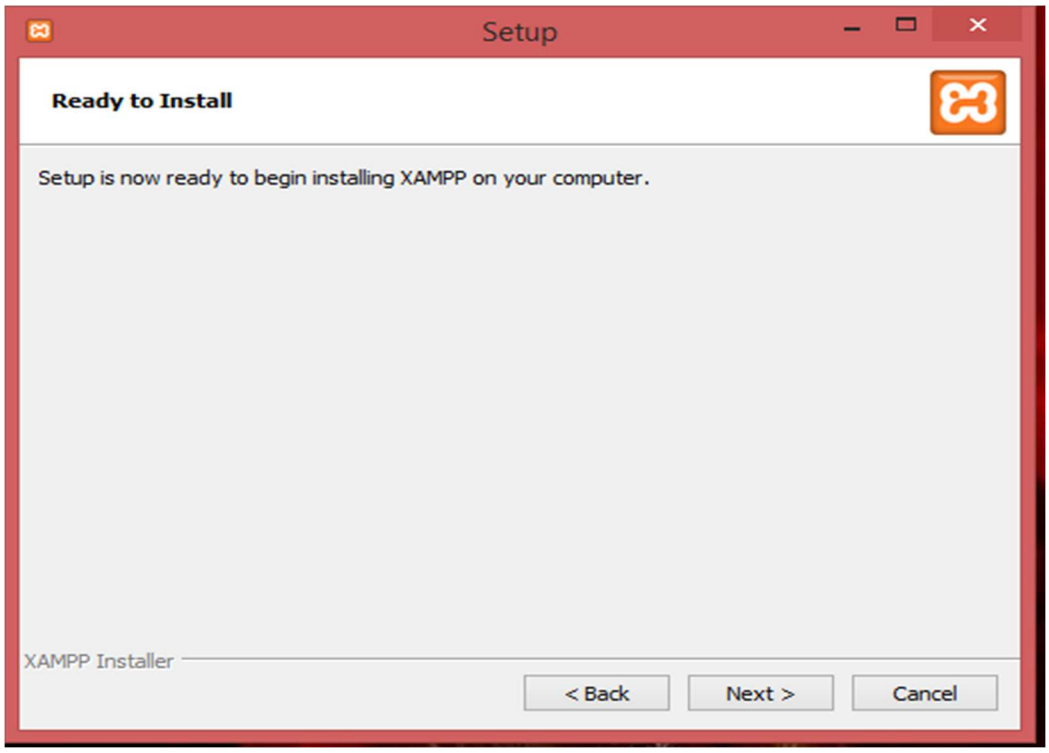
L'installateur demande alors de choisir le chemin d'installation de XAMPP qui est par défaut dans le c:\. Cliquer sur NEXT



Ensuite l'installateur propose d'en savoir plus sur XAMPP. Il faudra cocher la case ou la laisser vide selon l'option choisie.

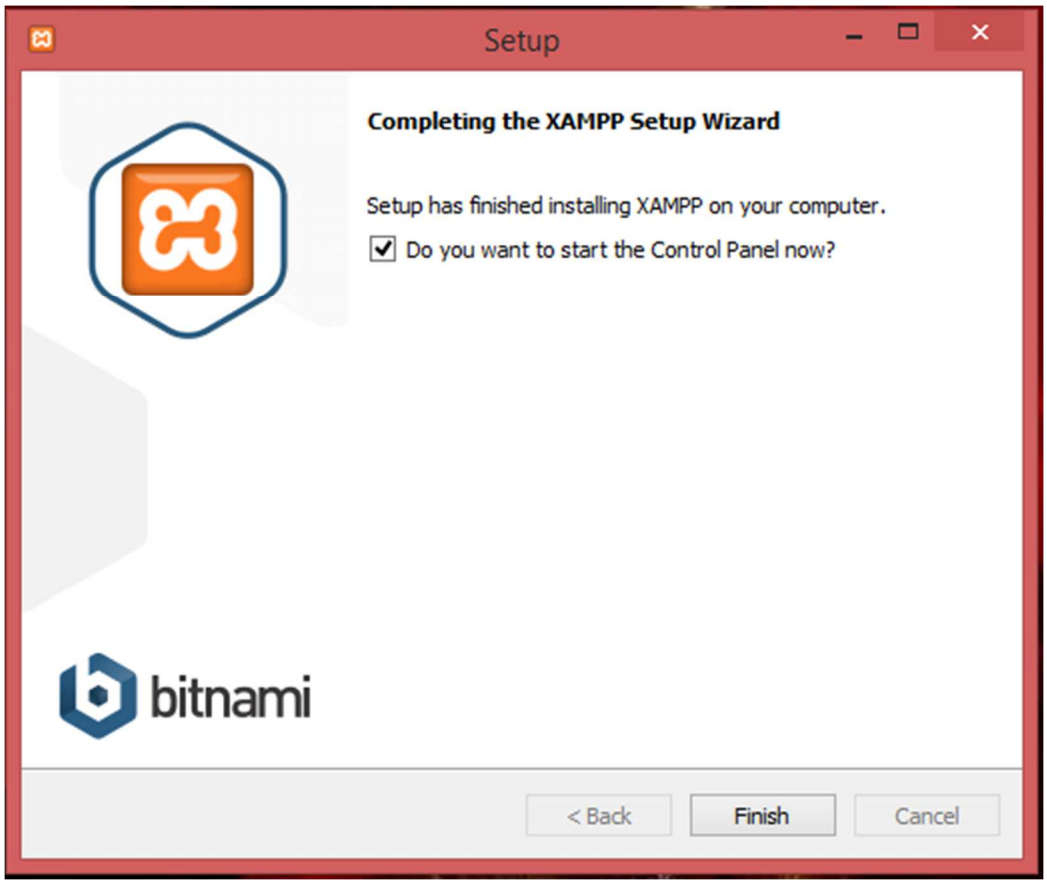


Lancement de l'installation : cliquer sur NEXT



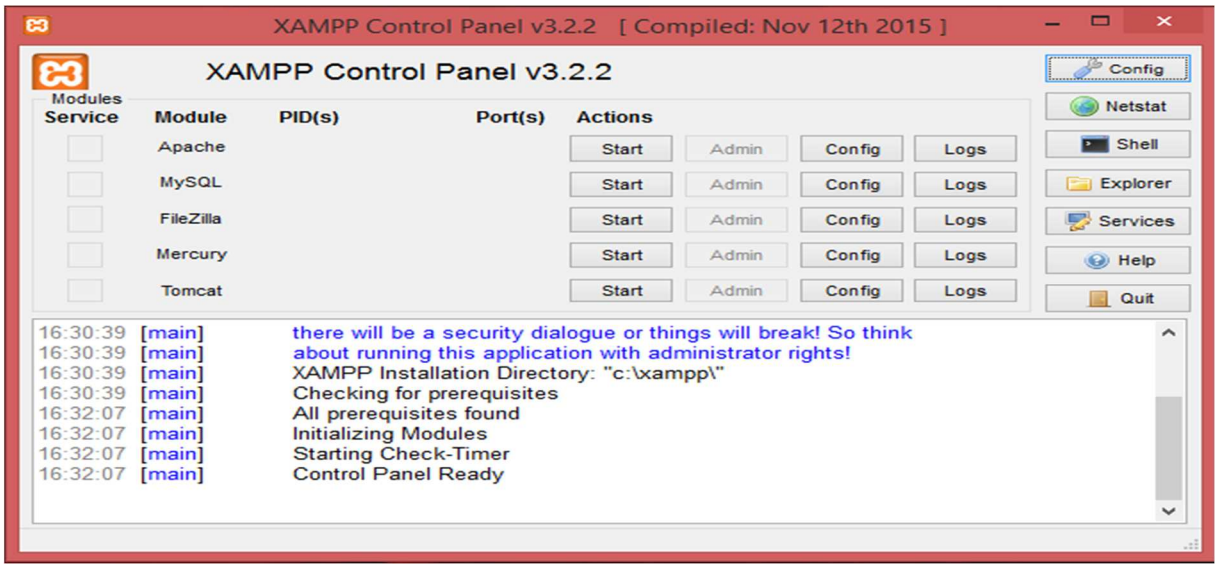
Progression de l'installation de XAMPP

Fin de l'installation : cliquer sur FINISH



Choix de la langue : ici seules deux options se présentent : ANGLAIS et ESPAGNOL

Ensuite le panel de control de XAMPP s'affiche prêt à être utilisé.

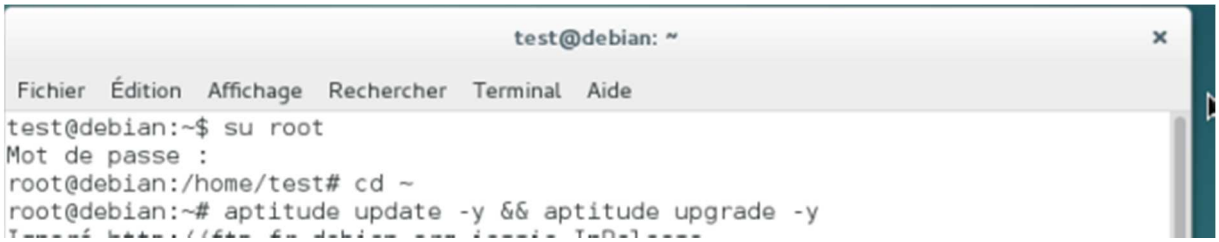


Section 3. MySQL server sous Debian

- **Installation de MySQL Server sous Debian**

Tout d'abord, mettre à jour les serveurs à l'aide du gestionnaire de paquets choisi (ici, aptitude).

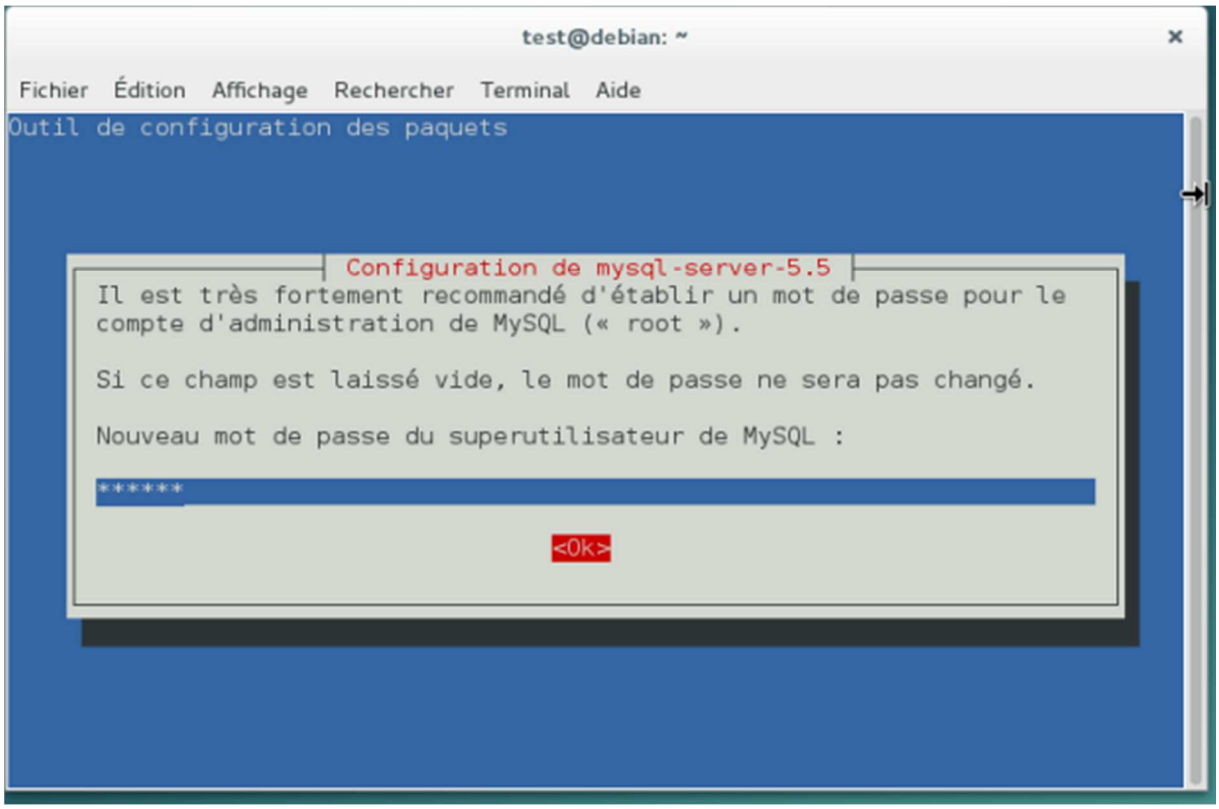
```
root@debian:~# aptitude update –y && aptitude upgrade –y
```

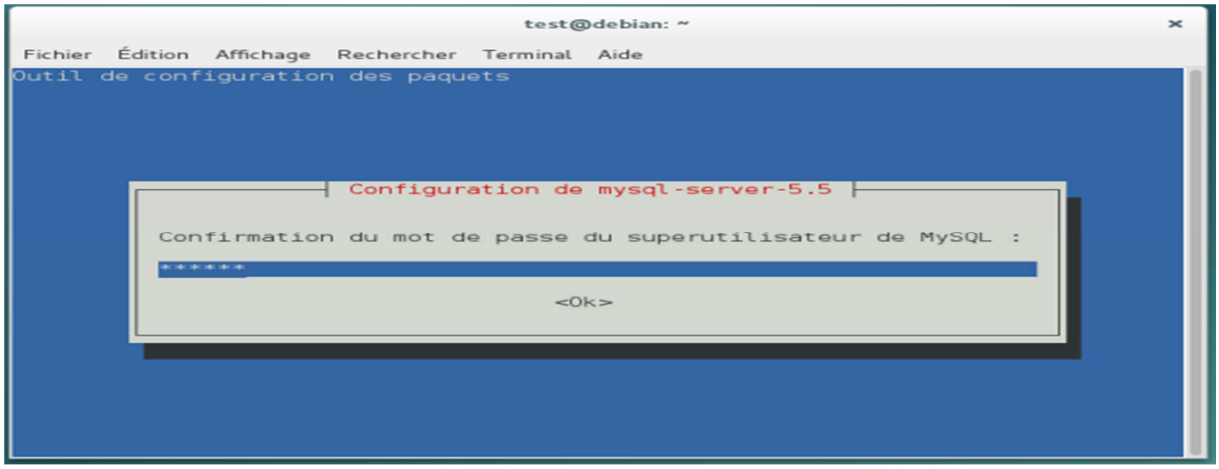


Ensuite nous installons le service MySQL en tant que serveur à l'aide de la commande ci-dessous

```
root@debian:~# aptitude installmysql-server –y
```

Le mot de passe de l'administrateur de la base de données (root) sera demandéIl faudra le saisir à deux reprises pour le confirmer





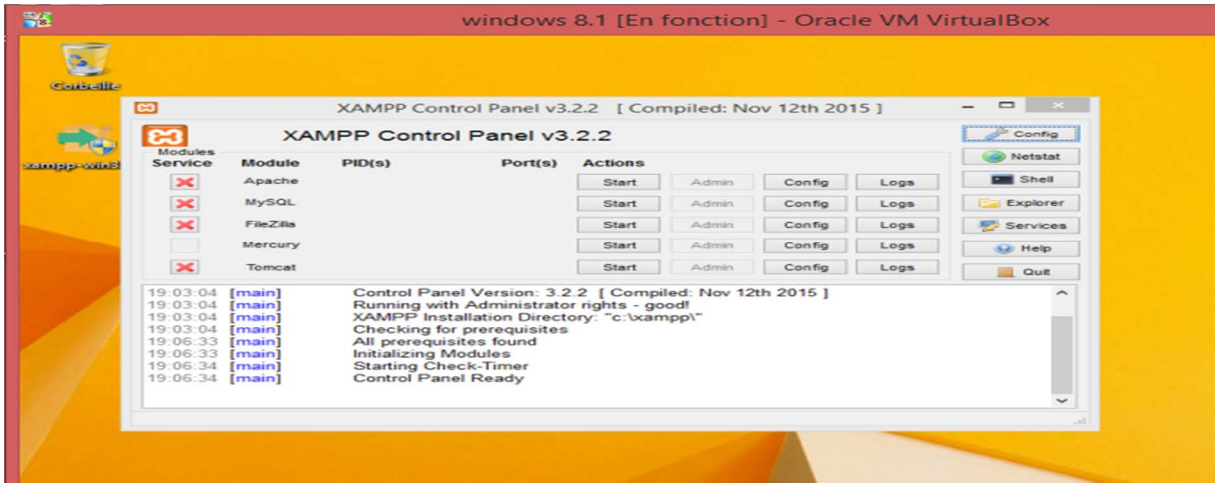
Bien entendu, cette opération devra être répétée sur chacun des deux serveurs pour que les serveurs MySQL soient fonctionnels.

Chapitre 2 : Mise en œuvre de la réplication synchrone symétrique en virtuel

Section 1. Sous Windows en virtuel

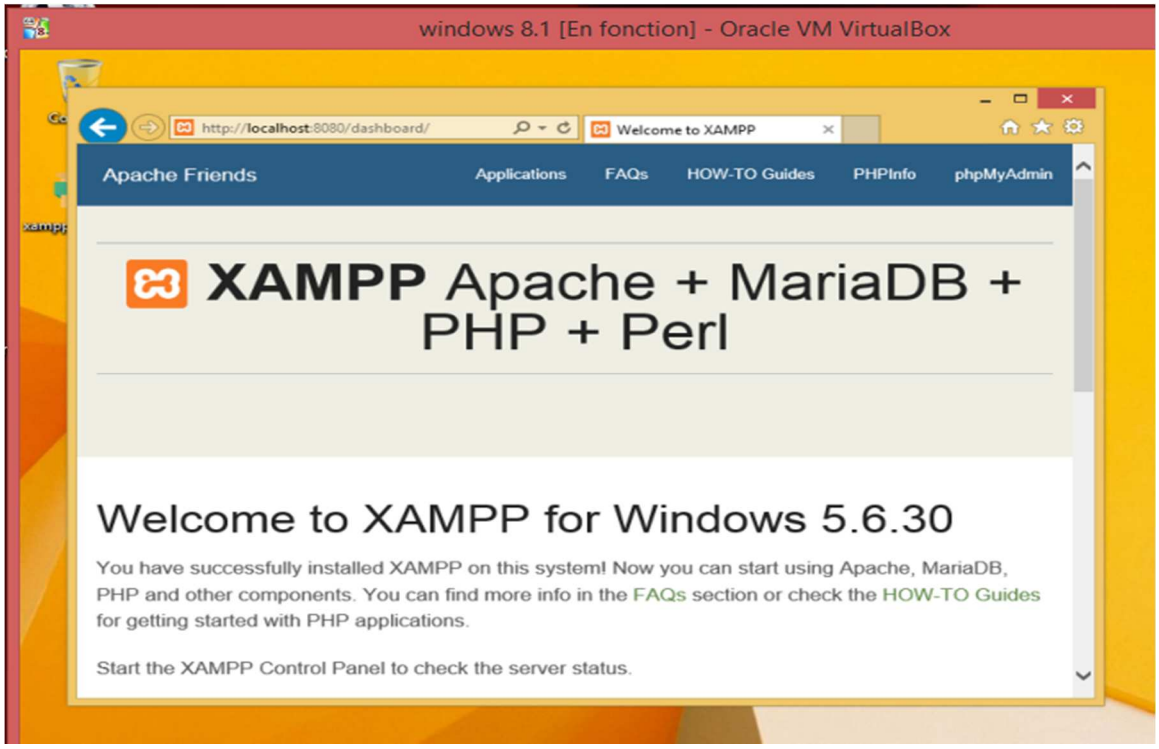
Réplication synchrone et symétrique sous Windows (phpMyAdmin)

Tout d'abord il faudrait télécharger et installer un petit programme appelé (XAMPP)

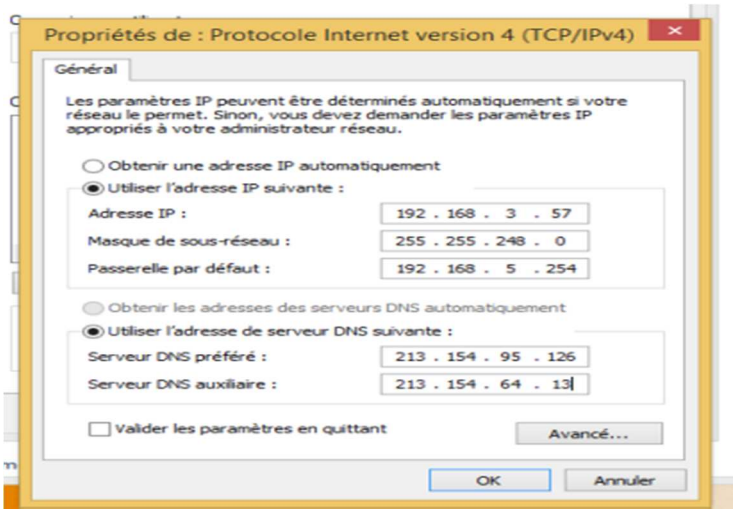


Ici nous utiliserons comme adresse IPV4 pour nos deux serveurs les adresses IP suivantes :Serveur 1 : 192.168.3.57 et Serveur 2 : 192.168.3.58

Pour accéder à l'outil phpMyAdmin il suffit de cliquer sur le bouton (Admin) qui se trouve juste après le bouton (Start) sur la ligne de apache dans le xampp control panel ainsi nous verrons l'onglet phpMyAdmin dans le coin supérieur droit comme ceci :



Configuration des adresses IP

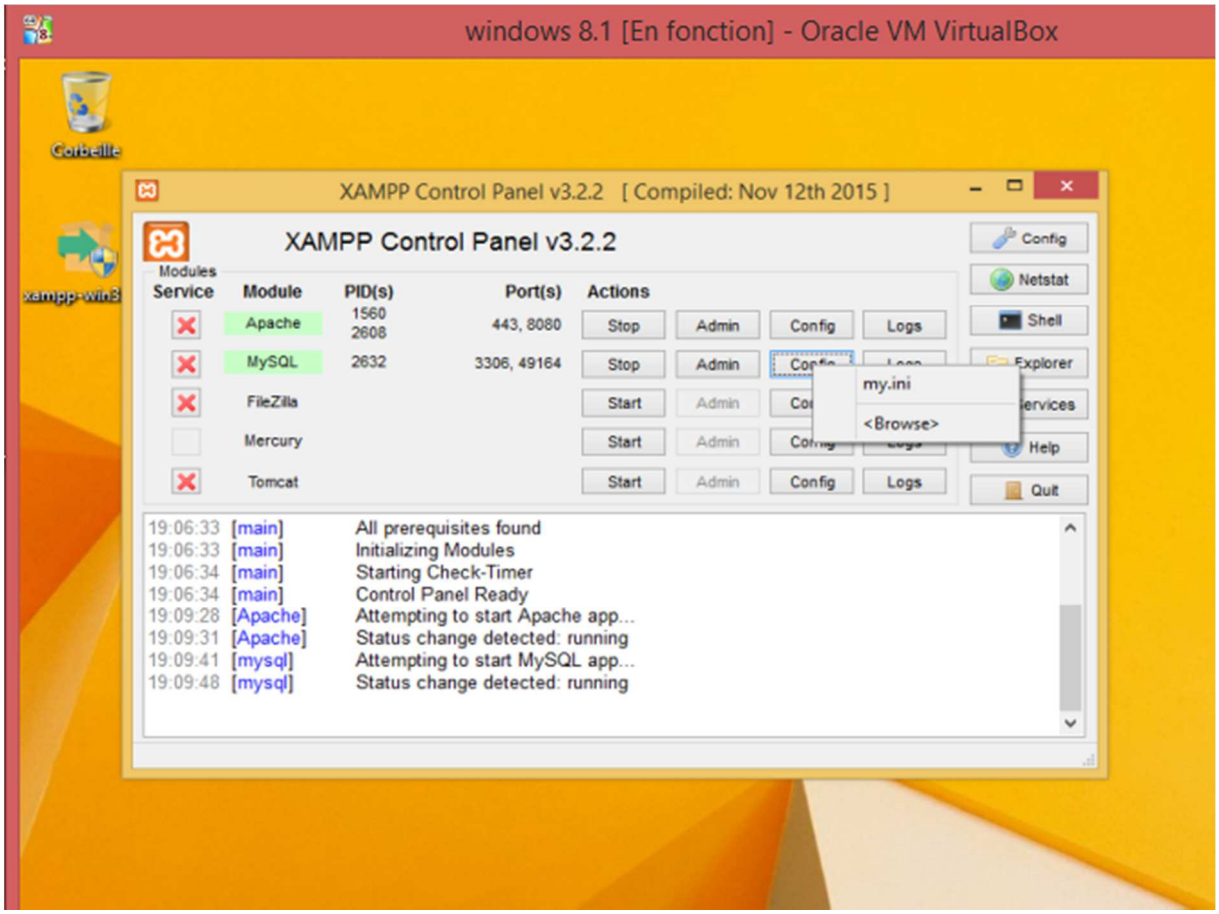


Une fois les adresses IP configurées il faut faire un Ping pour savoir si la communication entre les deux machines s'effectue. Si le Ping ne passe pas alors il faudra arrêter les par feu Windows et recommencer les Ping

Bien entendu, cette opération devra être répétée sur chacune des deux machines.

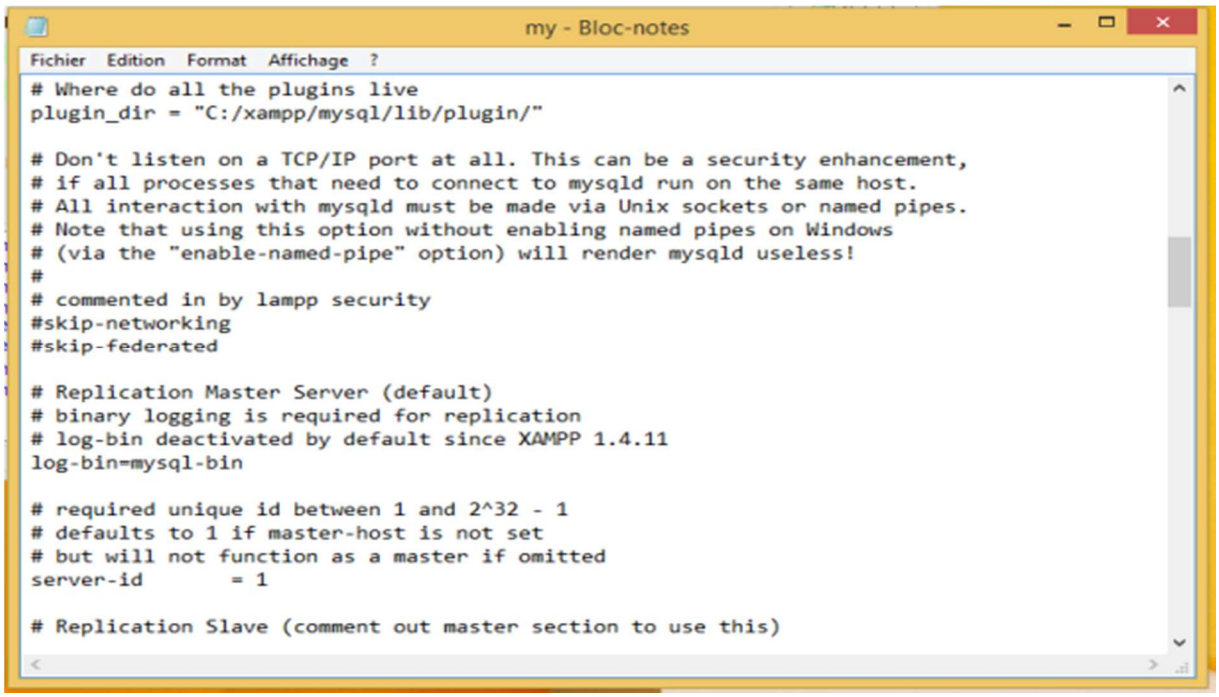
Configuration de MySQL

Nous allons maintenant démarrer la configuration initiale dont MySQL doit faire l'objet pour préparer celui-ci à accepter la réplication. Sur le premier serveur, il faudra éditer le fichier my.ini. Pour cela il suffit de cliquer sur le bouton (config) ensuite cliquer sur my.ini, qui se trouve juste sur la ligne MySQL qui se trouve sur le xampp control panel comme ceci.



Une fois dans le fichier il suffira de de-commenter les lignes

- Log-bin=mysql-bin
- Server-id = 1



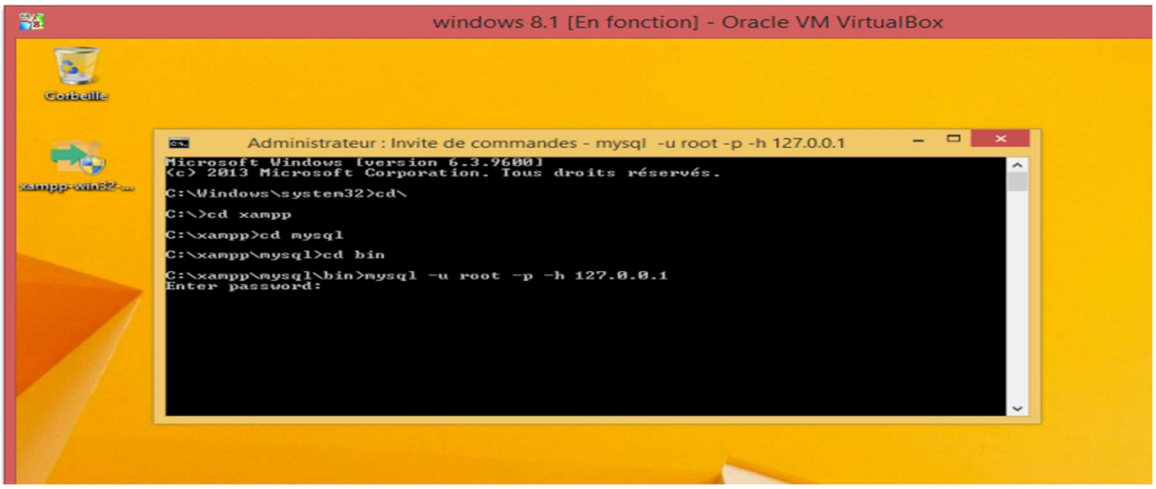
Cette manipulation devra être effectuée sur chacun des deux serveurs MySQL.

Puis il faudra redémarrer ensuite le service pour appliquer les modifications. Pour ce faire il faut cliquer sur le bouton (start) dans le xampp control panel sur la ligne de mysql.

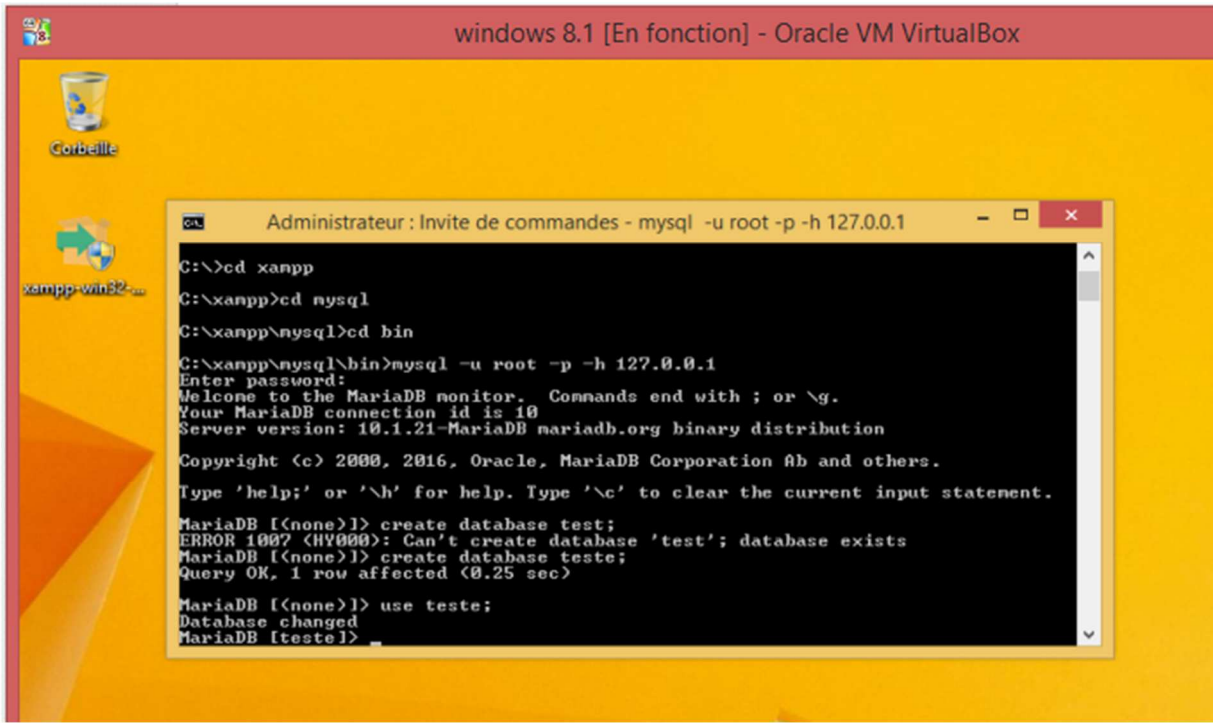
- **Préparation de l'environnement pour la réplication MySQL**

L'étape qui va suivre est à réaliser sur les deux serveurs. Voici comment se connecter au Shell MySQL (MariaDB) :

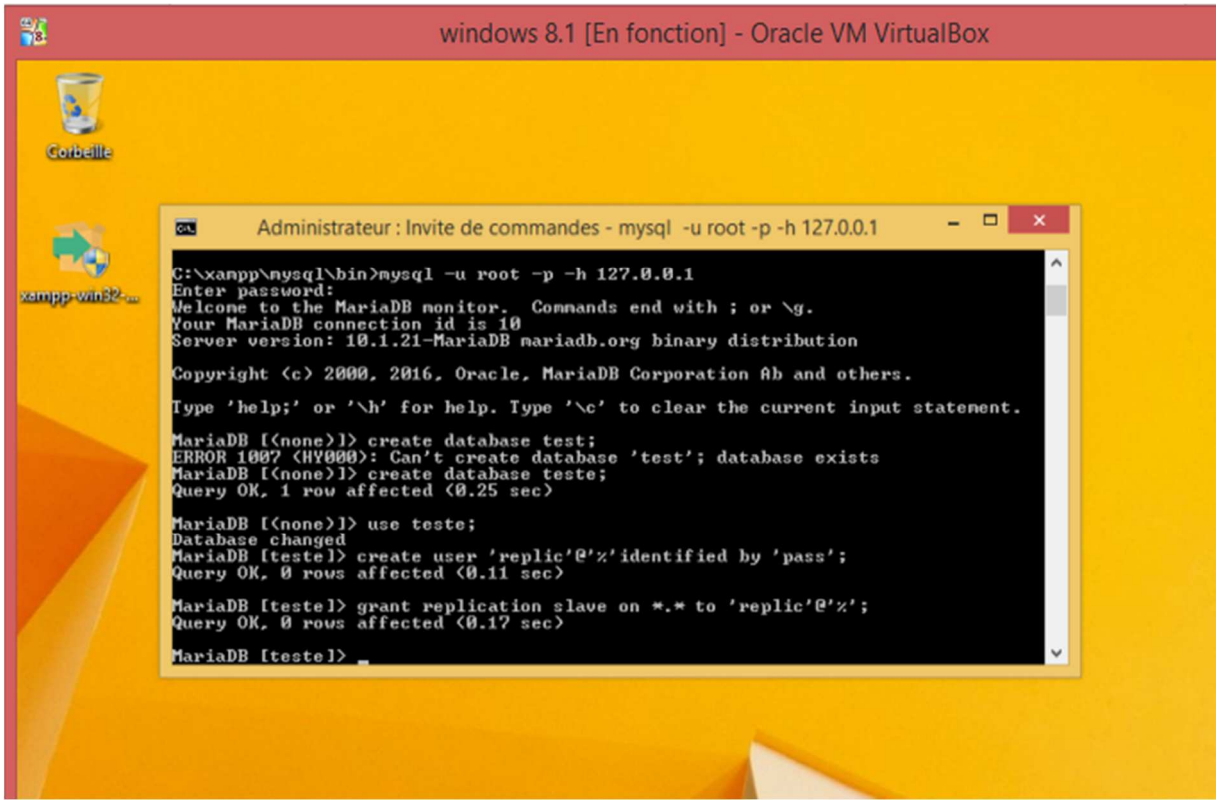
Sur le premier serveur :



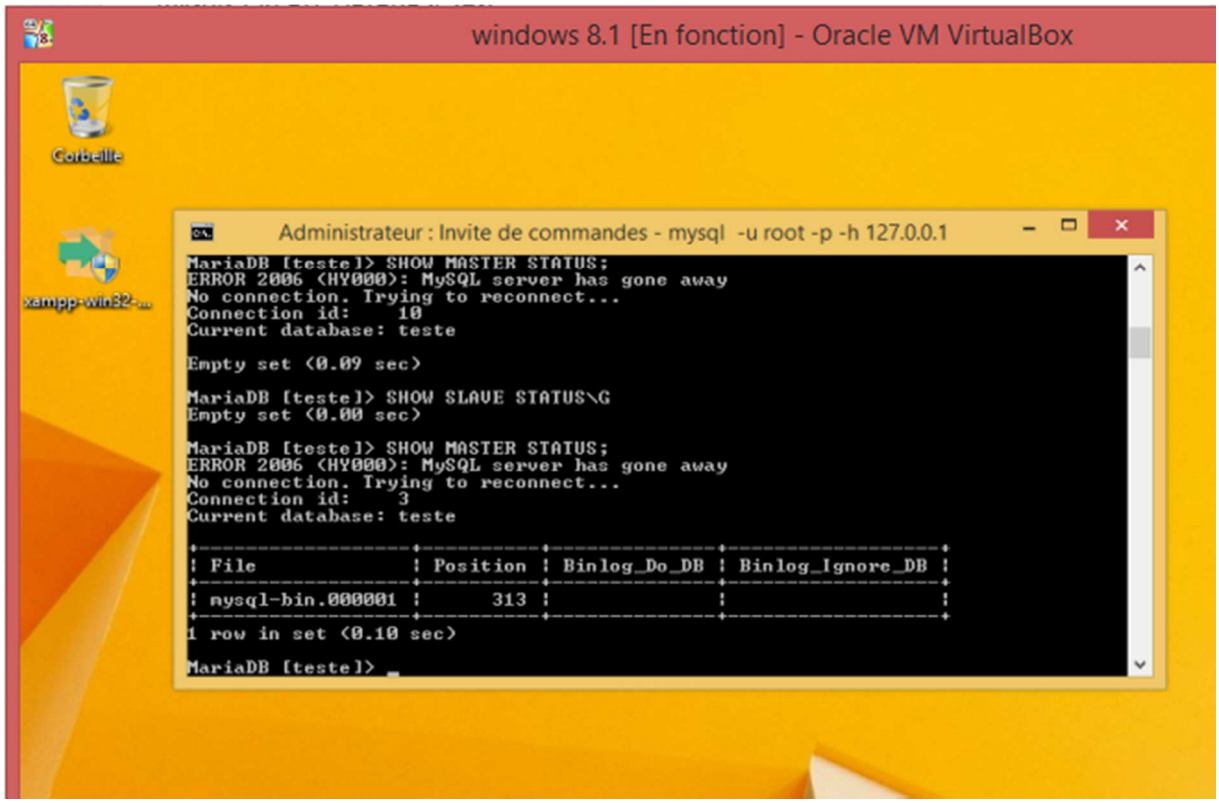
Création de la base de données « teste » et positionnement sur elle.



Création d'un utilisateur qui sera dédié à la réplication et à l'ajout des données dans les bases de données qui seront répliquées. L'utilisateur sera nommé « 'replic'@'%' sur les deux serveurs MySQL. Et on lui attribuera aussi les privilèges de réplication.

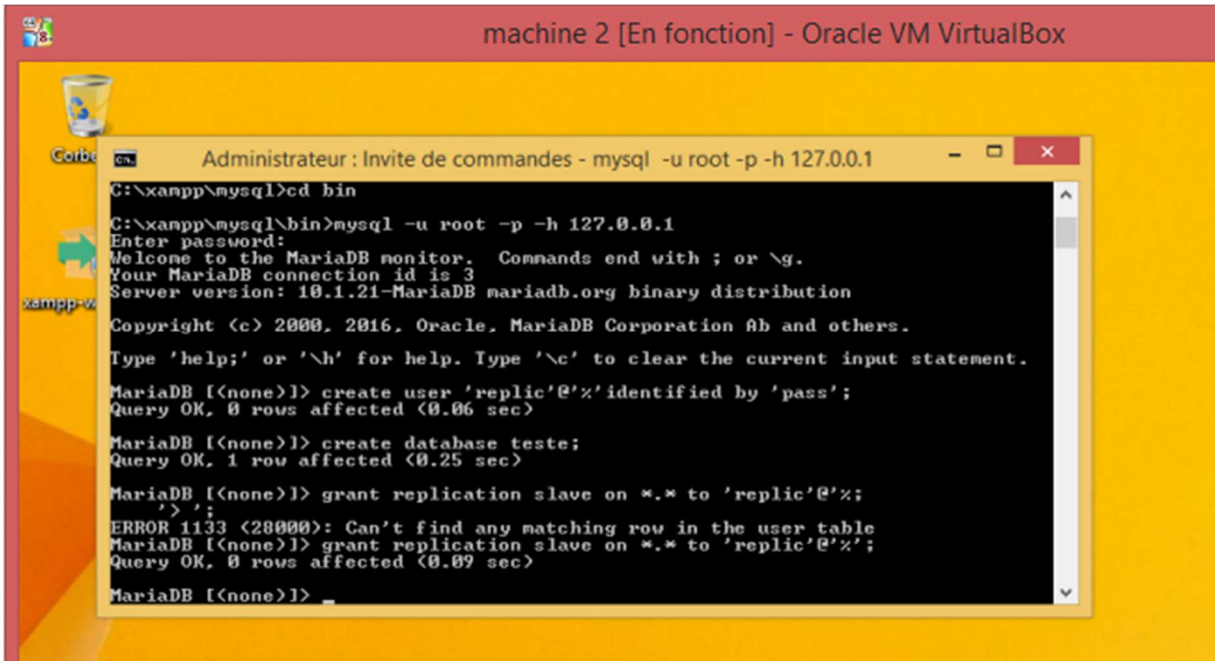


Ensuite on exécute la commande « SHOW MASTER STATUS » le résultat de cette commande doit être mémorisé car il sera réutilisé plus bas.

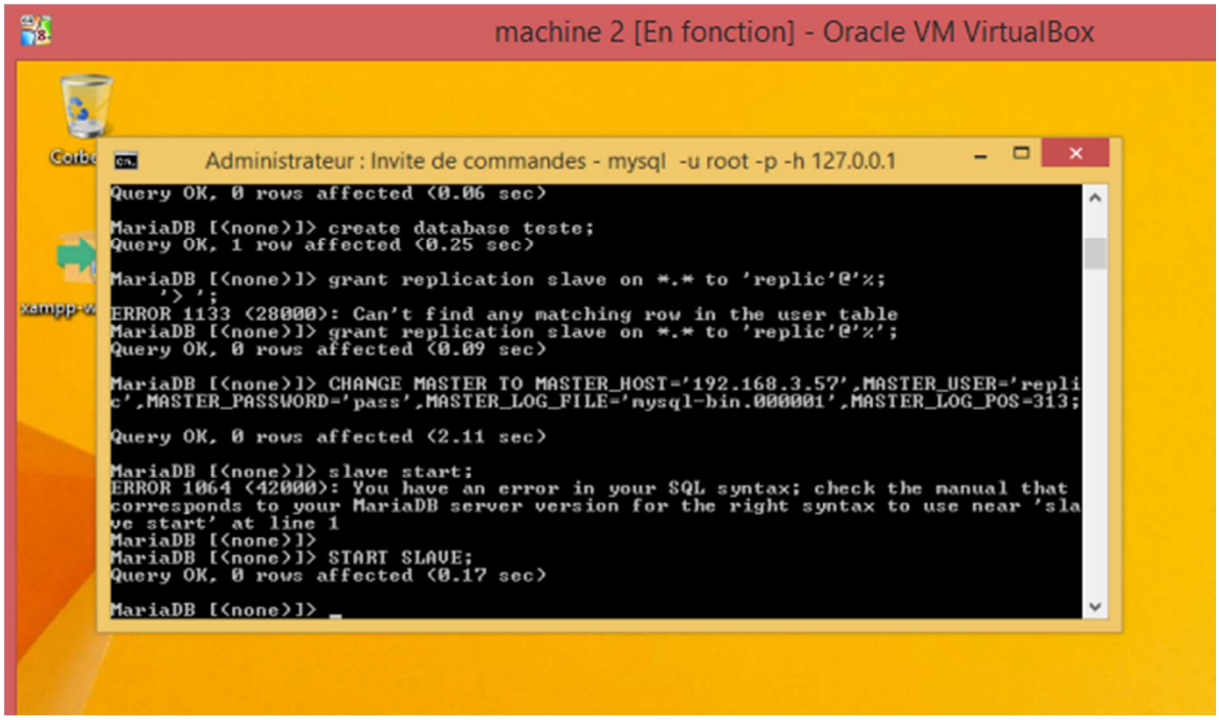


Sur le deuxième serveur :

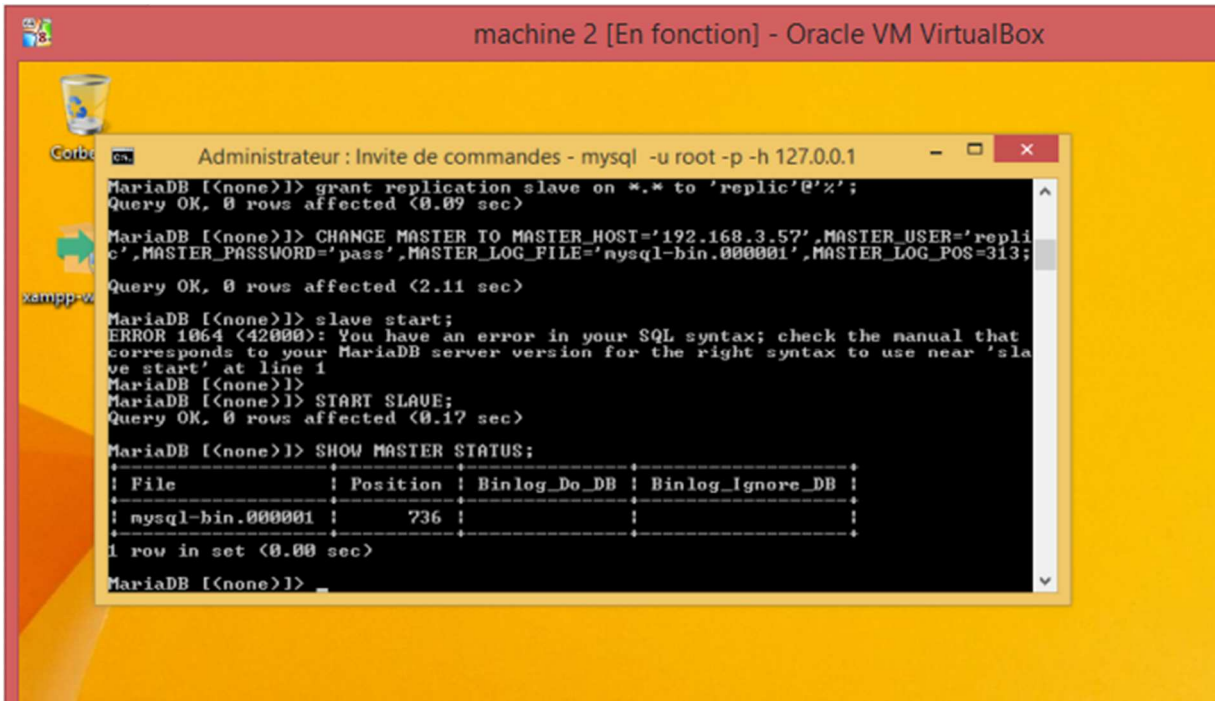
Création de l'utilisateur qui sera dédié à la réplication comme sur le premier serveur, création en même temps de la base de données « teste ». Ensuite on lui attribuera le privilège de réplication.



Exécution de la requête suivante en prenant soin de modifier les paramètres. Les valeurs des paramètres « MASTER_LOG_FILE » et « MASTER_LOG_POS » correspondront aux valeurs récupérées précédemment. Ensuite nous ferons un « START SLAVE » pour démarrer le serveur en tant qu'esclave.

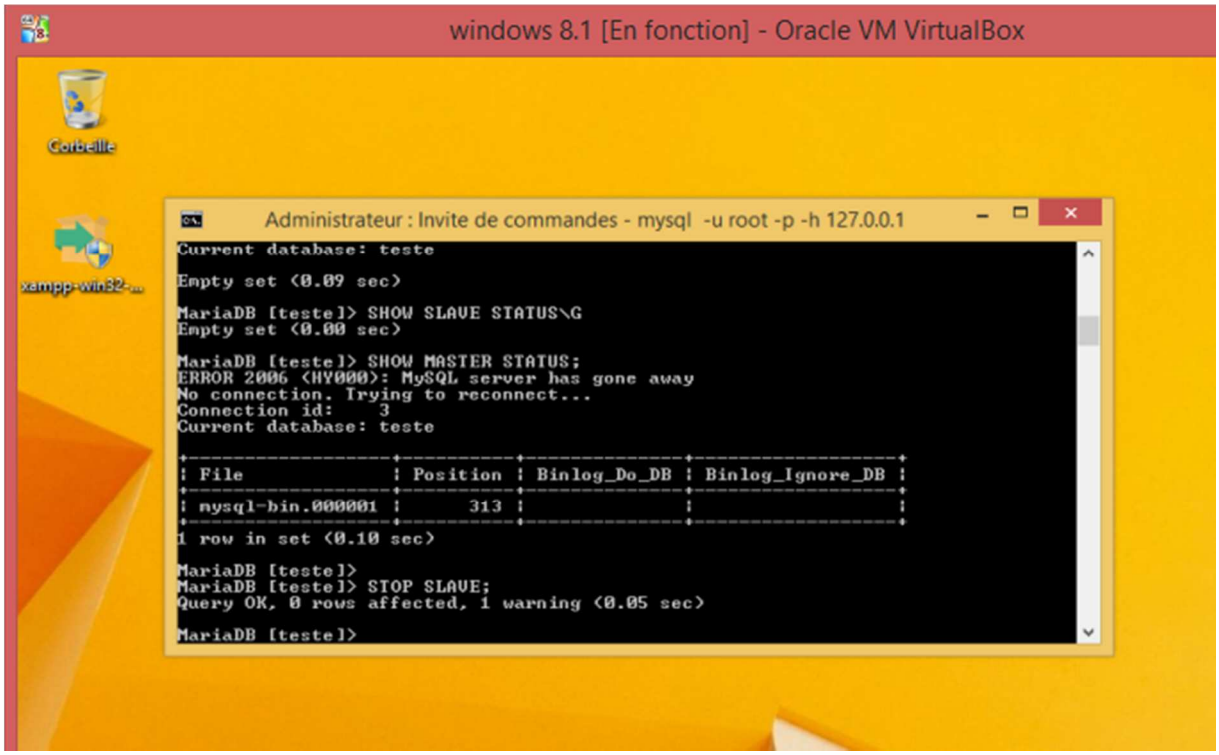


Exécutons maintenant la requête « SHOW MASTER STATUS »



Sur le premier serveur

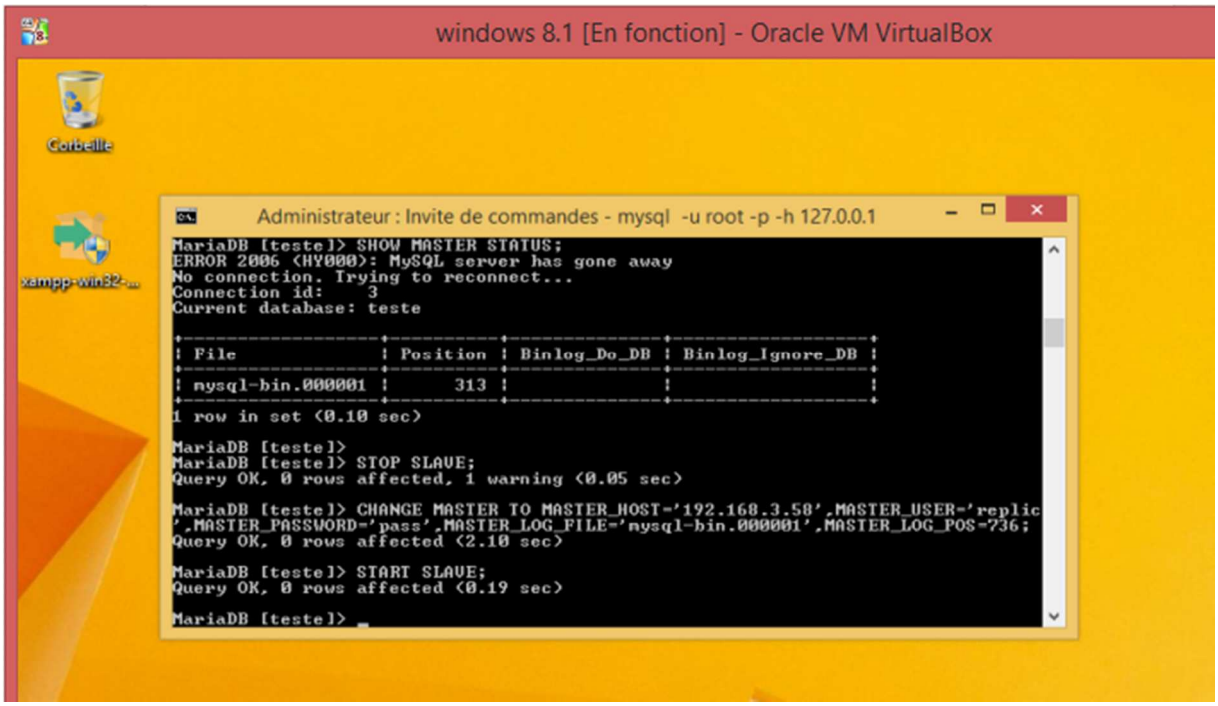
Faire un « STOP SLAVE » pour arrêter le serveur en tant qu'esclave.



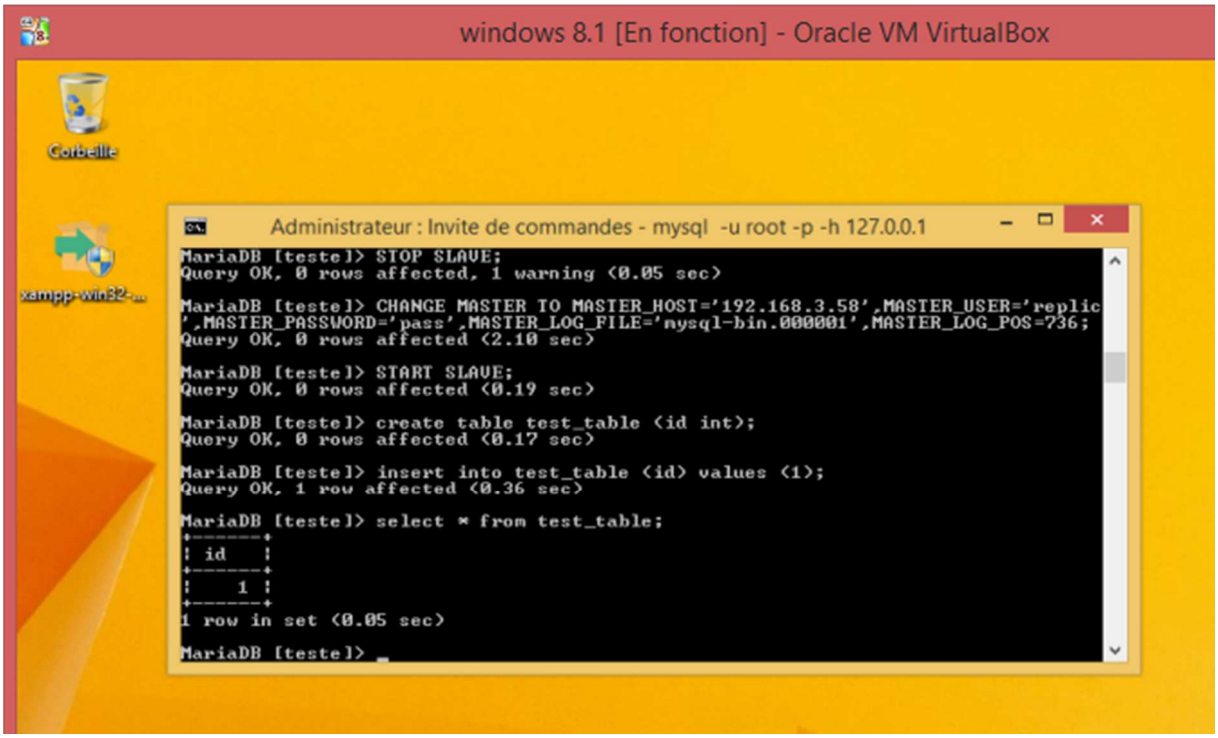
Ensuite nous exécutons la requête suivante :

MariaDB [teste]> CHANGE MASTER TO MASTER_LOG_HOST='192.168.43.27', MASTER_USER='replic', MASTER_PASSORD='pass', MASTER_LOG_FILE='mysql-bin.000001', MASTER_LOG_POS=737;

Ensuite « START SLAVE » pour démarrer le serveur en tant qu'esclave.

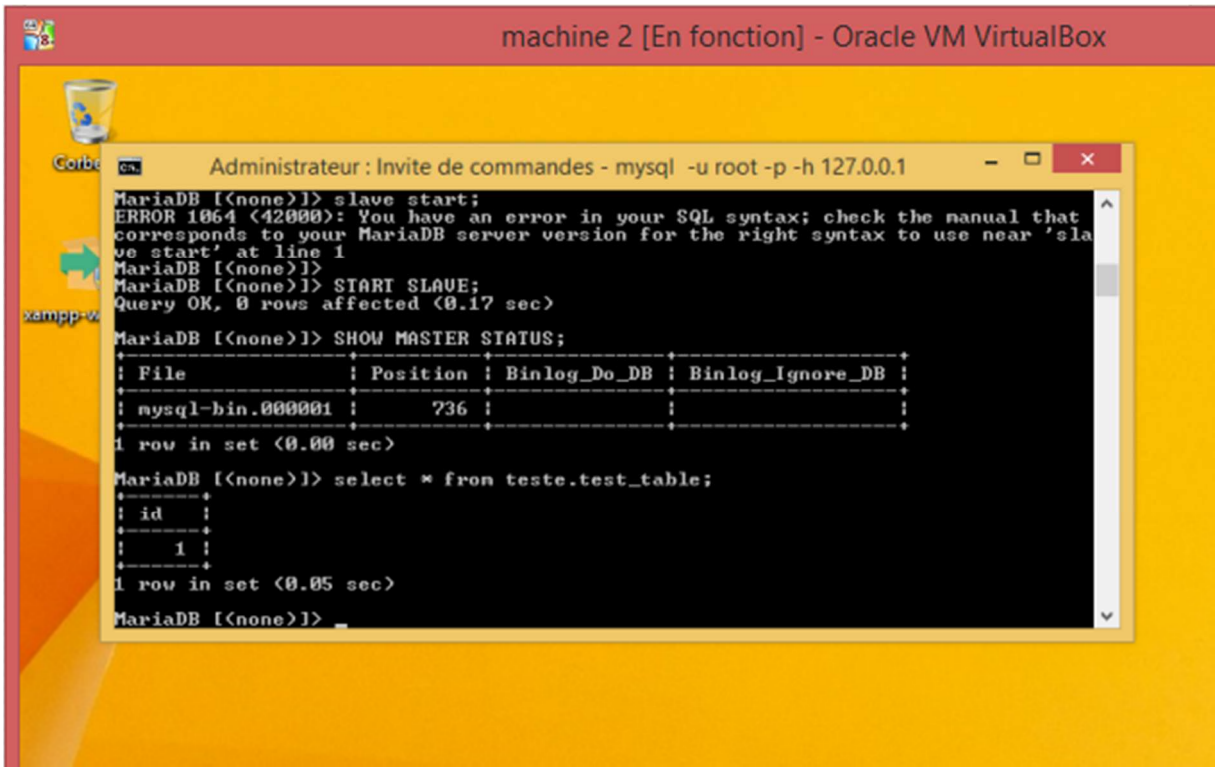


Création de la table « test_table » puis insertion de valeur et un select sur cette table



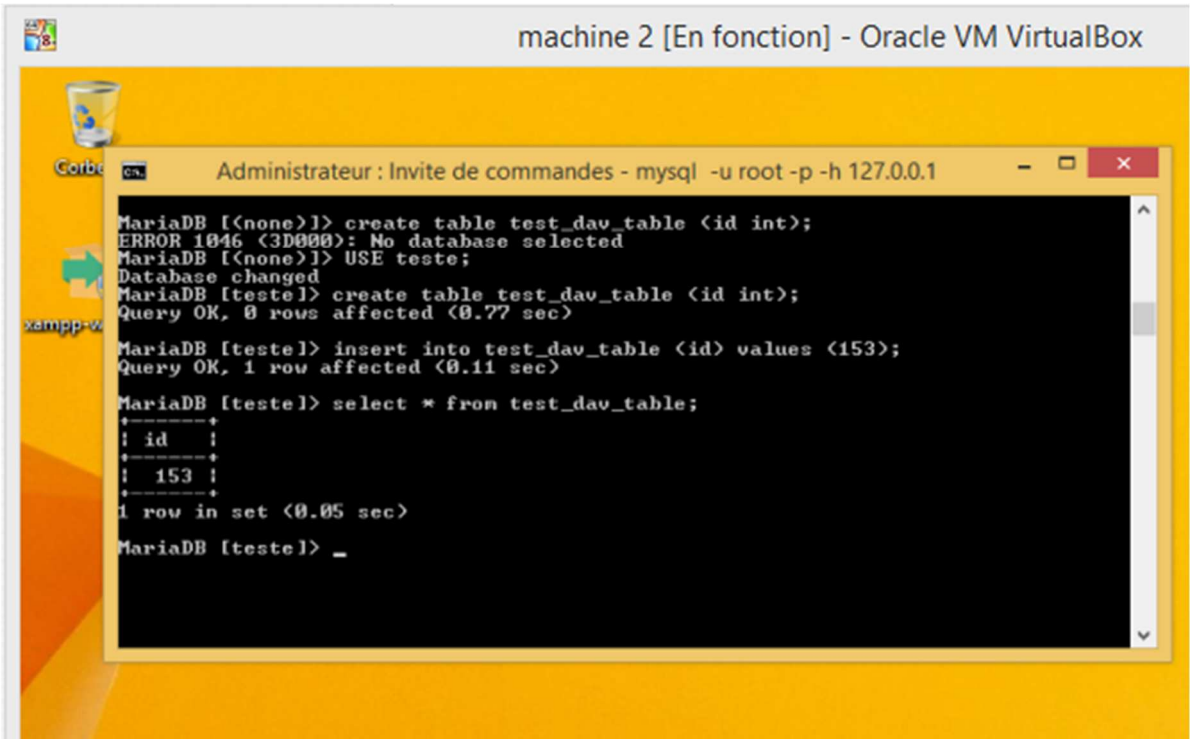
Sur le deuxième serveur

Vérification de la mise à jour des enregistrements

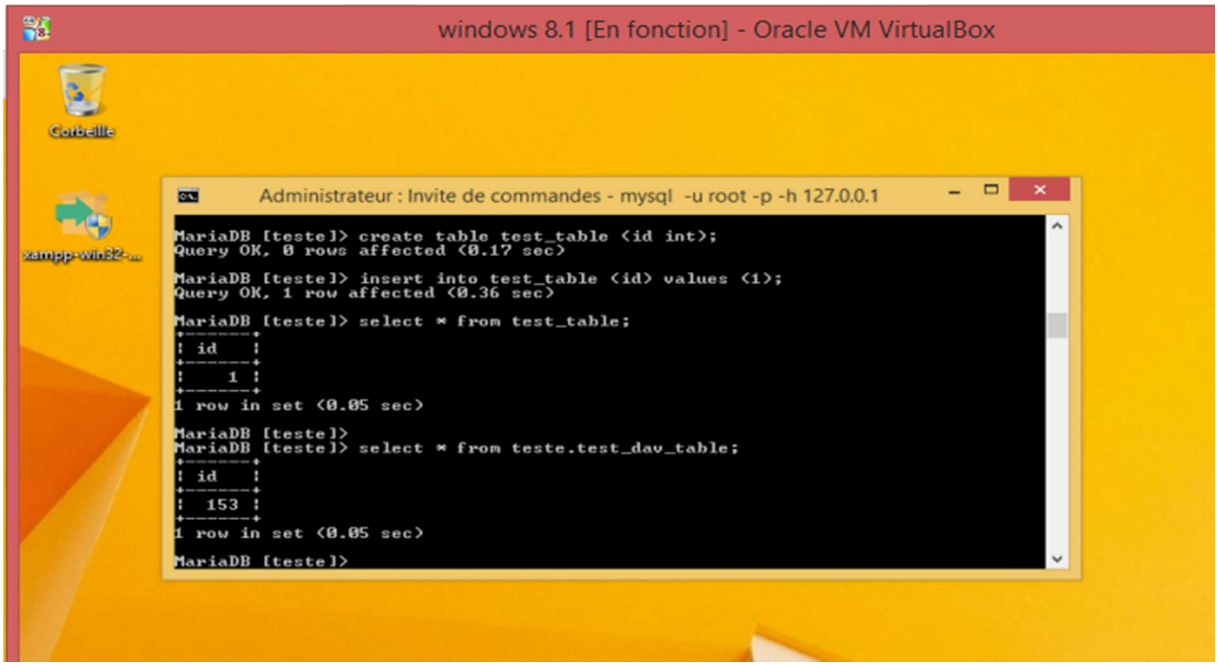


Création de la table « test_table » puis insertion de valeur et un select sur cette table

Avant tout il faudrait d'abord se positionner dans la base de données pour que la création de la table soit possible.



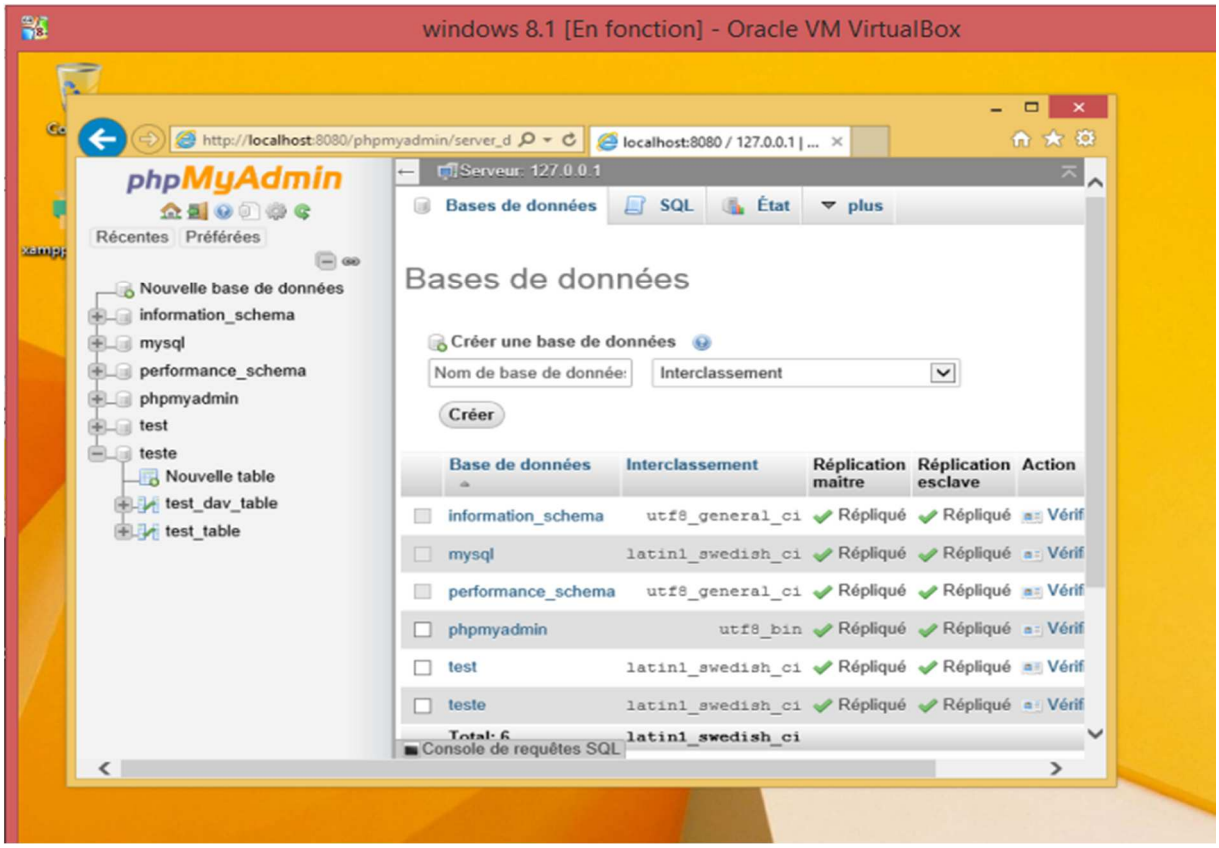
Nous allons maintenant vérifier si les mises à jour ont bien été effectuées d'un serveur à un autre : sur le premier serveur



Nous pouvons constater que les mises à jour ont bien été effectuées.

Sur phpMyAdmin

Premier serveur



La réplication a bien été effectuée.

Sur le deuxième serveur

La réplication a bien été effectuée.

Section 2. Sous Debian en virtuel

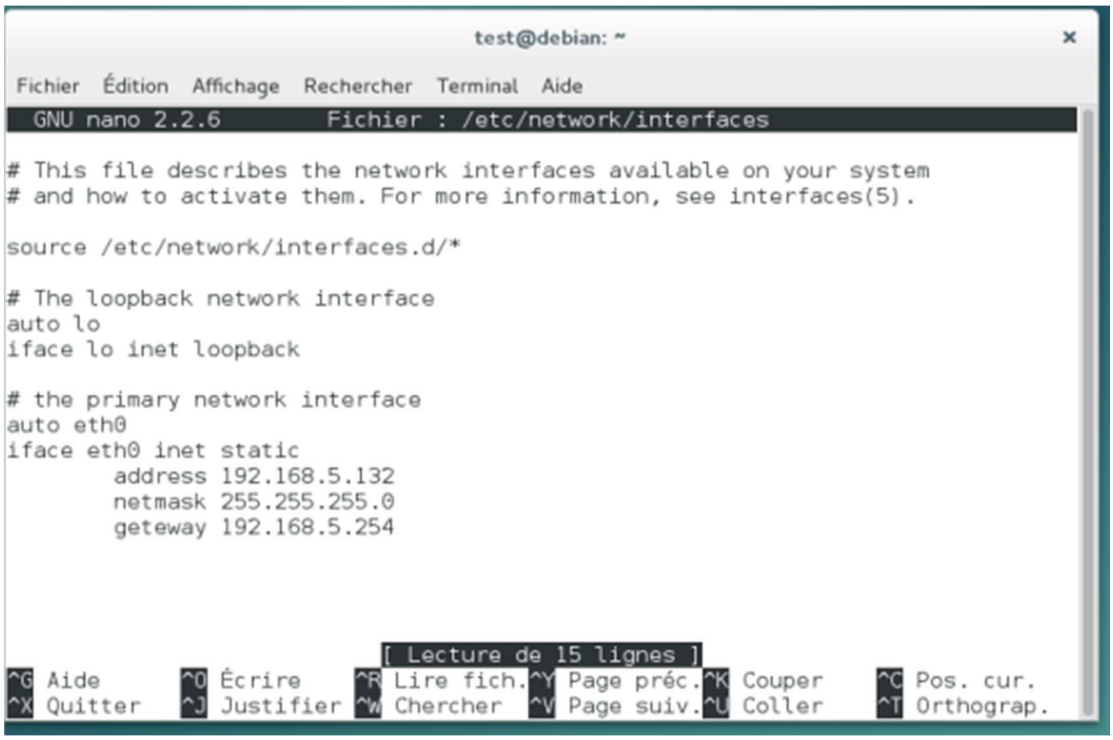
Dans une entreprise, il est primordial de s'assurer qu'aucune donnée ne sera perdue. Il est donc important de faire des sauvegardes régulières et de mettre en place tous les mécanismes qui participeront à minimiser le risque de perte. Nous allons mettre en place une réplication Maître-Maître entre deux serveurs MySQL. Les deux serveurs seront nommés Debian et debian1. Ils auront respectivement les adresses IPV4 192.168.5.132 et 192.168.5.133

- Nous allons donc montrer comment fixer une adresse IP sous Debian.

Pour cela nous accédons dans le fichier de configuration en tapant la commande suivante :

```
root@debian:~# nano /etc/network/interfaces
```

Ensuite nous accedons au fichier de configuration.



Une fois les différentes lignes ajoutées nous quittons en enregistrant les paramètres modifiés. Et pour finir nous tapons la commande suivante pour appliquer les modifications:

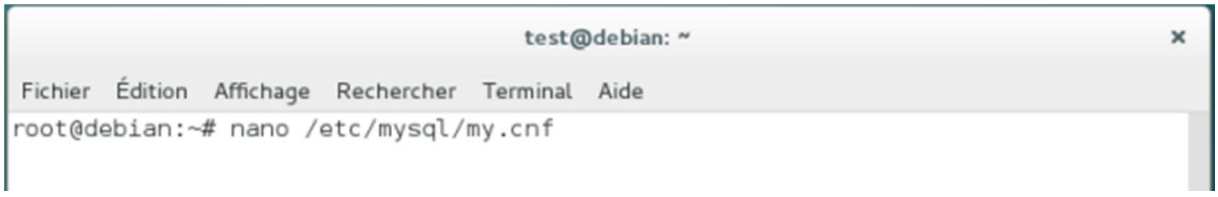
```
/etc/init.d/networking restart
```

Bien entendu, cette opération devra être répétée sur chacun des deux serveurs.

- **Configuration initiale de MySQL**

Nous allons maintenant démarrer la configuration initiale dont MySQL doit faire l'objet pour préparer celui-ci à accepter la réplication. Sur le premier serveur, éditons donc le fichier /etc/mysql/my.cnf avec un éditeur de texte tel que nano.

```
root@debian:~# nano /etc/mysql/my.cnf
```

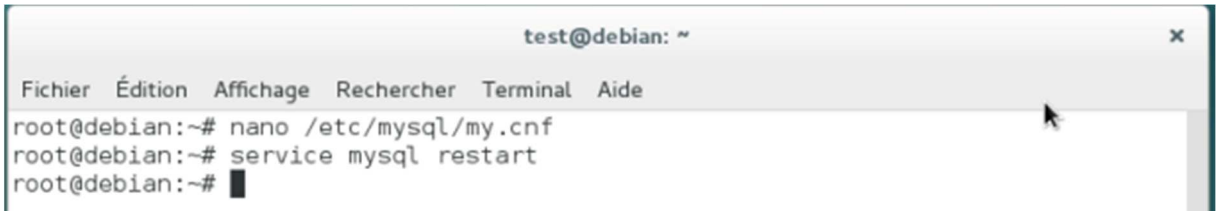


Commentons ensuite la ligne « bind-address » afin que le service MySQL écoute sur toutes les interfaces réseau du serveur et non pas uniquement sur l'adresse locale.

Cette manipulation devra être effectuée sur chacun des deux serveurs MySQL.

Redémarrons ensuite le service pour appliquer les modifications

```
root@debian:~# service mysql restart
```

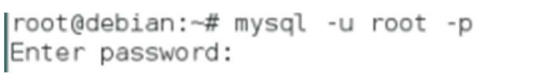


Les prochaines étapes vont consister en un paramétrage du service MySQL afin que celui-ci puisse faire fonctionner correctement la réplication sur chacun des deux serveurs.

- **Préparation de l'environnement pour la réplication MySQL**

L'étape qui va suivre est a réaliser sur les deux serveurs. Connectons-nous au Shell MySQL à l'aide de la commande suivante, en indiquant notre mot de passe.

```
root@debian;~# mysql –u root –p
```



Tout d'abord, nous allons créer une base de données avec la commande suivante

```
Mysql> CREATE DATABASE test ;
```

Ensuite il faut se positionner sur la base de donnée crée avec la commande suivante

```
Mysql> USE test ;
```

```
mysql> USE test;
Database changed
mysql>
```

Ensuite nous allons créer un utilisateur qui sera dédié à la réplication et à l'ajout des données dans les bases de données qui seront répliquées. L'utilisateur sera nommé « 'replic'@'%' » sur les deux serveurs MySQL.

```
mysql> create user'replic'@'%'identified by'pass';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

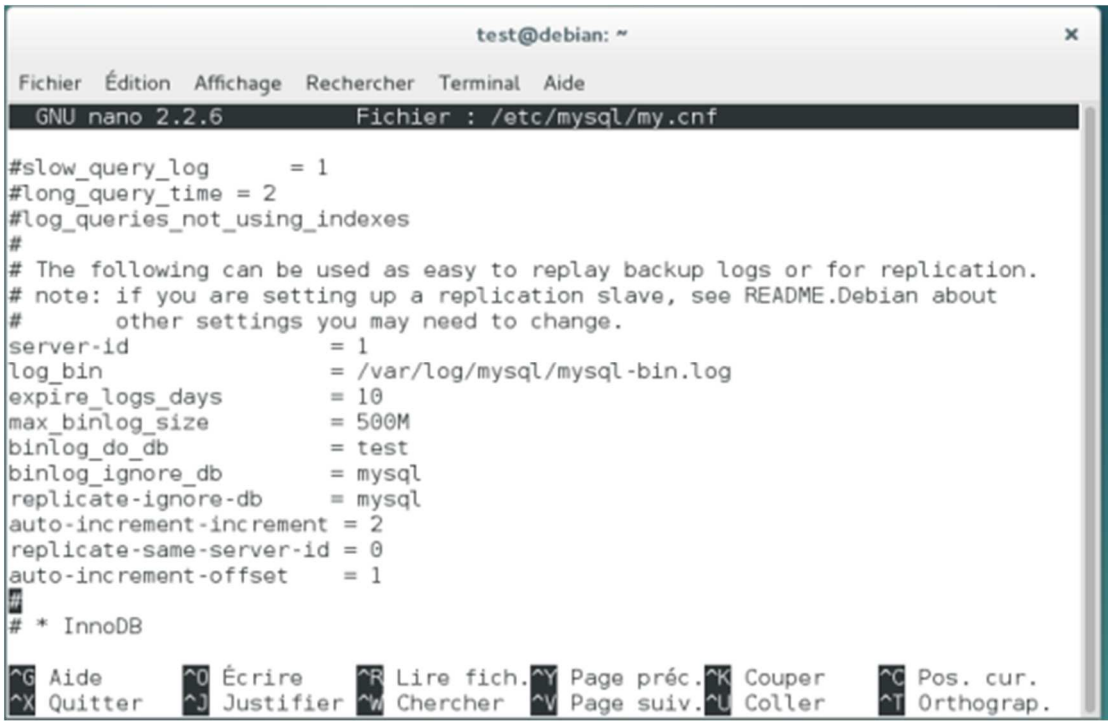
Ensuite nous lui attribuons les privilèges de réplication

```
mysql> grant replication slave on *.* to 'replic'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

- **Configuration de la réplication Maître-Maître MySQL**

La configuration doit maintenant être modifiée pour que les deux serveurs MySQL soient répliqués l'un sur l'autre. Commençons par éditez le fichier /etc/mysql/my.cnf sur le premier serveur.

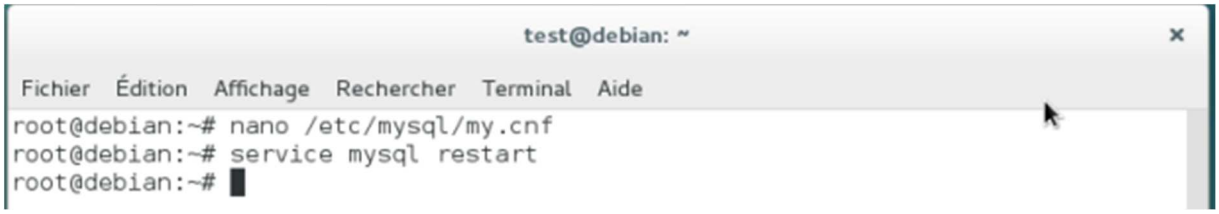
Modifier ou insérer (si nécessaire) les lignes de configuration qui vont de « server-id à auto-increment-offset.



Les paramètres à ajouter ou modifier sont les suivants :

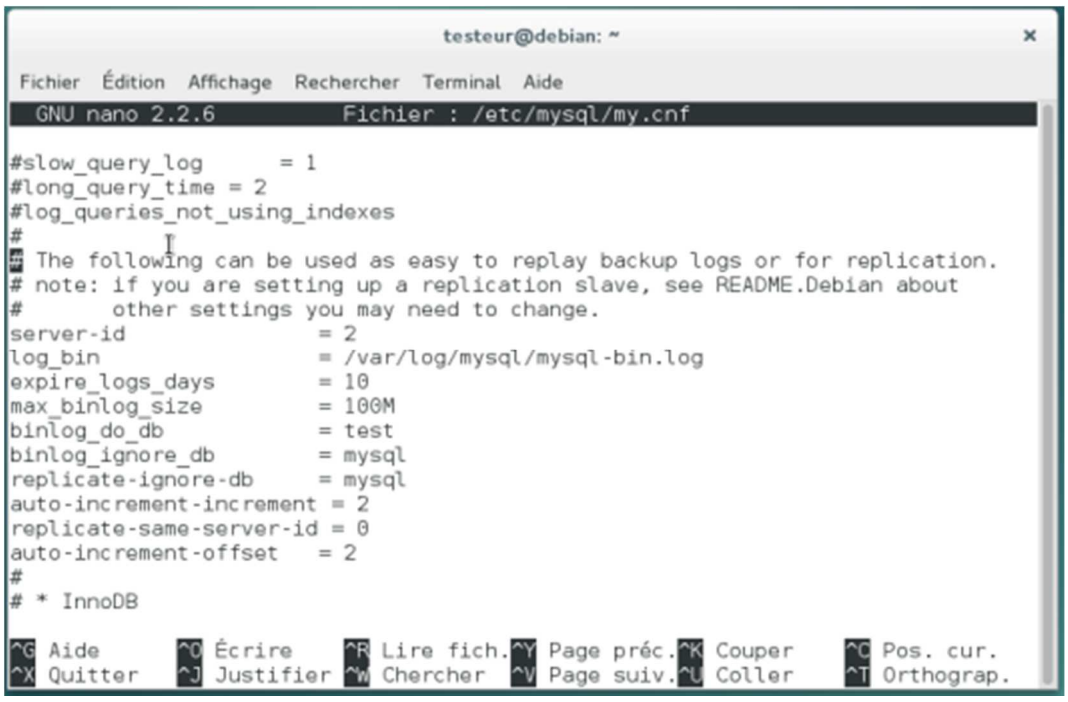
- Server-id permet de définir l'identification unique associé au serveur MySQL
- Log_bin permet de définir l'emplacement du fichier binlog de MySQL
- Expire_logs_days correspond au nombre de jours au cours desquels les données du fichier binlog sont conservés
- Max_binlog_size définit la taille maximale du fichier binlog
- Binlog-ignore-db et replicate-ignore-db permettent de définir les bases de données qui ne sont pas concernées par la réplication (ici, la base mysql)
- Replicate-same-serer-id permet de ne pas répliquer les requêtes sur le serveur MySQL local
- Binlog_do_db permet de montrer au système la base de données qui est concernée par la réplication (ici, la base test)

Pour finir nous devons redémarrez le service MySQL



Il va maintenant falloir faire de même avec le deuxième serveur.

De la même façon, modifions le fichier /etc/mysql/my.cnf. La configuration ici est différente du premier serveur afin d'assurer le bon fonctionnement de la réplication.



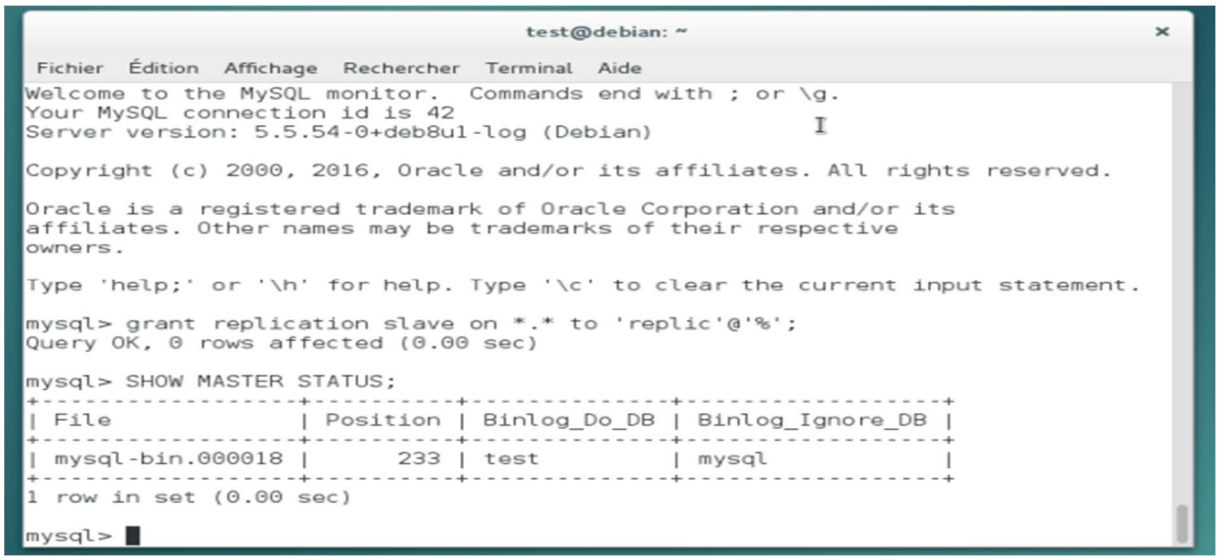
Redémarrer le service de la même manière que pour le premier serveur afin d'appliquer les modifications effectuées précédemment.

- **Sur le premier serveur**

Exécuter la requête ci-dessous pour connaitre le statut du premier serveur.

Ce parti doit être mémorisé

```
Mysql> SHOW MASTER STATUS ;
```



- **Creation des utilisateurs Sur le deuxième serveur**

Création de l'utilisateur qui sera dédié à la réplication

```
mysql> create user'replic'@'%'identified by'pass';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Création de la base de données test.

```
mysql> CREATE DATABASE test;
Query OK, 1 row affected (0.00 sec)

mysql>
```

Attribuer les privilèges de réplication à l'utilisateur

```
mysql> grant replication slave on *.* to 'replic'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Ensuite exécution de la requête slave stop pour arrêter le mode réplication

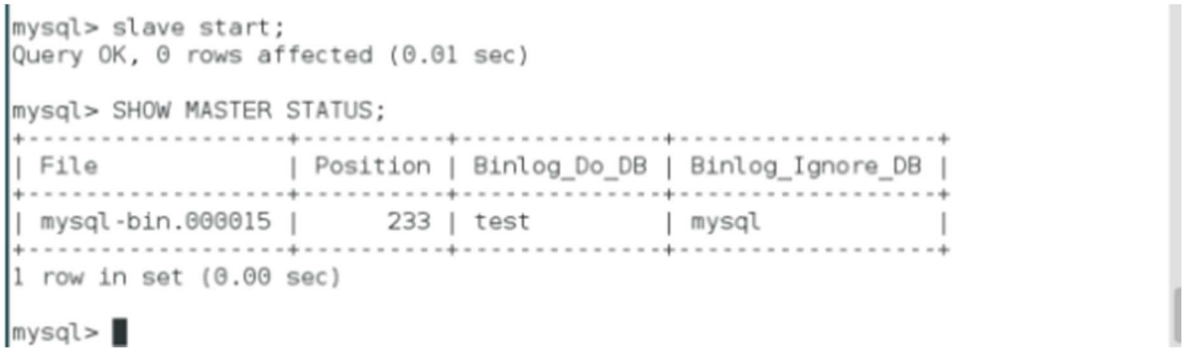
```
mysql> slave stop;
Query OK, 0 rows affected (0.01 sec)

mysql>
```

Exécuter la requête suivante en prenant soin de modifier les paramètres. Les valeurs des paramètres « MASTER_LOG_FILE » et « MASTER_LOG_POS » correspondront aux valeurs récupéré précédemment lors de l'exécution de la requête « SHOW MASTER STATUS »

Ensuite exécution de la commande « slave start » pour démarrer ensuite le serveur en tant qu'esclave. Puis nous exécutons un « show master status »

```
Mysql> slave start ;
```

```
Mysql> SHOW MASTER STATUS ;
```



- **Sur le premier serveur**

Exécutons les requêtes suivantes:

Tout d'abord

```
Mysql> slave stop ;
```

Ensuite

```
Mysql> CHANGE MASTER TO MASTER_LOG_HOST='192.168.5.133', MASTER_USER='replic', MASTER_PASSORD='pass', MASTER_LOG_FILE='mysql-bin.0000015', MASTER_LOG_POS=233 ;
```

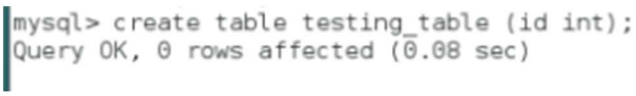
Enfin exécutons « slave start » pour démarrer le serveur en tant qu'esclave

```
Mysql> slave start ;
```

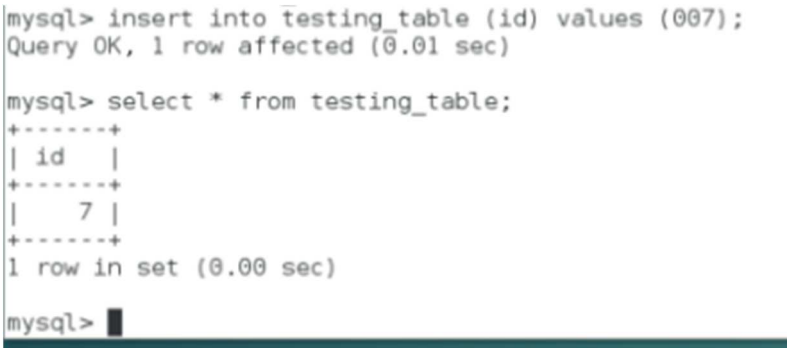
- **Sur le deuxième serveur**

Positionnement sur la base de donnée « test »

Création de la table

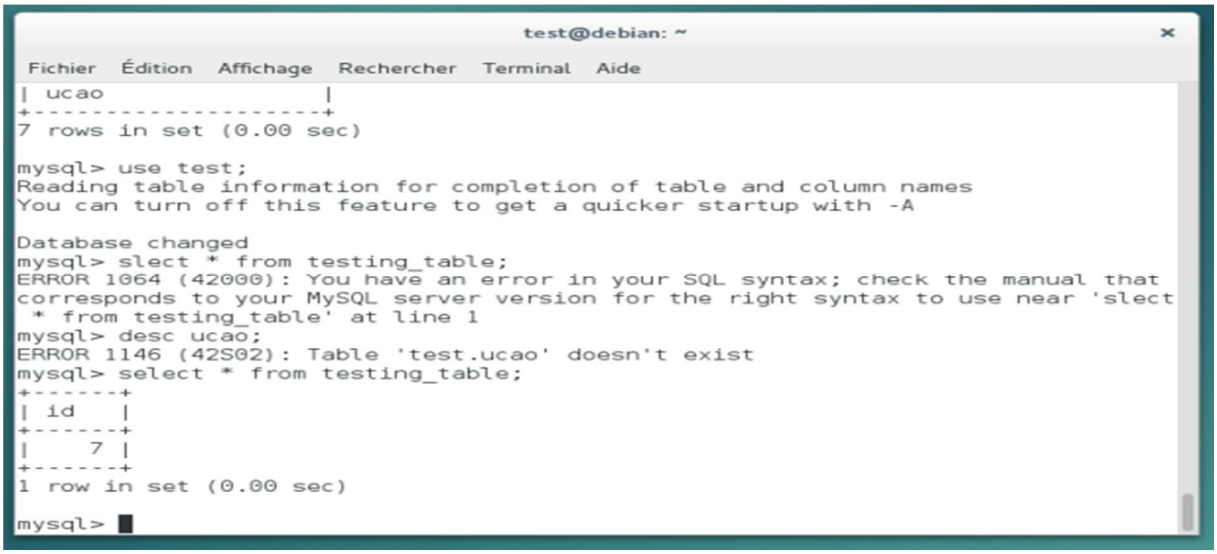


Insertion des données dans la table et nous sélectionnons la table



Sur le premier serveur

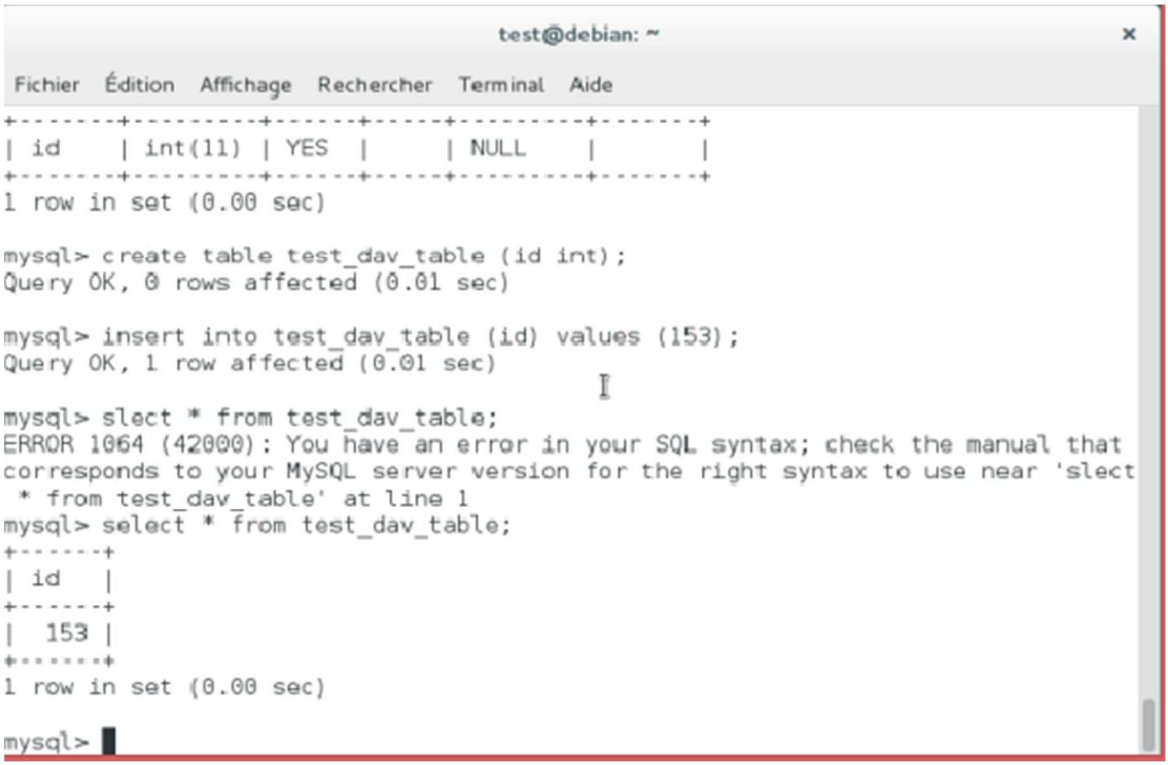
Vérifions que les enregistrements sont bien mis à jour et que la réplication passe.



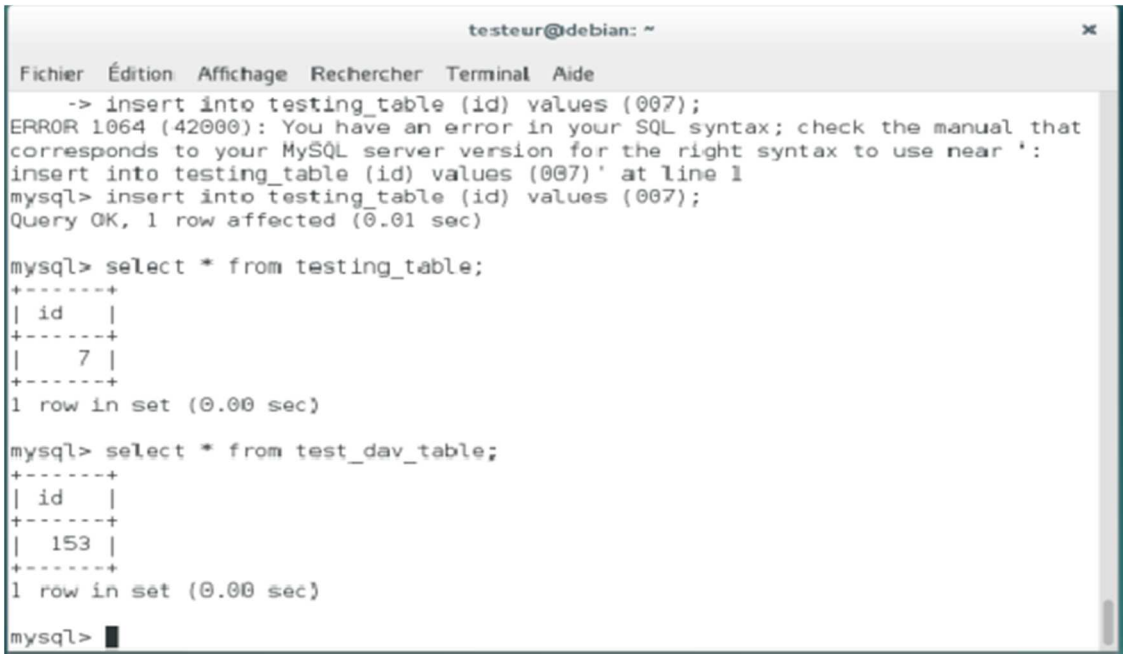
Nous voyons bien que la réplication passe et la table a été créer dans les deux bases de données instantanément.

Sur le premier serveur

Création de la table « test_dav_table » avec insertion des valeurs et sélection de la table.



Vérifions que les enregistrements sont bien mis à jour et que la table a été créé aussi dans l'autre sens c'est à dire sur le deuxième serveur.

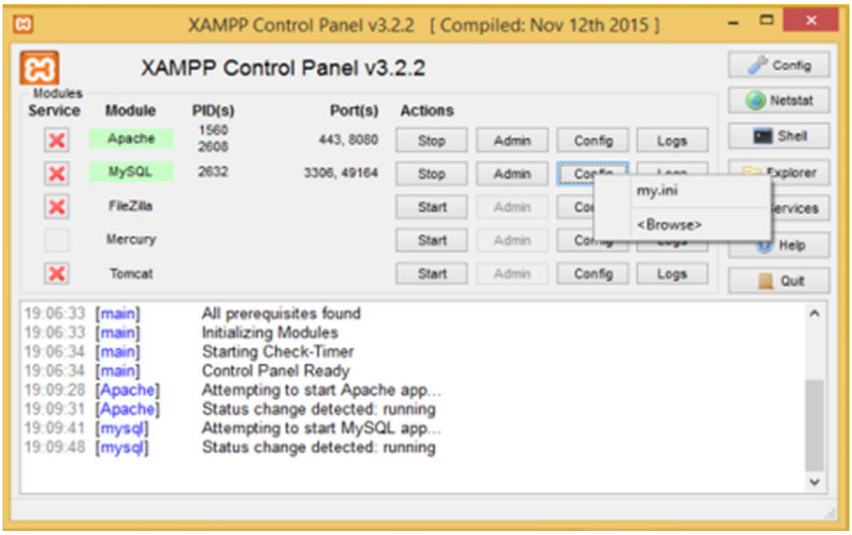


Nous venons donc de réaliser une réplication de donnée synchrone et symétrique

Chapitre 3 : Mise en œuvre de la réplication synchrone symétrique dans un environnement physique

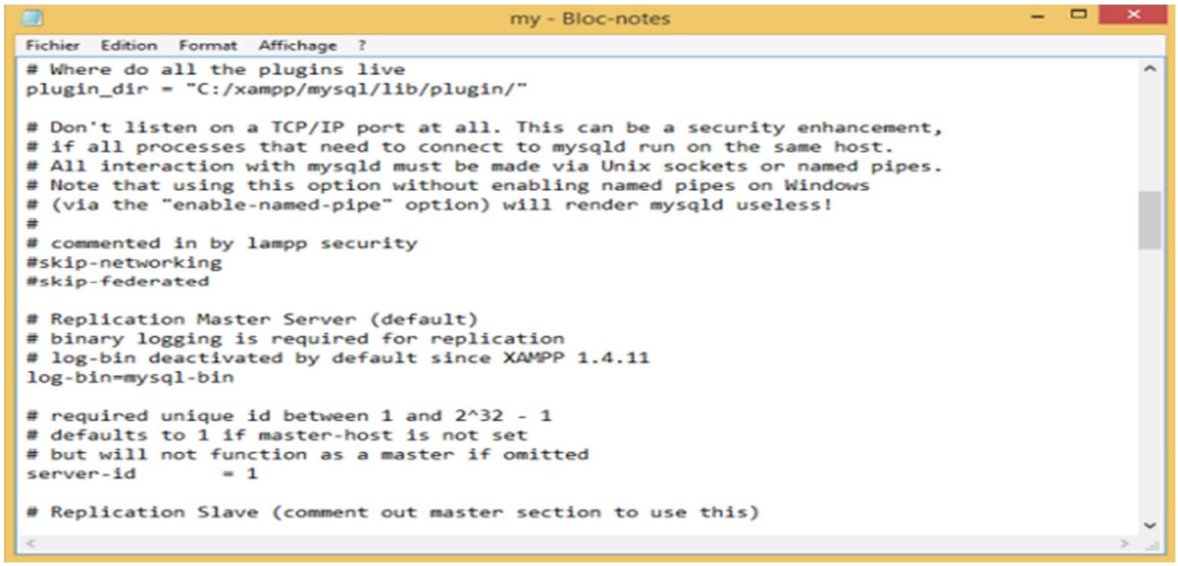
Section 1. Configuration nécessaire

Nous allons maintenant démarrer la configuration initiale dont MySQL doit faire l'objet pour préparer celui-ci à accepter la réplication. Sur le premier serveur, il faudra éditer le fichier my.ini. Pour cela il suffit de cliquer sur le bouton (config) ensuite cliquer sur my.ini, qui se trouve juste sur la ligne MySQL qui se trouve sur le xampp control panel comme ceci. Aussi, les deux serveurs MySQL ont pour ip 192.168.1.15 et 192.168.1.23



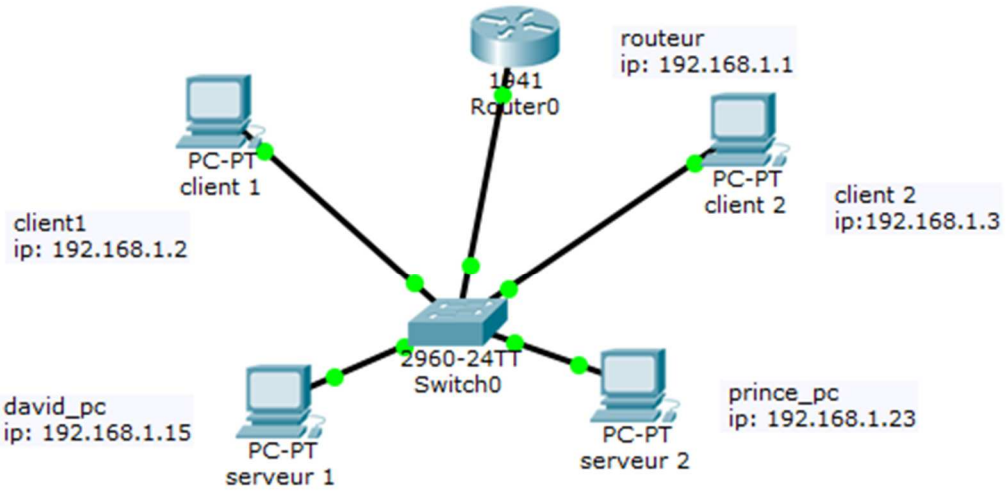
Une fois dans le fichier il suffira de de-commenter les lignes

- Log-bin=mysql-bin
- Server-id = 1



Cette manipulation devra être effectuée sur chacun des deux serveurs MySQL.

Section 2. Sous Windows en environnement réel



SERVEUR 1

```
MariaDB [(none)]> create database teste;
Query OK, 1 row affected (0.03 sec)

MariaDB [(none)]>
```

SERVEUR 2

```
MariaDB [(none)]> create user 'replic'@'%'identified by 'pass';
Query OK, 0 rows affected (0.16 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> use teste;
Database changed
MariaDB [teste]>
```

```
MariaDB [teste]> create user 'replic'@'%'identified by 'pass';
Query OK, 0 rows affected (0.06 sec)

MariaDB [teste]>
```

```
MariaDB [teste]> grant replication slave on *.* to 'replic'@'%';
Query OK, 0 rows affected (0.04 sec)

MariaDB [teste]>
```

```
MariaDB [teste]> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000003 |      746 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

MariaDB [teste]>
```

```
MariaDB [(none)]> STOP SLAVE;
Query OK, 0 rows affected, 1 warning (0.00 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> CHANGE MASTER TO MASTER_HOST='192.168.1.23',MASTER_USER='repli
c',MASTER_PASSWORD='pass',MASTER_LOG_FILE='mysql-bin.000002',MASTER_LOG_POS=736;

Query OK, 0 rows affected (0.39 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> create database teste;
Query OK, 1 row affected (0.05 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> grant replication slave on *.* to 'replic'@'%';
Query OK, 0 rows affected (0.03 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> CHANGE MASTER TO MASTER_HOST='192.168.1.15',MASTER_USER='replic',MASTER_PASSWORD='pass',MASTER_LOG_FIL
E='mysql-bin.000003',MASTER_LOG_POS=746;
Query OK, 0 rows affected (0.26 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> SHOW MASTER STATUS;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000002 |      736 |              |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)

MariaDB [(none)]>
```

```
Administrateur : Invite de commandes - mysql  -u root -p -h 127.0.0.1
MariaDB [(none)]> select * from teste.test_dav;
+------+
| id   |
+------+
|  130 |
+------+
1 row in set (0.00 sec)
MariaDB [(none)]>
```

```
MariaDB [(none)]> START SLAVE;
Query OK, 0 rows affected (0.00 sec)

MariaDB [(none)]>
```

```
MariaDB [(none)]> use teste;
Database changed
MariaDB [teste]>
```

```
MariaDB [teste]> create table test_dav (id int);
Query OK, 0 rows affected (0.53 sec)

MariaDB [teste]>
```

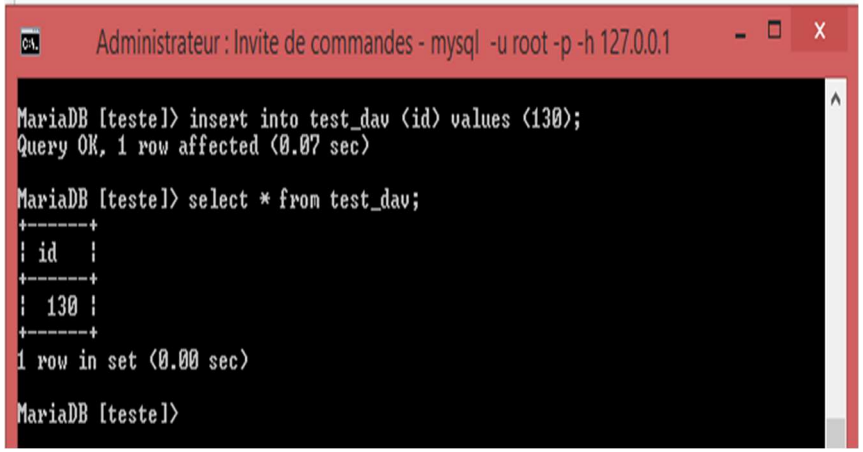
```
MariaDB [teste]> insert into test_dav (id) values (130);
Query OK, 1 row affected (0.07 sec)

MariaDB [teste]>
```

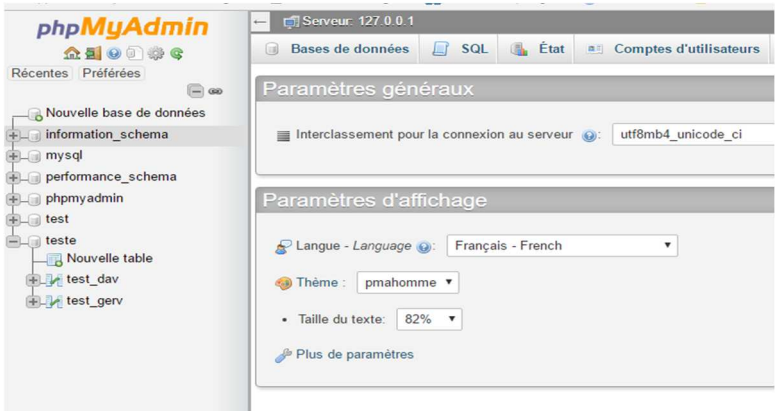
```
MariaDB [(none)]> use teste;
Database changed
MariaDB [teste]>
```

```
MariaDB [teste]> create table test_gerv (id int);
Query OK, 0 rows affected (0.22 sec)

MariaDB [teste]>
```

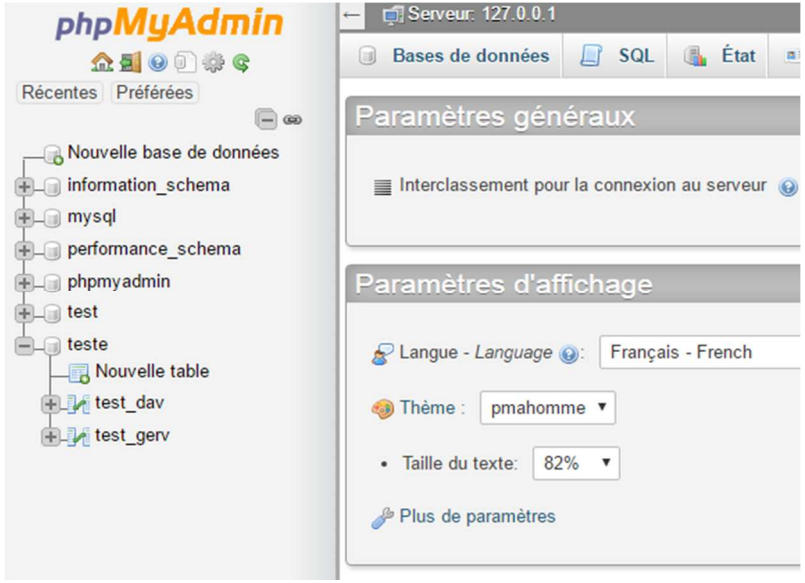
```
MariaDB [teste]> insert into test_gerv (id) values (133);
Query OK, 1 row affected (0.06 sec)
```

```
MariaDB [teste]> select * from test_gerv;
+------+
| id   |
+------+
|  133 |
+------+
1 row in set (0.00 sec)

MariaDB [teste]>
```





```
MariaDB [teste]> select * from test_dav;
+------+
| id   |
+------+
|  130 |
+------+
1 row in set (0.00 sec)

MariaDB [teste]> select * from teste.test_gerv;
+------+
| id   |
+------+
|  133 |
+------+
1 row in set (0.00 sec)

MariaDB [teste]>
```



CONCLUSION GENERALE

L'objectif de ce mémoire étant la proposition d'une réplication de données synchrone et symétrique entre plusieurs serveurs sous le SGBD MySQL, il a été établi les causes, les carences des données répliquées qui justifieraient une telle opération. De même, les atouts, les points fort de la réplication synchrone symétrique qui motiveraient plusieurs entreprise à adopter ce mode de propagation de données en temps réel ont étés abordés. Ensuite une approche de comparaison et d'étude détaillée sur les différents modes de réplication et leur mise à jour de données ainsi qu'un cas pratique en virtuel et en environnement physique pour servir la validation de l'approche proposé selon le bilan suivant furent proposés.

S'agissant des problèmes lors de l'utilisation d'une application de base de données, il a été établi que la disponibilité des données est une inquiétude majeure. En effet, les données en entreprise doivent être disponibles en tout temps pour pouvoir apporter des réponses à différentes interrogations et pour qu'elles soient traités. Comme autre problème lié aux bases de données, le temps de réponse est soulevé. Le temps de réponse est très important car il fait gagner du temps et de l'argent en entreprise. Si les données ne sont pas propagées à temps il y'a risque de perte au niveau de la clientèle et une mauvaise image de l'entreprise. Un autre problème rencontré est la résistance de l'application à des pannes. Une bonne résistance aux pannes permet de mieux gérer les problèmes coté base de données ou serveur pour un bon fonctionnement du flux de données. Voici en quelque sorte les problèmes majeurs rencontré par les bases de données. Pour pallier à ces différents problèmes, l'une des solutions est la réplication des bases de données. La réplication met en relation au moins deux SGBD et consiste à propager les données et modifications réalisées sur la base de données. La réplication est là pour améliorer les performances, l'équilibrage de charge, avoir un meilleur temps de réponse, augmenter la disponibilité des données et améliorer la tolérance aux pannes. Comme point fort de la réplication nous avons la performance qui est améliorée et une probabilité de panne plus faible. Comme point faible la gestion de mises à jour peut poser un problème s'il y a surcoût. A cet effet plusieurs solutions de réplication s'offre à nous.

La réplication mono maître / multi maître ; le maître peut exécuter des requêtes de type lecture ou écriture, l'esclave ne peut exécuter que des requêtes de type lecture et des requêtes de réplication. Et réplication multi maître pour des systèmes avec plus d'un maître.

La réplication symétrique / asymétrique : Par rapport au sens de propagation de la réplication on parle de réplication bidirectionnelle ou symétrique si la réplication se passe dans le sens de l'esclave vers le maître et inversement. La réplication est unidirectionnelle si la réplication se passe dans un seul sens, du maître vers l'esclave.

La mise à jour synchrone : La synchronisation est effectuée en temps réel puisque chaque requête est déployée sur l'ensemble des bases de données avant validation de la requête sur le serveur ou la requête est exécutée.

La réplication synchrone asymétrique : Elle utilise un site primaire qui propage les mises à jour en temps réel vers un ou plusieurs sites secondaires.

La réplication synchrone symétrique : Il n'y a pas de table maître. Pour la mise à jour asynchrone, La réplication asynchrone stock les opérations intervenues sur une base de données locale pour les propager plus tard à l'aide d'un processus de synchronisation.

La réplication asynchrone symétrique : Elle propage les mises à jour en temps différé via une file persistante. Les mises à jour seront exécutées ultérieurement, à partir d'un déclencheur externe, l'horloge par exemple.

La réplication asynchrone symétrique : Dans ce cas, la mise à jour des tables répliquées est différée, cette technique risque de provoquer des incohérences de données.

Toutefois, la réplication des bases de données n'est pas facile à mettre en œuvre c'est pourquoi il ne faut l'appliquer que selon le besoin et dans les scénarios ou il sera vraiment utile comme : le partage des données à travers des postes distants connectées via un réseau étendu (WAN), améliorer l'accessibilité aux données du serveur et la sauvegarde des données. A cet effet, nous avons opté pour une réplication synchrone symétrique qui est une réplication performante assurant un temps de réponse rapide, très couteuse en terme de ressources mais bénéfique et qui permettra de soulager l'organisme qui aura mise en place un tel système de réplication.

BIBLIOGRAPHIE

[1]. Edouard Ngor SARR, Cours base de données avancées, Ucao, (2016)
[2]. Edouard Ngor SARR, TP réplication base de données MySQL, UCAO (2016)
[3]. Olivier Dasini, Cours haute disponibilité avec MySQL, (2008)
[4]. Stéphane Gançarski, cours réplication et base de données, Université Paris, (2010)
[5]. Esther Pacitti, rapport scientifique, Université de Nante, (2008)

REFERENCES WEB

[w1]. http://www.evidian.com/fr/produits/haute-disponibilite-logiciel-clustering-application/replication-synchrone-versus-replication-asynchrone/, novembre 2016

[w2]. http://greg.rubyfr.net/pub/?page_id=20, novembre 2016

[w3]. http://www.jurastick.fr/files/howto/replication_mysql_sxw.pdf, novembre 2016

[w4]. http://www-inf.int-evry.fr/cours/BD/MSCurriculum/FRENCH/PDF/replication.pdf, novembre 2016

[w5]. https://fr.wikipedia.org/wiki/R%C3%A9plication_(informatique), novembre 2016

[w6]. http://www-inf.int-evry.fr/cours/BD/MSCurriculum/FRENCH/PDF/replication.pdf, novembre 2016

[w7]. http://lig-membres.imag.fr/donsez/cours/bdrepartireplica.pdf, novembre 2016

[w8]. http://fr.slideshare.net/sie92/rplication-des-bases-de-donnes, novembre 2016

[w9]. http://www.journaldunet.com/solutions/0301/030127_faq_replication.shtml, novembre 2016

[w10]. https://technet.microsoft.com/fr-fr/library/ms172355(v=sql.105).aspx, novembre 2016

[w11]. http://libresavoir.org/index.php?title=R%C3%A9plication_des_bases_de_donn%C3%A9es_MySQL_(installation_et_configuration), décembre 2016

[w12]. https://www.supinfo.com/articles/single/1759-replication-donnees-avec-mysql, décembre 2016

[w13]. http://dev.mysql.com/doc/refman/5.7/en/replication-howto.html, décembre 2016

[w14]. https://www.synergeek.fr/la-replication-de-bases-de-donnees, décembre 2016

[w15]. https://www.digitalocean.com/community/tutorials/how-to-set-up-mysql-master-master-replication, février 2017

[w16]. http://b3d.bdpedia.fr/replication.html, février 2017

[w17]. https://techan.fr/la-replication-multi-maitres-avec-mysql/, février 2017

[w18]. http://www.zem.fr/installation-dun-serveur-mysql-sous-ubuntu/, février 2017

[w19]. http://docs.postgresql.fr/8.4/high-availability.html, février 2017

[w20]. https://www.howtoforge.com/mysql_master_master_replication#step--2, février 2017

[w21]. https://www.supinfo.com/articles/single/1819-replication-master-master-donnees-mysql, février 2017

[w22]. http://www.univ-eloued.dz/images/memoir/file/M.T-024-1.pdf, février 2017

[w23]. http://www.zem.fr/replication-entre-2-serveurs-mysql/, février 2017

[w24]. https://bucardo.org/wiki/Bucardo, février 2017

[w25]. http://docs.postgresql.fr/9.0/high-availability.html, mars 2017

[w26]. https://technet.microsoft.com/fr-fr/library/ms152565(v=sql.105).aspx, mars 2017

[w27]. https://www.howtoforge.com/mysql_master_master_replication, mars 2017

[w28]. http://jgrondin.developpez.com/article/MySQL/Replication-MySQL/, mars 2017

[w29]. https://www.verticalassertions.fr/blog/outil-monitoring-replication-mysql, mars 2017

[w30]. http://www.woueb.net/2007/11/30/outils-dadministration-de-gestion-et-de-supervision-pour-mysql/, mars 2017

[w31]. http://cdimage.debian.org/debian-cd/current/amd64/iso-cd/, mars 2017

[w32]. https://www.supinfo.com/articles/single/1819-replication-master-master-donnees-mysql, mars 2017

[w33]. http://www.responsive-mind.fr/replication-mysql-master-master/, mars 2017

[w34]. https://blondeau.users.greyc.fr/tutoriaux/bases-de-donnees/replication-mysql-maitre-maitre, mars 2017

[w35]. http://pbarbier.fr/mettre-en-place-une-replication-maitre-maitre-de-mysql-sous-debian-8/, mars 2017

TABLE DES MATIERES