ENSAM
ÉCOLE NATIONALE SUPÉRIEURE D'ARTS ET MÉTIERS
UNIVERSITÉ HASSAN II DE CASABLANCA

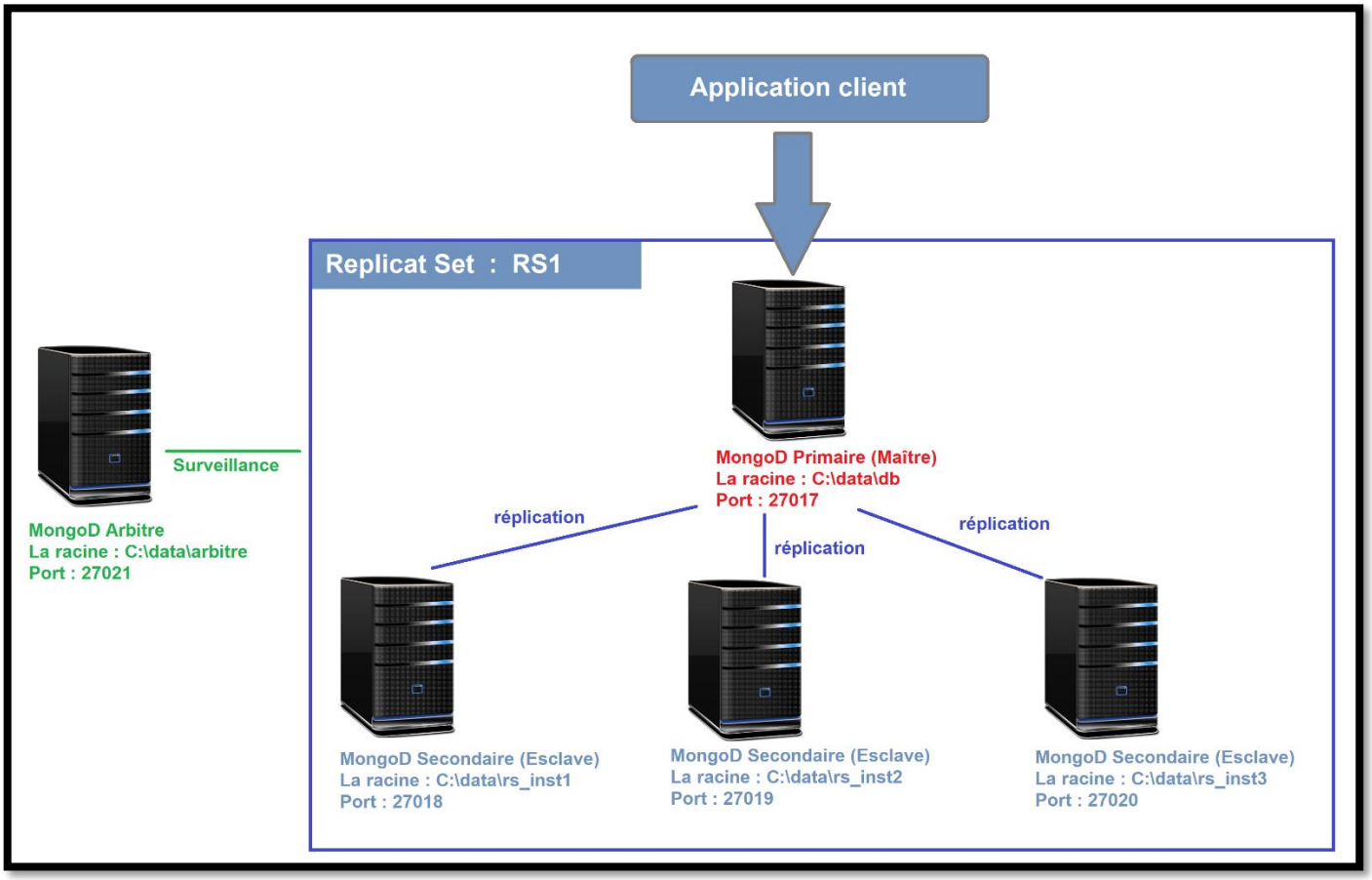
TD/TP 4 :

LA REPLICATION SOUS MONGODB

Le but de ce TP4 est la configuration d'un réseau de réplicas capable de tolérer les pannes

Important : les manipulations sont consécutives, il est demandé d'utiliser plusieurs fenêtres console dont il faut les repérer à travers leurs titres.

Nous allons créer ici un ***replicaSet*** évoqué dans le cours, qui est composé de 5 serveurs, dont 1 primaire, 3 secondaires et un arbitre, suivant la schématisation suivante :



1. CONFIGURATION DU REPLICA SET

Nous considérant le serveur initial de vos données comme étant le serveur primaire.

1. À chaque serveur, un répertoire de données doit être créé :
    - C:\data\rs_inst1

- C:\data\rs_inst2
- C:\data\rs_inst3
- C:\data\arbitre

2. Le réseau des réplicas doit porter un nom. Choisir un nom de réplication : RS1
3. Lancer quatre fenêtres consoles, et repérer leurs emplacements graphiquement.
4. Démarrer les ensembles des instances des serveurs répliqués. Chacune dans l'un des quatre différentes consoles :

```
mongod --replSet RS1 --port 27017 --dbpath C:\data\db
mongod --replSet RS1 --port 27018 --dbpath C:\data\rs_inst1
mongod --replSet RS1 --port 27019 --dbpath C:\data\rs_inst2
mongod --replSet RS1 --port 27020 --dbpath C:\data\rs_inst3
```

- --replSet En spécifiant sous quel nom de réseau de réplication doit-il appartenir
- --journal En activant ensuite la journalisation si votre version est 32 bits (la journalisation est par défaut activé pour les versions 64 bits)
- --port En spécifiant le port d'écoute de chaque serveur instancié
- --dbpath En spécifiant le chemin du répertoire du serveur de données

5. Ouvrir une nouvelle console pour se connecter au serveur qui sera choisi comme serveur primaire. Par exemple, ici dans notre cas c'est le serveur ou vous avez déjà créé des bases (port par défaut 27017) :

```
mongo --port 27017
```

6. Ensuite, Lancer le mode de réplication :

```
rs.initiate()
```

7. Rajouter chaque réplica séparément (en associant le "localhost" de votre machine, ou l'adresse IP de la machine distante) :

```
rs.add("localhost:27018")
rs.add("localhost:27019");
rs.add("localhost:27020");
```

8. Vérifier la configuration de l'ensemble des serveurs :

```
rs.conf();
```

Notre serveur (port :27017) est le premier membre du ReplicaSet "RS1", il a récupéré 1 voix (la sienne) pour devenir PRIMARY, toutefois, sans arbitre. Vous pouvez d'ailleurs constater dans votre console que le serveur est bien primary pour RS1 :

9. Lorsqu'il y a plusieurs réplicas, il est nécessaire de définir un arbitre pour élire un des serveurs secondaires afin de devenir primaire dans le cas où le serveur primaire tombe. Ouvrir une nouvelle console :

```
mongod --replSet RS1 --dbpath C:\data\arbitre --port 30000
```

10. Ajouter l'arbitre dans la **console mongo** RS1:PRIMARY> :

```
rs.addArb("localhost:30000")
```

2. TEST DE LA REPLICATION

11. Dans la console RS1:PRIMARY> vérifier le contenu de l'une des base que vous avez créé dans les TPs précédents, par exemple : use zip_code et ensuite comptez le nombre des documents.
12. Ouvrir une nouvelle console sur l'un serveur réplicat, par exemple

```
mongo --port 27018
```

Remarquez que dans la console mongo apparaîtra RS1:SECONDARY> désignant le serveur secondaire (réplicat).

13. Essayez de refaire le 11 dans cette fenêtre et donner vos remarques
14. La commande retourne une erreur car l'utilisation d'un réplica n'a pas été autorisé. Pour ce faire, taper la commande :

```
rs.slaveOk();
```

15. Refaire le 13
16. Dans la console RS1:PRIMARY>, ajouter un document quelconque à la base zip_code

```
db.code.insert({"titre":"nouveau"})
```

17. Dans la console RS1:SECONDARY>, vérifier son existance :

```
use zip_code
db.code.find({titre:"nouveau"});
```

Cela peut prendre quelques secondes (souvent quelques milli-secondes)

3. TEST DE TOLERANCE AUX PANNES

18. Dans la console RS1:PRIMARY>, ajouter un nouveau document à la base zip_code :

```
db.code.insert({"titre":"nouveau2"})
```

19. Tuer le processus mongod “primaire” (console ouvrant mongod avec le port 27017) avec le raccourcis “ctrl+C”. Ou dans la console RS1:PRIMARY>, taper :

```
use admin;
db.shutdownServer();
```

20. Regarder le comportement des autres mongod, en particulier l’arbitre (port 30000). Sera alors désigné le serveur qui deviendra le primaire. (normalement puisque les poids sont identiques c’est le deuxième serveur ajouté (auparavant secondaire) qui est élu Primaire, remarquez tous cela dans :

```
rs.status()
```

    Remarquez aussi que l’arbitre lance toujours un ***heartbeat*** au serveur 27017 que nous avons crasher expert
21. Dans la console RS1:PRIMARY>, taper ’exit’. Relancer “mongo” mais avec le port du nouveau serveur primaire
22. Vérifier l’existence du dernier document que vous avez ajoutez dans la collection « code » :

```
db.code.insert({"titre":"nouveau2"})
```

    Des explications ???

4. AUTOMATISATION DE LA REPLICATION PAR LES FICHIERS DE CONFIGURATION

Nous souhaitons automatiser le lancement d’un serveur réplica. Pour cela, nous allons créer un fichier de configuration (C\:data\mongo0.conf) Il est nécessaire de créer un fichier de configuration par serveur (chaque serveur en théorie est sur une machine distincte).
Ce fichier de configuration comporte les définitions suivantes pour le serveur initial:

```
# mongo1.conf
#Fichier de log du serveur. Le répertoire doit être créé et mongo doit avoir les droits d’écriture.
#Un fichier de log par serveur
logpath=C:\data\mongod0.log
logappend=true
# Permet de reprendre la main dans le terminal lorsque le serveur est lancé
fork=true
# Paramètres de connexion et de stockage à changer pour chaque serveur
bind_ip=127.0.0.1
port=27017
dbpath=C:\data\db
# Fichier créé pour vérifier si le serveur tourne
pidfilepath C:\data\run\mongod.pid
# Niveau de "oplog" pour la réplication
# 0=off (default), 1=W, 2=R, 3=both, 7=W+some reads
diaglog=1
# Replication : Mettre le nom du replicaSet
replSet=RS1
# Taille maximum de l"oplog" pour la réplication
oplogSize=1024
```

23. Créer un fichier de configuration pour chaque serveur du Replica Set RS1
24. Lancer chaque serveur en utilisant le fichier de configuration correspondant :

```
mongod -config C:\data\mongo0.conf
mongod -config C:\data\mongo1.conf
```

    …
25. Dans le cas où le replica Set ne serait pas initialisé (ce qui n’est pas le cas actuellement), il est possible d’initialiser avec l’ensemble des serveurs contenus dans un document :

```
rsconf = {
_id: "RS1",
members: [
```

```
{_id: 0,host: "<hostname>:27017"},
{_id: 1,host: "<hostname>:27018"},
{_id: 2,host: "<hostname>:27019"},
{_id: 3,host: "<hostname>:27020"},
{_id: 4,host: "<hostname>:30000",arbiterOnly:true},
]
};
rs.initiate(rsconf);
```

5. LE FICHIER OPLOG

Comment les données sont-elles répliquées ?

Une collection particulière est définie automatiquement au niveau du serveur primaire : le fichier ***oplog***. Cette collection particulière va stocker toutes les opérations de mises à jour (journalisation), et les réplicas vont lire cette collection pour les appliquer directement sur leur image.

Par défaut, la réplication se fait en mode ***asynchrone*** ; les réplicas lisent en mode pull ce fichier ***oplog***. Toutefois, cela ne veut pas dire que notre système NoSQL est asynchrone. À partir du moment où nos données ne sont accessibles que sur le ***PRIMARY*** en permanence, la donnée est toujours cohérente.

26. Retrouvez cette collection dans la base de données « **local** » et la collection « **oplog.rs** ». Vous pouvez effectuer une requête pour voir le contenu.
    use local

```
db.oplog.rs.find().pretty()
```

    On pourra y retrouver le timestamp d'insertion (ts), la version (v), le type d'opération ("op" : "i" pour insertion) et la collection cible ("ns")…

Comme vous pouvez le voir, cette collection est régulièrement interrogée pour mettre à jour les réplicas. Il faut toutefois savoir que celle-ci a une taille limitée (***capped collection***). Elle garantit une lecture en continu des mises à jour. Toutefois, en fonction du taux de mises à jour effectuées sur la base MongoDB, il peut être important d'augmenter la taille de l'oPlog pour ne pas perdre d'information. Sa taille par défaut dépend du type de moteur de stockage (par défaut ***WiredTiger***) et du système d'exploitation : comme indiqué dans la documentation. Par défaut, il correspond ***à 5%*** de l'espace disque libre. L'option peut être mise à jour dans le fichier de configuration pour spécifier la taille maximale de ce fichier (***par défaut 5Go***).

6. CHANGEMENT DES POIDS DES REPLICAS LORS DES ELECTIONS DU MAITRE

Une dernière possibilité qu'offre un ReplicaSet est de favoriser le vote dans le cas d'une panne du PRIMARY. En effet, le système de vote peut prendre un certain temps.

Vous pouvez accélérer ce vote en donnant une priorité entre les réplicas. Les réplicas qui ont la priorité la plus haute vont donc passer avant tous ceux qui ont une priorité inférieure. Pour ce faire, il suffit de modifier celles-ci avec :

```
cfg = rs.conf();
cfg.members[0].priority = 0.5;
cfg.members[1].priority = 2;
cfg.members[2].priority = 3;

rs.reconfig(cfg);
```

Le serveur « 2 » sera alors prioritaire lors d'une élection.