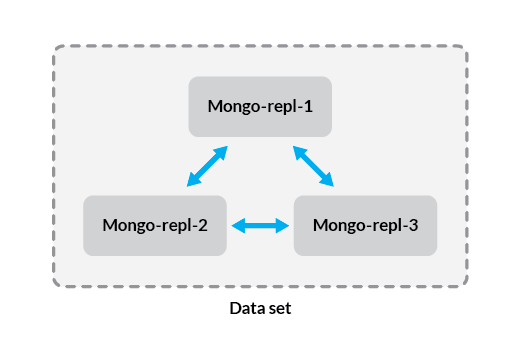
**Mettre en place l le failover avec MongoDB**

1. **On commence par mettre en place la réplication**

Afin d’assurer la redondance et la haute disponibilité, MongoDB possède une fonctionnalité appelée « **replica set** » qui permet aux données d’être dupliquées de manière transparente pour le développeur. Le concept est de créer un groupe de serveur (set) qui possédera un nœud principal (primary) et n serveurs de backup (secondary). Si à un moment donné le nœud principal est inopérant, automatiquement l’un des serveurs de backup deviendra serveur primaire.

Pour permettre une réplication cohérente, chaque nœud devra communiquer avec tous les autres du cluster. Par exemple, dans un ensemble à trois nœuds, le transfert de données ressemblera à ceci :



On commence par arrêter le daemon :

|  |
| --- |
| eldorplus@eldorplus-vm /lib/systemd/system # service mongod stop |

Puis on crée les dossiers où seront stockées les bases :

|  |
| --- |
| eldorplus@eldorplus-vm ~ # mkdir /var/lib/mongodb/rs0  eldorplus@eldorplus-vm ~ # mkdir /var/lib/mongodb/rs1  eldorplus@eldorplus-vm ~ # mkdir /var/lib/mongodb/rs2 |

1. **Configurer le fichier Hosts**

Chaque membre de **replica set** doit avoir un nom d'hôte qui l'identifie en tant que membre de l'ensemble. De cette façon, vous pourrez organiser votre infrastructure à grande échelle (par exemple, si vous ajoutez d'autres **replica set**). Afin de simplifier la configuration de votre jeu de réplicas, ajoutez les lignes suivantes au fichier **/etc/hosts** sur chaque membre du replica set :

|  |
| --- |
| /etc/hosts |
| 127.0.0.1 mongo-primary  127.0.0.1 mongo-secondary-rs1  127.0.0.1 mongo-secondary-rs2 |

1. **Configurer l'authentification MongoDB**

Dans cette section, vous allez créer une clé qui sera utilisé pour sécuriser l'authentification entre les membres de **replica set**. Alors que dans cet exemple vous utiliserez un fichier de clé généré avec *openssl*, MongoDB recommande d'utiliser un [certificat X.509](https://docs.mongodb.com/v3.2/core/security-x.509) pour sécuriser les connexions entre les systèmes de production.

|  |
| --- |
| eldorplus@eldorplus-vm ~ # mkdir /etc/mongodb  eldorplus@eldorplus-vm ~ # cd /etc/mongodb  eldorplus@eldorplus-vm /etc/mongodb # openssl rand -base64 756 > mongod-keyfile  eldorplus@eldorplus-vm /etc/mongodb # chmod 400 mongod-keyfile |

|  |
| --- |
| **Ubuntu / Debian:** |
| eldorplus@eldorplus-vm /etc/mongodb # chown mongodb:mongodb mongod-keyfile |
| **CentOS / RedHat:** |
| eldorplus@eldorplus-vm /etc/mongodb # chown mongod:mongod mongod-keyfile |

1. **Configurer MongoDB**

Pour chacune de vos instances mongodb, apportez les modifications suivantes à vos fichiers /etc/mongod.conf, /etc/mongod.rs1.conf et /etc/mongod.rs2.conf :

|  |
| --- |
| /etc/mongod.conf |
| storage:  dbPath: /var/lib/mongodb/rs0  journal:  enabled: true  systemLog:  destination: file  logAppend: true  path: /var/log/mongodb/mongod.rs0.log  net:  port: 37017  bindIp: 127.0.0.1  security:  keyFile: /etc/mongodb/mongod-keyfile  replication:  replSetName: rs0 |

|  |
| --- |
| /etc/mongod.rs1.conf |
| storage:  dbPath: /var/lib/mongodb/rs1  journal:  enabled: true  systemLog:  destination: file  logAppend: true  path: /var/log/mongodb/mongod.rs1.log  net:  port: 37018  bindIp: 127.0.0.1  security:  keyFile: /etc/mongodb/mongod-keyfile  replication:  replSetName: rs0 |

|  |
| --- |
| /etc/mongod.rs2.conf |
| storage:  dbPath: /var/lib/mongodb/rs2  journal:  enabled: true  systemLog:  destination: file  logAppend: true  path: /var/log/mongodb/mongod.rs2.log  net:  port: 37019  bindIp: 127.0.0.1  security:  keyFile: /etc/mongodb/mongod-keyfile  replication:  replSetName: rs0 |

Pour chacune de vos instances mongodb on crée un service, et on apporte les modifications suivantes à vos fichiers /lib/systemd/system/mongod.service, /lib/systemd/system/mongod-rs1.service et /lib/systemd/system/mongod-rs2.service :

|  |
| --- |
| /lib/systemd/system |
| eldorplus@eldorplus-vm ~ # cd /lib/systemd/system  eldorplus@eldorplus-vm /lib/systemd/system # cp –p mongod.service mongod-rs1.service  eldorplus@eldorplus-vm /lib/systemd/system # cp –p mongod.service mongod-rs2.service  eldorplus@eldorplus-vm /lib/systemd/system # ll | grep mongod  -rw-r--r-- 1 root root 720 août 4 01:17 mongod-rs1.service  -rw-r--r-- 1 root root 720 août 4 01:18 mongod-rs2.service  -rw-r--r-- 1 root root 712 juin 21 21:46 mongod.service |

|  |
| --- |
| /lib/systemd/system/mongod-rs1.service |
| eldorplus@eldorplus-vm /lib/systemd/system $ vim mongod-rs1.service  [Unit]  Description=MongoDB Database Server  After=network.target  Documentation=https://docs.mongodb.org/manual  [Service]  User=mongodb  Group=mongodb  EnvironmentFile=-/etc/default/mongod  ExecStart=/usr/bin/mongod --config /etc/mongod.rs1.conf  PIDFile=/var/run/mongodb/mongod.rs1.pid  # file size  LimitFSIZE=infinity  # cpu time  LimitCPU=infinity  # virtual memory size  LimitAS=infinity  # open files  LimitNOFILE=64000  # processes/threads  LimitNPROC=64000  # locked memory  LimitMEMLOCK=infinity  # total threads (user+kernel)  TasksMax=infinity  TasksAccounting=false  # Recommended limits for for mongod as specified in  # http://docs.mongodb.org/manual/reference/ulimit/#recommended-settings  [Install]  WantedBy=multi-user.target |

|  |
| --- |
| /lib/systemd/system/mongod-rs2.service |
| eldorplus@eldorplus-vm /lib/systemd/system $ vim mongod-rs1.service  [Unit]  Description=MongoDB Database Server  After=network.target  Documentation=https://docs.mongodb.org/manual  [Service]  User=mongodb  Group=mongodb  EnvironmentFile=-/etc/default/mongod  ExecStart=/usr/bin/mongod --config /etc/mongod.rs2.conf  PIDFile=/var/run/mongodb/mongod.rs2.pid  # file size  LimitFSIZE=infinity  # cpu time  LimitCPU=infinity  # virtual memory size  LimitAS=infinity  # open files  LimitNOFILE=64000  # processes/threads  LimitNPROC=64000  # locked memory  LimitMEMLOCK=infinity  # total threads (user+kernel)  TasksMax=infinity  TasksAccounting=false  # Recommended limits for for mongod as specified in  # http://docs.mongodb.org/manual/reference/ulimit/#recommended-settings  [Install]  WantedBy=multi-user.target |

On active par la suite les nouveaux services créent

|  |
| --- |
| eldorplus@eldorplus-vm /lib/systemd/system $ systemctl enable mongod.service  eldorplus@eldorplus-vm /lib/systemd/system $ systemctl enable mongod-rs1.service  eldorplus@eldorplus-vm /lib/systemd/system $ systemctl enable mongod-rs2.service  eldorplus@eldorplus-vm /lib/systemd/system $ systemctl list-unit-files --type=service | grep mongod  mongod-rs1.service enabled  mongod-rs2.service enabled  mongod.service enabled |

Une fois ces modifications apportées, redémarrez le service mongodb :

|  |
| --- |
| eldorplus@eldorplus-vm /lib/systemd/system $ systemctl restart mongod  eldorplus@eldorplus-vm /lib/systemd/system $ systemctl restart mongod-rs1  eldorplus@eldorplus-vm /lib/systemd/system $ systemctl restart mongod-rs2 |

1. **Créer un utilisateur administratif**

Sur la première instance que vous avez l'intention d'utiliser comme membre principal de votre **replica set**, connectez-vous au shell mongo :

|  |
| --- |
| eldorplus@eldorplus-vm /lib/systemd/system $ mongo –port 37017  mongo> use admin # connect to the admin database  switched to db admin  rs0:PRIMARY> use admin |

Créez un administrateur avec les privilèges **root**. Remplacez "mot de passe" par un mot de passe fort de votre choix :

|  |
| --- |
| rs0:PRIMARY> db.createUser({user: "admin", pwd: "password", roles:[{role: "root", db: "admin"}]}) |

1. **Démarrer la réplication et ajouter des membres**

Connectez-vous au shell MongoDB *primary* en utilisant l’administrateur que vous avez créé précédemment :

|  |
| --- |
| eldorplus@eldorplus-vm /lib/systemd/system $ mongo –port 37017 -u admin -p password --authenticationDatabase admin  rs0:PRIMARY> rs.initiate()  {  "info2" : "no configuration specified. Using a default configuration for the set",  "me" : "127.0.0.1:37017",  "ok" : 1  }  rs0:PRIMARY> rs.add("mongo-secondary-rs1:37018")  rs0:PRIMARY> rs.add("mongo-secondary-rs2:37019") |

Une fois tous les membres ajoutés, vérifiez la configuration de votre **replica set** :

|  |
| --- |
| rs0:PRIMARY> rs.status() |

Cette commande affichera un objet de configuration de **replica set** contenant des informations sur chaque membre, ainsi que des métadonnées sur le **replica set**.

**Conclusion**

La mise en place de la réplication est assez simple. Ainsi on peut rapidement mettre en place une infrastructure fiable. Pour augmenter la capacité de montée en charge il suffira d’ajouter des réplica set. MongoDB s’occupera automatiquement de répartir le stockage des données et l’exécution des différentes requêtes. Pour augmenter la fiabilité, on pourra également augmenter le nombre de noeud dans chaque replica set.