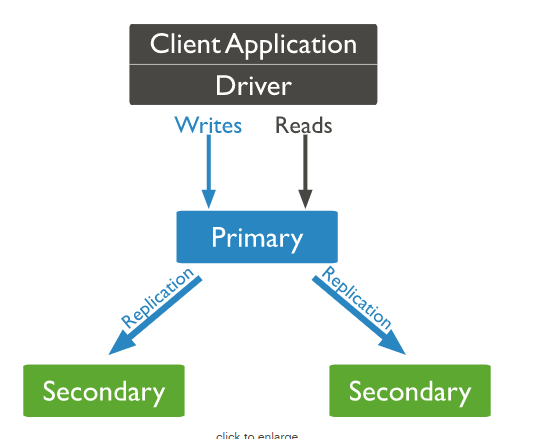
**DECQ Marceau**

**5A – 2I**

**Mise en place d’une architecture tolérante aux pannes avec MongoDB** :

**Notion de Replicat Set :**

C’est une architecture assez simple qui permet de garantir la haute disponibilité d’un service



**Théorème CAP :**

Cohérence : tous les nœuds ont les mêmes données au même moment.

Availability (Disponibilité) : Garantie que toutes les requêtes soient traitées.

Partition tolerance : Tolérance au partitionnement, cela correspond à la faculté d’avoir accès a toutes les données même si elles ne se situent pas sur les mêmes machines.

**Procédure :**

* On créer un répertoire de sauvegarde pour chacun des serveurs (ici 3 dossiers : rs1,rs2,rs3). Ici nous avons donc 3 serveurs avec les ports d’écoutes respectifs suivants : 27018,27019,27020

On va créer nos trois serveurs mongoDB (dans 3 terminaux). On utilise pour chacun la commande suivante :

On se place au préalable dans le /bin pour exécuter mongoDB.



--replSet « nom » permet de configurer la replication

--dbpath « dossier » permet de sauvegarder la base dans ce dossier spécifique

--port « numéro de port » permet de préciser le port utilisé par le serveur mongoDB

--bind\_ip permet de préciser l’IP du serveur

Désormais, on a 3 serveurs mongoDB qui tournent en même temps sur mon serveur (VM)

Remarque : En tant que client, dans la commande de connexion à la base mongoDB, on doit préciser le port, car par défaut, le client tente de se connecter sur le port 27017.



On initialise alors la réplication avec la commande :

rs.initiate()

On observe alors la chose suivante :



On en déduit que le primaire correspond au serveur avec le port 27018.

La commande suivante permet d’afficher la configuration de la réplication :



On obtient alors la configuration suivante :



On voit donc que nos serveurs sont configurés.

Remarque : Lors de l’initialisation, on peut préciser une configuration sous cette forme avec la commande suivante : rs.initiate(rsconf)

On configure le fichier rsconf au préalable.

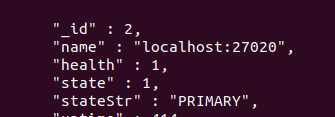
De plus, on peut ajouter un serveur à la réplication à l’aide de la commande :

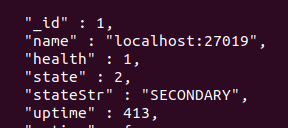
rs.add(hostname :port)

Pour connaitre le statut des serveurs, on utilise la commande suivante :

rs.status()

On obtient alors :





**TEST :**

On teste alors de déconnecter le serveur primaire, on observe que les deux autres montrent que le serveur primaire a été déconnecté :



On se reconnecte alors avec le client sur un autre port et avec la commande rs.status(), on observe que le primary a basculé sur un autre serveur.

**Arbitre :**

L’arbitre ne stocke pas de données mais va faire partie du réplica set. Il est utile pour élire le nœud primaire en cas de panne d’un nœud car il va pouvoir voter. En effet, ayant 2 nœuds secondaires, il est possible que le vote n’aboutisse pas (50/50) alors qu’avec un troisième serveur pour voter, il y aura forcément un nouveau nœud primaire.

On le lance de la même manière que précédemment :



On change le dbpath ainsi que le numéro de port.

On l’ajoute à la réplication avec la commande suivante :

