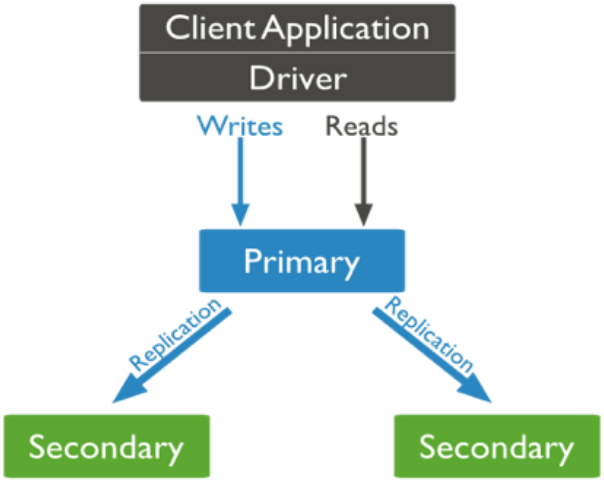
Notes TP du 09-12-20

Introduction à replicaSet :

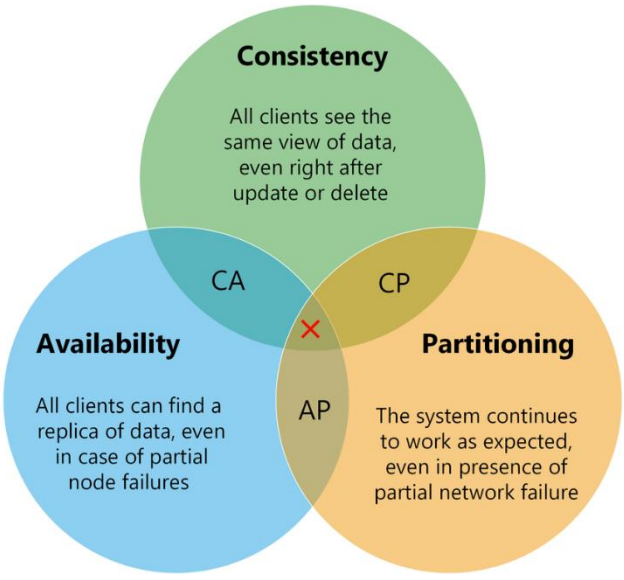
Permet d'avoir des données sauvegardées sur un seul support physique mais accessible depuis plusieurs serveurs.
Réplication : maitre/esclave, maitre stock les données et les esclaves assurent l'accessibilité en cas de problème. Si primary tombe en panne alors élection du nouveau primary parmis les secondary par l'arbitre.



Aucune écriture sur les secondary, lecture possible sur les deux si configuré. Si lecture sur les secondaires quand primaire en panne, possibilité de récupérer une ancienne valeur d'une variable (écriture et lecture avant que les secondaires aient récupérer la modification).
Doc replicaset : Deploy a Replica Set — MongoDB Manual

Disponibilité constante du système, en cas de panne, la tache effectuée par le serveur en panne peut être effectuée par un autre serveur.
Scalabilité, pouvoir distribuer les requêtes read/write. Write seulement sur le primary.

Théorème CAP :
C : cohérence (consistent)
A : disponibilité (availability)
P : tolérance au fractionnement (partition tolerance)



Mise en place : architecture tolérante aux pannes avec mongodb

- Un nom de replicaset : rstest
- Utiliser des ports d'écoutes pour les serveurs (27018, 27019, 27020)
- Définir 3 répertoires dans le bin rs1, 2 et 3 (3 serveurs)
- Lancer les serveurs : mongod.exe --replSet rstest --port 27018 --dbpath rs1 (rs2 et rs3 aussi en changeant le port et le dbpath)
- Se connecter sur le serveur qui sera maitre : mongo.exe --port 27018
- rs.initiate() → lance la config en replicaSet
- rs.conf() → défini le serveur courant en primary
- rs.status() → affiche la config replicaset
- rs.add("localhost:27019") et 27020 → ajoute les secondary
- Créer l'arbitre : mongod.exe --port 30000 --dbpath arb --replSet rstest (créer un répertoire appelé « arb » toujours dans le bin)
- Ajouter l'arbitre sur le mongo : rs.addArb("localhost:30000")
- rs.status → pour vérifier la config

```
{
        "_id" : 1,
        "name" : "localhost:27019",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 265,
        "optime" : {
                "ts" : Timestamp(1607529093, 1),
                "t" : NumberLong(1)
        },
        "optimeDurable" : {
                "ts" : Timestamp(1607529093, 1),
                "t" : NumberLong(1)
        },
        "optimeDate" : ISODate("2020-12-09T15:51:33Z"),
        "optimeDurableDate" : ISODate("2020-12-09T15:51:33Z"),
        "lastHeartbeat" : ISODate("2020-12-09T15:51:35.925Z"),
        "lastHeartbeatRecv" : ISODate("2020-12-09T15:51:35.986Z"),
        "pingMs" : NumberLong(0),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "localhost:27018",
        "syncSourceHost" : "localhost:27018",
        "syncSourceId" : 0,
        "infoMessage" : "",
        "configVersion" : 4
},
```



```
{
        "_id" : 2,
        "name" : "localhost:27020",
        "health" : 1,
        "state" : 2,
        "stateStr" : "SECONDARY",
        "uptime" : 256,
        "optime" : {
                "ts" : Timestamp(1607529093, 1),
                "t" : NumberLong(1)
        },
        "optimeDurable" : {
                "ts" : Timestamp(1607529093, 1),
                "t" : NumberLong(1)
        },
        "optimeDate" : ISODate("2020-12-09T15:51:33Z"),
        "optimeDurableDate" : ISODate("2020-12-09T15:51:33Z"),
        "lastHeartbeat" : ISODate("2020-12-09T15:51:35.925Z"),
        "lastHeartbeatRecv" : ISODate("2020-12-09T15:51:35.981Z"),
        "pingMs" : NumberLong(0),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "localhost:27018",
        "syncSourceHost" : "localhost:27018",
        "syncSourceId" : 0,
        "infoMessage" : "",
        "configVersion" : 4
},
```

```
{
        "_id" : 3,
        "name" : "localhost:30000",
        "health" : 1,
        "state" : 7,
        "stateStr" : "ARBITER",
        "uptime" : 118,
        "lastHeartbeat" : ISODate("2020-12-09T15:51:35.925Z"),
        "lastHeartbeatRecv" : ISODate("2020-12-09T15:51:36.152Z"),
        "pingMs" : NumberLong(0),
        "lastHeartbeatMessage" : "",
        "syncingTo" : "",
        "syncSourceHost" : "",
        "syncSourceId" : -1,
        "infoMessage" : "",
        "configVersion" : 4
}
```


On observe grâce aux zones jaunes la configuration suivante :

- rs1 en PRIMARY
- rs2 et rs3 en SECONDARY
- un arbitre sur le port 30000

Questions :

Comment accepter de lire des informations sur un esclave ?

On peut choisir de changer le mode du paramètre « Read Preference Modes » pour « nearest », dans ce cas la lecture se fait dans serveur avec la latence la plus faible peut importe s'il s'agit d'un maitre ou d'un esclave.

Que signifie le C du théorème CAP :

Que signfie le C du théorème de CAP :

1. **Un client récupère systimatiquement la dernière version d'un document.**
2. **Une transaction ne verra pas les mises à jour des autres transactions qui ne sont pas encore validées.**
3. **Il y'a un ordre préservé des lectures et écritures.**

La bonne réponse est la réponse 1.