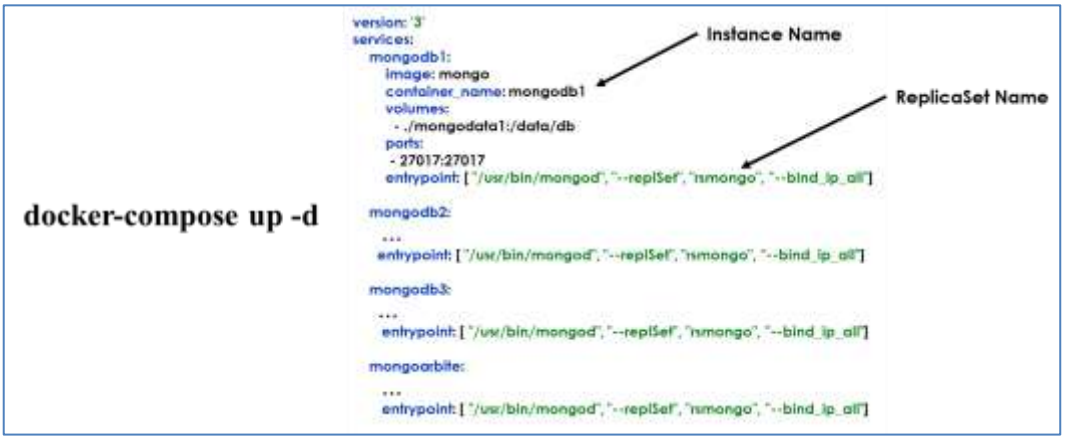
# Fiche de TD/TP N°12- BDD NoSQL :

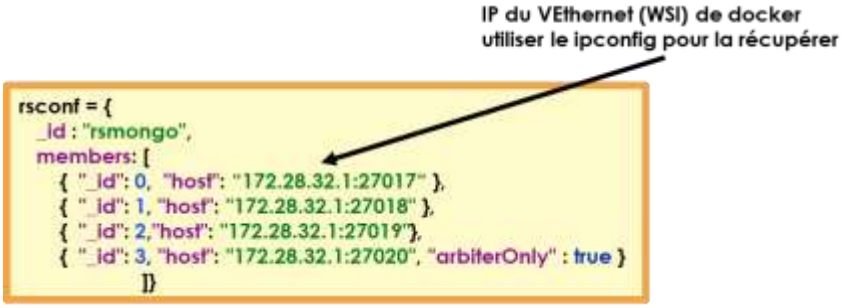
# Réplication et Partitionnement des Bdds MongoDB

## Exercice 1 : Réplication

- Nous allons créer ici un **replicaSet** composé de 4 serveurs, dont 1 ***primaire***, 2 ***secondaires*** et un ***arbitre***.
- À chaque serveur, un répertoire de données doit être créé :
  - ./mongodata1 ; ./mongodata2 ; ./mongodata3; ./mongoarbitre
- Lancer les 4 serveurs en utilisant docker compose :




- Lancer un MongoClient sur le conteneur **mongodb1**
  - **docker-compose exec mongodb1 mongosh**
- Créer un objet **rsconf.**




- Lancer le mode de réplication : **rs.initiate(rsconf);**
  - Dans la console "mongo" apparaîtra **rsmongo:PRIMARY>** désignant le serveur primaire.
- Taper **rs.status();** pour connaitre la configuration du **ReplicaSet**
- Dans la console **rsmongo:PRIMARY>** importer le contenu de la base de données "***DBLP***" :
  - Importer le fichier **dblp.json** dans **DBLP.PUBLIS**
    - Docker cp dblp.json mongodb1:/dblp.json
    - docker exec mongodb1 mongoimport -d dblp -c publis /dblp.json

- **use dblp; db.publis.countDocuments();**

- Lancer un MongoClient sur le conteneur **mongodb2**
  - docker-compose exec mongodb2 mongosh
- Dans la console mongo apparaîtra **rsmongo :SECONDARY>** désignant le serveur secondaire (réplicat).
- La commande : **use dblp; db.publis.countDocuments();** retourne une erreur car l'utilisation d'un réplicat n'a pas été autorisé. Pour ce faire, taper la commande :
  - **rs.secondaryOk() ;** et Refaire le test
- Dans la console **rsmongo:PRIMARY>,** ajouter un document **"{ test : 1}"**
  - **db.publis.save({"test":1});**
- Dans la console **rsmongo:SECONDARY>,** vérifier son existence :
  - **use DBLP;db.publis.find({'test':1});**
- Connecter au **ReplicaSet rsmongo** with **MongoDBCompass**:
  - **mongodb://localhost:27017 /?replicaSet=rsmongo**
- Arrêter le conteneur **Primary** (**mongodb1**)
- Exécuter sur **Compass** quelques requêtes

# Exercice 2: Sharding

- Création d'un replicaSet de configuration composé de 2 serveurs ,
  - Configurer le replicaSet: **docker-compose exec cfgsvr1 mongosh**

```
{"_id" : "rsconfig",
  configsvr: true,
  members: [
   { "_id" : 0, "host" : "cfgsvr1:27017" },
   { "_id" : 1, "host" : "cfgsvr2:27017" }
  ]}
```

- Création d'un replicaSet pour le premier ***shard***
  - Configurer le replicaSet: **docker-compose exec shard1svr1 mongosh**

```
{"_id" : "shard1rs",
  members: [
   { "_id" : 0, "host" : "shard1svr1:27017" },
   { "_id" : 1, "host" : "shard1svr2:27017" }
  ]}
```

- Création d'un replicaSet pour le deuxième ***shard***
  - Configurer le replicaSet: **docker-compose exec shard2svr1 mongosh**

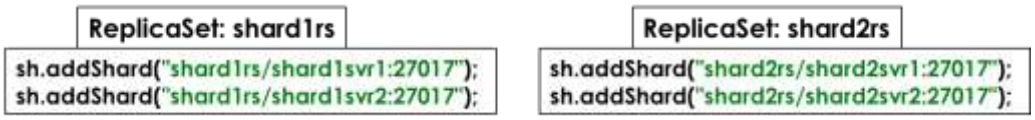
```
{ "_id" : "shard2rs",
  members: [
   { "_id" : 0, "host" : "shard2svr1:27017" },
   { "_id" : 1, "host" : "shard2svr2:27017" }
  ]}
```

- Création d'un routeur de sharding "***mongos***" lié aux serveurs de configuration :

```
mongos:
  container_name: mongos
  image: mongo
  command: mongos --configdb rsconfig/cfgsvr1:27017,cfgsvr2:27017 --bind_ip_all
  ports:
    - 60000:27017
```

- Configurer les serveurs de ***sharding*** à travers mongos :

- **docker-compose exec mongos mongosh**




- Permettre le sharding sur un BD :
  - **sh.enableSharding("dblp");**
- Définir la clé de distribution (type + year) :
  - **db.adminCommand(**

    **{ shardCollection: "dblp.publis", key: {type:"hashed", year: 1 } } )**
- Lancer le script d'importation à travers le serveur de routage ***mongos***
  - **Docker cp dblp.json mongos:/dblp.json**
  - **docker exec mongos mongoimport -d dblp -c publis /dblp.json**
- Afficher la configuration de la distribution
  - **Sh.status() ou db.printShardingStatus()**
  - **db.publis.getShardDistribution();**