GESTION DES DONNEES REDHAT

redhat®

ASISR	Selim AKALAN	15/02/24

SOMMAIRE

Introduction

- La gestion des données avec Red Hat repose sur des solutions robustes et sécurisées, offrant une infrastructure flexible et évolutive pour la collecte, le stockage, l'analyse et la gestion des données. Red Hat propose des outils tels que Red Hat OpenShift et Red Hat Enterprise Linux, qui permettent aux entreprises de tirer parti de l'open source pour gérer efficacement leurs données, en garantissant la conformité, la disponibilité et la performance à chaque étape du processus. Grâce à son approche centrée sur l'open source et sa forte expertise en matière de sécurité, Red Hat permet aux organisations de maximiser la valeur de leurs données tout en réduisant les risques et les coûts associés à leur gestion.

Prérequis

- Installation d'une machine Redhat.

Partie 1: Configuration de l'Environnement

- **Installation de MongoDB sur Redhat:**

- Installez MongoDB sur une machine Redhat. Assurez-vous que le service MongoDB est actif et fonctionne correctement.
- Pour installer Mongodb pour CentOS7 (et plus) :

```
[root@localhost ~]# vi /etc/yum.repos.d/mongodb-org-7.0.repo
[root@localhost ~]#
```

- Ajouter dans le fichier les lignes comme indiquez sur l'image ci-dessous :

```
[mongodb-org-7.0]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/7.0/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-7.0.asc
~
```

- Commande pour l'installation de MongoDB :

```
sudo yum install -y mongodb-org
```

- Commande pour le démarrage du service MongoDB :

```
sudo systemctl start mongod
```

- Commande pour l'activation du démarrage automatique :

```
sudo systemctl enable mongod
```

- Sur une machine Ubuntu, on utilise la commande suivante :

```
sudo apt install -y mongodb
```

- La commande pour vérifier que MongoDB est actif : *sudo service mongod status*

```
[user@localhost ~]$ sudo service mongod status
Redirecting to /bin/systemctl status mongod.service
● mongod.service - MongoDB Database Server
     Loaded: loaded (/usr/lib/systemd/system/mongod.service; en
abled; preset: disabled)
     Active: active (running) since Wed 2023-12-06 11:27:32 CET
; 3s ago
       Docs: https://docs.mongodb.org/manual
   Main PID: 37418 (mongod)
     Memory: 149.7M
        CPU: 744ms
     CGroup: /system.slice/mongod.service
             └─37418 /usr/bin/mongod -f /etc/mongod.conf
```

- La commande Pour se connecter à Mongodb: *sudo mongosh*
- Entrez le mot de passe root :

```
[user@localhost ~]$ sudo mongosh
[sudo] Mot de passe de user :
Current Mongosh Log ID: 65707d2486497e296bc0a8b6
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serv
erSelectionTimeoutMS=2000&appName=mongosh+2.1.1
Using MongoDB:          7.0.4
Using Mongosh:          2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2023-12-06T14:30:43.320+01:00: Access control is not enabled for the datab
ase. Read and write access to data and configuration is unrestricted
   2023-12-06T14:30:43.321+01:00: /sys/kernel/mm/transparent_hugepage/enabled
 is 'always'. We suggest setting it to 'never'
   2023-12-06T14:30:43.321+01:00: vm.max_map_count is too low
------

test>
```

- **Configuration de SELinux:**

- La commande pour vérifier que SELinux est activé en mode 'Enforcing' : *sestatus*
- Sinon aller dans le fichier /etc/sysconfig/selinux

```
[user@localhost ~]$ sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

- La commande pour afficher les règles de politique SELinux actuellement chargées : *seinfo -a*

```
seinfo -a
```

Partie 2: Administration de Groupe sous Redhat

- **Création de Groupes Utilisateurs :**
- Créez deux groupes utilisateurs : **groupeA** et **groupeB**.
- Ajoutez des utilisateurs dans ces groupes.
- **Attribution de Permissions :**
- Attribuez des permissions spécifiques sur les répertoires de ces groupes pour les droits de modification des fichiers systèmes de MongoDB.
- La commande pour créer un groupe d'utilisateur : *sudo groupadd groupeA/B*

```bash
sudo groupadd nom_du_groupe
```

- La commande pour ajouter des utilisateurs : *useradd -m utilisateurs1/2*

```
[root@localhost ~]# useradd -m Utilisateur1
[root@localhost ~]# useradd -m Utilisateur2
```

- Ajout des utilisateurs dans les groupes + persmissions :

sudo usermod -aG groupeA Utilisateur1
sudo usermod -aG groupeB Utilisateur2
sudo chown -R : groupeA /var/lib/mongo/
sudo chown -R : groupeB /var/lib/mongo/
sudo chown -R 750 /var/lib/mongo/

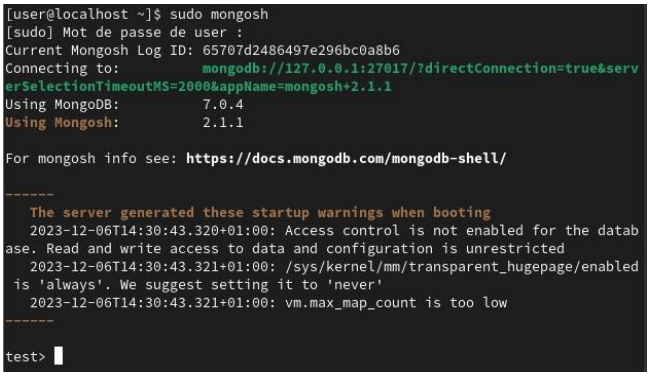
750 = permission du propriétaire en lecture, écriture et exécution, et le groupe à la permission en lecture et exécution dans le répertoire /var/lib/mongo/.

```
[root@localhost ~]# usermod -aG groupeA Utilisateur1
[root@localhost ~]# usermod -aG groupeB Utilisateur2
[root@localhost ~]# chown -R :groupeA /var/lib/mongo/
[root@localhost ~]# chmod -R 750 /var/lib/mongo/
[root@localhost ~]# chown -R :groupeB /var/lib/mongo/
[root@localhost ~]# chmod -R 750 /var/lib/mongo/
```

Partie 3: Manipulation de Données avec MongoDB

- **Connexion à MongoDB :**

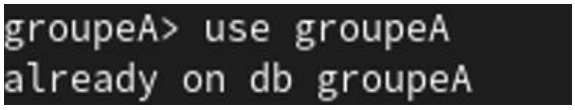
➢ La commande pour se connecter à MongoDB en utilisant les commandes shell de MongoDB : *sudo mongosh*



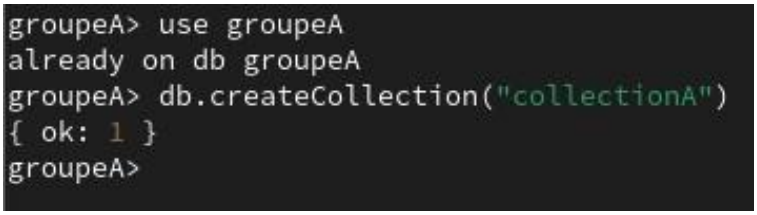
- **Création de Bases de Données et Collections :**

➢ Créez deux bases de données distinctes pour **groupeA** et **groupeB**. Dans chaque base de données, créez des collections pertinentes.

➢ La commande pour créer une base de données distinctes pour chaque groupe : *use groupe*..



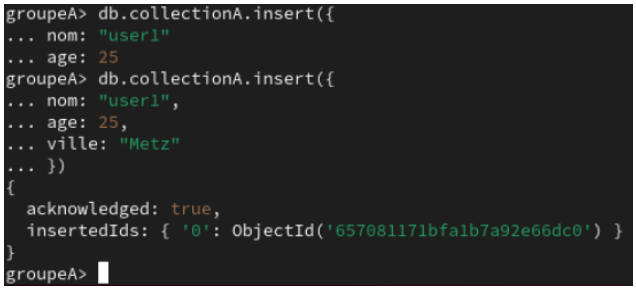
➢ La commande pour créer des collections pertinente pour chaque groupe : *db.createCollection(«nomdelacollection »)*



- **Insertion de Données :**

➢ Insérez des données de test dans les collections des deux groupes.

➢ Pour chaque collection, on crée un user :



➢ La commande qui permet de vérifier les collections : *groupeA> db.collectionA.find()*

```
groupeA> db.collectionA.find()
[
  {
    _id: ObjectId('65707ef114f52d1fc0d7bcd8'),
    nom: 'Utilisateur1',
    age: 21,
    ville: 'Guénange'
  }
]
```

```
groupeA> use groupeB
switched to db groupeB
groupeB> db.collectionB.find()
[
  {
    _id: ObjectId('65707f6f14f52d1fc0d7bcd9'),
    nom: 'Utilisateur2',
    age: 20,
    ville: 'Thionville'
  }
]
```

- **Extraction de Données :**

➢ Exécutez des requêtes pour extraire des données des collections de chaque groupe.

➢ La commande pour extraire les données des collections pour chaque groupe : *mongoexport –host="127.0.0.1 :27017" --collection=collectionA --db=groupea --out=groupeA.json*

➢ La commande pour ouvrir le fichier créé groupeA.json : *cat groupeA.json*

```
[user@localhost ~]$ mongoexport --host="127.0.0.1:27017" --collection=collectionA --db=groupeA --out=groupeA.json
2023-12-06T15:54:50.835+0100    connected to: mongodb://127.0.0.1:27017/
2023-12-06T15:54:50.840+0100    exported 1 record
[user@localhost ~]$ ls
Bureau  Documents  groupeA.json  Images  Modèles  Musique  Public  Téléchargements  Vidéos
[user@localhost ~]$ cat groupeA.json
{"_id":{"$oid":"65707ef114f52d1fc0d7bcd8"},"nom":"Utilisateur1","age":21,"ville":"Guénange"}
[user@localhost ~]$
```

➢ Pour la collection B du groupe B, on effectue la même commande que la commande précédente : *mongoexport –host="127.0.0.1 :27017" --collection=collectionB --db=groupeB --out=groupeB.json*

➢ La commande pour ouvrir le fichier créé groupeB.json : *cat groupeB.json*

```
[user@localhost ~]$ mongoexport --host="127.0.0.1:27017" --collection=collectionB --db=groupeB --out=groupeB.json
2023-12-06T15:55:56.578+0100    connected to: mongodb://127.0.0.1:27017/
2023-12-06T15:55:56.588+0100    exported 1 record
[user@localhost ~]$ ls
Bureau  Documents  groupeA.json  groupeB.json  Images  Modèles  Musique  Public  Téléchargements  Vidéos
[user@localhost ~]$ cat groupeB.json
{"_id":{"$oid":"65707f6f14f52d1fc0d7bcd9"},"nom":"Utilisateur2","age":20,"ville":"Thionville"}
```

Partie 4: Sécurité SELinux avec MongoDB

- **Politiques SELinux pour MongoDB :**

➢ Etudiez comment SELinux gère l'accès aux données de MongoDB

SELinux (Security-Enhanced Linux) gère l'accès aux données de MongoDB en appliquant des politiques de sécurité basées sur le contrôle d'accès obligatoire (MAC). Le fonctionnement précis dépend de la politique SELinux spécifique en place sur votre système et des modules SELinux associés à MongoDB.

Voici comment SELinux peut interagir avec MongoDB :

1. Contexte de sécurité des fichiers et processus ;
2. Accès aux ports réseau ;
3. Logs et audits
4. Politiques personnalisées ;
5. Adaptation de la politique SELinux.

Modifiez les politiques SELinux pour restreindre/autoriser l'accès aux données de MongoDB pour groupeA et groupeB

- Pour commencer, j'ajouter un rôle « roleA » dans mon groupe A, qui va me permettre d'attribuer mes permissions dans la base de données puis je crée un utilisateur à qui j'attribue ce rôle :

```
groupe> use groupeA
switched to db groupeA
groupeA> db.createRole(
...     {
...         role: "roleA",
...         privileges: [{ resource: { db: "groupeA", collection: "" }, actions: [ "find", "update", "insert", "remove" ] }],
...         roles: [
...           { role: "userAdmin", db: "groupeA" },
...           { role: "readWrite", db: "groupeA" }
...         ]
...     }
... );
{ ok: 1 }
groupeA>

groupeA>

groupeA> db.createUser(
...     {
...         user: "user1",
...         pwd: "12345",
...         roles: [{ role: "roleA", db: "groupeA" }]
...     }
... );
{ ok: 1 }
```

- Ensuite, je fais la même chose pour la base de données du groupe B :

```
groupeA> use groupeB
switched to db groupeB
groupeB> db.createRole(
...     {
...         role: "roleB",
...         privileges: [{ resource: { db: "groupeB", collection: "" }, actions: [ "find", "update", "insert", "remove" ] }],
...         roles: [
...           { role: "userAdmin", db: "groupeB" },
...           { role: "readWrite", db :"groupeB" }
...         ]
...     }
... );
{ ok: 1 }
groupeB> db.createUser(
... {
...     user: "user2",
...     pwd: "12345",
...     roles: [{ role: "roleB", db: "groupeB" }]
...   }
... );
{ ok: 1 }
groupeB>
```

Pour vérifier que mes rôles ce sont bien créés, je peux effectuer la commande : *db.getRoles()*. En ce qui concerne les utilisateur je peux faire la commande *db.getUsers()* :

- Pour le groupe A :

```
groupeA> db.getRoles()
{
  roles: [
    {
      _id: 'groupeA.roleA',
      role: 'roleA',
      db: 'groupeA',
      roles: [
        { role: 'userAdmin', db: 'groupeA' },
        { role: 'readWrite', db: 'groupeA' }
      ],
      isBuiltin: false,
      inheritedRoles: [
        { role: 'userAdmin', db: 'groupeA' },
        { role: 'readWrite', db: 'groupeA' }
      ]
    }
  ],
  ok: 1
}
```

```
groupeA> db.getUsers()
{
  users: [
    {
      _id: 'groupeA.Lebron',
      userId: UUID('901c939e-728f-41f4-8800-72dd3b3b9a69')
,
      user: 'Lebron',
      db: 'groupeA',
      roles: [ { role: 'readWrite', db: 'groupeA' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    },
    {
      _id: 'groupeA.user1',
      userId: UUID('a331d77a-05c9-4f1d-84d9-cae9bed79a5d')
,
      user: 'user1',
      db: 'groupeA',
      roles: [ { role: 'roleA', db: 'groupeA' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    }
  ],
  ok: 1
}
```

➢ Pour le groupe B :

```
groupeB> db.getRoles()
{
  roles: [
    {
      _id: 'groupeB.roleB',
      role: 'roleB',
      db: 'groupeB',
      roles: [
        { role: 'userAdmin', db: 'groupeB' },
        { role: 'readWrite', db: 'groupeB' }
      ],
      isBuiltin: false,
      inheritedRoles: [
        { role: 'userAdmin', db: 'groupeB' },
        { role: 'readWrite', db: 'groupeB' }
      ]
    }
  ],
  ok: 1
}
```

```
groupeB> db.getUsers()
{
  users: [
    {
      _id: 'groupeB.Steph',
      userId: UUID('ca292498-acb1-49fc-b6b3-0ea94fb7ae95')
,
      user: 'Steph',
      db: 'groupeB',
      roles: [ { role: 'readWrite', db: 'groupeB' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    },
    {
      _id: 'groupeB.user2',
      userId: UUID('de003cd0-f36c-41c2-9ca3-68bd2d6e2d9c')
,
      user: 'user2',
      db: 'groupeB',
      roles: [ { role: 'roleB', db: 'groupeB' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
    }
  ],
  ok: 1
}
```

- **Test des Politiques :**

➢ Tester les politiques en tentant d'accéder à MongoDB avec des utilisateurs de différents groupes.

➢ Premier test avec l'utilisateur « user1 » de mon groupeA :

```
[root@localhost ~]# mongosh -u user1 groupeA
Enter password: *****
Current Mongosh Log ID: 65708ecb9aac5b9a60bcdd0d
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/groupeA?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.1
Using MongoDB:          7.0.4
Using Mongosh:          2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

groupeA> show users;
[
  {
    _id: 'groupeA.user1',
    userId: UUID('7dff08a4-def0-471f-802e-0f3114e5fd28'),
    user: 'user1',
    db: 'groupeA',
    roles: [ { role: 'userAdmin', db: 'groupeA' } ],
    mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
  }
]
groupeA> show dbs;
groupeA  72.00 KiB
groupeA> exit
```

➢ Deuxième test avec l'utilisateur « user2 » de mon groupeB :

```
[root@localhost ~]# mongosh -u user2
Enter password: *****
Current Mongosh Log ID: 65707b48e5224be4ef8c5072
Connecting to:          mongodb://<credentials>@127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.1.1
Using MongoDB:          7.0.4
Using Mongosh:          2.1.1

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

test> show dbs;
groupeB  40.00 KiB
test>
```

Partie 5 : Apache

- **Installation Apache** :

➢ Installer Apache sur votre Redhat

➢ La commande pour installer Apache sur Redhat, il suffit d'effectuer la commande suivante : *sudo yum install httpd*

```
[user@localhost ~]$ sudo yum install httpd
[sudo] Mot de passe de user :
Mise à jour des référentiels de gestion des abonnements.
Dernière vérification de l'expiration des métadonnées effectuée il y a 1:51:31 le mer. 10 janv. 2024 09:21:30.
Dépendances résolues.
================================================================================
 Paquet             Architecture
                          Version        Dépôt                           Taille
================================================================================
Installation:
 httpd              x86_64 2.4.57-5.el9   rhel-9-for-x86_64-appstream-rpms  52 k
Installation des dépendances:
 apr                x86_64 1.7.0-12.el9_3 rhel-9-for-x86_64-appstream-rpms 126 k
 apr-util           x86_64 1.6.1-23.el9   rhel-9-for-x86_64-appstream-rpms  97 k
 apr-util-bdb       x86_64 1.6.1-23.el9   rhel-9-for-x86_64-appstream-rpms  14 k
 httpd-core         x86_64 2.4.57-5.el9   rhel-9-for-x86_64-appstream-rpms 1.5 M
 httpd-filesystem   noarch 2.4.57-5.el9   rhel-9-for-x86_64-appstream-rpms  16 k
 httpd-tools        x86_64 2.4.57-5.el9   rhel-9-for-x86_64-appstream-rpms  87 k
 redhat-logos-httpd noarch 90.4-2.el9     rhel-9-for-x86_64-appstream-rpms  18 k
Installation des dépendances faibles:
 apr-util-openssl   x86_64 1.6.1-23.el9   rhel-9-for-x86_64-appstream-rpms  17 k
 mod_http2          x86_64 1.15.19-5.el9  rhel-9-for-x86_64-appstream-rpms 152 k
 mod_lua            x86_64 2.4.57-5.el9   rhel-9-for-x86_64-appstream-rpms  62 k

Résumé de la transaction
================================================================================
Installer  11 Paquets

Taille totale des téléchargements : 2.1 M
Taille des paquets installés : 5.9 M
Voulez-vous continuer ? [o/N] : o
Téléchargement des paquets :
(1/11): apr-util-bdb-1.6.1-23.el9.x86_64.rpm                    46 kB/s |  14 kB     00:00
(2/11): httpd-filesystem-2.4.57-5.el9.noarch.rpm                51 kB/s |  16 kB     00:00
(3/11): apr-util-1.6.1-23.el9.x86_64.rpm                       299 kB/s |  97 kB     00:00
(4/11): apr-util-openssl-1.6.1-23.el9.x86_64.rpm               119 kB/s |  17 kB     00:00
(5/11): mod_lua-2.4.57-5.el9.x86_64.rpm                        375 kB/s |  62 kB     00:00
(6/11): mod_http2-1.15.19-5.el9.x86_64.rpm                     786 kB/s | 152 kB     00:00
(7/11): redhat-logos-httpd-90.4-2.el9.noarch.rpm               129 kB/s |  18 kB     00:00
(8/11): httpd-2.4.57-5.el9.x86_64.rpm                          355 kB/s |  52 kB     00:00
(9/11): httpd-core-2.4.57-5.el9.x86_64.rpm                     5.9 MB/s | 1.5 MB     00:00
(10/11): httpd-tools-2.4.57-5.el9.x86_64.rpm                   527 kB/s |  87 kB     00:00
(11/11): apr-1.7.0-12.el9_3.x86_64.rpm                         846 kB/s | 126 kB     00:00
--------------------------------------------------------------------------------
```

➢ La commande pour lancer Apache : *Sudo systemctl start httpd*

➢ La commande pour vérifier son état : *Sudo systemctl status httpd*

```
[user@localhost ~]$ sudo systemctl start httpd
[user@localhost ~]$ sudo systemctl status httpd
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; disabled; preset: disabled)
     Active: active (running) since Wed 2024-01-10 11:14:09 CET; 43s ago
       Docs: man:httpd.service(8)
   Main PID: 46362 (httpd)
     Status: "Total requests: 0; Idle/Busy workers 100/0;Requests/sec: 0; Bytes served/sec:   0 B/sec"
      Tasks: 213 (limit: 10685)
     Memory: 25.5M
        CPU: 123ms
     CGroup: /system.slice/httpd.service
             ├─46362 /usr/sbin/httpd -DFOREGROUND
             ├─46431 /usr/sbin/httpd -DFOREGROUND
             ├─46432 /usr/sbin/httpd -DFOREGROUND
             ├─46433 /usr/sbin/httpd -DFOREGROUND
             └─46437 /usr/sbin/httpd -DFOREGROUND

janv. 10 11:14:09 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
janv. 10 11:14:09 localhost.localdomain httpd[46362]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using localhost.local>
janv. 10 11:14:09 localhost.localdomain systemd[1]: Started The Apache HTTP Server.
janv. 10 11:14:09 localhost.localdomain httpd[46362]: Server configured, listening on: port 80
lines 1-20/20 (END)
```

- **Création de groupe et assignation des privileges**

➢ Créer un nouveau groupe "**groupeC**"

➢ Le **groupeC** peut accéder en lecture aux tables **groupeA** et **groupeB**

➢ Le **groupeC** peut consulter le site Web mais pas le **groupeA** ni le **groupeB**

➢ La commande pour créer un groupe C dans le mongosh : *use groupeC*

```
test> use groupeC
switched to db groupeC
groupeC>
```

- Pour mon groupeC, je lui atrribue des rôles qui lui permettront d'accéder en lecture aux collections des groupes A et B :

```
test> use groupeC
switched to db groupeC
groupeC> db.createRole({
...   role: "lecture_groupeA",
...   privileges: [
...     { resource: { db: "groupeC", collection: "groupeA"
 }, actions: ["find"] }
...   ],
...   roles: []
... })
{ ok: 1 }
groupeC> db.createRole({
...   role: "lecture_groupeB",
...   privileges: [
...     { resource: { db: "groupeC", collection: "groupeB"
 }, actions: ["find"] }
... ],
... roles: []
... })
{ ok: 1 }
groupeC>
```

- Création d'un utilisateur admin dans le groupeC :

```
admin> use admin
already on db admin
admin> db.createUser({
...   user: "user3",
...   pwd: "12345",
...   roles: []
... })
{ ok: 1 }
admin>
```

- On vérifie les rôles de l'utilisateur du groupeC, nous pouvons voir ci-dessous que l'utilisateur *user3* du groupeC possède bien les privilèges qu'on lui a attribué, c'est-à-dire qu'il peut accéder en lecture aux tables groupeA et groupeB :

```
admin> db.grantRolesToUser(
...   "user3",
...   [
...     { role: "lecture_groupeA", db: "groupeC" },
...     { role: "lecture_groupeB", db: "groupeC" }
...   ]
... )
{ ok: 1 }
admin>
```

- Dans l'image ci-dessous, nous pouvons également confirmer que le groupeC peut consulter le site Web, mais que cela n'est pas le cas du groupeA ni du groupeB :

```
<Directory /var/www/html/groupeC>
        Require group groupeC
</Directory>

<Directory /var/www/html/groupeA>
        Require all denied
</Directory>

<Directory /var/www/html/groupeB>
        Require all denied
</Directory>
```

- Petit test de l'accès à Apache via l'interface web :

Partie 6 : Quotas et ACL (j'ai pas eu le temps de la faire)

- **Faire une analyse du disque**

- **Activation des quotas sur les systèmes de fichiers :**
- Premièrement, vous devez activer les quotas sur une partition, pour se faire, il faut éditer le fichier /etc/fstab et ajouter les options usrquota et grpquota pour le système de fichiers concerné :

- *Nano /etc/fstab*
- */dev/sdb2 /home ext4 defaults,usrquota,grpquota*

- Il faut ensuite remonter la partition. Pour cela nous pouvons utiliser mount (monter des systèmes de fichier) avec l'option -o qui démonte et remonte dans la foulée :

- Mount -o remount /home
- Mount /dev/sdb2 on /home type ext (rw, usrquota, grpquota)

Analyse du disque :

- Pour l'analyse du disque, qui servira à savoir quelle est l'utilisation du système de fichiers, c'està-dire quel volume de données est utilisé par quels utilisateurs et quels groupes. Utilisez la commande suivante :

- *Quotacheck -cug /home*

Après cette analyse, initialisez une base de quotas qui servira ensuite à affecter les quotas aux utilisateurs et aux groupes :

- *Quotacheck -avug*

Attribuer un quota en M° pour les groupes et les utilisateurs

- Nous devons d'abord activer les quotas pour un utilisateurs ou un groupe, dans notre cas nous allons entrez les commandes suivantes :

  Sudo edquota -g GroupeA
  Sudo edquota -g GroupeB

- Pour activer les quotas pour des utilisateurs nous changerons le *-g* par *-u*, par exemple :

  Sudo edquota -u Utilisateur1
  Sudo edquota -u Utilisateur2

- Nous pouvons désormais, définir les quotas en mégaoctets, pour se faire, ouvrez l'éditeur avec les commandes :

  Sudo edquota -g GroupeA
  Sudo edquota -g GroupeB

- Lorsque vous exécutez la commande edquota, elle ouvre un éditeur de texte avec des informations sur les quotas en termes d'inodes et de blocs. Vous pouvez définir les quotas en mégaoctets en convertissant la taille désirée en blocs.
- Par exemple, si vous souhaitez définir un quota de 100 Mo pour le groupe "groupeA" et 200 Mo pour le groupe "groupeB" :

- Convertissez 100 Mo en blocs (1 bloc = 1024 Ko) : 100 * 1024 = 102400 blocs. - Convertissez 200 Mo en blocs : 200 * 1024 = 204800 blocs.

- Par exemple, dans l'éditeur, vous pouvez ajuster les valeurs 'soft' et 'hard' pour le groupeA comme suit :

```
Disk quotas for group groupeA (gid XXXX):
  Filesystem           blocks       soft       hard     inodes     soft     hard
  /dev/sdXn            102400          0          0          4        0        0
```

- Après avoir exécutez les commandes pour le groupeA et le groupeB, vous pouvez vérifier les quotas pour tous les groupes sur le système de fichiers spécifié avec la commande suivante :

```
sudo repquota -g
```

- En dernier lieu, il est nécessaire d'activer les quotas avec la commande suivante :

- *Quotaon*

Attribuer des ACL aux utilisateurs

- Pour modifier les droites utilisez la commande setfaci de la manière suivante :

setfacl -m u:<utilisateur>:<droits>,g:<groupe>:<droits> fichier

- L'option -m veut dire modify.
- Vous pouvez mettre les droits pour un utilisateur supplémentaire, ou un groupe, ou les deux en même temps, ou plusieurs groupes ou plusieurs utilisateurs en même temps.
- Les droits s'appliquent en écriture alphabétique (type rwx).

- Si j'adapte la commande à ma situation, la commande est la suivante :

Setfacl -m u :Utilisateur1 :rwx, u :Utilisateur2 :rx/home

Vous pouvez afficher les ACL pour un fichier avec la

Conclusion

- Red Hat, l'une des distributions Linux les plus populaires, offre une robuste plateforme pour la gestion des systèmes et des données, incluant l'administration des groupes. Avec Red Hat, l'administration des groupes implique la gestion des utilisateurs et de leurs permissions, garantissant ainsi une sécurité et une organisation efficaces au sein du système. Pour la manipulation de MongoDB, une base de données NoSQL populaire, Red Hat propose des outils et des supports adéquats. Il permet d'installer, de configurer et de gérer MongoDB de manière sécurisée, en intégrant des mécanismes de sécurité tels que SELinux.
  Parlant de sécurité, SELinux (Security-Enhanced Linux) joue un rôle crucial dans la protection des données et des services, y compris MongoDB et Apache, en imposant des politiques de sécurité strictes et en contrôlant les accès aux ressources du système.
  En ce qui concerne Apache, le serveur web largement utilisé, Red Hat fournit des directives et des modules de sécurité pour protéger les serveurs web contre les vulnérabilités et les attaques, renforçant ainsi la sécurité des applications déployées sur Apache.
  En outre, Red Hat propose des mécanismes de gestion des quotas et des contrôles d'accès (ACL) pour limiter l'utilisation des ressources système et contrôler l'accès aux fichiers et aux répertoires, offrant ainsi une protection supplémentaire et une gestion fine des permissions au niveau du système de fichiers.
  En somme, Red Hat offre un écosystème complet pour la gestion sécurisée des données et des systèmes, incluant des outils avancés pour l'administration des groupes, la manipulation de bases de données comme MongoDB, la sécurité avec SELinux et Apache, ainsi que la gestion des quotas et des ACL, garantissant ainsi la fiabilité et la sécurité des infrastructures informatiques.