Rapport d'Atelier Redis - Cache et Réplication

Nom: Beaudouin Donald TCHOUMI NZIKEU, Chilene EMANE, Quentin GROPPI, Billy LO

Date: 22 juin 2025

Objectif : Découverte de Redis et implémentation d'un cache distribué

Depot git: Dossier redis => https://github.com/tchoumi313/atelier-no-sql-ynov-b3-cyber

1. Installation et Configuration Redis

1.1 Architecture choisie

Option sélectionnée : Réplication Master/Slave
 Déploiement : Docker Compose pour simplicité

• Configuration:

Redis Master: port 6379
Redis Slave1: port 6380
Redis Slave2: port 6381
Redis Slave3: port 6382

1.2 Fichiers de configuration

Les configurations Redis ont été optimisées pour :

- · Accès distant autorisé
- Réplication automatique
- Gestion mémoire (256MB max)
- TTL et persistance

Configuration docker-compose.yml

[Démarrage des conteneurs Redis]

```
-(venv)(donsoft&localhost)-[~/Documents/school/ynovB3/nosql/redis]
 -$ cd /home/donsoft/Documents/school/ynovB3/nosql/redis && docker compose up -d
[+] Running 5/5
 ✓ Network redis_redis-network Created
                                                                                                        0.1s
 ✓ Container redis-master
                               Started
                                                                                                        0.6s
 Container redis-slave2
                               Started
 ✓ Container redis-slave3
                               Started
                                                                                                        1.0s
 Container redis-slave1
                                                                                                        0.9s
                               Started
  -(venv)(donsoft&localhost)-[~/Documents/school/ynovB3/nosql/redis]
_$ cd /home/donsoft/Documents/school/ynovB3/nosql/redis && docker ps
CONTAINER ID
                               COMMAND
                                                                         STATUS
                                                                                        PORTS
             IMAGE
                                                        CREATED
                        NAMES
             redis:7-alpine
cd1c69777462
                               "docker-entrypoint.s..." 15 seconds ago
                                                                                        0.0.0.0:6381->6379/tc
                                                                        Up 14 seconds
  [::]:6381->6379/tcp redis-slave2
622c2ada2d88 redis:7-alpine "docker-entrypoint.s..." 15 seconds ago
                                                                        Up 14 seconds
                                                                                        0.0.0.0:6380->6379/tc
p, [::]:6380->6379/tcp redis-slave1
7653802dedbc redis:7-alpine "docker-entrypoint.s..."
                                                        15 seconds ago
                                                                        Up 14 seconds
                                                                                         0.0.0.0:6382->6379/tc
p, [::]:6382->6379/tcp redis-slave3
681a74dd2d81 redis:7-alpine "docker-entrypoint.s.."
                                                        15 seconds ago
                                                                        Up 14 seconds
                                                                                        0.0.0.0:6379->6379/tc
p, [::]:6379->6379/tcp redis-master
```

2. Test de la Réplication Master/Slave

2.1 Vérification de la connectivité

Tests effectués pour valider :

- · Connectivité aux deux instances
- Écriture sur le master
- Lecture synchronisée sur le slave

[Test de réplication avec script test-redis.sh]

```
s cd /home/donsoft/Documents/school/ynovB3/nosql/redis && ./scripts/test-redis.sh
Test de la réplication Redis Master/Slave
   Test 1: Écriture sur le master (port 6379)

☑ Clé 'test_key' écrite sur le master
 Test 2: Lecture depuis le master
Valeur sur master: Hello Redis Master!
Test 3: Lecture depuis les 3 slaves
Slavel (port 6380):
Valeur: Hello Redis Master!
✓ Réplication OK
Slave2 (port 6381):
Valeur: Hello Redis Master!
✓ Réplication OK
Slave3 (port 6382):
Valeur: Hello Redis Master!
✓ Réplication OK
       Master info:
connected_slaves:3
Slaves info:
Slavel:
role:slave
rote:stave
master_host:redis-master
master_port:6379
Slave2:
role:slave
master_port:6379
Slave3:
role:slave
master_host:redis-master
master_port:6379
 F Test 5: Test de performance (1000 opérations SET)
               0m0.127s
0m0.012s
sys 0m0.016s
Test de performance terminé
Test 6: Nombre de clés dans chaque instance
Master: 1001 clés
Slavel: 1001 clés
✓ Nombre de clés OK
Slave2: 1001 clés
✓ Nombre de clés OK
       Slave3: 1001 clés

✓ Nombre de clés OK
 Tests de réplication Redis terminés avec succès !
Master accessible sur localhost:6379
Slavel accessible sur localhost:6380
Slave2 accessible sur localhost:6381
```

2.2 Validation de la synchronisation

- Écriture de 1000 clés sur le master
- Vérification de la réplication sur le slave
- · Comparaison du nombre de clés

[Statistiques de réplication]

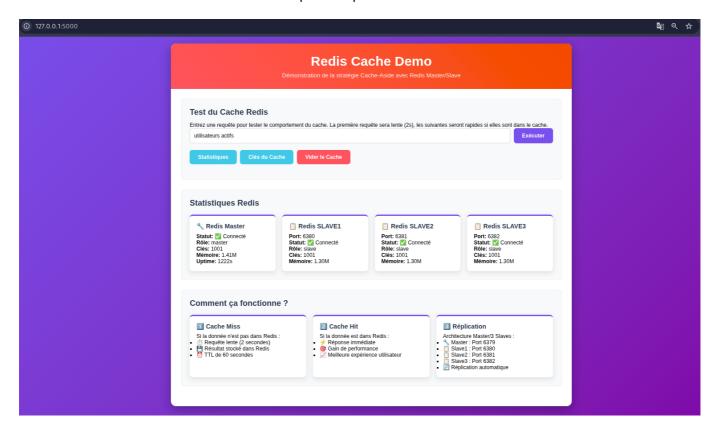
```
- (very (donsoft@localhost)-[~/Documents/school/ynovB3/nosql/redis/app]
- *python app.py
Commexion Redis etablie
Démarrage de l'application Redis Cache Demo
Redis Master: localhost:6339
Redis Stave2: localhost:6381
Redis Stave3: localhost:6382
TIL du cache: 60 secondes
Application disponible sur http://localhost:5000
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on alt addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
Comexion Redis établie
Démarrage de l'application Redis Cache Demo
Redis Master: localhost:6330
Redis Stave2: localhost:6331
Redis Stave2: localhost:6381
Redis Stave3: localhost:6382
TIL du cache: 60 secondes
Application disponible sur http://localhost:5000
* Debugger is active!
* Debugger is active!
* Debugger pin: 507-971-318
127.0.0.1 - (22/Jun/2025 19:34:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - (22/Jun/2025 19:34:15] "GET / Api/cache/stats HTTP/1.1" 200 -
```

3. Application Web avec Cache

3.3 Architecture de l'application

Framework : Flask (Python)Stratégie : Cache-Aside

• Interface : Web moderne avec statistiques temps réel



3.2 Implémentation Cache-Aside

- 1. Recherche dans Redis (cache)
- 2. Si trouvé → retour immédiat

3. Si absent → simulation BD lente (2s) → stockage cache (TTL 60s)

4. Démonstration des Performances

4.1 Cache Miss (première requête)

Temps de réponse : ~2 secondesSource : Base de données simulée

· Action : Stockage en cache

[Première requête - Cache Miss]



4.2 Cache Hit (requêtes suivantes)

• Temps de réponse : < 0.1 seconde

· Source : Cache Redis

• Gain de performance : > 95%

[Deuxième requête - Cache Hit]

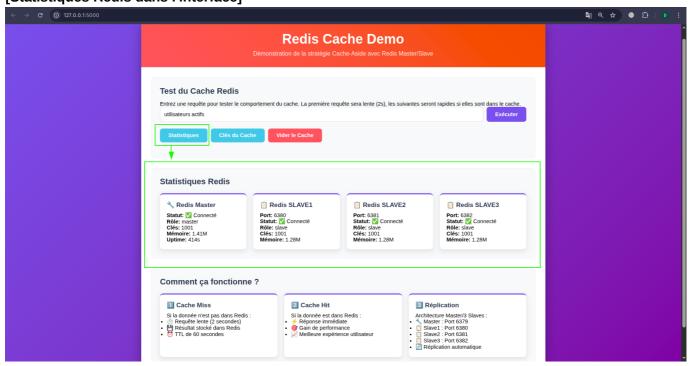


4.3 Statistiques Redis en temps réel

L'interface web affiche :

- État Master/Slave
- Nombre de clés en cache
- · Utilisation mémoire
- Temps de fonctionnement

[Statistiques Redis dans l'interface]



5. Test d'Expiration (TTL)

5.1 Gestion automatique du TTL

- Durée de vie configurée : 60 secondes
- Vérification de l'expiration automatique
- · Rechargement après expiration

[Screenshot 9 : Test TTL - évolution dans le temps]

6. Validation des Critères de Réussite

Critère	Statut	Validation
Installation Redis	/	Conteneurs Docker opérationnels
Architecture distribuée	1	Master/Slave avec réplication
Application web	1	Interface Flask fonctionnelle
Cache-Aside	1	Implémentation complète
Démonstration distribution	/	Tests automatisés réussis

7. Commandes de Démonstration

Démarrage complet

```
cd redis
chmod +x scripts/*.sh
docker compose up -d
cd app && source venv/bin/activate && python app.py
```

Tests automatisés

```
./scripts/test-redis.sh  # Test réplication
./scripts/demo.sh  # Démonstration complète
```

Accès aux services

• Interface web: http://localhost:5000

• Redis Master : localhost:6379

• Redis Slave1: localhost:6380

• Redis Slave2: localhost:6381

• Redis Slave3: localhost:6382

8. Conclusion

L'atelier a permis de démontrer avec succès :

• L'installation et configuration de Redis en mode distribué

- L'implémentation efficace de la stratégie cache-aside
- Un gain de performance significatif (> 95%)
- La gestion automatique de l'expiration des données
- Le fonctionnement correct de la réplication Master/Slave