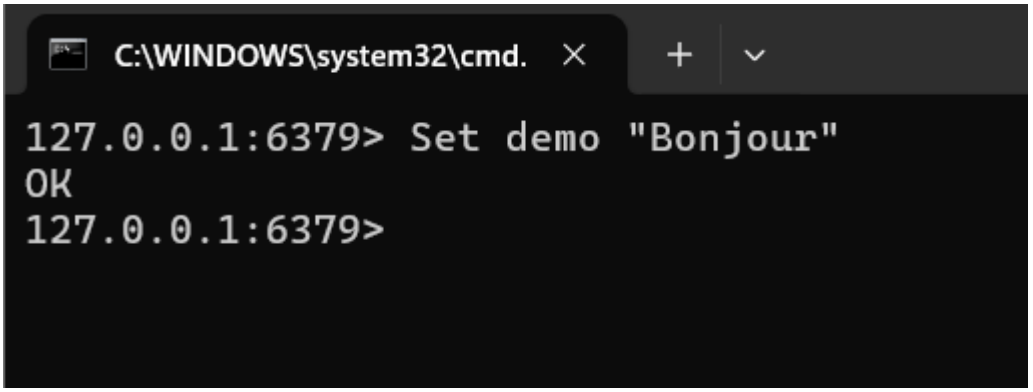
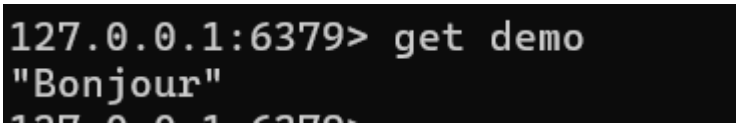
Jouini Seyfeddine - 3 ING INFO

TP1 : Prise en main de REDIS

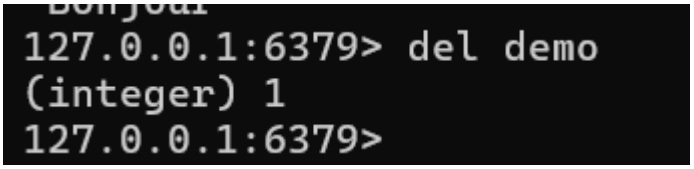
Définition de variable : avec **Set** :



Récupération de la valeur de la variable : avec **get** :

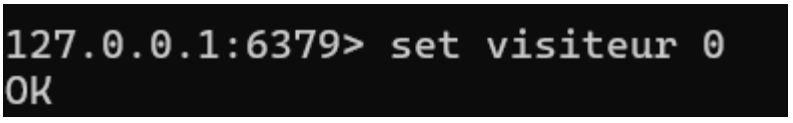


suppression d'une variable : avec **del**

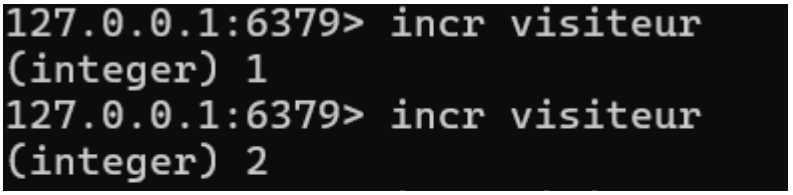


Si je veux calculer le nombre de visiteurs :

je défini une variable globale :



Incrémentation de la variable : avec **incr** :



décrémentation : avec **decr** :

```
127.0.0.1:6379> decr visiteur
(integer) 4
127.0.0.1:6379> decr visiteur
(integer) 3
```

RQ : Les données dans redis sont en RAM pas dans le disque dur.

On peut définir la durée de vie d'une clé :

```
127.0.0.1:6379> set macle val
OK
127.0.0.1:6379> ttl macle
(integer) -1
```

ttl: permet d'avoir la durée de vie d'une clé.

=>Ici ça retourne -1 ce qui veut dire que c'est non défini.

Pour définir la durée de vie : avec expire + nombre de secondes :

```
127.0.0.1:6379> EXPIRE macle 120
(integer) 1
127.0.0.1:6379> ttl macle
(integer) 116
127.0.0.1:6379>
```

Listes :

RPUSH pour ajouter des éléments à droite et **PUSH** pour ajouter à gauche.

```
127.0.0.1:6379> RPUSH mescours "BDA"
(integer) 1
127.0.0.1:6379> RPUSH mescours "Web"
(integer) 2
```

Pour voir les données : **LANGE Liste start stop**

si je mets 0 -1: ça donne tous les éléments.

```
127.0.0.1:6379> LRANGE mescours 0 -1
1) "BDA"
2) "Web"
127.0.0.1:6379> LRANGE mescours 0 1
1) "BDA"
2) "Web"
127.0.0.1:6379> LRANGE mescours 0 0
1) "BDA"
127.0.0.1:6379>
```

Suppression : avec **LPOP** ou **RPOP** (gauche et droite)

```
127.0.0.1:6379> LPOP mescours
"BDA"
127.0.0.1:6379> RPOP mescrous
(nil)
127.0.0.1:6379> LRANGE mescours 0 -1
1) "Web"
2) "Math"
127.0.0.1:6379> RPOP mescours
"Math"
127.0.0.1:6379>
```

Set :

C’est une liste sans redondance.

Ajout d'éléments : avec **SADD** :

```
127.0.0.1:6379> SADD users "Seyf"
(integer) 1
127.0.0.1:6379> SADD users "Samir"
(integer) 1
127.0.0.1:6379> SADD users "Seyf"
(integer) 0
127.0.0.1:6379>
```

Affichage des éléments : **SMEMBERS**

```
127.0.0.1:6379> SMEMBERS users
1) "Seyf"
2) "Samir"
127.0.0.1:6379>
```

RQ : pas de indice dans set (ordre ne compte pas)

Suppression : **SREM**

```
127.0.0.1:6379> SREM users "Seyf"
(integer) 1
127.0.0.1:6379>
```

Union de deux ensembles :

On définit un autre **SET** :

```
127.0.0.1:6379> SADD otherUsers "Antoine"
(integer) 1
127.0.0.1:6379> SADD otherUsers "Ines"
(integer) 1
127.0.0.1:6379> SADD otherUsers "Melissa"
(integer) 1
127.0.0.1:6379> SMEMBERS otherUsers
1) "Melissa"
2) "Antoine"
3) "Ines"
127.0.0.1:6379>
```

Union : avec **SUNION**

```
127.0.0.1:6379> SUNION users otherUsers
1) "Samir"
2) "John"
3) "Melissa"
4) "Seyf"
5) "Antoine"
6) "Ines"
127.0.0.1:6379>
```

SET Ordonnées :

On va avoir pour chaque élément une clé et une valeur : ordonnancement par rapport à la clé.

Ajout de valeur avec **ZADD** :

```
127.0.0.1:6379> ZADD score 19 "Seyf"
(integer) 1
127.0.0.1:6379> ZADD score 18 "Samir"
(integer) 1
127.0.0.1:6379> ZADD score 8 "Melissa"
(integer) 1
127.0.0.1:6379> ZADD score 20 "Nozha"
(integer) 1
127.0.0.1:6379>
```

Affichage avec **ZRANGE** Liste Start End : **Ordre croissant** :

```
127.0.0.1:6379> ZRANGE score 0 -1
1) "Melissa"
2) "Samir"
3) "Seyf"
4) "Nozha"
127.0.0.1:6379>
```

Ordre décroissant : avec **ZREVRANGE** :

```
127.0.0.1:6379> ZREVRANGE score 0 -1
1) "Nozha"
2) "Seyf"
3) "Samir"
4) "Melissa"
127.0.0.1:6379>
```

Avoir la clé (rang) d'un élément : avec **ZRANK**

```
127.0.0.1:6379> ZRANK score "Seyf"
(integer) 2
127.0.0.1:6379>
```

Hash : une liste de valeurs :

HSET Nom Attribut valeur => défini attribut par attribut :

```
127.0.0.1:6379>  HSET user:1 name "Seyf"
(integer) 1
127.0.0.1:6379>  HSET user:1 age 25
(integer) 1
127.0.0.1:6379>  HSET user:1 email seyf@gmail.com
(integer) 1
```

Récupérer le tout : avec **HGETALL**

```
127.0.0.1:6379> HGETALL user:1
1) "name"
2) "Seyf"
3) "age"
4) "25"
5) "email"
6) "seyf@gmail.com"
127.0.0.1:6379>
```

Récupérer que les valeurs : **HVAL** :

```
127.0.0.1:6379> HVALS user:1
1) "Seyf"
2) "25"
3) "seyf@gmail.com"
127.0.0.1:6379>
```

Définition du Hash en 1 ligne : avec **HMSET**

```
127.0.0.1:6379> HMSET user:2 name "SAMIR" age 30
OK
127.0.0.1:6379> HGETALL user:2
1) "name"
2) "SAMIR"
3) "age"
4) "30"
127.0.0.1:6379>
```

Opérations sur les éléments de HASH :

Incrémentation de valeur : avec **HINCRBY** :

```
127.0.0.1:6379> HINCRBY user:2 age 4
(integer) 34
127.0.0.1:6379> HGET user:2 age
"34"
127.0.0.1:6379>
```

PUB/SUB :

Utilisé pour les applications en temps réel (comme envoie de message …)

Il nous faut 2 terminaux : pour qu'ils communiquent ensemble.

User 1 va s'abonner au canal pour écouter les messages :

```
127.0.0.1:6379> SUBSCRIBE mesCours user:1
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "mesCours"
3) (integer) 1
1) "subscribe"
2) "user:1"
3) (integer) 2
```

User 2 va envoyer un message sur le canal : avec **PUBLISH** : (Broadcast)

```
127.0.0.1:6379> PUBLISH mesCours "Un nouveau cours"
(integer) 1
127.0.0.1:6379>
```

tous les abonnés vont le recevoir en temps réel :

```
1) "message"
2) "mesCours"
3) "Un nouveau cours"
```

Message que pour un user : **PUBLISH** user:1 message (pas de canal)

```
127.0.0.1:6379> PUBLISH user:1 "Bonjour user1"
(integer) 1
```

Un user peut d'abonner à plusieurs canaux : avec **PSUBSCRIBE**:

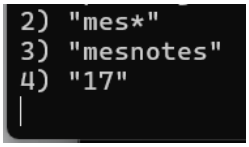
User 1 va s'abonner à tous les canaux commencent par mes :

```
127.0.0.1:6379> PSUBSCRIBE mes*
Reading messages... (press Ctrl-C to quit)
1) "psubscribe"
2) "mes*"
3) (integer) 1
```

Envoie:

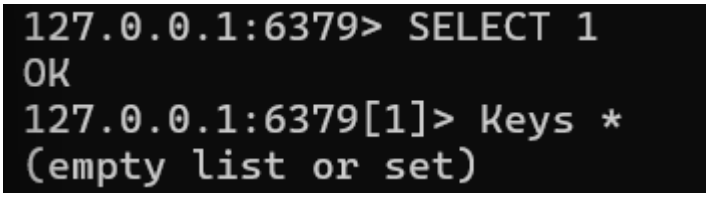
```
127.0.0.1:6379> PUBLISH mesnotes 17
(integer) 1
127.0.0.1:6379>
```

Réception :



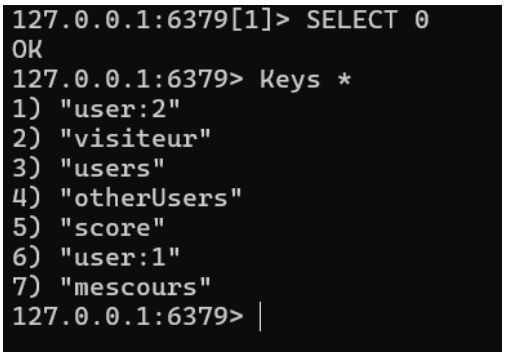
RQ : Redis se connecte sur 16 bases de données : par défaut sur la 0

Si je veux me connecter à une autre base : avec **SELECT** :



On voit que les clés définies précédemment ne sont plus là.

Quand on revient à la base 0 on retrouve tout:



Panne : En cas de panne, par défaut : pas de reprise de panne, il faut la configurer.