Résumé des Livres Redis par Niveau

🟢 Niveau 1 – Introduction à Redis

1. Redis Essentials

- **Auteurs** : Maxwell Dayvson Da Silva & Hugo Lopes Tavares
- **Éditeur** : Packt Publishing

Contenu :

- Introduction aux structures de données Redis : strings, lists, sets, hashes.
- Cas d'utilisation courants (cache, sessions, files).
- Installation et configuration de base.
- Opérations de base en CLI ou via des clients (Node.js, Python, Java).

Pour qui ?

- Développeurs débutants.
- Idéal pour une première approche de Redis.

🟡 Niveau 2 – Pratique intermédiaire

2. Learning Redis

- **Auteur** : Vinoo Das
- **Éditeur** : Packt Publishing

Contenu :

- Utilisation avancée des types de données.
- Pub/Sub, transactions, Lua scripting.
- Introduction au clustering et à la réplication.
- Surveillance de Redis et gestion de la mémoire.

Pour qui ?

- Développeurs back-end, ingénieurs logiciels.
- Ceux qui veulent intégrer Redis dans une application web ou microservice.

🔴 Niveau 3 – Redis avancé / DevOps

3. Mastering Redis

- **Auteur** : Jeremy Nelson
- **Éditeur** : Packt Publishing

Contenu :

- Architecture Redis en profondeur.
- Configuration de la haute disponibilité avec Redis Sentinel.
- Clustering Redis distribué.
- Performance tuning et surveillance.
- Cas d'utilisation complexes : leaderboard, file prioritaire, analytics.

Pour qui ?

- DevOps, architectes cloud, SRE.
- Projets à fort trafic avec exigences de performance et de résilience.

🗺️ Récapitulatif

Niveau	Livre	Public cible	Ce que tu apprends
1	Redis Essentials	Débutant	Types de données, cas d'usage de base
2	Learning Redis	Intermédiaire / dev	Pub/Sub, transactions, clustering
3	Mastering Redis	Avancé / DevOps	HA, Sentinel, clustering, performance

I. How it work

Redis est un magasin de données open source, en mémoire, de type clé-valeur. Il est souvent utilisé comme cache d'application ou base de données à réponse rapide, offrant des performances élevées grâce au stockage en mémoire. Redis est une base de données NoSQL, ce qui signifie qu'elle ne suit pas le modèle relationnel traditionnel.

🧱 1. Types de données Redis courants

- 🔹 **Strings (Chaînes)**
  - **Description** : Type le plus basique pour stocker du texte, des nombres, ou même des données binaires.
  - **Utilisations** : Pseudonymes, emails, compteurs de pages, images encodées.
  - **Avantages** : Très simple, supporte des opérations atomiques comme `INCR`.

conn.set("username", "alice");

conn.get("username");

- 🔹 **Lists (Listes)**
  - **Description** : Listes ordonnées de chaînes, triées par ordre d'insertion.
  - **Utilisations** : Files de messages, logs d'activités, listes d'accès récents.
  - **Avantages** : Insertion/suppression rapide aux deux extrémités. Parfait pour des structures FIFO ou LIFO.

```
conn.lpush("events", "login");

conn.lrange("events", 0, -1);
```

🔹 **Sets (Ensembles)**

- **Description** : Collections non ordonnées d'éléments uniques.
- **Utilisations** : Tags, IPs visitées, liste d'amis sans doublons.
- **Avantages** : Vérification rapide d'existence, opérations d'ensembles (union, intersection...).

```
conn.sadd("tags", "redis");

conn.smembers("tags");
```

🔹 **Sorted Sets (Ensembles triés)**

- **Description** : Comme les ensembles, mais avec un score pour chaque élément.
- **Utilisations** : Classements (leaderboards), files de priorité.
- **Avantages** : Triés automatiquement par score, accès par plage de scores.

```
conn.zadd("leaderboard", "alice", 100);

conn.zrange_withscores("leaderboard", 0, -1);
```

🔹 **Hashes (Hachages)**

- **Description** : Tableau clé-valeur, comme des dictionnaires.
- **Utilisations** : Profils utilisateurs, attributs de session.
- **Avantages** : Lecture/écriture efficace de plusieurs champs en une seule commande.

```
conn.hset("user:100", "email", "alice@example.com");

conn.hgetall("user:100");
```

Bitmaps

- **Description** : Tableaux de bits (0 ou 1), pour stocker des booléens efficacement.
- **Utilisations** : Indicateurs de fonctionnalités actives, connexions utilisateurs.
- **Avantages** : Très économique en mémoire.

```
conn.setbit("features", 0, true);

conn.getbit("features", 0);
```

HyperLogLogs

- **Description** : Structure probabiliste pour estimer le nombre d'éléments uniques.
- **Utilisations** : Visiteurs uniques d'un site, analytics massives.
- **Avantages** : Très peu gourmand en mémoire, parfait pour gros volumes.

```
conn.pfadd("pageviews", "user1");

conn.pfcount("pageviews");
```

✅ 2. Bonnes pratiques par cas d'usage

Scénario	Type recommandé	Pourquoi ?
Sessions utilisateur	Hash	Stocker ID, token, timestamp facilement.
File d'attente ou messages temps réel	List	Comportement FIFO naturel.
Compteurs ou limitations de fréquence	String	Supporte les incréments atomiques.
Classement ou score	Sorted Set	Tri automatique selon le score.
Données uniques (tags, catégories)	Set	Évite les doublons, rapide à consulter.
Objet multi-attributs	Hash	Représentation naturelle d'un objet.
États booléens (feature toggles)	Bitmap	Économie de mémoire pour indicateurs activés/désactivés.

II. Why are we using it

Voici quelques raisons courantes pour lesquelles on utilise Redis :

- **Mise en cache:**
  Redis excelle dans la mise en cache de données, réduisant la charge sur les bases de données principales et améliorant les performances des applications.
-

En résumé, Redis est un outil puissant et polyvalent pour améliorer les performances et la réactivité des applications, en particulier dans les situations où la rapidité d'accès aux données est cruciale.

Stockage de sessions:
Redis peut servir de stockage de sessions distribué, permettant à différents serveurs d'applications de partager les mêmes données de session.

Pub/Sub (publication/abonnement):
Redis permet de mettre en place des systèmes de messagerie et de chat en temps réel grâce à son modèle de publication et d'abonnement.

Files d'attente:
Redis peut être utilisé pour gérer des files d'attente de tâches, permettant aux applications de traiter des tâches de manière asynchrone.

Classements:
Les structures de données de Redis, comme les ensembles triés, facilitent la création et la gestion de classements pour les jeux, les réseaux sociaux, etc.

Analyse en temps réel:
Redis permet de traiter des données en temps réel, ce qui est utile pour des applications comme la surveillance de flux de données ou l'analyse de données de capteurs.

Haute disponibilité et évolutivité:
Redis offre des options pour assurer la haute disponibilité et l'évolutivité horizontale grâce au sharding et à d'autres fonctionnalités.

III. Some examples or use cases