Gestion d'un système de bases de données

Halim Djerroud

LISV
Laboratoire d'ingénierie
des systèmes de Versailles

UVSQ
université PARIS-SACLAY

révision : 0.1

Plan du cours

Introduction

Gestion d'un système de bases de données

Objectifs du chapitre :

- Comprendre le rôle d'un SGBD dans l'infrastructure réseau
- Identifier les principaux SGBD du marché
- Maîtriser l'architecture client-serveur d'un SGBD

Contexte professionnel

- **Le professionnel RT** peut être amené à installer et administrer un SGBD.
- **Le SGBD = un service réseau à part entière** :
  - à l'instar d'un serveur web, DNS ou DHCP,
  - nécessite une intégration dans l'infrastructure réseau,
  - doit respecter les contraintes de sécurité et de disponibilité.
- **Missions principales** :
  - Installation et configuration,
  - Gestion des comptes et des droits d'accès,
  - Sécurisation des données et des connexions,
  - Sauvegarde et restauration.

Rappels : Qu'est-ce qu'une base de données ?

- **Définition** : collection organisée de données structurées, stockées électroniquement.
- **Modèle relationnel** : données organisées en tables (relations).
- **Composants principaux** :
  - Tables (relations),
  - Colonnes (attributs),
  - Lignes (tuples),
  - Clés primaires et étrangères,
  - Contraintes d'intégrité.
- **Langage SQL** : interface standardisée pour interroger et manipuler les données.

Fichiers plats vs SGBD

Fichiers plats :

- − Redondance des données
- − Risques d'incohérences
- − Accès concurrent difficile
- − Sécurité limitée
- − Pas de gestion transactionnelle

SGBD :

- \+ Élimination de la redondance
- \+ Intégrité des données
- \+ Gestion de la concurrence
- \+ Sécurité avancée
- \+ Transactions ACID

Le SGBD comme service réseau

- **Architecture client-serveur** :
  - Serveur SGBD : gère les données, traite les requêtes,
  - Clients : applications, utilisateurs, services web.
- **Avantages** :
  - Accès simultané par plusieurs utilisateurs/applications,
  - Centralisation des données,
  - Gestion centralisée de la sécurité,
  - Partage de ressources.
- **Communication réseau** : protocoles spécifiques sur ports dédiés.

Architecture réseau à 3 niveaux

- **Niveau 1 - Présentation** :
  - Interface utilisateur (navigateur web, application cliente),
  - Affichage et interaction.
- **Niveau 2 - Application** :
  - Serveur web/applicatif,
  - Logique métier, traitement des requêtes.
- **Niveau 3 - Données** :
  - Serveur SGBD,
  - Stockage et gestion des données.

Positionnement sécurisé

Le SGBD est placé dans un segment réseau protégé (DMZ interne, VLAN dédié).

Panorama des SGBD relationnels

- **MySQL / MariaDB** :
  - Open source (GPL), très populaire pour le web,
  - MariaDB = fork communautaire de MySQL,
  - Port par défaut : 3306.
- **PostgreSQL** :
  - Open source, SGBD objet-relationnel,
  - Conformité SQL stricte, extensible,
  - Port par défaut : 5432.
- **Oracle Database** :
  - Commercial, grandes entreprises,
  - Port par défaut : 1521.
- **Microsoft SQL Server** :
  - Commercial, écosystème Windows/.NET,
  - Port par défaut : 1433.

MySQL / MariaDB

Caractéristiques :

- SGBD le plus populaire (web)
- Open source (GPL)
- Performant en lecture
- Écosystème riche (phpMyAdmin)

Cas d'usage :

- Applications web
- PME/startups
- Prototypage rapide

Points forts :

+ Simplicité
+ Communauté
+ Documentation

Limites :

− Fonctionnalités avancées limitées
− Conformité SQL partielle

PostgreSQL

Caractéristiques :

- SGBD objet-relationnel
- Licence PostgreSQL (libre)
- Conformité SQL stricte
- Extensible (plugins, types personnalisés)

Cas d'usage :

- Applications d'entreprise
- Données géospatiales (PostGIS)
- Finance, scientifique

Points forts :

+ Robustesse
+ Standards SQL
+ Fonctionnalités avancées

Limites :

− Courbe d'apprentissage
− Configuration plus complexe

Architecture client-serveur d'un SGBD

- **Serveur SGBD** :
  - Processus daemon (mysqld, postgres),
  - Écoute sur un port réseau,
  - Gère les connexions entrantes,
  - Traite les requêtes SQL,
  - Gère les transactions et la concurrence.
- **Clients** :
  - Outils CLI : mysql, psql,
  - Interfaces graphiques : phpMyAdmin, pgAdmin,
  - Applications web/métier,
  - Connecteurs/drivers : PHP, Python, Java, etc.

Protocoles et ports réseau

- **Protocole de transport** : TCP (connexion fiable).
- **Ports par défaut** :
  - MySQL/MariaDB : 3306/tcp,
  - PostgreSQL : 5432/tcp,
  - Oracle : 1521/tcp,
  - SQL Server : 1433/tcp.
- **Configuration réseau** :
  - Définir les interfaces d'écoute (localhost, IP publique),
  - Configurer le pare-feu (ouverture des ports),
  - Sécuriser avec SSL/TLS pour les connexions distantes.

Besoins en ressources

- **CPU** :
  - Traitement des requêtes complexes,
  - Gestion de la concurrence.
- **RAM** :
  - Cache des données (buffer pool),
  - Requêtes en cours,
  - Recommandation : 2 Go minimum, 8 Go+ pour production.
- **Stockage** :
  - Données, index, logs,
  - I/O intensif : privilégier SSD,
  - Planifier la croissance (dimensionnement).
- **Réseau** :
  - Bande passante suffisante,
  - Latence faible pour les applications critiques.

Contraintes et bonnes pratiques

- **Disponibilité** :
  - Service critique : prévoir la haute disponibilité,
  - Réplication, clustering.
- **Sécurité** :
  - Principe du moindre privilège,
  - Chiffrement des connexions (SSL/TLS),
  - Isolation réseau (VLAN, pare-feu).
- **Performance** :
  - Optimisation des requêtes,
  - Indexation appropriée,
  - Monitoring régulier.
- **Sauvegarde** :
  - Stratégie de sauvegarde régulière,
  - Tests de restauration.

Récapitulatif

- Le SGBD est un **service réseau essentiel** de l'infrastructure IT.
- Il permet un accès **centralisé, sécurisé et concurrent** aux données.
- Choix du SGBD selon les besoins : MySQL/MariaDB, PostgreSQL, Oracle, SQL Server.
- Architecture **client-serveur** avec communication sur ports dédiés.
- Nécessite une planification des **ressources** (CPU, RAM, stockage, réseau).
- La sécurité et la disponibilité sont des **priorités absolues**.

Prochaine séance

TP : Installation et configuration d'un SGBD sur Linux

Chapitre 2

Installation et configuration d'un SGBD

Objectifs du chapitre :

- Installer un SGBD sur un serveur Linux
- Configurer les paramètres de base
- Sécuriser l'accès réseau
- Maîtriser les outils d'administration

Choix du système d'exploitation

- **Linux** : plateforme privilégiée pour les SGBD
  - Stabilité, performance, sécurité,
  - Distributions courantes : Debian, Ubuntu, Rocky Linux, RHEL.
- **Windows Server** :
  - Utilisé principalement pour SQL Server,
  - Interface graphique facilitée.
- **Notre choix** : Linux (Debian/Ubuntu)
  - Open source, gratuit,
  - Largement déployé en production,
  - Documentation abondante.

Préparation du système

- **Mise à jour du système** :

```
sudo apt update
sudo apt upgrade -y
```

- **Vérification de l'espace disque** :

```
df -h
```

- **Configuration du hostname** :

```
sudo hostnamectl set-hostname sgbd-server
```

Installation de MariaDB

- **Installation via le gestionnaire de paquets** :

```
sudo apt install mariadb-server mariadb-client -y
```

- **Vérification de l'installation** :

```
mysql --version
```

- **Vérification du service** :

```
sudo systemctl status mariadb
```

Installation de PostgreSQL

- **Installation via le gestionnaire de paquets** :

```
sudo apt install postgresql postgresql-contrib -y
```

- **Vérification de l'installation** :

```
psql --version
```

- **Vérification du service** :

```
sudo systemctl status postgresql
```

Gestion du service

- **Démarrer le service** :

```
sudo systemctl start mariadb
sudo systemctl start postgresql
```

- **Arrêter le service** :

```
sudo systemctl stop mariadb
sudo systemctl stop postgresql
```

- **Redémarrer le service** :

```
sudo systemctl restart mariadb
sudo systemctl restart postgresql
```

Activation au démarrage

- **Activer le démarrage automatique** :
  - Le service démarre automatiquement au boot du système,
  - Essentiel pour un serveur de production.

```
sudo systemctl enable mariadb
sudo systemctl enable postgresql
```

- **Vérifier l'activation** :

```
sudo systemctl is-enabled mariadb
sudo systemctl is-enabled postgresql
```

Sécurisation initiale - MariaDB

- **Script de sécurisation** : mysql_secure_installation

```
sudo mysql_secure_installation
```

- **Actions réalisées** :
  - Définir un mot de passe root,
  - Supprimer les utilisateurs anonymes,
  - Désactiver la connexion root à distance,
  - Supprimer la base de données test,
  - Recharger les privilèges.

Sécurisation initiale - Recommandations

- **Mot de passe root** :
  - Complexe (majuscules, minuscules, chiffres, symboles),
  - Minimum 12 caractères,
  - Conservé dans un gestionnaire de mots de passe sécurisé.
- **Utilisateurs anonymes** : toujours supprimer.
- **Connexion root à distance** : désactiver (sécurité).
- **Base test** : supprimer (inutile en production).

Principe de sécurité

Appliquer le principe du moindre privilège dès l'installation.

Fichiers de configuration - MariaDB

- **Fichier principal** : /etc/mysql/mariadb.conf.d/50-server.cnf
- **Paramètres importants** :
  - bind-address : adresse d'écoute du serveur (`127.0.0.1` : accès local uniquement, `0.0.0.0` : accès réseau),
  - port : port d'écoute (3306 par défaut),
  - datadir : répertoire des données,
  - log_error : fichier de log des erreurs,
  - max_connections : nombre maximum de connexions simultanées.
- **Fichiers de logs** :
  - /var/log/mysql/error.log : erreurs,
  - /var/log/mysql/mysql.log : requêtes (si activé).

Fichiers de configuration - PostgreSQL

- **Répertoire de configuration** : /etc/postgresql/(version)/main/
- **Fichiers principaux** :
  - postgresql.conf : configuration générale,
  - pg_hba.conf : contrôle d'accès (Host-Based Authentication),
  - pg_ident.conf : mapping des utilisateurs système.
- **Répertoire des données** : /var/lib/postgresql/(version)/main/
- **Fichiers de logs** : /var/log/postgresql/

Configuration réseau - MariaDB

- **Par défaut** : écoute uniquement sur localhost (127.0.0.1)
- **Éditer le fichier de configuration** :

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

- **Modifier bind-address** :
  - 127.0.0.1 : accès local uniquement,
  - 0.0.0.0 : écoute sur toutes les interfaces,
  - IP spécifique : écoute sur une interface donnée.

```
bind-address = 0.0.0.0
```

Configuration réseau - PostgreSQL

- **Éditer postgresql.conf** :

```
sudo nano /etc/postgresql/15/main/postgresql.conf
```

- **Modifier listen_addresses** :

```
listen_addresses = '*'  # écoute sur toutes les interfaces
# ou
listen_addresses = '192.168.1.10'  # IP spécifique
```

- **Port d'écoute** (par défaut 5432) :

```
port = 5432
```

Configuration du pare-feu

- **Ouverture du port pour MariaDB** :

```
sudo ufw allow 3306/tcp
```

- **Ouverture du port pour PostgreSQL** :

```
sudo ufw allow 5432/tcp
```

- **Vérification des règles** :

```
sudo ufw status
```

Sécurité avancée

Limiter l'accès à des adresses IP spécifiques :

```
sudo ufw allow from 192.168.1.0/24 to any port 3306
```

Redémarrage après configuration

- **Redémarrer le service pour appliquer les changements** :

```
sudo systemctl restart mariadb
sudo systemctl restart postgresql
```

- **Vérifier que le service écoute bien** :

```
sudo netstat -tlnp | grep mysql
sudo netstat -tlnp | grep postgres
```

- **Ou avec ss** :

```
sudo ss -tlnp | grep 3306
sudo ss -tlnp | grep 5432
```

Outils CLI - MariaDB

- **Client mysql** : interface en ligne de commande

```
mysql -u root -p
```

- **Commandes utiles dans le client** :
  - SHOW DATABASES; : lister les bases,
  - USE nom_base; : sélectionner une base,
  - SHOW TABLES; : lister les tables,
  - EXIT; ou QUIT; : quitter.

Outils CLI - PostgreSQL

- **Client psql** : interface en ligne de commande

```
sudo -u postgres psql
```

- **Commandes utiles (meta-commandes)** :
  - \l : lister les bases de données,
  - \c nom_base : se connecter à une base,
  - \dt : lister les tables,
  - \du : lister les utilisateurs,
  - \q : quitter.

Premier test de connexion

- **MariaDB - connexion locale** :

```
mysql -u root -p
```

- **PostgreSQL - connexion locale** :

```
sudo -u postgres psql
```

- **Créer une base de données test** :

```
# MariaDB
CREATE DATABASE test_db;

# PostgreSQL
CREATE DATABASE test_db;
```

Outils graphiques d'administration

- **phpMyAdmin** (pour MySQL/MariaDB) :
  - Interface web populaire,
  - Installation sur serveur Apache/Nginx + PHP,
  - Gestion complète via navigateur.
- **Adminer** :
  - Alternative légère à phpMyAdmin,
  - Un seul fichier PHP,
  - Support multi-SGBD.
- **pgAdmin** (pour PostgreSQL) :
  - Interface graphique officielle,
  - Version desktop ou web,
  - Fonctionnalités avancées.

Installation de phpMyAdmin

- **Prérequis** : Apache/Nginx + PHP

```
sudo apt install apache2 php libapache2-mod-php -y
sudo apt install phpmyadmin -y
```

- **Configuration** :
  - Sélectionner le serveur web (apache2),
  - Configurer la base de données pour phpMyAdmin,
  - Définir un mot de passe.
- **Accès** : http ://IP_SERVEUR/phpmyadmin

Sécurisation de phpMyAdmin

- **Problèmes de sécurité** :
  - URL connue : /phpmyadmin,
  - Cible fréquente d'attaques par force brute,
  - Exposition d'informations sensibles.
- **Bonnes pratiques** :
  - Changer l'URL d'accès (alias Apache),
  - Activer l'authentification HTTP basique,
  - Restreindre l'accès par IP,
  - Utiliser HTTPS uniquement,
  - Désactiver root via phpMyAdmin.

Paramètres de performance

- **MariaDB - paramètres clés** :
  - innodb_buffer_pool_size : cache des données (70% RAM),
  - max_connections : nombre de connexions simultanées,
  - query_cache_size : cache des requêtes.
- **PostgreSQL - paramètres clés** :
  - shared_buffers : mémoire partagée (25% RAM),
  - work_mem : mémoire par opération de tri,
  - effective_cache_size : indication du cache OS,
  - max_connections : connexions simultanées.

Attention

Adapter les paramètres selon les ressources disponibles.

Monitoring et logs

- **Surveillance des logs** :
  - Erreurs de connexion,
  - Requêtes lentes,
  - Problèmes de performance,
  - Tentatives d'intrusion.
- **Outils de monitoring** :
  - Commandes système : top, htop, iostat,
  - SHOW STATUS (MariaDB),
  - pg_stat_activity (PostgreSQL),
  - Solutions externes : Prometheus, Grafana, Zabbix.

Vérification de l'installation

- **Liste de vérification** :
  - ✓ Service démarré et activé au boot,
  - ✓ Écoute sur le port configuré,
  - ✓ Pare-feu configuré,
  - ✓ Connexion locale fonctionnelle,
  - ✓ Logs accessibles et surveillés,
  - ✓ Sécurisation initiale effectuée.

```
# Test complet
sudo systemctl status mariadb
sudo ss -tlnp | grep 3306
mysql -u root -p -e "SHOW DATABASES;"
```

Récapitulatif

- Installation simple via gestionnaire de paquets (apt).
- Configuration réseau : bind-address, listen_addresses, ports.
- Sécurisation initiale indispensable (mysql_secure_installation).
- Gestion du service avec systemd (start, stop, enable).
- Configuration du pare-feu pour autoriser les connexions.
- Outils CLI : mysql, psql.
- Outils graphiques : phpMyAdmin, pgAdmin (à sécuriser).
- Surveillance des logs et monitoring essentiels.

Prochaine séance

Gestion des utilisateurs et des droits d'accès

Chapitre 3

Gestion des utilisateurs et des droits

Objectifs du chapitre :

- Créer et gérer des comptes utilisateurs
- Comprendre et appliquer les différents types de privilèges
- Mettre en œuvre le principe du moindre privilège
- Gérer l'accès à l'interface d'administration

Architecture des utilisateurs

- **Distinction importante** :
  - Utilisateurs système (Linux) : accès au système d'exploitation,
  - Utilisateurs SGBD : accès aux bases de données.
- **Utilisateurs SGBD** :
  - Indépendants du système d'exploitation,
  - Gérés entièrement par le SGBD,
  - Identifiés par nom d'utilisateur + hôte (MariaDB),
  - Identifiés par nom + rôle (PostgreSQL).
- **Format d'identification MariaDB** :
  - 'utilisateur'@'hôte',
  - Exemple : 'jean'@'localhost', 'admin'@'192.168.1.

Principe du moindre privilège

Définition

Chaque utilisateur ne doit disposer que des privilèges strictement nécessaires à l'accomplissement de ses tâches.

- **Avantages** :
  - Limitation des dégâts en cas de compromission,
  - Réduction des erreurs de manipulation,
  - Traçabilité des actions,
  - Conformité aux normes de sécurité.
- **Exemples** :
  - Application web : SELECT, INSERT, UPDATE, DELETE uniquement,
  - Développeur : accès à la base de développement uniquement,
  - Administrateur : tous les privilèges sur toutes les bases.

Types de privilèges - Vue d'ensemble

- **Privilèges globaux** :
  - S'appliquent à toutes les bases de données,
  - Exemples : CREATE USER, RELOAD, SHUTDOWN.
- **Privilèges de base de données** :
  - S'appliquent à une base spécifique,
  - Exemples : CREATE, DROP, ALTER.
- **Privilèges de table** :
  - S'appliquent à une table spécifique,
  - Exemples : SELECT, INSERT, UPDATE, DELETE.
- **Privilèges de colonne** :
  - S'appliquent à des colonnes spécifiques,
  - Plus rarement utilisés.

Privilèges courants - MariaDB/MySQL

Données (DML) :

- SELECT : lire les données
- INSERT : ajouter des données
- UPDATE : modifier les données
- DELETE : supprimer des données

Structure (DDL) :

- CREATE : créer objets
- ALTER : modifier objets
- DROP : supprimer objets
- INDEX : gérer les index

Administration :

- GRANT OPTION : déléguer privilèges
- RELOAD : recharger config
- SHUTDOWN : arrêter serveur
- PROCESS : voir processus
- FILE : lire/écrire fichiers
- SUPER : opérations admin

Privilège spécial :

- ALL PRIVILEGES : tous les droits

Création d'utilisateurs - MariaDB

- **Syntaxe de base** :

```
CREATE USER 'nom_utilisateur'@'hote'
IDENTIFIED BY 'mot_de_passe';
```

- **Exemples** :

```
-- Utilisateur local uniquement
CREATE USER 'appweb'@'localhost'
IDENTIFIED BY 'MotDePasse123!';
-- Utilisateur depuis un réseau
CREATE USER 'admin'@'192.168.1.%'
IDENTIFIED BY 'SecurePass456!';
-- Utilisateur depuis n'importe où (déconseillé)
CREATE USER 'dev'@'%'
IDENTIFIED BY 'DevPass789!';
```

Création d'utilisateurs - PostgreSQL

- **Syntaxe de base** :

```
CREATE USER nom_utilisateur
WITH PASSWORD 'mot_de_passe';
```

- **Exemples avec options** :

```
-- Utilisateur simple
CREATE USER appweb WITH PASSWORD 'Pass123!';
-- Utilisateur avec droit de création de base
CREATE USER dev WITH PASSWORD 'DevPass!'
CREATEDB;
-- Utilisateur avec capacités de superuser
CREATE USER admin WITH PASSWORD 'AdminPass!'
SUPERUSER;
-- Utilisateur avec limite de connexions
CREATE USER readonly WITH PASSWORD 'ReadPass!'
CONNECTION LIMIT 5;
```

Attribution de privilèges - MariaDB

- **Syntaxe GRANT** :

```
GRANT privileges ON base.table
TO 'utilisateur'@'hote';
```

- **Exemples** :

```
-- Tous les droits sur une base
GRANT ALL PRIVILEGES ON mabase.*
TO 'appweb'@'localhost';

-- SELECT uniquement sur une table
GRANT SELECT ON mabase.clients
TO 'readonly'@'localhost';
-- Plusieurs privilèges
GRANT SELECT, INSERT, UPDATE ON mabase.produits
TO 'appweb'@'localhost';
```

Attribution de privilèges - PostgreSQL

- **Attribution sur une base de données** :

```
GRANT CONNECT ON DATABASE mabase TO appweb;
GRANT ALL PRIVILEGES ON DATABASE mabase TO admin;
```

- **Attribution sur des tables** :

```
-- Se connecter d'abord à la base
\c mabase
-- Privilèges sur une table
GRANT SELECT ON clients TO readonly;
GRANT SELECT, INSERT, UPDATE, DELETE
ON produits TO appweb;

-- Privilèges sur toutes les tables d'un schéma
GRANT ALL PRIVILEGES ON ALL TABLES
IN SCHEMA public TO admin;
```

Appliquer les changements - MariaDB

- **Recharger les privilèges** :

```
FLUSH PRIVILEGES;
```

Important

Après avoir modifié directement les tables de privilèges (mysql.user, mysql.db, etc.), il est nécessaire d'exécuter FLUSH PRIVILEGES.
Ce n'est pas nécessaire après GRANT ou REVOKE.

Consultation des utilisateurs

MariaDB :

```
-- Lister les utilisateurs
SELECT User, Host
FROM mysql.user;

-- Voir les privilèges
-- d'un utilisateur
SHOW GRANTS FOR
'appweb'@'localhost';

-- Utilisateur actuel
SELECT USER();
```

PostgreSQL :

```
-- Lister les utilisateurs
\du

-- ou en SQL
SELECT usename
FROM pg_user;

-- Privilèges sur tables
\dp nom_table

-- Utilisateur actuel
SELECT current_user;
```

LISV Laboratoire d'ingénierie des systèmes de Versailles | UVSQ université PARIS-SACLAY

Révocation de privilèges - MariaDB

- **Syntaxe REVOKE** :

```
REVOKE privileges ON base.table
FROM 'utilisateur'@'hote';
```

- **Exemples** :

```
-- Retirer un privilège spécifique
REVOKE DELETE ON mabase.*
FROM 'appweb'@'localhost';

-- Retirer tous les privilèges
REVOKE ALL PRIVILEGES ON mabase.*
FROM 'appweb'@'localhost';
-- Retirer le droit de déléguer (GRANT OPTION)
REVOKE GRANT OPTION ON mabase.*
FROM 'admin'@'localhost';
```

Révocation de privilèges - PostgreSQL

- **Syntaxe REVOKE** :

```
REVOKE privileges ON base FROM utilisateur;
```

- **Exemples** :

```
-- Retirer des privilèges sur une table
REVOKE INSERT, UPDATE ON clients FROM appweb;

-- Retirer tous les privilèges sur une base
REVOKE ALL PRIVILEGES ON DATABASE mabase
FROM appweb;

-- Retirer tous les privilèges sur toutes les tables
REVOKE ALL PRIVILEGES ON ALL TABLES
IN SCHEMA public FROM appweb;
```

Modification d'utilisateurs

MariaDB :

```
-- Changer le mot de passe
ALTER USER
'appweb'@'localhost'
IDENTIFIED BY
'NewPass123!';

-- Renommer
RENAME USER
'oldname'@'localhost'
TO
'newname'@'localhost';
```

PostgreSQL :

```
-- Changer le mot de passe
ALTER USER appweb
WITH PASSWORD
'NewPass123!';

-- Renommer
ALTER USER oldname
RENAME TO newname;

-- Modifier attributs
ALTER USER appweb
CREATEDB;
```

Suppression d'utilisateurs

MariaDB :

```
DROP USER
'appweb'@'localhost';

-- Plusieurs utilisateurs
DROP USER
'user1'@'localhost',
'user2'@'%';
```

PostgreSQL :

```
DROP USER appweb;

-- ou
DROP ROLE appweb;

-- Supprimer si existe
DROP USER IF EXISTS
appweb;
```

Attention

Supprimer l'utilisateur supprime tous ses privilèges.

Note

Ne peut pas supprimer un utilisateur qui possède des objets.

Rôles dans PostgreSQL

- **Concept de rôle** :
  - Dans PostgreSQL, utilisateurs et groupes sont des rôles,
  - Un rôle peut être membre d'un autre rôle,
  - Facilite la gestion des privilèges.
- **Avantages** :
  - Gestion centralisée des privilèges,
  - Réutilisation des configurations,
  - Héritage de privilèges,
  - Simplification de l'administration.

Création et utilisation de rôles - PostgreSQL

- **Créer un rôle** :

```
CREATE ROLE lecteur;
CREATE ROLE editeur;
```

- **Attribuer des privilèges au rôle** :

```
GRANT SELECT ON ALL TABLES IN SCHEMA public
TO lecteur;

GRANT SELECT, INSERT, UPDATE, DELETE
ON ALL TABLES IN SCHEMA public TO editeur;
```

- **Assigner un rôle à un utilisateur** :

```
GRANT lecteur TO jean;
GRANT editeur TO marie;
```

Contrôle d'accès host-based - PostgreSQL

- **Fichier pg_hba.conf** : contrôle qui peut se connecter
- **Format** : type base utilisateur adresse méthode

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
local   all             postgres                                peer
host    all             all             127.0.0.1/32            md5
host    all             all             192.168.1.0/24          md5
host    mabase          appweb          10.0.0.0/8              scram-sha-256
```

- **Méthodes d'authentification** :
  - peer : utilisateur système = utilisateur PostgreSQL,
  - md5 : mot de passe hashé MD5,
  - scram-sha-256 : méthode moderne plus sécurisée,
  - reject : refuser la connexion.

Gestion de l'accès à phpMyAdmin

- **Configuration des utilisateurs autorisés** :
  - Par défaut, tous les utilisateurs MariaDB peuvent se connecter,
  - Possibilité de restreindre via configuration phpMyAdmin.
- **Sécurisation** :
  - Désactiver l'accès root : $cfg('Servers')($i)('AllowRoot') = false ;,
  - Authentification HTTP en plus,
  - Restriction par IP,
  - Utiliser HTTPS uniquement.
- **Créer un utilisateur dédié pour phpMyAdmin** :
  - Avec privilèges limités,
  - Uniquement depuis localhost,
  - Mot de passe fort.

Exemple : Utilisateur pour application web

- **Besoin** : Application web nécessitant accès à une base
- **Privilèges nécessaires** : SELECT, INSERT, UPDATE, DELETE

MariaDB :

```
CREATE USER 'webapp'@'192.168.1.50'
IDENTIFIED BY 'WebApp2024!';

GRANT SELECT, INSERT, UPDATE, DELETE
ON ecommerce.* TO 'webapp'@'192.168.1.50';
FLUSH PRIVILEGES;
```

Analyse

- Accès uniquement depuis le serveur web (192.168.1.50),
- Pas de privilèges DROP, ALTER (sécurité),
- Base spécifique (ecommerce), pas toutes les bases.

Exemple : Utilisateur en lecture seule

- **Besoin** : Reporting, analyses, sans modification
- **Privilèges nécessaires** : SELECT uniquement

PostgreSQL :

```
CREATE USER analyste WITH PASSWORD 'Analyst2024!';

GRANT CONNECT ON DATABASE ventes TO analyste;

\c ventes

GRANT SELECT ON ALL TABLES IN SCHEMA public
TO analyste;

-- Appliquer aussi aux futures tables
ALTER DEFAULT PRIVILEGES IN SCHEMA public
GRANT SELECT ON TABLES TO analyste;
```

Exemple : Administrateur de base

- **Besoin** : Gestion complète d'une base spécifique
- **Privilèges nécessaires** : Tous, sauf super-admin

MariaDB :

```
CREATE USER 'dbadmin'@'localhost'
IDENTIFIED BY 'DBAdmin2024!';

GRANT ALL PRIVILEGES ON projet.*
TO 'dbadmin'@'localhost' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

WITH GRANT OPTION

Permet à l'utilisateur de déléguer ses privilèges à d'autres utilisateurs.

Audit et traçabilité

- **Importance de l'audit** :
  - Tracer les actions des utilisateurs,
  - Détecter les comportements anormaux,
  - Conformité réglementaire (RGPD, etc.),
  - Investigation en cas d'incident.
- **Moyens d'audit** :
  - Logs généraux du SGBD,
  - Logs de requêtes lentes,
  - Plugins d'audit (MariaDB Audit Plugin),
  - Extensions PostgreSQL (pgAudit),
  - Analyse régulière des connexions actives.

Surveillance des connexions actives

MariaDB :

```
-- Liste des processus
SHOW PROCESSLIST;

-- Plus détaillé
SHOW FULL
PROCESSLIST;

-- Tuer une connexion
KILL process_id;
```

PostgreSQL :

```
-- Connexions actives
SELECT * FROM
pg_stat_activity;

-- Tuer une connexion
SELECT
pg_terminate_backend(
  pid
);
```

Bonnes pratiques de gestion des utilisateurs

- **Sécurité** :
  - Ne jamais utiliser root/postgres pour les applications,
  - Mots de passe forts et complexes,
  - Rotation régulière des mots de passe,
  - Désactiver les comptes inutilisés.
- **Organisation** :
  - Nommage cohérent des utilisateurs,
  - Documentation des rôles et privilèges,
  - Utiliser des rôles/groupes pour faciliter la gestion,
  - Revue périodique des accès.
- **Principe du moindre privilège** :
  - Accorder uniquement les droits nécessaires,
  - Privilégier les accès spécifiques aux bases/tables,
  - Éviter GRANT ALL autant que possible.

Scénarios d'erreurs courantes

- **Erreur "Access denied"** :
  - Vérifier nom d'utilisateur et mot de passe,
  - Vérifier l'hôte de connexion (localhost vs IP),
  - Vérifier que l'utilisateur existe,
  - Vérifier pg_hba.conf (PostgreSQL).
- **Erreur "Permission denied"** :
  - L'utilisateur n'a pas les privilèges nécessaires,
  - Vérifier avec SHOW GRANTS ou \dp,
  - Accorder les privilèges manquants.
- **Impossibilité de supprimer un utilisateur** :
  - L'utilisateur possède des objets (PostgreSQL),
  - Réassigner ou supprimer les objets d'abord.

Récapitulatif

- Distinction entre utilisateurs système et utilisateurs SGBD.
- Principe du moindre privilège : fondamental pour la sécurité.
- Types de privilèges : globaux, base, table, colonne.
- Commandes principales : CREATE USER, GRANT, REVOKE, DROP USER.
- Format MariaDB : 'utilisateur'@'hôte'.
- PostgreSQL : rôles et pg_hba.conf pour contrôle d'accès.
- Sécurisation des interfaces d'administration (phpMyAdmin, pgAdmin).
- Audit et traçabilité : logs, surveillance des connexions.
- Bonnes pratiques : mots de passe forts, revue régulière, documentation.

Prochaine séance

Sécurisation des données et des connexions

Chapitre 4

Sécurisation des données et des connexions

Objectifs du chapitre :

- Comprendre les enjeux de sécurité d'un SGBD
- Mettre en place le chiffrement SSL/TLS
- Configurer des accès distants sécurisés
- Implémenter les bonnes pratiques de sécurité

Enjeux de sécurité

- **Menaces principales** :
  - Interception des communications (man-in-the-middle),
  - Attaques par force brute sur les mots de passe,
  - Injection SQL,
  - Accès non autorisés,
  - Vol ou perte de données,
  - Déni de service (DoS).
- **Conséquences potentielles** :
  - Fuite de données sensibles (RGPD),
  - Perte de confiance des clients,
  - Sanctions légales et financières,
  - Interruption de service.

Principes de sécurité

- **Défense en profondeur** :
  - Plusieurs couches de sécurité,
  - Ne jamais se reposer sur une seule mesure.
- **Principe du moindre privilège** :
  - Déjà abordé au chapitre 3,
  - S'applique aussi aux accès réseau.
- **Chiffrement des données** :
  - En transit : SSL/TLS,
  - Au repos : chiffrement du stockage.
- **Isolation réseau** :
  - VLAN dédiés,
  - Segmentation réseau,
  - Pare-feu et filtrage.

Architecture réseau sécurisée

- **Placement du SGBD** :
  - VLAN dédié ou DMZ interne,
  - Jamais directement accessible depuis Internet,
  - Isolation des environnements (dev, test, prod).
- **Flux réseau autorisés** :
  - Serveur web → SGBD (port 3306 ou 5432),
  - Poste admin → SGBD via VPN ou bastion (jump host),
  - SGBD → Serveur de sauvegarde,
  - Tout le reste : bloqué par défaut.
- **Principe du "deny all, allow specific"** :
  - Bloquer tout par défaut,
  - Autoriser uniquement le nécessaire.

Configuration du pare-feu - Règles strictes

- **Autoriser uniquement les sources connues** :

```
# MariaDB : accès depuis le serveur web uniquement
sudo ufw allow from 192.168.1.50 to any port 3306

# PostgreSQL : accès depuis le réseau admin
sudo ufw allow from 10.0.10.0/24 to any port 5432

# Bloquer tout le reste (implicite avec ufw)
sudo ufw default deny incoming
sudo ufw default allow outgoing

# Activer le pare-feu
sudo ufw enable
```

Vérification de la configuration pare-feu

- **Lister les règles actives** :

```
sudo ufw status verbose
```

- **Résultat attendu** :

```
Status: active

To                         Action      From
--                         ------      ----
3306                       ALLOW       192.168.1.50
5432                       ALLOW       10.0.10.0/24
22/tcp                     ALLOW       Anywhere
```

Sécurité SSH

Penser à sécuriser également l'accès SSH au serveur !

Chiffrement SSL/TLS - Principe

- **Pourquoi chiffrer ?**
  - Les communications SGBD transitent en clair par défaut,
  - Mot de passe et données visibles sur le réseau,
  - Vulnérable aux écoutes (sniffing).
- **SSL/TLS** :
  - Protocole de chiffrement des communications,
  - Authentification du serveur (certificat),
  - Intégrité et confidentialité des données,
  - Standard pour sécuriser les connexions.
- **Composants nécessaires** :
  - Certificat SSL/TLS du serveur,
  - Clé privée du serveur,
  - Autorité de certification (CA) - optionnelle.

Génération de certificats SSL - Auto-signés

- **Créer le répertoire pour les certificats** :

```
sudo mkdir -p /etc/mysql/ssl
cd /etc/mysql/ssl
```

- **Générer la clé privée et le certificat** :

```
# Clé privée de l'autorité de certification
sudo openssl genrsa 2048 > ca-key.pem

# Certificat de l'autorité de certification
sudo openssl req -new -x509 -nodes -days 3650 \
  -key ca-key.pem -out ca-cert.pem

# Clé privée du serveur
sudo openssl req -newkey rsa:2048 -days 3650 \
  -nodes -keyout server-key.pem -out server-req.pem
```

Génération de certificats SSL - Suite

```
# Signer le certificat serveur avec le CA
sudo openssl x509 -req -in server-req.pem \
  -days 3650 -CA ca-cert.pem -CAkey ca-key.pem \
  -set_serial 01 -out server-cert.pem

# Clé et certificat client (optionnel)
sudo openssl req -newkey rsa:2048 -days 3650 \
  -nodes -keyout client-key.pem -out client-req.pem
sudo openssl x509 -req -in client-req.pem \
  -days 3650 -CA ca-cert.pem -CAkey ca-key.pem \
  -set_serial 02 -out client-cert.pem
```

Note

Pour une production, utiliser des certificats signés par une CA reconnue (Let's Encrypt, DigiCert, etc.).

Configuration SSL - MariaDB

- **Éditer le fichier de configuration** :

```
sudo nano /etc/mysql/mariadb.conf.d/50-server.cnf
```

- **Ajouter dans la section (mysqld)** :

```
[mysqld]
ssl-ca=/etc/mysql/ssl/ca-cert.pem
ssl-cert=/etc/mysql/ssl/server-cert.pem
ssl-key=/etc/mysql/ssl/server-key.pem

# Forcer SSL pour toutes les connexions (optionnel)
require_secure_transport=ON
```

- **Redémarrer le service** :

```
sudo systemctl restart mariadb
```

Configuration SSL - PostgreSQL

- **Copier les certificats dans le répertoire de données** :

```
sudo cp /etc/mysql/ssl/server-cert.pem \
  /var/lib/postgresql/15/main/server.crt
sudo cp /etc/mysql/ssl/server-key.pem \
  /var/lib/postgresql/15/main/server.key
sudo chown postgres:postgres \
  /var/lib/postgresql/15/main/server.*
sudo chmod 600 /var/lib/postgresql/15/main/server.key
```

- **Éditer postgresql.conf** :

```
sudo nano /etc/postgresql/15/main/postgresql.conf
```

```
ssl = on
ssl_cert_file = 'server.crt'
ssl_key_file = 'server.key'
```

Configuration SSL - PostgreSQL (suite)

- **Modifier pg_hba.conf pour exiger SSL** :

```
sudo nano /etc/postgresql/15/main/pg_hba.conf
```

```
# TYPE  DATABASE    USER        ADDRESS         METHOD
# Exiger SSL pour les connexions distantes
hostssl all         all         192.168.1.0/24  md5
hostssl all         all         0.0.0.0/0       scram-sha-256

# Connexion locale sans SSL
local   all         postgres                    peer
```

- **Redémarrer PostgreSQL** :

```
sudo systemctl restart postgresql
```

Vérification de SSL - MariaDB

- **Se connecter et vérifier SSL** :

```
mysql -u root -p --ssl

-- Dans le client MySQL
SHOW VARIABLES LIKE '%ssl%';

-- Vérifier la connexion courante
SHOW STATUS LIKE 'Ssl_cipher';
```

- **Résultat attendu** :
  - have_ssl : YES,
  - Ssl_cipher : affiche l'algorithme de chiffrement utilisé (ex : TLS_AES_256_GCM_SHA384).

Vérification de SSL - PostgreSQL

- **Se connecter en vérifiant SSL** :

```
psql "host=localhost user=postgres sslmode=require"

-- Dans psql
\conninfo
```

- **Résultat attendu** :

```
You are connected to database "postgres" as user
"postgres" on host "localhost" at port "5432".
SSL connection (protocol: TLSv1.3, cipher:
TLS_AES_256_GCM_SHA384, bits: 256)
```

Forcer SSL pour un utilisateur - MariaDB

- **Créer un utilisateur avec obligation SSL** :

```
CREATE USER 'secure_user'@'%'
IDENTIFIED BY 'SecurePass123!'
REQUIRE SSL;
```

- **Modifier un utilisateur existant** :

```
ALTER USER 'appweb'@'192.168.1.50'
REQUIRE SSL;
```

- **Vérifier les exigences SSL** :

```
SELECT user, host, ssl_type
FROM mysql.user
WHERE user='secure_user';
```

Accès distant sécurisé via SSH Tunnel

- **Principe du tunnel SSH** :
  - Encapsuler la connexion SGBD dans SSH,
  - Chiffrement automatique par SSH,
  - Authentification SSH (clé ou mot de passe),
  - Pas besoin de configurer SSL sur le SGBD.
- **Avantages** :
  - Sécurité éprouvée de SSH,
  - Simple à mettre en place,
  - Ne nécessite que le port SSH ouvert,
  - Le SGBD peut rester en écoute locale uniquement.
- **Cas d'usage** :
  - Administration à distance,
  - Développement depuis un poste externe,
  - Accès ponctuel sécurisé.

Configuration d'un tunnel SSH

- **Configuration du SGBD** :
  - Laisser bind-address = 127.0.0.1 (écoute locale),
  - Pas besoin d'ouvrir le port SGBD sur le pare-feu.
- **Créer le tunnel depuis le client** :

Pour MariaDB :

```
ssh -L 3306:localhost:3306 user@serveur-sgbd.com
```

Pour PostgreSQL :

```
ssh -L 5432:localhost:5432 user@serveur-sgbd.com
```

Explication

-L port_local :hote_distant :port_distant

Redirige le port local vers le port distant via SSH.

Utilisation du tunnel SSH

- **Une fois le tunnel établi, se connecter localement** :

MariaDB :

```
mysql -h 127.0.0.1 -P 3306 -u appweb -p
```

PostgreSQL :

```
psql -h 127.0.0.1 -p 5432 -U appweb -d mabase
```

- **Avec un outil graphique** :
  - phpMyAdmin : impossible directement,
  - MySQL Workbench, DBeaver, pgAdmin : configurer le tunnel SSH dans les paramètres de connexion.

Accès distant via VPN

- **Principe** :
  - Créer un réseau privé virtuel,
  - Le client distant fait partie du réseau local,
  - Accès direct au SGBD comme si on était sur place.
- **Solutions VPN** :
  - OpenVPN (open source, flexible),
  - WireGuard (moderne, performant),
  - IPsec,
  - VPN matériels (Cisco, Fortinet, etc.).
- **Avantages** :
  - Sécurise tout le trafic réseau,
  - Idéal pour accès multiples (pas que SGBD),
  - Intégration transparente.

Protection contre les injections SQL

- **Injection SQL** : attaque exploitant une mauvaise validation des entrées
- **Exemple vulnérable (PHP)** :
  - `$query = "SELECT * FROM users WHERE id="$id"";`,
  - Si $id = "1 OR 1=1", retourne tous les utilisateurs !
- **Mesures de prévention** :
  - Utiliser des requêtes préparées (prepared statements),
  - Validation et échappement des entrées,
  - Principe du moindre privilège (pas de DROP, ALTER pour l'appli),
  - WAF (Web Application Firewall) en complément.
- **Responsabilité** :
  - Principalement côté développement,
  - Mais l'admin SGBD peut limiter les dégâts.

Requêtes préparées - Exemple

Vulnérable :

```
$query = "SELECT *
FROM users
WHERE id=$id";
$result =
mysqli_query(
  $conn, $query
);
```

Sécurisé :

```
$stmt = $conn->prepare(
  "SELECT * FROM users
   WHERE id=?"
);
$stmt->bind_param(
  "i", $id
);
$stmt->execute();
```

Requête préparée

Le SGBD sépare la structure de la requête des données, empêchant l'injection.

Protection contre les attaques par force brute

- **Attaque par force brute** :
  - Tentatives répétées de connexion avec différents mots de passe,
  - Cible les comptes aux mots de passe faibles.
- **Mesures de protection** :
  - Mots de passe forts et complexes,
  - Limitation du nombre de tentatives (fail2ban),
  - Accès restreint par IP (pare-feu, pg_hba.conf),
  - Désactivation des comptes par défaut (root, admin),
  - Authentification à deux facteurs (si supportée),
  - Surveillance des logs d'authentification.

Fail2ban pour SGBD

- **Fail2ban** : outil de protection contre les attaques par force brute
- **Installation** :

```
sudo apt install fail2ban -y
```

- **Configuration pour MariaDB** :
  - Créer /etc/fail2ban/jail.d/mariadb.conf,
  - Surveiller /var/log/mysql/error.log,
  - Bannir après X tentatives échouées.
- **Configuration pour PostgreSQL** :
  - Surveiller /var/log/postgresql/postgresql-*.log,
  - Détecter "FATAL : password authentication failed".

Exemple de configuration Fail2ban

- **Fichier /etc/fail2ban/jail.d/mariadb.conf** :

```
[mariadb]
enabled  = true
port     = 3306
filter   = mariadb-auth
logpath  = /var/log/mysql/error.log
maxretry = 3
bantime  = 3600
findtime = 600
```

- **Redémarrer fail2ban** :

```
sudo systemctl restart fail2ban
sudo fail2ban-client status mariadb
```

Sécurisation du système d'exploitation

- **Le SGBD n'est qu'une couche** :
  - Sécuriser aussi l'OS sous-jacent,
  - Un OS compromis = SGBD compromis.
- **Bonnes pratiques OS** :
  - Mises à jour régulières du système,
  - Désactivation des services inutiles,
  - Configuration SSH sécurisée (clés, pas de root),
  - SELinux ou AppArmor activé,
  - Surveillance des logs système,
  - Antivirus/anti-malware si pertinent.
- **Principe** :
  - Durcissement du système (hardening),
  - Suivre les guides CIS Benchmarks.

Chiffrement des données au repos

- **Chiffrement au repos** : protéger les fichiers de données sur le disque
- **Pourquoi ?**
  - Protection en cas de vol physique du serveur/disque,
  - Conformité réglementaire (RGPD, PCI-DSS),
  - Protection contre accès non autorisé aux fichiers.
- **Solutions** :
  - Chiffrement du système de fichiers (LUKS, dm-crypt),
  - Chiffrement au niveau SGBD (MariaDB : InnoDB encryption),
  - Chiffrement au niveau stockage (SAN, NAS),
  - Tablespaces chiffrés (PostgreSQL via extensions).
- **Attention** :
  - Impact sur les performances,
  - Gestion des clés de chiffrement critique.

Chiffrement au niveau fichiers - LUKS

- **LUKS** : Linux Unified Key Setup (chiffrement de partition)
- **Principe** :
  - Créer une partition chiffrée,
  - Monter le datadir du SGBD sur cette partition,
  - Transparent pour le SGBD.

```
# Créer une partition chiffrée (exemple simplifié)
sudo cryptsetup luksFormat /dev/sdb1
sudo cryptsetup luksOpen /dev/sdb1 encrypted_data
sudo mkfs.ext4 /dev/mapper/encrypted_data
sudo mount /dev/mapper/encrypted_data /var/lib/mysql
```

Gestion des clés

La clé de déchiffrement doit être disponible au démarrage ou stockée de manière sécurisée.

Audit de sécurité

- **Audits réguliers** :
  - Vérifier les configurations de sécurité,
  - Analyser les logs d'accès et d'erreurs,
  - Détecter les comportements anormaux,
  - Vérifier les privilèges utilisateurs.
- **Outils d'audit** :
  - MySQL Enterprise Audit,
  - MariaDB Audit Plugin,
  - pgAudit (PostgreSQL),
  - Scripts personnalisés.
- **Que surveiller ?**
  - Tentatives de connexion échouées,
  - Requêtes sensibles (DROP, ALTER, GRANT),
  - Accès depuis des IPs inhabituelles,
  - Volumes de données anormaux.

Activation de l'audit - MariaDB

- **Installer le plugin d'audit** :

```
INSTALL PLUGIN server_audit
SONAME 'server_audit.so';
```

- **Configurer dans my.cnf** :

```
[mysqld]
server_audit_logging=ON
server_audit_events=CONNECT,QUERY,TABLE
server_audit_file_path=/var/log/mysql/audit.log
server_audit_file_rotate_size=1000000
```

- **Redémarrer MariaDB** :

```
sudo systemctl restart mariadb
```

Activation de l'audit - PostgreSQL

- **Installer pgAudit** :

```
sudo apt install postgresql-15-pgaudit
```

- **Configurer dans postgresql.conf** :

```
shared_preload_libraries = 'pgaudit'
pgaudit.log = 'read,write,ddl'
pgaudit.log_catalog = off
pgaudit.log_parameter = on
```

- **Activer dans la base** :

```
CREATE EXTENSION pgaudit;
```

Conformité RGPD

- **RGPD** : Règlement Général sur la Protection des Données
- **Obligations pour les SGBD** :
  - Chiffrement des données personnelles,
  - Traçabilité des accès et modifications,
  - Droit à l'effacement (suppression effective),
  - Limitation de l'accès (principe du moindre privilège),
  - Notification en cas de violation de données,
  - Conservation limitée dans le temps.
- **Mesures techniques** :
  - Pseudonymisation et anonymisation,
  - Logs d'audit détaillés,
  - Procédures de suppression sécurisée,
  - Sauvegardes chiffrées.

Checklist de sécurité

- ✓ Isolation réseau (VLAN, pare-feu)
- ✓ Chiffrement SSL/TLS activé
- ✓ Accès distants sécurisés (VPN, SSH tunnel)
- ✓ Mots de passe forts pour tous les comptes
- ✓ Principe du moindre privilège appliqué
- ✓ Comptes par défaut désactivés ou sécurisés
- ✓ Fail2ban ou équivalent configuré
- ✓ Logs d'audit activés
- ✓ Sauvegardes régulières et chiffrées
- ✓ Mises à jour de sécurité appliquées
- ✓ Surveillance et monitoring actifs
- ✓ Plan de réponse aux incidents

Récapitulatif

- La sécurité d'un SGBD est multicouche : réseau, chiffrement, authentification, audit.
- Isolation réseau : VLAN, pare-feu avec règles strictes.
- Chiffrement SSL/TLS : protège les données en transit.
- Accès distants : VPN ou tunnel SSH recommandés.
- Protection contre injections SQL : responsabilité partagée dev/admin.
- Fail2ban : protection contre force brute.
- Chiffrement au repos : LUKS, chiffrement SGBD natif.
- Audit de sécurité : plugins d'audit, analyse des logs.
- Conformité RGPD : chiffrement, traçabilité, droit à l'oubli.

Prochaine séance

Sauvegarde et restauration des données