

Programme : Administration PostgreSQL Server

Module 1 : Installation et Configuration de PostgreSQL Server

- **Préparation de l'installation**
 - Prérequis système et configuration matérielle
 - Choix de la version (communauté, entreprise, cloud)
 - Paramètres de configuration initiaux (`postgresql.conf`, `pg_hba.conf`)
- **Installation de PostgreSQL Server**
 - Installation sous Linux et Windows
 - Utilisation des packages (`apt`, `yum`) et méthodes alternatives (compilation manuelle)
 - Création de clusters PostgreSQL
- **Post-installation**
 - Initialisation d'une base de données
 - Création d'utilisateurs et gestion des privilèges

Module 2 : Comprendre les Modes de Journalisation et Récupération dans PostgreSQL

- **Architecture du système de récupération**
 - Fonctionnement des journaux de transactions (Write-Ahead Logging - WAL)
 - Signification des paramètres de récupération (`checkpoint_timeout`, `max_wal_size`, `wal_keep_segments`)
- **Stratégies de sauvegarde et récupération**
 - Sauvegardes physiques vs sauvegardes logiques
 - Modes de récupération disponibles : point de restauration (`PITR`), bas niveau (`pg_restore`), réplicas

Module 3 : Sauvegardes dans PostgreSQL

- **Sauvegarde d'une base de données**
 - Utilisation de `pg_dump` et `pg_dumpall` (sauvegardes logiques)
 - Sauvegarde des journaux de transactions (`archive_command`, `pg_basebackup`)
- **Gestion des sauvegardes**
 - Automatisation avec des scripts shell
 - Rotation et archivage des backups

Module 4 : Restauration de Bases de Données

- **Comprendre le processus de restauration**
 - Utilisation de `pg_restore` pour restaurer des sauvegardes logiques
 - Utilisation des archives WAL pour une restauration point-in-time (`PITR`)
- **Restauration spécifique**
 - Restauration d'un schéma ou d'une table

- Restauration incrémentale
- **Gestion des erreurs lors de la restauration**

Module 5 : Automatiser la Maintenance de PostgreSQL

- **Surveillance et planification**
 - Paramétrage des logs d'activité et des erreurs
 - Utilisation de `pg_stat_*` pour le monitoring
- **Tâches de maintenance**
 - Analyse (`ANALYZE`) et optimisation des tables (`VACUUM`, `VACUUM FULL`)
 - Planification de tâches via `cron` (Linux) ou `pgAgent`
- **Automatisation des sauvegardes et nettoyage**
 - Scripts pour backups automatiques
 - Rotation des fichiers de logs

Module 6 : Réplication PostgreSQL

- **Réplication physique et logique**
 - Configuration du `streaming replication`
 - Création d'un serveur maître et d'un ou plusieurs réplicas
- **Réplication avancée**
 - Utilisation de `pglogical` pour la réplication logique
 - Réplication partielle (publication/subscription)
- **Scénarios d'utilisation**
 - Réplication pour tolérance aux pannes
 - Réplication en lecture seule pour la répartition des charges

Module 7 : Clustering et Haute Disponibilité dans PostgreSQL

- **Clustering avec Patroni ou Pgpool-II**
 - Gestion d'un cluster maître/réplices avec `Patroni`
 - Répartition de charge et bascule automatique
- **Failover et Recovery**
 - Configuration d'un failover automatique
 - Utilisation de `repmgr` pour la gestion des bascules

Objectifs pédagogiques

À la fin du cours, les participants sauront :

- Installer, configurer et sécuriser un serveur PostgreSQL

- Planifier et exécuter des stratégies de sauvegarde et restauration
- Automatiser les tâches de maintenance
- Configurer la réplication et gérer la tolérance aux pannes
- Déployer un cluster PostgreSQL pour assurer la haute disponibilité

Ce programme est modulable et peut inclure des exercices pratiques à chaque étape (installation sur VM, scénarios de panne, réplication). Si vous souhaitez plus de détails sur un module ou un ajout spécifique (par exemple l'intégration avec des outils comme Prometheus/Grafana pour le monitoring),

Exercice Pratique : Installation et Configuration de PostgreSQL Server

Objectifs de l'exercice :

- Installer PostgreSQL sur un système Linux Ubuntu.
- Modifier les fichiers de configuration (postgresql.conf et pg_hba.conf) pour autoriser les connexions distantes.
- Créer une base de données, un utilisateur, et tester la connexion avec psql et pgAdmin.

Partie 1 : Installation de PostgreSQL

Étape 1 : Préparation du système

1. Mettez à jour votre système pour garantir que tous les paquets sont à jour :

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

2. Vérifiez que votre système dispose des prérequis :

- RAM : au moins 4 Go
- Stockage : au moins 20 Go de libre

Étape 2 : Installation de PostgreSQL

1. Installez PostgreSQL et ses outils :

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

- postgresql : le serveur de base de données
- postgresql-contrib : des outils complémentaires pour PostgreSQL

2. Vérifiez l'installation :

```
psql --version
```

La commande devrait afficher la version installée, par exemple : psql (PostgreSQL) 15.x.

Partie 2 : Configuration de PostgreSQL

Étape 1 : Modifier postgresql.conf pour autoriser les connexions distantes

1. Ouvrez le fichier de configuration :

```
sudo nano /etc/postgresql/ <<PG_VERSION_NUMBER>> /main/postgresql.conf
```

2. Recherchez la ligne listen_addresses et modifiez-la ainsi :

```
listen_addresses = '*'
```

Cela permet à PostgreSQL d'écouter sur toutes les adresses IP.

Étape 2 : Modifier pg_hba.conf pour configurer les accès utilisateurs

1. Ouvrez le fichier pg_hba.conf :

```
sudo nano /etc/postgresql/ <<PG_VERSION_NUMBER>> /main/pg_hba.conf
```

2. Ajoutez la ligne suivante pour autoriser les connexions distantes avec mot de passe :

```
host all all 0.0.0.0/0 md5
```

- 0.0.0.0/0 : permet à toutes les adresses IP de se connecter (pour le test uniquement).
 - md5 : méthode d'authentification par mot de passe chiffré.
3. Sauvegardez et quittez le fichier (CTRL + O, ENTER, puis CTRL + X).

Étape 3 : Redémarrer PostgreSQL

Après avoir modifié les fichiers de configuration, redémarrez le service PostgreSQL :

```
sudo systemctl restart postgresql
```

Partie 3 : Création d'une base de données et d'un utilisateur

Étape 1 : Accéder à psql

Connectez-vous à PostgreSQL en tant que superutilisateur postgres :

```
sudo -u postgres psql
```

Étape 2 : Créer une base de données

Dans psql, exécutez la commande suivante pour créer une base de données :

```
CREATE DATABASE testdb;
```

Étape 3 : Créer un utilisateur

Créez un utilisateur avec un mot de passe sécurisé :

```
CREATE USER testuser WITH PASSWORD 'securepassword123';
```

Étape 4 : Attribuer des privilèges

Donnez tous les privilèges sur la base de données testdb à l'utilisateur testuser :

```
GRANT ALL PRIVILEGES ON DATABASE testdb TO testuser;
```

Étape 5 : Quitter psql

Tapez \q pour quitter l'interface psql.

Partie 4 : Test de Connexion avec psql

1. Depuis un poste local, essayez de vous connecter à la base de données distante :

```
psql -h <IP_DU_SERVEUR> -U testuser -d testdb
```

Remplacez <IP_DU_SERVEUR> par l'adresse IP de la machine PostgreSQL.

2. Entrez le mot de passe de l'utilisateur lorsque demandé.

Partie 5 : Test de Connexion avec pgAdmin

1. Ouvrez **pgAdmin** sur votre poste client.
2. Ajoutez un nouveau serveur :
 - **Nom** : PostgreSQL_Test_Server
 - **Hôte** : <IP_DU_SERVEUR>
 - **Port** : 5432
 - **Utilisateur** : testuser
 - **Mot de passe** : securepassword123
3. Connectez-vous et vérifiez que vous voyez la base de données testdb.

Partie 6 : Vérification des Logs de Connexion

Sur le serveur PostgreSQL, consultez les logs pour vérifier les connexions :

```
sudo tail -f /var/log/postgresql/postgresql-<<PG_VERSION_NUMBER>>-main.log
```

Les logs afficheront des messages indiquant l'établissement des connexions.

Bonus : Sécurisation Supplémentaire

1. **Restreindre l'accès distant à une plage IP spécifique :**
Dans pg_hba.conf, remplacez 0.0.0.0/0 par une plage d'adresses IP autorisée, par exemple :

```
host all all 192.168.1.0/24 md5
```

2. **Désactiver le compte postgres pour les connexions distantes :**
Ajoutez dans pg_hba.conf :

```
host all postgres 0.0.0.0/0 reject
```

Critères de Réussite :

- PostgreSQL est correctement installé et fonctionne.
- Les connexions locales et distantes fonctionnent avec psql et pgAdmin.
- Les utilisateurs et bases de données sont correctement créés et sécurisés.

Exercice Pratique 2 : Manipulation de la Console psql

Objectifs de l'exercice :

- S'habituer à l'utilisation de psql, l'interface en ligne de commande de PostgreSQL.
- Créer et manipuler des bases de données, des tables et des enregistrements.
- Apprendre à interroger et à gérer les données avec SQL via psql.
- Pratiquer les fonctionnalités avancées : transactions, jointures, exports, imports.

Durée totale : 2 heures

Partie 1 : Connexion et Navigation (20 minutes)

Étape 1 : Connexion à psql

Depuis le terminal, connectez-vous à PostgreSQL avec l'utilisateur postgres :

bash

Copier le code

sudo -u postgres psql

Étape 2 : Affichage des informations de base

- Afficher la version de PostgreSQL :

```
SELECT version();
```

- Afficher la liste des bases de données :

```
\l
```

- Afficher les informations sur la connexion en cours :

```
\conninfo
```

Étape 3 : Créer une nouvelle base de données

Dans psql, créez une nouvelle base de données :

```
CREATE DATABASE exercice_db;
```

- Affichez la base de données nouvellement créée avec \l.

Étape 4 : Se connecter à la base de données

Pour se connecter à la base exercice_db :

```
\c exercice_db
```

Partie 2 : Création et Gestion des Tables (30 minutes)

Étape 1 : Création d'une table

Créez une table employes dans la base exercice_db :

```
CREATE TABLE employes (  
  id SERIAL PRIMARY KEY,  
  nom VARCHAR(50) NOT NULL,  
  poste VARCHAR(50),  
  salaire NUMERIC(10, 2),
```

```
date_embauche DATE  
);
```

Étape 2 : Afficher la structure de la table

Utilisez la commande \d pour afficher la structure de la table employes :

```
\d employes
```

Étape 3 : Insérer des enregistrements dans la table

Ajoutez des données à la table :

```
INSERT INTO employes (nom, poste, salaire, date_embauche) VALUES  
( 'Alice Dupont', 'Développeur', 3500.50, '2022-03-15'),  
( 'Marc Durand', 'Administrateur Systèmes', 4200.00, '2021-09-01'),  
( 'Sophie Martin', 'Analyste', 3700.25, '2020-01-10');
```

Étape 4 : Afficher le contenu de la table

Affichez toutes les données de la table employes :

```
SELECT * FROM employes;
```

Partie 3 : Interrogation des Données (30 minutes)

Étape 1 : Requêtes simples

1. Affichez uniquement le nom et le poste des employés :

```
SELECT nom, poste FROM employes;
```

2. Affichez les employés ayant un salaire supérieur à 4000 euros :

```
SELECT * FROM employes WHERE salaire > 4000;
```

Étape 2 : Requêtes avec tri et limite

1. Affichez les employés triés par date d'embauche :

```
SELECT * FROM employes ORDER BY date_embauche;
```

2. Affichez les 2 premiers employés ayant le plus haut salaire :

```
SELECT * FROM employes ORDER BY salaire DESC LIMIT 2;
```

Étape 3 : Requêtes avec fonctions d'agrégation

1. Calculez le salaire moyen des employés :

```
SELECT AVG(salaire) AS salaire_moyen FROM employes;
```

2. Affichez le nombre total d'employés :


```
SELECT COUNT(*) FROM employes;
```

Partie 4 : Transactions et Mise à Jour des Données (20 minutes)

Étape 1 : Transactions

1. Démarrez une transaction :

```
BEGIN;
```

2. Modifiez le salaire d'un employé :

```
UPDATE employes SET salaire = 3900.00 WHERE nom = 'Alice Dupont';
```

3. Vérifiez les modifications :

```
SELECT * FROM employes WHERE nom = 'Alice Dupont';
```

4. Annulez la transaction pour revenir en arrière :

```
ROLLBACK;
```

5. Recommencez avec un COMMIT pour valider les changements :

```
COMMIT;
```

Partie 5 : Joindre des Tables et Manipulation Avancée (20 minutes)

Étape 1 : Créer une table supplémentaire

Créez une table departements :

```
CREATE TABLE departements (  
  id SERIAL PRIMARY KEY,  
  nom_depart VARCHAR(50),  
  responsable VARCHAR(50)  
);
```

Ajoutez des données :

```
INSERT INTO departements (nom_depart, responsable) VALUES  
( 'Informatique', 'Marc Durand'),  
( 'Ressources Humaines', 'Sophie Martin');
```

Étape 2 : Jointure entre deux tables

1. Affichez les informations des employés avec leur département (en utilisant une jointure) :

```
SELECT e.nom, e.poste, d.nom_depart  
FROM employes e  
LEFT JOIN departements d ON e.nom = d.responsable;
```

Partie 6 : Export et Import de Données (20 minutes)

Étape 1 : Exporter une table au format CSV

Depuis psql :

```
\copy (SELECT * FROM employes) TO '/tmp/employes.csv' CSV HEADER;
```

Étape 2 : Importer des données depuis un CSV

1. Préparez un fichier CSV contenant des données supplémentaires (par exemple nouveaux_employes.csv).
2. Importez le fichier dans la table employes :

```
\copy employes(nom, poste, salaire, date_embauche) FROM '/tmp/nouveaux_employes.csv' CSV HEADER;
```

Critères de Réussite :

- Connexion à PostgreSQL via psql.
- Création et manipulation des tables et des données.
- Exécution de requêtes SQL simples et avancées.
- Utilisation des transactions pour sécuriser les modifications.
- Export et import des données avec des fichiers CSV.

Exercice Pratique3 : Manipulation des Données - Import, Sauvegarde, Restauration Complète et Vérification

Durée totale : 2 heures

Cet exercice guidera pas à pas les stagiaires dans l'importation de données, la sauvegarde et la restauration d'une base PostgreSQL, ainsi que la vérification de l'intégrité des données restaurées.

Objectifs de l'exercice :

- Importer un fichier CSV dans une table PostgreSQL.
- Effectuer une sauvegarde complète de la base de données.
- Restaurer la base de données depuis une sauvegarde.
- Vérifier la cohérence et l'intégrité des données restaurées.

Prérequis :

- PostgreSQL installé et fonctionnel.
- Un utilisateur avec accès SUPERUSER.
- Fichier CSV avec des données (par exemple, nouveaux_employes.csv).
(Si vous ne l'avez pas encore, vous pouvez utiliser le fichier généré précédemment ou en télécharger un exemple.)

Partie 1 : Préparation de la Base de Données (15 minutes)

Étape 1 : Connexion à PostgreSQL

Dans le terminal, connectez-vous à PostgreSQL en utilisant psql :

```
sudo -u postgres psql
```

Étape 2 : Création d'une Base de Données

Créez une nouvelle base de données exercice_db :

```
CREATE DATABASE exercice_db;
```

Étape 3 : Connexion à la Base de Données exercice_db

Changez de base de données :

```
\c exercice_db
```

Partie 2 : Création de la Table (10 minutes)

Étape 1 : Création de la Table employes

Dans la base de données exercice_db, créez la table suivante :

```
CREATE TABLE employes (  
  id SERIAL PRIMARY KEY,  
  nom VARCHAR(50) NOT NULL,  
  poste VARCHAR(50),  
  salaire NUMERIC(10, 2),  
  date_embauche DATE  
);
```

Partie 3 : Importation des Données CSV (30 minutes)

Étape 1 : Placer le fichier CSV sur le serveur

Placez le fichier nouveaux_employes.csv dans un répertoire accessible, par exemple /tmp/.

Étape 2 : Importer le Fichier CSV

Dans la console psql, utilisez la commande \copy pour importer les données :

```
\copy employes(nom, poste, salaire, date_embauche) FROM '/tmp/nouveaux_employes.csv'  
DELIMITER ',' CSV HEADER;
```

- **DELIMITER ','** : Utilise la virgule comme séparateur.
- **CSV HEADER** : Indique que le fichier CSV contient une ligne d'en-tête.

Étape 3 : Vérification de l'Importation

Affichez les 10 premières lignes de la table employes pour vérifier que les données ont été importées correctement :

```
SELECT * FROM employes LIMIT 10;
```

Partie 4 : Sauvegarde Complète de la Base de Données (20 minutes)

Étape 1 : Sauvegarde avec pg_dump

Quittez la console psql en tapant \q, puis exécutez la commande suivante dans le terminal :

```
pg_dump -U postgres -d exercice_db -F c -f /tmp/exercice_db.backup
```

- **-U postgres** : Utilisateur PostgreSQL utilisé pour la sauvegarde.
- **-d exercice_db** : Nom de la base de données à sauvegarder.
- **-F c** : Format custom (binaire compressé).
- **-f** : Emplacement du fichier de sauvegarde.

Étape 2 : Vérification du Fichier de Sauvegarde

Vérifiez que le fichier /tmp/exercice_db.backup a bien été créé :

```
ls -lh /tmp/exercice_db.backup
```

Partie 5 : Suppression de la Base de Données (10 minutes)

Étape 1 : Supprimer la Base de Données

Dans la console psql, exécutez :

```
DROP DATABASE exercice_db;
```

Cette étape simule une perte de la base de données.

Partie 6 : Restauration Complète (30 minutes)

Étape 1 : Restauration avec pg_restore

Dans le terminal, utilisez pg_restore pour restaurer la base de données :

```
createdb -U postgres exercice_db  
pg_restore -U postgres -d exercice_db /tmp/exercice_db.backup
```

- **createdb** : Crée une nouvelle base de données vide exercice_db.
- **pg_restore** : Restaure la sauvegarde dans la base de données exercice_db.

Étape 2 : Vérification des Données Restaurées

Connectez-vous à la base restaurée :

```
psql -U postgres -d exercice_db
```

Puis exécutez la commande SQL suivante pour vérifier que toutes les données ont bien été restaurées :

```
SELECT COUNT(*) FROM employees;
```

Partie 7 : Comparaison des Données et Vérification Finale (15 minutes)

Étape 1 : Comparaison avec le Fichier CSV

Vérifiez si le nombre de lignes dans la base de données correspond au nombre de lignes du fichier CSV :

```
wc -l /tmp/nouveaux_employees.csv
```

Le résultat doit être équivalent au résultat de SELECT COUNT(*).

Étape 2 : Test de Requêtes sur les Données Restaurées

1. Afficher les 5 employés avec les salaires les plus élevés :

```
SELECT * FROM employees ORDER BY salaire DESC LIMIT 5;
```

2. Compter le nombre d'employés embauchés après 2020 :

```
SELECT COUNT(*) FROM employees WHERE date_embauche > '2020-01-01';
```

Bonus : Automatisation de la Sauvegarde

Configurez une tâche cron pour automatiser la sauvegarde de la base de données :

1. Ouvrez l'éditeur cron :

```
crontab -e
```

2. Ajoutez la ligne suivante pour effectuer une sauvegarde quotidienne à 2h du matin :

```
0 2 * * * pg_dump -U postgres -d exercice_db -F c -f /backup/exercice_db_$(date +%Y-%m-%d).backup
```

Critères de Réussite :

- Importation correcte des données CSV dans la table employes.
- Sauvegarde complète générée sans erreur.
- Restauration de la base de données avec l'intégralité des données.
- Résultat des requêtes identique avant et après la restauration.