

warning "Mise à jour du backend (octobre 2026)" L'inférence GitHub Models a été retirée le 30 juillet 2026. Les références ci-dessous sont conservées uniquement comme contexte historique pendant la migration de ce lab. Utilisez le Lab 013 ? Backends LLM gratuits pour configurer Foundry Local ou Ollama avant de continuer.

Lab 031 : Recherche sémantique pgvector sur Azure

Niveau : L300 Parcours : RAG Durée : ~60 min ? Coût : ~15 \$/mois (PostgreSQL Flexible Server Burstable B1ms)

warning "Abonnement Azure requis" Ce lab déploie Azure PostgreSQL Flexible Server. ? Guide des prérequis

Ce que vous apprendrez

Déployer Azure PostgreSQL Flexible Server avec pgvector activé en un clic Se connecter de manière sécurisée avec SSL et les règles de pare-feu Migrer le pipeline RAG du Lab 022 vers Azure Utiliser Azure AI Services pour les embeddings (ou continuer à utiliser GitHub Models ? gratuit) Exposer le magasin de vecteurs comme un outil MCP pour les agents

Déployer l'infrastructure

Option A ? Déployer sur Azure (un clic)

Deploy to Azure

Ceci déploie : PostgreSQL Flexible Server (Burstable B1ms, ~15 \$/mois) Extension pgvector activée Base de données ragdb pré-crée Règle de pare-feu pour les services Azure

Paramètres que vous définirez : administratorLoginPassword ? votre mot de passe de base de données (notez-le !) location ? choisissez une région proche de vous

Option B ? Azure CLI (Bicep)

[diagram or code sample]

? Fichiers de support

note "Téléchargez ces fichiers avant de commencer le lab" Enregistrez tous les fichiers dans un dossier lab-031/ dans votre répertoire de travail.

Fichier | Description | Téléchargement | 001_init.sql | Migration de base de données | ? Télécharger |

Exercice du lab

Étape 1 : Définir les variables d'environnement de connexion

Après le déploiement, enregistrez vos détails de connexion :

[diagram or code sample]

Étape 2 : Se connecter et vérifier pgvector

[diagram or code sample]

Étape 3 : Créer le schéma

[diagram or code sample]

Étape 4 : Ingérer le jeu de données exemple

Téléchargez et ingérez les données exemple OutdoorGear :

[diagram or code sample]

Réutilisez le contrat de découpage et d'embedding du Lab 022, puis remplacez uniquement l'adaptateur mémoire par Azure Database for PostgreSQL :

[diagram or code sample]

Étape 5 : Exposer comme outil MCP

Encapsulez la fonction de recherche comme outil MCP pour que tout agent puisse interroger votre base de connaissances :

[diagram or code sample]

Ajoutez à .vscode/mcp.json : [diagram or code sample]

Nettoyage

[diagram or code sample]

? Vérification des connaissances

question "Q1 (Exécuter le lab) : Après avoir exécuté 001_init.sql sur votre base de données, combien de lignes y a-t-il dans la table products ? Exécutez `SELECT COUNT(*) FROM products;` pour vérifier."

Exécutez la migration puis interrogez la table.

??? success "? Révéler la réponse" 7 lignes

La migration insère 7 produits OutdoorGear : P001 (TrailBlazer Tent 2P), P002 (Summit Dome 4P), P003 (TrailBlazer Solo), P004 (ArcticDown -20°C), P005 (SummerLight +5°C), P006 (Osprey Atmos 65L), P007 (DayHiker 22L). Exécutez `SELECT id, name, category FROM products ORDER BY id;` pour les voir tous.

question "Q2 (Exécuter le lab) : Quel type d'index est créé sur la table product_embeddings, et quelle colonne indexe-t-il ?"

Ouvrez lab-031/migrations/001_init.sql et trouvez l'instruction CREATE INDEX.

??? success "? Révéler la réponse" Index IVFFlat sur la colonne embedding

La migration crée : `CREATE INDEX ON product_embeddings USING ivfflat (embedding vector_cosine_ops) WITH (lists = 100);`

IVFFlat (Inverted File with Flat quantization) est un index de plus proche voisin approximatif ? il est plus rapide que la recherche exacte mais sacrifie une petite quantité de rappel. Le paramètre lists = 100 définit le nombre de cellules de Voronoi pour le regroupement. vector_cosine_ops signifie que les distances sont calculées en utilisant la similarité cosinus.

question "Q3 (Choix multiple) : Vous exécutez `SELECT * FROM search_products_by_vector($1::vector, 3)` et obtenez 3 résultats. Que représentent les résultats ?"

- A) Les 3 produits les plus récemment insérés - B) Les 3 produits avec le prix le plus élevé - C) Les 3 produits dont les vecteurs d'embedding sont les plus similaires (distance cosinus la plus proche) au vecteur de requête - D) 3 produits sélectionnés aléatoirement dans la base de données

??? success "? Révéler la réponse" Correct : C

La fonction search_products_by_vector() effectue une recherche de plus proche voisin approximatif (ANN) en utilisant l'index IVFFlat. Elle calcule la distance cosinus entre le vecteur de requête et tous les embeddings de produits stockés, puis retourne les k produits avec la plus petite distance (= similarité sémantique la plus élevée). Le résultat représente les produits les plus sémantiquement pertinents pour la requête de l'utilisateur.

Prochaines étapes

Sécurité au niveau des lignes pour les agents multi-locataires : ? Lab 032 ? Sécurité au niveau des lignes RAG agentique par-dessus : ? Lab 026 ? Pattern RAG agentique
Kit d'accompagnement

Le build du site génère des téléchargements PDF et ZIP stables. Le ZIP contient le guide du lab et tous les fichiers d'accompagnement déclarés.

Télécharger le PDF généré Télécharger le ZIP généré