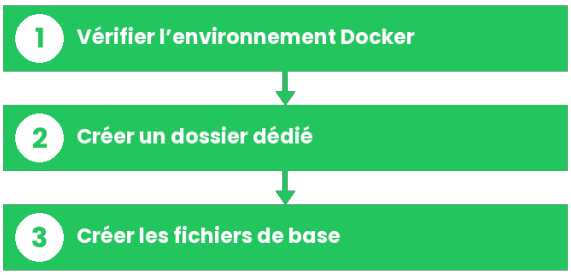
GUIDE DE DÉPLOIEMENT

Votre premier stack auto-hébergé avec Docker Compose

Une feuille de route pratique pour organiser, valider et administrer plusieurs services depuis un unique fichier Compose. À suivre avant chaque premier déploiement ou modification importante.

Au sommaire

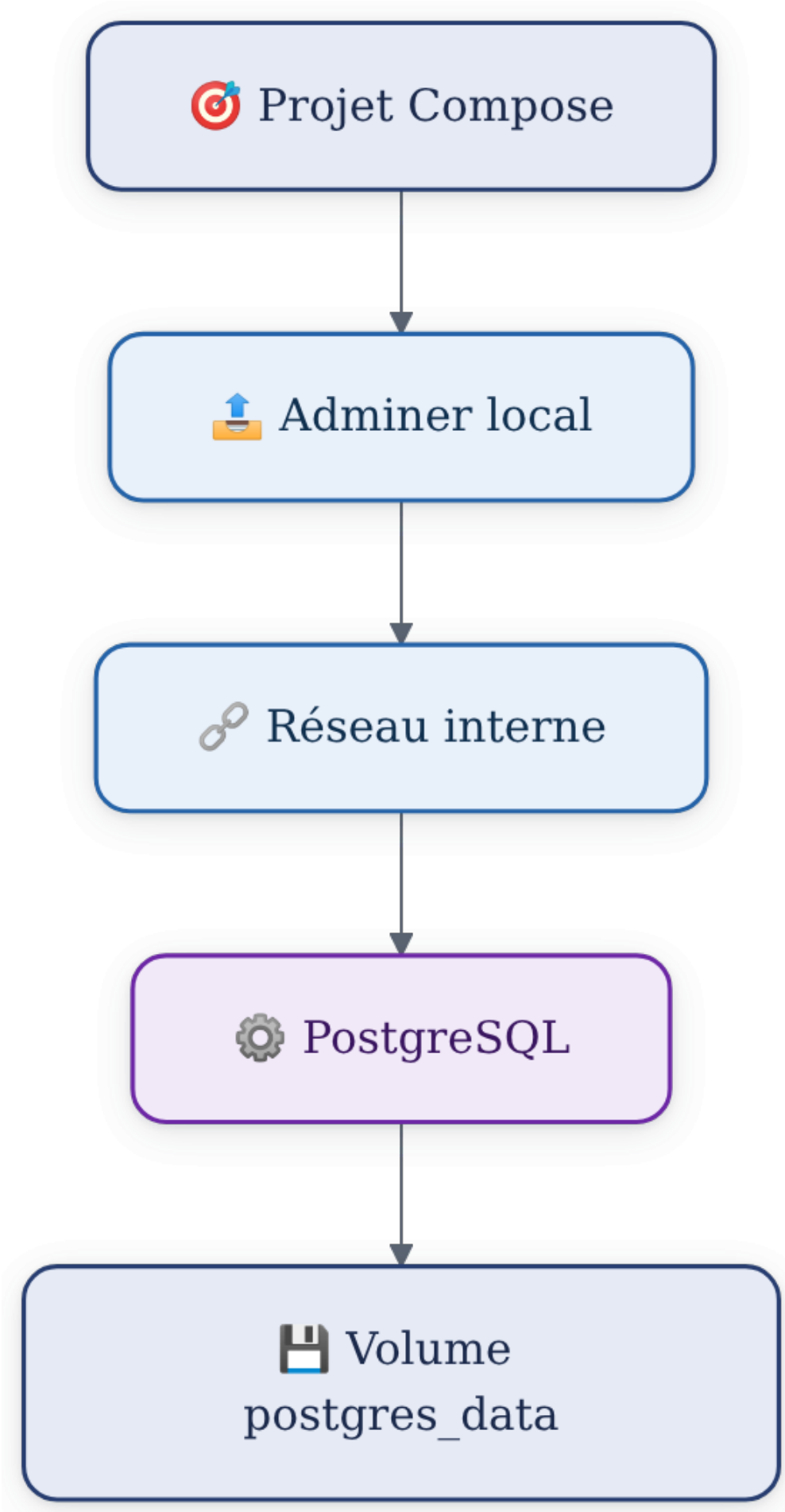
1. Préparer un projet isolé



1 **Vérifier l'environnement Docker**
Confirmez que le moteur Docker et le plugin Compose répondent avant de créer la configuration.

2 **Créer un dossier dédié**
Regroupez chaque application ou groupe de services dans son propre répertoire afin de séparer clairement configuration, variables locales et exports de journaux.

3 **Créer les fichiers de base**
Préparez compose.yaml, .env et .gitignore. Ajoutez .env au fichier .gitignore si le projet est suivi avec Git.


🎯 Projet Compose
📤 Adminer local
🔗 Réseau interne
⚙️ PostgreSQL
💾 Volume
postgres_data

INFOGRAPHIE *Schéma Docker Compose avec services auto-hébergés, réseau privé et volume PostgreSQL*

> ¡ **Commandes de vérification initiale**
> Exécutez « docker --version » puis « docker compose version ». La syntaxe actuelle utilise « docker compose » avec un espace ; ne poursuivez pas si l'une des commandes ne répond pas.

Les composants à déclarer clairement

Élément	Utilité dans le stack	Réflexe à adopter
Image	Modèle utilisé pour lancer un logiciel	Choisir une source officielle et une version maîtrisée
Service	Déclaration d'un composant, comme PostgreSQL ou Adminer	Rendre explicites les variables, ports, volumes et dépendances
Volume	Stockage persistant hors du cycle de vie du conteneur	Le sauvegarder séparément avant une opération destructive
Réseau	Communication privée entre les services du projet	Ne publier que les ports réellement nécessaires

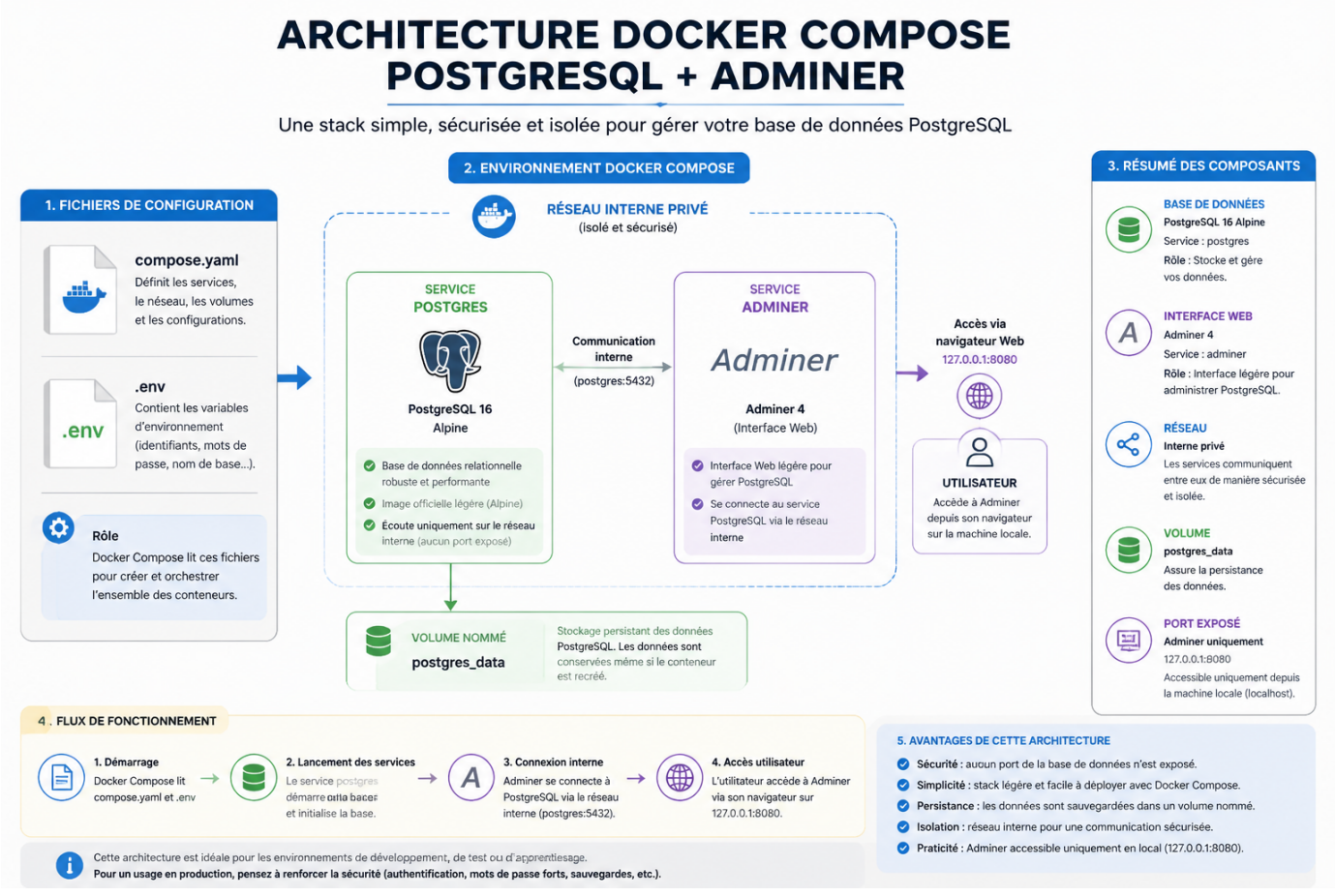
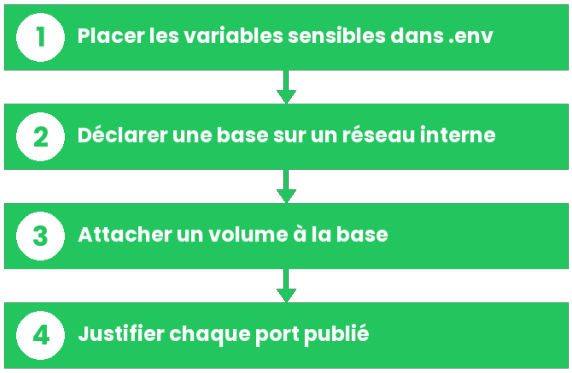


SCHÉMA *Exemple de séparation entre PostgreSQL, Adminer, un réseau interne et un volume persistant.*

2. Configurer sans exposer les données



1 **Placer les variables sensibles dans .env**
Conservez notamment le nom de la base, l'utilisateur et le mot de passe hors du fichier Compose partagé. Remplacez tout emplacement d'exemple par un secret unique.

2 **Déclarer une base sur un réseau interne**
Laissez PostgreSQL accessible aux seuls services qui en ont besoin via le réseau Docker privé, sans ouvrir son port directement vers Internet.

3 **Attacher un volume à la base**
Associez le répertoire de données PostgreSQL à un volume afin de préserver les données lors de la recréation ou de la mise à jour du conteneur.

4 **Justifier chaque port publié**
Une interface telle qu'Adminer peut être rendue accessible depuis la machine hôte pour l'administration ; n'exposez pas un service par défaut.

3. Contrôles avant et après le lancement

- [ ] **Vérifier que .env n'est ni public ni inclus dans un dépôt Git**
Les secrets doivent rester locaux.
- [ ] **Vérifier les images choisies**
Éviter latest et appliquer une politique de version maîtrisée.
- [ ] **Valider le fichier Compose avec docker compose config**
Corriger les erreurs de configuration avant le démarrage.
- [ ] **Démarrer les services depuis le répertoire du projet avec docker compose up**
Compose utilise la configuration et les ressources du projet courant.
- [ ] **Contrôler que la base reste privée**
Elle doit communiquer avec les autres services par le réseau interne.
- [ ] **Prévoir une sauvegarde des volumes et des fichiers de configuration**
Un conteneur recréable n'est pas une sauvegarde.

! **Les trois erreurs à éviter**
Ne mettez pas les mots de passe dans un fichier Compose partagé. Ne publiez pas le port d'une base de données sans besoin réel. Ne confondez pas persistance et sauvegarde : un volume conserve les données, mais doit aussi être protégé par une stratégie de sauvegarde.

Docker Compose facilite la reproductibilité des services, mais ne remplace ni l'administration du système hôte, ni les mises à jour de sécurité, ni les sauvegardes.