Créer une extension Postgres en Rust

Comment marier un crabe et un éléphant ?

D'un côté, le crabe : Rust, langage compilé, obsédé par la sécurité mémoire, intransigeant sur les types, fier de ses performances.

De l'autre, l'éléphant : PostgreSQL, la base de données qui refuse de vieillir, extensible jusqu'à l'infini, gardien de vos données depuis 30 ans.

À première vue, rien ne prédestine ces deux-là à s'entendre. Et pourtant, le framework PGRX a décidé de célébrer cette union et de donner naissance à des extensions extraordinaires.

Cette série d'articles est leur faire-part de mariage.

Version en ligne

Vous pouvez suivre ce tutoriel directement à l'address suivante:

https://daamien.gitlab.io/pgrx-tuto/

A propos

Ce tutoriel est rédigé et maintenu par Damien CLOCHARD.

Damien est un DBA PostgreSQL français et l'un des cofondateurs de Dalibo, société de services spécialisée PostgreSQL, qu'il a créée en 2005.

Il est également membre actif au sein de l'association PostgreSQLFr.

On lui doit notamment le projet PostgreSQL Anonymizer et plus récemment l'extension The Cryptic Elephant, deux extensions Postgres développée en Rust.

Avertissement

> Important: Cette série d'articles n'est pas une introduction au langage Rust ! Il existe d'autres ressources pour ça, notamment rust gentle intro, Le "Rust Book" en français et rust by example. On se focalisera ici sur la mécanique interne d'une extension Postgres.

Préambule

Ce tutoriel fait partie d'une série consacrée au framework PGRX, un environnement de développement qui facilite la conception d'extensions PostgreSQL avec le langage Rust.

Retrouvez les autres articles de la série ci-dessous:

- Episode 1 : Un nouveau monde
- Episode 2 : Demander le luhn
- Episode 3 : Le chant du SIREN
- Episode 4 : Accrocher le parapet

Chaque épisode est conçu en 2 parties : une démo de 30 minutes environ, suivi d'exercices pour aller plus loin. Les exercices sont optionnels.

Avant de démarrer ce tutorial, vérifier que PGRX est bien installé et fonctionnel:

```
cargo pgrx --version
cargo pgrx status
```

Si ce n'est pas le cas, revenir à la section installation classique ou installation via docker.

Pré-requis

Il est fortement recommandé d'installer l'environement PGRX au préalable pour gagner du temps au démarrage du tutoriel.

Pour suivre le tutoriel vous avez besoin de :

- Un ordinateur
- Docker
- 3Go d'espace disque

Installation via Docker (recommandée)

L'image docker daamien/pgrx contient un environnement PGRX complet sous debian trixie avec une instance PostgreSQL 18 initialisée. Vous pourrez simplement:

- vous connecter dans le conteneur pour lancer les commandes `cargo pgrx`
- créer/modifier les fichiers avec votre éditeur habituel

vous pouvez lancer le conteneur avec la commandes suivante:

- Linux / bash: `docker run -it --rm -v ${PWD}:/pgrx daamien/pgrx`
- MacOS / zsh: `docker run -it --rm -v $(pwd):/pgrx daamien/pgrx`
- Powershell: `docker run -it --rm -v ${PWD}:/pgrx daamien/pgrx`

Pour faciliter les commandes, on peut également créer un alias pour la commande `cargo`

```
alias cargo='docker run -it --rm -v ${PWD}:/pgrx daamien/pgrx cargo'
```

Une fois le tutoriel effectué, on peut nettoyer supprimer l'image avec

```
docker rmi daamien/pgrx
```

Installation classique

> Ce chapitre remplace le précédent :)

Si vous ne souhaitez pas utiliser docker, il est possible d'installer Rust et PGRX de manière permanente sur la machine hôte. Sur Windows, il est possibles d'installer Linux via WSL.

1. Installer rust avec l'installateur rustup
2. Installer les pré-requis de PGRX
3. Installer PGRX avec `cargo install --locked --version 0.18.0 cargo-pgrx`
4. Créer une instance avec `cargo pgrx init --pg18=download`
5. Definir la version par défaut avec `export PG_VERSION=pg18`

Suivant la puissance de votre machine, ces étapes peuvent prendre entre 10 et 30 minutes.

Solutions

Pour chaque épisode du tutoriel, un exemple d'implémentation est fourni dans le dépot gitlab:

https://gitlab.com/daamien/pgrx-tuto

Ressources complémentaires

- Le “Rust Book” en français
- Rust Playground
- PostgresML is Moving to Rust for our 2.0 Release
- Was Rust Worth it ?
- Traduction des termes