NestJSでつくるマルチテナントSaaS

Agenda

- はじめに
- NestJS × マルチテナント × MongoDB
- NestJS × マルチテナント × 認証
- NestJS × マルチテナント × カスタム要件
- NestJS × マルチテナント × 運用

NestJS × マルチテナント × MongoDB

TL;DR

- MongoDBのDatabaseでテナントを分割した
- ORMにMongooseを選定した
- MongooseのコネクションはDatabaseと1:1
- リクエストスコープでMongooseをInjectするとメモリ不足になる
- Serviceのメソッド実行時、適切なコネクションでModelを生成する

MongoDBのDatabaseでテナントを分割した

前提

- AWS DocumentDBを用いる
-

MongoDBによるマルチテナント構成

Databaseでテナントを分割した。

単位	Pros	Cons
Cluster	セキュリティが最も高い	インフラ費用、管理コストいずれも高い, テナント数に比例してコストが増加
Database	インフラ費用がテナント数に比例しない, RBACを活用しやすい	DatabaseをまたいだJOINのような処理ができないため、マスターデータとテナント固有データのJOIN処理は工夫が必要
	インフラ費用がテナント数に比	特定のテナントのCollectionのみアク

ORMにMongooseを選定した

ORMにMongooseを選定した

- MongoDB | NestJS - A progressive Node.js framework
- MongoDB (Mongoose) | NestJS - A progressive Node.js framework

リクエストスコープでMongooseをInjectするとメモリ不足になる

- node.js - How to change a Database connection dynamically with Request Scope Providers in Nestjs? - Stack Overflow

EOS