

SOLUTIONS OCPIZZA

DOSSIER D'EXPLOITATION



1. TABLE DES MATIERES

1. TABLE DES MATIERES	1
Versions	3
Introduction	5
Objet du Document	5
Références	5
Pre-REQUIs	6
Systeme et serveur	6
Hébergement de la Solution :	6
Base de Données :	6
Stockage et Gestion de Fichiers :	6
Microservices et Intégration :	6
Autres Services et Dépendances :	7
Base de données	7
MySQL sur PLANETSCALE :	7
Gestion des Schémas avec Prisma et PlanetScale MySQL:	8
PROCEDURE DE DEPLOIEMENT	12
Préparation de l'Application	12
Préparation de l'Application :	12
Configuration de Vercel :	15

Intégration Continue et Déploiement Continu	18
Surveillance, Logging et Performance	19
PROCÉDURE DE DÉMARRAGE / ARRÊT	20
Démarrage de l'Application	20
Arrêt de l'Application	20
PROCÉDURE DE MISE À JOUR	21
Mise à Jour de la Base de Données	21
Gestion des mises à jour de schéma :	21
Mise à Jour de l'Application Web avec Vercel	22
Gestion des Variables d'Environnement	22
SUPERVISION/MONITORING	23
Configuration du Monitoring	23
Glossaire	25

VERSIONS

Auteur	Date	Description	Version
Younes ouasmi	02/05/2024		1.0

INTRODUCTION

OBJET DU DOCUMENT

Ce document est destiné à servir de guide opérationnel pour le déploiement, la gestion, et la surveillance du système de gestion de pizzeria développé pour OC Pizza. Il détaille les infrastructures utilisées, les configurations et les procédures associées à l'opérationnalisation de la solution.

RÉFÉRENCES

1. PDOCPizza_01_fonctionnelle : Dossier de conception fonctionnelle de l'application
2. PDOCPizza_02_technique : Dossier d'exploitation de l'application

PRE-REQUIS

SYSTEME ET SERVEUR

HÉBERGEMENT DE LA SOLUTION :

- **Vercel** : Plateforme de déploiement et d'hébergement pour applications Next.js, offrant gestion automatisée de la mise à l'échelle, de la performance et du déploiement.

BASE DE DONNÉES :

- **PlanetScale** : Base de données MySQL compatible qui offre une scalabilité sans serveur, sans verrouillage et une intégration native avec Vercel.

STOCKAGE ET GESTION DE FICHIERS :

- **Cloudinary** : Pour le stockage des images et des fichiers multimédias.

MICROSERVICES ET INTÉGRATION :

- **Vercel Serverless Functions** : Pour la création de microservices sans serveur, directement intégrés dans le même écosystème que votre application Next.js.

AUTRES SERVICES ET DÉPENDANCES :

- **Prisma** : ORM utilisé pour interagir avec la base de données MySQL, gérant les relations de données et les requêtes de manière efficace.
- **Next.js** : Framework utilisé pour le développement côté serveur et la génération des interfaces utilisateurs dynamiques.
- Autres dépendances :
 - **Node.js** : Environnement d'exécution pour JavaScript nécessaire pour Next.js.
 - **Certificats SSL/TLS** : Pour sécuriser les communications entre les clients et le serveur via HTTPS, possiblement gérés par AWS Certificate Manager.

BASE DE DONNÉES

MYSQL SUR PLANETSCALE :

PlanetScale est une plateforme de base de données comme service (DBaaS) basée sur Vitess, une technologie éprouvée développée par YouTube pour gérer des bases de données MySQL à très grande échelle. PlanetScale est conçu pour offrir une scalabilité sans limite, une haute disponibilité et une gestion simplifiée des opérations de base de données, rendant cette solution idéale pour les applications web modernes déployées sur des plateformes comme Vercel.

- **Haute Disponibilité** :PlanetScale garantit une haute disponibilité grâce à sa capacité à répliquer les données entre plusieurs régions géographiques. Cela minimise le risque de perte de données et réduit la latence pour les utilisateurs en localisant les données plus près d'eux.

GESTION DES SCHÉMAS AVEC PRISMA ET PLANETSCALE MYSQL:

Dans ce tutoriel, nous apprendrons comment réaliser des migrations Prisma dans PlanetScale dans le cadre de votre processus de déploiement en utilisant prisma db push.Développement et Déploiement :

Mise en Place Rapide avec prisma db push

- Prisma db push permet d'introspecter votre base de données PlanetScale pour inférer et exécuter les changements nécessaires afin que votre schéma de base de données reflète l'état de votre schéma Prisma. Lorsque vous exécutez prisma db push, il garantit que le schéma dans la branche PlanetScale à laquelle vous êtes connecté correspond à votre schéma Prisma actuel.

Prérequis

- **Ajoutez Prisma à votre projet via**

```
npm install prisma --save-dev ou yarn add prisma --dev.
```

- **Exécutez** `npx prisma init` dans votre projet pour créer les fichiers initiaux nécessaires pour Prisma.
- Installez le CLI de PlanetScale.

```
brew install planetscale/tap/pscale
```

- Authentifiez le CLI avec la commande :

```
pscale auth login
```

- Exécuter votre premier Prisma db push
- Les migrations Prisma suivent le workflow de changement de schéma non bloquant de PlanetScale. Tout d'abord, le schéma est appliqué à une branche de développement puis la branche de développement est fusionnée dans la base de données de production principale.
- Créez une base de données prisma-playground :

```
pscale db create prisma-playground
```

- Connectez-vous à la branche de la base de données :

```
pscale connect prisma-playground main --port 3309
```

- Mettez à jour votre fichier prisma/schema.prisma avec le schéma suivant :

```
kotlin Copy code

datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
  relationMode = "prisma"
}

model Post {
  id          Int      @default(autoincrement()) @id
  createdAt   DateTime @default(now())
  updatedAt   DateTime @updatedAt
  title       String    @db.VarChar(255)
  content     String?
  published   Boolean    @default(false)
  author      User       @relation(fields: [authorId], references: [id])
  authorId    Int
}
```

- Mettez à jour votre

fichier .env :

```
DATABASE_URL="mysql://root@127.0.0.1:3309/prisma-playground"
```

- Dans un autre terminal, utilisez la commande db push pour pousser le schéma défini dans prisma/schema.prisma :

```
npx prisma db push
```

- Après le succès de db push, vous pouvez voir la table créée dans votre terminal.
- **Activer les migrations sûres sur la branche principale**
 - Activez les migrations sûres sur la branche principale pour permettre des changements de schéma non bloquants :

```
pscale branch safe-migrations enable prisma-playground main
```

PROCEDURE DE DEPLOIEMENT

PRÉPARATION DE L'APPLICATION

PRÉPARATION DE L'APPLICATION :

- **Préparez votre application Next.js :** Assurez-vous que votre application est prête pour le déploiement, y compris l'installation de toutes les dépendances nécessaires et la configuration de votre projet Next.js.

Initialisation de Git

- **Ouvrir un terminal :** Accédez au dossier de votre projet Next.js.

Initialiser un dépôt Git : Si votre projet n'est pas déjà un dépôt Git, tapez la commande suivante pour l'initialiser :bash

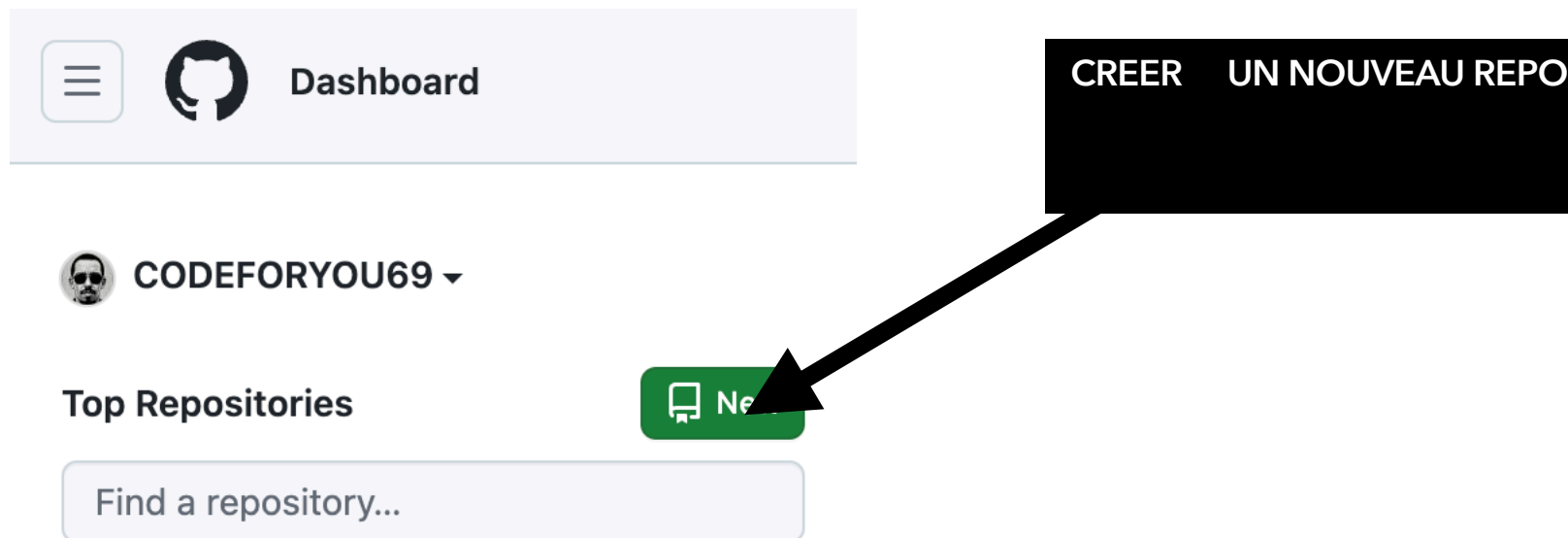
```
git init
```

Premier Commit : Ajoutez tous les fichiers à votre dépôt et faites un premier commit :bash

```
git add . git commit -m "Initial commit"
```

Connexion avec GitHub (ou autre service Git)

Créer un nouveau dépôt sur GitHub : Allez sur GitHub et créez un nouveau dépôt pour votre projet. Ne cochez pas l'option "Initialize this repository with a README", car votre projet est déjà initialisé.



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (*).

Repository template

No template ▾

Start your repository with a template repository's contents.


Owner *

Choose an owner ▾

Repository name *

Great repository names are short and memorable. Need inspiration? How about **bookish-memory** ?

Description (optional)

 Please choose an owner to see the available visibility options.

Initialize this repository with:

☐ Add a README file

This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: None ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Donner un nom a votre repo

Valider

Create repository

Lier votre dépôt local à GitHub : Utilisez les commandes indiquées par GitHub pour lier votre dépôt local au dépôt distant. Cela ressemblera généralement à ceci :

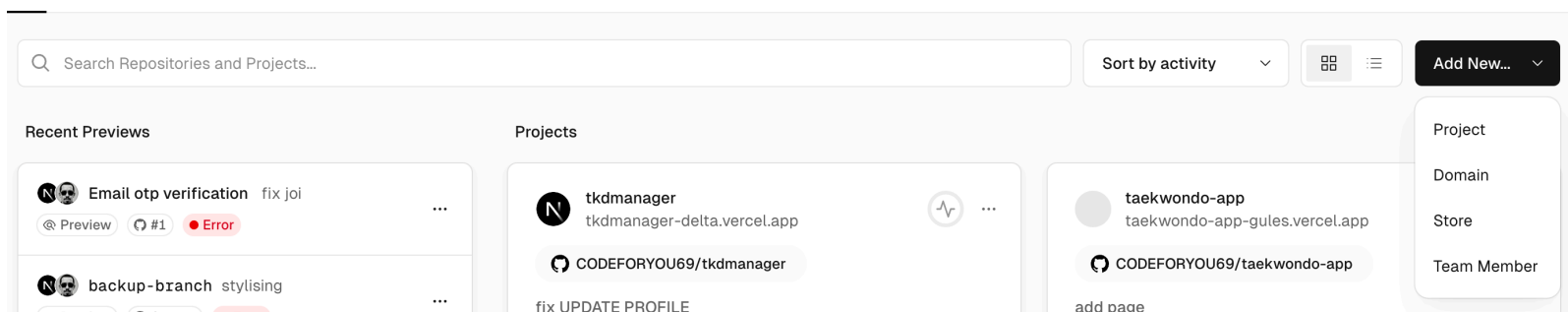
```
git remote add origin https://github.com/votre_username/votre_depot.git git push -u origin master
```

CONFIGURATION DE VERCEL :

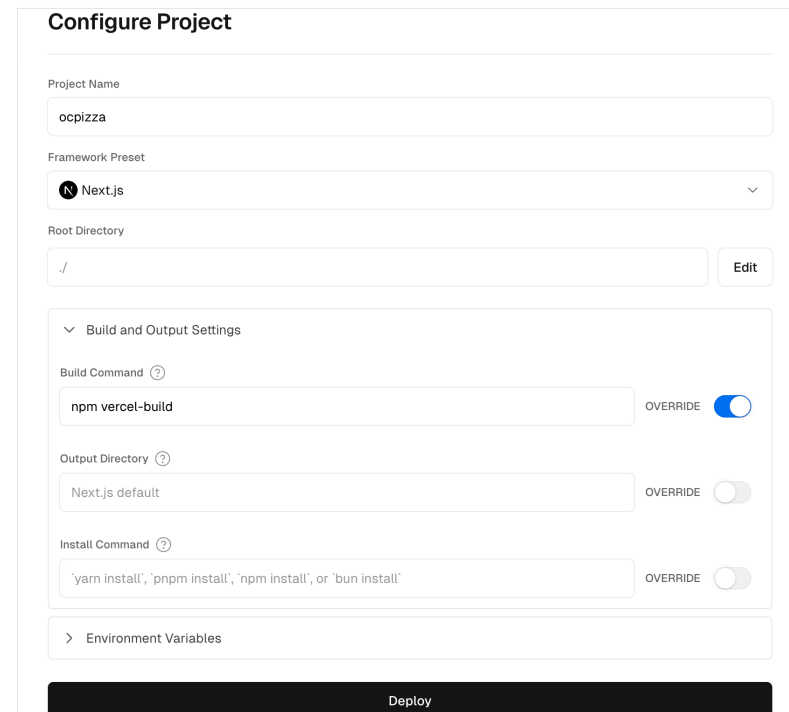
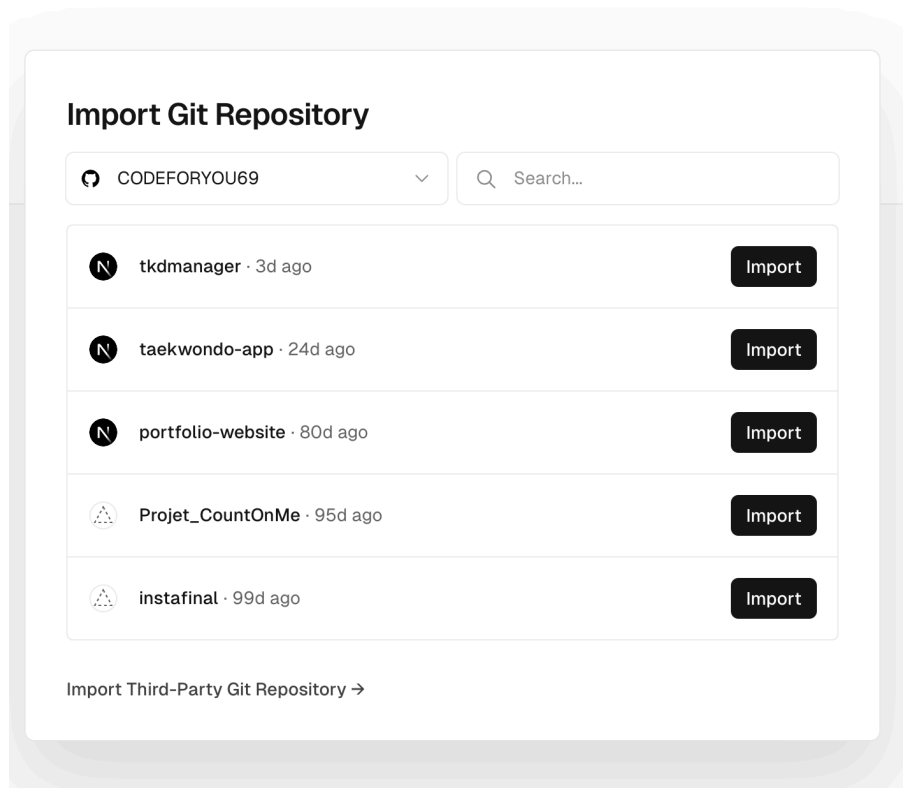
- **Connexion avec Vercel :** Créez un compte sur Vercel si ce n'est pas déjà fait et connectez votre dépôt de code (GitHub, GitLab, Bitbucket) avec Vercel.

Configurer le projet sur Vercel :

- Naviguez dans le dashboard de Vercel et créez un nouveau projet.



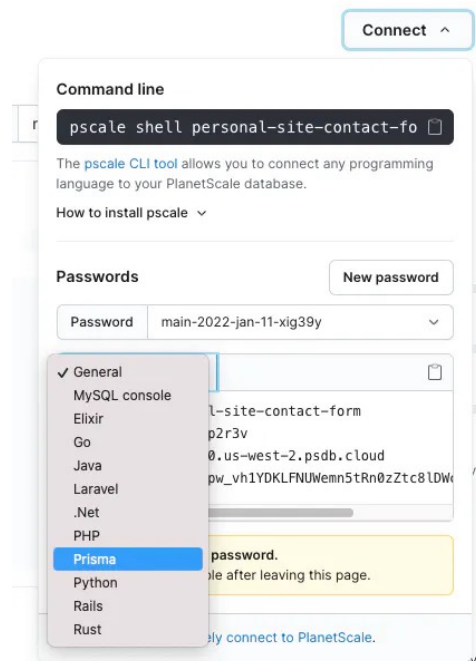
- Sélectionnez votre dépôt et configurez les paramètres de build spécifiques à Next.js



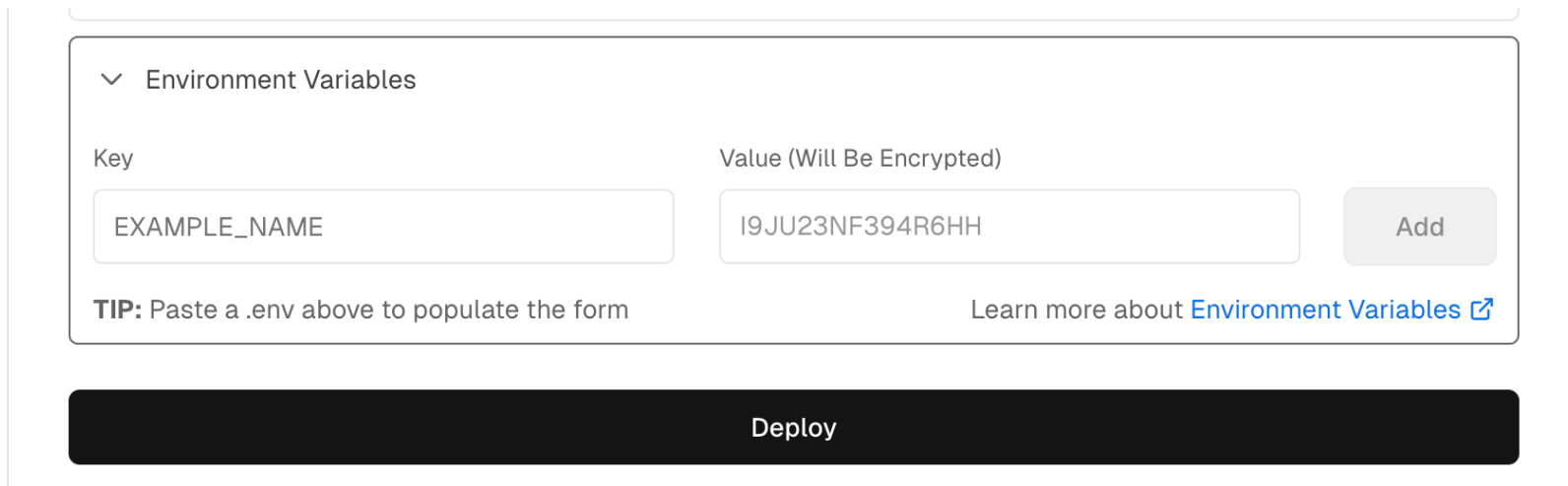
- Définissez les variables d'environnement nécessaires, y compris les informations de connexion à la base de données de PlanetScale.

Copier les variables d'environnement de votre fichier **.env** dans votre projet puis collez les dans environnement variables , elles seront copiez automatiquement

- **Connexion avec PlanetScale :**
- Retournez à votre base de données PlanetScale :
 - Cliquez sur le bouton "Connect" et sélectionnez "Prisma" dans le menu déroulant.



- Cliquez sur le bouton pour générer un nouveau mot de passe et assurez-vous de copier/coller ce mot de passe quelque part pour y accéder plus tard.
- Ajoutez une variable nommée **DATABASE_URL** et définissez la valeur sur l'URL qui vous a été donnée par PlanetScale lors de la génération du mot de passe. Assurez-vous de retirer les guillemets qui entourent l'URL.



The screenshot shows the 'Environment Variables' section of a deployment interface. It features a table with two columns: 'Key' and 'Value (Will Be Encrypted)'. The 'Key' column contains the text 'EXAMPLE_NAME' in a light gray box. The 'Value' column contains the text 'I9JU23NF394R6HH' in a light gray box. To the right of the 'Value' box is a gray button labeled 'Add'. Below the table, there is a tip: 'TIP: Paste a .env above to populate the form' and a link: 'Learn more about Environment Variables' with an external link icon. At the bottom of the form is a large black button labeled 'Deploy'.

Key	Value (Will Be Encrypted)
EXAMPLE_NAME	I9JU23NF394R6HH

TIP: Paste a .env above to populate the form

Learn more about [Environment Variables](#)

Deploy

INTÉGRATION CONTINUE ET DÉPLOIEMENT CONTINU

- **Déploiements Automatiques :** Avec Vercel, chaque 'push' sur votre branche principale (ou la branche configurée pour le déploiement) déclenchera un déploiement automatique.

```
git add . git commit -m « Your-commit »
```

- **Tests et Builds Automatiques :** Vercel exécutera automatiquement les tests et les builds spécifiés dans votre configuration de projet.

SURVEILLANCE, LOGGING ET PERFORMANCE

- **Vercel Analytics :** Utilisez Vercel Analytics pour surveiller la performance de votre application.
- **Logs :** Accédez aux logs de build et d'exécution directement depuis le dashboard de Vercel pour surveiller le comportement de votre application.

PROCÉDURE DE DÉMARRAGE / ARRÊT

DÉMARRAGE DE L'APPLICATION

- **Automatisation du Démarrage** : Vercel gère automatiquement le démarrage des applications déployées. Lorsqu'une application est déployée, Vercel provisionne les ressources nécessaires, exécute les builds, et déploie les artefacts générés. Il n'y a pas d'action manuelle nécessaire pour démarrer l'application.
- **Surveillance** : Vercel surveille continuellement la santé de l'application. En cas de problèmes détectés, comme un crash de l'application, Vercel tente automatiquement de redémarrer les instances affectées pour maintenir la disponibilité du service.

Utiliser l'option 'Actions' et choisir 'Restart App Server(s)' pour redémarrer les serveurs d'application.

ARRÊT DE L'APPLICATION

- **Arrêt Manuel** : il n'est pas courant de "stopper" une application dans Vercel pour la simple raison que Vercel gère dynamiquement l'allocation des ressources. Si vous devez retirer une application de la production :
- **Mettre l'Application Hors-Ligne** : Vous pouvez supprimer le projet directement depuis le tableau de bord de Vercel, ce qui arrêtera l'application et supprimera les ressources associées.
- **Scale Down** : Si vous souhaitez simplement réduire les ressources sans supprimer le projet, vous pouvez ajuster les paramètres de scale dans les configurations de votre projet sur Vercel.

PROCÉDURE DE MISE À JOUR

MISE À JOUR DE LA BASE DE DONNÉES

GESTION DES MISES À JOUR DE SCHÉMA :

Développement Local : Testez d'abord toutes les modifications de schéma localement en utilisant Prisma Migrate.

Déploiement de Schéma : Pour les modifications en production, utilisez les fonctionnalités de branches et de déploiements (Deploy Requests) de PlanetScale. Cela permet de proposer, réviser et appliquer des modifications de schéma sans temps d'arrêt :

1. Créer une branche sur PlanetScale pour les modifications de schéma.
2. Appliquer les modifications dans cette branche via Prisma Migrate ou directement sur le dashboard de PlanetScale.
3. Tester les modifications dans cette branche de développement ou de test.
4. Fusionner la branche dans la production via un Deploy Request, assurant une transition fluide et sans service interrompu.

MISE À JOUR DE L'APPLICATION WEB AVEC VERCEL

- **Déploiement Continu :**
 - **Push les modifications :** Poussez les mises à jour de votre code dans la branche de déploiement connectée à Vercel (généralement la branche main ou master).
 - **Déploiement Automatique :** Vercel détecte automatiquement les nouveaux commits et déclenche un processus de build et de déploiement. Vous pouvez suivre ce processus dans le dashboard de Vercel.
 - **Prévisualisation et Production :** Vercel offre des déploiements de prévisualisation pour chaque pull request et déploie automatiquement en production une fois les changements fusionnés dans la branche de déploiement.

GESTION DES VARIABLES D'ENVIRONNEMENT

Mise à jour des variables d'environnement :

- **Vercel Dashboard :** Accédez au dashboard de Vercel et ouvrez les paramètres de votre projet.

- **Configurer les variables** : Modifiez, ajoutez, ou supprimez des variables d'environnement sous la section 'Environnement Variables'.
- **Redéploiement** : Les modifications des variables d'environnement déclencheront un redéploiement de votre application pour appliquer les changements.

SUPERVISION/MONITORING

CONFIGURATION DU MONITORING

Vercel Analytics :

- **Performance Monitoring** : Vercel propose Vercel Analytics, un outil qui vous aide à mesurer la performance de votre application directement depuis le tableau de bord de Vercel. Cela inclut des mesures telles que le Largest Contentful Paint (LCP), Cumulative Layout Shift (CLS), et First Input Delay (FID) qui sont essentielles pour suivre les Web Vitals.
- **Activer Vercel Analytics** : Vous pouvez activer Vercel Analytics pour votre projet depuis le tableau de bord de Vercel. Cela nécessite un plan payant, mais c'est un investissement qui peut s'avérer précieux pour l'optimisation de votre application.
- **Surveillance en Temps Réel** :
- **Logs de Déploiement** : Vercel fournit des logs détaillés pour chaque déploiement. Vous pouvez accéder à ces logs via le tableau de bord de Vercel pour voir les actions effectuées, les erreurs de déploiement, et d'autres messages système pertinents.

- **Realtime Logs :** Pour les problèmes qui surviennent en production, utilisez les logs en temps réel de Vercel pour diagnostiquer et résoudre les problèmes rapidement. Ces logs incluent des requêtes HTTP, des erreurs de serveur, et d'autres événements d'exécution.

GLOSSAIRE

Vercel

Plateforme de déploiement et d'hébergement cloud pour applications front-end et JAMstack, qui automatise le déploiement, la mise à l'échelle, et la gestion de performance. Utilisée ici pour déployer et gérer une application Next.js.

PlanetScale

Service de base de données MySQL basé sur Vitess, conçu pour offrir scalabilité, gestion simplifiée et haute disponibilité sans verrouillage des schémas. Supporte les "Deploy Requests" pour les mises à jour de schéma sans interruption de service.

Prisma

ORM (Object-Relational Mapping) moderne qui facilite l'intégration entre le code de l'application Next.js et la base de données MySQL gérée par PlanetScale. Utilisé pour définir le modèle de données, effectuer des requêtes, et gérer les migrations de schéma.

MySQL sur PlanetScale

Version de MySQL adaptée pour le cloud, offrant des fonctionnalités avancées comme la réplication entre régions géographiques pour une haute disponibilité et des branchements de base de données pour tester les migrations.

Gestion des Schémas avec Prisma et PlanetScale

Processus d'utilisation de Prisma Migrate pour gérer les modifications de schéma de base de données en développement et utiliser les "Deploy Requests" de PlanetScale pour les appliquer en production sans interruption.

CI/CD (Intégration Continue et Déploiement Continu)

Pratiques de développement logiciel où les modifications de code sont automatiquement testées et déployées, assurant une livraison

rapide et sûre. Vercel automatise ces étapes pour les applications Next.js dès que les changements sont poussés sur un dépôt Git connecté.

Vercel Analytics

Outil de surveillance des performances de l'application directement intégré dans Vercel, mesurant des métriques critiques telles que Web Vitals pour optimiser l'expérience utilisateur.

Cloudinary

Service de gestion des médias en cloud, utilisé ici pour le stockage et l'optimisation des images et fichiers médias, offrant des fonctionnalités comme la transformation d'images à la demande.

Vercel Serverless Functions

Fonctionnalités qui permettent d'exécuter des morceaux de code backend sans gérer de serveurs, directement intégrés dans le déploiement de Vercel, facilitant la création de microservices.

Node.js

Environnement d'exécution JavaScript côté serveur nécessaire pour l'exécution de l'application Next.js.

SSL/TLS

Protocoles pour sécuriser les communications sur le réseau. Dans le contexte de Vercel, les certificats SSL/TLS sont automatiquement gérés pour sécuriser les sites web déployés.

Déploiements Automatiques

Fonctionnalité de Vercel où les modifications apportées dans un dépôt Git sont automatiquement déployées en ligne, facilitant la mise à jour continue de l'application sans intervention manuelle.

Realtime Logs

Journaux en temps réel accessibles via le tableau de bord de Vercel, utilisés pour surveiller et diagnostiquer les activités en temps réel de l'application.

Environment Variables

Variables utilisées pour stocker des configurations et des secrets, comme les chaînes de connexion à la base de données, sans les intégrer directement dans le code source. Gérées dans Vercel pour séparer les configurations entre les environnements de développement, de test et de production.