


NEXT.js

NEXT講座 デプロイ編

この資料について



オンライン学習プラットフォーム『Udemy』で
公開している

『NEXT.js 基本講座』の説明用資料です

<https://www.udemy.com/course/nextjs-basic>



デプロイ ホスティング

本番環境

デプロイ (Deploy) 開発環境->本番環境へ配置・公開するプロセス
ホスティング (Hosting) ・ ・ デプロイされたアプリを実際に動かす場所

レンタルサーバー (Xserver, さくら, ConoHa etc...)

VPS (XserverVPS etc...)

IaaS (AWS (EC2, RDS), Azure etc...)

PaaSやBaaS (Vercel, Heroku, Netlify, supabase, firebase etc...)

今回のデプロイ先



Vercel (PaaS)

Next.jsプロジェクトのアップロード先

Supabase (BaaS)

postgresql(DB), ストレージ

Vercel

Next.jsの開発元であるVercel社が提供するホスティングサービス

<https://vercel.com/>

Next.jsとの親和性が高い

GitHubなどと連携して簡単にデプロイできる

無料プランあり

最近はpostgreSQLやストレージ機能も

提供するようになっているが機能は限定的

Supabase



2020年に設立

Firebaseの代替として開発された

PostgreSQLデータベースを基盤

<https://supabase.com/>

DB、認証、リアルタイム機能、ストレージ、 etc..

無料版 ・ ・ プロジェクト2つまで



supabase

supabase



<https://supabase.com/>

アカウント作成

メールアドレス, パスワード

GitHubアカウント

プロジェクト



New Project


プロジェクト名: `udemy_next_blog`

DBパスワード: (ご自身で変更ください)

リージョン: 東京

URLとKEYを確認

ストレージ



Storageタブ->New Bucket

バケット名 udemy_next_blog_bucket

publicで設定

DB接続の切り替え

providerはenv()が使えないので直接入力

prisma/schema.prisma

```
provider = "postgresql"
```

supabaseのdatabase->connect->session pooler

のurlをコピーし、パスワードを入力

.env

```
# DATABASE_URL="file:./dev.db"
```

```
DATABASE_URL="postgresql://postgres..."
```


マイグレーション

prisma/migrationsフォルダに
sqliteの記載が残っているので削除

```
// マイグレーション(テーブル作成)  
npx prisma migrate dev --name init  
// シード実行(ダミーデータ)  
npx prisma db seed  
// prisma クライアント再作成  
npx prisma generate
```

supabaseのdatabaseタブからテーブルが生成されていることを確認

接続設定

```
npm install @supabase/supabase-js
```

lib/supabase.ts

```
import { createClient } from '@supabase/supabase-js';
```

```
const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL!;
```


```
const supabaseAnonKey =
```

```
process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!;
```

```
export const supabase = createClient(supabaseUrl, supabaseAnonKey);
```

参考: <https://supabase.com/docs/reference/javascript/initializing>

.envに追記



```
NEXT_PUBLIC_SUPABASE_URL=https://your-  
project.supabase.co
```

```
NEXT_PUBLIC_SUPABASE_ANON_KEY=your-anon-key
```

```
NEXT_PUBLIC_USE_SUPABASE_STORAGE=false # ローカルで  
は false
```

※NEXT_PUBLIC_がついた環境変数はクライアントサイドで使用可能

画像アップロードの切り替え

src/utils/image.ts

```
import { supabase } from '@lib/supabase';
```

```
// 環境によってローカル保存とsupabase保存を切り替えるようにする
```

```
export async function saveImage(file: File): Promise<string | null> {
```

```
  const useSupabase = process.env.NEXT_PUBLIC_USE_SUPABASE_STORAGE  
  === 'true';
```

```
  if (useSupabase) {
```

```
    return await saveImageToSupabase(file);
```

```
  } else {
```

```
    return await saveImageToLocal(file);
```

```
  }
```

```
}
```



```
async function saveImageToSupabase(file: File): Promise<string | null> {
```

```
  const fileName = `${Date.now()}_${file.name}`;
```

```
  const { error } = await supabase.storage
```

```
    .from('udemy\_next\_blog\_bucket')
```

```
    .upload(fileName, file, {
```

```
      cacheControl: '3600',
```

```
      upsert: false,
```

```
    });
```

```
  if (error) {
```

```
    console.error('Upload error:', error.message);
```

```
    return null;  }
```

```
  const { data: publicUrlData } = supabase.storage
```

```
    .from('udemy\_next\_blog\_bucket')
```

```
    .getPublicUrl(fileName);
```

```
  return publicUrlData.publicUrl; }
```

supabaseに保存

next.config.js

外部URLの画像なので追記必要

```
images: {  
  remotePatterns: [  
    {  
      protocol: 'https',  
      hostname: 'xxx.supabase.co',  
    },  
  ],  
}
```


supabase storage policies

ストレージ->コンフィグ

test-next-bucket

-> Allow access to JPG images in a public folder to anonymous users

Other policies under storage.objects

->Enable read access for all users

Allで作成

Policies under storage.buckets

->Enable read access for all users

Allで作成



Vercel

Vercel



<https://vercel.com/>

GitHubアカウントで接続

環境変数の設定が必要

ビルドコマンドを変更(prismaクライアントの生成)

```
npx prisma generate && npm run build
```


このセクションのまとめ



デプロイ先として

supabase, vercelへの接続方法を解説

本格運用する場合は料金などをご確認ください