Deploy de Backend com Express.js no Vercel e Banco MySQL no Clever Cloud

1. O Que é Deploy?

Definição:

Deploy é o processo de **colocar uma aplicação no ar**, ou seja, **torná-la disponível para os usuários na internet**.

Quando desenvolvemos um sistema ou site em nossas máquinas locais, ele só pode ser acessado por nós mesmos, e geralmente está rodando em um ambiente de desenvolvimento. O deploy é o passo em que pegamos essa aplicação e a movemos para um **servidor público**, permitindo que qualquer pessoa com acesso à internet possa utilizá-la.

Exemplo de Deploy:

- Você está criando um site que roda localmente no seu computador usando **Visual Studio Code** com a extensão "Go Live". Quando você faz deploy desse site, está colocando os arquivos e o código em um **servidor remoto**, onde eles podem ser acessados por qualquer pessoa através de um domínio (um endereço web, como `meu-site.com`).

O Que Acontece Durante o Deploy?

- **Arquivos e Código**: Os arquivos do projeto, como HTML, CSS, JavaScript (frontend) e o código do backend (Node.js, por exemplo), são enviados para um servidor.
- **Configuração do Servidor**: O servidor precisa ser configurado para rodar o código, conectar-se ao banco de dados e servir as páginas web para os usuários.

- **Publicação**: Após o deploy, o servidor estará ativo, e o site ou a API estará acessível por meio de uma URL.

---

2. Por Que Usar Diferentes Serviços para Hospedar a Aplicação Completa?

Uma aplicação web geralmente é composta de **diferentes partes**, e nem sempre é ideal ou possível colocar tudo em um único lugar. Por exemplo, você pode ter:

- Um **frontend** (HTML, CSS, JavaScript) que precisa ser servido como arquivos estáticos.
- Um **backend** (Node.js, Python, etc.) que processa lógica de negócios e comunicação com APIs.
- Um **banco de dados** (MySQL, PostgreSQL, etc.) que armazena informações importantes da aplicação.

Cada parte da aplicação pode exigir um tipo de serviço ou configuração específica, e isso pode nos levar a utilizar **diferentes serviços para o deploy**. Aqui estão alguns motivos para isso:

a) Serviços Especializados:

- **Frontend**: Arquivos estáticos de frontend (HTML, CSS, JavaScript) geralmente são melhor servidos por **CDNs (Content Delivery Networks)** ou serviços de hospedagem de arquivos estáticos, como **Vercel** ou **Netlify**. Esses serviços são otimizados para entregar arquivos rapidamente.
- **Backend**: Aplicações backend, que lidam com a lógica do servidor e a comunicação com APIs, podem precisar de um ambiente de execução específico, como **Node.js** ou **Python**, e requerem servidores que suportem isso.
- **Banco de Dados**: Um banco de dados precisa de um serviço especializado que ofereça **segurança** e **escalabilidade**, como **Clever Cloud**, **Amazon RDS** ou **Google Cloud SQL**. Eles fornecem backups automáticos e suporte para grandes volumes de dados.

b) Separação de Responsabilidades:

Separar a aplicação em diferentes serviços permite que você:

- Escale o **backend** independentemente do **frontend**. Se sua API está sendo muito requisitada, você pode aumentar a capacidade do servidor sem precisar mexer no frontend.
- Ter mais controle sobre cada parte da aplicação. Se há um erro no backend, isso não necessariamente afeta o frontend.

c) Performance e Disponibilidade:

Colocar o **frontend** em um serviço otimizado, como uma CDN, e o **backend** em servidores especializados melhora a **performance**. Uma CDN garante que os arquivos do site sejam entregues de forma rápida, enquanto um serviço backend dedicado pode processar requisições sem interferir na entrega dos arquivos estáticos.

---

3. Por Que Hospedagens São Pagas?

Embora existam serviços gratuitos para pequenos projetos (como o plano gratuito do Vercel ou Netlify), muitos serviços de hospedagem são pagos. Aqui estão algumas razões para isso:

a) Recursos de Servidores e Infraestrutura:

- **Servidores Físicos e Virtuais**: Empresas de hospedagem precisam manter grandes centros de dados, com muitos servidores potentes que requerem manutenção, eletricidade, e refrigeração constante. Tudo isso tem um custo.
- **Rede e Banda Larga**: O tráfego gerado pelos sites e aplicações hospedadas precisa ser transmitido pela internet, o que consome largura de banda, que também tem um custo para os provedores.

b) Suporte e Escalabilidade:

- Hospedagens pagas geralmente incluem **suporte técnico** para ajudar quando surgem problemas, o que é essencial para empresas ou projetos maiores.
- Quando seu site ou aplicação cresce e você precisa de mais poder de processamento ou armazenamento, os serviços pagos permitem escalar os

recursos conforme a necessidade, garantindo que o serviço continue rápido e estável, mesmo com muitos usuários acessando ao mesmo tempo.

c) Segurança e Manutenção:

- Os serviços de hospedagem fornecem **segurança** para proteger contra ataques de hackers e perda de dados. Isso inclui backups automáticos, firewalls, e proteção contra DDoS (ataques de negação de serviço).
- Hospedagens também oferecem **atualizações automáticas** de servidores e softwares, para garantir que sua aplicação continue segura e otimizada.

d) Garantia de Uptime e Desempenho:

- Para empresas ou projetos sérios, é fundamental que o site esteja disponível 24/7. Serviços pagos garantem **uptime** (tempo de disponibilidade) de quase 100% e oferecem compensações caso o serviço falhe.
- Além disso, serviços pagos normalmente oferecem **melhor performance** em termos de velocidade de resposta do servidor, algo crucial para aplicações de alta demanda.

Parte 1: Preparando o Projeto Backend em Express.js

Repositório para consulta: https://github.com/nhndev/nodejs-mysql-api

Passo 1: Estrutura Básica do Projeto

1. Verifique se o projeto em **Express.js** está configurado e funcional. A estrutura básica de um projeto Express.js pode ser algo assim:

   Arquivos importantes:

   - `app.js` ou `index.js`: Ponto de entrada do servidor.
   - `package.json`: Lista das dependências do projeto.

   Exemplo básico de um servidor Express.js:

   ```
   const express = require('express');
   const mysql = require('mysql2');
   ```

```
const app = express();
const port = process.env.PORT || 3000;

// Conexão ao MySQL
const db = mysql.createConnection({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASS,
  database: process.env.DB_NAME
});

db.connect((err) => {
  if (err) {
    console.error('Erro ao conectar ao banco de dados:', e
rr);
  } else {
    console.log('Conectado ao MySQL');
  }
});

app.get('/', (req, res) => {
  res.send('Servidor Express.js rodando!');
});

app.listen(port, () => {
  console.log(`Servidor rodando na porta ${port}`);
});
```

2. Certifique-se de que o projeto está funcionando localmente antes de fazer o deploy. Teste com `npm start` ou `node app.js`.

Passo 2: Instalar as Dependências

1. Caso ainda não tenha, instale as dependências principais:

```
npm install express mysql2
```

2. Garanta que o arquivo `package.json` tenha os scripts de start e as dependências listadas:

```
{
  "name": "meu-projeto-backend",
  "version": "1.0.0",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "^4.17.1",
    "mysql2": "^2.3.3"
  }
}
```

Passo 3: Configurar as Variáveis de Ambiente

1. Em produção, vamos usar variáveis de ambiente para conectar o banco de dados MySQL. Crie um arquivo `.env` localmente para testar:

```
DB_HOST=localhost
DB_USER=seu_usuario
DB_PASS=sua_senha
DB_NAME=seu_banco
```

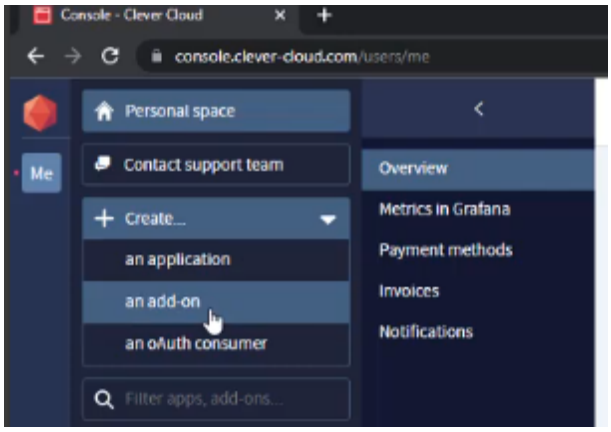
2. Certifique-se de que no código (como no exemplo acima), você usa `process.env` para buscar os valores dessas variáveis.

Parte 2: Preparar o Banco de Dados MySQL

Passo 1: Criar o Banco MySQL no Clever Cloud

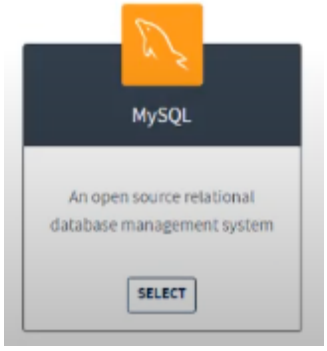
1. Acessando o Clever Cloud

1. Acesse o site do Clever Cloud e faça login ou crie uma conta se ainda não tiver uma.
2. Uma vez logado, você será direcionado ao seu **dashboard**, que é onde gerenciará seus aplicativos e addons.



2. Criar um Addon MySQL

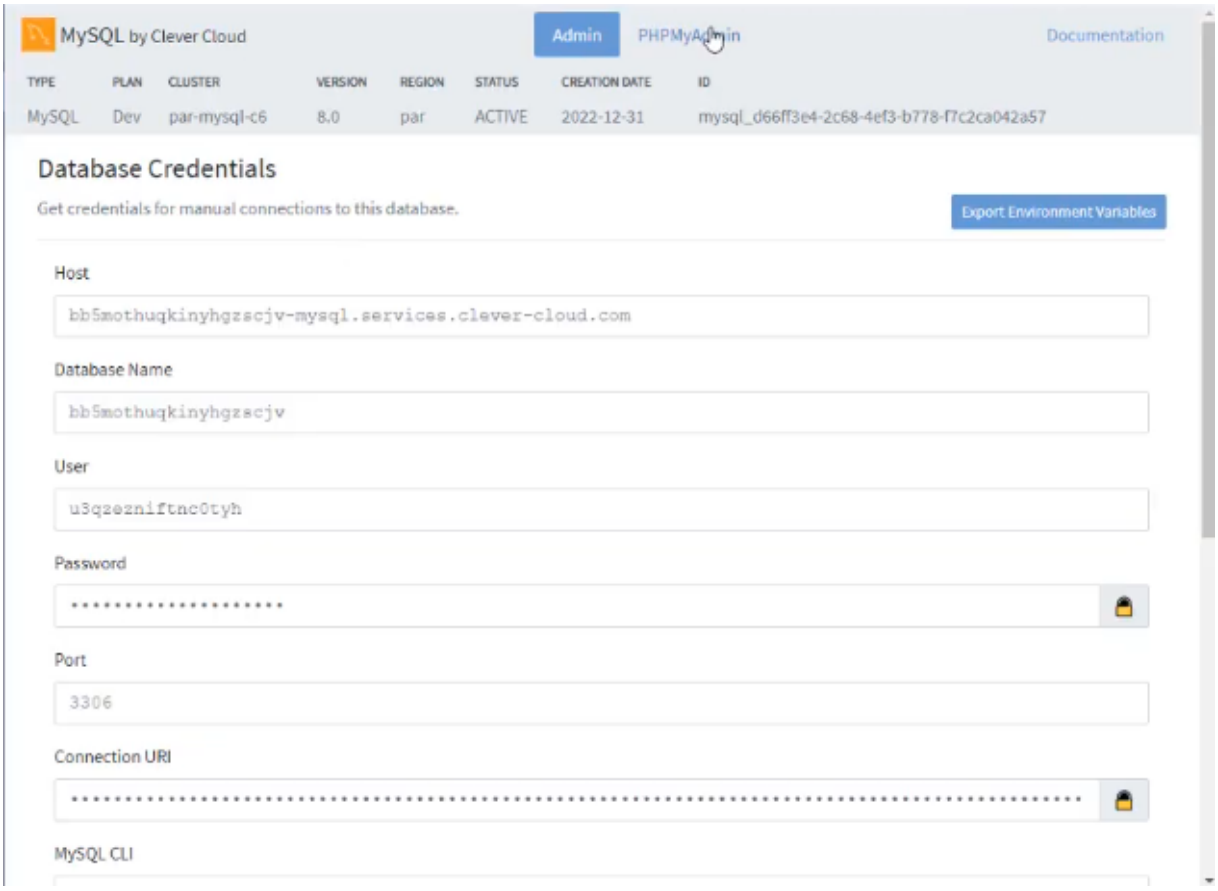
1. No dashboard, clique no botão **"New"** no canto superior direito.
2. Escolha **"Addon"** no menu suspenso. Um *addon* é um serviço extra que pode ser adicionado ao seu aplicativo, e neste caso será um banco de dados MySQL.
3. Na próxima tela, procure por **"MySQL"** na lista de addons disponíveis. Quando encontrar, clique em **MySQL**.



1. Preencha os seguintes detalhes:

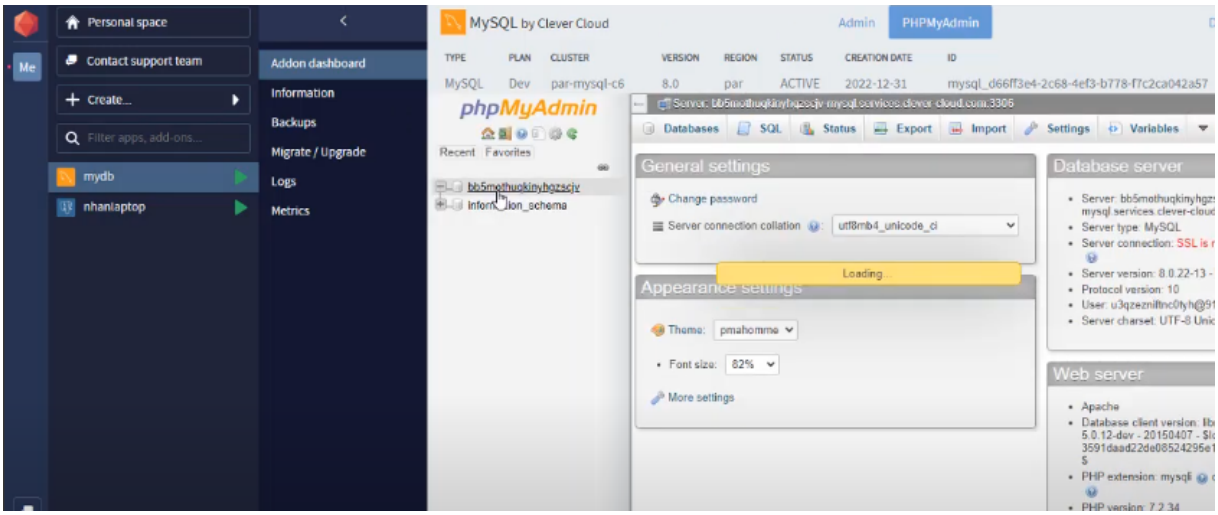
- **Name**: O nome que você deseja dar ao banco de dados. Exemplo: `meu-banco-mysql`.
- **Plan**: Escolha o plano gratuito. O primeiro plano chamado DEV.
- **Region**: Selecione a região mais próxima. Escolha algo como `Europe (Paris)` ou `North America (Montreal)` se não houver Brasil.

2. Clique em **Create Addon**. O Clever Cloud agora provisionará seu banco de dados MySQL.

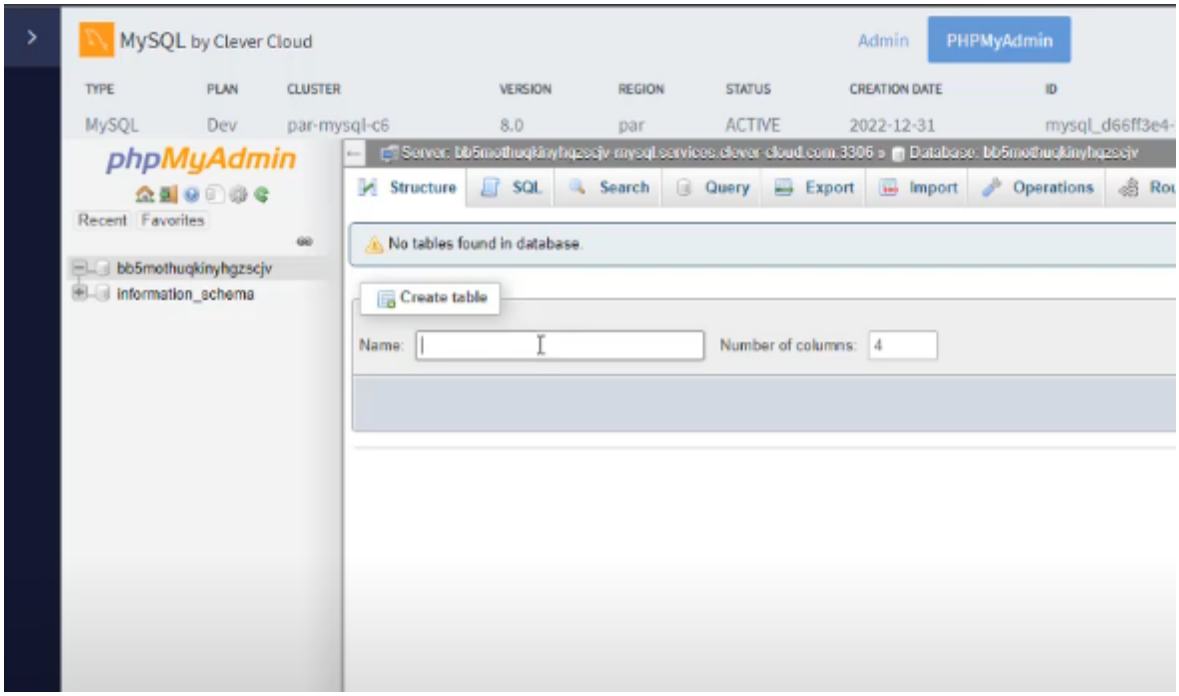


2.1. Criar uma database dentro no Clever Cloud:

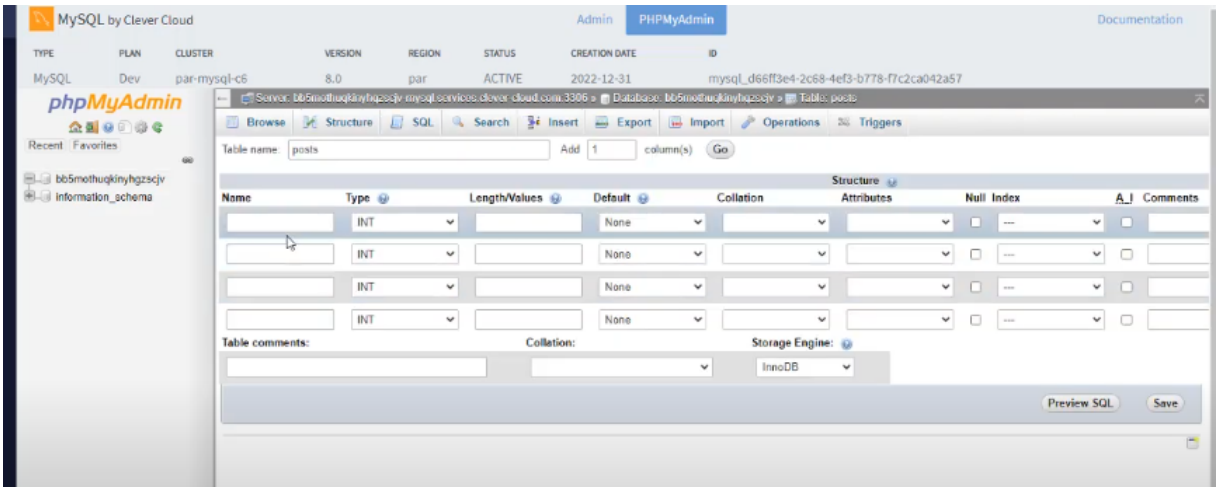
Clicar na Database gerada:



Inserir o nome:



Configurar as tabelas:



3. Obter as Credenciais do Banco de Dados

1. Após o banco de dados ser criado, você será redirecionado para a página do addon. Nessa página, você verá as **credenciais** do banco de dados que acabou de criar.
    - **Host**: O endereço do servidor MySQL (normalmente algo como `xxx.clever-cloud.com`).
    - **Port**: A porta para conexão, geralmente `3306` (a porta padrão para MySQL).
    - **Database Name**: O nome do banco de dados que foi gerado ou que você especificou.
    - **User**: O nome de usuário do banco.
    - **Password**: A senha gerada automaticamente para o banco.
2. Essas credenciais são importantes e você as usará para conectar seu backend ao banco de dados. Anote-as ou mantenha essa página aberta para quando for configurar o ambiente do seu backend.

4. Testar a Conexão ao Banco de Dados (Opcional)

1. Você pode testar a conexão ao banco de dados diretamente do seu MySQL Workbench (ou outra ferramenta de gerenciamento MySQL):

- Abra o **MySQL Workbench**.
- Clique em **+** para criar uma nova conexão.
- Em **Connection Name**, dê um nome à sua conexão, como `Clever Cloud MySQL`.
- Em **Hostname**, coloque o valor de **Host** que o Clever Cloud forneceu.
- Em **Port**, insira `3306`.
- Em **Username**, coloque o **User** fornecido.
- Em **Password**, selecione a opção de salvar a senha e insira o valor de **Password** fornecido.
- Clique em **Test Connection** para verificar se a conexão está funcionando.

2. Se a conexão for bem-sucedida, você verá uma mensagem de sucesso e poderá gerenciar o banco diretamente pelo MySQL Workbench, criando tabelas e inserindo dados.

Resumo das Credenciais do Banco

- **DB_HOST**: Hostname do banco de dados fornecido pelo Clever Cloud.
- **DB_USER**: Nome de usuário gerado.
- **DB_PASS**: Senha gerada.
- **DB_NAME**: Nome do banco de dados.
- **DB_PORT**: Geralmente será a porta `3306`.

Você utilizará essas informações ao configurar as variáveis de ambiente no backend.

Passo 2: Testar a Conexão com o Banco Hospedado

1. Localmente, no arquivo `.env`, substitua os valores pelas credenciais fornecidas pelo Clever Cloud:

```
DB_HOST=meu_host
DB_USER=meu_usuario
```

```
DB_PASS=minha_senha
DB_NAME=meu_banco
```

2. Teste a conexão com o banco no código. Você pode usar uma rota no Express.js para verificar se o banco está acessível:

```
app.get('/db-test', (req, res) => {
  db.query('SELECT 1', (err, results) => {
    if (err) {
      return res.status(500).send('Erro ao conectar ao ban
co de dados');
    }
    res.send('Conexão com o MySQL bem-sucedida!');
  });
});
```

Parte 3: Deploy do Backend no Vercel

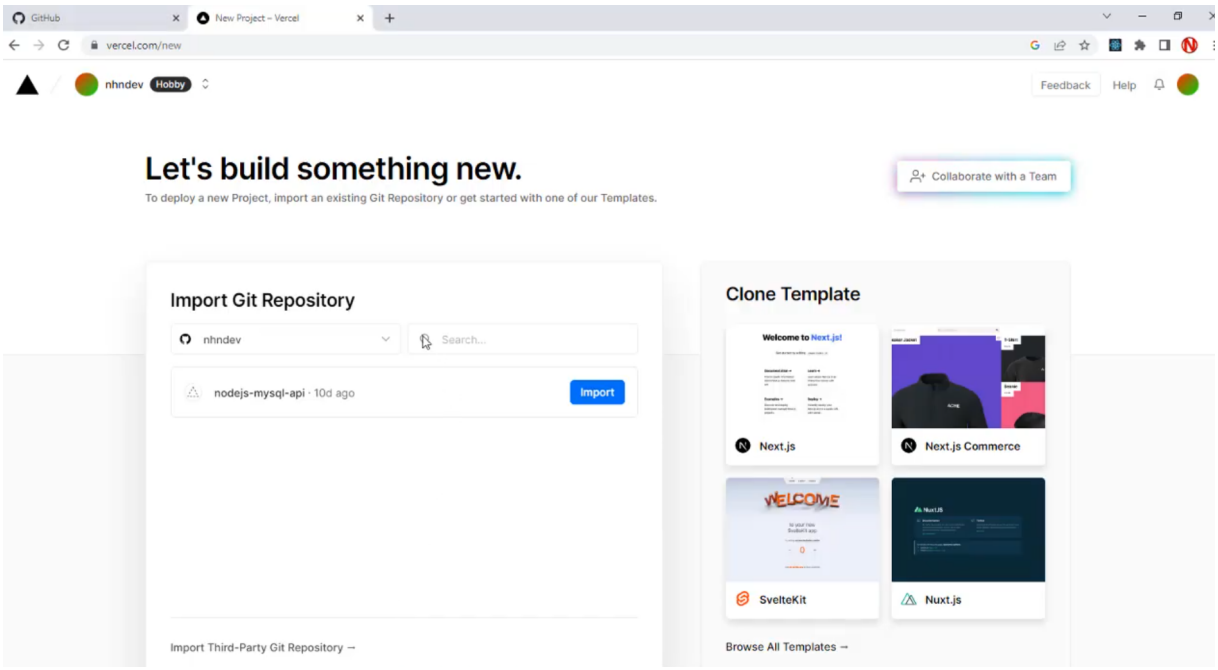
Passo 1: Subir o Projeto no GitHub

1. Suba o projeto para um repositório GitHub, conforme feito anteriormente:

```
git add .
git commit -m "Deploy do backend Express.js"
git push -u origin main
```

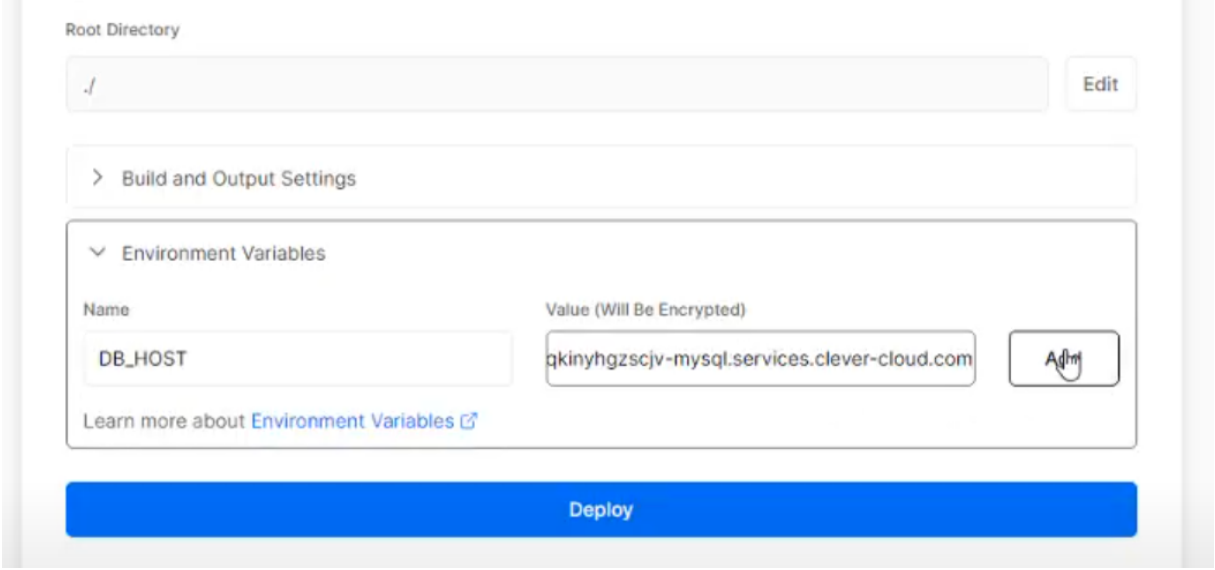
Passo 2: Conectar o Repositório ao Vercel

1. Acesse o Vercel e faça login.
2. Clique em **New Project** e selecione o repositório do seu backend no GitHub.
    - O Vercel detectará automaticamente que se trata de um projeto Node.js e configurará o deploy.



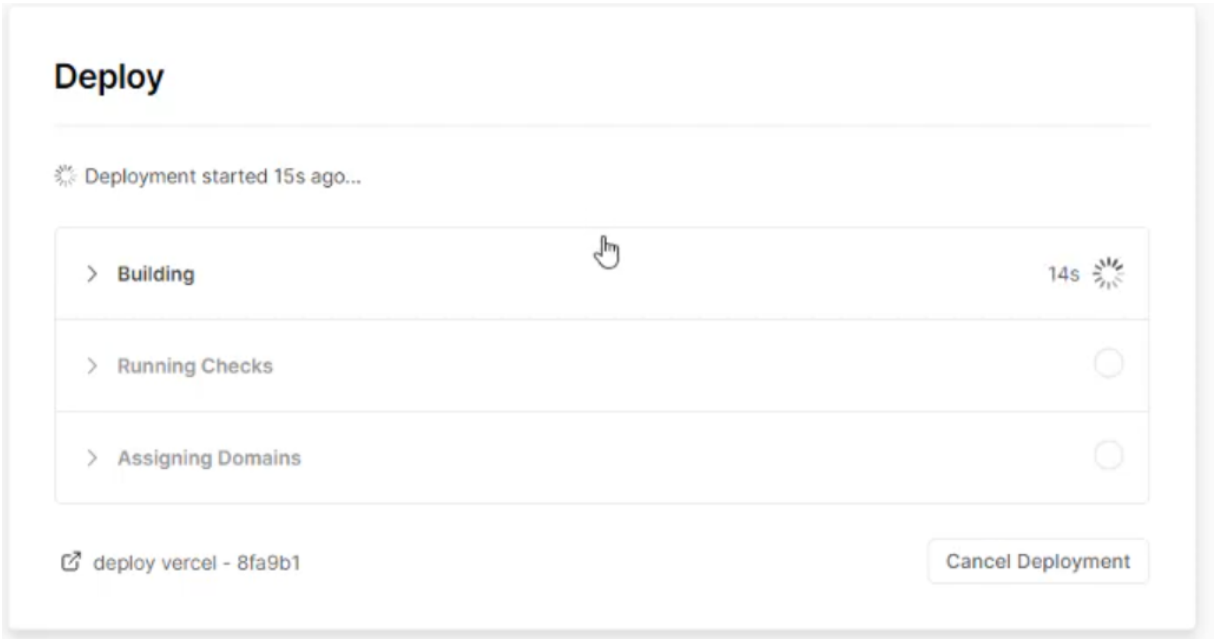
Passo 3: Configurar as Variáveis de Ambiente no Vercel

1. Durante a configuração do projeto no Vercel, vá até a seção de **Environment Variables (todo o seu arquivo .env)** e adicione as variáveis necessárias para conectar ao MySQL:
   - `DB_HOST`: Host do banco fornecido pelo Clever Cloud.
   - `DB_USER`: Usuário do banco.
   - `DB_PASS`: Senha do banco.
   - `DB_NAME`: Nome do banco.
2. Clique em **Save** para salvar as variáveis.



Passo 4: Deploy

1. Clique em **Deploy** e aguarde enquanto o Vercel faz o build e o deploy do seu projeto.
2. Após o deploy, o Vercel fornecerá um link para o backend.



Parte 4: Testar a Conexão e Integração

Passo 1: Testar as Rotas

1. Acesse a URL fornecida pelo Vercel e teste as rotas do backend, especialmente as que dependem da conexão com o MySQL.
2. Se tudo estiver configurado corretamente, você verá os dados retornando do banco MySQL para o frontend.

Passo 2: Debug de Erros

1. Se houver algum problema, você pode verificar os **logs** no painel do Vercel. Isso ajudará a identificar erros de conexão com o banco ou outros problemas no backend.

Seguiremos com o Frontend no próximo material.