VPSデプロイ手順

前提条件

- VPSへのSSHアクセス権限
- VPSにNode.js、Nginx、Certbotがインストール済み
- ドメイン名が設定済み（DNSがVPSのIPアドレスを指している）

1. VPS側の準備

1.1 Nginxのインストール

```
# VPSにSSH接続
ssh your-username@your-vps-ip

# Nginxインストール（Ubuntu/Debian）
sudo apt update
sudo apt install nginx -y

# Nginxの起動と自動起動設定
sudo systemctl start nginx
sudo systemctl enable nginx
```

1.2 SSL証明書の取得 (Let's Encrypt)

```
# Certbotのインストール
sudo apt install certbot python3-certbot-nginx -y

# SSL証明書の取得
sudo certbot --nginx -d your-domain.com

# 自動更新の設定確認
sudo certbot renew --dry-run
```

1.3 デプロイ先ディレクトリの作成

```
# Webルートディレクトリの作成
sudo mkdir -p /var/www/reaction-sharing-app

# 権限設定
sudo chown -R $USER:$USER /var/www/reaction-sharing-app
sudo chmod -R 755 /var/www/reaction-sharing-app
```

1.4 Nginx設定

```
# 設定ファイルをコピー（後でローカルから送信）
sudo nano /etc/nginx/sites-available/reaction-sharing-app

# シンボリックリンクの作成
sudo ln -s /etc/nginx/sites-available/reaction-sharing-app
/etc/nginx/sites-enabled/

# デフォルト設定の削除（必要に応じて）
sudo rm /etc/nginx/sites-enabled/default

# 設定テスト
sudo nginx -t

# Nginx再起動
sudo systemctl reload nginx
```

2. ローカル側の準備

2.1 環境変数の設定

```
# .env.productionファイルの作成
cp .env.production.example .env.production

# 実際の値を設定（エディタで編集）
nano .env.production
```

`.env.production`の内容例：

```
VITE_APP_NAME=ReactionSharingPlatform
VITE_API_BASE_URL=https://api.your-domain.com
VITE_SIGNALING_URL=wss://api.your-domain.com/ws
VITE_ION_SFU_URL=https://sfu.your-domain.com
VITE_STUN_SERVERS=stun:stun.l.google.com:19302
```

2.2 デプロイスクリプトの設定

`deploy.sh`を編集：

```
VPS_USER="your-username"      # VPSのユーザー名
VPS_HOST="your-vps-ip"        # VPSのIPアドレス
VPS_PATH="/var/www/reaction-sharing-app"
```

2.3 Nginx設定ファイルの送信

```
# Nginx設定をVPSに送信
scp nginx.conf your-username@your-vps-ip:/tmp/

# VPS側で設定ファイルを配置
ssh your-username@your-vps-ip
sudo mv /tmp/nginx.conf /etc/nginx/sites-available/reaction-sharing-app

# ドメイン名を実際の値に置き換え
sudo sed -i 's/your-domain.com/actual-domain.com/g' /etc/nginx/sites-
available/reaction-sharing-app

# Nginx再起動
sudo nginx -t && sudo systemctl reload nginx
```

3. デプロイの実行

3.1 初回デプロイ

```
# ビルド & デプロイ
./deploy.sh
```

3.2 動作確認

```
# ブラウザで確認
https://your-domain.com
```

3.3 トラブルシューティング

Nginxのログ確認

```
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log
```

カメラアクセスエラーの場合

- HTTPSが正しく設定されているか確認
- ブラウザのコンソールでエラーメッセージを確認
- 証明書が有効か確認：`openssl s_client -connect your-domain.com:443`

WebSocket接続エラーの場合

- バックエンドのURLが正しいか確認
- `wss://`（HTTPS環境用）を使用しているか確認

- ファイアウォールでWebSocketポートが開いているか確認

4. 更新デプロイ

```
# コード変更後
git pull  # 最新コードを取得（必要に応じて）
./deploy.sh  # ビルド & デプロイ
```

5. CI/CDによる自動デプロイ（オプション）

GitHub Actionsを使う場合：

`.github/workflows/deploy.yml`を作成：

```
name: Deploy to VPS

on:
  push:
    branches: [ main ]

jobs:
  deploy:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3

      - name: Setup Node.js
        uses: actions/setup-node@v3
        with:
          node-version: '18'

      - name: Install dependencies
        run: npm install

      - name: Build
        run: npm run build
        env:
          VITE_API_BASE_URL: ${{ secrets.VITE_API_BASE_URL }}
          VITE_SIGNALING_URL: ${{ secrets.VITE_SIGNALING_URL }}

      - name: Deploy to VPS
        uses: easingthemes/ssh-deploy@v2.1.5
        env:
          SSH_PRIVATE_KEY: ${{ secrets.SSH_PRIVATE_KEY }}
          REMOTE_HOST: ${{ secrets.REMOTE_HOST }}
          REMOTE_USER: ${{ secrets.REMOTE_USER }}
          TARGET: /var/www/reaction-sharing-app
          SOURCE: dist/
```

6. パフォーマンス最適化

Nginx追加設定（オプション）

```
# Brotli圧縮（gzipより高効率）
brotli on;
brotli_comp_level 6;
brotli_types text/plain text/css application/json application/javascript
text/xml application/xml application/xml+rss text/javascript;

# HTTP/2 Server Push
http2_push_preload on;
```

セキュリティチェックリスト

- [ ] HTTPS証明書が有効
- [ ] `.env`ファイルがgitignoreされている
- [ ] APIキー等の機密情報が環境変数に保存されている
- [ ] Nginxセキュリティヘッダーが設定されている
- [ ] ファイアウォールが適切に設定されている
- [ ] 定期的なセキュリティアップデート

サポート

問題が発生した場合は、以下を確認してください：

1. Nginxエラーログ
2. ブラウザのコンソール
3. ネットワークタブ（DevTools）
4. SSL証明書の有効性