Render staging & CI — Instrucciones completas

Este documento recopila los pasos para provisionar un servicio "staging" en Render, añadir las variables de entorno necesarias, configurar el secret STAGING_BASE_URL en GitHub y ejecutar las pruebas. Incluye comandos listos para copiar.

1) Crear la rama staging (si no existe)

powershell

crear y subir rama staging

git checkout -b staging

git push origin staging

2) Crear el servicio Web (staging) en Render

- En Render: New → Web Service.

- Source Code: conecta tu repo (trebolsoftv1-collab / TrebolsoftV1).

- Name: por ejemplo trebolsoft-staging.

- Branch: selecciona staging.

- Environment / Runtime: elige Docker (el repo ya tiene Dockerfile).

- Start command: con Docker no necesitas, Dockerfile ejecutará /app/entrypoint.sh. Si no usas Docker, usa ./entrypoint.sh.

- Region: selecciona la región que prefieras.

- Crea el servicio y lanza el deploy.

3) Provisionar la base de datos PostgreSQL en Render

- En Render: New → Database → PostgreSQL.

- Selecciona plan y crea la DB.

- Copia la DATABASE_URL que Render muestra. Ajusta el prefijo para SQLAlchemy si es necesario:

- Cambia postgres://... a postgresql+psycopg2://....

4) Añadir Environment Variables en Render (Service → Settings → Environment)

Añade estas variables (Name -> Value):

- DATABASE_URL -> postgresql+psycopg2://user:pass@host:5432/dbname

- SECRET_KEY -> cadena aleatoria segura (ver sección Generar SECRET_KEY)

- ALGORITHM -> HS256

- ACCESS_TOKEN_EXPIRE_MINUTES -> 30

- CORS_ALLOWED_ORIGINS -> https://trebolsoft.com,https://api.trebolsoft.com,http://localhost:8000,http://localhost:3000

(Usa el botón "Add Environment Variable" o "Add from .env" si prefieres.)

5) Verificar entrypoint.sh y migraciones

- entrypoint.sh del repo ya ejecuta alembic upgrade head y luego arranca uvicorn. Esto garantiza que las migraciones se aplican en cada deploy.

- Si usas Docker, asegúrate de que el CMD en Dockerfile ejecuta entrypoint.sh.

6) Desplegar y revisar logs

- En Render, en la sección del servicio, revisa "Deploys" y "Logs".

- Busca que durante el build se instalen dependencias y que alembic upgrade head termine sin errores.

- Si hay errores, corrígelos localmente y pushea los cambios.

7) Comprobar el endpoint /health

- Espera a que el deploy termine, luego prueba:

PowerShell:

powershell

```
Invoke-RestMethod -Uri "https://tu-staging.onrender.com/health"
```

curl:

bash

```
curl -i https://tu-staging.onrender.com/health
```

Respuesta esperada: HTTP 200 y JSON: {"status":"ok"}.

8) Generar un SECRET_KEY seguro (comandos listos)

PowerShell (sin Python):

powershell

Genera 64 hex chars (32 bytes)

```
$s = [System.BitConverter]::ToString((New-Object
System.Security.Cryptography.RNGCryptoServiceProvider).GetBytes(32)).Replace("-",""
).ToLower(); $s
```

PowerShell con Python (si tienes Python instalado):

powershell

```
python -c "import secrets; print(secrets.token_hex(32))"
```

OpenSSL (si está instalado):

powershell

```
openssl rand -hex 32
```

Copia el valor generado y pégalo en SECRET_KEY en Render y en tu .env local si deseas reproducir exactamente el entorno.

9) Añadir STAGING_BASE_URL como secret en GitHub

- En GitHub repo → Settings → Secrets and variables → Actions → New repository secret.
- Name: STAGING_BASE_URL
- Value: https://tu-staging.onrender.com (sin barra final)

10) Ejecutar tests localmente y contra staging

Local (requiere servidor local corriendo):

powershell

activar venv

. .\.venv\Scripts\Activate.ps1

instalar deps

pip install -r requirements.txt

iniciar servidor en otra terminal

python -m uvicorn app.main:app --reload --port 8000

ejecutar tests

pytest -q

Contra staging:

powershell

$env:BASE_URL = "https://tu-staging.onrender.com"

pytest -q tests/test_auth_flow.py::test_register_login_and_get_user

11) Ajustes en el workflow si staging es lento

- El workflow .github/workflows/staging-tests.yml espera hasta 30 * 6s = 3 minutos por default. Si tu deploy tarda más, aumenta el número de intentos (por ejemplo for i in {1..60} para ~6 minutos).

12) Troubleshooting rápido

- Error de DB: revisa DATABASE_URL, firewall y que la DB esté disponible.
- Error de migraciones: replica localmente con alembic upgrade head y corrige.
- Error en tests: revisa logs del step "Run tests" en GitHub Actions para ver el body de la respuesta.

Contacto

Si quieres, puedo generar un SECRET_KEY aquí para que lo pegues en Render/GitHub. Indica si lo quieres o si prefieres generarlo localmente.