Python App Deploy

Tags: Deploy, Build, Git, Server, Cloud

Inhalt

1 Mini-Projekt

1.1 Frontend und Backend

- Frontend und Backend werden als Mono-App auf render.com gehostet.
- Backend und Frontend (ui) sind eigenständige Webservice-Anwendungen
- Die Datenbank wird unter neon.com gehostet
- render.com und neon.com stellen jeweils Free-Lizenzen zu Verfügung, die vor allem zu Testzwecken nützen können

1.2 Verzeichnisstruktur

Verzeichnisstruktur

bash

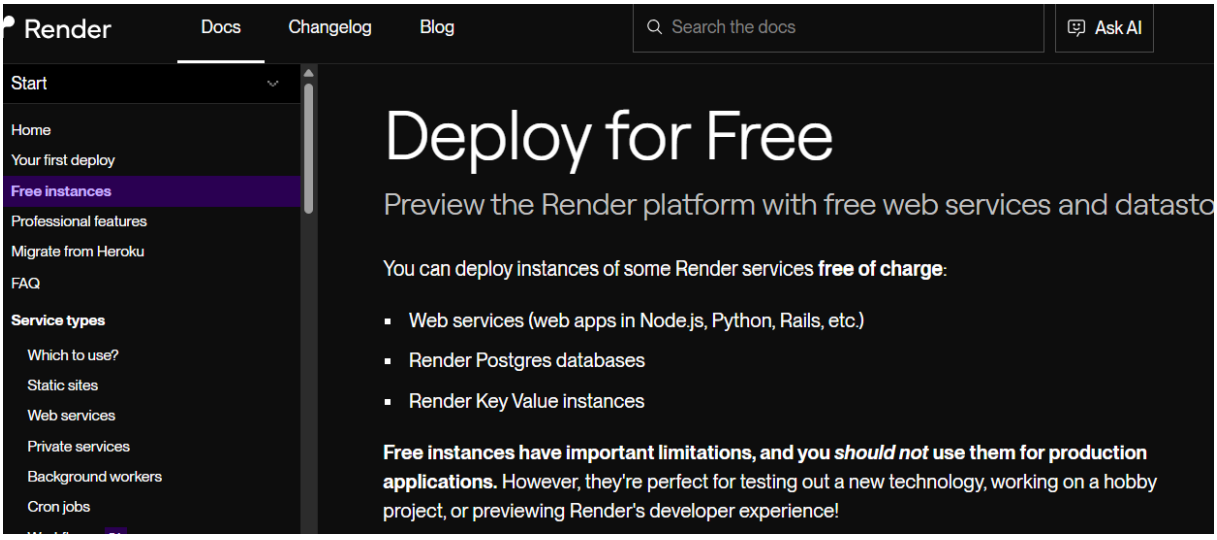
```
your-app/
├─ render.yaml
├─ backend/
│  ├─ app/main.py
│  ├─ app/db.py
│  ├─ app/models.py
│  ├─ requirements.txt
│  └─ start.sh
└─ ui/
   ├─ app.py
   ├─ requirements.txt
   └─ .streamlit/config.toml   # optional
```

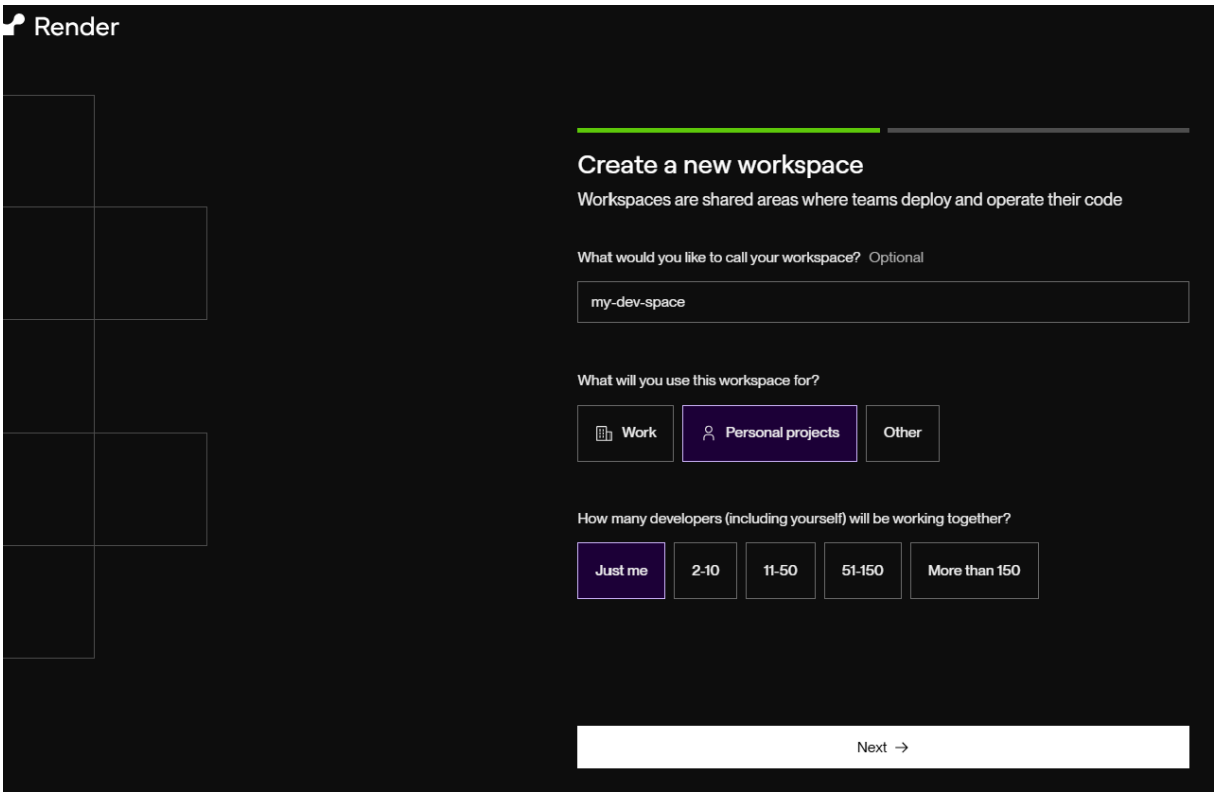
1.3 Sourcecode

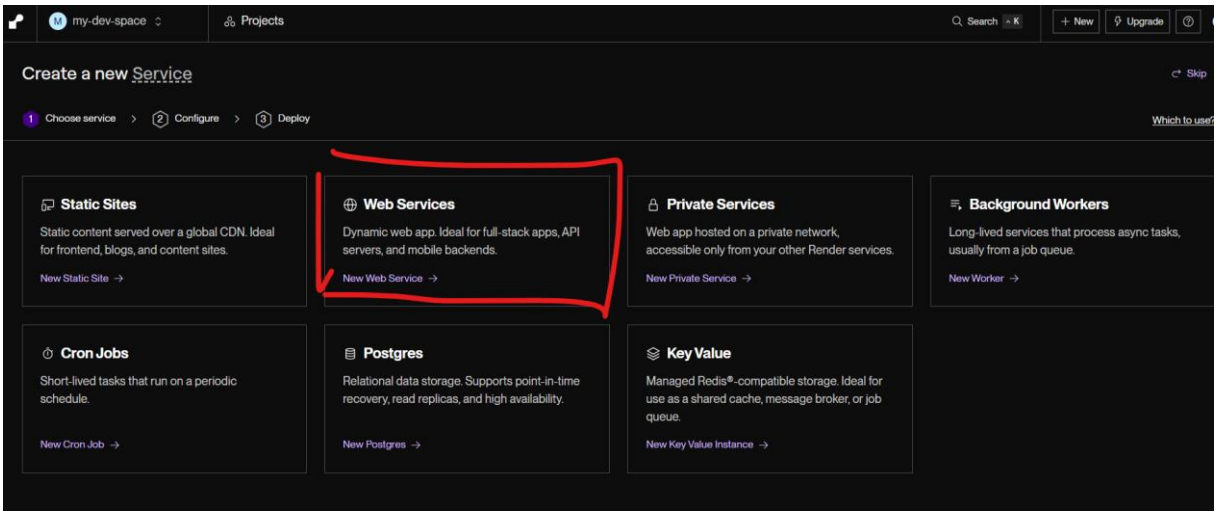
https://github.com/michadozent25/my_render_test_project/tree/master

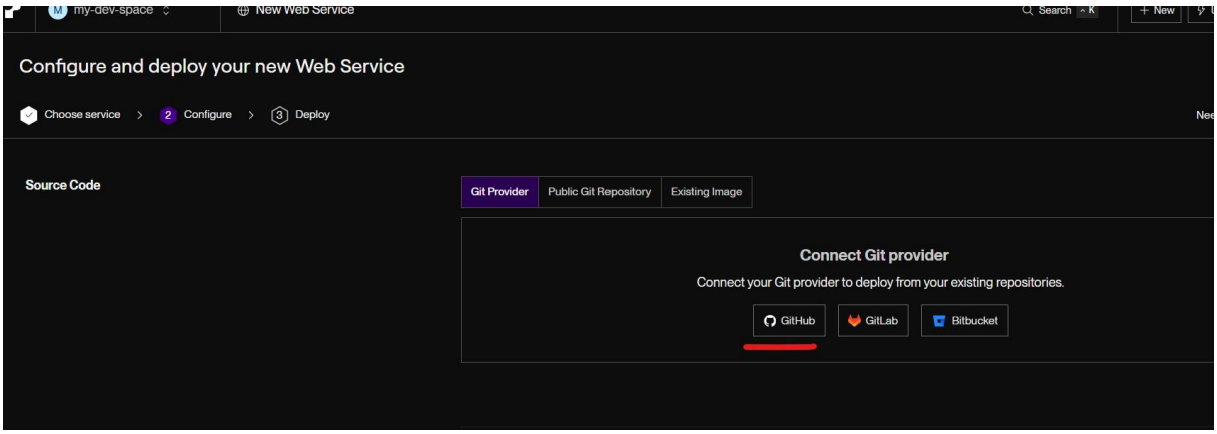
2 Deploy auf Render

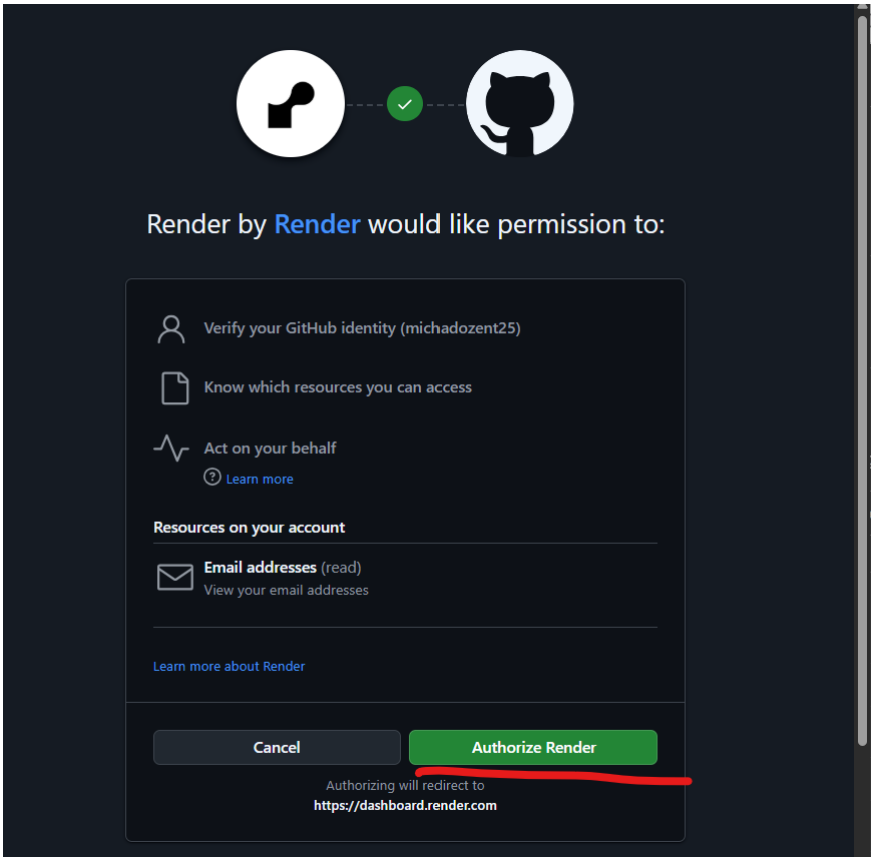
https://render.com/docs/free



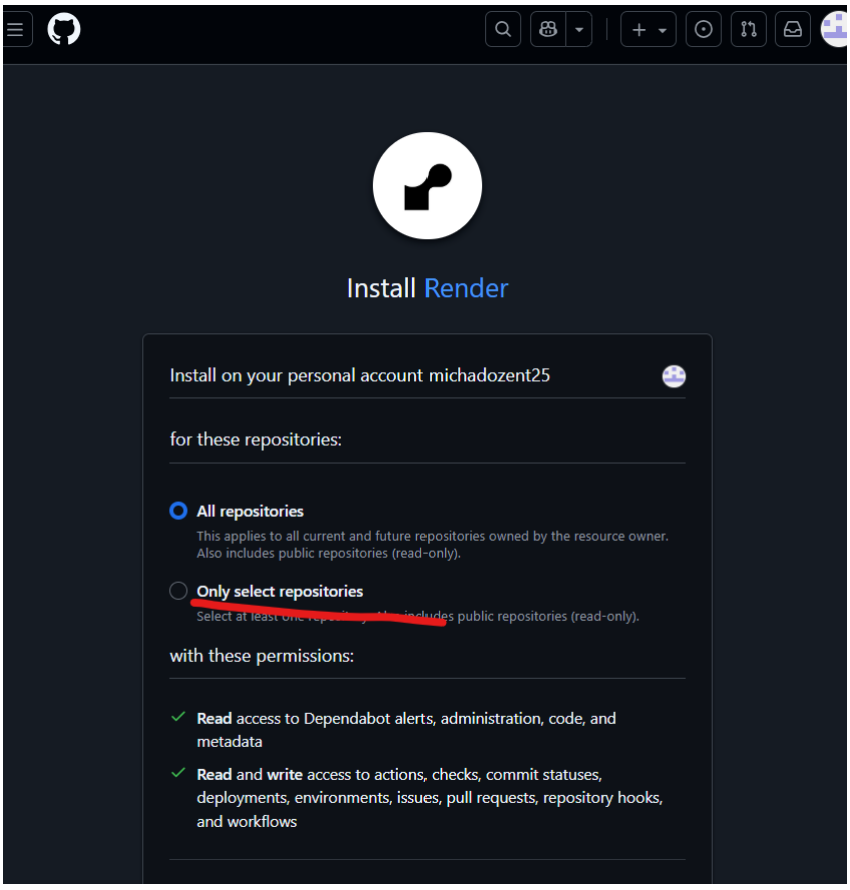

Render
Create a new workspace
Workspaces are shared areas where teams deploy and operate their code
What would you like to call your workspace? Optional
my-dev-space
What will you use this workspace for?
Work
Personal projects
Other
How many developers (including yourself) will be working together?
Just me
2-10
11-50
51-150
More than 150
Next →

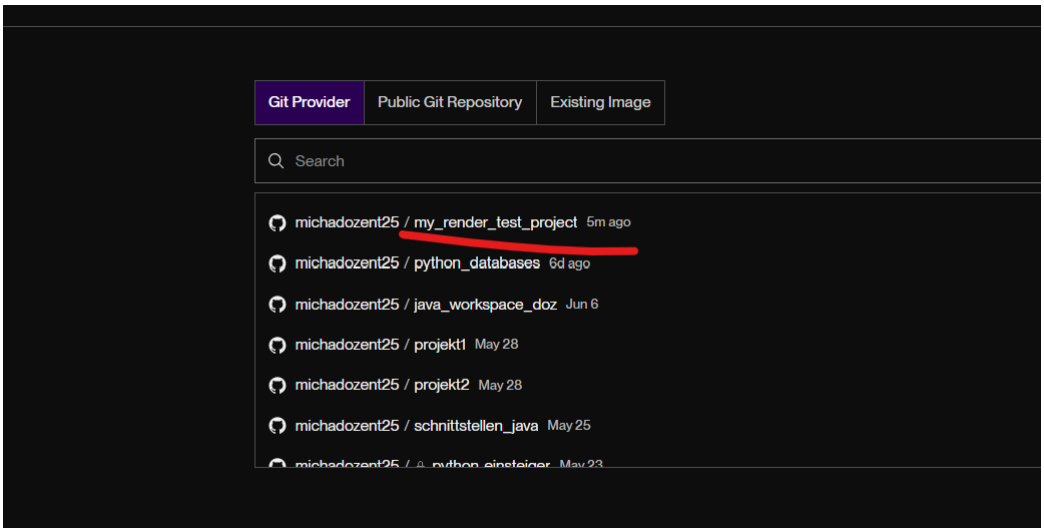

my-dev-space
Projects
Search
New
Upgrade
Create a new Service
Skip
1 Choose service
2 Configure
3 Deploy
Which to use?
Static Sites
Static content served over a global CDN. Ideal for frontend, blogs, and content sites.
New Static Site →
Web Services
Dynamic web app. Ideal for full-stack apps, API servers, and mobile backends.
New Web Service →
Private Services
Web app hosted on a private network, accessible only from your other Render services.
New Private Service →
Background Workers
Long-lived services that process async tasks, usually from a job queue.
New Worker →
Cron Jobs
Short-lived tasks that run on a periodic schedule.
New Cron Job →
Postgres
Relational data storage. Supports point-in-time recovery, read replicas, and high availability.
New Postgres →
Key Value
Managed Redis®-compatible storage. Ideal for use as a shared cache, message broker, or job queue.
New Key Value Instance →

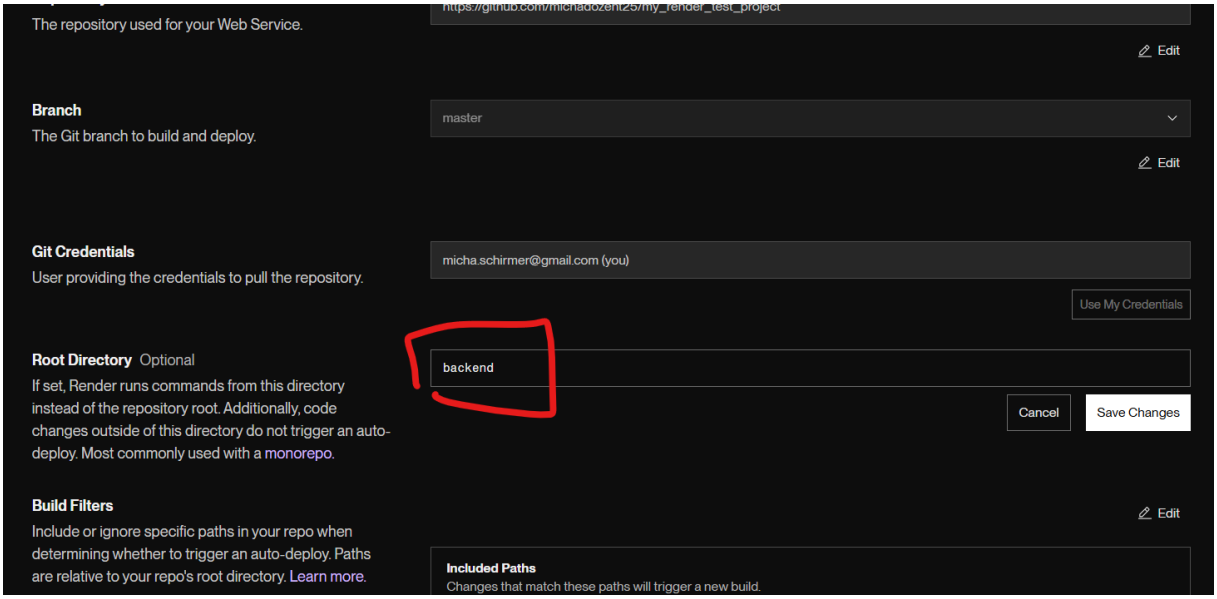

my-dev-space
New Web Service
Search
New
Configure and deploy your new Web Service
Choose service
2 Configure
3 Deploy
Source Code
Git Provider
Public Git Repository
Existing Image
Connect Git provider
Connect your Git provider to deploy from your existing repositories.
GitHub
GitLab
Bitbucket

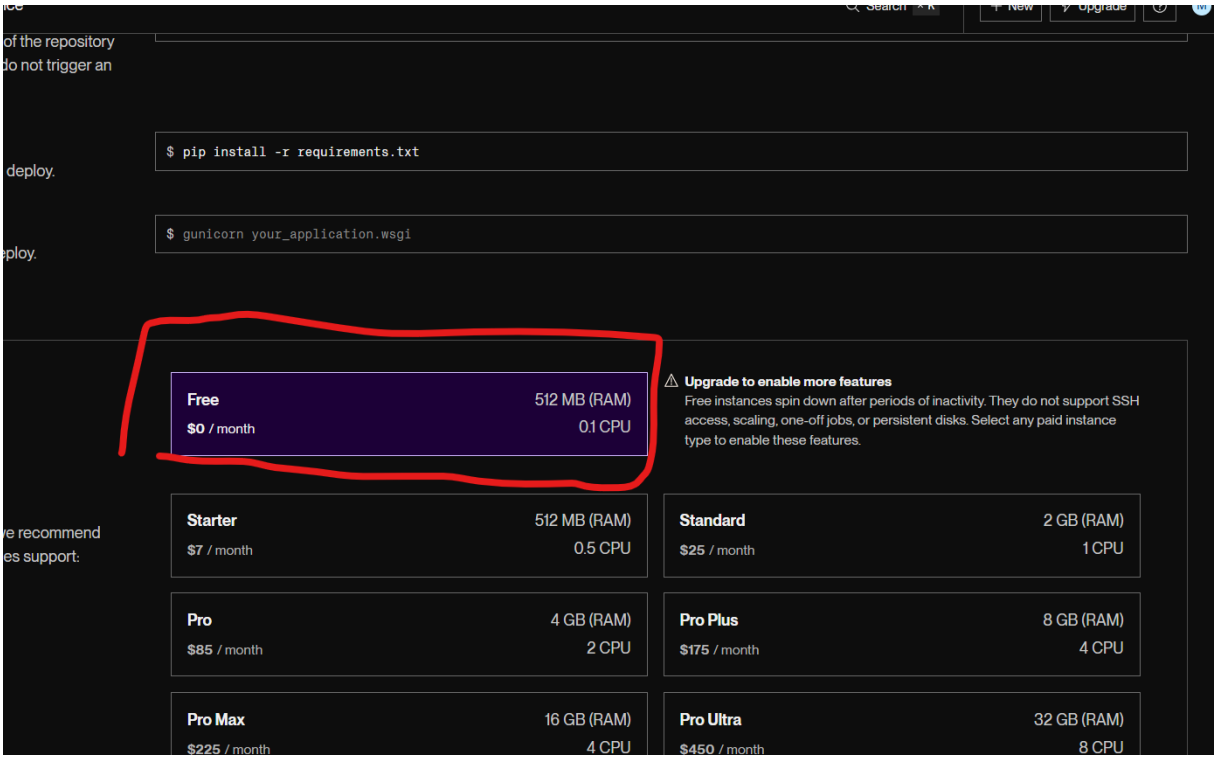


All oder Only




Git Provider
Public Git Repository
Existing Image
Search
michadozent25 / my_render_test_project 5m ago
michadozent25 / python_databases 6d ago
michadozent25 / java_workspace_doz Jun 6
michadozent25 / projekt1 May 28
michadozent25 / projekt2 May 28
michadozent25 / schnittstellen_java May 25


The repository used for your Web Service.
Edit
Branch
The Git branch to build and deploy.
master
Edit
Git Credentials
User providing the credentials to pull the repository.
micha.schirmer@gmail.com (you)
Use My Credentials
Root Directory Optional
If set, Render runs commands from this directory instead of the repository root. Additionally, code changes outside of this directory do not trigger an auto-deploy. Most commonly used with a monorepo.
backend
Cancel
Save Changes
Build Filters
Include or ignore specific paths in your repo when determining whether to trigger an auto-deploy. Paths are relative to your repo's root directory. Learn more.
Edit
Included Paths
Changes that match these paths will trigger a new build.



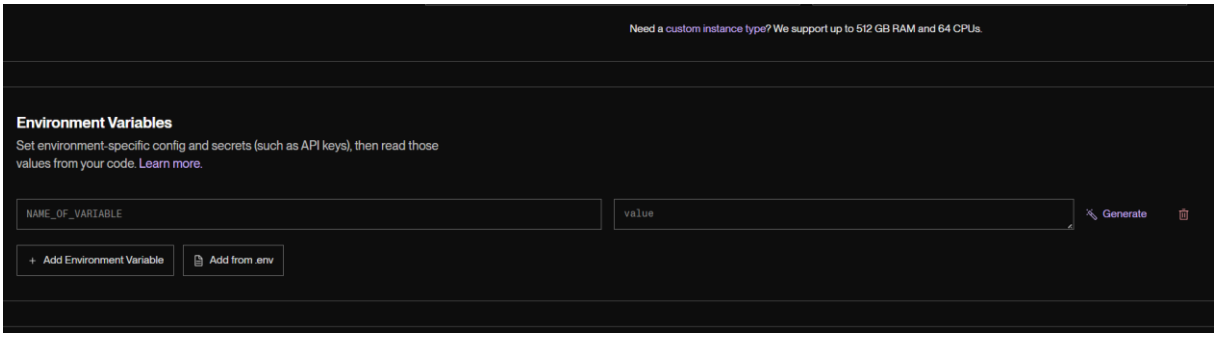
Start command

start.sh ausführbar machen:

git update-index --chmod=+x start.sh

git commit /push

Tab **"Environment" / "Environment Variables"**



✅ **So trägst du das jetzt im Render-Dashboard ein**

1. Du bist bei deinem Service → Tab **"Environment" / "Environment Variables"**
2. **„Add Environment Variable“**

3. Name: DATABASE_URL
   Wert: **deine Neon-DB-Verbindungs-URL**

4. Genauso für FRONTEND_ORIGIN hinzufügen (oder erstmal weglassen → dann erlaubt CORS alles).

💡 **Wichtig:**
– **Keine Hochkommas**, kein psql '...' drum herum – nur den reinen Connection String.
– Groß-/Kleinschreibung muss exakt passen (z. B. DATABASE_URL nicht db_url).

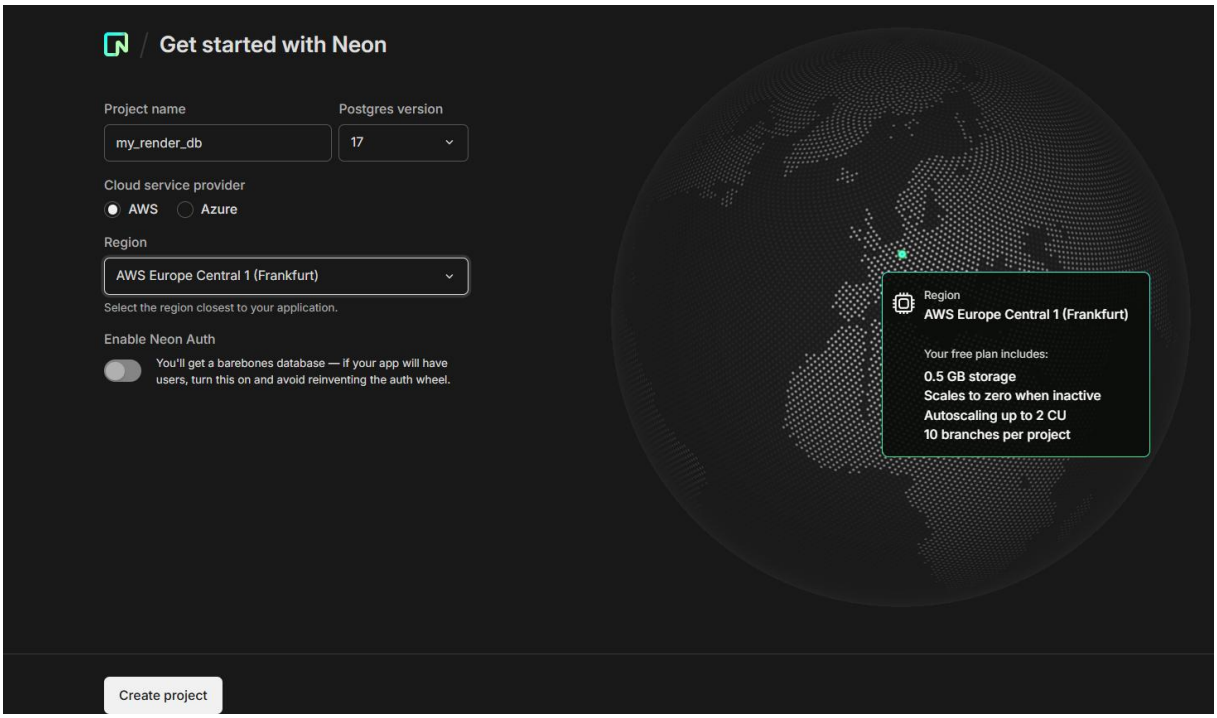
API_BASE https://<dein-backend-service>.onrender.com

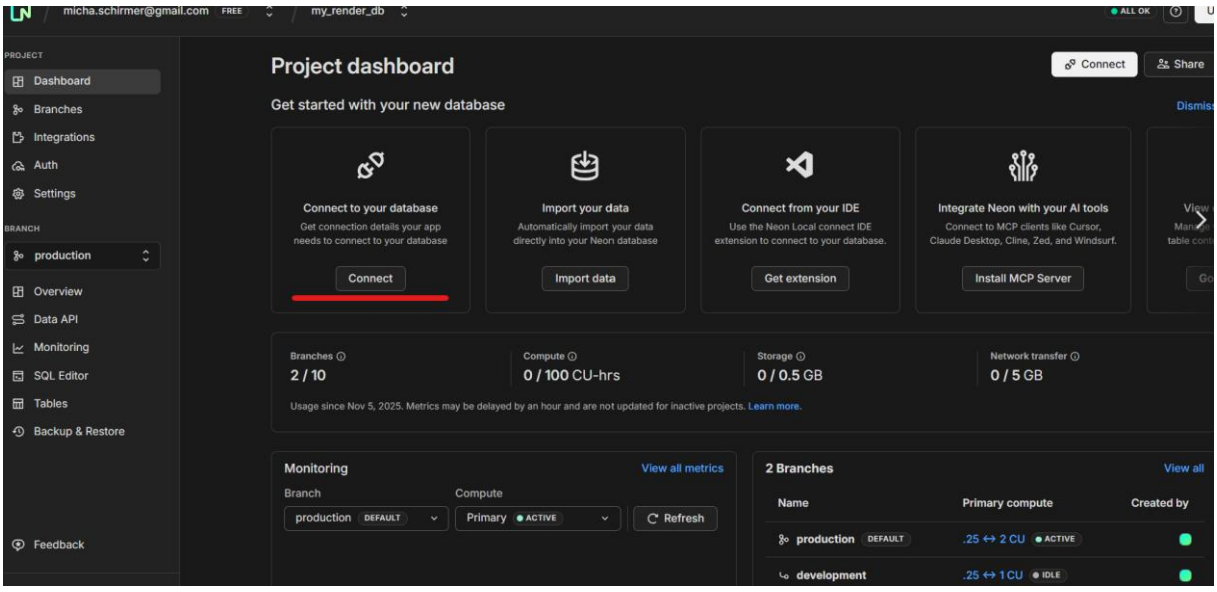
(Diese URL weißt du erst **nach dem ersten Deployment** – du kannst jetzt auch erstmal http://localhost:8000 oder leer lassen, und später ändern.)

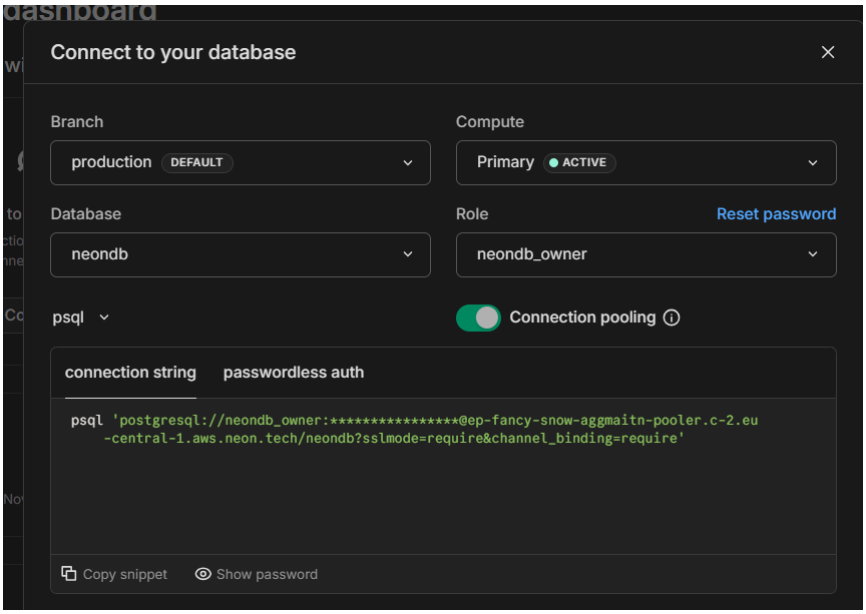
➔ Bei mir ist die korrekte Backend URL:
https://my-render-test-project.onrender.com

2.1 Datenbank: PostgreSQL

Da die Datenbank bei dem kostenlosen render.com Account nur 30 Tage verwendet werden kann, legen wir die Datenbank bei Neon an – ebenfalls eine Free-Version möglich!


Get started with Neon
Project name
my_render_db
Postgres version
17
Cloud service provider
AWS
Azure
Region
AWS Europe Central 1 (Frankfurt)
Select the region closest to your application.
Enable Neon Auth
You'll get a barebones database — if your app will have users, turn this on and avoid reinventing the auth wheel.
Region
AWS Europe Central 1 (Frankfurt)
Your free plan includes:
0.5 GB storage
Scales to zero when inactive
Autoscaling up to 2 CU
10 branches per project
Create project


micha.schirmer@gmail.com
FREE
my_render_db
ALL OK
PROJECT
Dashboard
Branches
Integrations
Auth
Settings
BRANCH
production
Overview
Data API
Monitoring
SQL Editor
Tables
Backup & Restore
Feedback
Project dashboard
Connect
Share
Get started with your new database
Dismiss
Connect to your database
Get connection details your app needs to connect to your database
Connect
Import your data
Automatically import your data directly into your Neon database
Import data
Connect from your IDE
Use the Neon Local connect IDE extension to connect to your database.
Get extension
Integrate Neon with your AI tools
Connect to MCP clients like Cursor, Claude Desktop, Cline, Zed, and Windsurf.
Install MCP Server
Branches
2 / 10
Compute
0 / 100 CU-hrs
Storage
0 / 0.5 GB
Network transfer
0 / 5 GB
Usage since Nov 5, 2025. Metrics may be delayed by an hour and are not updated for inactive projects. Learn more.
Monitoring
View all metrics
Branch
Compute
production DEFAULT
Primary ACTIVE
Refresh
2 Branches
View all
Name
Primary compute
Created by
production DEFAULT
.25 ↔ 2 CU ACTIVE
development
.25 ↔ 1 CU IDLE

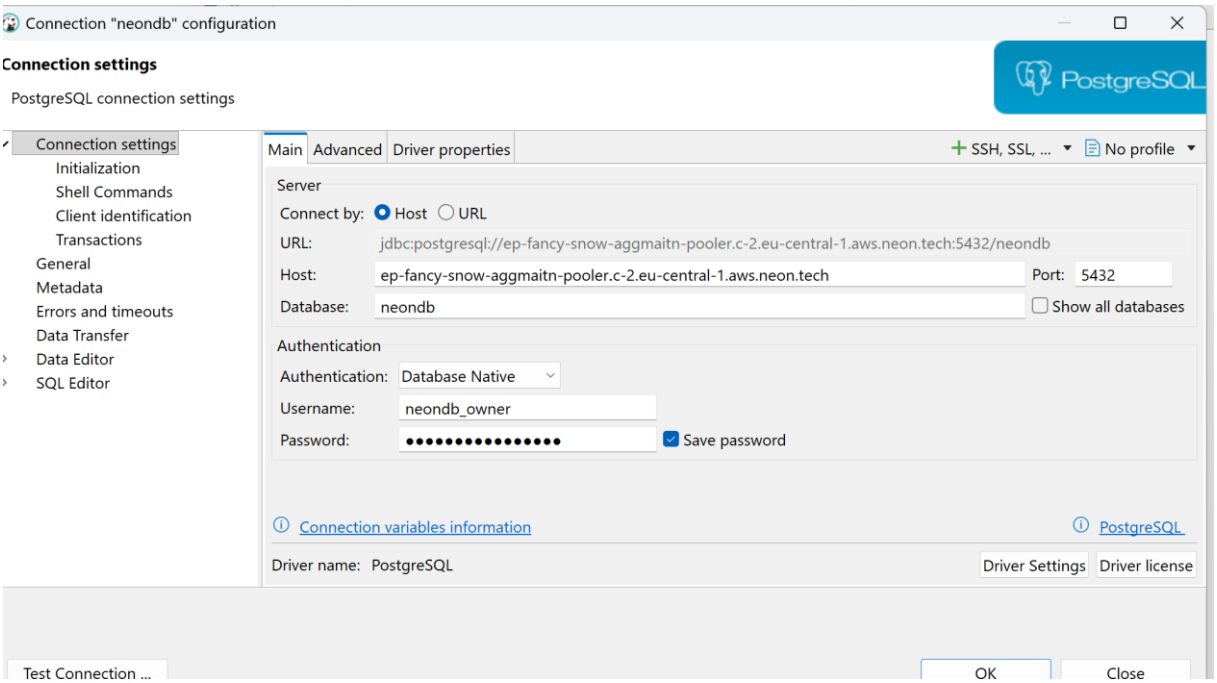


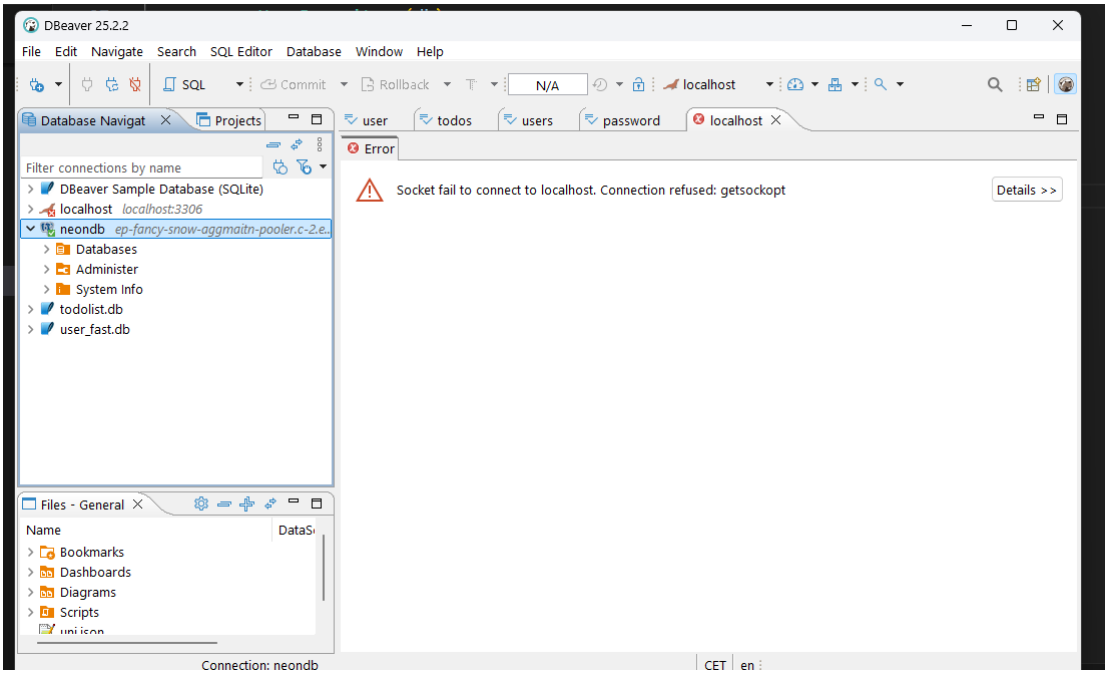
psql 'postgresql://neondb_owner:npg_2jclyw7dAPui@ep-fancy-snow-aggmaitn-pooler.c-2.eu-central-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require'

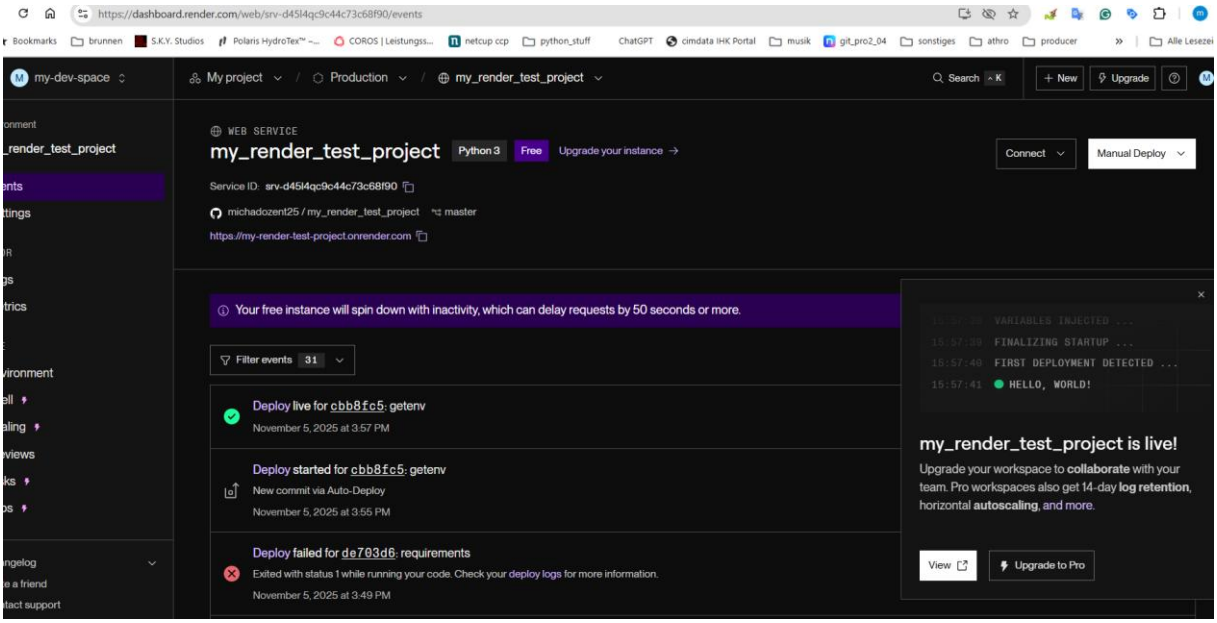
angepasst:

postgresql+psycopg://neondb_owner:npg_2jclyw7dAPui@ep-fancy-snow-aggmaitn-pooler.c-2.eu-central-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require

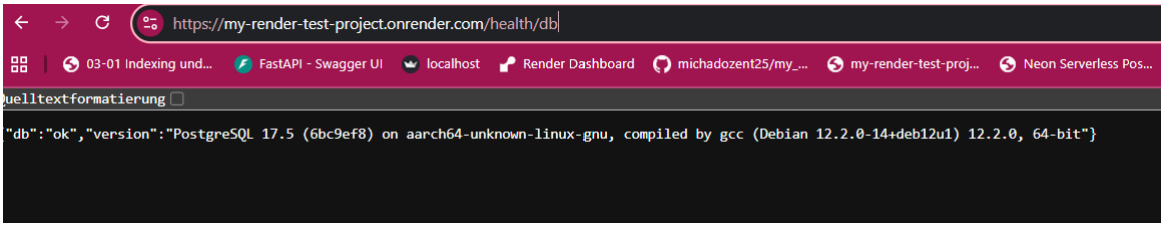
Conection in DBeaver:


Connection "neondb" configuration
Connection settings
PostgreSQL connection settings
PostgreSQL
Connection settings
Initialization
Shell Commands
Client identification
Transactions
General
Metadata
Errors and timeouts
Data Transfer
Data Editor
SQL Editor
Main
Advanced
Driver properties
SSH, SSL, ...
No profile
Server
Connect by: Host URL
URL: jdbc:postgresql://ep-fancy-snow-aggmaitn-pooler.c-2.eu-central-1.aws.neon.tech:5432/neondb
Host: ep-fancy-snow-aggmaitn-pooler.c-2.eu-central-1.aws.neon.tech
Port: 5432
Database: neondb
Show all databases
Authentication
Authentication: Database Native
Username: neondb_owner
Password:
Save password
Connection variables information
PostgreSQL
Driver name: PostgreSQL
Driver Settings
Driver license
Test Connection ...
OK
Close


DBeaver 25.2.2
File Edit Navigate Search SQL Editor Database Window Help
SQL
Commit
Rollback
N/A
localhost
Database Navigat
Projects
Filter connections by name
DBeaver Sample Database (SQLite)
localhost localhost:3306
neondb ep-fancy-snow-aggmaitn-pooler.c-2.e...
Databases
Administer
System Info
todolist.db
user_fast.db
Files - General
Name
Bookmarks
Dashboards
Diagrams
Scripts
user
todos
users
password
localhost
Error
Socket fail to connect to localhost. Connection refused: getsockopt
Details >>
Connection: neondb
CET
en

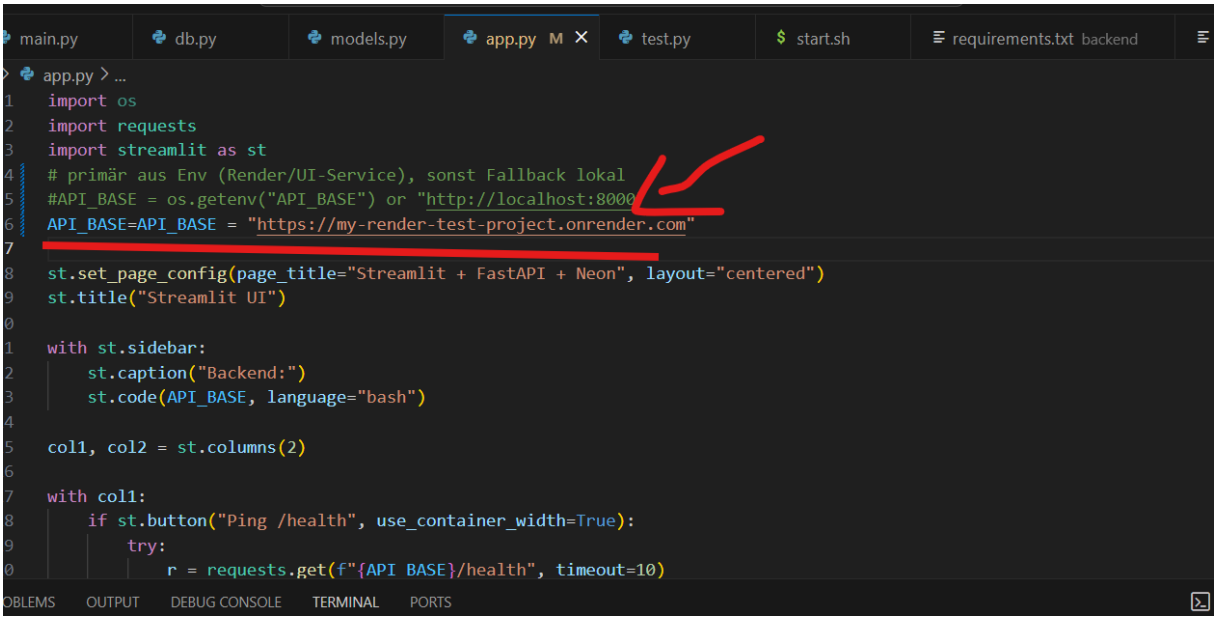


Backend-Test im Browser



2.2 Frontend

Lokales Testen:



Dann:

streamlit run app.py

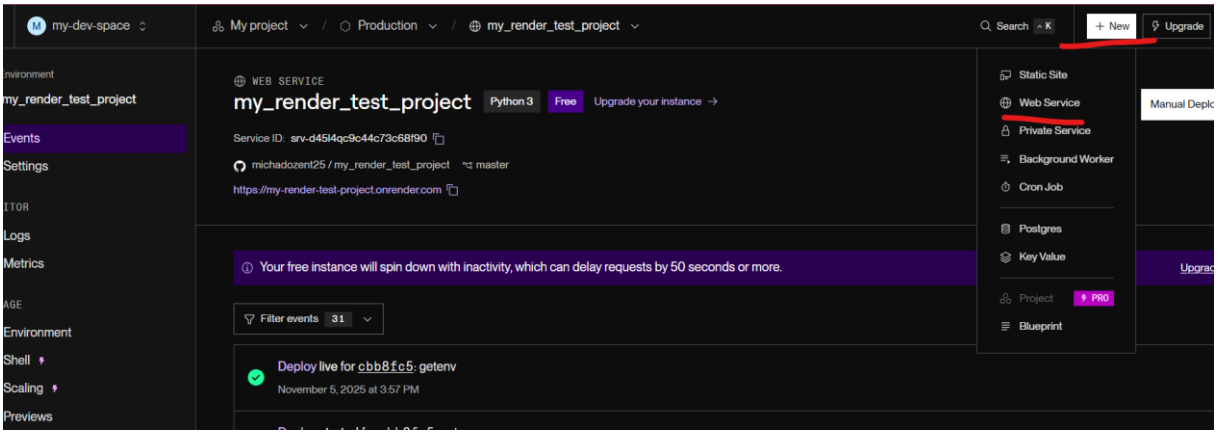
dann vorbereiten für die prod-Variante:

```python
import os
import requests
import streamlit as st
# primär aus Env (Render/UI-Service), sonst Fallback lokal
API_BASE = os.getenv("API_BASE") or "http://localhost:8000"
#API_BASE API_BASE = "https://my-render-test-project.onrender.com"

st.set_page_config(page_title="Streamlit + FastAPI + Neon", layout="centered")
st.title("Streamlit UI")

with st.sidebar:
    st.caption("Backend:")
    st.code(API_BASE, language="bash")

col1, col2 = st.columns(2)

with col1:
    if st.button("Ping /health", use_container_width=True):
```

Commit + push

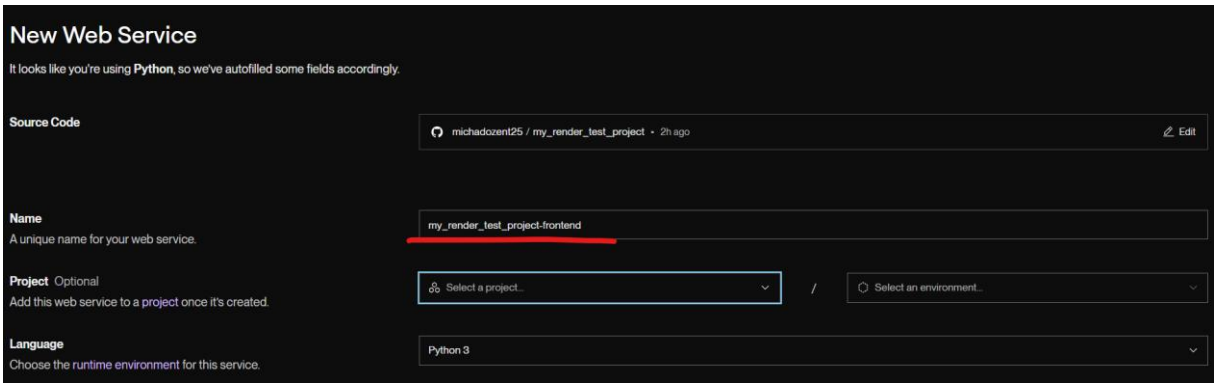
https://dashboard.render.com/

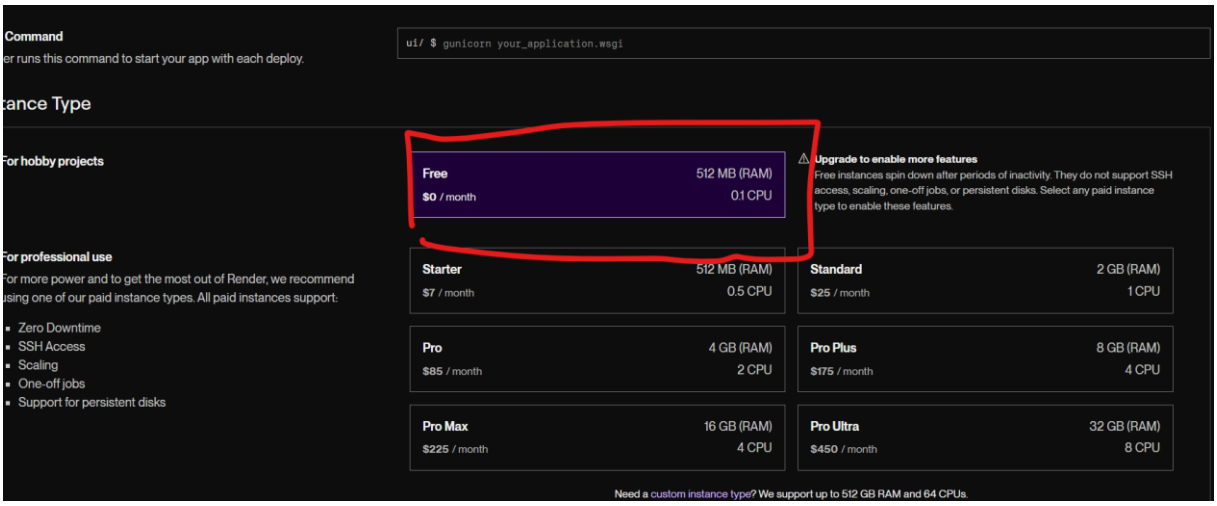
Wie legen den Client ebenfalls als Webservice an.

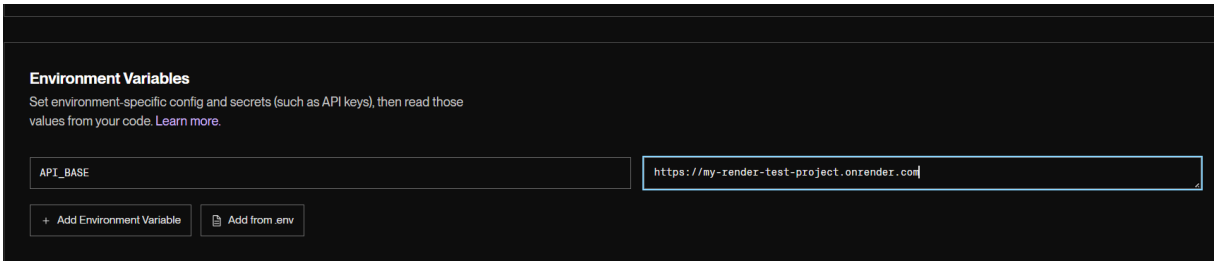


Service-Name

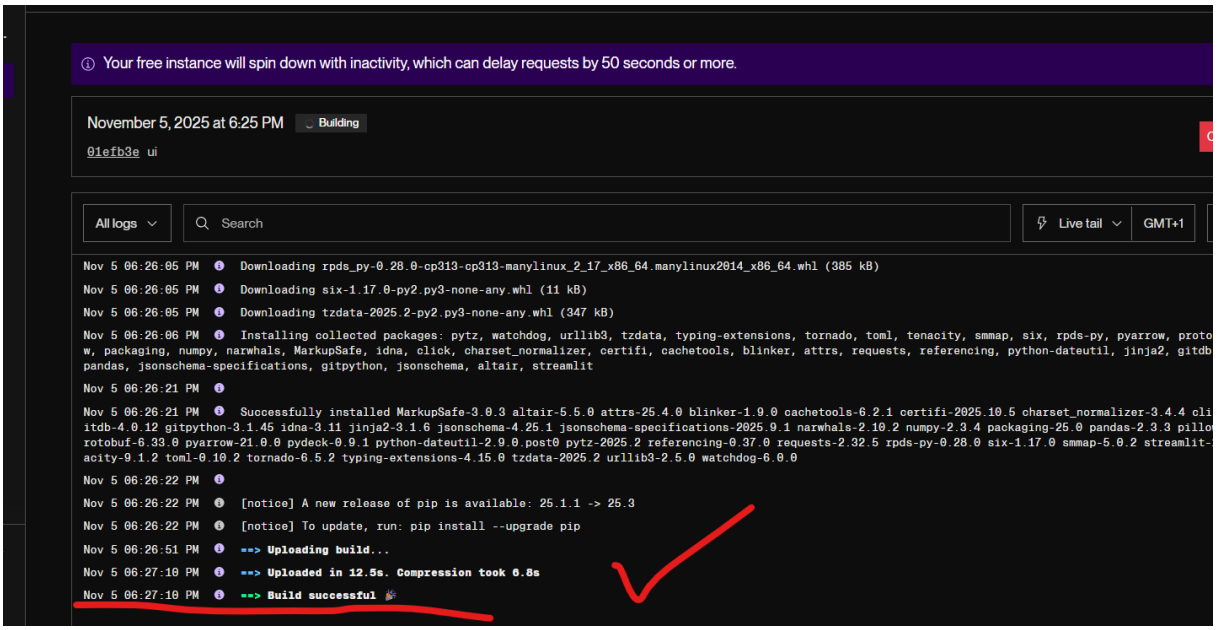
my_render_test_project-frontend



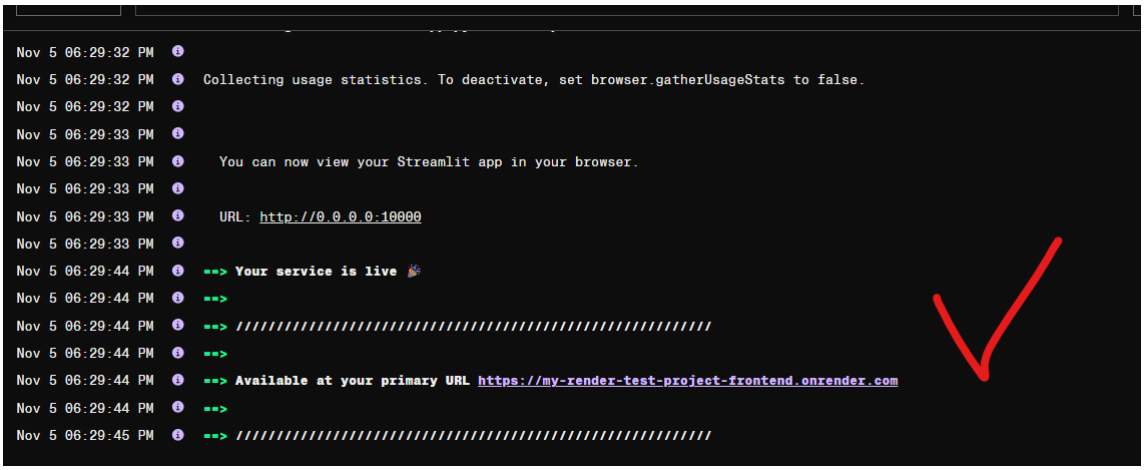




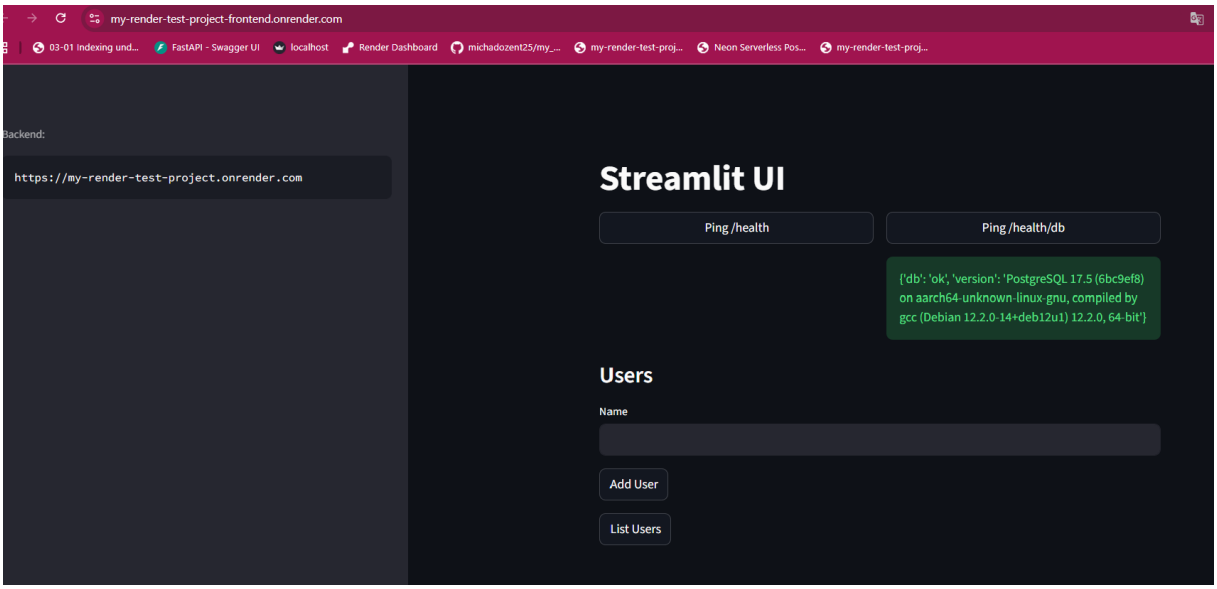
➔ **Dann Deploy-Button**



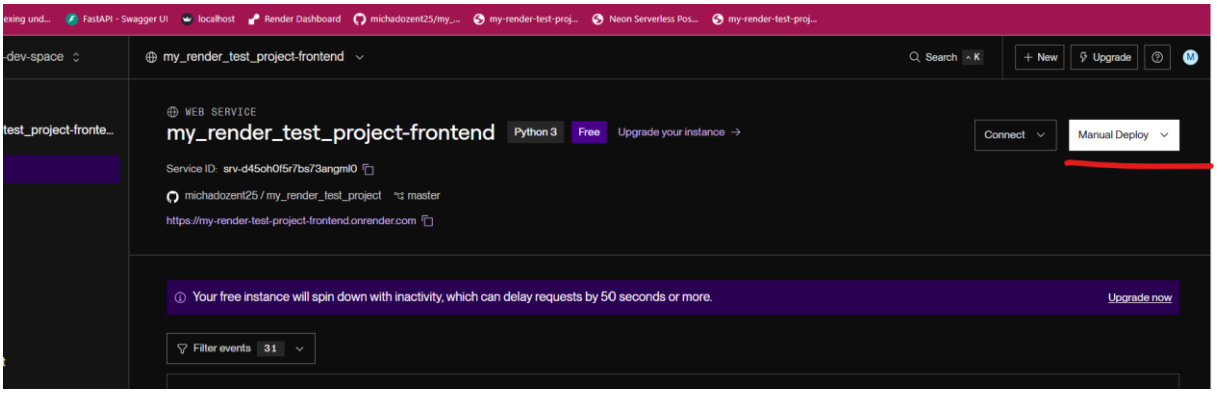
Danach dauert es etwas , deploy...



Client kann nun aufgerufen werden:



Nach Änderungen kann man jederzeit ein neues Deploy anstoßen:



2.3 Links

Frontend:

https://my-render-test-project-frontend.onrender.com/

Backend:

https://my-render-test-project.onrender.com/health

Datenbank (PostgreSQL):

https://neon.com/

Alternativ DB (MySQL)

https://aiven.io/