

Python App Deploy

Tags: Deploy, Build, Git, Server, Cloud

Inhalt

1	Mini-Projekt	1
1.1	Frontend und Backend.....	1
1.2	Verzeichnisstruktur.....	1
1.3	Sourcecode	2
2	Deploy auf Render	2
2.1	Datenbank: PostgreSQL	7
2.2	Frontend	11
2.3	Links.....	15

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

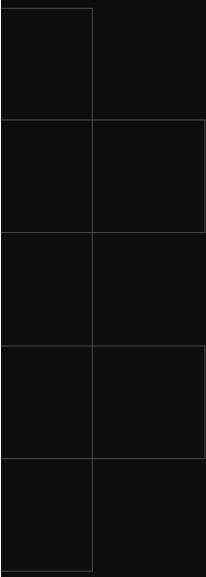
https://github.com/michadozent25/my_render_test_project/tree/master

2 Deploy auf Render

<https://render.com/docs/free>

The screenshot shows the Render documentation website. The top navigation bar includes links for 'Render', 'Docs' (which is active), 'Changelog', and 'Blog'. A search bar and an 'Ask AI' button are also present. The main content area features a large heading 'Deploy for Free' with the subtext 'Preview the Render platform with free web services and datastores'. It explains that you can deploy instances of some Render services free of charge, listing 'Web services (web apps in Node.js, Python, Rails, etc.)', 'Render Postgres databases', and 'Render Key Value instances'. A note at the bottom states: 'Free instances have important limitations, and you **should not** use them for production applications. However, they're perfect for testing out a new technology, working on a hobby project, or previewing Render's developer experience!'

 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

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

Create a new Service

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 →](#)
- KeyValue**
Managed Redis®-compatible storage. Ideal for use as a shared cache, message broker, or job queue.
[New KeyValue Instance →](#)

my-dev-space New Web Service

Configure and deploy your new Web Service

1 Choose service > 2 Configure > 3 Deploy

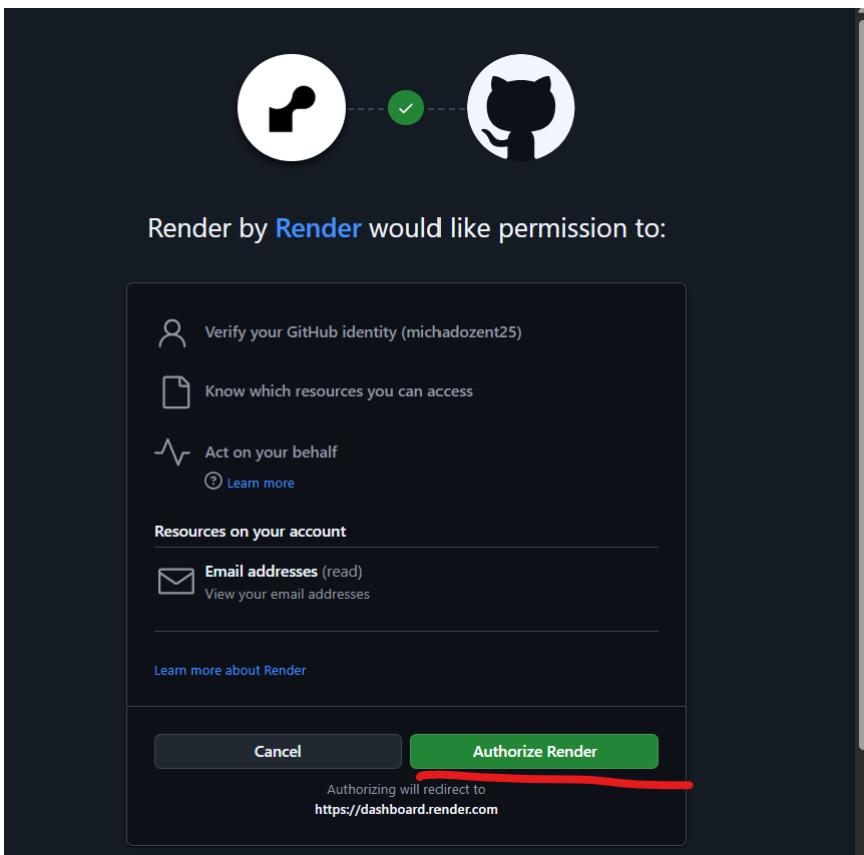
Source Code

Git Provider Public Git Repository Existing Image

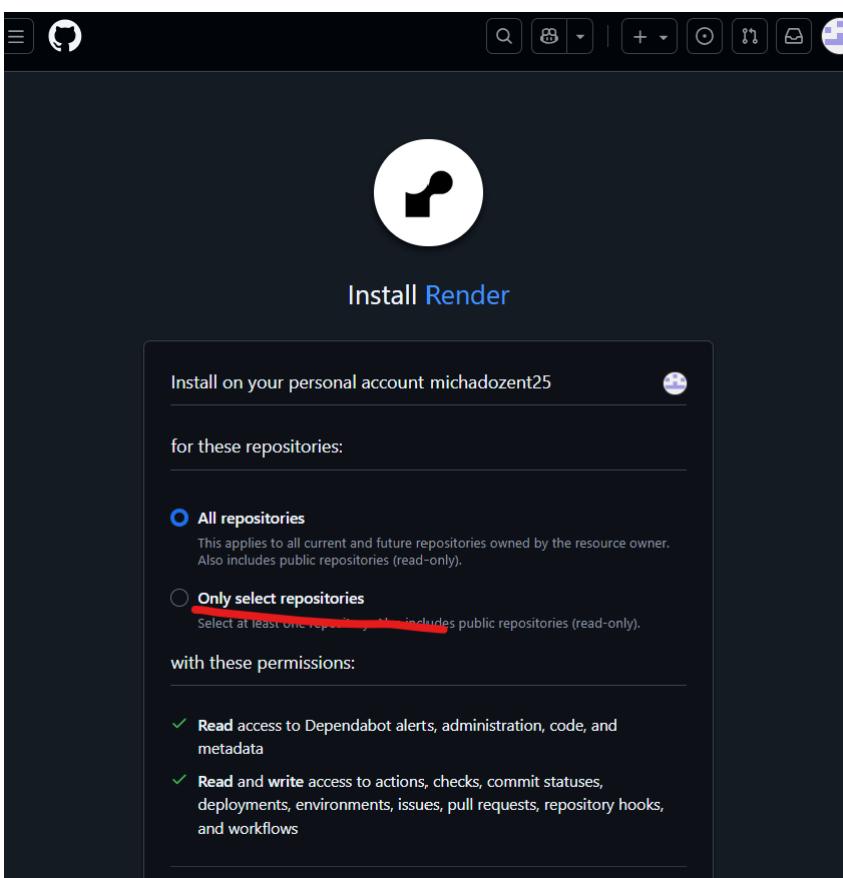
Connect Git provider

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
- michadozent25 / a_python_einsteiger May 23

The repository used for your Web Service. https://github.com/michadozent25/my_render_test_project [Edit](#)

Branch master [Edit](#)

Git Credentials micha.schirmer@gmail.com (you) [Edit](#) [Use My Credentials](#)

Root Directory backend [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. [Cancel](#) [Save Changes](#)

Build Filters [Edit](#) 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.](#)

Included Paths Changes that match these paths will trigger a new build.

The screenshot shows the Render dashboard interface. At the top, there are deployment logs:

```
$ pip install -r requirements.txt
```

```
$ gunicorn your_application.wsgi
```

Below the logs, there's a section about recommended instance types:

We recommend these support:

A red box highlights the "Free" instance plan. The "Free" plan is described as follows:

Free	\$0 / month	512 MB (RAM)	0.1 CPU
-------------	--------------------	---------------------	----------------

To the right of the "Free" plan, there's a note: "Upgrade to enable more features" which states: "Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features."

Below the "Free" plan, there are other instance options:

Starter	\$7 / month	512 MB (RAM)	0.5 CPU
Standard	\$25 / month	2 GB (RAM)	1 CPU
Pro	\$85 / month	4 GB (RAM)	2 CPU
Pro Plus	\$175 / month	8 GB (RAM)	4 CPU
Pro Max	\$225 / month	16 GB (RAM)	4 CPU
Pro Ultra	\$450 / month	32 GB (RAM)	8 CPU

Start command

start.sh ausführbar machen:

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

```
git commit /push
```

Tab "Environment" / "Environment Variables"

The screenshot shows the "Environment Variables" tab in the Render dashboard. At the top, it says "Need a [custom instance type](#)? We support up to 512 GB RAM and 64 CPUs."

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. [Learn more](#).

NAME_OF_VARIABLE	value	Generate	Remove
NAME_OF_VARIABLE	value	Generate	Remove
+ Add Environment Variable	Add from env		

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).

The screenshot shows the 'Environment Variables' section of a Render.com project. It lists three variables: DATABASE_URL, FRONTEND_ORIGIN, and API_BASE. Each variable has a value field containing three dots ('....') and a delete icon. At the bottom left are buttons for '+ Add Environment Variable' and 'Add from env'.

Variable	Value	Action
DATABASE_URL	Delete
FRONTEND_ORIGIN	Delete
API_BASE	Delete

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

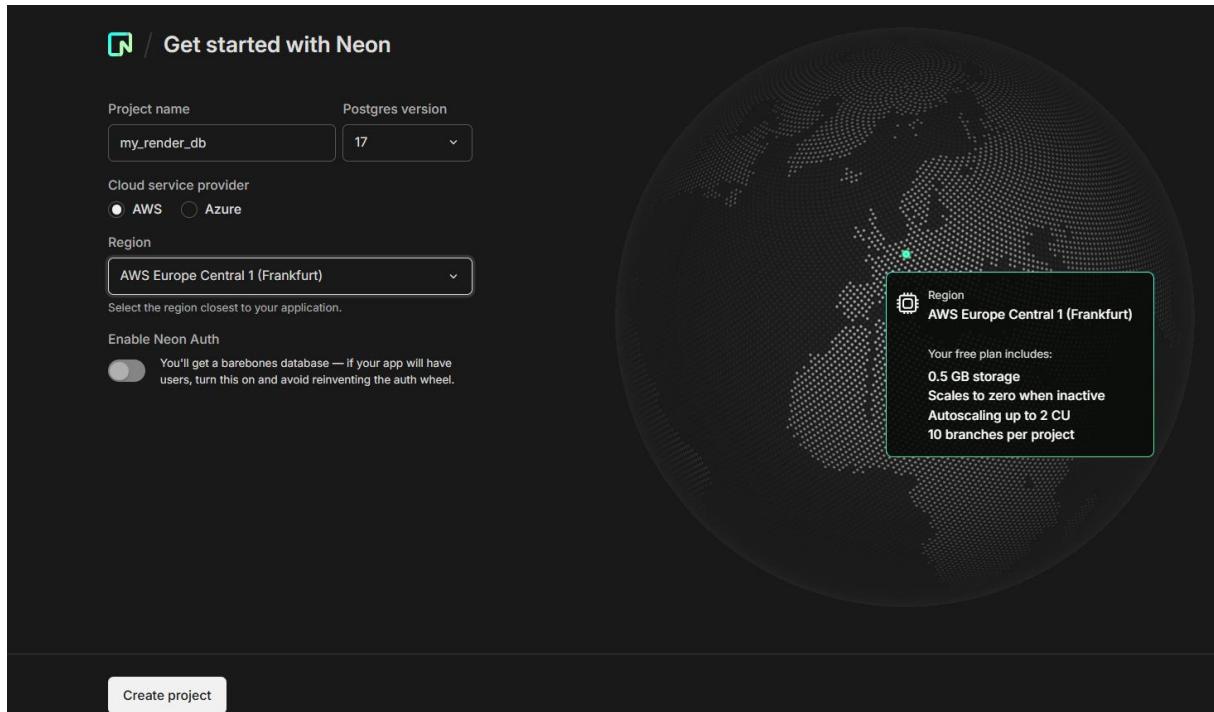
(Diese URL weißt du erst **nach dem ersten Deployment** – du kannst jetzt auch erstmal <http://localhost:8000> oder leer lassen, und später ändern.)

➔ Bei mir ist die korrekte Backend URL:

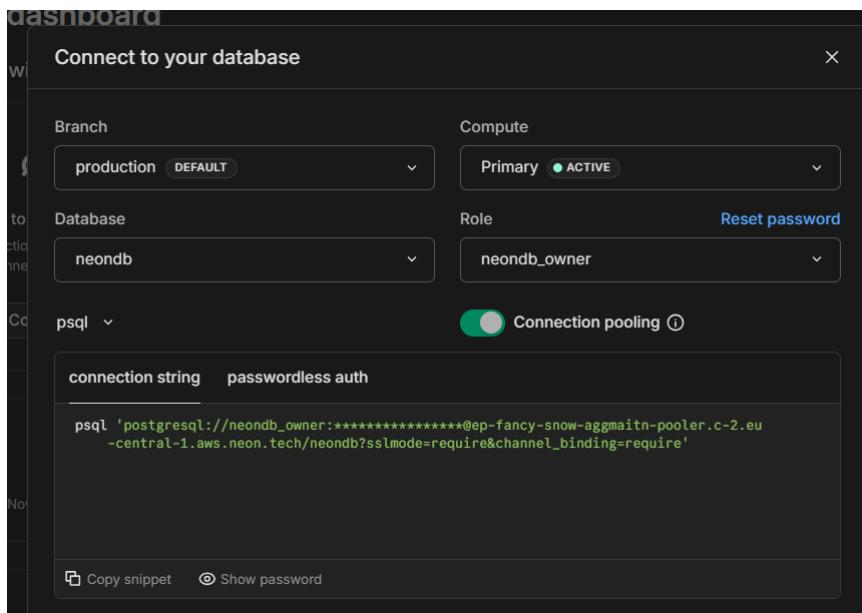
<https://my-render-test-project.onrender.com>

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!



The screenshot shows the Neon Project dashboard for the 'my_render_db' project. The left sidebar includes links for 'Dashboard', 'Branches', 'Integrations', 'Auth', 'Settings', 'Overview', 'Data API', 'Monitoring', 'SQL Editor', 'Tables', 'Backup & Restore', and 'Feedback'. The main dashboard area has a title 'Project dashboard' and a section 'Get started with your new database' with four cards: 'Connect to your database' (with a 'Connect' button), 'Import your data' (with a 'Import data' button), 'Connect from your IDE' (with a 'Get extension' button), and 'Integrate Neon with your AI tools' (with a 'Install MCP Server' button). Below this are summary metrics: 'Branches 2 / 10', 'Compute 0 / 100 CU-hrs', 'Storage 0 / 0.5 GB', and 'Network transfer 0 / 5 GB'. A note at the bottom states: 'Usage since Nov 5, 2025. Metrics may be delayed by an hour and are not updated for inactive projects. [Learn more](#)'. At the bottom of the dashboard are sections for 'Monitoring' (with dropdowns for 'Branch' set to 'production' and 'Compute' set to 'Primary ACTIVE') and '2 Branches' (listing 'production' and 'development' with their respective details).

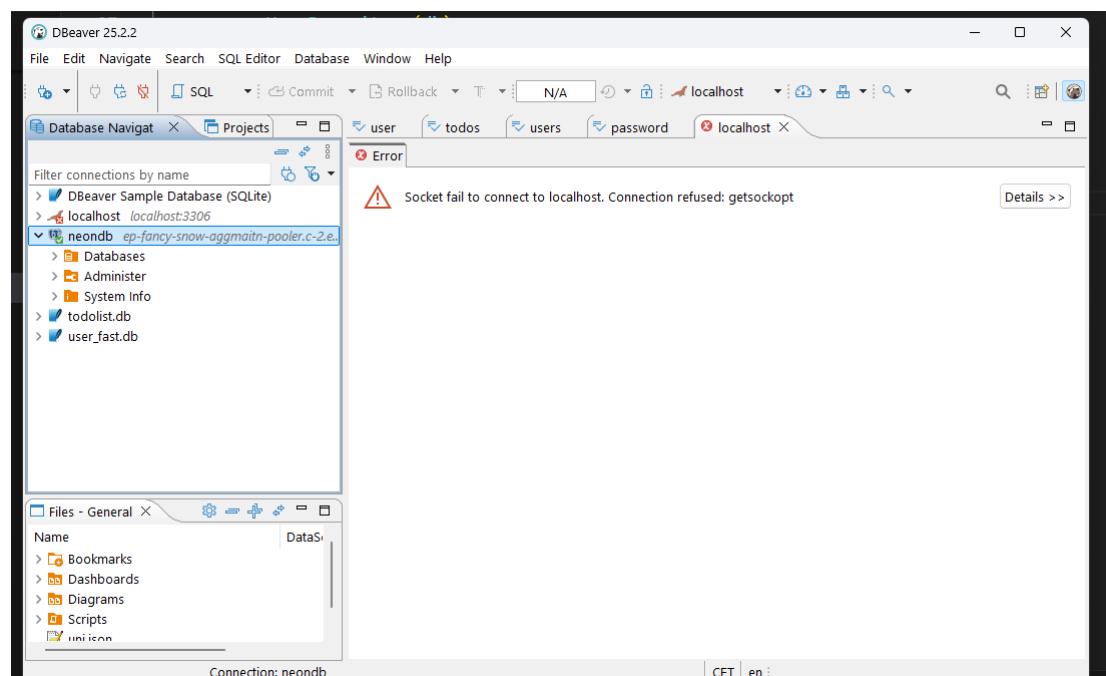
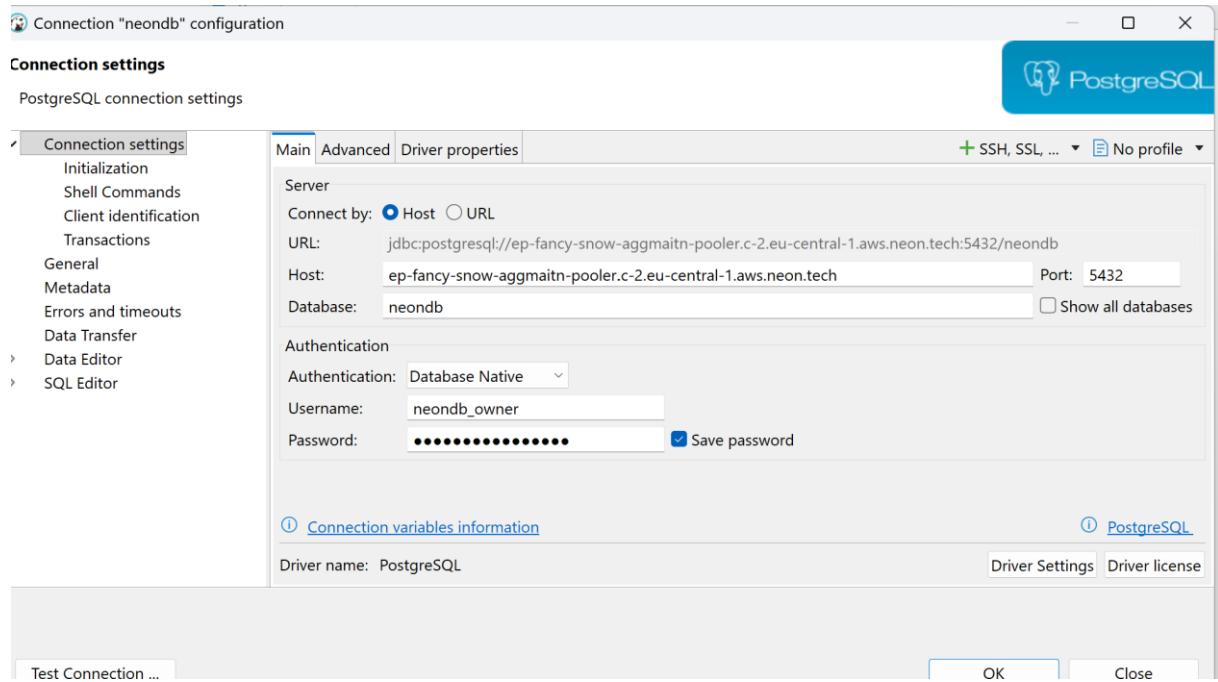


```
psql 'postgresql://neondb_owner:npg_2jclwy7dAPui@ep-fancy-snow-aggmaitn-pooler.c-2.eu-central-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require'
```

angepasst:

```
postgresql+psycopg://neondb_owner:npg_2jclwy7dAPui@ep-fancy-snow-aggmaitn-pooler.c-2.eu-central-1.aws.neon.tech/neondb?sslmode=require&channel_binding=require
```

Conection in DBeaver:



The screenshot shows the Render Dashboard for the project 'my_render_test_project'. The sidebar on the left contains links for 'comment', 'render test project', 'events', 'settings', 'API', 'logs', 'metrics', 'environment', 'scaling', 'reviews', 'tasks', 'logs', and 'support'. The main content area displays deployment logs with the following entries:

- 15:57:38 VARIABLES INJECTED ...
- 15:57:39 FINALIZING STARTUP ...
- 15:57:40 FIRST DEPLOYMENT DETECTED ...
- 15:57:41 HELLO, WORLD!

A message box on the right says 'my_render_test_project is live!'. Below it, a note encourages upgrading to Pro workspace for collaboration and log retention.

Backend-Test im Browser

The screenshot shows a browser window with the URL <https://my-render-test-project.onrender.com/health/db>. The page content is a JSON response from the database health check:

```
{"db": "ok", "version": "PostgreSQL 17.5 (6bc9ef8) on aarch64-unknown-linux-gnu, compiled by gcc (Debian 12.2.0-14+deb12u1) 12.2.0, 64-bit"}
```

2.2 Frontend

Lokales Testen:

```
main.py db.py models.py app.py M X test.py start.sh requirements.txt backend
> app.py > ...
1 import os
2 import requests
3 import streamlit as st
4 # primär aus Env (Render/UI-Service), sonst Fallback lokal
5 #API_BASE = os.getenv("API_BASE") or "http://localhost:8000"
6 API_BASE=API_BASE = "https://my-render-test-project.onrender.com"
7
8 st.set_page_config(page_title="Streamlit + FastAPI + Neon", layout="centered")
9 st.title("Streamlit UI")
10
11 with st.sidebar:
12     st.caption("Backend:")
13     st.code(API_BASE, language="bash")
14
15 col1, col2 = st.columns(2)
16
17 with col1:
18     if st.button("Ping /health", use_container_width=True):
19         try:
20             r = requests.get(f"{API_BASE}/health", timeout=10)
```

Dann:

```
streamlit run app.py
```

dann vorbereiten für die prod-Variante:

```
app.py > ...
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.

The screenshot shows the Render dashboard interface. On the left, there's a sidebar with project navigation: 'my-dev-space' (selected), 'My project' (dropdown), 'Production' (dropdown), and 'my_render_test_project' (selected). The main area displays a 'WEB SERVICE' named 'my_render_test_project' running on Python 3. It shows a Service ID: 'srv-d4514qc9c44c73c68f90' and a link to the service: 'https://my-render-test-project.onrender.com'. A purple banner at the bottom states: 'Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.' On the right, there's a sidebar with various service options: 'Static Site', 'Web Service' (selected), 'Private Service', 'Background Worker', 'Cron Job', 'Postgres', 'Key Value', 'Project' (PRO), and 'Blueprint'. A red box highlights the '+ New' button in the top right corner of the main dashboard area.

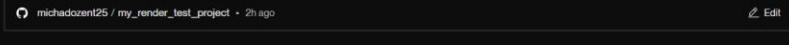
Service-Name

my_render_test_project-frontend

New Web Service

It looks like you're using **Python**, so we've autofilled some fields accordingly.

Source Code



Name
A unique name for your web service.
`my_render_test_project-frontend`

Project Optional
Add this web service to a project once it's created.


Language
Choose the **runtime environment** for this service.
`Python 3`

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.
`ui`

Build Command
Render runs this command to build your app before each deploy.
`ui/ $ pip install -r requirements.txt`

Start Command
Render runs this command to start your app with each deploy.
`ui/ $ streamlit run app.py --server.port $PORT --server.address 0.0.0.0`

Command
Render runs this command to start your app with each deploy.
`ui/ $ gunicorn your_application.wsgi`

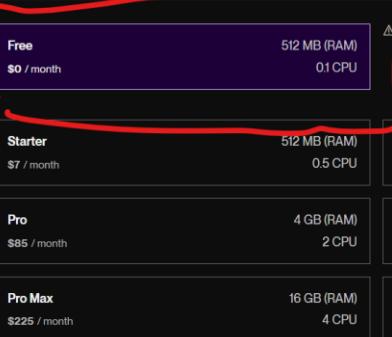
Instance Type

For hobby projects

Free \$0 / month	512 MB (RAM)	0.1 CPU	 Upgrade to enable more features Free instances spin down after periods of inactivity. They do not support SSH access, scaling, one-off jobs, or persistent disks. Select any paid instance type to enable these features.
Starter \$7 / month	512 MB (RAM)	0.5 CPU	Standard \$25 / month 2 GB (RAM) 1 CPU
Pro \$85 / month	4 GB (RAM)	2 CPU	Pro Plus \$175 / month 8 GB (RAM) 4 CPU
Pro Max \$225 / month	16 GB (RAM)	4 CPU	Pro Ultra \$450 / month 32 GB (RAM) 8 CPU

For professional use
For more power and get the most out of Render, we recommend using one of our paid instance types. All paid instances support:

- Zero Downtime
- SSH Access
- Scaling
- One-off jobs
- Support for persistent disks



Need a custom instance type? We support up to 512 GB RAM and 64 CPUs.

Environment Variables

Set environment-specific config and secrets (such as API keys), then read those values from your code. Learn more.

API_BASE	https://my-render-test-project.onrender.com
+ Add Environment Variable	

→ Dann Deploy-Button

The screenshot shows a Streamlit build log from November 5, 2025, at 6:25 PM. The status bar indicates "Building". The log output is as follows:

```
Nov 5 06:26:05 PM ① Downloading rpds_py-0.28.0-cp313-cp313-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (385 kB)
Nov 5 06:26:05 PM ① Downloading six-1.17.0-py2.py3-none-any.whl (11 kB)
Nov 5 06:26:05 PM ① Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
Nov 5 06:26:06 PM ① Installing collected packages: pytz, watchdog, urllib3, tzdata, typing-extensions, tornado, toml, tenacity, smmap, six, rpds-py, pyarrow, proto
w, packaging, numpy, narwhals, MarkupSafe, idna, click, charset_normalizer, certifi, cachetools, blinker, attrs, requests, referencing, python-dateutil, jinja2, gitdb
pandas, jsonschema-specifications, gitpython, jsonschema, altair, streamlit
Nov 5 06:26:21 PM ①
Nov 5 06:26:21 PM ① Successfully installed MarkupSafe-3.0.3 altair-5.5.0 attrs-25.4.0 blinker-1.9.0 cachetools-6.2.1 certifi-2025.10.5 charset_normalizer-3.4.4 cli
itdb-4.0.12 gitpython-3.1.45 idna-3.11 jinja2-3.1.6 jsonschema-4.25.1 jsonschema-specifications-2025.9.1 narwhals-2.10.2 numpy-2.3.4 packaging-25.0 pandas-2.3.3 pillow
rotobuf-6.33.0 pyarrow-21.0.0 pydeck-0.9.1 python-dateutil-2.9.0.post0 pytz-2025.2 referencing-0.37.0 requests-2.32.5 rpds-py-0.28.0 six-1.17.0 smmap-5.0.2 streamlit-
acity-9.1.2 toml-0.10.2 tornado-6.5.2 typing-extensions-4.15.0 tzdata-2025.2 urllib3-2.5.0 watchdog-6.0.0
Nov 5 06:26:22 PM ①
Nov 5 06:26:22 PM ① [notice] A new release of pip is available: 25.1.1 -> 25.3
Nov 5 06:26:22 PM ① [notice] To update, run: pip install --upgrade pip
Nov 5 06:26:51 PM ① ➞ Uploading build...
Nov 5 06:27:10 PM ① ➞ Uploaded in 12.6s. Compression took 6.8s
Nov 5 06:27:10 PM ① ➞ Build successful
```

A red arrow points to the "Build successful" message at the bottom of the log.

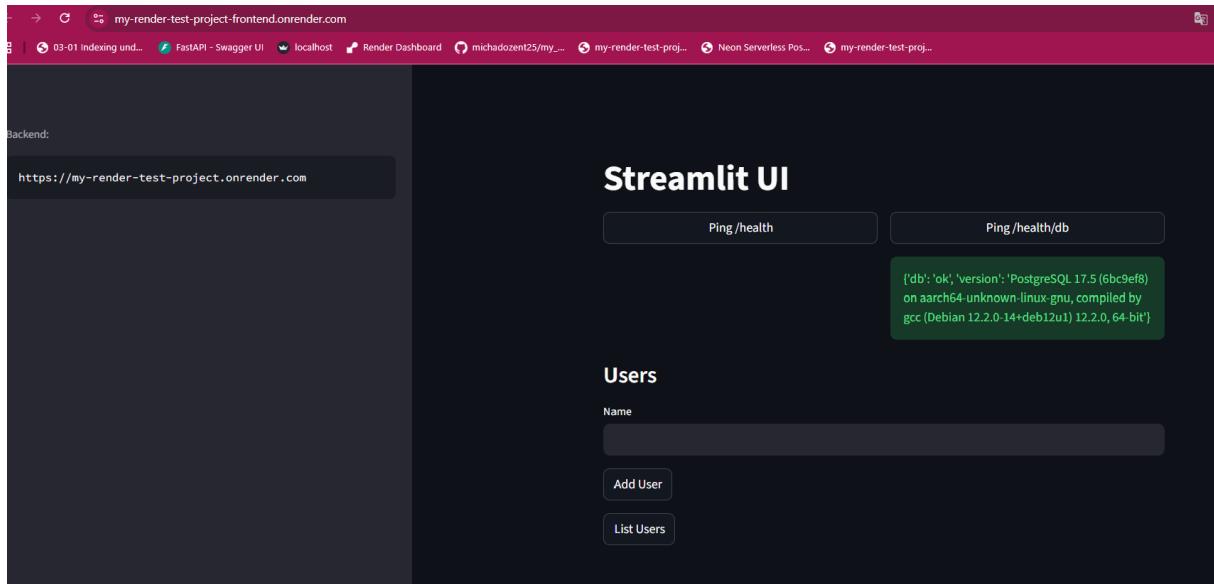
Danach dauert es etwas , deploy...

The screenshot shows a Streamlit deployment log from November 5, 2025, at 6:29:32 PM. The log output is as follows:

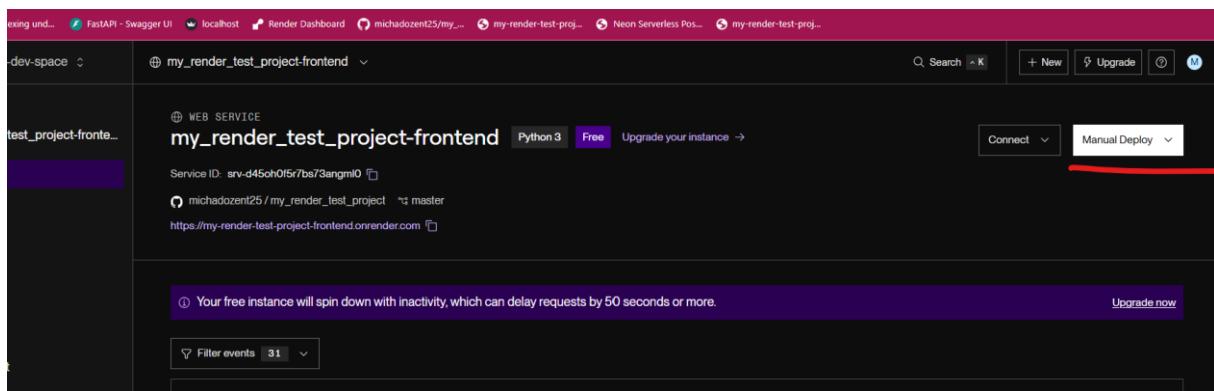
```
Nov 5 06:29:32 PM ①
Nov 5 06:29:32 PM ① Collecting usage statistics. To deactivate, set browser.gatherUsageStats to false.
Nov 5 06:29:32 PM ①
Nov 5 06:29:33 PM ①
Nov 5 06:29:33 PM ① You can now view your Streamlit app in your browser.
Nov 5 06:29:33 PM ①
Nov 5 06:29:33 PM ① URL: http://0.0.0.0:10000
Nov 5 06:29:33 PM ①
Nov 5 06:29:44 PM ① ➞ Your service is live 🎉
Nov 5 06:29:44 PM ① ➞
Nov 5 06:29:44 PM ① ➞ ///////////////////////////////////////////////////
Nov 5 06:29:44 PM ① ➞
Nov 5 06:29:44 PM ① ➞ Available at your primary URL https://my-render-test-project-frontend.onrender.com
Nov 5 06:29:44 PM ① ➞
Nov 5 06:29:45 PM ① ➞ ///////////////////////////////////////////////////
```

A red checkmark is placed next to the "Your service is live 🎉" message.

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/>