

Deployment: Render App & Neon DB

Copyright © 2026 by
Robert M. Dondero, Ph.D
Princeton University

Objectives

- The lecture will cover:
 - How to deploy a database to Neon
 - Database connection pooling
 - How to deploy a Web app to Render

Agenda

- **Neon**
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Render
- Defining the app
- Deploying the app (demo)
- Using the app (demo)

Neon



Nikita Shamgunov (CEO)

Stas Kelvich

Heikki Linnakangas

Agenda

- Neon
- **Creating the DB (demo)**
- Using the DB (demo)
- Using the DB: Python
- Render
- Defining the app
- Deploying the app (demo)
- Using the app (demo)

Creating the DB

- Create a Neon account
 - Browse to <https://neon.com>
 - Click *Sign Up*
 - Follow the instructions
 - Provide an email address
 - Provide a password
 - Need not provide a credit card number

Creating the DB

- Log into Neon
 - Browse to <https://neon.com>
 - Click on *Log In*
 - For *Email address* enter *youremailaddress*
 - For *Password* enter *yourpassword*

Creating the DB

- From your Neon Console
 - Click *New Project*
 - For *Project name* enter `pennyproj`
 - For *Region* select `AWS US East 2 (Ohio)`
 - For *Services* select `Postgres database`
 - Click *Create Project*
- In the resulting dialog
 - Click *Go to project*

Creating the DB

- In the resulting Branch overview page
 - Click *Connect*
 - In the dialog
 - Turn off *Connection pooling*
 - Note the *Connection string*
 - Save it somewhere
 - Let's call it ***YourNeonDbUrl***
 - Dismiss the dialog

Agenda

- Neon
- Creating the DB (demo)
- **Using the DB (demo)**
- Using the DB: Python
- Render
- Defining the app
- Deploying the app (demo)
- Using the app (demo)

Using the DB

- From a command-line
 - Install `psql`
 - Without installing PostgreSQL server
 - See <https://www.risingwave.dev/docs/current/install-psql-without-postgresql/>

Using the DB

- From a command-line (cont.)
 - Install `psql` (Mac)
 - First install homebrew; then:

```
$ brew update
$ brew install libpq
$ brew link --force libpq
$
```

Using the DB

- From a command-line (cont.)
 - Install `psql` (MS Windows)
 - Download the installer from <https://www.postgresql.org/download/windows/>
 - Run the installer
 - Select *Command Line Tools* and uncheck other options during installation

Using the DB

- From a command-line (cont.)
 - Install `psql` (Linux)

```
$ sudo apt update
$ sudo apt install postgresql-client
$
```

Using the DB

Some `psql` statements

sqlite3 Statement	psql Statement
<code>.help</code>	<code>\h</code>
<code>.quit</code>	<code>\q</code>
<code>.tables</code>	<code>\d</code>
<code>.schema <i>table</i></code>	<code>\d <i>table</i></code>
<code>.read <i>file</i></code>	<code>\i <i>file</i></code>

Using the DB

```
$ cat penny.sql
DROP TABLE IF EXISTS books;

CREATE TABLE books (isbn TEXT PRIMARY KEY, author TEXT, title TEXT);

INSERT INTO books (isbn, author, title)
    VALUES ('123', 'Kernighan', 'The Practice of Programming');
INSERT INTO books (isbn, author, title)
    VALUES ('234', 'Kernighan', 'The C Programming Language');
INSERT INTO books (isbn, author, title)
    VALUES ('345', 'Sedgewick', 'Algorithms in C');
$
```

Using the DB

Don't forget
the quotes

```
$ psql "YourNeonDbUrl"
yourdbname=> \i penny.sql
DROP TABLE
CREATE TABLE
INSERT 0 1
INSERT 0 1
INSERT 0 1
yourdbname=> SELECT * FROM books;
 isbn | author | title
-----+-----+-----
 123  | Kernighan | The Practice of Programming
 234  | Kernighan | The C Programming Language
 345  | Sedgewick | Algorithms in C
(3 rows)

yourdbname=> \q
$
```

Aside: Graphical Clients

- Graphical PostgreSQL clients:
 - pgAdmin4
 - DBeaver
 - Postico (Mac)
 - ...
- List of graphical clients:
https://wiki.postgresql.org/wiki/PostgreSQL_Clients

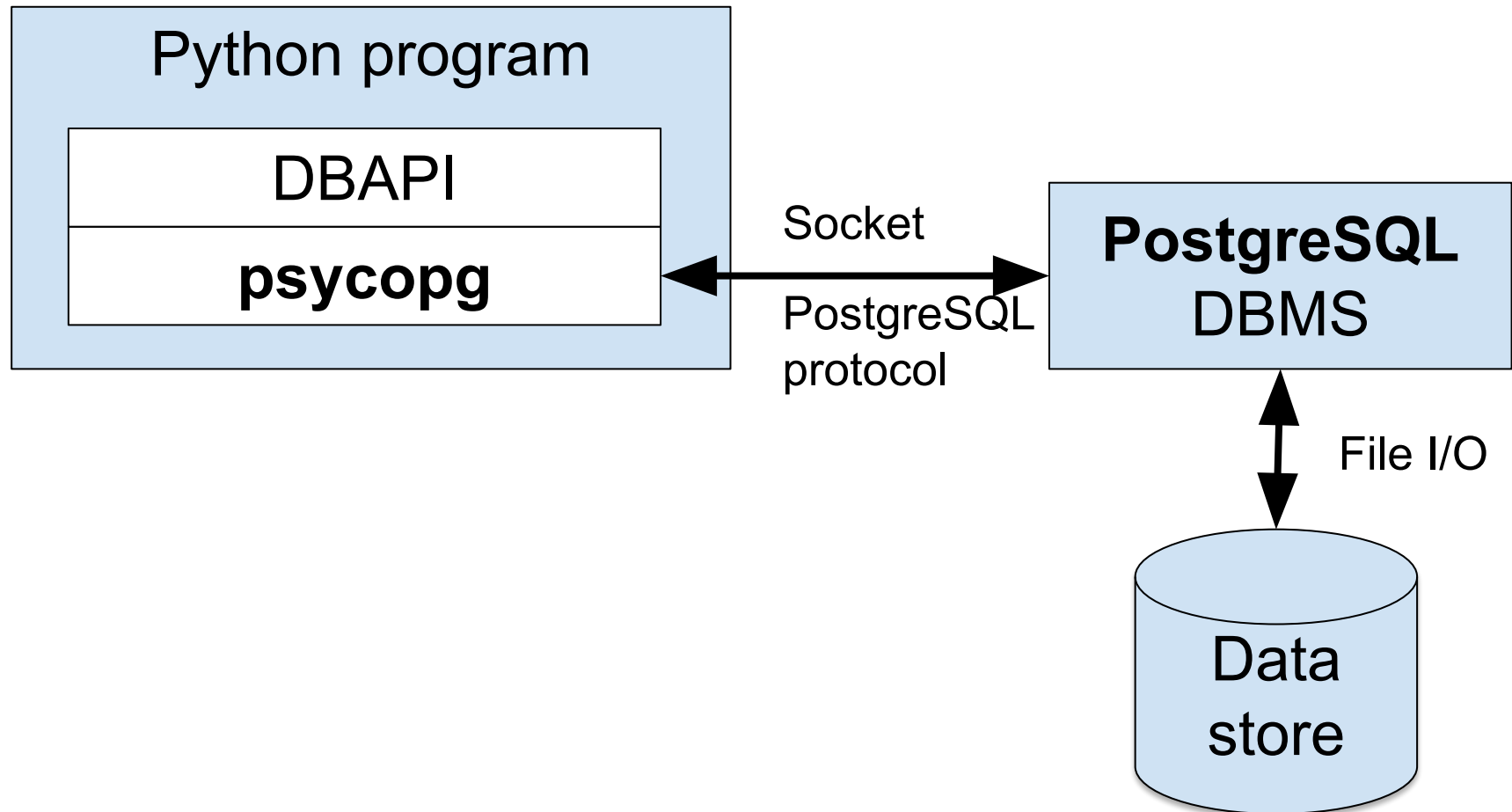
Using the DB

- See also these features on Neon website...
 - *Postgres database → SQL Editor*
 - *Postgres database → Tables*

Agenda

- Neon
- Creating the DB (demo)
- Using the DB (demo)
- **Using the DB: Python**
- Render
- Defining the app
- Deploying the app (demo)
- Using the app (demo)

Using the DB: Python



Using the DB: Python

- Installing the `psycopg` driver:

```
$ activate333  
$ python -m pip install psycopg  
$
```

Using the DB: Python

- In subsequent slides all source code files are in the pennyrenderneon directory

Using the DB: Python

- See **databasesqlite.py**
 - Baseline version
 - Uses SQLite

```
$ python databasesqlite.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- See **databasepostgres.py**
 - Uses PostgreSQL
 - Uses an environment variable

```
$ export DATABASE_URL="YourNeonDbUrl"
$ python databasepostgres.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- **Problem:**
 - Creating PostgreSQL DB connections is *very* slow
- **Attempted solution:**
 - One global DB connection

Using the DB: Python

- See **databasepostgresbad.py**

```
$ export DATABASE_URL="YourNeonDbUrl"
$ python databasepostgres2bad.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- **Problem 1:**
 - What if the DB connection fails?
- **Problem 2:**
 - Race condition!!!
- **Solution:**
 - ***DB connection pooling***
 - Maintain a “pool” of open DB connections that threads can use

Using the DB: Python

- Installing PostgreSQL connection pooling

```
$ activate333  
$ python -m pip install psycopg-pool  
$
```

Using the DB: Python

- See **databasepostgrespool.py**

```
$ export DATABASE_URL="YourNeonDbUrl"
$ python databasepostgrespool.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- **Alternative solution:**
 - *SQLAlchemy*

Using the DB: Python

- ***SQLAlchemy***
 - An ***Object Relational Mapper (ORM)*** for Python
 - Maps each DB table to a Python class
 - Maps each DB row to a Python object
 - Works for many popular relational DBMSs
 - SQLite, PostgreSQL, MySQL and MariaDB, Oracle, Microsoft SQL Server
 - See optional lecture for more thorough description

Using the DB: Python

- Installing SQLAlchemy

```
$ activate333  
$ python -m pip install SQLAlchemy  
$
```

Using the DB: Python

- See **databasesqlalchemy.py**

```
$ export DATABASE_URL="YourNeonDbUrl"
$ python databasesqlalchemy.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- See **databasesqlalchemy.py** (cont.)
 - Bonus:

```
$ export DATABASE_URL="sqlite:///penny.sqlite"
$ python database.py
123
Kernighan
The Practice of Programming

234
Kernighan
The C Programming Language

$
```

Using the DB: Python

- SQLAlchemy in COS 333 projects?
 - Pros:
 - Hides SQL
 - Portable across many relational DBMSs
 - Automatically uses prepared statements
 - Automatically provides DB connection pooling
 - Allows use of *Alembic* (database migration)
 - Harder to make a mistake

Using the DB: Python

- SQLAlchemy in COS 333 projects?
 - Cons:
 - Hides SQL!!!
 - Must learn SQLAlchemy syntax
 - More complex than SQL
 - Less important than SQL
 - Python only
 - Cannot be used via CLIs (`sqlite3`, `psql`)
 - Poor documentation

Using the DB: Python

For PostgreSQL DB on local computer:

DB SW	Optimization Technique	Time * (sec)	Thread safe?
psycopg	(None)	236.9	Yes
psycopg	One global connection	3.2	No
psycopg	DB connection pooling	6.7	Yes
SQLAlchemy	DB connection pooling	12.2	Yes

* Time to call `get_books()` 10000 times

Using the DB: Python

- Copy, to database.py, any one of:
 - databasepostgres.py
 - databasepostgrespool.py
 - databasesqlalchemy.py

Agenda

- Neon
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- **Render**
- Defining the app
- Deploying the app (demo)
- Using the app (demo)

Render



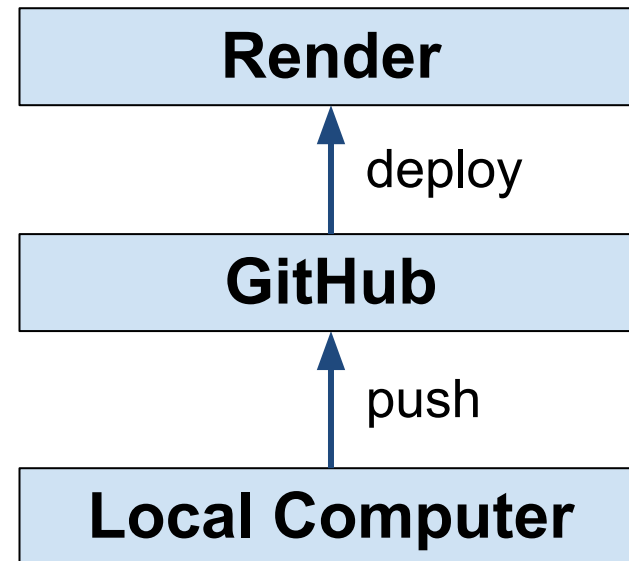
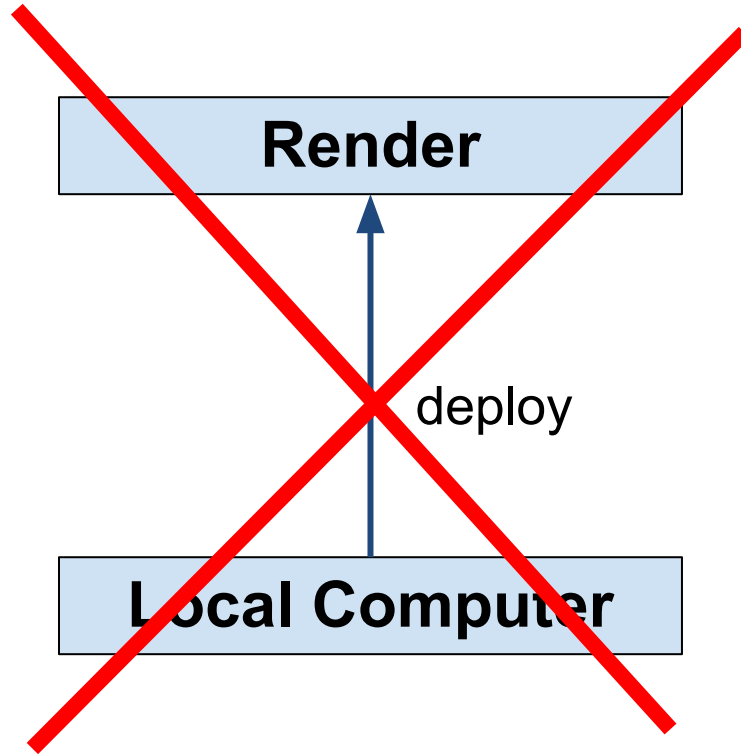
Anurag Goel
Founder and CEO

Agenda

- Neon
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Render
- **Defining the app**
- Deploying the app (demo)
- Using the app (demo)

Defining the App

- Deployment



Defining the App

- You must change names as appropriate...

Defining the App

- After performing setup steps in *Git and GitHub Primer* document...
- Create a GitHub repo
 - Browse to <https://github.com>
 - Click the *New* button
 - For *Repository name* enter `pennyrender`
 - Click the *Private* radio button
 - Check the *Add a README file* check button
 - Click the *Create Repository* button

Defining the App

- Clone the GitHub repo to your local computer

```
$ git clone https://github.com/rdondero/pennyrender.git
```

- Creates `pennyrender` dir on local computer

Defining the App

- Create these files in pennyrender dir:
 - From the PennyFlaskJinjaState app
 - runserver.py (optionally)
 - penny.sql
 - penny.sqlite (optionally)
 - header.html, footer.html, index.html, searchform.html, searchresults.html
 - penny.py
 - From this lecture
 - database.py

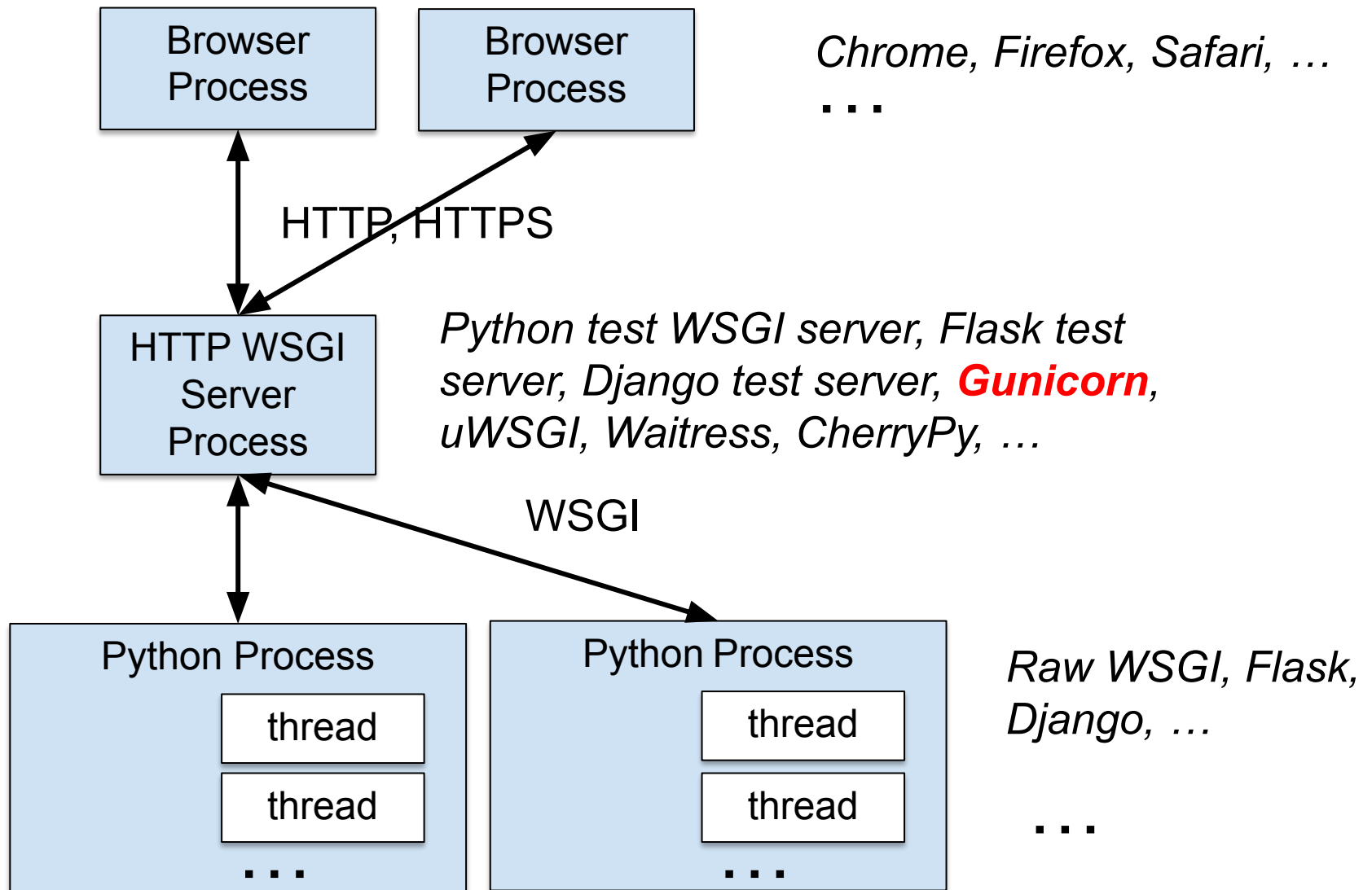
Defining the App

- Also create in pennyrender dir:
 - **requirements.txt**

```
Flask  
psycopg  
psycopg-pool  
SQLAlchemy  
python-dotenv  
gunicorn
```

- Tells Render what additional modules the app uses
- Create manually, or issue command:
`python -m pip freeze > requirements.txt`

Aside: Gunicorn



Defining the App

- Stage the app to the local git repo, commit the app to the local git repo, and push it to the GitHub repo

```
$ cd pennyrender  
$ git add -A  
$ git commit -m "initial load"  
$ git push origin main
```

Agenda

- Neon
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Render
- Defining the app
- **Deploying the app (demo)**
- Using the app (demo)

Deploying the App

- Create a Render account
 - Browse to <https://render.com>
 - Click *Start for Free*
 - Follow the instructions
 - Need not provide a credit card number

Deploying the App

- “Install Render” on GitHub
 - Inform GitHub that Render is permitted to access the repo
 - Browse to <https://github.com/apps/render/installations/new>
 - Click *rdontero*

Deploying the App

- “Install Render” on GitHub (cont.)
 - In the resulting Render page
 - Click *Only select repositories*
 - Click *Select repositories*
 - Choose *rdontero/pennyrender*
 - Click *Save*

Deploying the App

- Browse to Render dashboard
 - Browse to <https://dashboard.render.com>
 - Click *Dashboard*

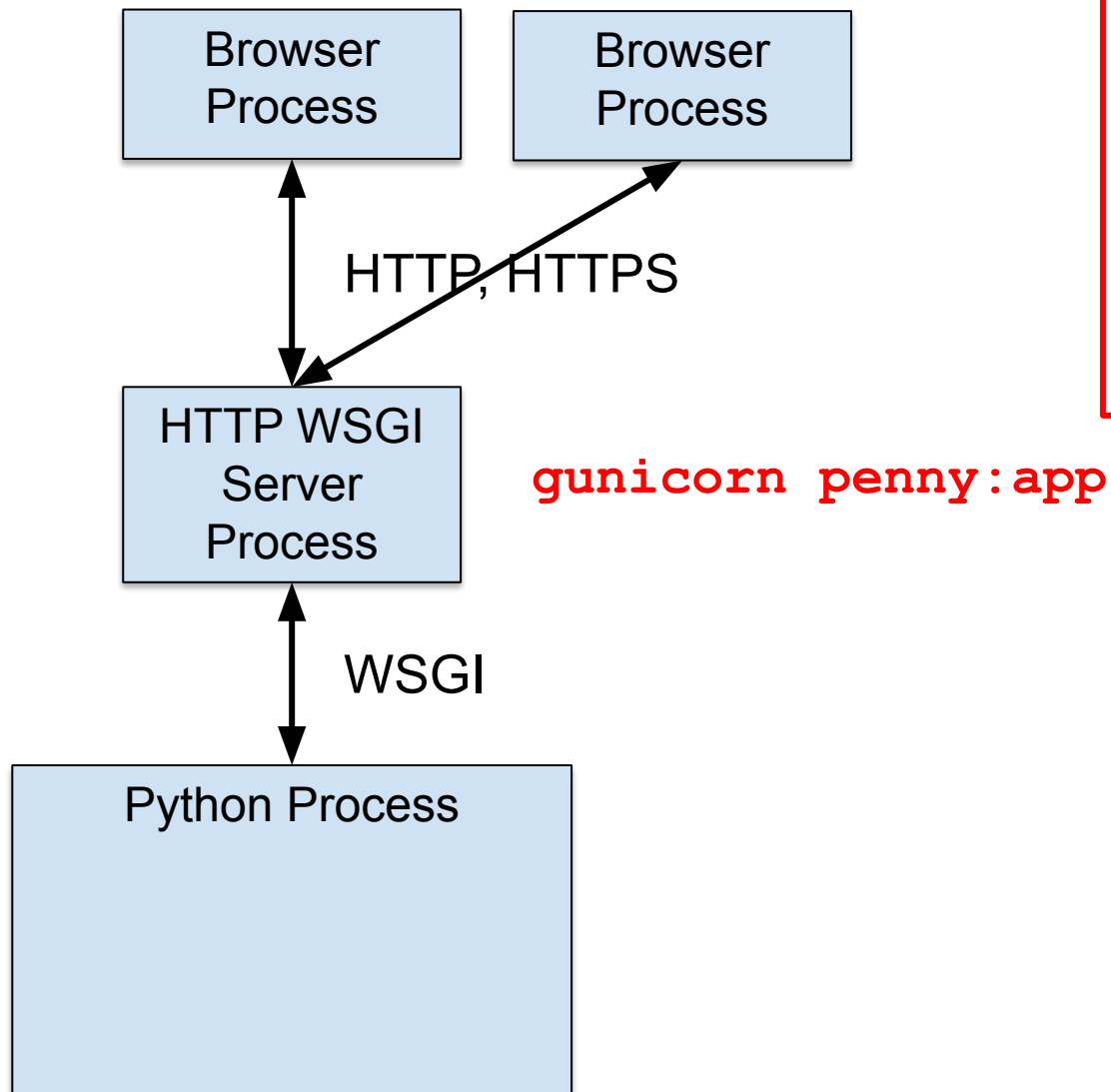
Deploying the App

- Create a new Render app
 - In the *Dashboard* page
 - Click + *New*
 - Select *Web Service*
 - Select *Git Provider*
 - Select `rdondero/pennyrender`

Deploying the App

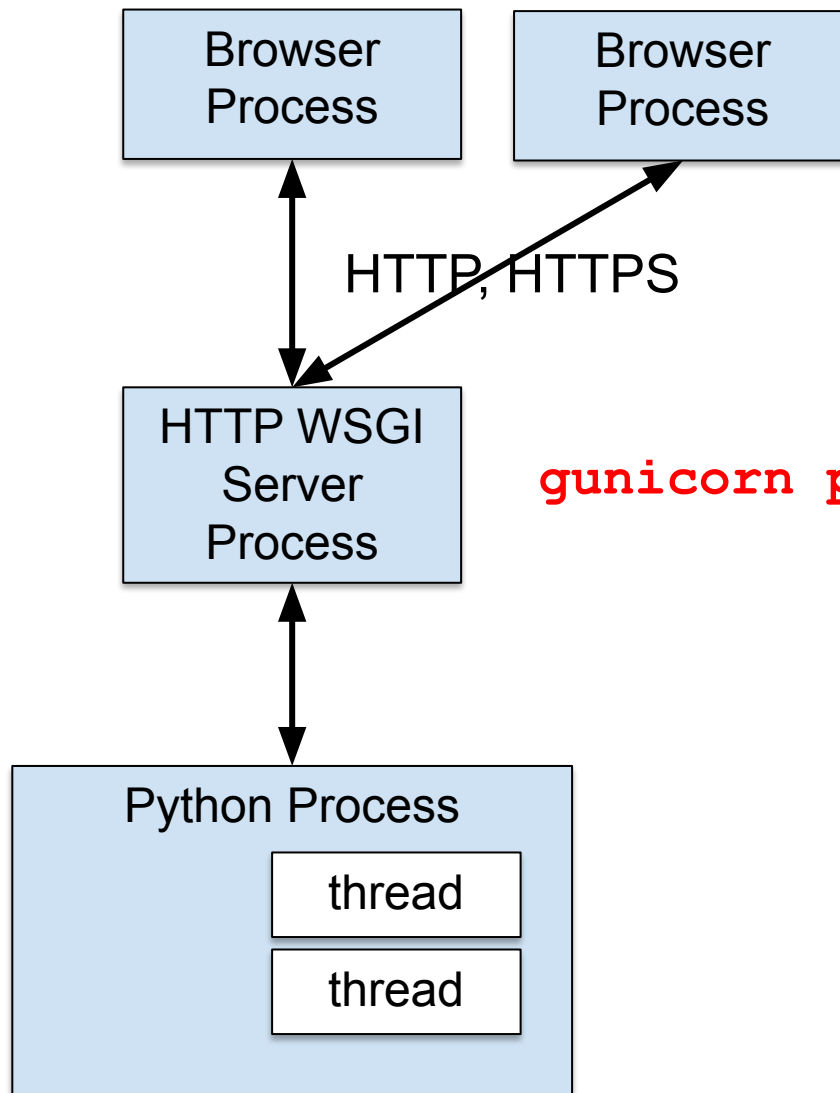
- Create a new app (cont.)
 - In the resulting page
 - For *Name* enter **pennyrender**
 - For *Region* choose **Ohio (US East)**
 - For *Start Command* enter `gunicorn penny:app`
 - For *Instance Type* choose **Free**
 - In *Environment Variables*
 - For *NAME_OF_VARIABLE* enter `DATABASE_URL`
 - For *value* enter `YourNeonDbUrl`
 - In *Advanced*
 - Set *Auto-Deploy* to *Off*
 - Click *Deploy Web Service*

Aside: Gunicorn



Should implement
DB connection
pooling via
ConnectionPool or
SQLAlchemy
(min_size=1)

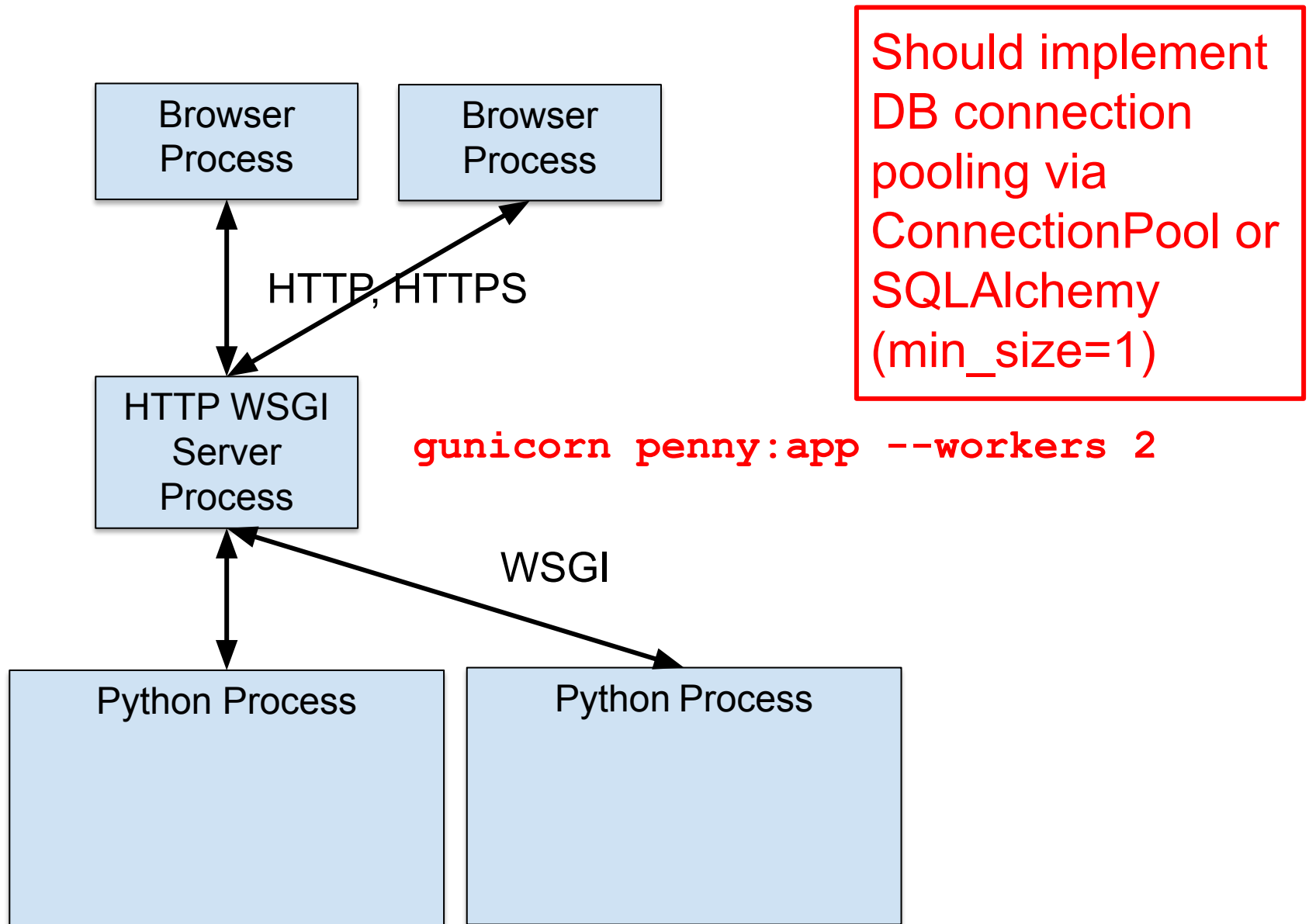
Aside: Gunicorn



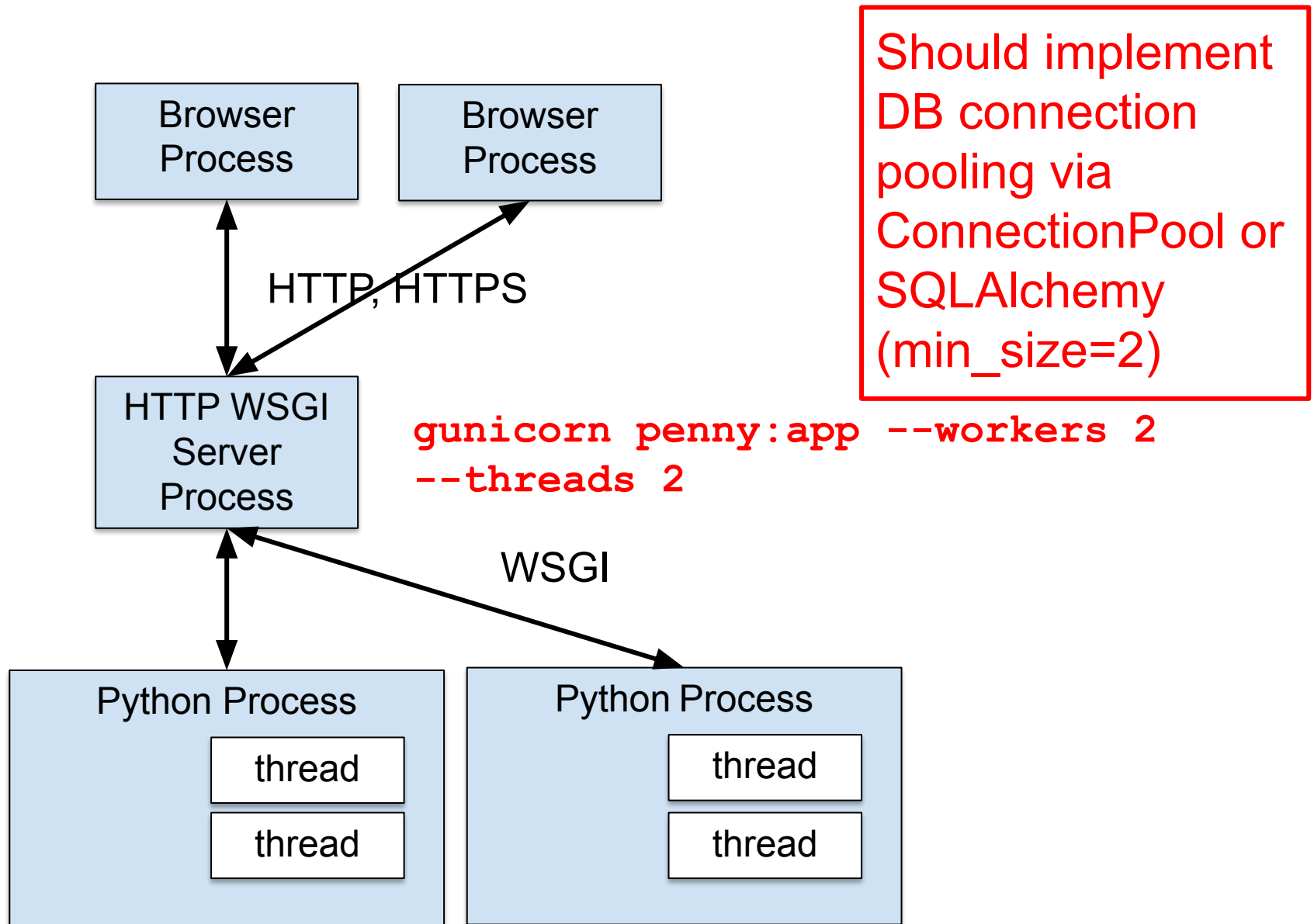
Should implement
DB connection
pooling via
ConnectionPool or
SQLAlchemy
(min_size=2)

`gunicorn penny:app --threads 2`

Aside: Gunicorn



Aside: Gunicorn



Deploying the App

- Create a new app (cont.)
 - The app deployment begins
 - Wait up to 5 minutes for deployment to finish



Deploying the App

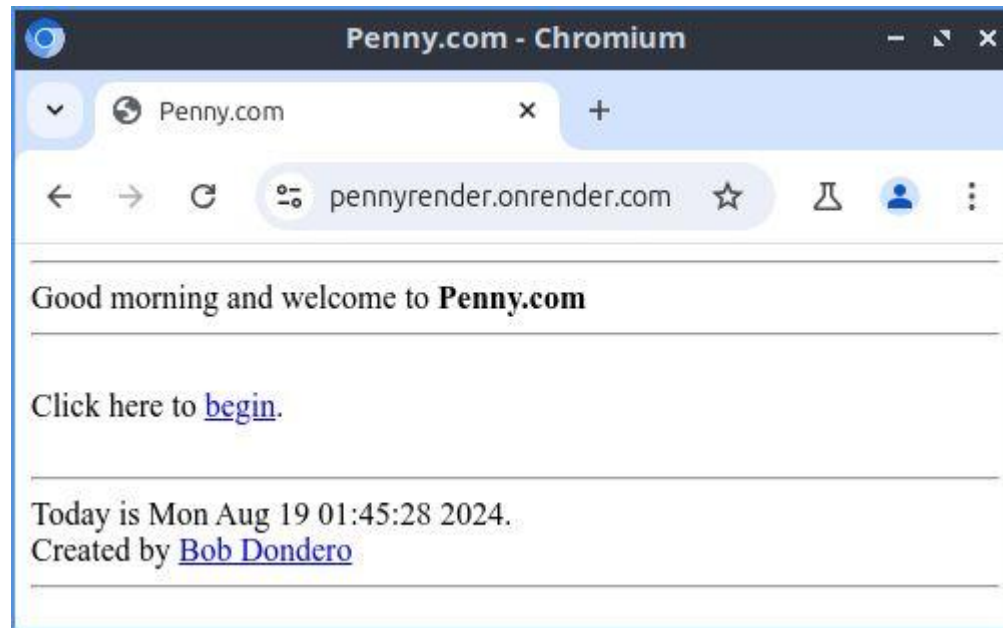
- Thereafter:
 - Push new code from your local git repo to the pennyrender GitHub repo
 - In Render, click *Deploys* → *Manual Deploy*
 - Repeat as necessary

Agenda

- Neon
- Creating the DB (demo)
- Using the DB (demo)
- Using the DB: Python
- Render
- Defining the app
- Deploying the app (demo)
- **Using the app (demo)**

Using the App

- Browse to the app!
 - <https://pennyrender.onrender.com>



Lecture Summary

- In this lecture we covered:
 - How to deploy a database to Neon
 - Database connection pooling
 - How to deploy a Web app to Render

Lecture Summary

- See also:
 - **Optional lecture:** SQLAlchemy
 - **Optional lecture:** PostgreSQL
 - **Optional lecture:** Deployment: Render App & Render DB
 - **Optional lecture:** Deployment: Heroku App & Heroku DB