


CS-GY 6083 A: Principles of Database Systems

Lab 6: Deploying and Building a Streamlit Application



Streamlit Demonstration

1. Review Streamlit API
2. Upload files to jedi
3. Create and populate a database
4. Run the Streamlit application
5. Common questions and issues
6. Ship it 

Streamlit API

<https://docs.streamlit.io/library/api-reference>

The screenshot shows the Streamlit API reference page. The left sidebar contains a 'Streamlit library' menu with categories like 'Get started', 'API reference', 'Write and magic', 'Text elements', 'Data display elements', 'Chart elements', 'Input widgets', 'Media elements', 'Layouts and containers', 'Status elements', 'Control flow', 'Utilities', 'Mutate charts', 'State management', 'Performance', and 'Advanced features'. The main content area is titled 'API reference' and includes a brief introduction: 'Streamlit makes it easy for you to visualize, mutate, and share data. The API reference is organized by activity type, like displaying data or optimizing performance. Each section includes methods associated with the activity type, including examples.' Below this, it says 'So browse our API below and click to learn more about any of our available commands!'. The section 'Display almost anything' is highlighted, showing two examples: 'st.write' (Write arguments to the app.) and 'Magic' (Any time Streamlit sees either a variable or literal value on its own line, it automatically writes that to your app using st.write). The 'st.write' example shows code: `st.write("Hello **world**!")`, `st.write(my_data_frame)`, and `st.write(my_mpl_figure)`. The 'Magic' example shows code: `"Hello **world**!"`, `my_data_frame`, and `my_mpl_figure`.

The screenshot shows the Streamlit Input widgets documentation page. The left sidebar is the same as the previous page, but the 'API reference' category is selected. The main content area is titled 'Input widgets' and displays a grid of widget examples. Each example consists of a visual representation of the widget and a description of its usage. The widgets shown are: 'Button' (Display a button widget. `clicked = st.button("Click me")`), 'Download button' (Display a download button widget. `st.download_button("Download file", ...)`), 'Checkbox' (Display a checkbox widget. `selected = st.checkbox("I agree")`), 'Radio' (Display a radio button widget. `classified = st.radio("Classify image", ["Dog", "Cat", "Goldfish"])`), 'Selectbox' (Display a select widget. `selected = st.selectbox("Select a widget", ...)`), and 'Multiselect' (Display a multiselect widget. The multiselect widget starts as empty. `selected = st.multiselect("Multiselect widget", ...)`).

Uploading files

Step 2: Upload code and data to the jedi server

The code and data (including this document itself) is in [this Google Drive folder](#).

You need to change the postgres account info in the [database.ini](#) file.

Download the code and data, then use `scp` to upload the files to jedi. Replace **PATH** with the path to the files on your local machine and **NET_ID** with your netId.

```
scp -r ~/PATH NET_ID@jedi.poly.edu:~/project_demo
```

path to directory on **your**
local machine

project directory name (on remote)

[Tutorial on how to use scp](#) (click me)

Create and populate database

Step 3: Create and populate the database

1. Log into jedi:

Replace **NET_ID**

Sorry about the confusion port forwarding will appear later in this document

```
ssh NET_ID@jedi.poly.edu
```

2. Create tables using the postgres script ./data/create_db.sql

Replace **NET_ID**

```
psql -d NET_ID_db -a -f project_demo/data/create_db.sql
```

3. Populate the tables with data in the CSV files.

Replace **NET_ID**

```
cat project_demo/data/customers.csv | psql -U NET_ID -d NET_ID_db -c "COPY  
customers from STDIN CSV HEADER"
```

```
cat project_demo/data/orders.csv | psql -U NET_ID -d NET_ID_db -c "COPY  
orders from STDIN CSV HEADER"
```

Run the Streamlit application

Step 4: Port forwarding and run the Streamlit app

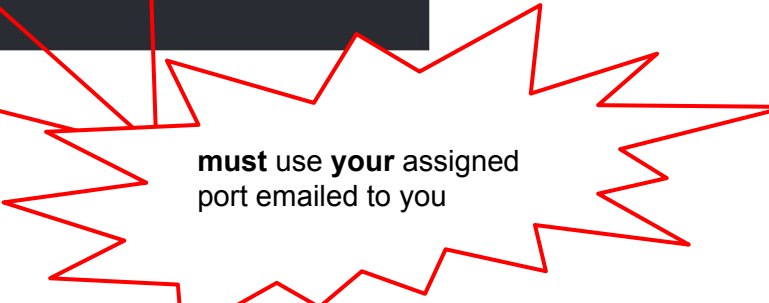
The database has been created. Log out of the Jedi server above, then run SSH port forwarding in a local terminal to the dedicated port number (**ASSIGNED_PORT**).

```
ssh NET_ID@jedi.poly.edu -L ASSIGNED_PORT:localhost ASSIGNED_PORT
```

Change directory to `./project_demo` and run the streamlit app.

```
cd project_demo
```

```
streamlit run demo.py --server.address=localhost  
--server.port=ASSIGNED_PORT
```



must use **your** assigned
port emailed to you

Common questions and issues

How do I keep my Streamlit project running?

Tmux

<https://tmuxcheatsheet.com/>

<https://linuxize.com/post/getting-started-with-tmux/>

My port is already in use, what do I do?

Killing errant processes

```
lsof -i :port_number  
kill -9 PID
```

```
> lsof -i  
COMMAND  PID    USER  FD  TYPE  DEVICE  SIZE/OFF  NODE NAME  
loginwind 142    iansolano  8u  IPv4  0x2058281d79c135cb  0t0  UDP *:*
```

*** only works on unix-like machines

Also see: [Project FAQs](#)