Contents

[STREAMLIT 3](#_Toc188006326)

[SETTING UP STREAMLIT 3](#_Toc188006327)

[SIMPLE STREAMLIT APP 3](#_Toc188006328)

[ENABLE AUTO RELOAD 4](#_Toc188006329)

[TEXT DISPLAY ELEMENTS 4](#_Toc188006330)

[LATEX 5](#_Toc188006331)

[WRITE 5](#_Toc188006332)

[INPUT WIDGETS 6](#_Toc188006333)

[BUTTONS 6](#_Toc188006334)

[CHECKBOX 6](#_Toc188006335)

[RADIO 6](#_Toc188006336)

[SELECTBOX 7](#_Toc188006337)

[MULTISELECT 7](#_Toc188006338)

[SLIDER 7](#_Toc188006339)

[TEXT INPUT 8](#_Toc188006340)

[NUMBER INPUT 8](#_Toc188006341)

[TEXT AREA 8](#_Toc188006342)

[IMAGE 8](#_Toc188006343)

[DATA DISPLAY ELEMENTS 8](#_Toc188006344)

[DATAFRAMES 8](#_Toc188006345)

[DISPLAYING DATAFRAMES 8](#_Toc188006346)

[DISPLAYING TABLES 9](#_Toc188006347)

[DISPLAYING METRIC 9](#_Toc188006348)

[CHARTING ELEMENTS 10](#_Toc188006349)

[CREATING LINE PLOT 10](#_Toc188006350)

[CREATING AREA CHART 11](#_Toc188006351)

[CREATING BAR CHART 12](#_Toc188006352)

[MAP 12](#_Toc188006353)

[FORMS 13](#_Toc188006354)

[LAYOUT 15](#_Toc188006355)

[CREATING SIDEBAR 15](#_Toc188006356)

[CREATING CONTAINERS 16](#_Toc188006357)

[CREATING COLUMNS 16](#_Toc188006358)

[CREATING TABS 16](#_Toc188006359)

[EXPANDER(ACCORDION) 17](#_Toc188006360)

[CACHING 17](#_Toc188006361)

[SESSION MANAGEMENT 17](#_Toc188006362)

[MULTIPAGE APPLICATIONS 17](#_Toc188006363)

[PAGE CONFIGURATION 17](#_Toc188006364)

[AUTHENTICATION 17](#_Toc188006365)

[CONNECT TO DATASOURCE 17](#_Toc188006366)

[DEPLOYMENT – RENDER 17](#_Toc188006367)

[STEP 1: LIST THE APP DEPENDENCIES 17](#_Toc188006368)

[STEP 2: COMMIT THE CODE IN DEDICATED REPO 18](#_Toc188006369)

[STEP 3: CREATE A NEW WEB SERVICE AND CONNECT GITHUB REPOS 18](#_Toc188006370)

[STEP 4: SETUP DEPLOYMENT SETTINGS 19](#_Toc188006371)

[SETTING ENVIRONMENT VARIABLES 22](#_Toc188006372)

# STREAMLIT



* Streamlit is an open-source Python library used **for building interactive web applications** for data science and machine learning projects.
* It simplifies the process of creating and sharing interactive applications by allowing developers to write simple Python scripts that are automatically transformed into interactive web apps (no JS / CSS)
* With Streamlit, **we can create interactive dashboards, data visualizations, and machine learning models**, among other things.
* It **provides a range of built-in components and widgets** **that make it easy to create user interfaces** for your applications.

## SETTING UP STREAMLIT

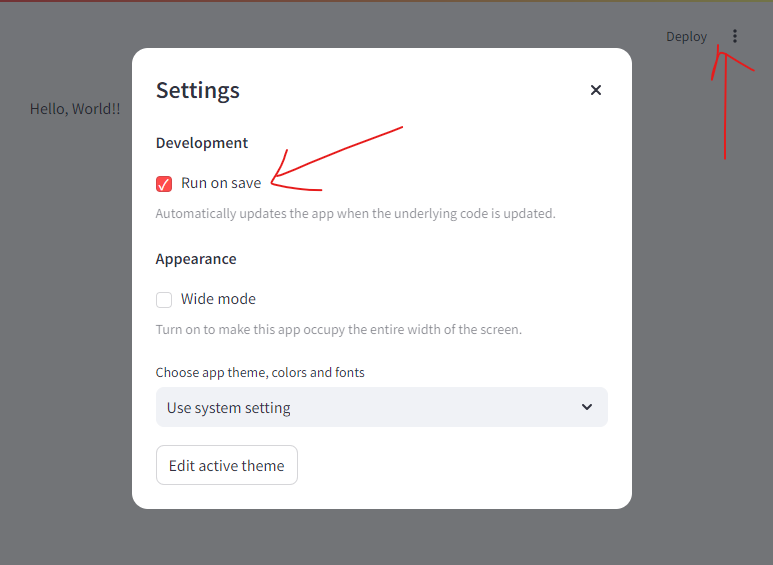
1. Create a virtual environment
2. INSTALL streamlit package : **pip install streamlit**

## SIMPLE STREAMLIT APP

|  |  |
| --- | --- |
|  | 1. Create an entry point Python file (main.py) 2. RUN THE STREAMLIT APP   **streamlit run main.py** |
|  | |

### ENABLE AUTO RELOAD

* To Enable auto reloads , keep the below settings on.



## TEXT DISPLAY ELEMENTS

|  |  |
| --- | --- |
| **import streamlit as st  # Page Title st.title("Page Title")  # Page Header st.header("Page Header")  # divider st.divider()  # Page Subheader  st.subheader("Page Sub header")  # Page Text st.text("Hello Streamlit")  # Page Markdown st.markdown("# Markdown Header 1")  st.markdown("## Markdown Header 2")  st.markdown("### Markdown Header 1")  # Page Error st.error("Error Message")  # Page Warning st.warning("Warning Message")  # Page Success  st.success("Success Message")  # Page Info st.info("Info Message")** |  |

### LATEX

* LaTeX is a typesetting system commonly used for creating documents with complex mathematical and scientific content. It is widely used in academic and scientific communities for writing research papers, technical reports, theses, and other documents.

|  |  |
| --- | --- |
| import streamlit as st  st.latex("f(x) = x^2") |  |

### WRITE

* The write function in Streamlit is a versatile function that allows you to display various types of content, including text, data, and visualizations.

|  |  |
| --- | --- |
| # Display text  st.write("Hello, world!")  # Display a list  st.write(["apple", "banana", "cherry"])  # Display a dictionary  st.write({"name": "John", "age": 30}) |  |

## INPUT WIDGETS

### BUTTONS

|  |  |
| --- | --- |
| **PRIMARY BUTTONS**  primary\_btn = st.button(**type="primary", label="Primary Button")** |  |
| **SECONDARY BUTTONS**  secondary\_btn = st.button(type="secondary", label="Secondary Button") |  |
| **BUTTON INTERACTIVITY – HANDLING BUTTON EVENTS**   * **st.button()** returns a Boolean * For example – The text will show up when button is clicked and primary\_btn or secondary\_btn will evaluate to **True** | **if primary\_btn:  st.write("Hello from primary")  if secondary\_btn:  st.write("Hello from secondary")** |

### CHECKBOX

|  |  |
| --- | --- |
| **checkbox = st.checkbox("Remember me")** |  |
| **CHECKBOX INTERACTIVITY – HANDLING CHECKBOX EVENTS**   * **st. checkbox()** returns a Boolean * For example – The text will show up accordingly based on when checkbox is clicked, and “**checkbox”** will evaluate to **True** | **if checkbox:  st.write("I will remember you") else:  st.write("I will forget you")** |

### RADIO

|  |  |
| --- | --- |
| **radio = st.radio("What's your favorite color?", options=["Red", "Green", "Blue"]) st.write(f"Your favorite color is {radio}")** |  |
| **radio = st.radio("What's your favorite color?", options=["Red", "Green", "Blue"], index=2, horizontal=True) st.write(f"Your favorite color is {radio}")**   * Horizontal=True 🡪 Alignment of Radio Button * Index = Index of radio button which will be selected by default |  |

### SELECTBOX

|  |  |
| --- | --- |
| **import pandas as pd**  **select\_box = st.selectbox("Choose a column", options=df.columns[1:], index=0) st.write(select\_box)** | **Let’s we have a csv**  **year,col1,col2,col3 2018,10,15,20 2019,12,18,22 2020,14,20,25 2021,16,22,28 2022,18,25,30** |
|  | |

### MULTISELECT

|  |  |  |
| --- | --- | --- |
| **EXAMPLE -2** |  |  |
| **Let’s we have a csv**  **year,col1,col2,col3 2018,10,15,20 2019,12,18,22 2020,14,20,25 2021,16,22,28 2022,18,25,30** | **We want to read the col1,col2 and col3 in multiselectbox**  **import pandas as pd**  **df = pd.read\_csv("data/sample.csv")**  **multiselect = st.multiselect("Choose as many columns as you want", options=df.columns[1:], default=["col2"], max\_selections=2) st.write(multiselect)** | |

### SLIDER

|  |
| --- |
| slider = st.slider("Pick a number", min\_value=0, max\_value=10, value=0, step=1) st.write(slider) |
|  |

### TEXT INPUT

|  |
| --- |
| text\_input = st.text\_input("What's your name?", placeholder="John Doe") st.write(f"Your name is {text\_input}") |
|  |

### NUMBER INPUT

|  |
| --- |
| num\_input = st.number\_input("Pick a number", min\_value=0, max\_value=10, value=0, step=1) st.write(f"You picked {num\_input}") |
|  |

### TEXT AREA

|  |
| --- |
| txt\_area = st.text\_area("What do you want to tell me?", height=500, placeholder="Write your message here") st.write(txt\_area) |
|  |

### IMAGE

|  |
| --- |
| import streamlit as st  st.**image**("data/images/aarav.jpg", width=300) |

## DATA DISPLAY ELEMENTS

* Elements to show tabular data / table

### DATAFRAMES

* A data frame is a two-dimensional data structure, similar to a table in a database or a spreadsheet, where data is organized in rows and columns.
* Each column in a data frame can have a different data type (e.g., numeric, character, factor), and each row represents an observation or record.
* Data frames are commonly used for data manipulation, analysis, and visualization in statistical and data science tasks.

### DISPLAYING DATAFRAMES

* Streamlit provides a convenient way to display and visualize tabular data using the st.dataframe() function.
* We can pass a Pandas DataFrame or any other tabular data structure to this function, and Streamlit will render it as an **interactive table** in the web application.
* This allows users to explore and interact with the data, such as **sorting columns, filtering rows, and searching for specific value**

|  |  |  |
| --- | --- | --- |
| **import streamlit as st**  **import pandas as pd**  **df = pd.read\_csv("data/sample.csv", dtype=int)**  **st.dataframe(df)** | **sample.csv**  **year,col1,col2,col3 2018,10,15,20 2019,12,18,22 2020,14,20,25 2021,16,22,28 2022,18,25,30** |  |

### DISPLAYING TABLES

* The term "table" in Streamlit generally refers to the visual representation of tabular data using the st.table() function.
* Similar to st.dataframe(), we can pass a Pandas DataFrame or any other tabular data structure to this function, and Streamlit will display it as a table.
* **However, unlike st.dataframe(), the st.table() function doesn't provide interactive features like sorting or filtering. It simply displays the data as a static table.**

|  |  |
| --- | --- |
| **import streamlit as st**  **import pandas as pd**  **df = pd.read\_csv("data/sample.csv", dtype=int)**  **st.table(df)** | **year,col1,col2,col3 2018,10,15,20 2019,12,18,22 2020,14,20,25 2021,16,22,28 2022,18,25,30** |
|  | |

### DISPLAYING METRIC

* The ***st.metric()*** function is used to display a metric value along with its label in a visually appealing format.
* It is designed to showcase a single numeric value, such as a key performance indicator (KPI) or a measurement.

|  |  |
| --- | --- |
| st.metric("Revenue", "$1,000,000") |  |
| st.metric("Total Sales", "$1,000,000", delta=10, delta\_color="**normal**") |  |
| st.metric("Total Sales", "$1,000,000", delta=-10, delta\_color="**normal**") |  |
| st.metric("Total Expenses", "$6,000", delta=-10, delta\_color="**inverse**")   * delta\_color=”inverse” - for the cases where drop/ decrease are positive e.g expenses |  |
| st.metric("Total Expenses", "$6,000", delta=10, delta\_color="**inverse**") |  |

## CHARTING ELEMENTS

**SOURCE CSVs**

|  |  |
| --- | --- |
| sample.csv | [sample\_map.csv](https://github.com/marcopeix/MachineLearningModelDeploymentwithStreamlit/blob/master/07_charting_elements/data/sample_map.csv) |
| **year,col1,col2,col3**  **2018,10,15,20**  **2019,12,18,22**  **2020,14,20,25**  **2021,16,22,28**  **2022,18,25,30** | **latitude,longitude**  **43.6532,-79.3832**  **49.2827,-123.1207**  **51.0447,-114.0719**  **45.4215,-75.6981**  **53.5444,-113.4909** |

### CREATING LINE PLOT

* The st.line\_chart() function is used to create a line chart. We can pass a Pandas DataFrame or a list of values to this function, and Streamlit will generate a line chart based on the data.

|  |  |
| --- | --- |
| **import streamlit as st**  **import pandas as pd**    **# Create a line chart**  **data = {'x': [1, 2, 3, 4, 5], 'y': [10, 15, 7, 12, 9]}**  **df = pd.DataFrame(data)**  **st.line\_chart(df)** | In this example, a line chart is created with the x-axis values [1, 2, 3, 4, 5] and the corresponding y-axis values [10, 15, 7, 12, 9]. |
|  | |
| **import streamlit as st import pandas as pd  df = pd.read\_csv("data/sample.csv", dtype=int)  # line chart st.line\_chart(df, x="year", y=["col1", "col2", "col3"])** | |
|  | |

### CREATING AREA CHART

* The **st.area\_chart()** function is used to create an area chart. It accepts a Pandas DataFrame or a list of values and generates an area chart based on the data.

|  |  |
| --- | --- |
| **import streamlit as st**  **import pandas as pd**  **# Create an area chart**  **data = {'x': [1, 2, 3, 4, 5], 'y': [10, 15, 7, 12, 9]}**  **df = pd.DataFrame(data)**  **st.area\_chart(df)** | In this example, an area chart is created with the x-axis values [1, 2, 3, 4, 5] and the corresponding y-axis values [10, 15, 7, 12, 9]. |
|  | |
| **import streamlit as st import pandas as pd  df = pd.read\_csv("data/sample.csv", dtype=int)  # area chart st.area\_chart(df, x="year", y=["col1", "col2", "col3"])** | |
|  | |

### CREATING BAR CHART

|  |
| --- |
| **import streamlit as st import pandas as pd  df = pd.read\_csv("data/sample.csv", dtype=int) st.bar\_chart(df, x="year", y=["col1", "col2", "col3"])** |
|  |

### MAP

* We can create map plots using the `st.map()` function. This function allows us to display geographical data on an interactive map in Streamlit web application.
* To use `st.map()`, we need to provide the latitude and longitude coordinates of the locations we want to plot.
* We can pass a Pandas DataFrame or a list of coordinates to the function. Additionally, we can customize the appearance of the map using parameters such as zoom level and map style.

|  |
| --- |
| **import streamlit as st**  **import pandas as pd**    **# Create a DataFrame with latitude and longitude coordinates**  **data = {'lat': [51.5074, 40.7128, 48.8566], 'lon': [-0.1278, -74.0060, 2.3522]}**  **df = pd.DataFrame(data)**    **# Display the map plot**  **st.map(df)** |

|  |  |
| --- | --- |
| [sample\_map.csv](https://github.com/marcopeix/MachineLearningModelDeploymentwithStreamlit/blob/master/07_charting_elements/data/sample_map.csv) | # Streamlit map geo\_df = pd.read\_csv("data/sample\_map.csv") st.map(geo\_df) |
| **latitude,longitude**  **43.6532,-79.3832**  **49.2827,-123.1207**  **51.0447,-114.0719**  **45.4215,-75.6981**  **53.5444,-113.4909** |
|  | |

## FORMS

* With st.form(key="form1"): is part of a Streamlit application that creates a form for users to fill out. The st.form function is used to group multiple input widgets together, allowing the user to submit all the inputs at once.
* ***The key parameter uniquely identifies the form.***

|  |
| --- |
| **import streamlit as st  with st.form(key="form1"):  st.write("What u like to order")  appetizers = st.selectbox("Appetizers", index=0, options=["Fries", "Onion Rings", "Cheese Sticks"])  main\_course = st.selectbox("Main Course", index=0, options=["Pizza", "Burger", "Pasta"])  deserts = st.selectbox("Deserts", index=0, options=["Ice Cream", "Cake", "Donuts"])  checkbox = st.checkbox("Are you bringing a friend?")  day = st.date\_input("When you are coming")  time = st.time\_input("At what time you are coming")  allergies = st.text\_area("Any allergies we should know about?", height=100, placeholder="Write here")  submit\_button = st.form\_submit\_button("Submit")  st.write(f"Your order is {appetizers}") st.write(f"Your order is {main\_course}") st.write(f"Your order is {deserts}") st.write(f"Are you bringing a friend? {"Yes" if checkbox else "No"}") st.write(f"You are coming on {day}") st.write(f"You are coming at {time}") if allergies:  st.write(f"Your allergies are {allergies}")** |
|  |

## LAYOUT

* We can control the layout of the app using various layout functions and decorators provided by the Streamlit library

### CREATING SIDEBAR

|  |
| --- |
| **import streamlit as st  # Sidebar with st.sidebar:  st.write("Text in the sidebar")** |

### CREATING CONTAINERS

|  |
| --- |
| **with st.container():**  **st.write("This is inside the container")**  **st.write("This is outside the container")** |

### CREATING COLUMNS

|  |
| --- |
| # Three column layout col1, col2, col3 = st.columns(3) |

#### PLACING ELEMENTS IN COLUMNS

|  |
| --- |
| **# Three column layout col1, col2, col3 = st.columns(3)  # First column with col1:  st.write("First column")  st.text\_input("Text input")  # Second column with col2:  st.write("Second column")  st.slider("Slider", min\_value=0, max\_value=10, step=1)  # Third column with col3:  st.write("Third column")  st.button(type="primary", label="Button")** |
|  |

### CREATING TABS

|  |
| --- |
| **tab1, tab2 = st.tabs(["Tab 1", "Tab 2"])** |

#### PLACING ELEMENTS IN TABS

|  |
| --- |
| **tab1, tab2 = st.tabs(["Tab 1", "Tab 2"])  with tab1:  st.write("Content of Tab 1")  st.checkbox("Are u in Tab1 ")  with tab2:  st.write("Content of Tab 2")  st.selectbox("Select box in Tab2", options=["Option 1", "Option 2", "Option 3"])** |
|  |

### EXPANDER(ACCORDION)

|  |
| --- |
| **with st.expander("Click to expand"):  st.write("Content inside the expander")** |
|  |

# CACHING

# SESSION MANAGEMENT

# MULTIPAGE APPLICATIONS

## PAGE CONFIGURATION

# AUTHENTICATION

# CONNECT TO DATASOURCE

# DEPLOYMENT – RENDER

* Render : [Cloud Application Platform | Render](https://render.com/)

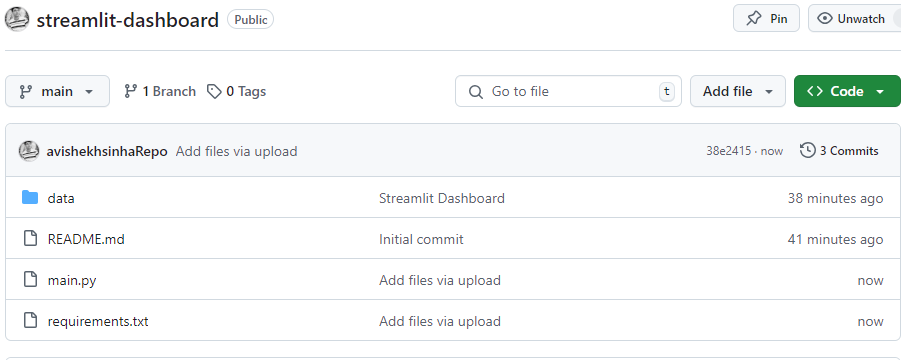
### STEP 1: LIST THE APP DEPENDENCIES

* List the project dependencies in the requirements.txt

|  |  |
| --- | --- |
| COMMAND TO CREATE requirements.txt | **pip freeze > requirements.txt** |

### STEP 2: COMMIT THE CODE IN DEDICATED REPO

* To deploy to render, we need to commit the code to GitHub.
* Any further commit to in the repo will re-deploy the application.
* Example : [avishekhsinhaRepo/streamlit-dashboard](https://github.com/avishekhsinhaRepo/streamlit-dashboard)



* **Note: Do not commit the secrets in Github**

### STEP 3: CREATE A NEW WEB SERVICE AND CONNECT GITHUB REPOS

A screenshot of a computer

Description automatically generated

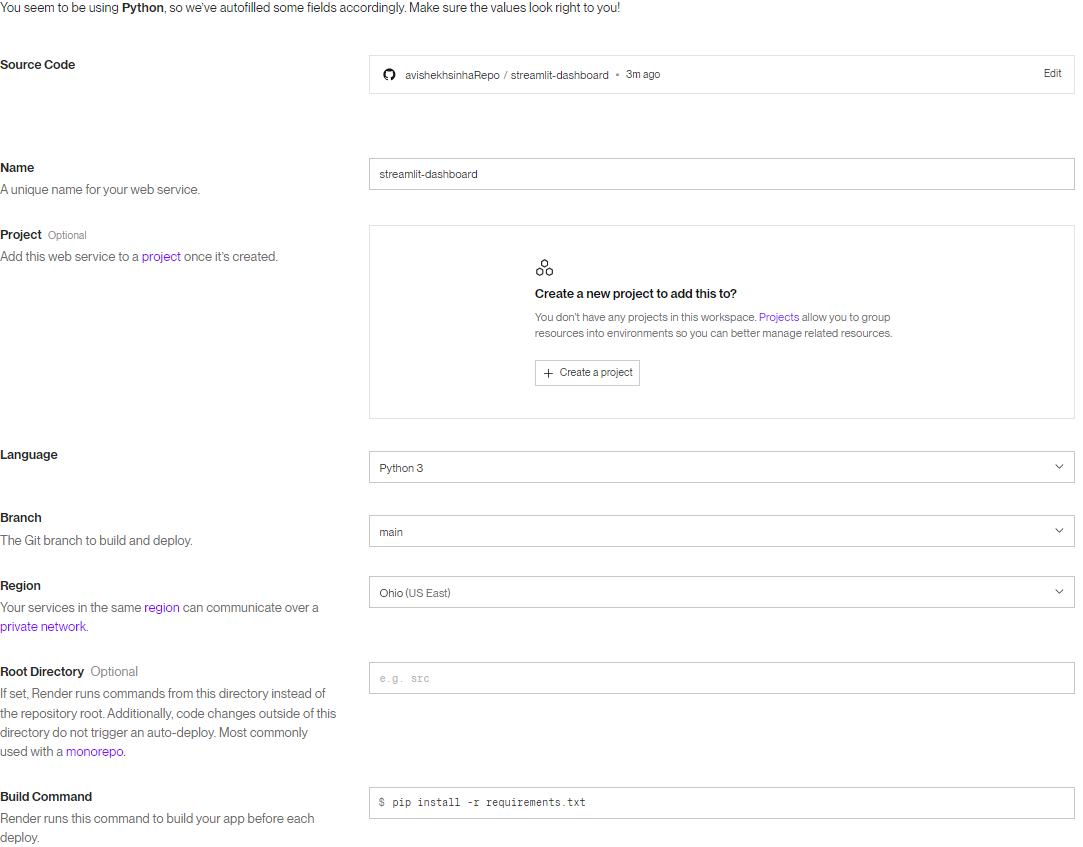
A screenshot of a computer

Description automatically generated

1. Select the application and click on Connect

### STEP 4: SETUP DEPLOYMENT SETTINGS

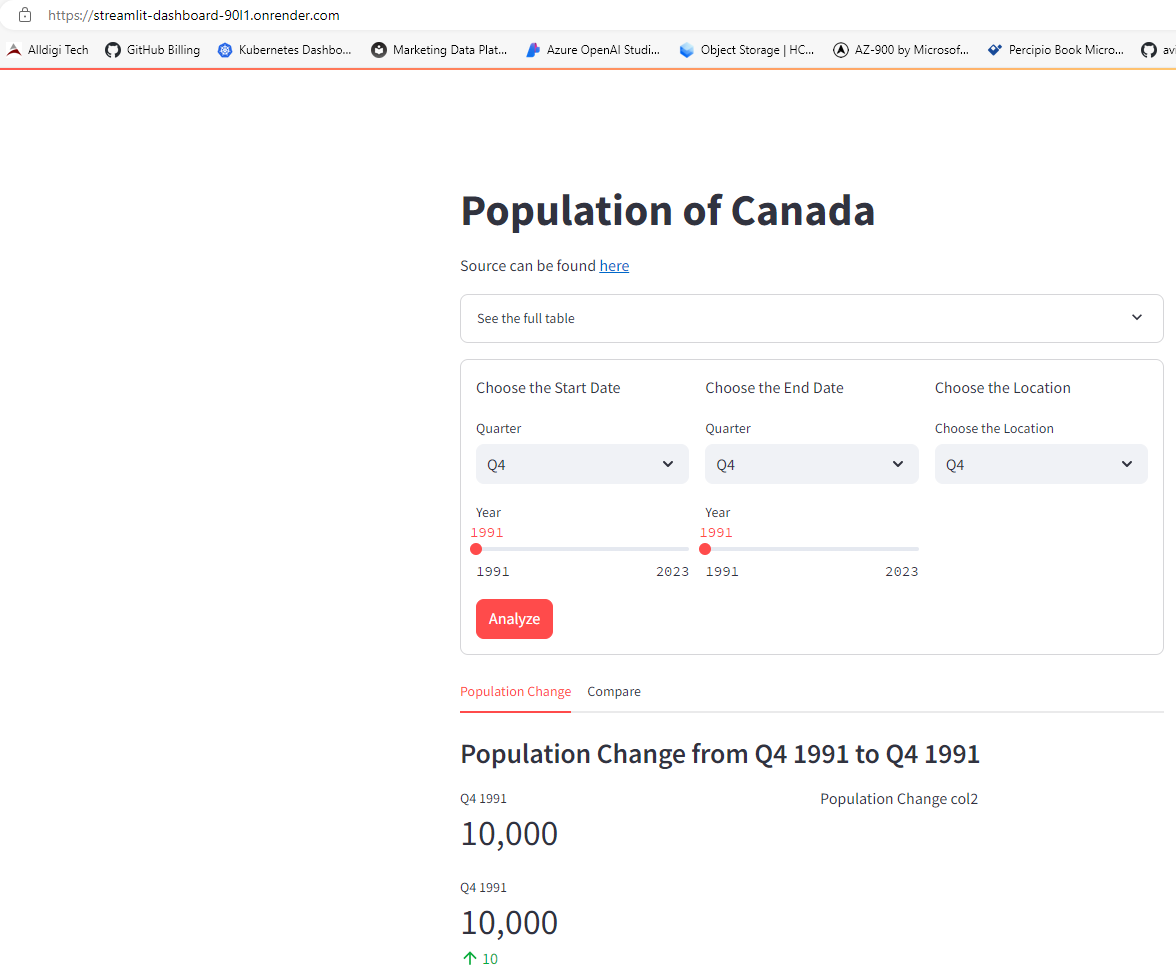
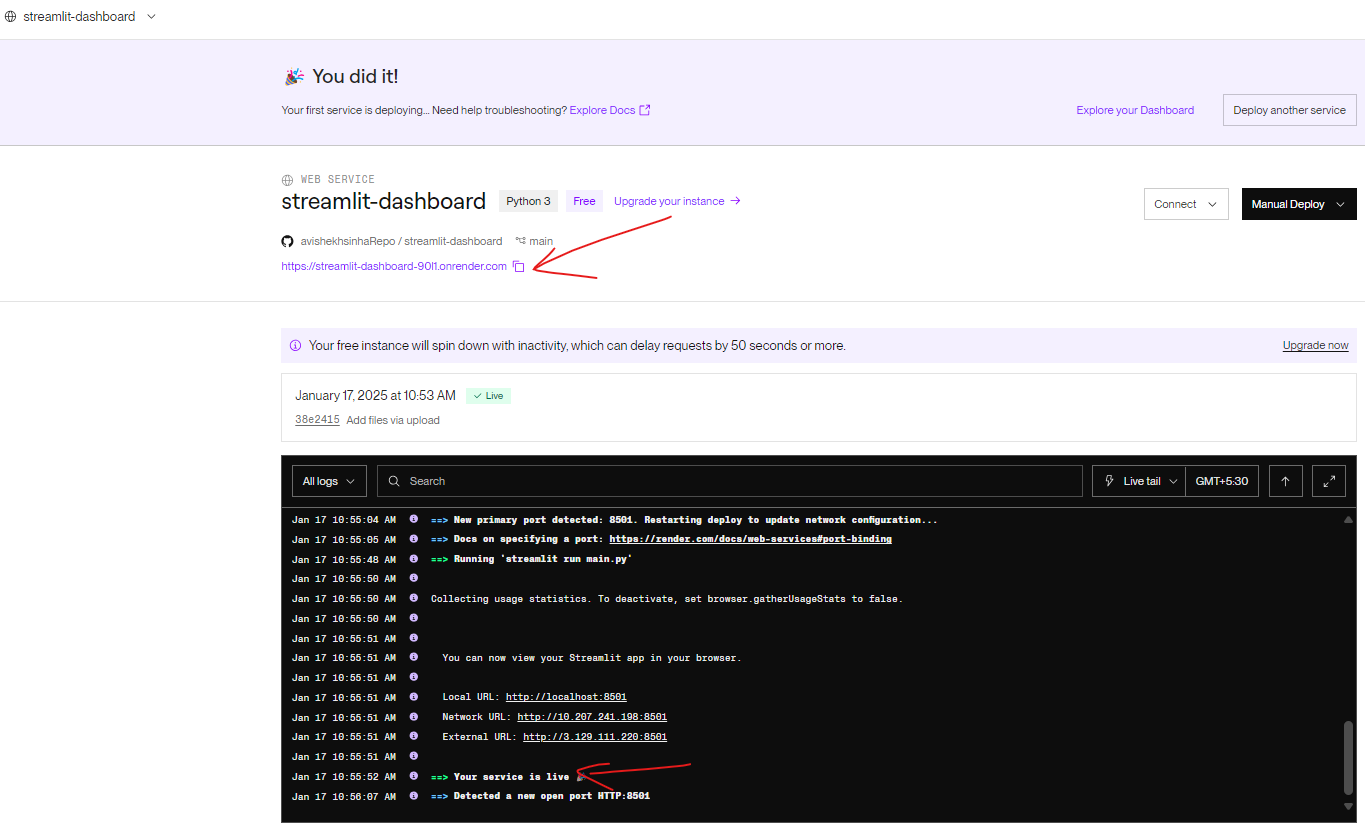
1. Specify the runtime of the app : e.g. **Python**
2. Specify the build command: **pip install -r requirements.txt**
3. Specify the start command: **streamlit run main.py**



1. Enter the deployment settings 🡪 Deploy Web Services

#### DEPLOYMENT LOGS

* Render will provide the public URL to access the application
* Example : [https://streamlit-dashboard-90l1.onrender.com/](https://streamlit-dashboard-90l1.onrender.com/%20)



### SETTING ENVIRONMENT VARIABLES