Building and hosting a DASH website

—

Data Science Project

Create the Dashboard in Python

1. app.py script

imports

```
import pandas as pd
import dash
from dash import dcc
from dash import html
from dash.dependencies import Input, Output
import plotly.express as px
```

1. app.py script

```
app = dash.Dash( __name __)
server = app.server
```

```
# LAYOUT:
```

```
app.layout = html.Div([
   html.H1("This is a test”),
   html.H3("This is a subheading”),
])
```

```
# CALLBACKS
```

```
if __name __ == '__main __':
  app.run_server(debug= True)
```

Adding Elements to the Dashboard

- HTML Elements for Layout: `"Div", "H1", "H2",..., "P","header", "footer",`

- Dash Core Components for inputs/interactive and graphs: `"datepicker", "dropdown", "graph", "highlight", "markdown", "mathjax", "slider","upload"`

Callbacks

- Callbacks make the Dashboard interactive
- Snippets of code, that are connected to the components in the Layout
- If the input element changes, the function of that callback is run
- A callback for each interactive element on the dashboard
- Definition of a sample Callback where data is plotted based on dropdown menu of feature selection

```
### Callback
@app.callback(
    [Output(component_id= 'output_container' , component_property= 'children' ),
     Output(component_id= 'full_line_single_graph' , component_property= 'figure' )],
     [Input(component_id= 'feature_selection' , component_property= 'value' )])
def update_graph(option_selection):
    '''Update the overall line graph'''
    container = f"The plot shows: {option_selection}"
    # build graph
    fig = px.line(
        data_frame = df,
        x = 'timeCode_x',
        y = option_selection,
        range_y=[0,100]
    )
    return container, fig
```

Graphs

- Use Plotly.Express Graph options:
- Check their documentation for code examples

- **Basics**: scatter, line, area, bar, funnel, timeline
- **Part-of-Whole**: pie, sunburst, treemap, icicle, funnel_area
- **1D Distributions**: histogram, box, violin, strip, ecdf
- **2D Distributions**: density_heatmap, density_contour
- **Matrix or Image Input**: imshow
- **3-Dimensional**: scatter_3d, line_3d
- **Multidimensional**: scatter_matrix, parallel_coordinates, parallel_categories
- **Tile Maps**: scatter_mapbox, line_mapbox, choropleth_mapbox, density_mapbox
- **Outline Maps**: scatter_geo, line_geo, choropleth
- **Polar Charts**: scatter_polar, line_polar, bar_polar
- **Ternary Charts**: scatter_ternary, line_ternary

- In the callback function filter the data based on the input and output the plotly express graph

https://plotly.com/python/plotly-express/

A quick guide

https://dash.plotly.com/tutorial

Create the Dashboard in Python

2. requirements.txt

a) create pipenv environment

pip install ***dash***

pip install ***plotly-express***

pip install ***pandas***

pip install gunicorn

b) create requirement.txt

pipenv lock -r > requirements.txt (old command)

pip freeze > requirements.txt (latest pip version)

Error while adding the requirements.txt

https://stackoverflow.com/questions/51845562/how-to-freeze-a-requirement-with-pipenv

116

As of v2022.8.13 of pipenv, the "old" `lock -r` functionality has been removed.

Going forward, this should be accomplished with:

```
pipenv requirements > requirements.txt
```

Share Improve this answer Follow

answered Aug 14, 2022 at 14:59

Collin Heist

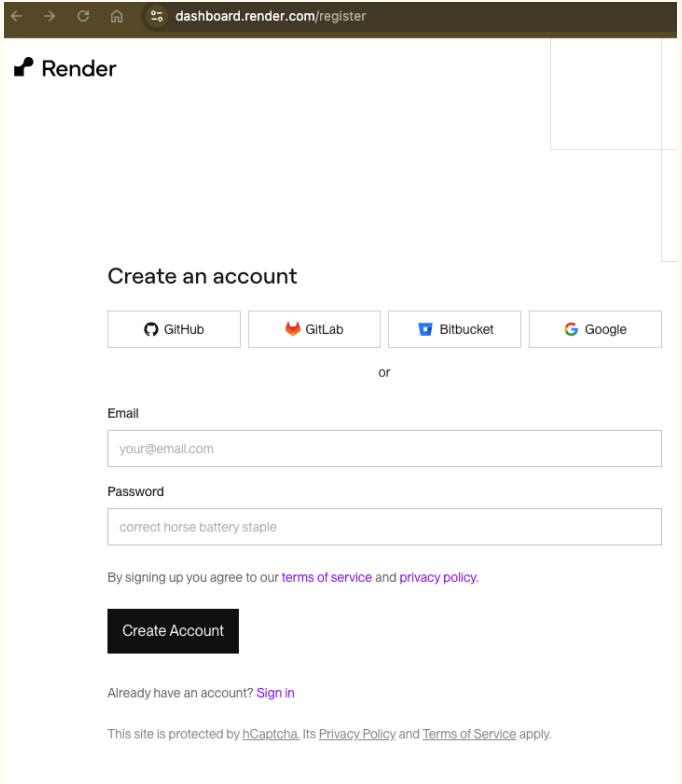
2,382 • 2 • 11 • 22

Push the Dashboard onto a Git Repo

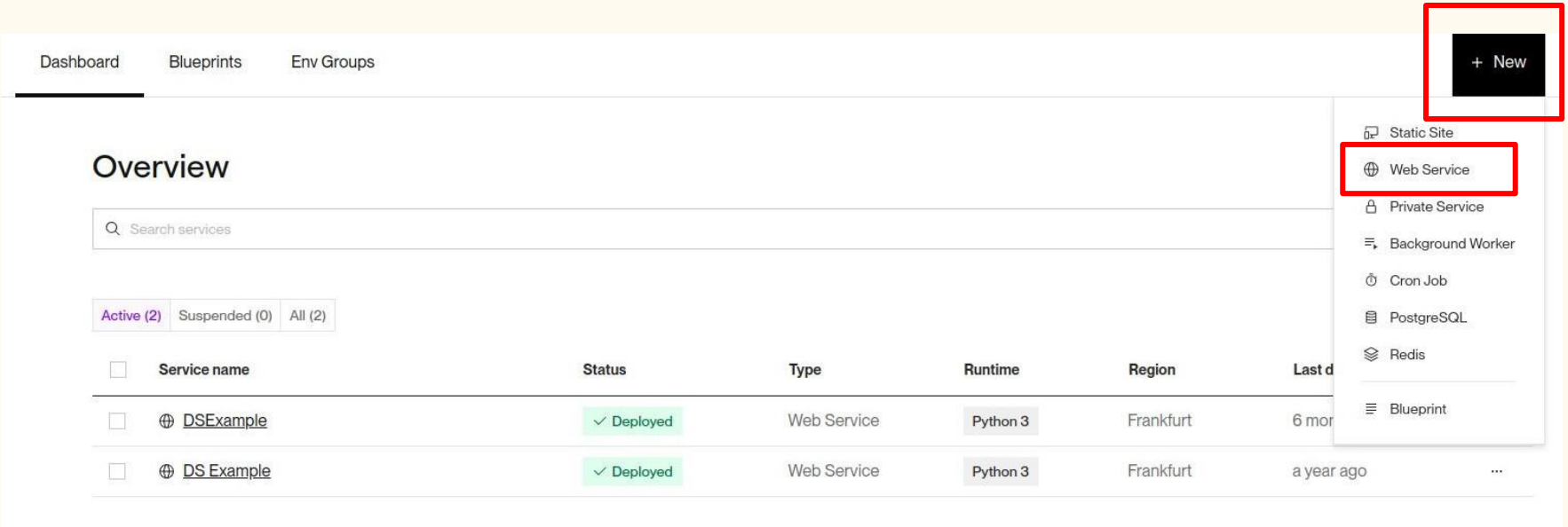
- University GitLab not compatible (neither cau nor informatic)
- Use GitLab or GitHub, can be a private repository
- If not d’accord with using these services, please reach out to your supervisor

Create a Render account

https://render.com



Start a new Web Service



Set up steps

1. Build and deploy from a Git repository
2. Connect to a public repository, or connect to private GitLab
3. Public Repository: Copy & Paste the URL of the repo (Make sure you are in the right branch!)

Render | Dashboard | Blueprints | Env Groups | + New

Source Code

miriskalt / DSExample Edit

Name
A unique name for your web service.

DSExample-SS-2024

Language

Python 3

Branch
The Git branch to build and deploy.

main

Region
Your services in the same region can communicate over a private network. You currently have services running in **Frankfurt**.

Frankfurt (EU Central) 2 existing services

Deploy in a new region +

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.

e.g. src

Build Command
Render runs this command to build your app before each deploy.

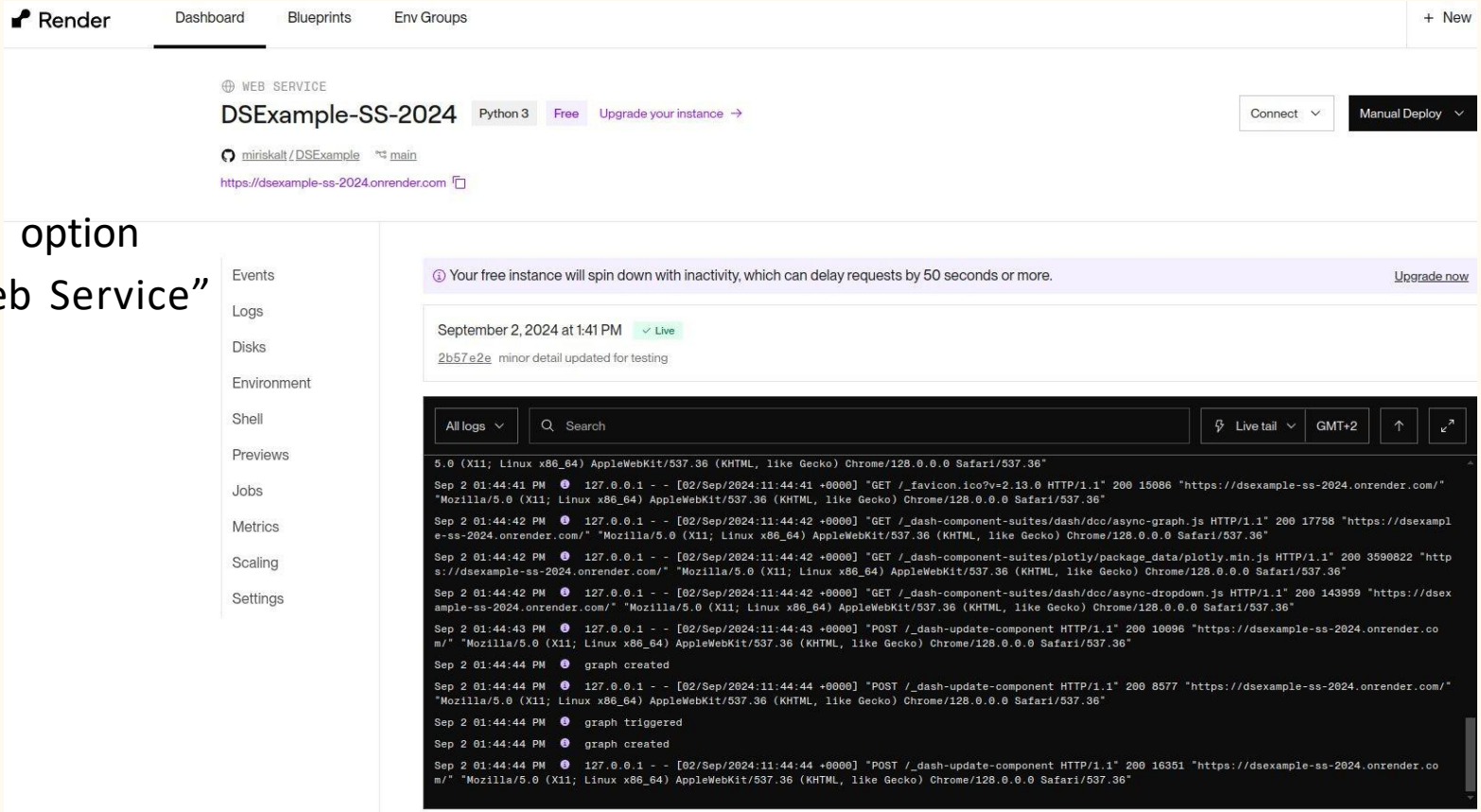
```
$ pip install -r requirements.txt
```

Start Command
Render runs this command to start your app with each deploy.

```
$ gunicorn app:server
```

Deploy

- Select Free option
- “Create Web Service”



If there are errors, look for…

Potential Troubleshooting:

- requirement.txt not included (`pipenv lock -r > requirements.txt`)

- server not defined in app.py script

```
app = dash.Dash(__name__)
server = app.server
```

- Dashboard files not in the main directory but in subfolder

  -> in setup of the web service, define the subfolder as root folder

- Set up and deploy with the basic dashboard and add elements step by step, remember to push and test the dashboard online

Try out an example at https://dsexample-ss-2024.onrender.com/

The sample dashboard can be cloned at: GitHub - miriskalt/DSExample

View previous Data Science Projects at: https://miriskalt.github.io/hall-of-fame/