

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](#)

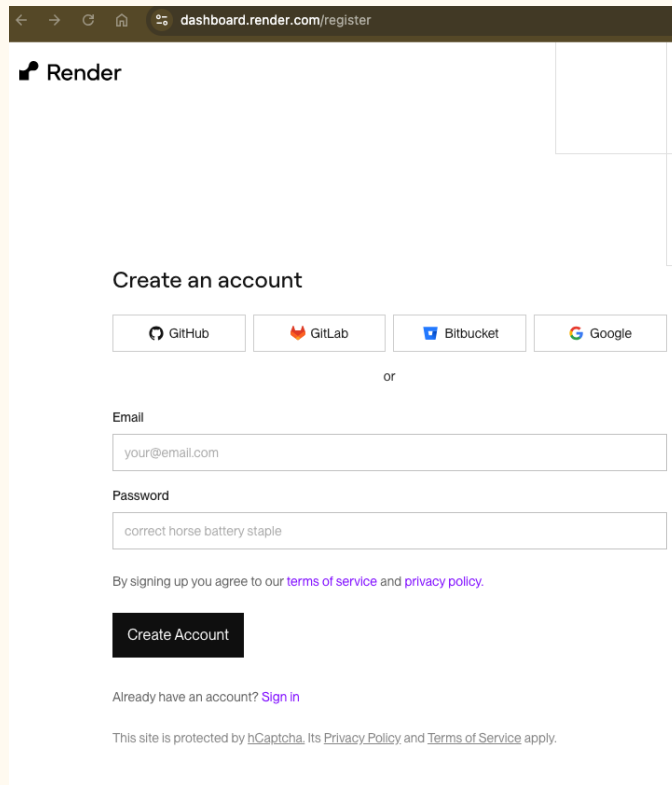
2,382 ● 2 ● 11 ● 22

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>



A screenshot of the Render website's registration page. The browser's address bar shows 'dashboard.render.com/register'. The page features the Render logo at the top left. The main heading is 'Create an account'. Below this, there are four buttons for social login: GitHub, GitLab, Bitbucket, and Google. A small 'or' text is centered below these buttons. The 'Email' field contains the placeholder text 'your@email.com'. The 'Password' field contains the placeholder text 'correct horse battery staple'. Below the password field, a line of text states: 'By signing up you agree to our [terms of service](#) and [privacy policy](#).' A black button with the text 'Create Account' is positioned below this. At the bottom, a link says 'Already have an account? [Sign in](#)'. The footer text reads: 'This site is protected by [hCaptcha](#). Its [Privacy Policy](#) and [Terms of Service](#) apply.'

Render

Create an account

 GitHub  GitLab  Bitbucket  Google

or

Email

your@email.com

Password

correct horse battery staple

By signing up you agree to our [terms of service](#) and [privacy policy](#).

Create Account

Already have an account? [Sign in](#)

This site is protected by [hCaptcha](#). Its [Privacy Policy](#) and [Terms of Service](#) apply.

Start a new Web Service

The screenshot shows the AWS Management Console 'Overview' page. At the top, there are tabs for 'Dashboard', 'Blueprints', and 'Env Groups'. Below the tabs is a search bar labeled 'Search services'. Under the search bar, there are filters for 'Active (2)', 'Suspended (0)', and 'All (2)'. A table lists the services, with columns for 'Service name', 'Status', 'Type', 'Runtime', 'Region', and 'Last d'. Two services are listed: 'DSEExample' and 'DS Example', both with a status of '✓ Deployed', type of 'Web Service', runtime of 'Python 3', and region of 'Frankfurt'. On the right side, a dropdown menu is open, showing options: 'Static Site', 'Web Service', 'Private Service', 'Background Worker', 'Cron Job', 'PostgreSQL', 'Redis', and 'Blueprint'. The '+ New' button at the top right of the dropdown is highlighted in a red box, and the 'Web Service' option is also highlighted in a red box.

Dashboard Blueprints Env Groups

Overview

Search services

Active (2) Suspended (0) All (2)

<input type="checkbox"/>	Service name	Status	Type	Runtime	Region	Last d
<input type="checkbox"/>	DSEExample	✓ Deployed	Web Service	Python 3	Frankfurt	6 mor
<input type="checkbox"/>	DS Example	✓ Deployed	Web Service	Python 3	Frankfurt	a year ago ...

+ New

- Static Site
- Web Service
- Private Service
- Background Worker
- Cron Job
- PostgreSQL
- Redis
- Blueprint

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 U R L of the repo (Make sure you are in the right branch!)



Source Code

miriskalt / DSEExample

Edit

Name

A unique name for your web service.

DSEExample-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”

The screenshot displays the Render dashboard for a project named "DSEExample-SS-2024". The top navigation bar includes "Render", "Dashboard", "Blueprints", "Env Groups", and a "+ New" button. The project name "DSEExample-SS-2024" is prominently displayed, along with "Python 3", "Free", and an "Upgrade your instance" link. A "Connect" button and a "Manual Deploy" button are also visible. Below the project name, the repository "miriskalt / DSEExample" and the branch "main" are shown, along with the URL "https://dsexample-ss-2024.onrender.com".

A sidebar on the left lists various management options: Events, Logs, Disks, Environment, Shell, Previews, Jobs, Metrics, Scaling, and Settings. The "Logs" section is currently selected.

The main content area shows a message: "Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more." with an "Upgrade now" link. Below this, a log entry for "September 2, 2024 at 1:41 PM" is shown, marked as "Live". The log entry includes a link "2b57e2e" and a note "minor detail updated for testing".

At the bottom, a terminal window displays the application logs. The logs show the following sequence of events:

- 6.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:41 PM 127.0.0.1 - - [02/Sep/2024:11:44:41 +0000] "GET /_favicon.ico?v=2.13.0 HTTP/1.1" 200 15086 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:42 PM 127.0.0.1 - - [02/Sep/2024:11:44:42 +0000] "GET /_dash-component-suites/dash/dcc/async-graph.js HTTP/1.1" 200 17758 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:42 PM 127.0.0.1 - - [02/Sep/2024:11:44:42 +0000] "GET /_dash-component-suites/plotly/package_data/plotly.min.js HTTP/1.1" 200 3590822 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:42 PM 127.0.0.1 - - [02/Sep/2024:11:44:42 +0000] "GET /_dash-component-suites/dash/dcc/async-dropdown.js HTTP/1.1" 200 143959 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:43 PM 127.0.0.1 - - [02/Sep/2024:11:44:43 +0000] "POST /_dash-update-component HTTP/1.1" 200 10096 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:44 PM graph created
- Sep 2 01:44:44 PM 127.0.0.1 - - [02/Sep/2024:11:44:44 +0000] "POST /_dash-update-component HTTP/1.1" 200 8577 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"
- Sep 2 01:44:44 PM graph triggered
- Sep 2 01:44:44 PM graph created
- Sep 2 01:44:44 PM 127.0.0.1 - - [02/Sep/2024:11:44:44 +0000] "POST /_dash-update-component HTTP/1.1" 200 16351 "https://dsexample-ss-2024.onrender.com/" "Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/128.0.0.0 Safari/537.36"

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/>