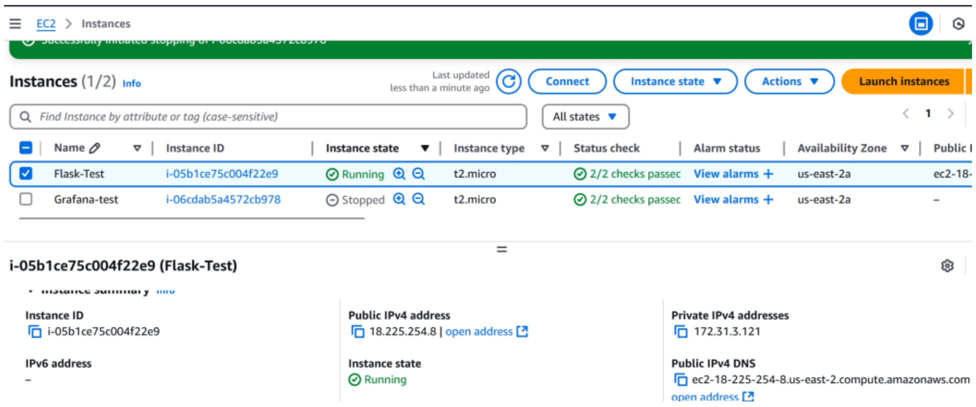
CSE 4939W Week 12 (Spring 2025) - Sage Pia

2/17/2025

This week, I researched ways to deploy a Flask application in the cloud. Last week, I was only running Flask and Grafana locally. Because we want to deploy our interface in such a way that any user can access it, we need to find a way to host the app online. I initially tried using Heroku, but I soon moved on after realizing there was no free version. Afterward, I researched the capabilities of AWS (Amazon Web Services).

AWS has a large suite of tools for computing, machine learning, database development, internet-of-things applications, and much more. Importantly, AWS also provides some free services that could host our app. To start, I found a tutorial for setting up a basic Flask application with Amazon EC2 (Elastic Cloud Compute) virtual servers.

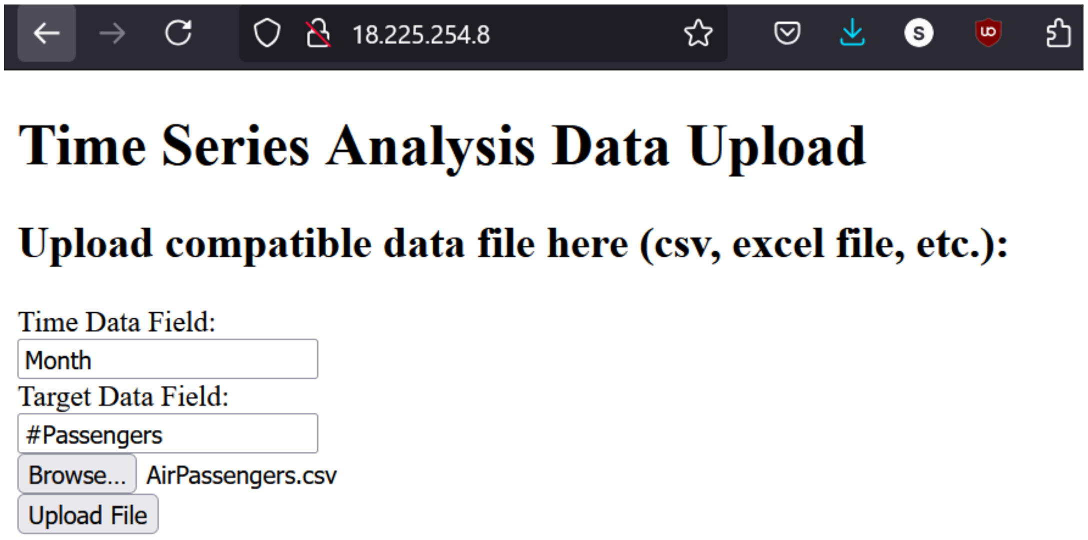


Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
Flask-Test	i-05b1ce75c004f22e9	Running	t2.micro	2/2 checks passec	View alarms +	us-east-2a	ec2-18-
Grafana-test	i-06cdab5a4572cb978	Stopped	t2.micro	2/2 checks passec	View alarms +	us-east-2a	–

i-05b1ce75c004f22e9 (Flask-Test)

Instance ID: i-05b1ce75c004f22e9
IPv6 address: –
Public IPv4 address: 18.225.254.8 | open address
Instance state: Running
Private IPv4 addresses: 172.31.3.121
Public IPv4 DNS: ec2-18-225-254-8.us-east-2.compute.amazonaws.com open address

After a fair amount of troubleshooting, I was able to get my application up and running. It can be accessed at the public IP address 18.225.254.8.

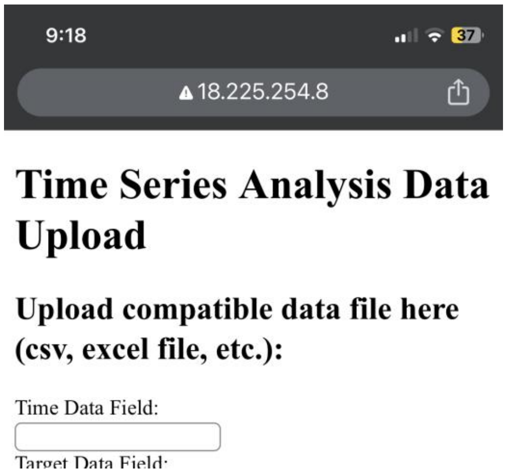


Time Series Analysis Data Upload

Upload compatible data file here (csv, excel file, etc.):

Time Data Field: Month
Target Data Field: #Passengers
Browse... AirPassengers.csv
Upload File

To confirm it was running online and not locally, I loaded the webpage from my phone.


9:18

18.225.254.8

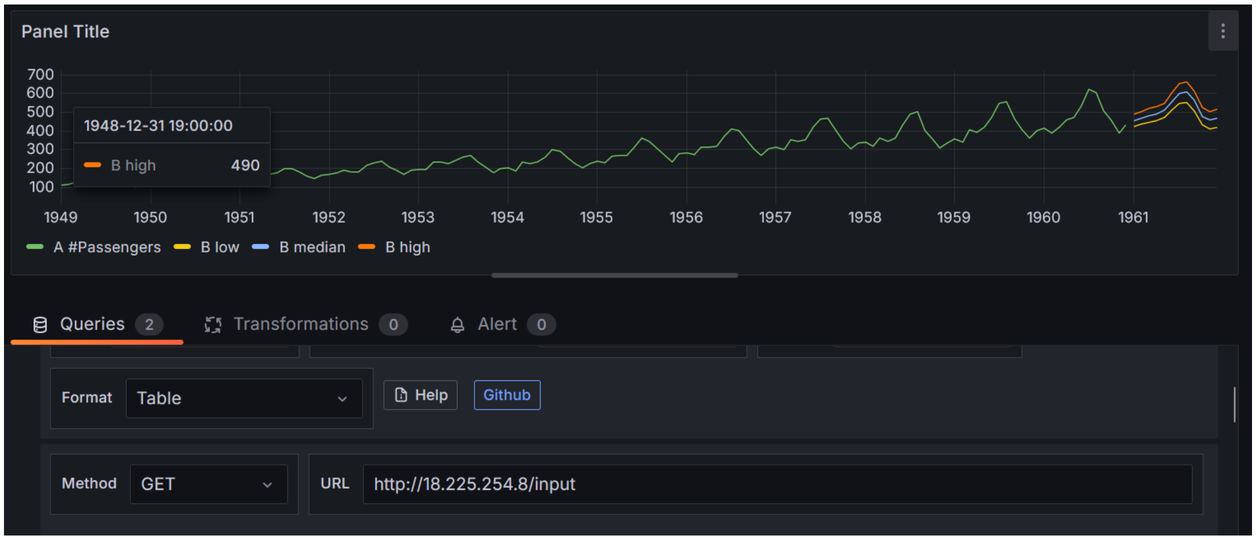
Time Series Analysis Data Upload

Upload compatible data file here (csv, excel file, etc.):

Time Data Field:

Target Data Field:

In the web version of Grafana, I pulled data from the Flask application. This proves that we can use AWS as a platform for posting and grabbing time series data. Note that in this example, I was using precomputed results. I was not able to run any Chronos tasks through EC2.

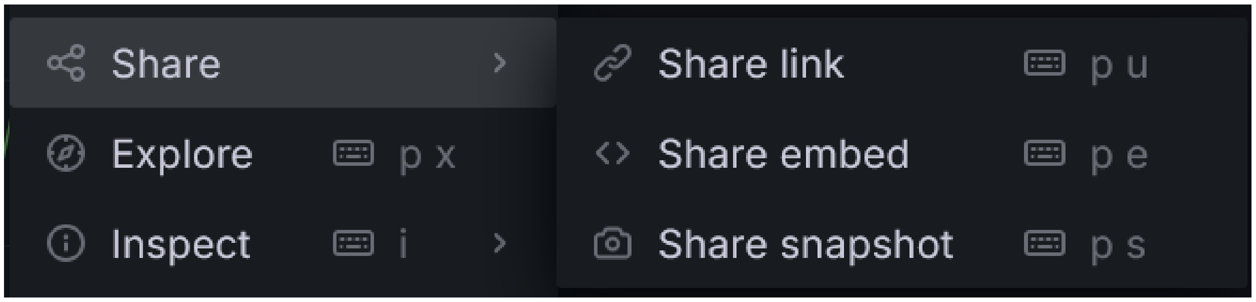


Format Table

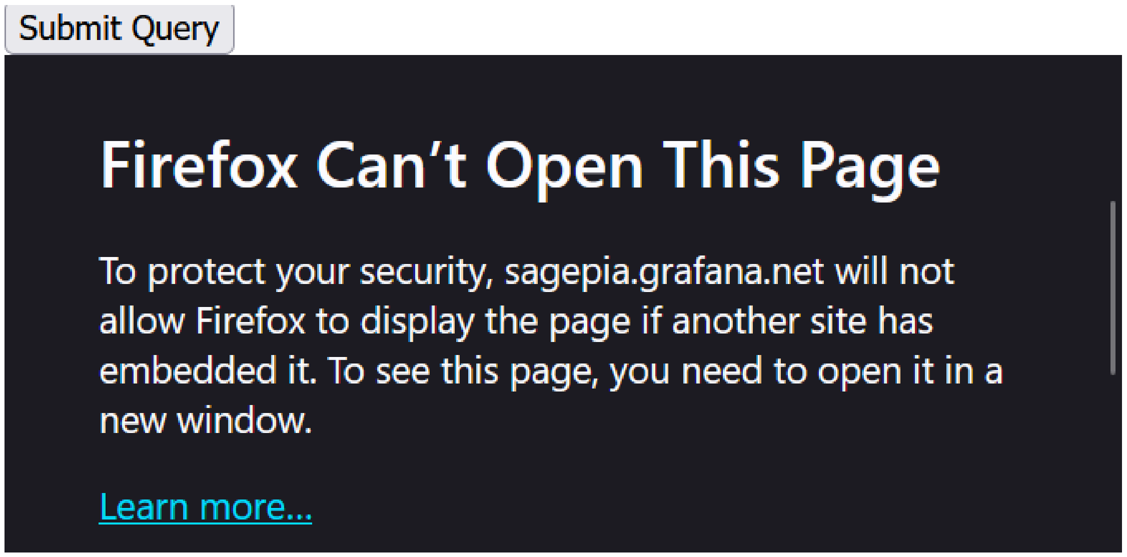
Method GET

URL http://18.225.254.8/input

Then, I looked at options for sharing Grafana dashboards: link, embed, and snapshot.



Of the three options, shared links appear to require a Grafana account for access. Since we likely would not expect users of our application to have a Grafana account, this is useless. The embed option generates HTML code to display on a webpage. When I tested it locally, I was met with a warning message, so I am unsure how viable this setting is.


Submit Query

Firefox Can't Open This Page

To protect your security, sagepia.grafana.net will not allow Firefox to display the page if another site has embedded it. To see this page, you need to open it in a new window.

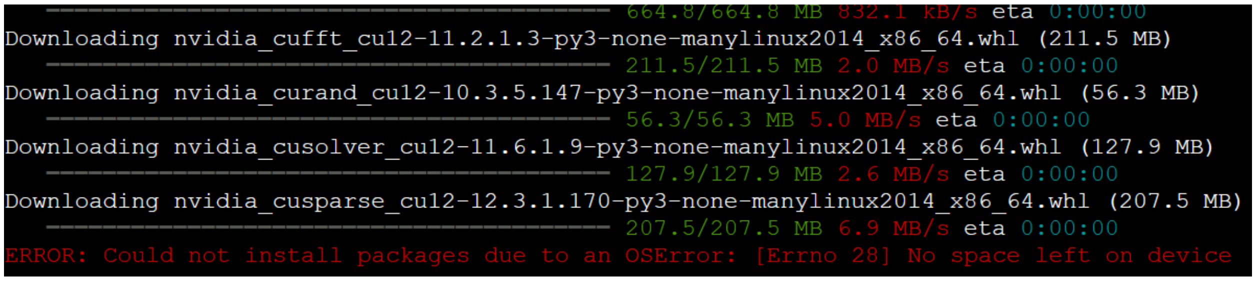
Learn more...

Lastly, the snapshot mode is the most straightforward. It simply generates a link to a static display of the dashboard. Users cannot modify the graph in any way, and Grafana accounts are not required. Something to note is that the refresh button does not seem to update the data at all. The two following snapshots were generated from the same dashboard at different times, yet refreshing does not update the earlier display to match the later one.

- Earlier snapshot
- Later snapshot

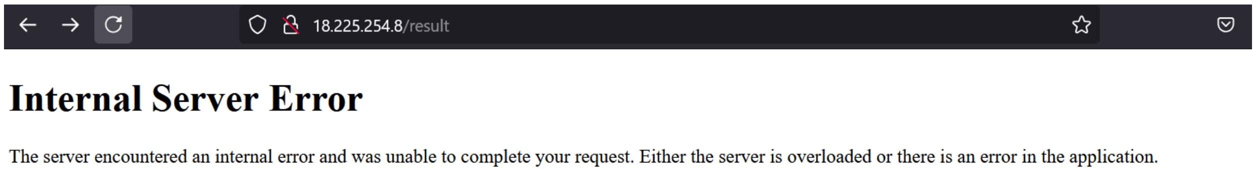
This property is useful because even if the dashboard updates with data from a new user, previously generated snapshots will stay intact. The next frontend challenge is figuring out some way to automatically create a snapshot.

Despite AWS's potential, we may need to find another place to compute the forecasting. I tried installing the Chronos library on EC2, but the operation failed because there was not enough space. Given the storage and computational power required for forecasting, I am uncertain as to where we will run it.



```
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 664.8/664.8 MB 832.1 kB/s eta 0:00:00
Downloading nvidia_cufft_cu12-11.2.1.3-py3-none-manylinux2014_x86_64.whl (211.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 211.5/211.5 MB 2.0 MB/s eta 0:00:00
Downloading nvidia_curand_cu12-10.3.5.147-py3-none-manylinux2014_x86_64.whl (56.3 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.3/56.3 MB 5.0 MB/s eta 0:00:00
Downloading nvidia_cusolver_cu12-11.6.1.9-py3-none-manylinux2014_x86_64.whl (127.9 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 127.9/127.9 MB 2.6 MB/s eta 0:00:00
Downloading nvidia_cusparse_cu12-12.3.1.170-py3-none-manylinux2014_x86_64.whl (207.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.5/207.5 MB 6.9 MB/s eta 0:00:00
ERROR: Could not install packages due to an OSError: [Errno 28] No space left on device
```

Also, we will need to investigate AWS further, as some of my Flask tabs broke despite being fine on localhost.



Internal Server Error

The server encountered an internal error and was unable to complete your request. Either the server is overloaded or there is an error in the application.

References:

- https://www.youtube.com/watch?v=ct1GbTvgVNM
- https://github.com/yeshwanthlm/YouTube/blob/main/flask-on-aws-ec2.md
- https://www.youtube.com/watch?v=xXirbnUB3NU