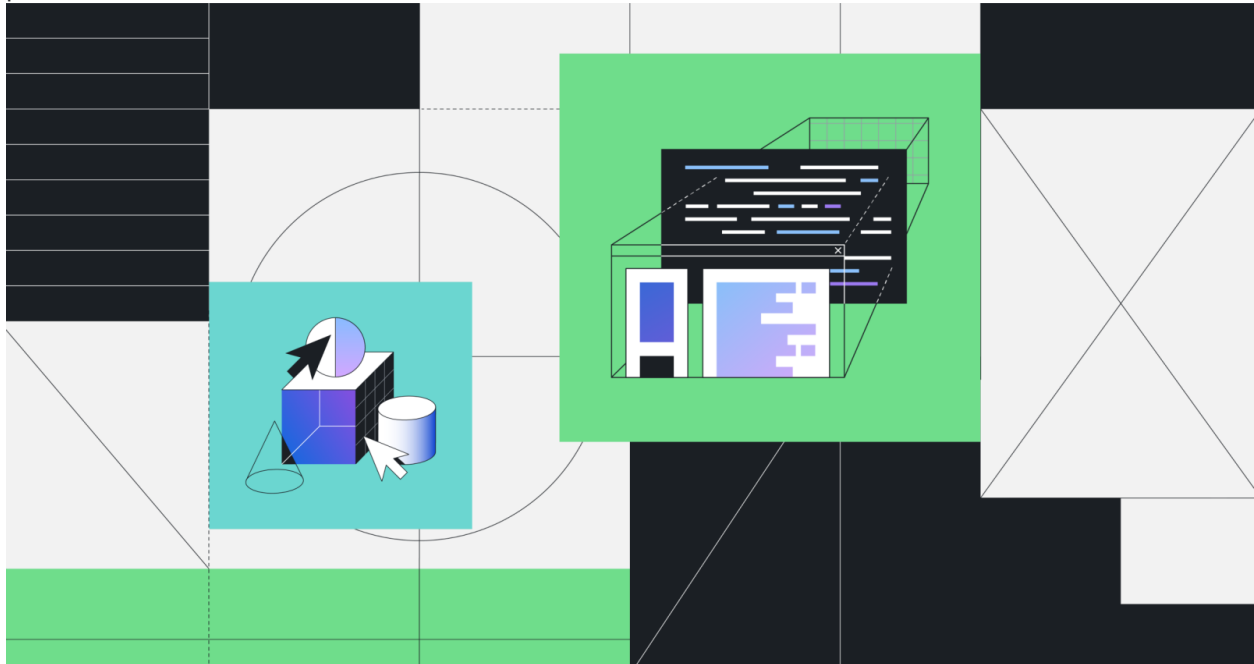


## Exciting new GitHub features powering machine learning seth juarez

Discover the exciting enhancements in GitHub that empower Machine Learning practitioners to do more.



I'm a huge fan of machine learning: as far as I'm concerned, it's an exciting way of creating software that combines the ingenuity of developers with the intelligence (sometimes hidden) in our data. Naturally, I store all my code in GitHub – but most of my work primarily happens on either my beefy desktop or some large VM in the cloud.

So I think it goes without saying, the [GitHub Universe announcements](#) made me super excited about building machine learning projects directly on GitHub. With that in mind, I thought I would try it out using one of my existing [machine learning repositories](#). Here's what I found.

### Jupyter Notebooks

Machine learning can be quite messy when it comes to the exploration phase. This process is made much easier by using Jupyter notebooks. With notebooks you can try several ideas with different data and model shapes quite easily. The challenge for me, however, has been twofold: it's hard to have ideas away from my desk, and notebooks are notoriously difficult to manage when working with others (WHAT DID YOU DO TO MY NOTEBOOK?!?!?).

sethjuarez / tlaloc

Search Type to search

<> Code Issues Pull requests 1 Actions Projects Wiki Security Insights Settings

main tlaloc / notebooks / generate.ipynb sethjuarez committed 1 hour ago History

21550 lines (21550 loc) · 556 KB

Code Blame Raw Copy Edit ...


```
In [8]: import warnings
warnings.filterwarnings("ignore")

import math
import numpy as np
import pandas as pd
from pathlib import Path
import plotly.express as px
from itertools import islice
import plotly.graph_objects as go
from typing import List, Generator
from datetime import datetime, timedelta

In [9]: def gen_gbm(period: float, start_amount: float, drift: float, volatility: float) -> Generator[float, None, None]:
    current_amt = start_amount
    i = 1
    while(True):
        c = (current_amt * drift * period) + \
            (current_amt * volatility * np.random.normal(0, math.sqrt(period))) + \
            math.cos(2 * math.pi * i * period) * .5
        yield current_amt + c
        current_amt += c
        i += 1

In [10]: fig = go.Figure()
fig.add_trace(go.Scatter(y=list(islice(gen_gbm((1/365.), 0, .01, .6), 365*3)), mode='lines', name='Actual'))
fig.update_layout(title=f'Geometric Brownian Motion (with superimposed period based scaled cosine wave)', xaxis_t=
```

This improved rendering experience is amazing (and there's a lovely dark mode too). In a recent pull-request I also noticed the following:


sethjuarez / tlaloc

<> Code
Issues
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Settings

## attempted smaller periodicity scaling (by half) #2

Open
sethjuarez wants to merge 1 commit into main from dev

Conversation 0
Commits 1
Checks 0
Files changed 1
+50 -150

Changes from all commits
File filter
Conversations
Jump to
0 / 1 files viewed
Review changes

200

notebooks/generate.ipynb

In [3]:

```

1 (...)
2 c = (current_amt * drift * period) + \
3   (current_amt * volatility * np.random.normal(0,
4   math.sqrt(period))) * \
5   - math.cos(2 * math.pi * i * period) + .5
6   yield current_amt + c
7   current_amt += c
8   i += 1

```

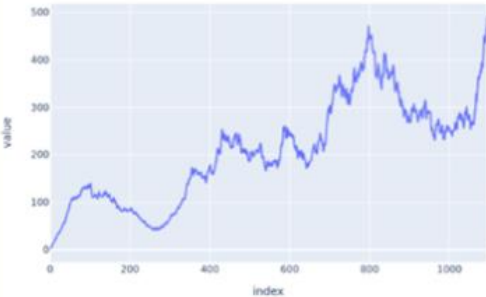
In [4]:

```

1 fig = go.Figure()
2 fig.add_trace(go.Scatter(y=list(islice(gen_gbm((1/365.), 0, .01, .6), 365*3)), mode='lines', name='Actual'))
3 fig.update_layout(title=f'Geometric Brownian Motion (added scaled cosine with periodicity)', xaxis_title='index', yaxis_title='value')
4 fig.show(renderer="png")

```

Output deleted



In [3]:

```

1 (...)
2 c = (current_amt * drift * period) + \
3   (current_amt * volatility * np.random.normal(0,
4   math.sqrt(period))) * \
5   + math.cos(2 * math.pi * i * period) + .25
6   yield current_amt + c
7   current_amt += c
8   i += 1

```

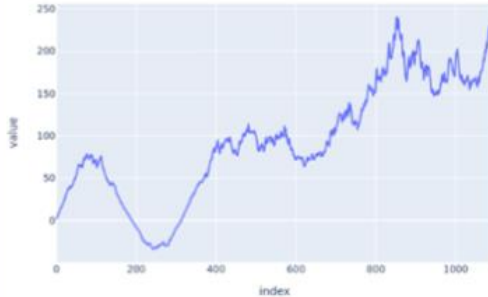
In [4]:

```

1 fig = go.Figure()
2 fig.add_trace(go.Scatter(y=list(islice(gen_gbm((1/365.), 0, .01, .6), 365*3)), mode='lines', name='Actual'))
3 fig.update_layout(title=f'Geometric Brownian Motion (added scaled cosine with periodicity)', xaxis_title='index', yaxis_title='value')
4 fig.show(renderer="png")

```

Output added



In [5]:

```

1 def generate_df(total: int, start_ams: List[float], drift: float, volatility: float,
2   end_date: datetime=datetime.now(), output_dir=None):
3   # current items
4   current_date = end_date - timedelta(days=total)
5

```

Not only can I see the cells that have been added, but I can also see side-by-side the code differences within the cells, as well as the literal outputs. I can see at a glance the code that has changed and the effect it produces thanks to NbDime running under the hood (shout out to the community for this awesome package).

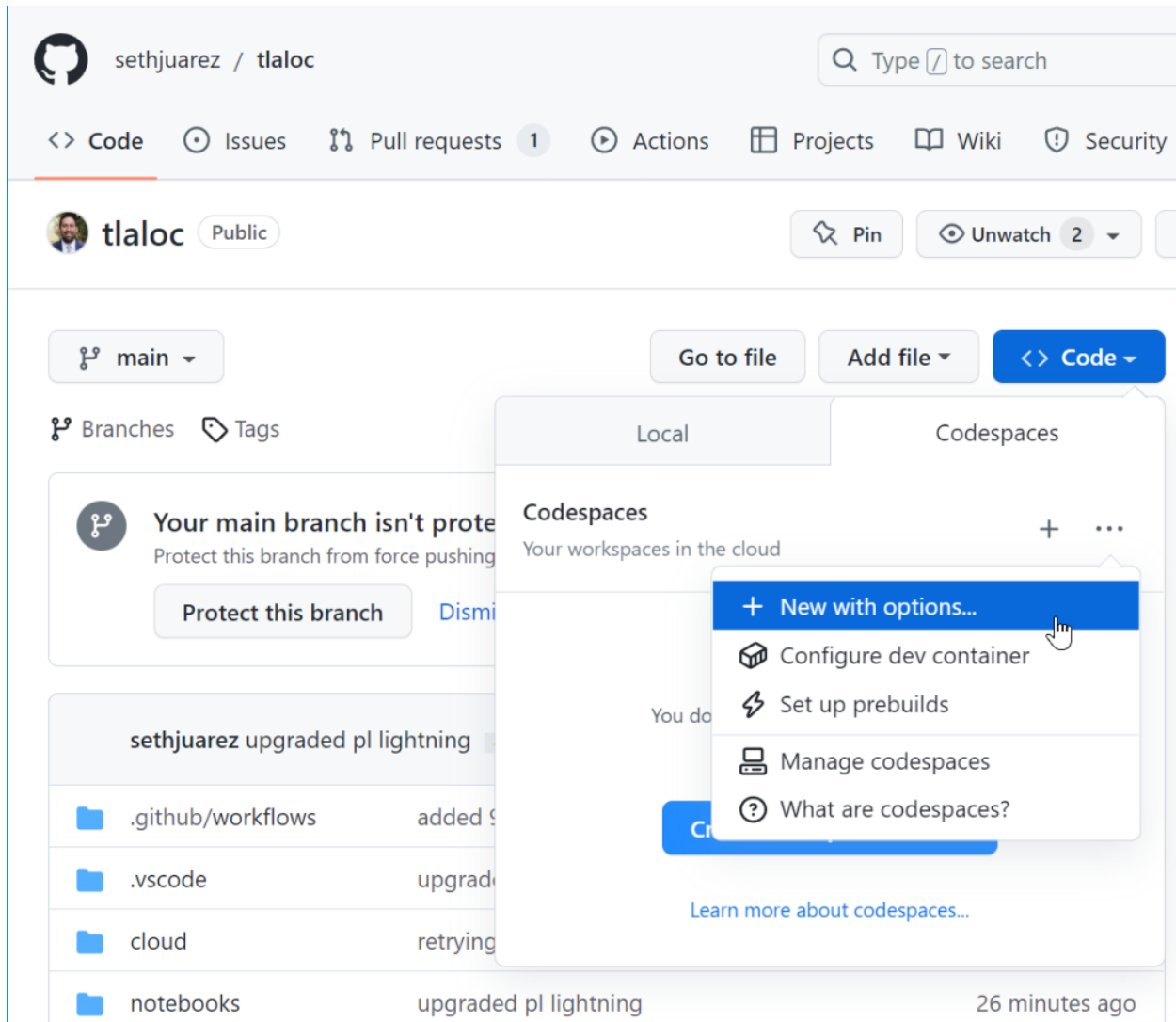
## Notebook Execution (and more)

While the rendering additions to GitHub are fantastic, there's still the issue of executing the things in a reliable way when I'm away from

my desk. Here's a couple of gems we [introduced at GitHub Universe](#) to make these issues go away:

1. GPUs for Codespaces
2. Zero-config notebooks in Codespaces
3. Edit your notebooks from VS Code, PyCharm, JupyterLab, on the web, or even using the [CLI](#) (powered by Codespaces)

I decided to try these things out for myself by opening an existing forecasting project that uses PyTorch to do time-series analysis. I dutifully created a new Codespace (but with options since I figured I would need to tell it to use a GPU).



Sure enough, there was a nice GPU option:

## Create codespace for sethjuarez/tlalloc

**Branch**  
This branch will be checked out on creation

**Region**  
Your codespace will run in the selected region

**Machine type**  
4-core • 8GB RAM • 32GB storage  
Need even more power? [Contact our team](#) to enable 32-  

✓ 4-core  
8GB RAM • 32GB

6-core (1 GPU)  
112GB RAM • 128GB

That was it! Codespaces found my requirements.txt file and went to work pip installing everything I needed.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE  JUPYTER

Detected following platforms:
  python: 3.10.4

Source directory      : /workspaces/tlalloc
Destination directory: /workspaces/tlalloc

Python Version: /opt/python/3.10.4/bin/python3.10
Creating directory for command manifest file if it does not exist
Removing existing manifest file

Running pip install...
```

After a few minutes (PyTorch is big) I wanted to check if the GPU worked (spoiler alert below):

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 1 AZURE JUPYTER
@sethjuarez → /workspaces/tlalloc (main) $ nvidia-smi
Thu Nov 3 19:13:41 2022

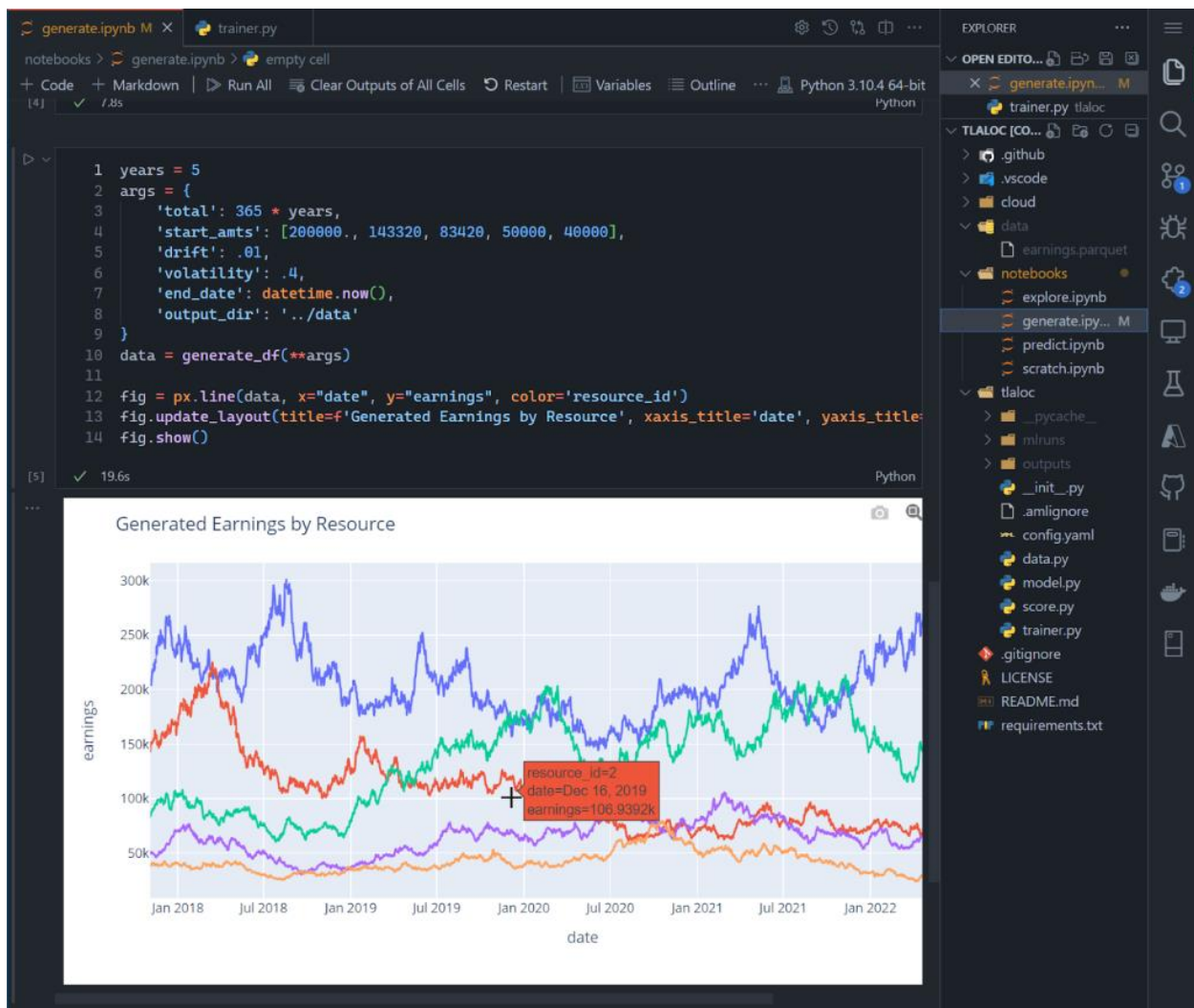
+-----+
| NVIDIA-SMI 515.65.01      Driver Version: 515.65.01      CUDA Version: 11.7      |
+-----+
| GPU   Name               Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf    Pwr:Usage/Cap|      Memory-Usage | GPU-Util  Compute M. |
|====+=====+====+=====+=====+=====+=====+=====+
|  0 Tesla V100-PCIE...    On          | 00000001:00:00:0 Off  |           0         |
| N/A   31C    P0      26W / 250W |  0MiB / 16384MiB |           0%      Default |
+-----+

+-----+
| Processes:                 |
| GPU   GI    CI          PID    Type    Process name                        GPU Memory |
| ID    ID    ID              |                   |           Usage |
+-----+
| No running processes found |
+-----+

@sethjuarez → /workspaces/tlalloc (main) $

```

This is incredible! And, the notebook also worked exactly as it does when working locally:





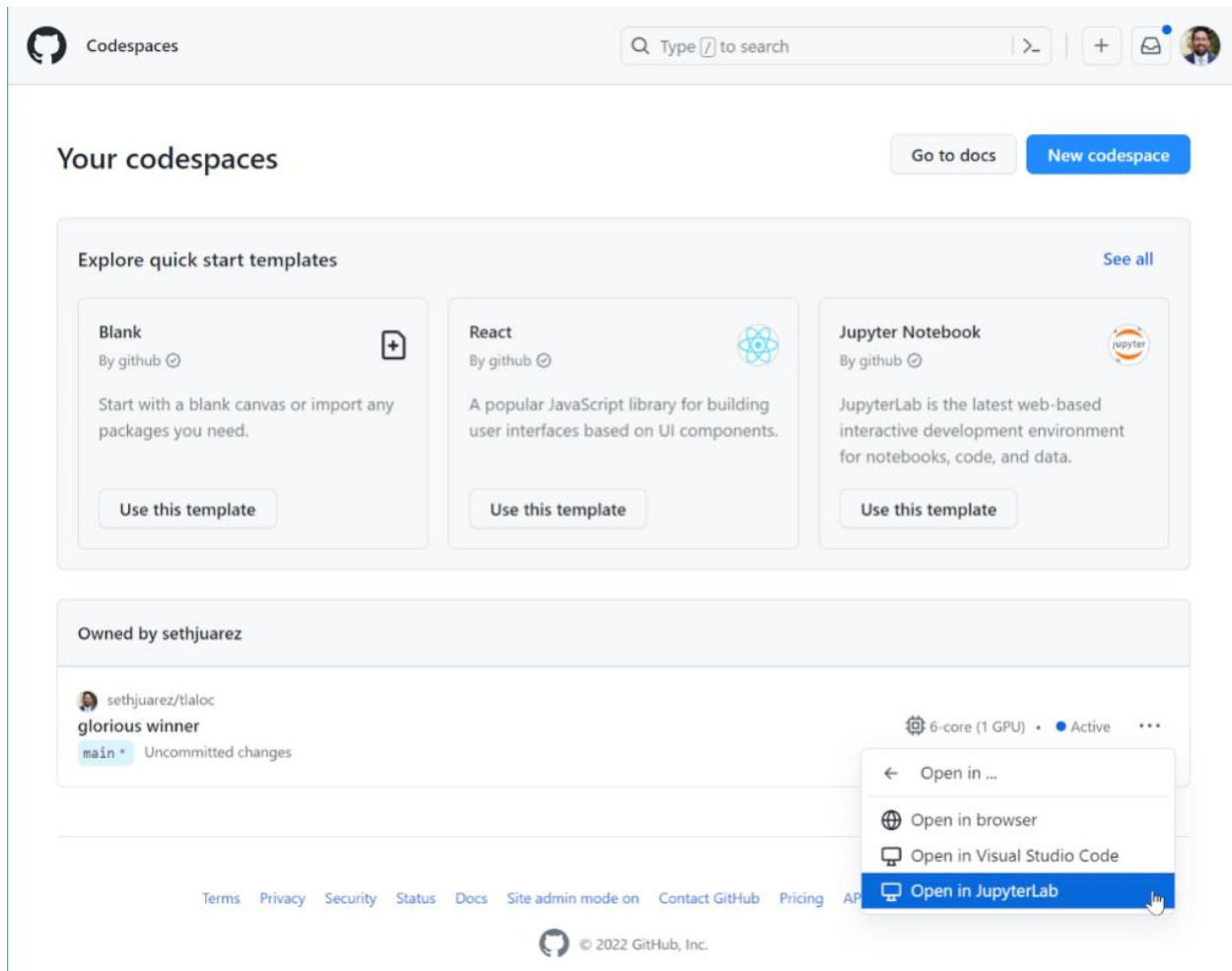
Again, this is in a browser! For kicks and giggles, I wanted to see if I could run the full blown model building process. For context, I believe notebooks are great for exploration but can become brittle when moving to repeatable processes. Eventually MLOps requires the movement of the salient code to their own scripts modules/scripts. In fact, it's how I structure all my ML projects. If you sneak a peek above, you will see a notebooks folder and then a folder that contains the model training Python files. As an avid VSCode user I also set up a way to debug the model building process. So I crossed my fingers and started the debugging process:

The screenshot displays the VS Code interface with the following components:

- Editor:** Shows a Python script named `trainer.py`. The script defines a `check_dir` function, an `EarningsCLI` class, and a `plot_simulation` method. The `plot_simulation` method is currently selected and highlighted.
- Output Panel:** Displays the execution output of the script. It shows a warning from `mlflow.utils.requirements_utils` regarding the local version of `torch` and `torchvision`. The output concludes with "Saving simulation to /workspaces/tlalloc/tlalloc/outputs/original\_data.png" and "Running Simulation...".
- Debug Console:** Shows the call stack for the `plot_simulation` method. The stack includes frames for `plot_simulation` (line 41), `after_fit` (line 120), and `<module>` (line 127). The `plot_simulation` frame is currently selected.
- Variables Panel:** Displays the local variables for the `plot_simulation` method. The variables include `sequence` (a `torch.Tensor`), `model` (an `EarningsGRUModel`), `output_dir` (a `PosixPath`), `p_seq` (a list), `self` (an `EarningsCLI` object), `sequence` (a `tensor`), and `window` (an integer).

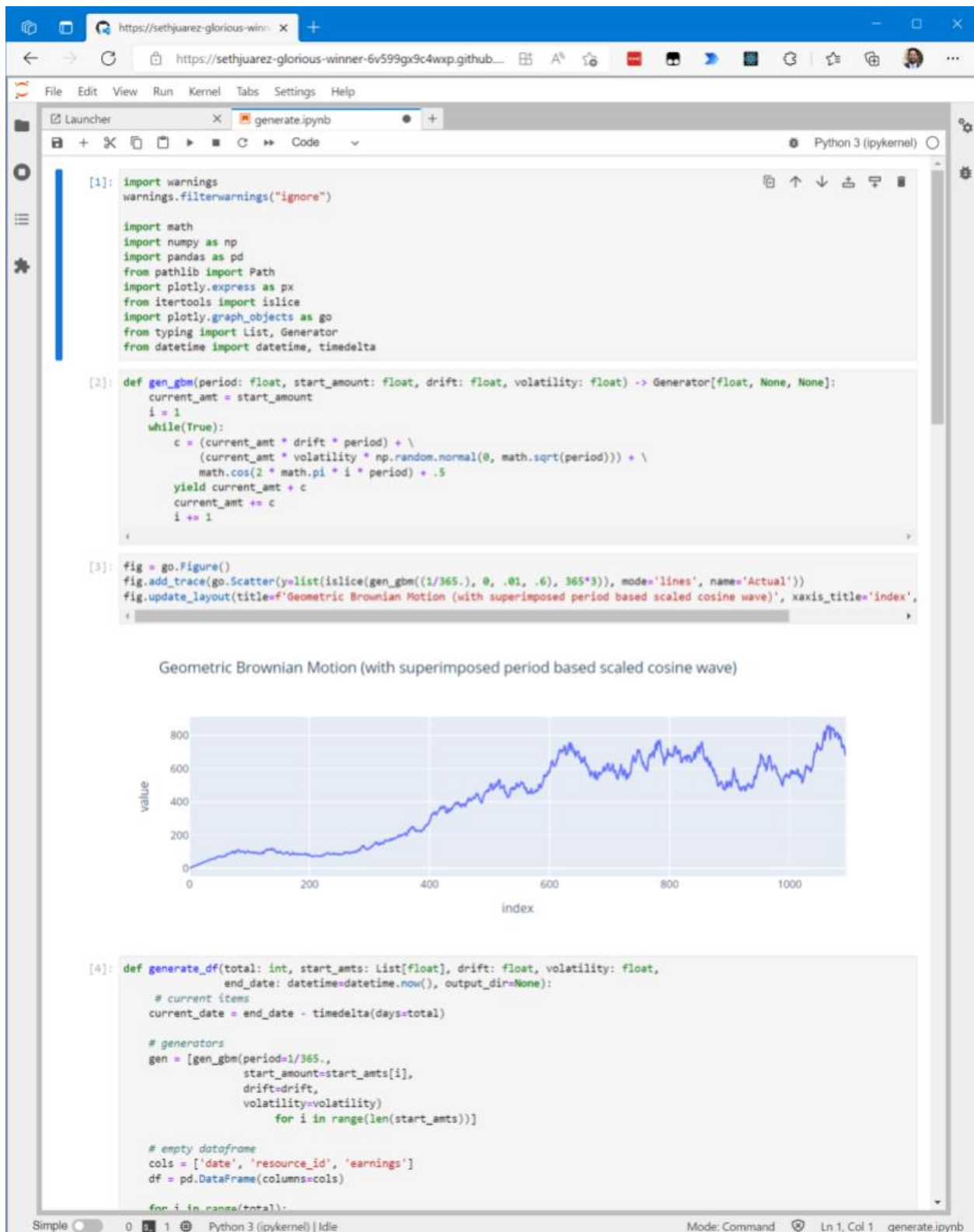
I know this is a giant screenshot, but I wanted to show the full gravity of what is happening in the browser: I am debugging the build of a deep learning PyTorch model – with breakpoints and everything – on a GPU.

The last thing I wanted to show is the new JupyterLab feature enabled via the CLI or directly from the Codespaces page:



For some, JupyterLab is an indispensable part of their ML process – which is why it's something we now support in its full glory:



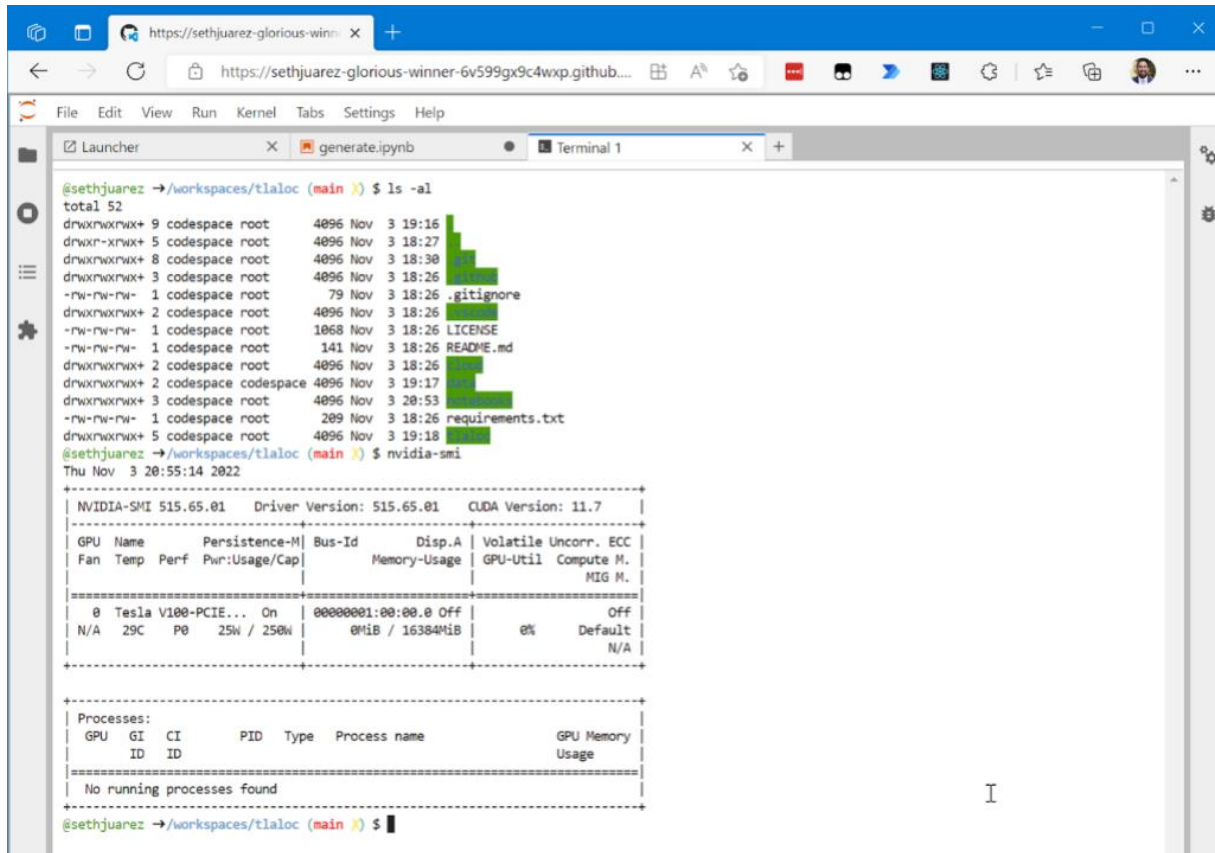


What if you're a JupyterLab user only and don't want to use the "Open In..." menu every time? There's a setting for that [here](#):

## Editor preference

- ☐ Visual Studio Code  
Connect to the cloud from your local desktop client. Requires Visual Studio Code with the GitHub Codespaces extension.
- ☒ Visual Studio Code for Web  
Edit and preview changes straight from the browser.
- ☐ JupyterLab  
Edit and run notebooks from the browser with JupyterLab.

And because there's always that one person who likes to do machine learning only from the command line (you know who I'm talking about):



```
@sethjuarez → /workspaces/tlaloc (main) $ ls -al
total 52
drwxrwxrwx+ 9 codespace root    4096 Nov  3 19:16 .
drwxr-xrwx+ 5 codespace root    4096 Nov  3 18:27 ..
drwxrwxrwx+ 8 codespace root    4096 Nov  3 18:30 .git
drwxrwxrwx+ 3 codespace root    4096 Nov  3 18:26 .gitignore
-rw-rw-rw-  1 codespace root      79 Nov  3 18:26 .gitignore
drwxrwxrwx+ 2 codespace root    4096 Nov  3 18:26 .vscode
-rw-rw-rw-  1 codespace root   1068 Nov  3 18:26 LICENSE
-rw-rw-rw-  1 codespace root    141 Nov  3 18:26 README.md
drwxrwxrwx+ 2 codespace root    4096 Nov  3 18:26 .vscode
drwxrwxrwx+ 2 codespace codespace 4096 Nov  3 19:17 .vscode
drwxrwxrwx+ 3 codespace root    4096 Nov  3 20:53 .vscode
-rw-rw-rw-  1 codespace root    209 Nov  3 18:26 requirements.txt
drwxrwxrwx+ 5 codespace root    4096 Nov  3 19:18 .vscode
@sethjuarez → /workspaces/tlaloc (main) $ nvidia-smi
Thu Nov  3 20:55:14 2022

+-----+
| NVIDIA-SMI 515.65.01 | Driver Version: 515.65.01 | CUDA Version: 11.7 |
+-----+
| GPU   Name           Persistence-M | Bus-Id  Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap |      0x00000001:00:00:0 Off |           |
|-----+-----+
| 0 Tesla V100-PCIE... On          | 00000001:00:00:0 Off |           |
| N/A   29C    P0      25W / 250W | 0MiB / 16384MiB |           |
+-----+-----+
| Processes: |
| GPU   GI   CI        PID   Type   Process name                      GPU Memory |
| ID    ID   ID                     |                       Usage            |
+-----+-----+
| No running processes found |
+-----+

@sethjuarez → /workspaces/tlaloc (main) $
```

For good measure I wanted to show you that given it's the same container, the GPU is still available.

Now, what if you want to just start up a notebook and try something? A File -> New Notebook experience is also available simply using this link: <https://codespace.new/jupyter>.

## Summary

Like I said earlier, I'm a huge fan of machine learning and GitHub. The fact that we're adding features to make the two better together is awesome. Now this might be a coincidence (I personally don't think so), but the container name selected by Codespaces for this little exercise sums up how this all makes me feel: *sethjuarez-glorious-winner* (seriously, look at container url).

Would love to hear your thoughts on these and any other features you think would make machine learning and GitHub better together. In the meantime, get ready for the upcoming GPU SKU launch by signing up to be on [waitlist](#). Until next time!