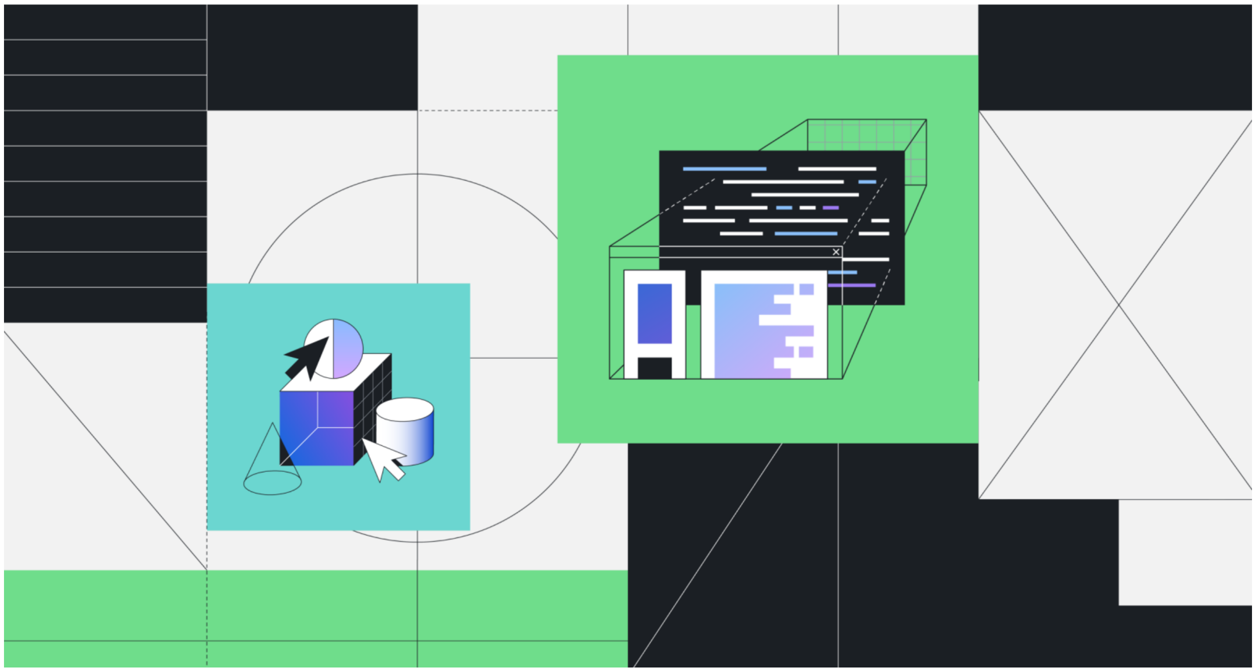
Exciting new GitHub features powering machine learning

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

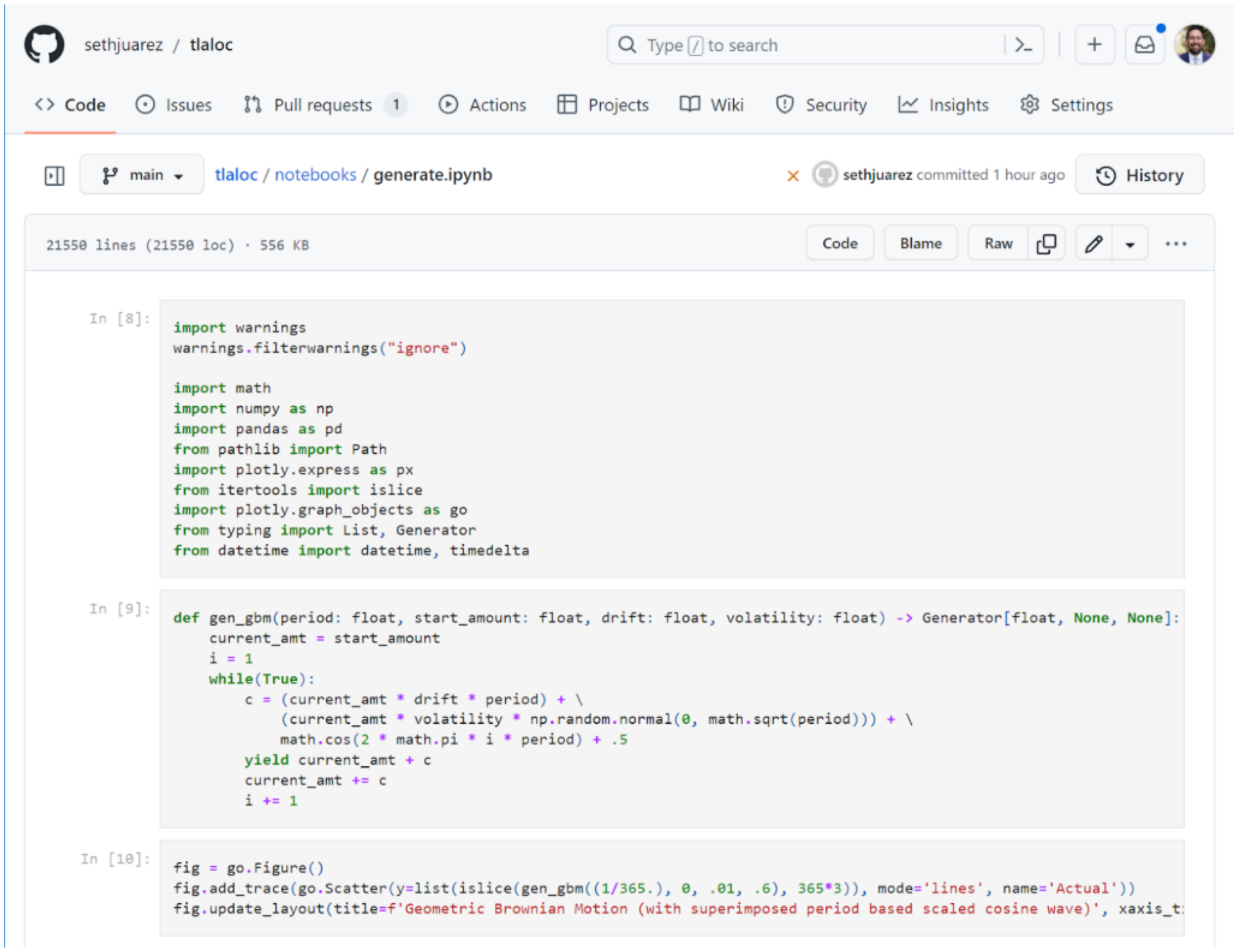


I'm a huge fan of machine learning: as far as I'm concerned, it's an exciting way of creating software that combines the ingenuity of developers with the intelligence (sometimes hidden) in our data. Naturally, I store all my code in GitHub – but most of my work primarily happens on either my beefy desktop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

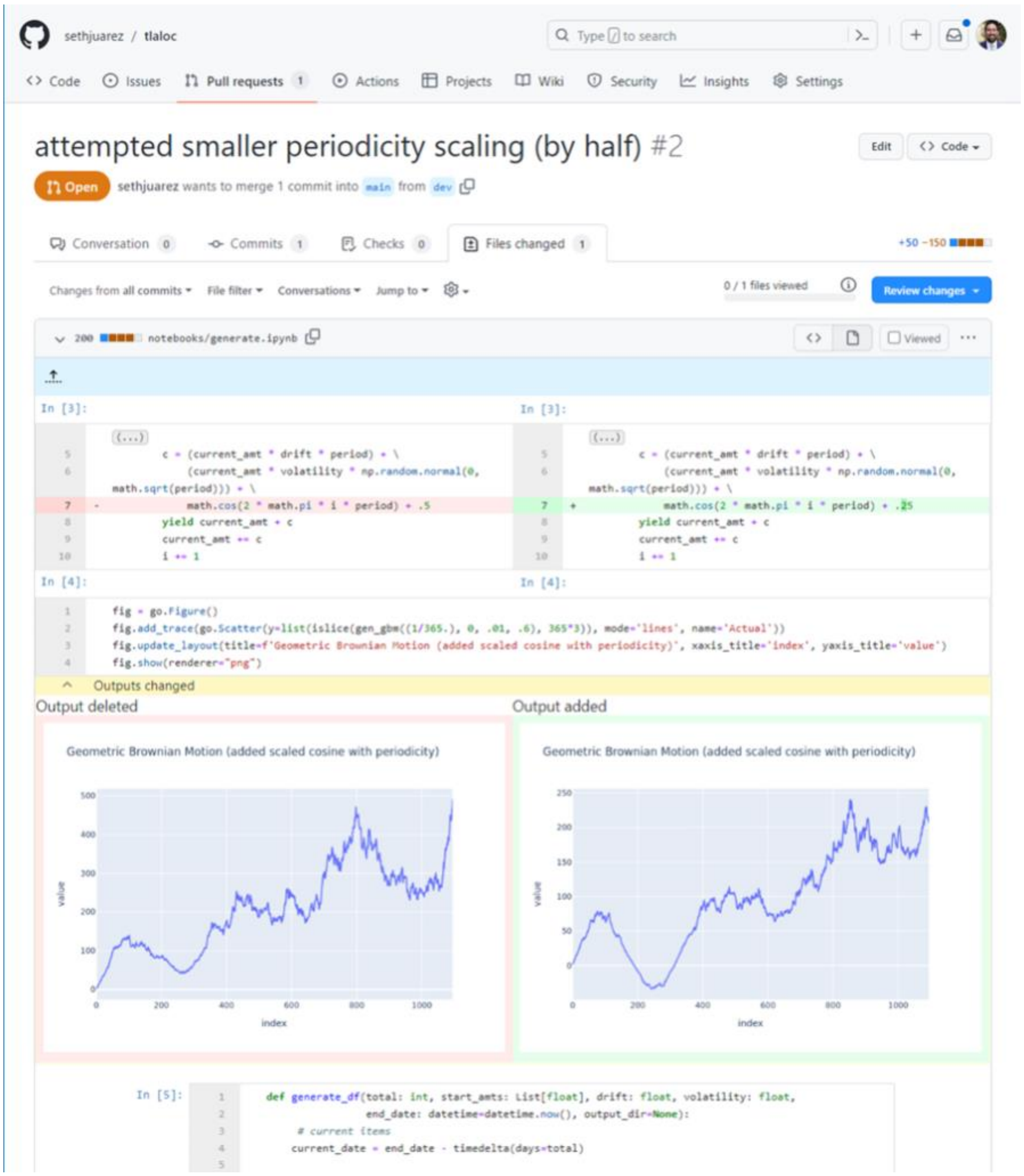


```
In [8]:
import warnings
warnings.filterwarnings("ignore")

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:



attempted smaller periodicity scaling (by half) #2

sethjuarez wants to merge 1 commit into main from dev

200 notebooks/generate.ipynb

In [3]:

```
(...)
        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 [3]:

```
(...)
        c = (current_amt * drift * period) + \
            (current_amt * volatility * np.random.normal(0,
math.sqrt(period))) + \
+               math.cos(2 * math.pi * i * period) + .25
        yield current_amt + c
        current_amt += c
        i += 1
```

In [4]: In [4]:

```
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 (added scaled cosine with periodicity)', xaxis_title='index', yaxis_title='value')
fig.show(renderer="png")
```

Outputs changed

Output deleted

Output added

In [5]:

```
def generate_df(total: int, start_amts: List[float], drift: float, volatility: float,
                end_date: datetime=datetime.now(), output_dir=None):
    # current items
    current_date = end_date - timedelta(days=total)

```

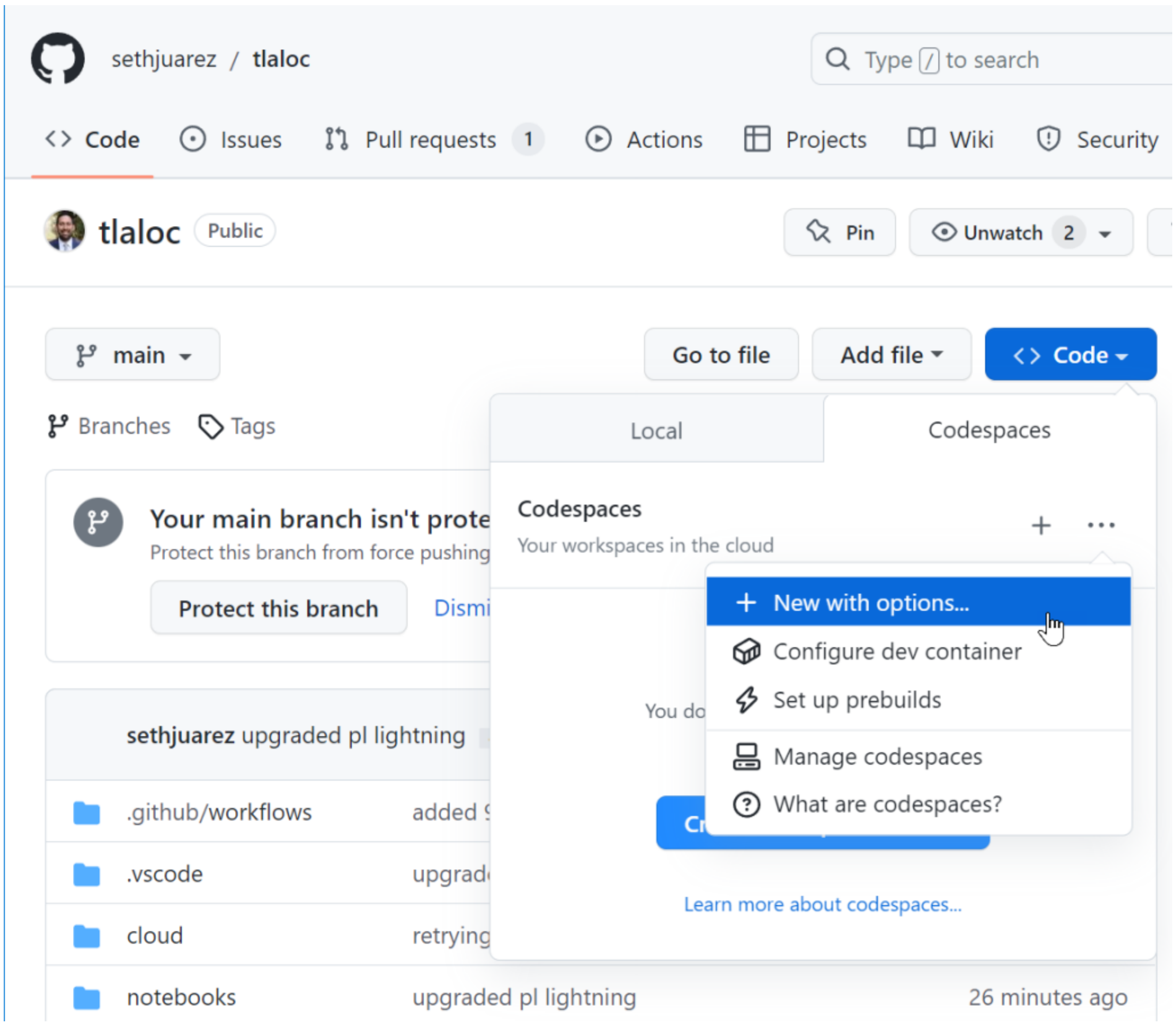
Not only can I see the cells that have been added, but I can also see side-by-side the code differences within the cells, as well as the literal outputs. I can see at a glance the code that has changed and the effect it produces thanks to NbDime running under the hood (shout out to the community for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

Notebook Execution (and more)

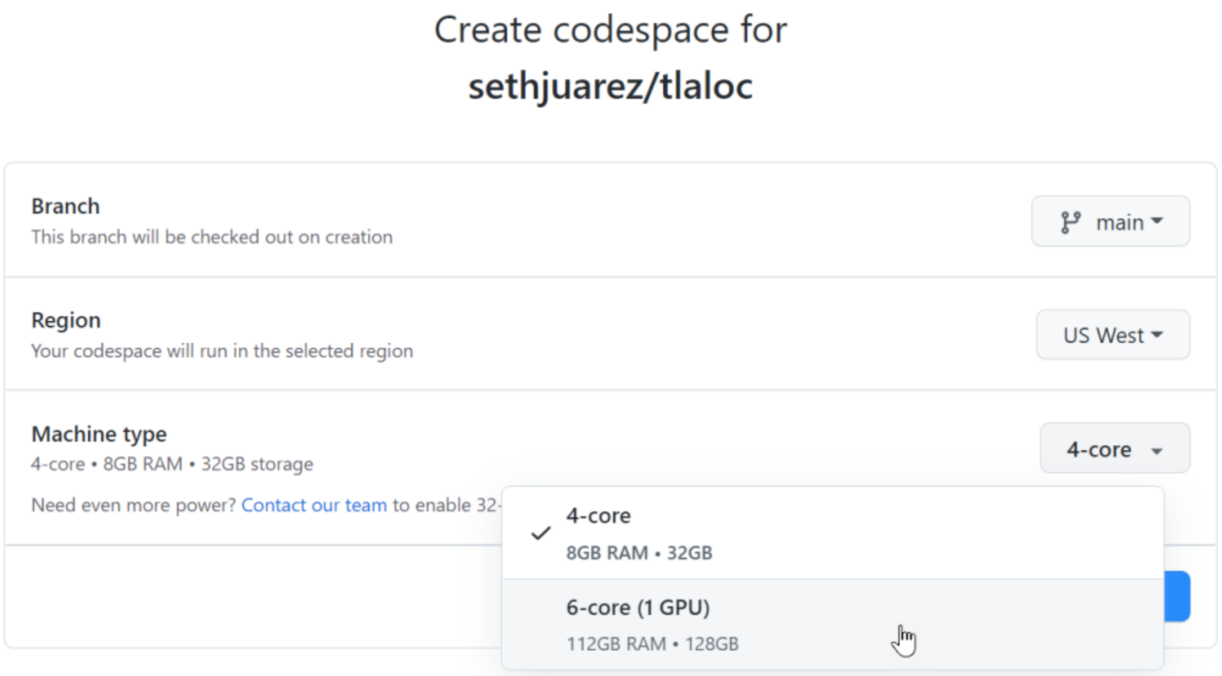
While the rendering additions to GitHub are fantastic, there’s still the issue of executing the things in a reliable way when I’m away from my desk. Here's a couple of gems we introduced at GitHub Universe to make these issues go away:

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/tlaloc

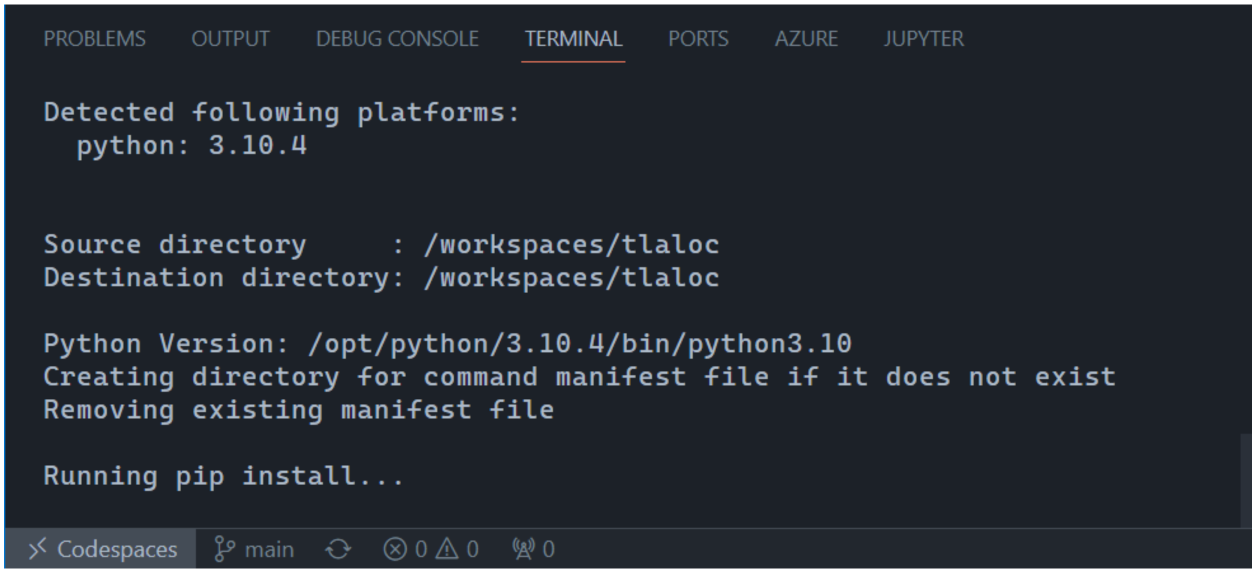
Branch
This branch will be checked out on creation — main

Region
Your codespace will run in the selected region — US West

Machine type
4-core • 8GB RAM • 32GB storage — 4-core

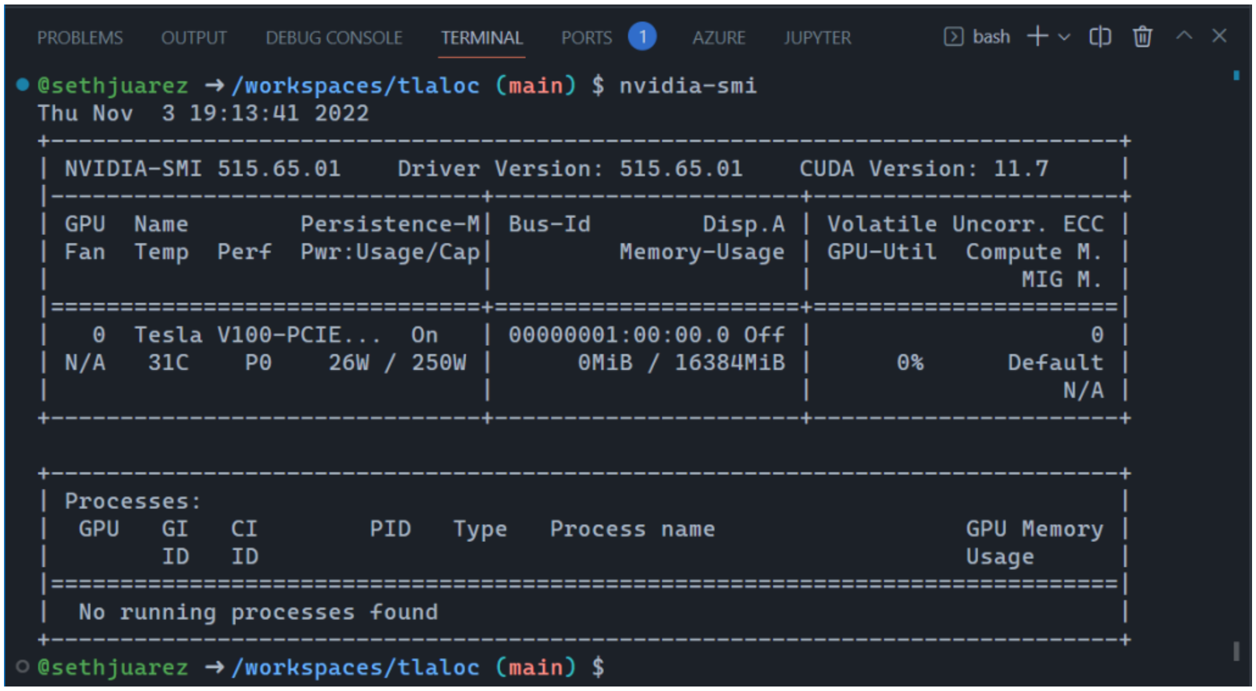
Need even more power? Contact our team to enable 32-

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



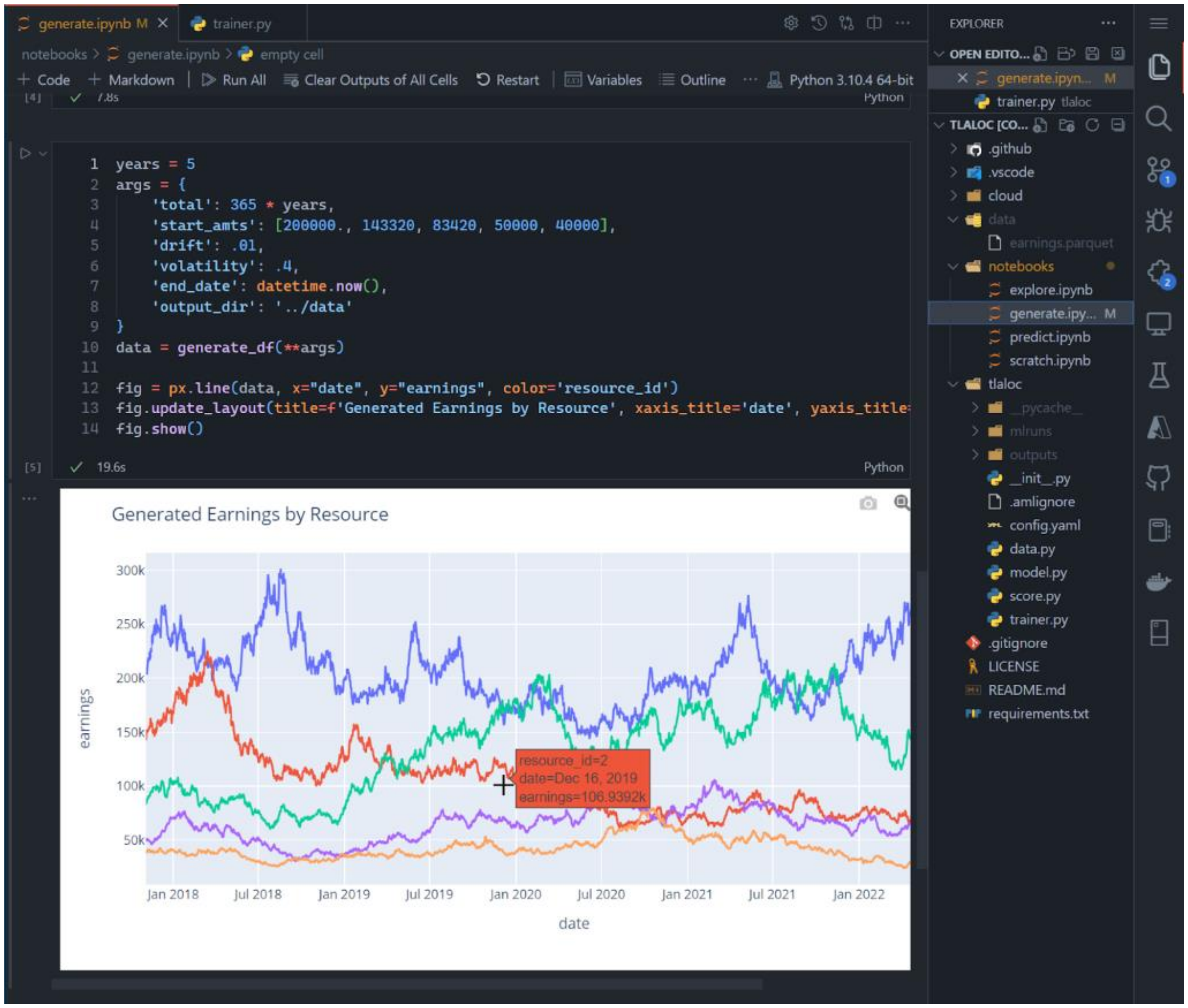
```
Detected following platforms:
  python: 3.10.4


Source directory      : /workspaces/tlaloc
Destination directory: /workspaces/tlaloc

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):



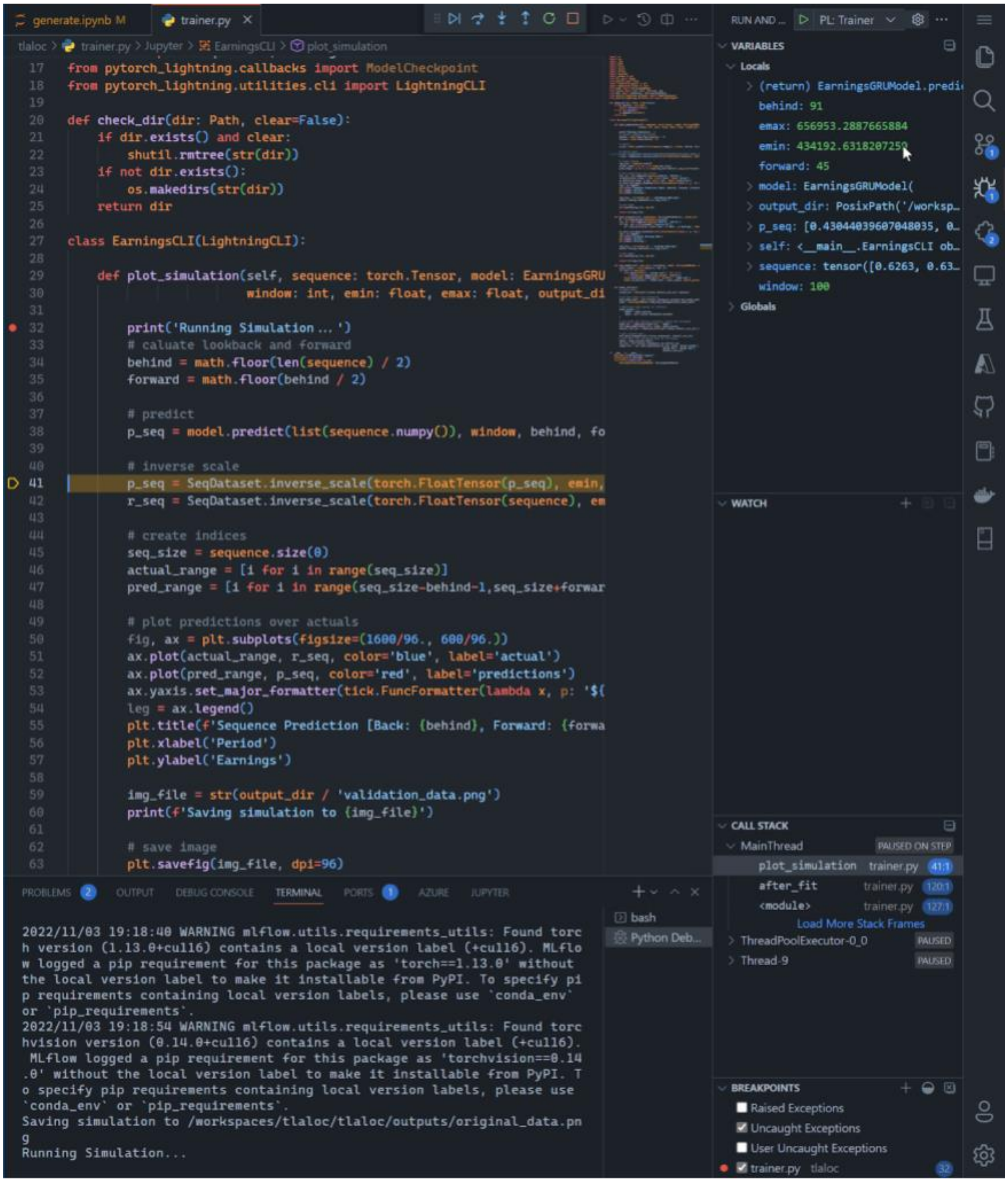
```
@sethjuarez ➜ /workspaces/tlaloc (main) $ nvidia-smi
Thu Nov  3 19:13:41 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 515.65.01    Driver Version: 515.65.01    CUDA Version: 11.7     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  On   | 00000001:00:00.0 Off |                    0 |
| N/A   31C    P0    26W / 250W |      0MiB / 16384MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
@sethjuarez ➜ /workspaces/tlaloc (main) $
```

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



```
years = 5
args = {
    'total': 365 * years,
    'start_amts': [200000., 143320, 83420, 50000, 40000],
    'drift': .01,
    'volatility': .4,
    'end_date': datetime.now(),
    'output_dir': '../data'
}
data = generate_df(**args)

fig = px.line(data, x="date", y="earnings", color='resource_id')
fig.update_layout(title=f'Generated Earnings by Resource', xaxis_title='date', yaxis_title=
fig.show()
```

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:



```
from pytorch_lightning.callbacks import ModelCheckpoint
from pytorch_lightning.utilities.cli import LightningCLI

def check_dir(dir: Path, clear=False):
    if dir.exists() and clear:
        shutil.rmtree(str(dir))
    if not dir.exists():
        os.makedirs(str(dir))
    return dir

class EarningsCLI(LightningCLI):

    def plot_simulation(self, sequence: torch.Tensor, model: EarningsGRU
                        window: int, emin: float, emax: float, output_di

        print('Running Simulation ... ')
        # caluate lookback and forward
        behind = math.floor(len(sequence) / 2)
        forward = math.floor(behind / 2)

        # predict
        p_seq = model.predict(list(sequence.numpy()), window, behind, fo

        # inverse scale
        p_seq = SeqDataset.inverse_scale(torch.FloatTensor(p_seq), emin,
        r_seq = SeqDataset.inverse_scale(torch.FloatTensor(sequence), em

        # create indices
        seq_size = sequence.size(0)
        actual_range = [i for i in range(seq_size)]
        pred_range = [i for i in range(seq_size-behind-1,seq_size+forwar

        # plot predictions over actuals
        fig, ax = plt.subplots(figsize=(1600/96., 600/96.))
        ax.plot(actual_range, r_seq, color='blue', label='actual')
        ax.plot(pred_range, p_seq, color='red', label='predictions')
        ax.yaxis.set_major_formatter(tick.FuncFormatter(lambda x, p: '${
        leg = ax.legend()
        plt.title(f'Sequence Prediction [Back: {behind}, Forward: {forwa
        plt.xlabel('Period')
        plt.ylabel('Earnings')

        img_file = str(output_dir / 'validation_data.png')
        print(f'Saving simulation to {img_file}')

        # save image
        plt.savefig(img_file, dpi=96)
```

```
> (return) EarningsGRUModel.predi
  behind: 91
  emax: 656953.2887665884
  emin: 434192.6318207250
  forward: 45
> model: EarningsGRUModel(
> output_dir: PosixPath('/worksp…
> p_seq: [0.43044039607048035, 0…
> self: <__main__.EarningsCLI ob…
> sequence: tensor([0.6263, 0.63…
  window: 100
```

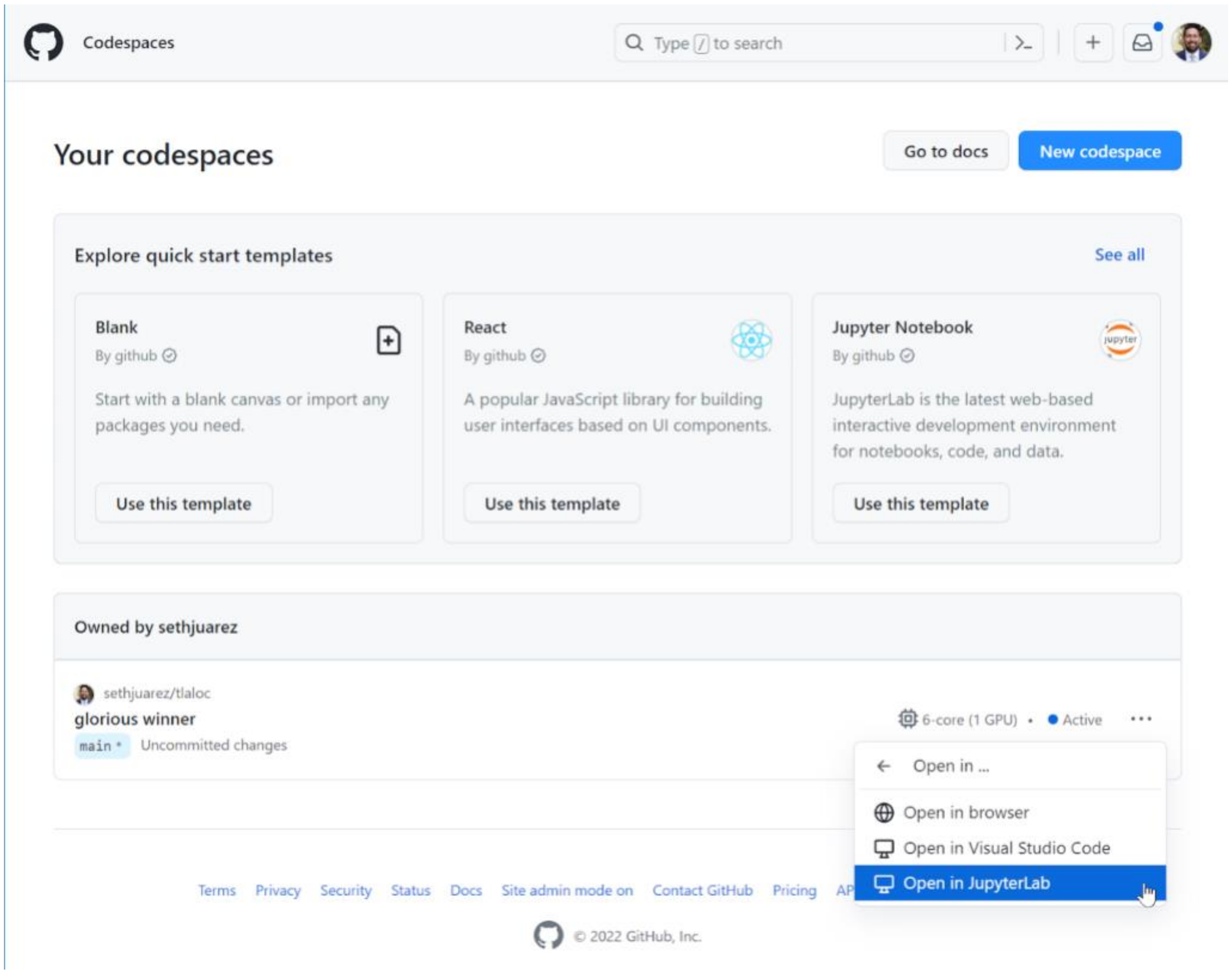
```
MainThread                      PAUSED ON STEP
    plot_simulation   trainer.py   41:1
    after_fit         trainer.py   120:1
    <module>          trainer.py   127:1
    Load More Stack Frames
ThreadPoolExecutor-0_0          PAUSED
Thread-9                        PAUSED
```

```
2022/11/03 19:18:40 WARNING mlflow.utils.requirements_utils: Found torc
h version (1.13.0+cu116) contains a local version label (+cu116). MLflo
w logged a pip requirement for this package as 'torch==1.13.0' without
the local version label to make it installable from PyPI. To specify pi
p requirements containing local version labels, please use `conda_env`
or `pip_requirements`.
2022/11/03 19:18:54 WARNING mlflow.utils.requirements_utils: Found torc
hvision version (0.14.0+cu116) contains a local version label (+cu116).
 MLflow logged a pip requirement for this package as 'torchvision==0.14
.0' without the local version label to make it installable from PyPI. T
o specify pip requirements containing local version labels, please use
`conda_env` or `pip_requirements`.
Saving simulation to /workspaces/tlaloc/tlaloc/outputs/original_data.pn
g
Running Simulation...
```

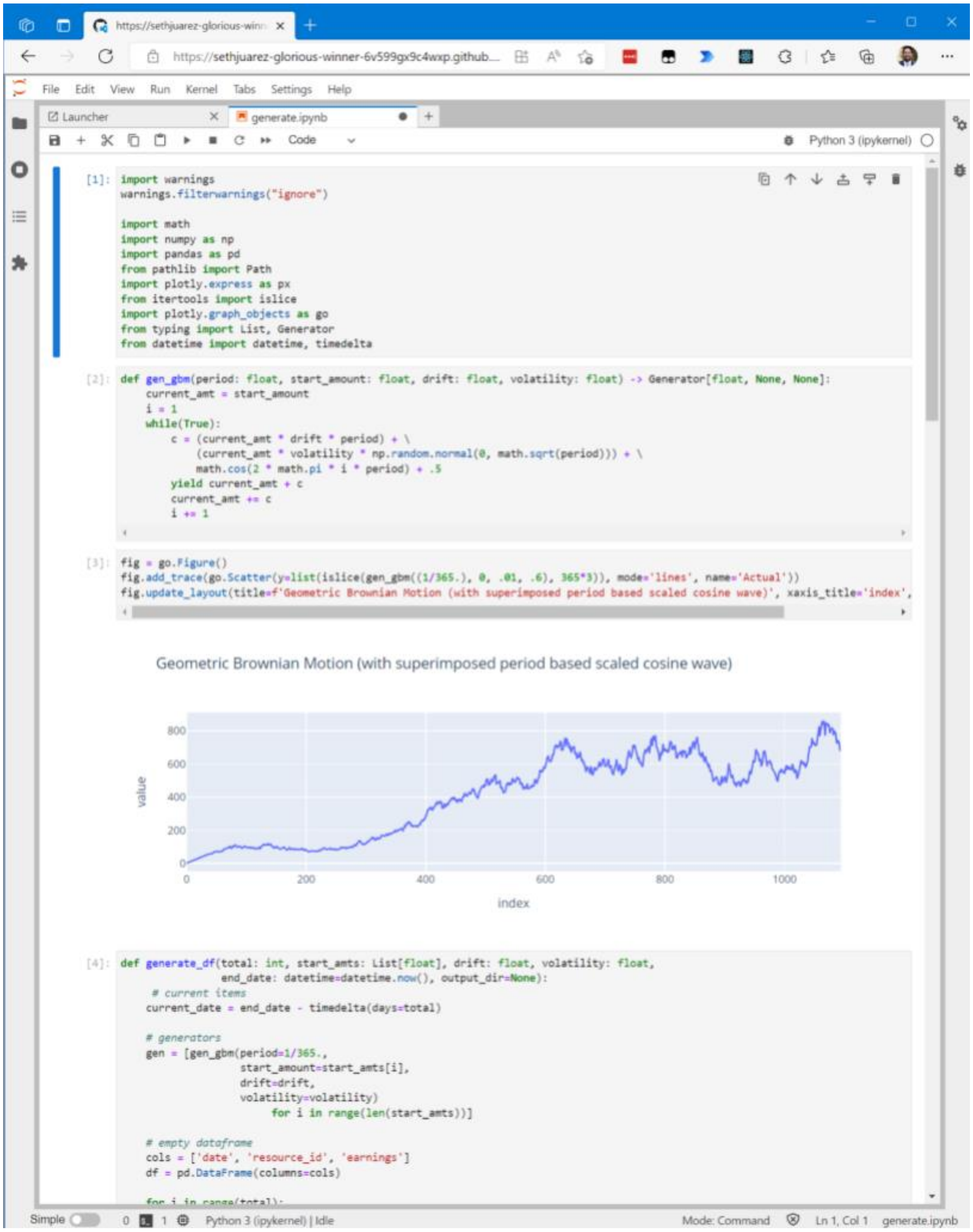
- [ ] Raised Exceptions
- [x] Uncaught Exceptions
- [ ] User Uncaught Exceptions
- [x] trainer.py tlaloc 32

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:


https://sethjuarez-glorious-winn
https://sethjuarez-glorious-winner-6v599gx9c4wxp.github...
File Edit View Run Kernel Tabs Settings Help
Launcher
generate.ipynb
Code
Python 3 (ipykernel)
value
index
Simple
Python 3 (ipykernel) | Idle
Mode: Command
Ln 1, Col 1
generate.ipynb

```
[1]: 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
```

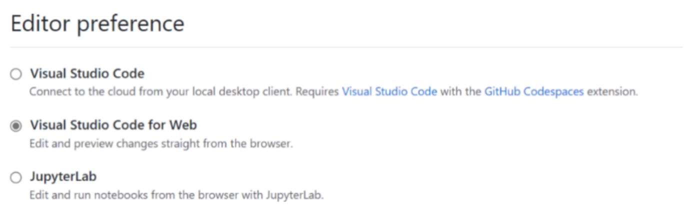
```
[2]: def gen_gbm(period: float, start_amount: float, drift: float, volatility: float) -> Generator[float, None, None]:
         current_amt = start_amount
         i = 1
         while(True):
             c = (current_amt * drift * period) + \
                 (current_amt * volatility * np.random.normal(0,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

```
[3]: 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_title='index',
```

Geometric Brownian Motion (with superimposed period based scaled cosine wave)

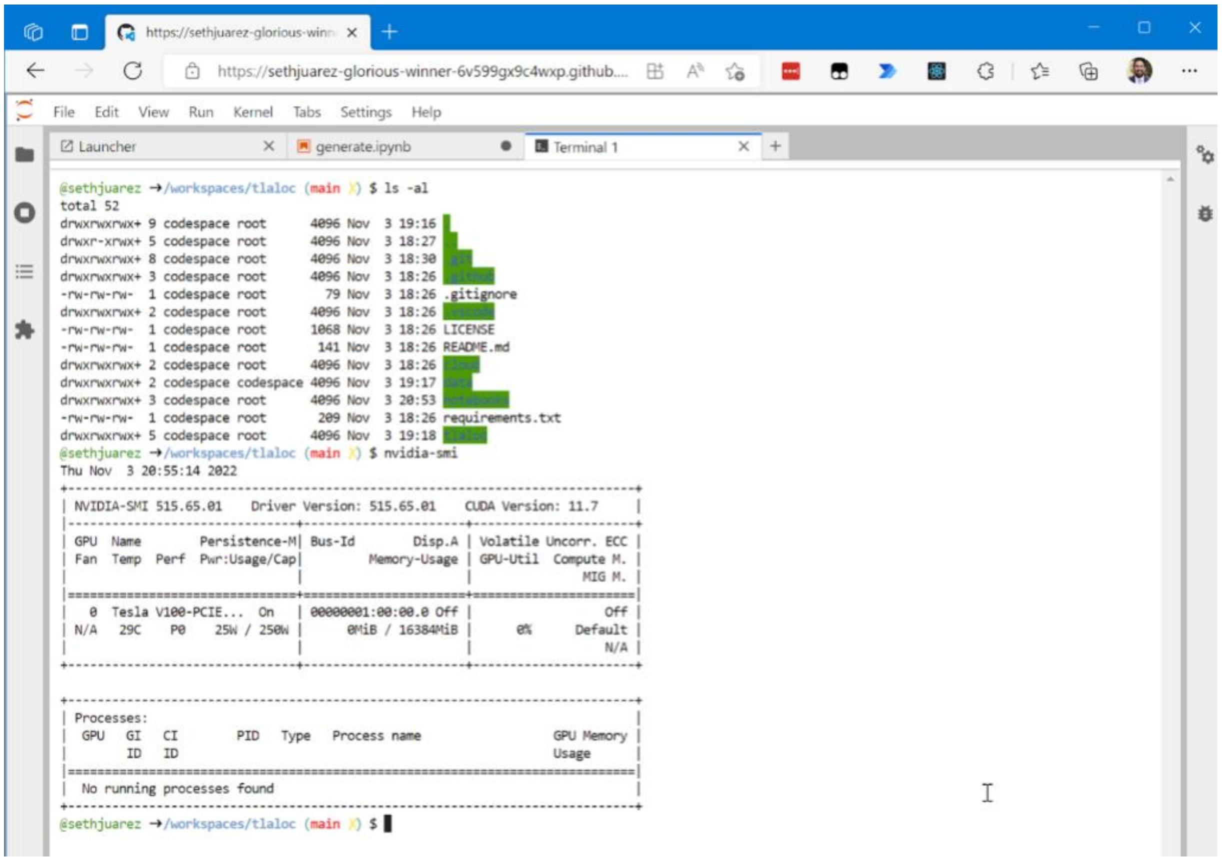
```
[4]: def generate_df(total: int, start_amts: List[float], drift: float, volatility: float,
                     end_date: datetime=datetime.now(), output_dir=None):
          # current items
         current_date = end_date - timedelta(days=total)

         # generators
         gen = [gen_gbm(period=1/365.,
                        start_amount=start_amts[i],
                        drift=drift,
                        volatility=volatility)
                            for i in range(len(start_amts))]

         # empty dataframe
         cols = ['date', 'resource_id', 'earnings']
         df = pd.DataFrame(columns=cols)

         for i in range(total):
```

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):


https://sethjuarez-glorious-winn
https://sethjuarez-glorious-winner-6v599gx9c4wxp.github....
File
Edit
View
Run
Kernel
Tabs
Settings
Help
Launcher
generate.ipynb
Terminal 1

```
@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 .github
-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 cloud
drwxrwxrwx+ 2 codespace codespace 4096 Nov  3 19:17 data
drwxrwxrwx+ 3 codespace root      4096 Nov  3 20:53 notebooks
-rw-rw-rw-  1 codespace root       209 Nov  3 18:26 requirements.txt
drwxrwxrwx+ 5 codespace root      4096 Nov  3 19:18 tlaloc
@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|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla V100-PCIE...  On   | 00000001:00:00.0 Off |                  Off |
| N/A   29C    P0    25W / 250W |      0MiB / 16384MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        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!