Hamilton Global User Group August 2024 Meetup

What is Hamilton?
Hamilton helps data scientists and engineers define testable, modular, self-documenting dataflows, that encode lineage and metadata.
Runs and scales everywhere python does.

Icebreaker: Name and what you're using Hamilton for/looking for.

DAGWORKS August 2024

Agenda

1. Community Spotlight
2. The “news”
3. Open 🎤

Community Spotlight: Gilad Rubin

DAGWORKS

The “News”

DAGWORKS

Some Recent Features / Improvements Released

- Graceful Error Adapter
- Schema Validator + Schema Tracking in the Hamilton UI
- VSCode Plugin
- Async graduation
- Exposing metadata via @datasaver and @dataloader
- Turning off plugin loading
- OpenTelemetry

Graceful Error Adapter

```
dr = (
    driver.Builder()
    .with_modules(my_module)
    .with_adapters(
        default.GracefulErrorAdapter(
            error_to_catch=DoNotProceed, # Catch specific error
            sentinel_value=None          # Sentinel value to inspect for
        )
    )
    .build()
)
# will return {'will_proceed': value, 'never_reached': None}
dr.execute(["will_proceed", "never_reached"])
```

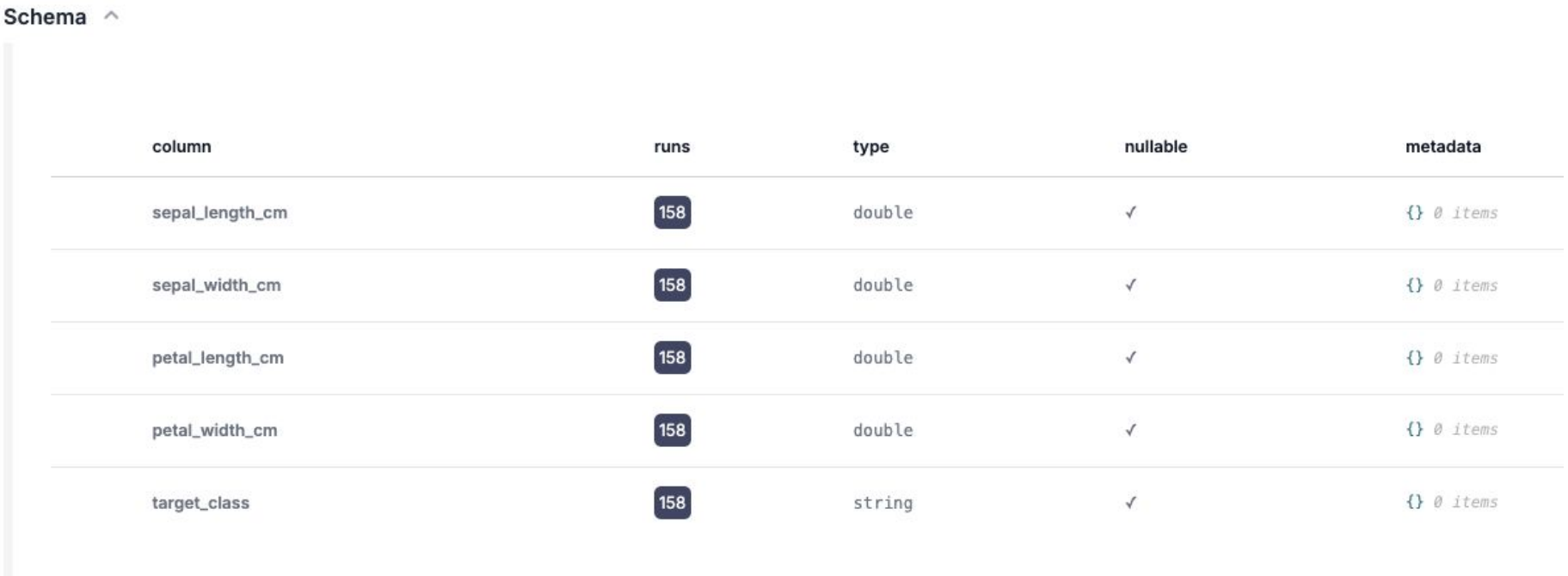
Schema Validator

```
from hamilton import driver
from hamilton.plugins import h_schema

# add to driver
validator_adapter = h_schema.SchemaValidator("./schemas")
dr = (
    driver.Builder()
            .with_modules(*my_modules)
            .with_adapters(validator_adapter) # <--- add it here
            .build()
)
# execute as you normally would
res = dr.execute(["OUTPUT_1", "OUTPUT_2"], inputs=...)
# print & extract the schemas captured
print(json.dumps(validator_adapter.json_schemas, indent=2))
```

Schema Tracking in the UI

Add HamiltonTracker and see it in the UI:

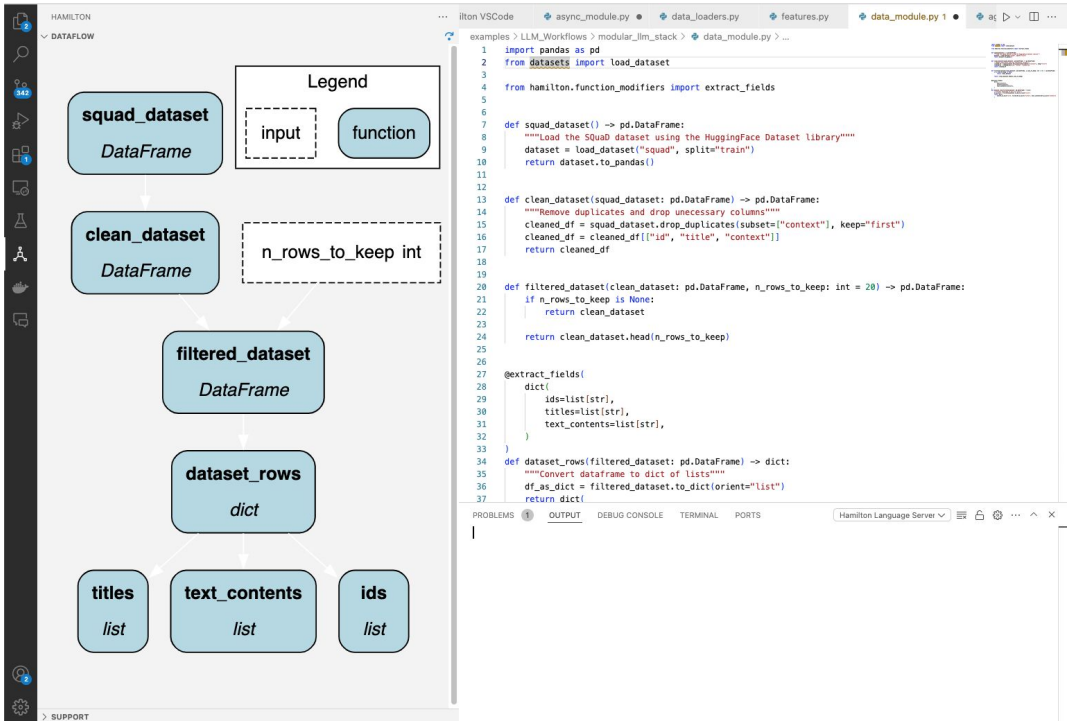


Schema

column	runs	type	nullable	metadata
sepal_length_cm	158	double	✓	{} 0 items
sepal_width_cm	158	double	✓	{} 0 items
petal_length_cm	158	double	✓	{} 0 items
petal_width_cm	158	double	✓	{} 0 items
target_class	158	string	✓	{} 0 items

VSCode Plugin For Hamilton

https://marketplace.visualstudio.com/items?itemName=DAGWorks.hamilton-vsc



In Alpha.
Would love feedback.
Doesn't support all decorator constructs.

Async Graduation + Tracker Support

```
from hamilton import async_driver
from hamilton_sdk import adapters

# initialize async tracker
tracker = adapters.AsyncHamiltonTracker(
   project_id=9,
   username="elijah@dagworks.io",
   dag_name="async_dag",
)
# use builder to construct driver
adr = (
   await async_driver.Builder()
   .with_config(config)
   .with_modules(async_module)
   .with_adapters(async_tracker) # add it here
   .build()
)
# execute like normal
result = await adr.execute(outputs, inputs=...)
```

@datasaver & @dataloader - lightweight

```python
from hamilton.function_modifiers import dataloader, datasaver
from hamilton.io import utils as io_utils

@dataloader() # <---
def raw_data() -> tuple[pd.DataFrame, dict]:
    data = datasets.load_digits()
    df = pd.DataFrame(data.data,
         columns=[f"feature_{i}" for i in range(data.data.shape[1])])
    metadata = io_utils.get_dataframe_metadata(df)
    return df, metadata


def transformed_data(...) -> pd.DataFrame:
    ...

@datasaver() # <---
def saved_data(transformed_data: pd.DataFrame, filepath: str) -> dict:
    transformed_data.to_csv(filepath)
    metadata = io_utils.get_file_and_dataframe_metadata(filepath, transformed_data)
    return metadata
```

Disabling auto plugin importing

disable autoload programmatically

```
from hamilton import registry
registry.disable_autoload()
```

set an environment variable

```
export HAMILTON_AUTOLOAD_EXTENSIONS=0
```

or (note that this is a string)

```
import os
os.environ["HAMILTON_AUTOLOAD_EXTENSIONS"] = "0"
```

or in jupyter notebooks

```
%env HAMILTON_AUTOLOAD_EXTENSIONS=0
```

To load a plugin you'd then do:
directly import hamilton plugins anywhere in your code

```
from hamilton.plugins import mlflow_extensions
```

or, directly register extensions in the registry (has good IDE support via `typing.Literal`)

```
from hamilton import registry
registry.load_extension("mlflow")
```

OpenTelemetry

Just spans right now.

```
from opentelemetry import trace
from opentelemetry.sdk.trace import TracerProvider
from opentelemetry.sdk.trace.export import SimpleSpanProcessor
from opentelemetry.exporter.jaeger import JaegerExporter

jaeger_exporter = JaegerExporter(agent_host_name='localhost',
agent_port=5775)
span_processor = SimpleSpanProcessor(jaeger_exporter)
provider = TracerProvider(active_span_processor=span_processor)
trace.set_tracer_provider(provider)

dr = (
    driver.Builder()
    .with_modules(__main__)
    .with_adapters(h_opentelemetry.OpenTelemetryTracer())
    .build()
)
```

Next meet-up - October:

Speaker? Want to speak?

DAGWORKS

Open Mic.

DAGWORKS

FIN. Thanks for coming!