influxdata®

Anomaly Detection Using InfluxDB and Mage

Anais Dotis-Georgiou | InfluxData

Matt Palmer | Mage

Anais Dotis-Georgiou
Developer Advocate

LinkedIn

Matt Palmer // 🧙‍♂️Mage
Developer Relations

🤓 LinkedIn

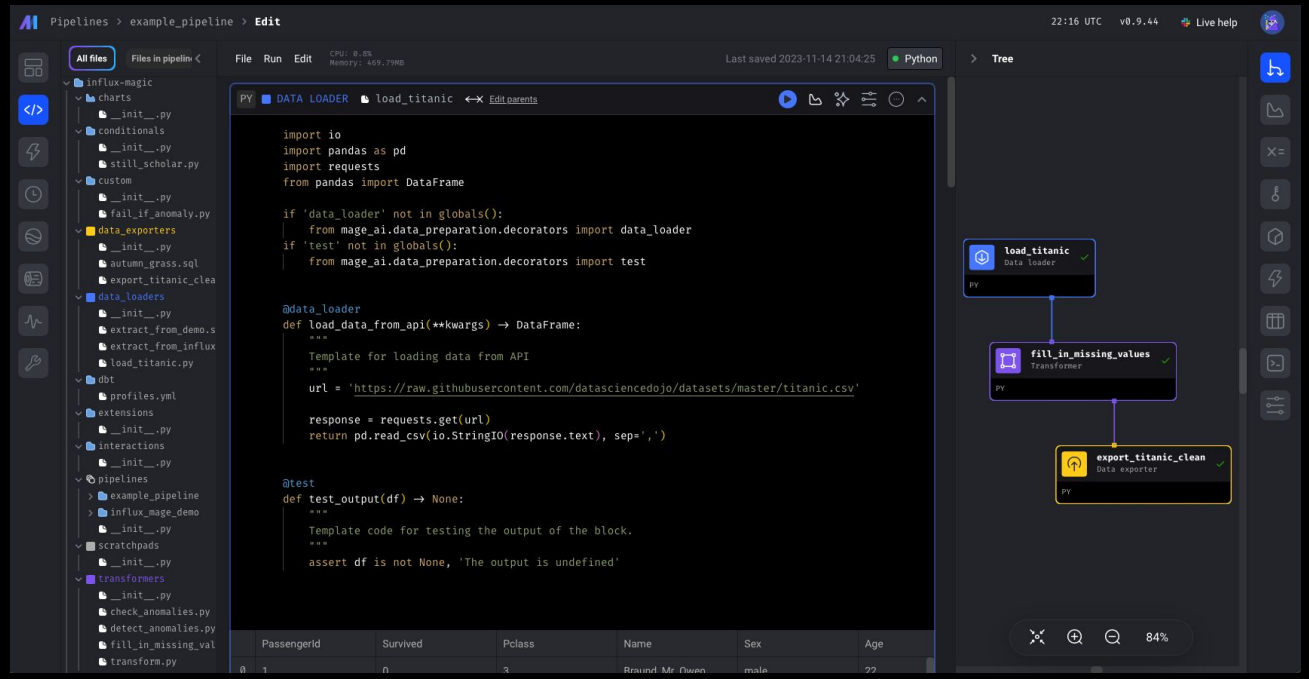
🗞️ Newsletter

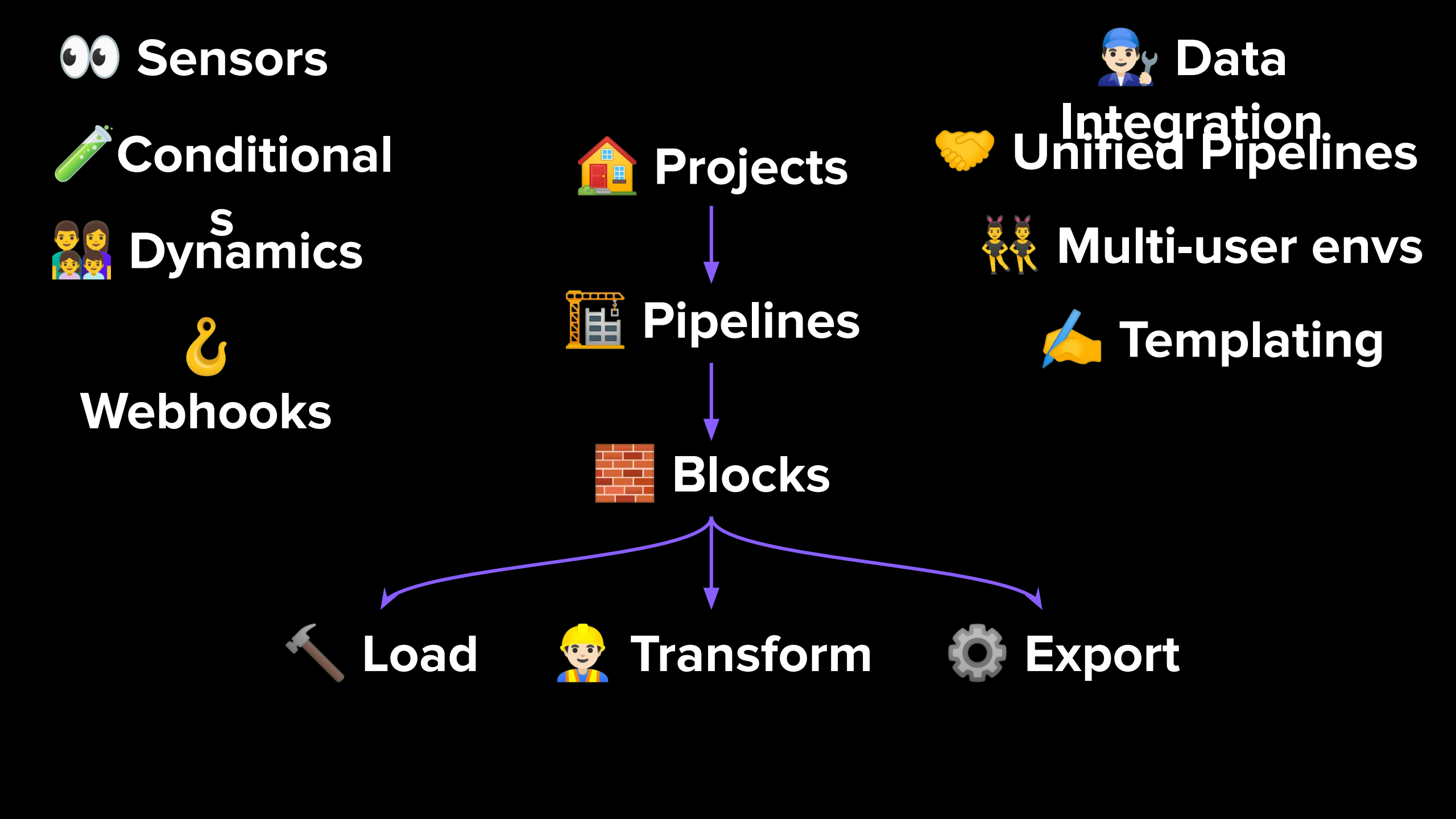
Agenda

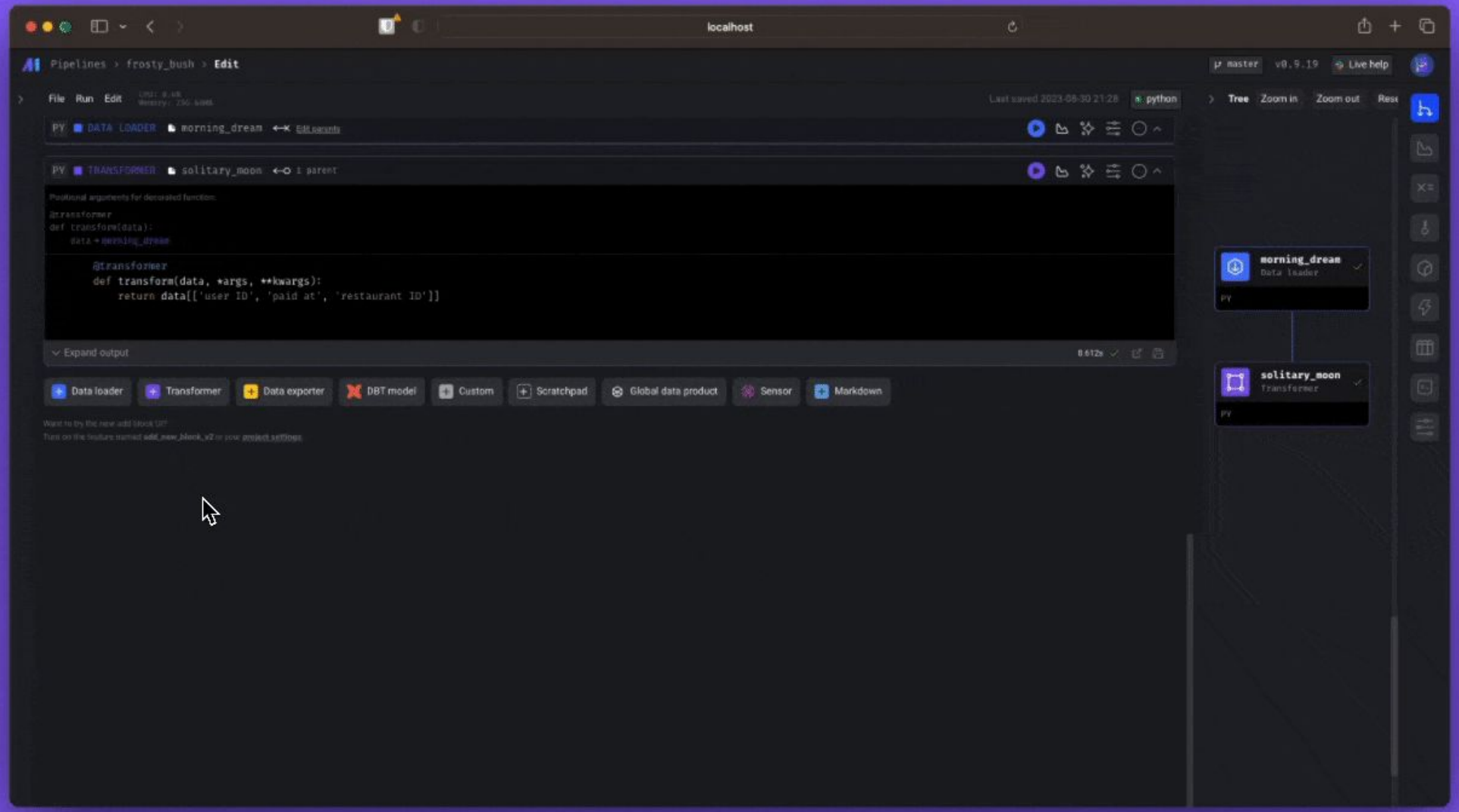
- What is Mage?
- What is InfluxDB?
- **Demo:** Using Mage and InfluxDB for anomaly detection
- Questions

What is **Mage**?

🪄 *An open-source tool for transforming and integrating data*




👀 Sensors
🧪 Conditionals
👨‍👩‍👧 Dynamics
🪝 Webhooks
🏠 Projects
🏗️ Pipelines
🧱 Blocks
🔨 Load
👷 Transform
⚙️ Export
👨‍🔧 Data Integration
🤝 Unified Pipelines
👯 Multi-user envs
✍️ Templating


localhost
Pipelines › frosty_bush › Edit
master
v0.9.19
Live help
File
Run
Edit
python
Tree
Zoom in
Zoom out
PY DATA LOADER morning_dream
PY TRANSFORMER solitary_moon 1 parent
@transformer
def transform(data, *args, **kwargs):
return data[['user ID', 'paid at', 'restaurant ID']]
Expand output
Data loader
Transformer
Data exporter
DBT model
Custom
Scratchpad
Global data product
Sensor
Markdown
morning_dream
Data loader
solitary_moon
Transformer

The Data DevEx

MDS shortcomings currently lead to non-optimal developer experiences.

- Flow state 🌊
  - *“I need to switch between 7 tools/services.”*
- Feedback Loops 🔁
  - *“I spent 5 hours locally testing this DAG.”*
- Cognitive Load 🧱
  - How much do you need to know to do your job?

We’re here to fix that.

🚀 Mage *accelerates* pipeline development

- Hybrid environment
  - Use our GUI for interactive development (or don't, I like VSCode, too) 😎
  - Use blocks as testable, reusable pieces of code.
- Improved DevEx
  - Code and test in parallel.
  - Reduce your dependencies, switch tools less, be efficient.

👨‍💻 Engineering best-practices built-in

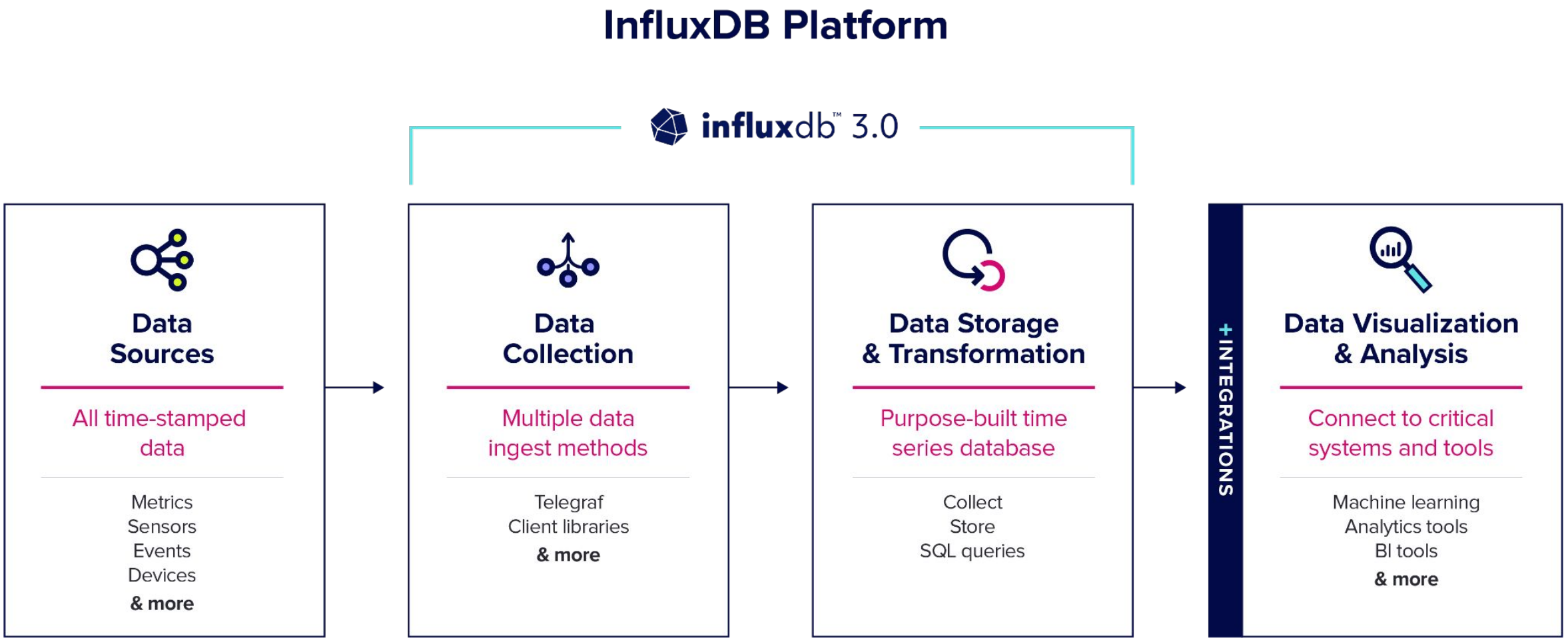
- 🧪 In-line testing and debugging
  - Familiar, notebook-style format
- 🔎 Fully-featured observability
  - Transformation *in one place*: dbt models, streaming, *& more*.
- 🏜️ DRY principles
  - No more 🍝 DAGs with duplicate functions and weird imports
  - DEaaS (sorry, I had to 😅)

🤩 Star Mage on **GitHub**

📚 Check out our **Docs**

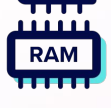
What is InfluxDB?

InfluxData Reference Architecture

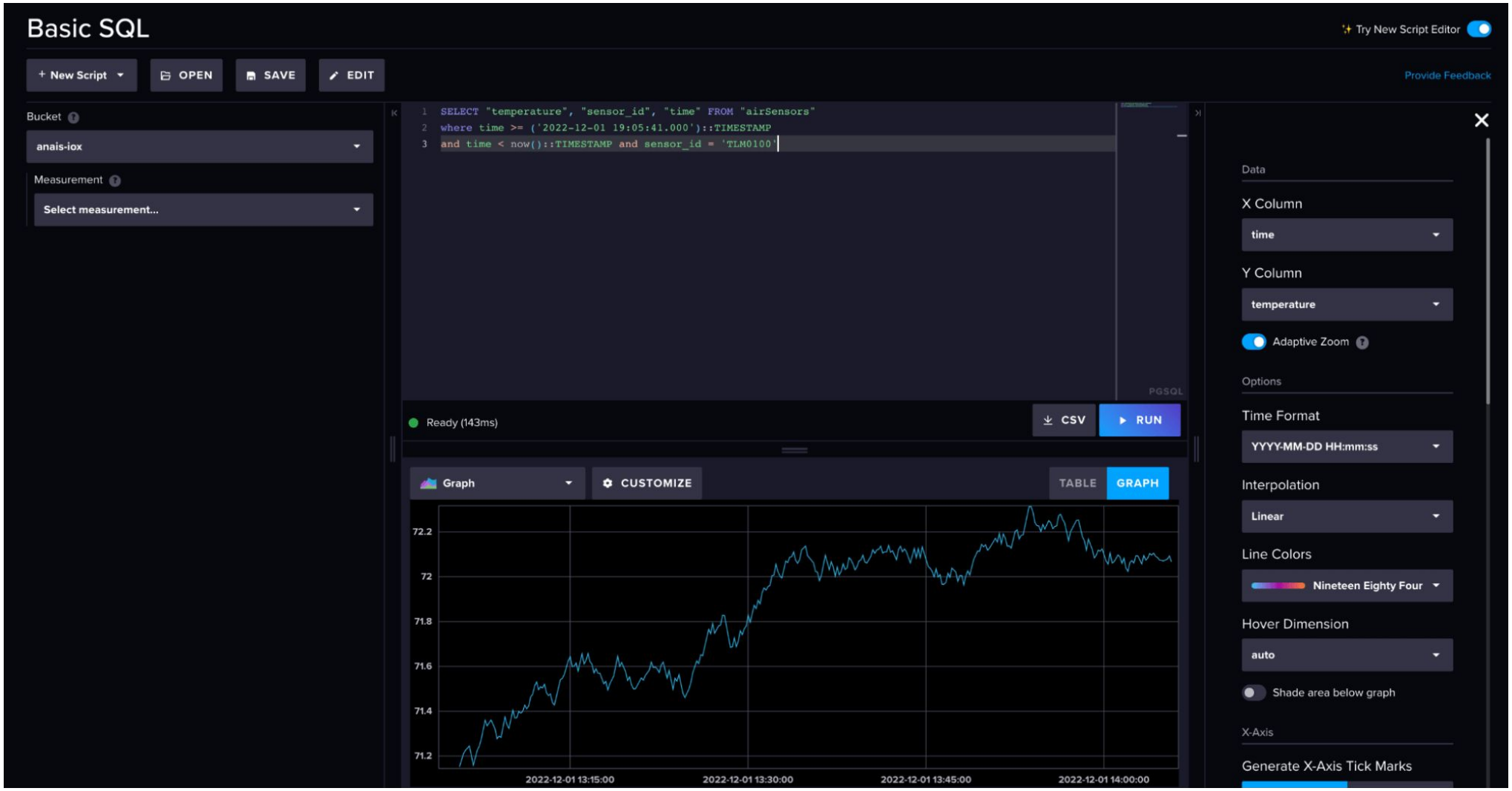


influxdata®

InfluxDB's new storage engine is built on

-  Rust
-   Apache Arrow
-  Apache Parquet
- Arrow Flight
-  DataFusion

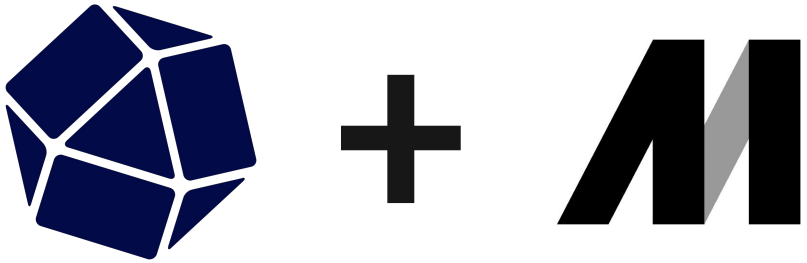
SQL and InfluxQL Support



 |

influxdata®

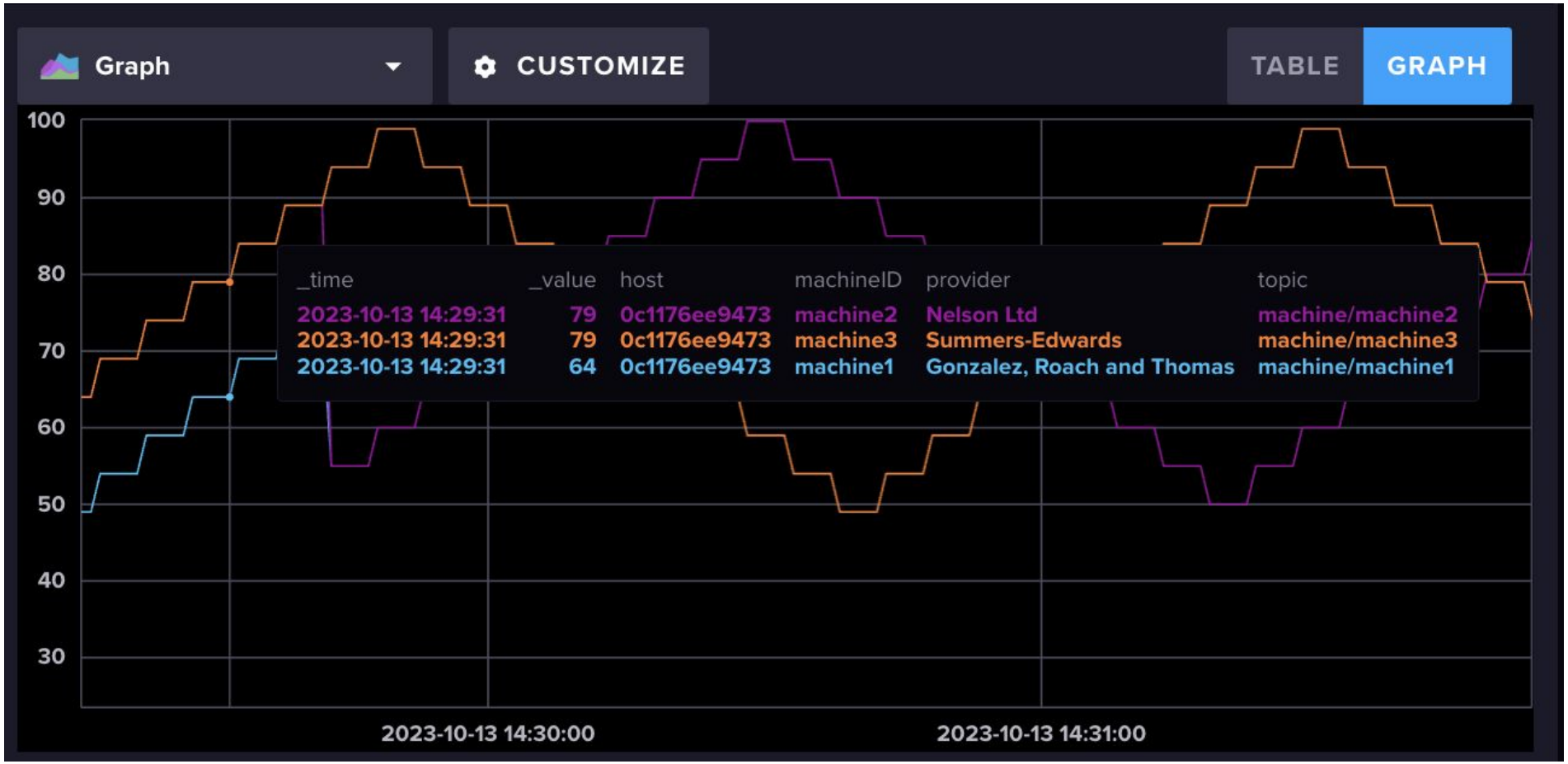
Demo time



🚀 Mage & InfluxDB - Anomaly Detection

- Find the demo here & replicate it yourself!
- Generates machine data— load, vibration, power, & temp
  - We've created anomalies in the machine data we'd like to detect.
- We'll use Mage to build a pipeline that:
  - Loads data
  - Performs anomaly detection (River)
  - Sends alerts via Slack

🚀 Mage & InfluxDB - Anomaly Detection



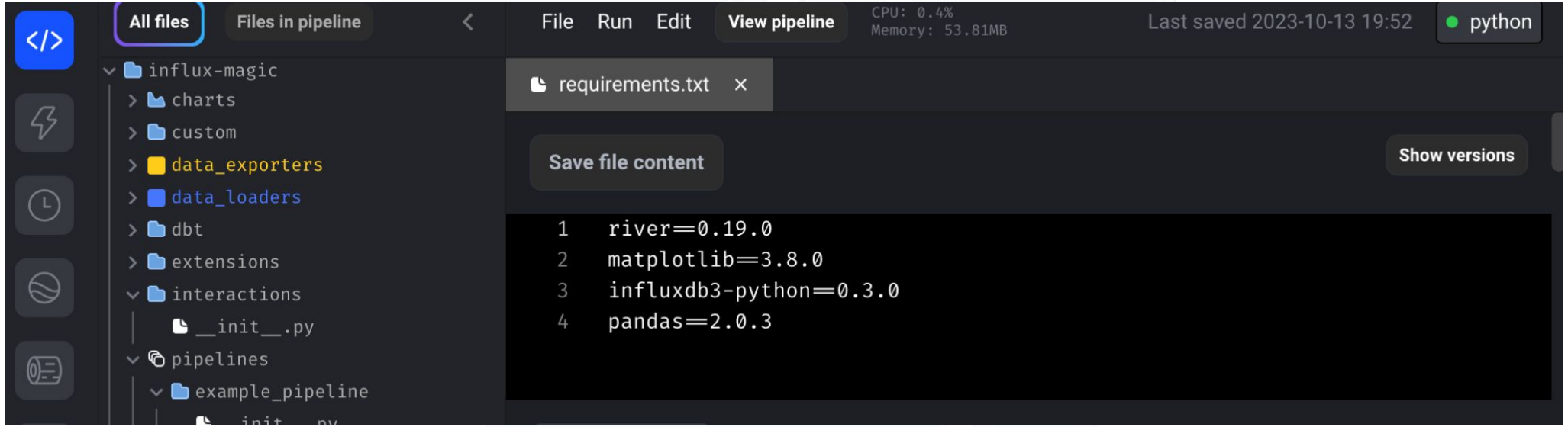
 |

influxdata®

.env

```
export INFLUX_HOST=
export INFLUX_TOKEN=
export INFLUX_ORG=
export INFLUX_DATABASE=
export
MAGE_SLACK_WEBHOOK_URL=https://hooks.slack.com/services/TH8RGQX5
Z/B0T2CMJHH7X/KtL0LNJfWRbyiZWHiG6oJx0T
export MAGE_PROJECT_NAME=influx-magic
export MAGE_ENV=dev
```

Install the requirements



load_influx_data

- Uses the InfluxDB v3 Python Client to query machine data and return a Pandas DataFrame


```
from influxdb_client_3 import InfluxDBClient3
import os
import pandas as pd

if 'data_loader' not in globals():
    from mage_ai.data_preparation.decorators import data_loader
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test


@data_loader
def load_data(*args, **kwargs):
    client = InfluxDBClient3(
        host=os.getenv('INFLUX_HOST'),
        token=os.getenv('INFLUX_TOKEN'),
        database=os.getenv('INFLUX_DATABASE')
    )

    table = client.query(
        query="SELECT * FROM machine_data",
        language="sql"
        )

    client.close()

    df = table.to_pandas()

    time_data = df['time'].values.astype('datetime64[us]')

    df['time'] = pd.to_datetime(time_data)
```

transform_data

- Creates a new column unique_id in the DataFrame. For each row in the DataFrame, it constructs a unique ID by slugifying the provider and then appending the machineID after a hyphen.
- Changes the timestamp datetime format


```
from slugify import slugify
import pandas as pd

if 'transformer' not in globals():
    from mage_ai.data_preparation.decorators import transformer
if 'test' not in globals():
    from mage_ai.data_preparation.decorators import test


@transformer
def transform(df, *args, **kwargs):
    df['unique_id'] = df.apply(lambda row:
f"{slugify(row.provider)}-{row.machineID}", axis=1)

    df['ds'] = pd.to_datetime(df['time'], unit='us')
    # .astype(int)
    print(df.machineID.unique())
    return df
```


detect_anomalies

1. Initialize a new DataFrame
2. Loop through unique IDs: For each unique ID (uid) present in the input DataFrame's unique_id column, the function proceeds with anomaly detection
3. Model Creation: This model employs the HalfSpaceTrees method, which is an online anomaly detection method.
4. Prepare Data.
5. Iterate Through Rows and Detect Anomalies.
6. Combine Results.
7. Return results

Model: Half-Space Trees

1. Data is partitioned with a hyperplane in the feature space. Hyperplanes are randomly generated.
2. Additional planes are added to recursive split the dataset and partition it until the number of points in a partition reach a minimum threshold. Each additional plane creates a node in the tree.
3. Anomalies are defined by the point's depth in the tree. Or whether they exist in smaller partitions.


```
def transform(df, *args, **kwargs):
    anomaly_dfs = pd.DataFrame(columns=["date", "score", "value", "is_anomaly", "unique_id"])

    for uid in df.unique_id.unique():
        print(uid)

        model = compose.Pipeline(
            anomaly.HalfSpaceTrees(
                n_trees=5,
                height=2,
                window_size=250,
                seed=1))

        anomaly_df = pd.DataFrame(columns=["date", "score", "value", "is_anomaly", "unique_id"])

        sliced_df = df[df.unique_id == uid]

        sliced_df['normalized_power'] = \
            (sliced_df.power - sliced_df.power.min() ) / (sliced_df.power.max() - sliced_df.power.min())
```

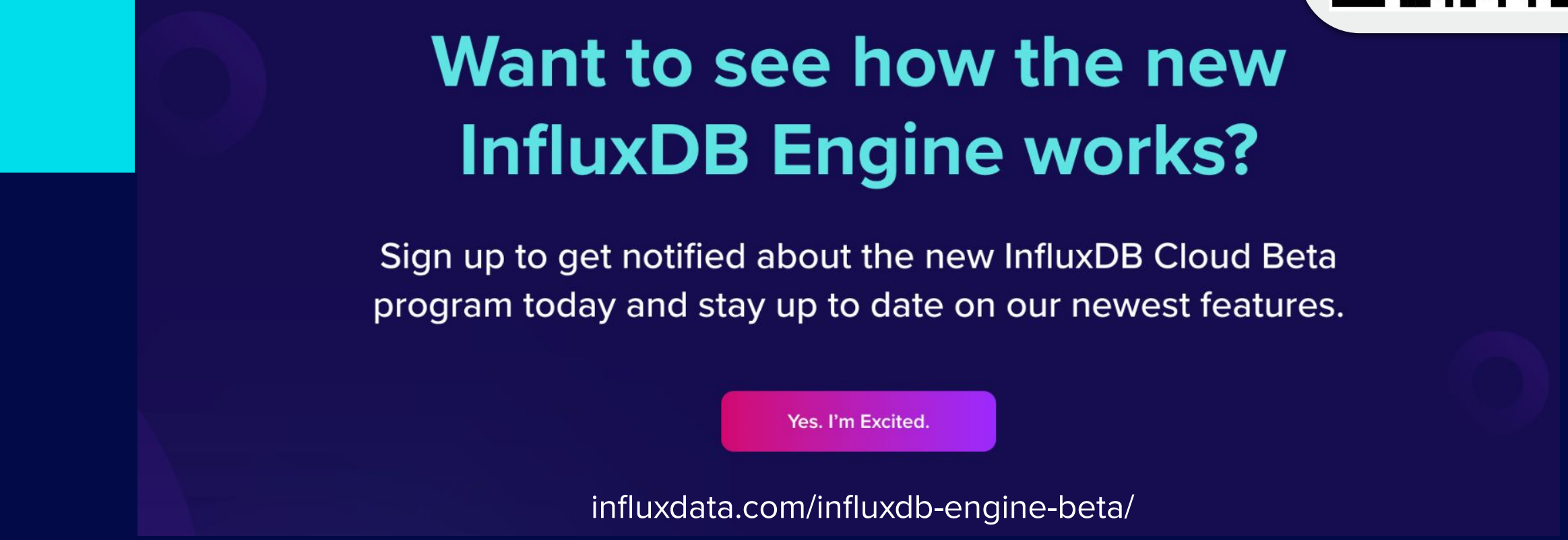
```
for index, row in sliced_df.iterrows():
            ds = row['ds']
            x = row['normalized_power']

            features = {uid: x}

            model = model.learn_one(features)
            score = model.score_one(features)

            anomaly_df = pd.concat([anomaly_df, pd.DataFrame(
                [{'date': ds, 'score': float(score), 'value': x, 'is_anomaly': score > .8, 'unique_id': uid}])
                                    ])

        anomaly_dfs = pd.concat([anomaly_dfs, anomaly_df])

    return anomaly_dfs
```

check_anomalies

1. Iterate Through Unique IDs
2. Filter Data by Unique ID
3. Check for Anomalies
4. Data Preparation for Plotting
5. Plot


```
def transform(df, *args, **kwargs):
    for uid in df.unique_id.unique():
        uid_df = df[df.unique_id == uid]
        if True in uid_df.is_anomaly.unique():

            anomalies = uid_df[uid_df.is_anomaly]
            # Create the plot
            fig, ax1 = plt.subplots()

            # Plot the first data set on the primary y-axis
            ax1.plot(uid_df.date, uid_df.score, label='Anomaly Score', color='blue')
            ax1.tick_params(axis='y', labelcolor='blue')

            # Add the scatter point
            ax1.scatter(anomalies.date, anomalies.value, color='red', s=100, edgecolors='black',
label='Anomaly')

            # Create a twin y-axis to plot the second data set
            ax2 = ax1.twinx()
            ax2.plot(uid_df.date, uid_df.value, label='Normalized Value', color='green')
            ax2.tick_params(axis='y', labelcolor='red')

            # Add title and labels
            plt.title(f'Anomalies - Power - {uid}')
            plt.xlabel('Value')
            plt.ylabel('Time')
            plt.legend()

            # Show the plot
            plt.show()
```


Get Started



influxdata.com/influxdb-engine-beta/

Mage

🤩 Star Mage on **GitHub:** https://github.com/mage-ai/mage-ai
📗 Check out our **docs** and get started: https://docs.mage.ai
👯 Join our **Slack:** https://www.mage.ai/chat
👋 Say hi on **LinkedIn:**
https://www.linkedin.com/company/magetech/

Matt

👋 Say hi on **LinkedIn:** https://www.linkedin.com/in/matt-palmer/
🤓 Check out my blog: https://mattpalmer.io
📰 Read my newsletter: https://newsletter.casewhen.xyz

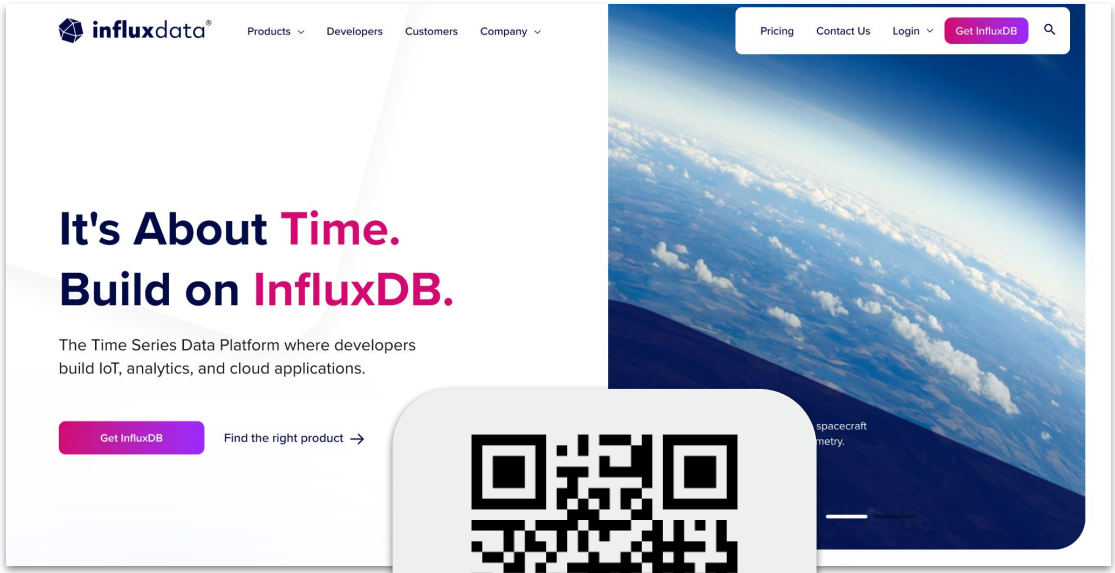
InfluxDB Community Slack workspace

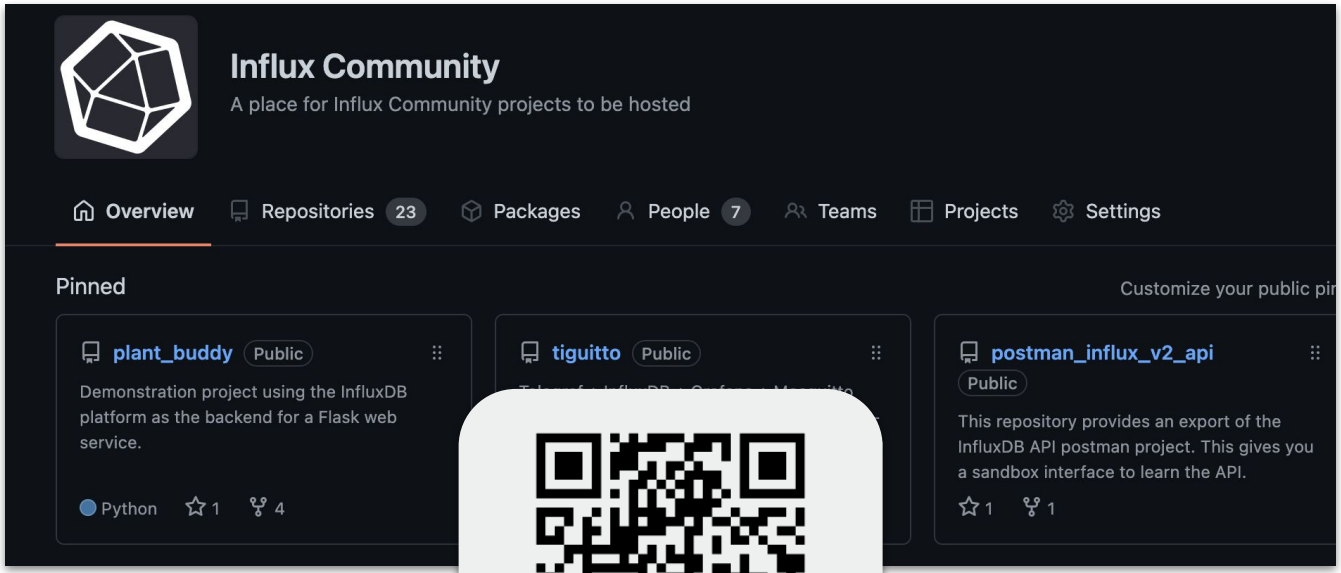
Please join us in the InfluxDB Community Slack at www.influxdata.com/slack.

To participate in conversations, join the #influxdb_iox channel.

Try it yourself

Get Started





34

influxdata

Get Help + Resources!

Forums: community.influxdata.com

Slack: influxcommunity.slack.com

GH: github.com/InfluxCommunity

Docs: docs.influxdata.com

Blogs: influxdata.com/blog

InfluxDB University: influxdata.com/university

35

InfluxDB Resources

Webinar: Gain Better Observability with OpenTelemetry and InfluxDB

Leverage OpenTelemetry and InfluxDB to collect and analyze metrics, logs, and traces, enabling better anomaly detection, root-cause analysis, and alerting.

Watch now bit.ly/3qhemCw

Save 96% on Data Storage Costs:

Learn more bit.ly/3NJEcGZ

Run a Proof of Concept:

Learn more bit.ly/3puRsal

influxdata®

influxdata®

THANK YOU