



Anomaly Detection Using InfluxDB and Mage

Anais Dotis-Georgiou | InfluxData
Matt Palmer | Mage



Anais Dotis-Georgiou

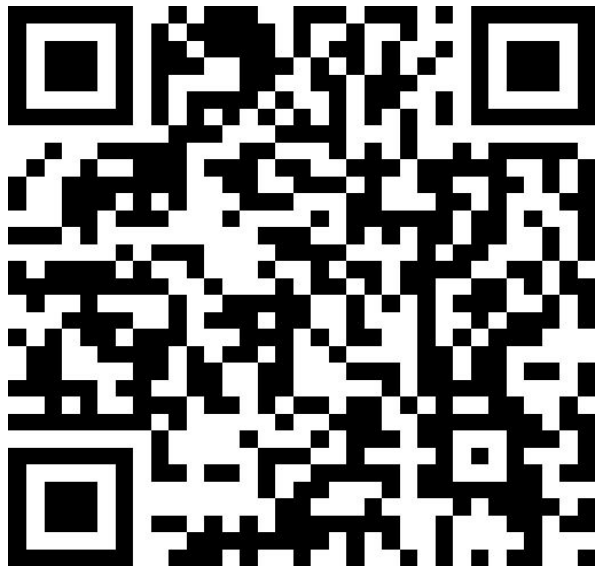
Developer Advocate



LinkedIn



Matt Palmer // 🧙 Mage Developer Relations



LinkedIn



Newsletter




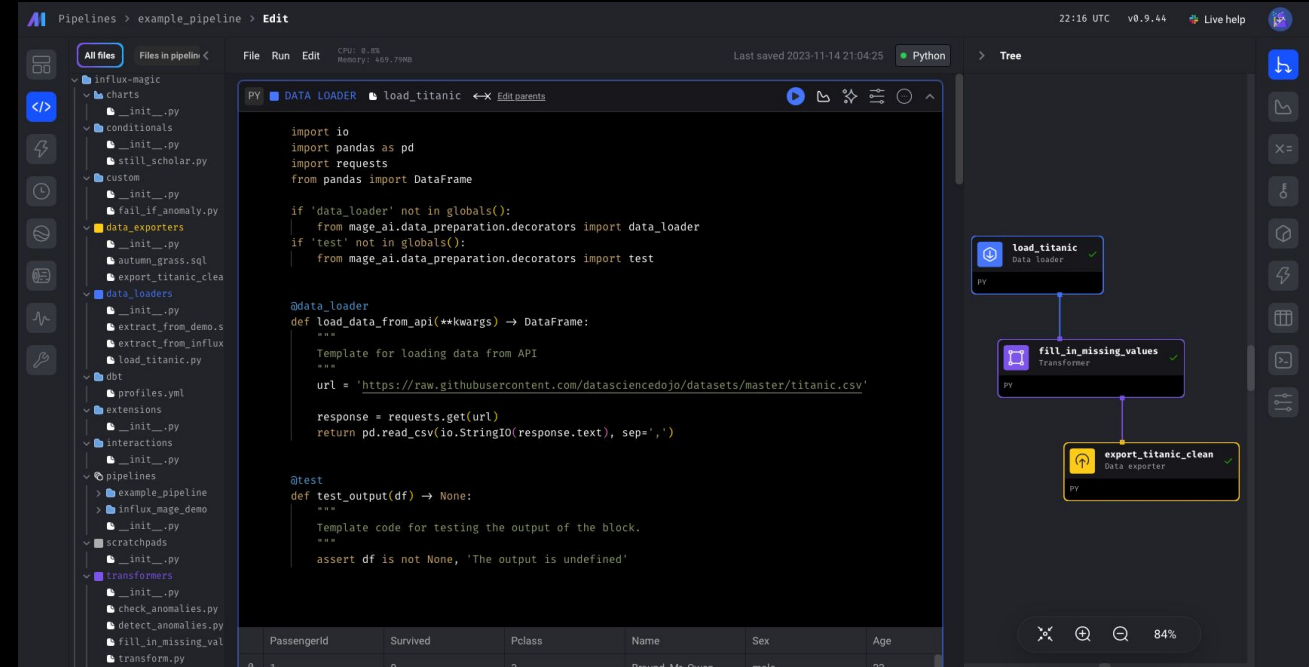
Agenda

- 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



The screenshot displays the Mage IDE interface. On the left is a file explorer showing a project structure with folders like 'influx-magic', 'charts', 'conditionals', 'custom', 'data_exporters', 'data_loaders', 'dbt', 'extensions', 'interactions', 'pipelines', 'scratchpads', and 'transformers'. The main editor shows a Python file named 'load_titanic.py' with the following code:

```
import io
import pandas as pd
import requests
from pandas import DataFrame

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_from_api(**kwargs) -> DataFrame:
    """
    Template for loading data from API
    """
    url = 'https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv'
    response = requests.get(url)
    return pd.read_csv(io.StringIO(response.text), sep=',')

@test
def test_output(df) -> None:
    """
    Template code for testing the output of the block.
    """
    assert df is not None, 'The output is undefined'
```

At the bottom of the editor, a preview of the loaded data is shown as a table:

PassengerId	Survived	Pclass	Name	Sex	Age
0	1	0	Braund, Mr. Owen	male	22

On the right side, a pipeline diagram shows three blocks: 'load_titanic' (Data loader), 'fill_in_missing_values' (Transformer), and 'export_titanic_clean' (Data exporter), connected in a vertical sequence. The interface also shows system information at the top right: '22:16 UTC v0.9.44 Live help'.

 **Sensors**

 **Conditional**

 **Dynamics**

 **Webhooks**

 **Data**

 **Integration.
Unified Pipelines**

 **Multi-user envs**

 **Templating**

 **Projects**

 **Pipelines**

 **Blocks**

 **Load**

 **Transform**

 **Export**



localhost

Pipelines > frosty_bush > Edit

File Run Edit 10:21:45 AM Wednesday, 25th June 2023 Last saved 2023-06-30 21:28 python Tree Zoom in Zoom out Rest

PY DATA LOADER morning_dream Edit source

PY TRANSFORMER solitary_moon 1 parent

Positional arguments for decorated function:

```
@transformer
def transform(data):
    data = morning_dream

    @transformer
    def transform(data, *args, **kwargs):
        return data[['user ID', 'paid at', 'restaurant ID']]
```

Expand output 0.612s

Data loader Transformer Data exporter DBT model Custom Scratchpad Global data product Sensor Markdown

Want to try the new add block UI? Turn on the feature named add_new_block_v2 in your project settings

morning_dream Data loader PY

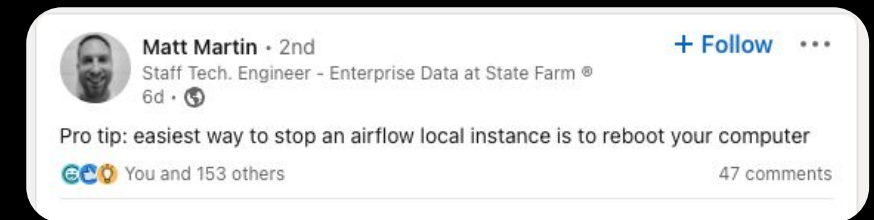
solitary_moon Transformer PY



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**

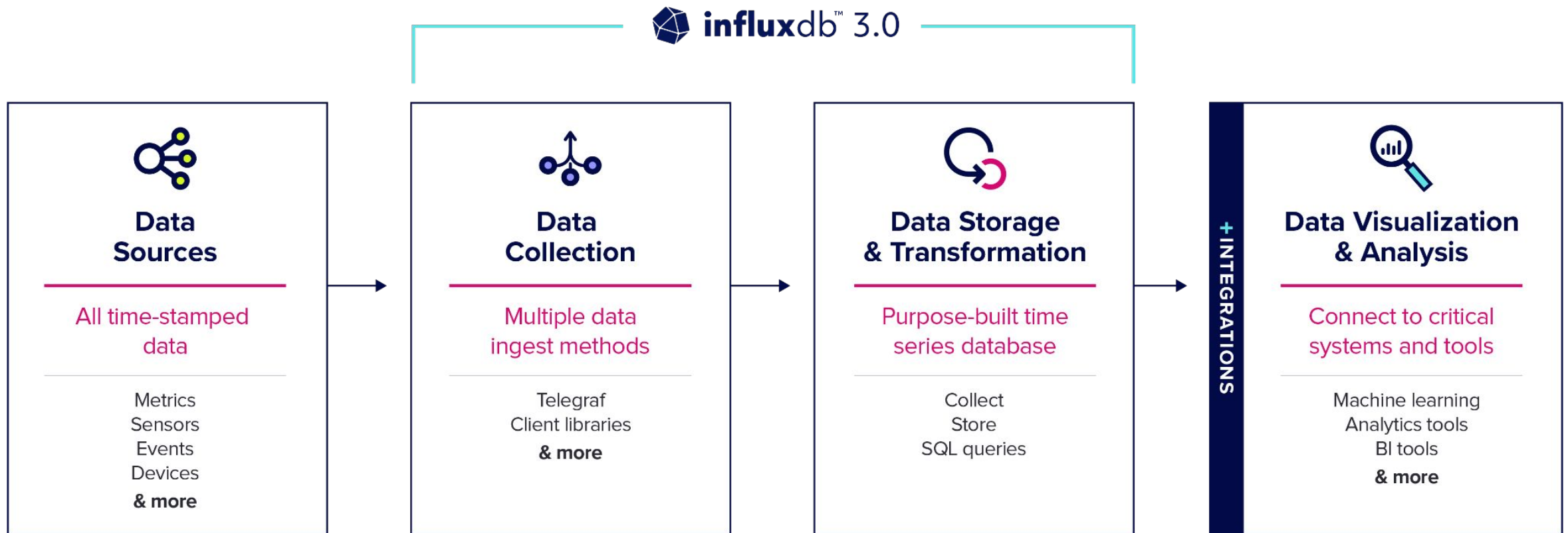


 Check out our **Docs**


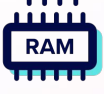



What is InfluxDB?

InfluxData Reference Architecture

InfluxDB Platform



InfluxDB's new storage engine is built on

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

SQL and InfluxQL Support

The screenshot displays the InfluxDB SQL editor interface. At the top, the title "Basic SQL" is visible. Below the title, there are buttons for "+ New Script", "OPEN", "SAVE", and "EDIT". On the right side, there is a "Try New Script Editor" toggle and a "Provide Feedback" link.

The main area is divided into three sections:

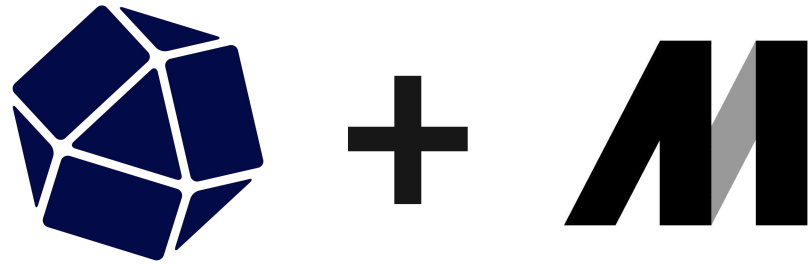
- Left Panel:** Contains a "Bucket" dropdown menu set to "anais-iox" and a "Measurement" dropdown menu set to "Select measurement...".
- Center Panel:** A code editor containing the following SQL query:

```
1 SELECT "temperature", "sensor_id", "time" FROM "airSensors"  
2 where time >= ('2022-12-01 19:05:41.000')::TIMESTAMP  
3 and time < now()::TIMESTAMP and sensor_id = 'TLM0100'
```
- Right Panel:** A configuration panel for the graph. It includes:
 - Data:** A dropdown menu for "X Column" set to "time" and a dropdown menu for "Y Column" set to "temperature".
 - Options:** A toggle for "Adaptive Zoom" which is turned on.
 - Time Format:** A dropdown menu set to "YYYY-MM-DD HH:mm:ss".
 - Interpolation:** A dropdown menu set to "Linear".
 - Line Colors:** A dropdown menu set to "Nineteen Eighty Four".
 - Hover Dimension:** A dropdown menu set to "auto".
 - Shade area below graph:** A toggle which is turned off.
 - X-Axis:** A section with a "Generate X-Axis Tick Marks" button.

Below the code editor, there is a status bar showing "Ready (143ms)" and buttons for "CSV" and "RUN". Below the status bar, there are tabs for "Graph" (selected), "CUSTOMIZE", "TABLE", and "GRAPH".

The main visualization is a line graph showing temperature data over time. The x-axis represents time from 2022-12-01 13:15:00 to 2022-12-01 14:00:00. The y-axis represents temperature, ranging from 71.2 to 72.2. The graph shows a fluctuating line that generally increases from approximately 71.2 at 13:15:00 to about 72.1 at 14:00:00.

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



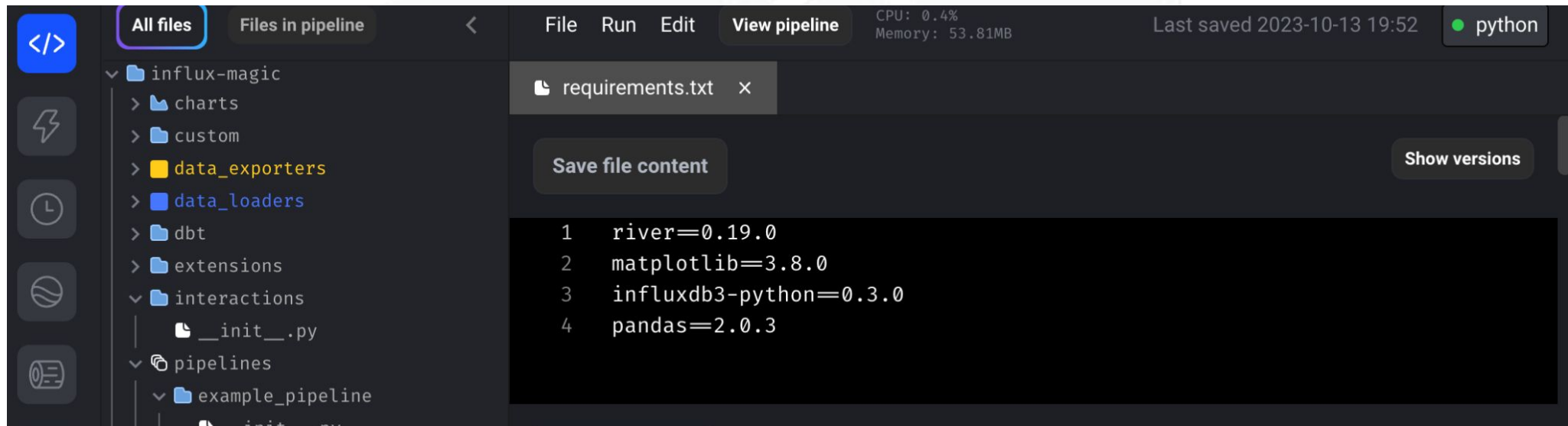
Mage & InfluxDB - Anomaly Detection



.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/B0I2CMJHH7X/KtL0LNJfWRbyiZWHiG6oJx0T  
export MAGE_PROJECT_NAME=influx-magic  
export MAGE_ENV=dev
```

Install the requirements



The screenshot shows a code editor interface with a dark theme. The left sidebar displays a file tree for a project named 'influx-magic', with folders like 'charts', 'custom', 'data_exporters', 'data_loaders', 'dbt', 'extensions', 'interactions', and 'pipelines'. The main editor area is open to a file named 'requirements.txt'. The file content is as follows:

```
1 river=0.19.0
2 matplotlib=3.8.0
3 influxdb3-python=0.3.0
4 pandas=2.0.3
```

At the top of the editor, there are tabs for 'All files' and 'Files in pipeline'. The top right corner shows system information: 'CPU: 0.4%', 'Memory: 53.81MB', 'Last saved 2023-10-13 19:52', and a 'python' environment indicator. A 'Save file content' button is visible above the code, and a 'Show versions' button is on the right side of the editor area.

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 recursively 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



Want to see how the new InfluxDB Engine works?

Sign up to get notified about the new InfluxDB Cloud Beta program today and stay up to date on our newest features.

Yes. I'm Excited.

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

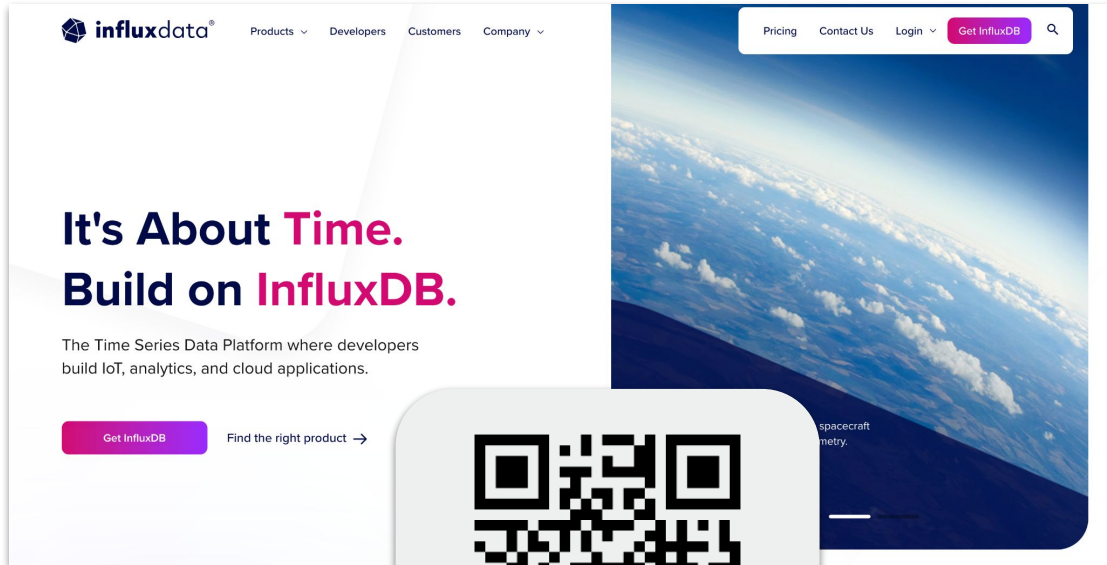


Please join us in the InfluxDB
Community Slack at
www.influxdata.com/slack.

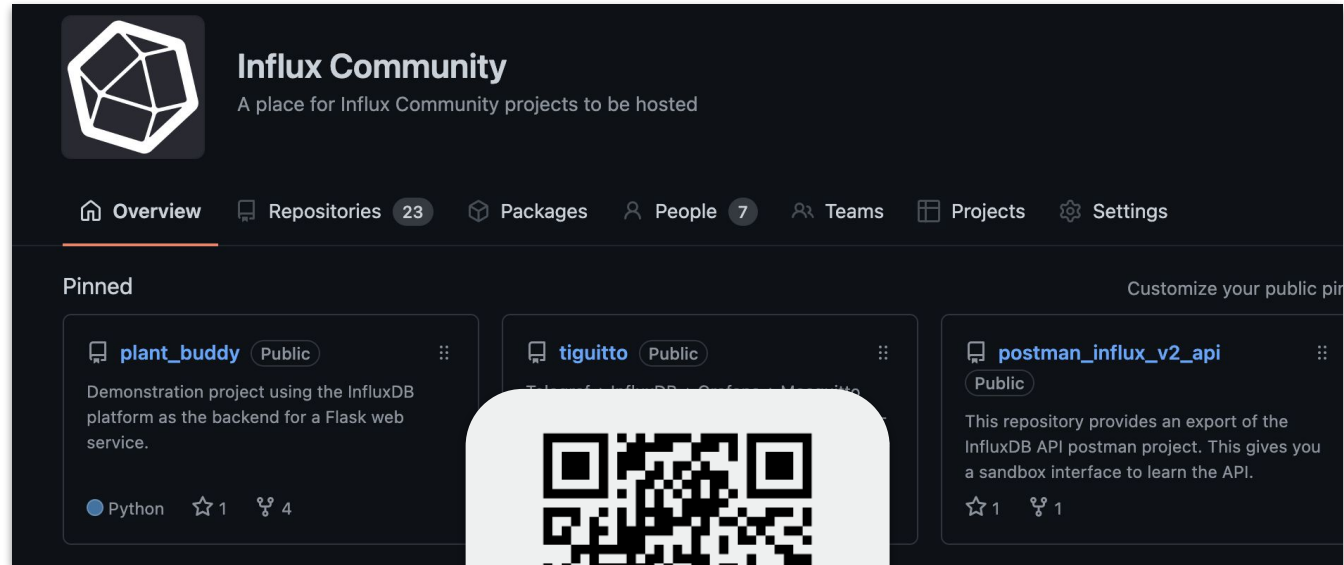
To participate in conversations,
join the `#influxdb_iox` channel.

Try it yourself

Get Started



The screenshot shows the InfluxData website homepage. At the top left is the InfluxData logo. Navigation links include Products, Developers, Customers, and Company. On the right, there are links for Pricing, Contact Us, Login, and a prominent 'Get InfluxDB' button. The main content area features the headline 'It's About Time. Build on InfluxDB.' followed by a sub-headline 'The Time Series Data Platform where developers build IoT, analytics, and cloud applications.' Below this is another 'Get InfluxDB' button and a link 'Find the right product ->'. The background image shows a view of Earth from space.



The screenshot shows the Influx Community GitHub page. At the top left is the Influx Community logo. The page title is 'Influx Community' with the subtitle 'A place for Influx Community projects to be hosted'. Navigation tabs include Overview, Repositories (23), Packages, People (7), Teams, Projects, and Settings. The 'Pinned' section displays three repositories: 'plant_buddy' (Public), 'tigitto' (Public), and 'postman_influx_v2_api' (Public). Each repository card shows a brief description and statistics like stars and forks.



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

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





THANK YOU