DATAMOLE

datamole.ai

# How to store data

Lecture 5, CTU FEL, MLE course


**Jan Lukány**

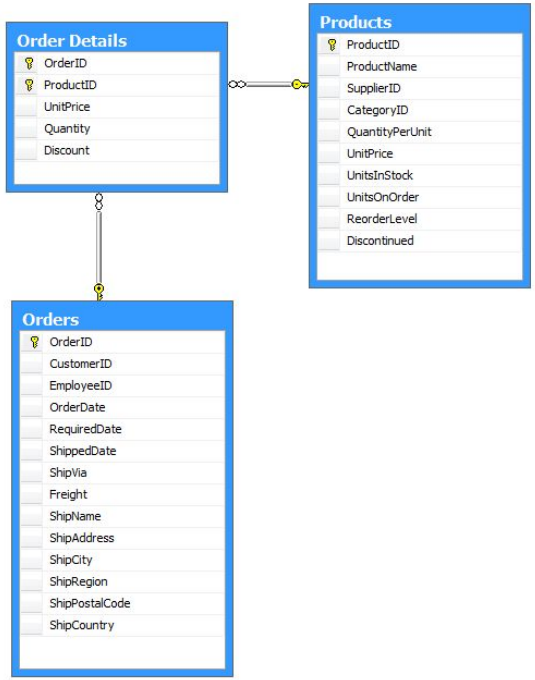
21st October 2025

DATAMOLE

datamole.ai

*a Czech **data and AI company** of **80+ people** founded in **2015.** We **develop custom AI, IoT & UI solutions** that innovate industrial companies worldwide - in agriculture, machinery or biotech.*


# Relational SQL Databases




- Tables with relations
- ACID guarantees
- Dedicated database machine/server

typical pricing:
- paying per machine/server + storage.
- ballpark minimal cost: $10-30 / month (small machine) + $0.15 per GB

examples: MSSQL, PostgreSQL


DATAMOLE

# Relational SQL Databases

```
import ...

db = ...connect(connection_string="...")

db.execute(
    "INSERT INTO logs (site, event_date, message) VALUES (?, ?, ?)",
    "site-A", "2025-10-23", "This is the data from site A..."
)
db.commit()

rows = db.execute(
    "SELECT message FROM logs WHERE site = ?",
    "site-A"
)

for row in rows:
    data = row[0]
```

# Blob/File Storage

- “normal file system” (on cloud)
- Cheaper, simpler
- operations:
  - read
  - write
  - list/walk

/path-to-file-1
/path-to-file-2
/a-prefix
    /file-3-with-prefix
    /file-4-with-prefix

typical pricing:
- per GB (e.g. $0.002 per GB)
- # of operations (e.g. $0.005 per 10K reads)

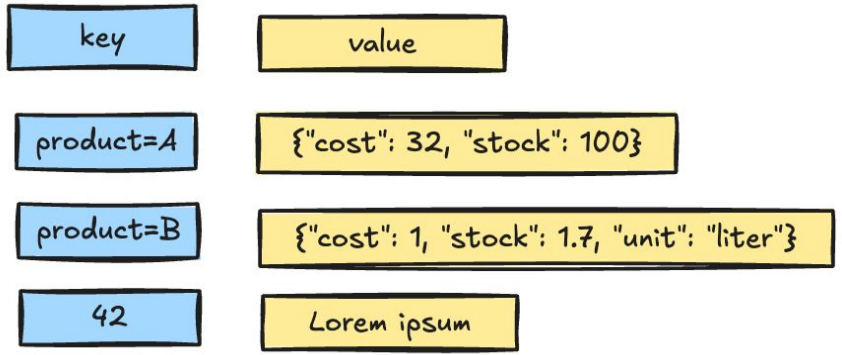
examples: Azure Blob Storage, AWS S3

# Blob/File Storage

```
import ...

client = Client(credentials=...)

client.upload(
    file_path="site-A/2025-10-21.txt",
    data="This is the data from site A from Oct 21st 2025."
)

for file_path in client.list_files(prefix="site-A"):
    data = client.download(file_path)
```

# Key-Value Stores

- Scalable, fast for key-based lookups and cost-effective
- Flexible schema




typical pricing:
- per GB (e.g. $0.045/GB)
- per # of operations (e.g. $0.005 / 10K reads)

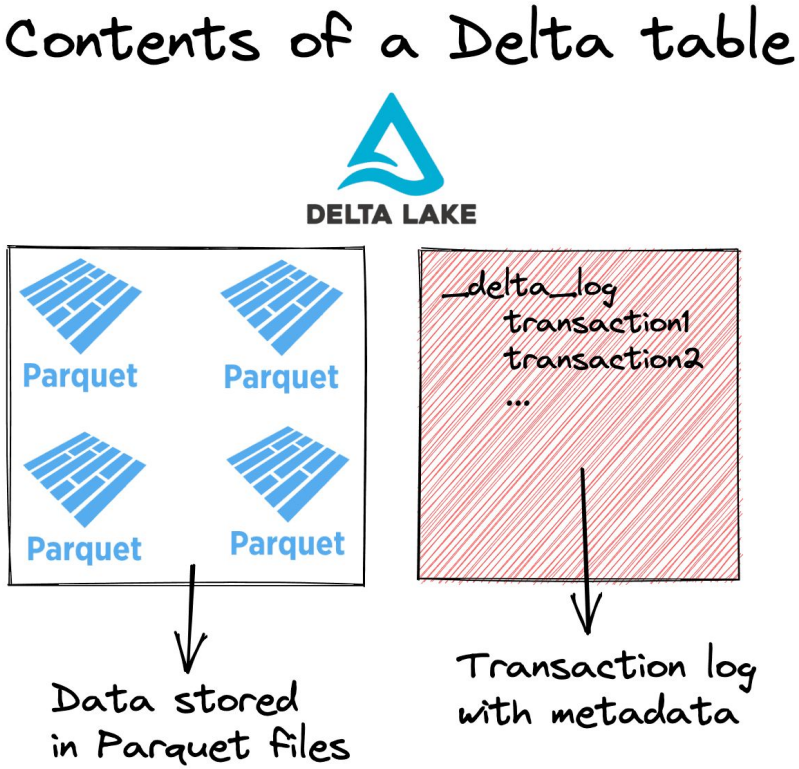
examples: Azure Table, AWS DynamoDB

# Key-Value Stores

```
import ...

db = ...connect(endpoint="...")

db.set(
    "site:A",
    {"location": "New York", "name": "Main Production Plant"}
)

db.set(
    "site:B",
    {"location": "London", "region": "EMEA", "contact_email": "..."}
)

db.set(
    "site:A:status",
    "ONLINE"
)

site_A_data = db.get("site:A")
# site_A_data = {"location": "New York", ...}

site_A_status = db.get("site:A:status")
# site_A_status = "ONLINE"
```

# Delta Lake

- format for tabular data on top of cheap file/blob storage
- ACID (single table)
- scalable
- unified batch & streaming
- time travel
- schema enforcement & evolution




ICEBERG

DATAMOLE

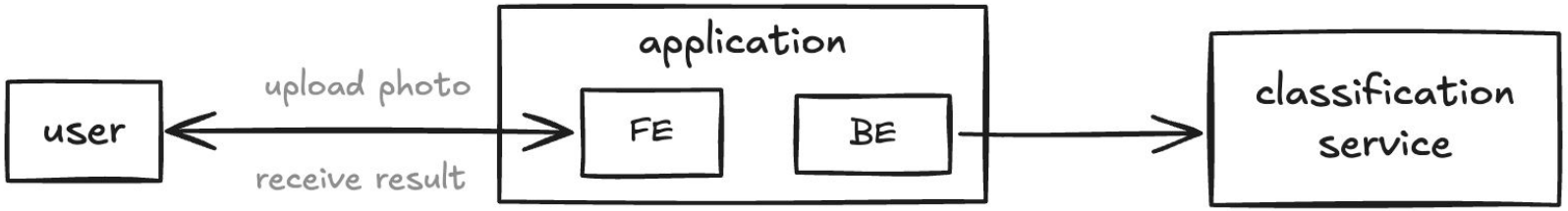
# Delta Lake

demo

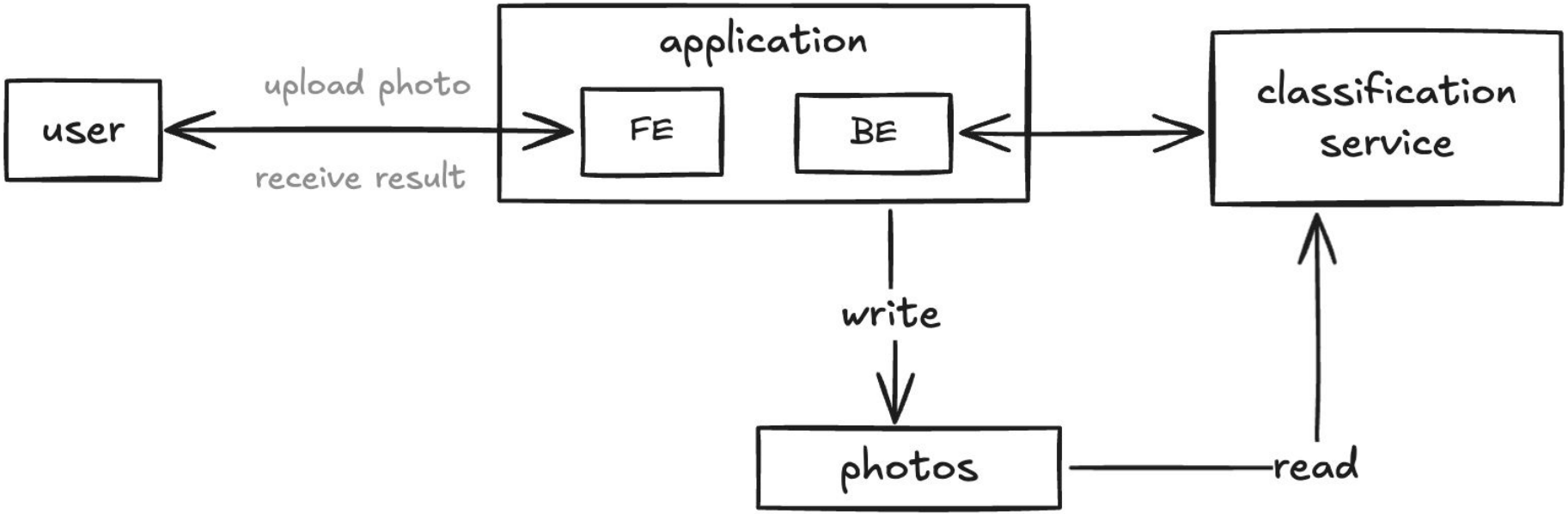
# Case studies

# Photo Classification

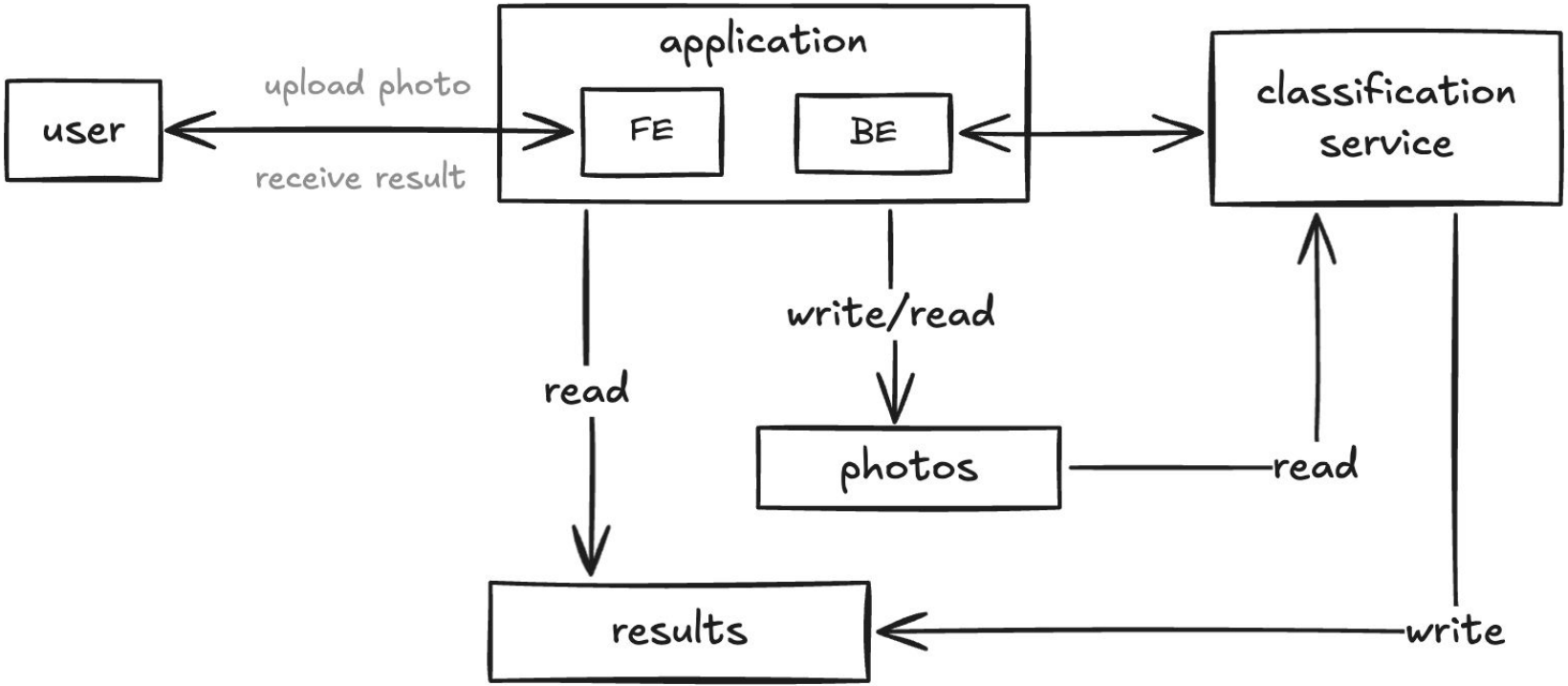
Design a system where users can upload a photo and receive classification what's in it—a cat, a dog, a car, etc.



# Photo Classification



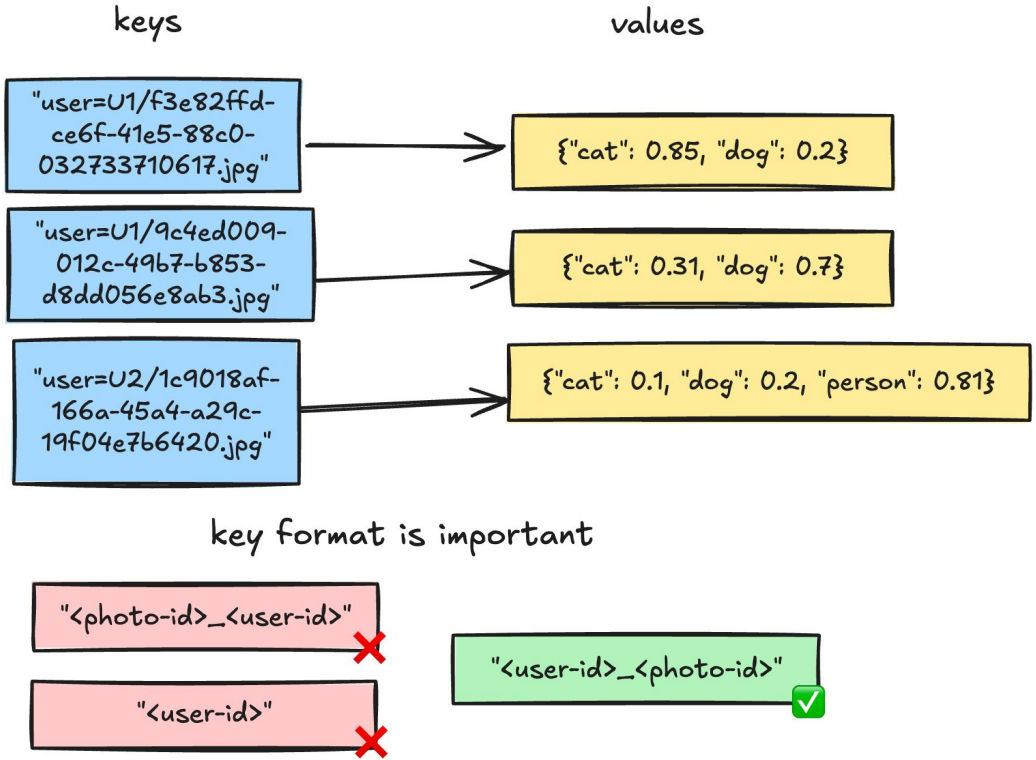
# Photo Classification



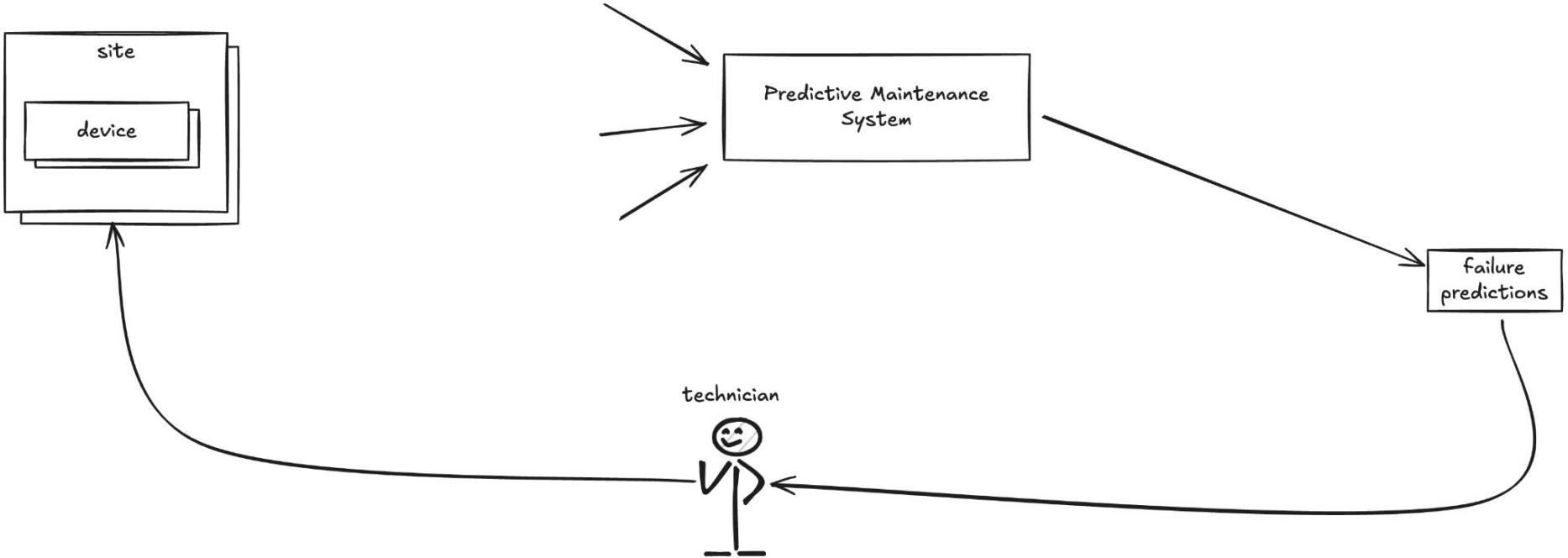
# Photo Classification

photos as files/blobs

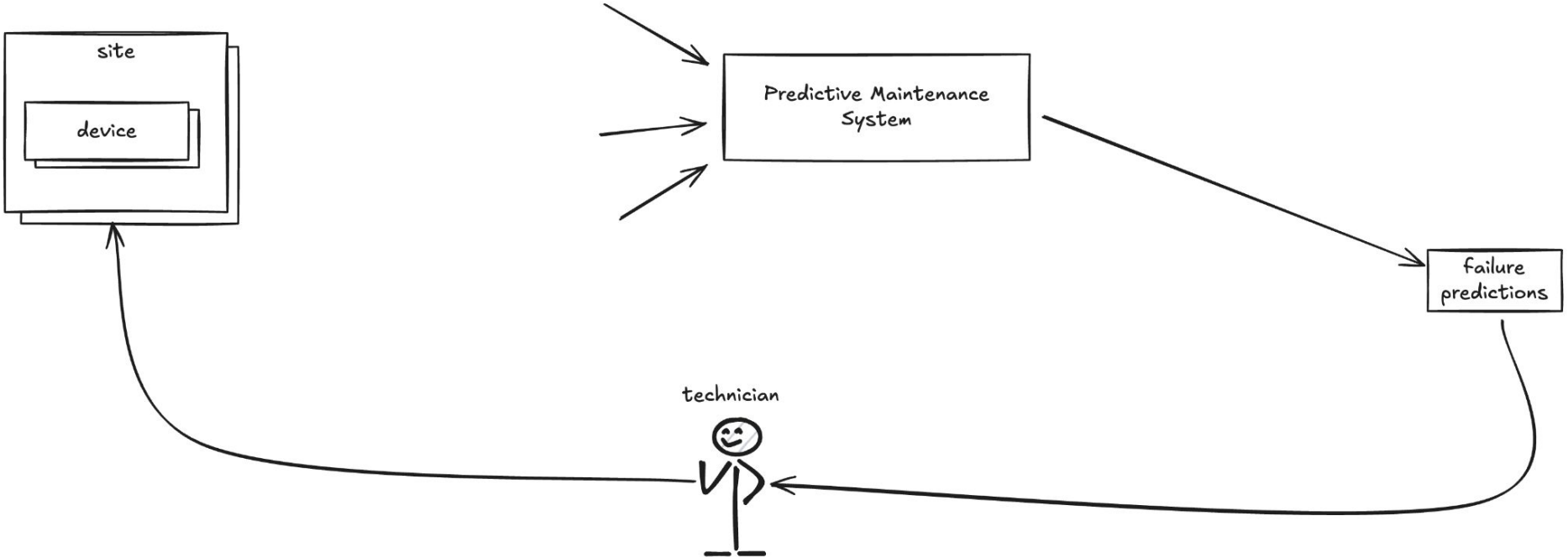
```
user=U1/
    f3e82ffd-ce6f-41e5-88c0-032733710617.jpg
    9c4ed009-012c-49b7-b853-d8dd056e8ab3.jpg
    1a8e8493-9eb5-45d5-ba5b-7d37d0180d60.jpg
user=U2/
    edc2a012-9b45-4a02-8c54-75c24034fd81.jpg
    6479e119-a924-42b6-9ecd-b4ada0409eb1.jpg
    1c9018af-166a-45a4-a29c-19f04e7b6420.jpg
user=U3/
user=U4/
```



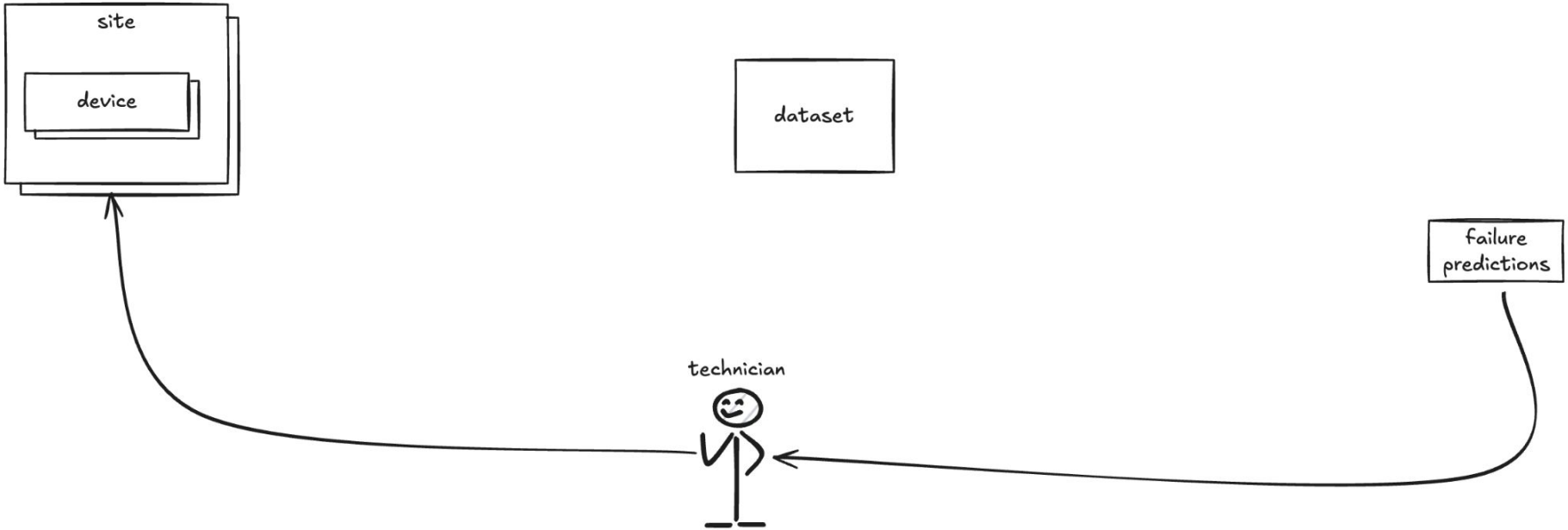
# Predictive Maintenance



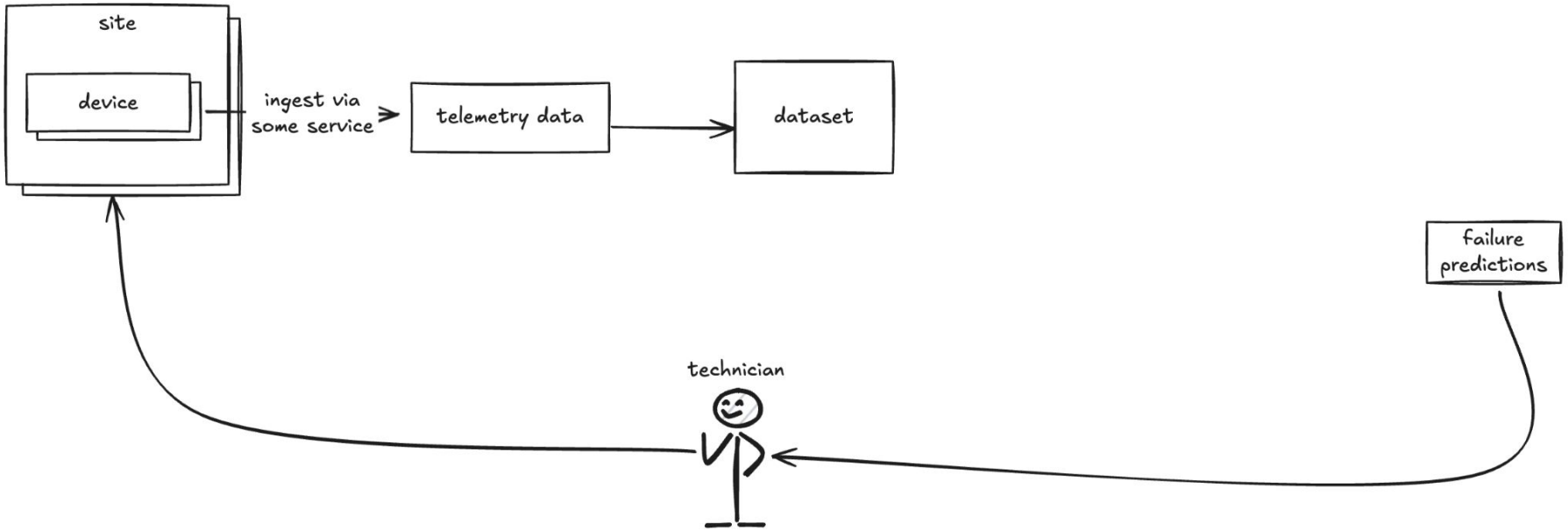
# Predictive Maintenance



# Predictive Maintenance

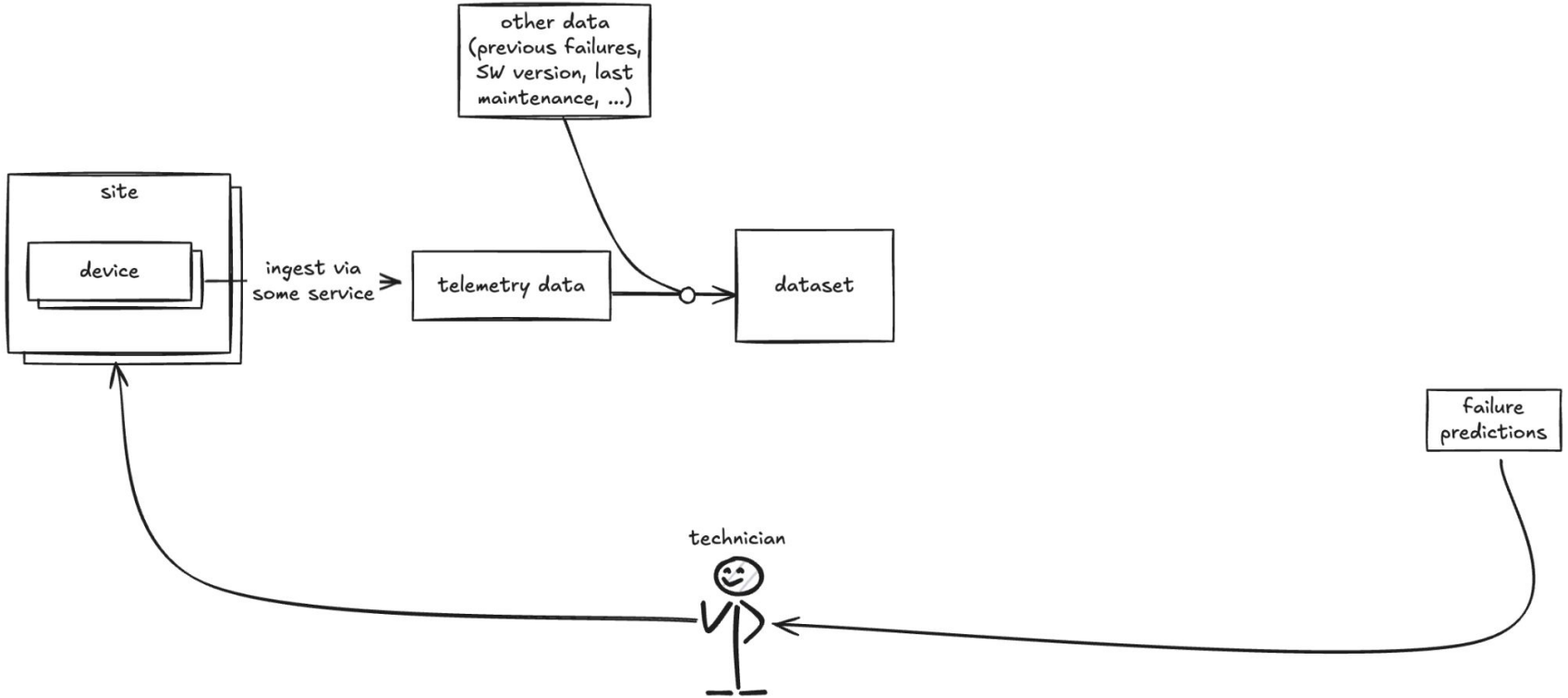


# Predictive Maintenance

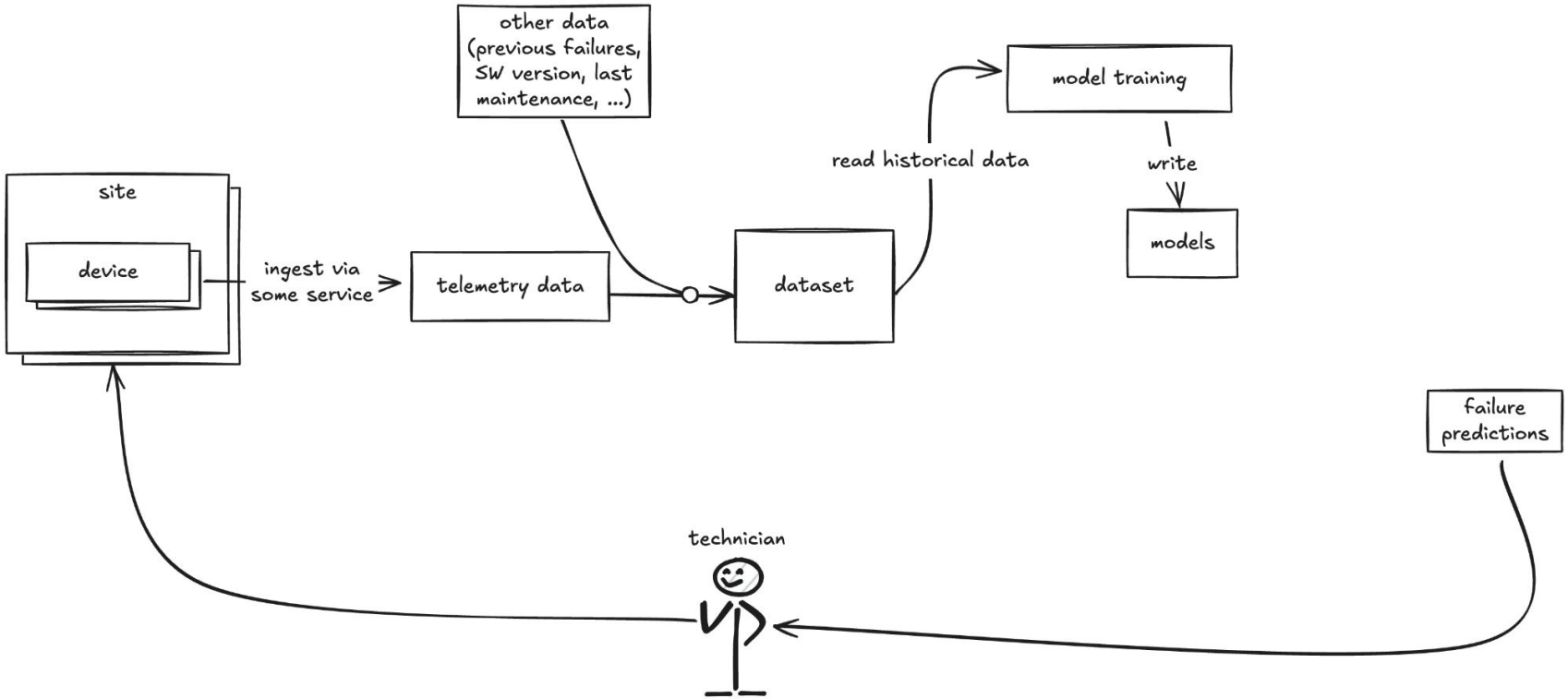



DATAMOLE

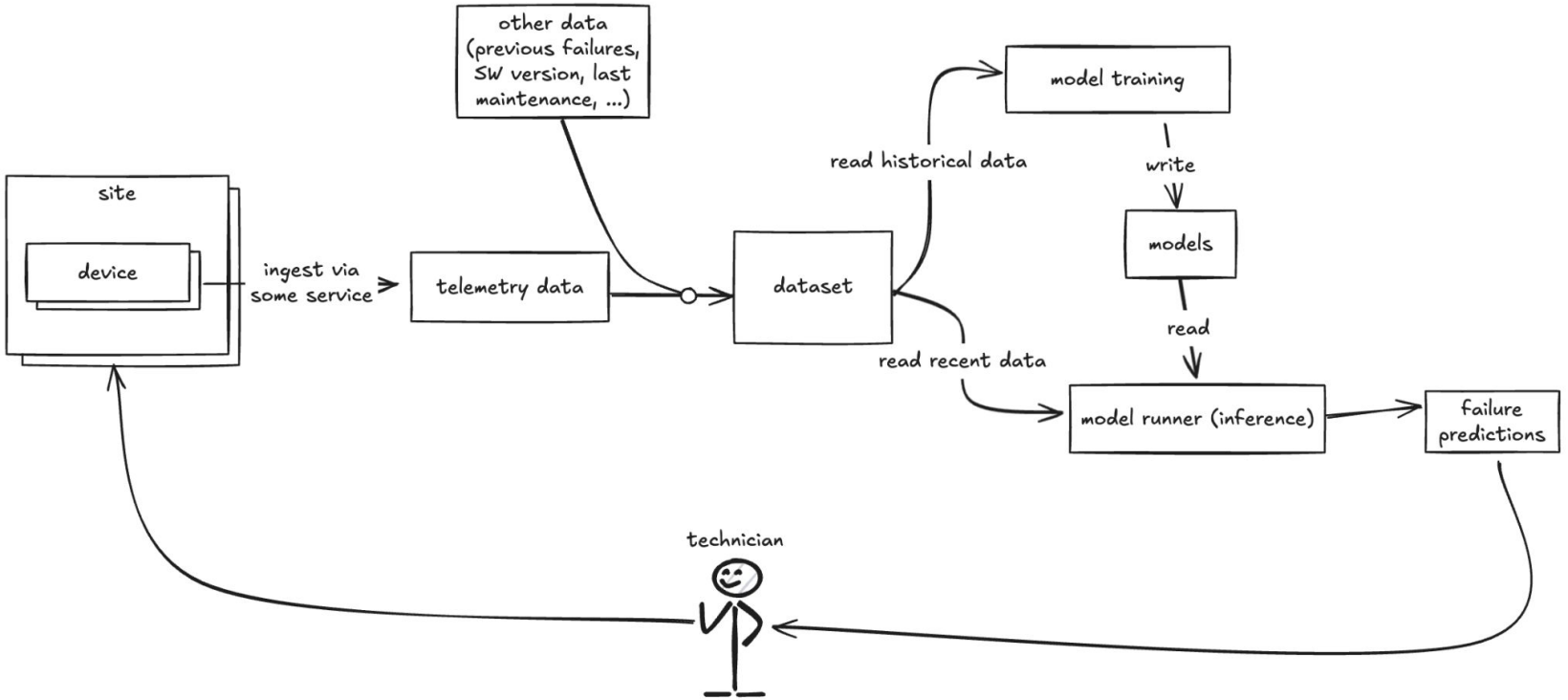
# Predictive Maintenance



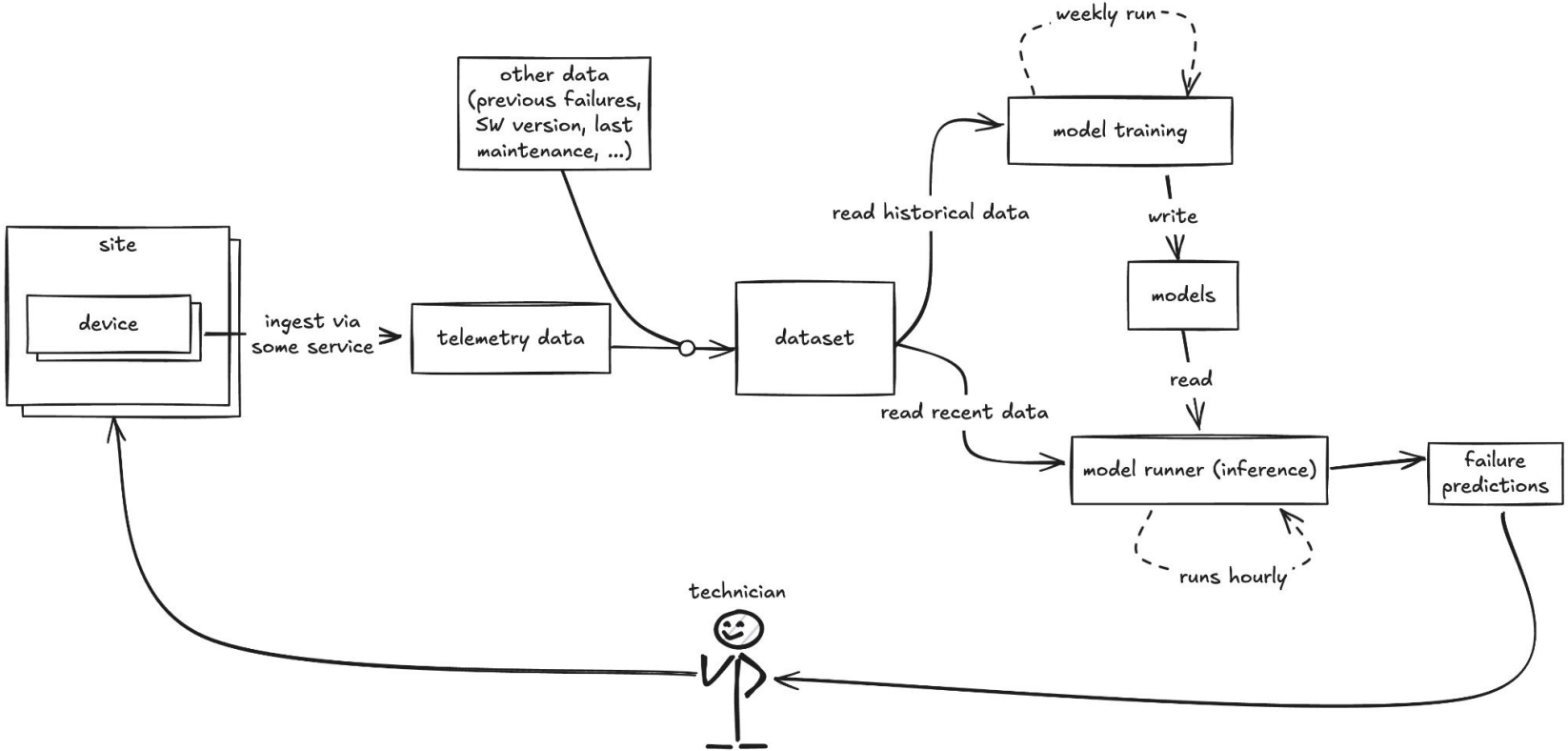
# Predictive Maintenance



# Predictive Maintenance



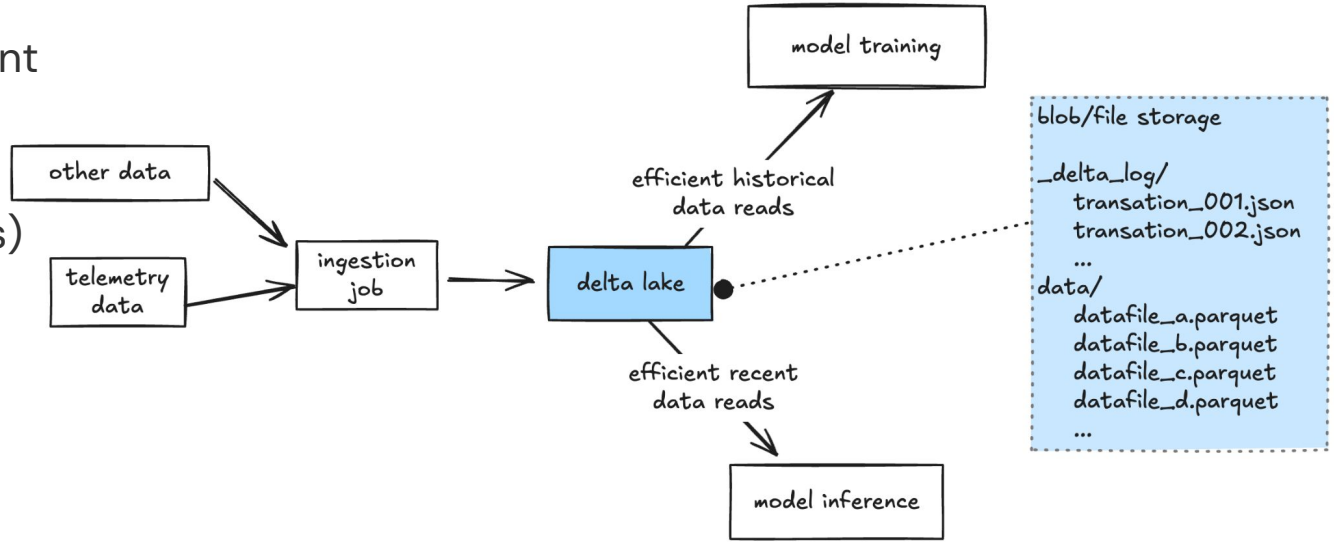
# Predictive Maintenance



# Predictive Maintenance

## Delta Lake

- ACID (per table)
- Time-travel
- Schema enforcement and evolution
- Efficient querying
- Scalable (petabytes)



# End words

- Not covered
  - model storage
  - graph DBs
  - timeseries DBs
  - …

- No silver bullet
- Don’t worry to make mistakes – but learn from them
- Continuous learning