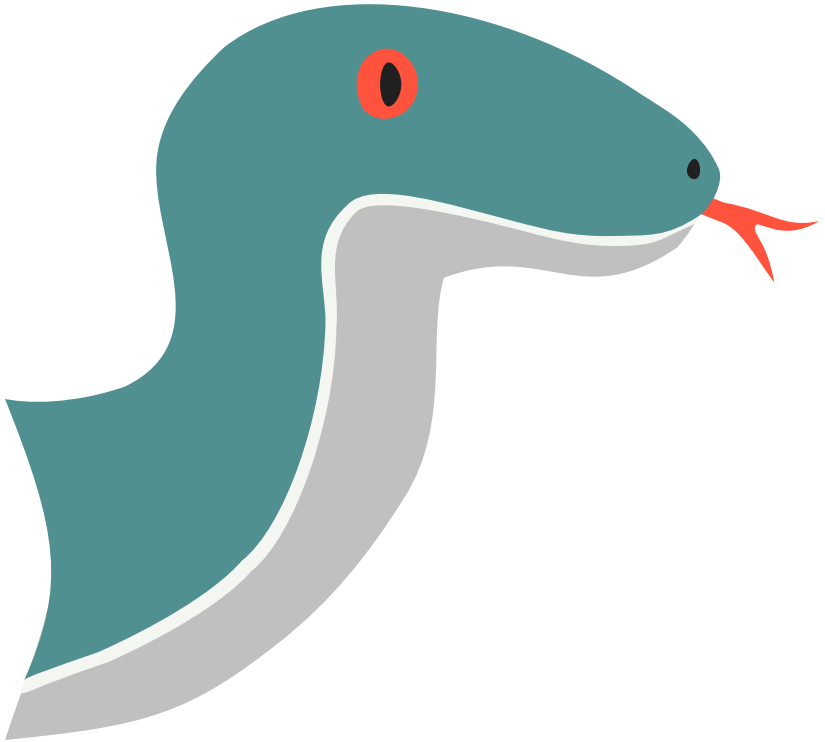


SNAKEDEV

All of Data Engineering in Three Hours

Operational Analytics Club

Summer Community Days

About Me

Pete Fein

- Trainer / Consultant / Coach
- Independent consultant for 12 years.
- 20 years in Python & Data.
- Writing a book: *Principles of Data Engineering* (Pearson).
- CTO for search-engine startup.
- I’m old.

Overview

- Bird's eye view of all of data engineering.
- Learn how it all fits together.
- A few practical tidbits.
- Something for everyone.
- Focus on analytics, not data science.
- Your head will explode - stay cool!
- Got a question? Use Q&A (moderated).

Summary

- Overview of Data Engineering
- Components of the Data Stack
- Data Operations
- Data Architecture
- Q&A after

Business Use Cases

- Sales & marketing
- Transportation logistics
- Large system monitoring
- Fraud detection
- Telephone
- Large scale billing
- Pure sciences, physics, & chemistry
- Mobile apps
- IOT

Optimize decisions!

Kinds of Data

- Log files
- Sensor data
- Images
- Numeric
- Short strings
- Long unformatted texts
- Web analytics & Customer Data Platforms (CDPs)
- GPS & satellite imagery
- APIs as a data source

Related Fields & Jobs

- Data Scientists
- Machine Learning Engineers
- Data Analysts
- Analytics Engineers
- ML Ops
- Data Ops

Related Fields & Jobs

- Data Scientists
- Machine Learning Engineers
- Data Analysts
- Analytics Engineers
- ML Ops
- Data Ops

What do Data Engineers do all day?

A day in the life of a data engineer ...

- Write SQL on Snowflake.
- Deploy Jupyter notebooks.
- Build Spark pipelines.
- Author Airflow workflows.
- Build Grafana dashboards.
- Go to meetings.

Core Concepts

- Data systems are derived systems.
- Analytics, not transactions.
- Big data, medium data, really big data, high velocity.
- Reports, analytics, predictions.
- Everything has a time dimension.
- Aggregates v. searches
- Batch v. streaming.

My Approach

- Bring software engineering best practices to the data space

Goals of Software Engineering

- Verifiable: Provably correct. "Is it doing what it's supposed to?"
- Reliable: Operate smoothly and handle problems gracefully.
- Reproducible: Re-run code with consistent results.
- Maintainable: Upgradable as external software environment changes.
- Extensible: Add new features in the future.
- Scalable: Grow to more data and developers.
- Reusable: Use code beyond its original purpose.
- Cost Effective: Minimize overhead and operate efficiently.

Recommended book: Practices of the Python Pro

Data Engineering Enables Self Serve

Level	Profile	Language	UX
Level 0	Sales team, CEO	None	Dashboard, PDF
Level 1	Data Analyst	Excel	BI User
Level 2	Analytics Engineer	SQL, YAML	BI Developer
Level 3	Data Scientist	Pandas	Jupyter Notebook
Level 4	Data Engineer	Python	Flask

See: ADAS for self-driving cars.

Poll

What level of self serve are you?

1. Level 0: Dashboards, reports
2. Level 1: Excel, BI user
3. Level 2: SQL, YAML, BI creator
4. Level 3: Pandas, Jupyter Notebook
5. Level 4: Python, Flask

Batch Processing

- Data processed in discrete chunks.
- Runs are sequential.
- Like laundry!

Streaming

- Continuous flow of data.
- Continuous ingest is common (change data capture).
- Streaming analytics is hard.
- You don't need this.

Example: Amazon Kinesis, Kafka, Spark streaming

Real time: A definition

"If there's a human in the loop, it's not real time." - Pete Fein

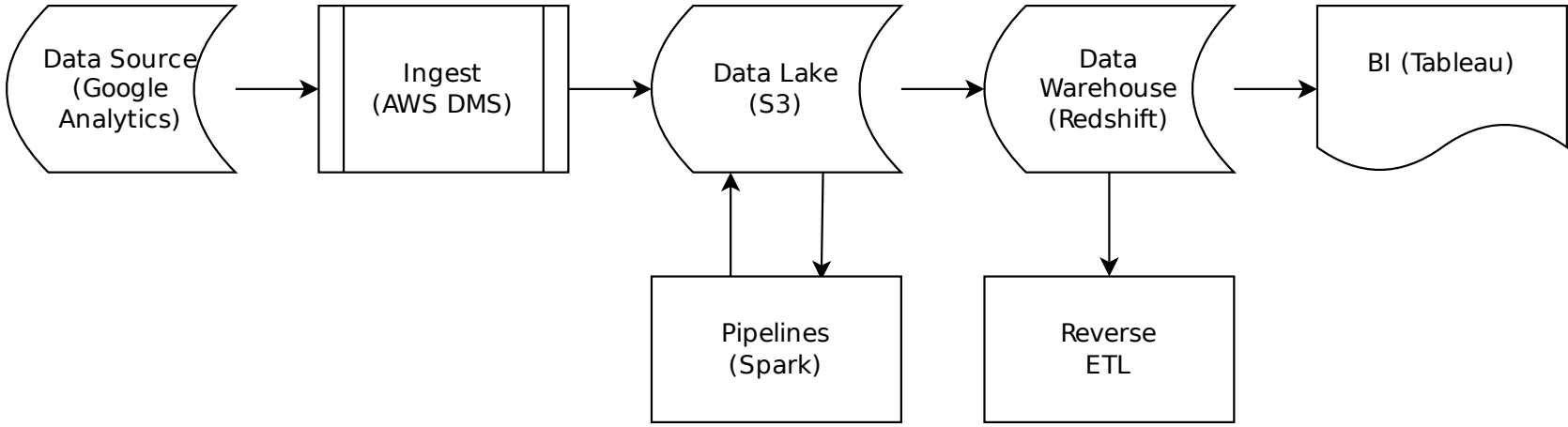
Build with Open Source

- Free (as in speech).
- Free (as in beer)!
- Transparency enables understanding and debugging.
- Add your own features.
- Community support.
- Hosted open source is great (SaaS).
- No vendor lock-in or shutdowns
    - You can still end up stuck with abandonware though.

BREAK

Data Stack

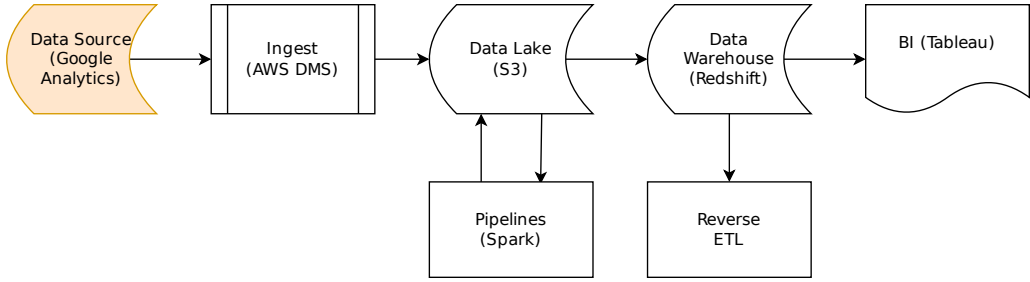
- Acquisition
- Ingest
- Data lakes
- Data warehouses
- Data pipelines
- Outputs
- Reverse ETL
- Orchestration



Acquisition

Turning external events into data.

- CDP: Segment, Snowplow, Honeypot.io.
- Web analytics: Google Analytics.
- Server logs & metrics: Fluentd, Datadog, statsd.
- Web scraping & crawling.
- Byproduct of transactional systems.
- Everything else - Bespoke Python.
- Sometimes data is just handed to you.

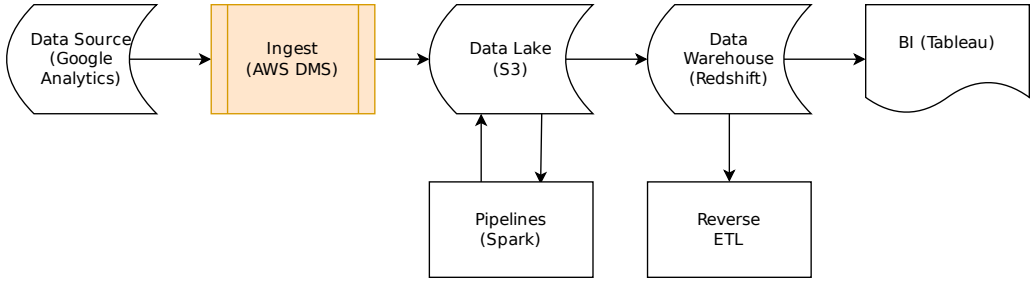


Ingest (ETL)

- Get data into platform.
- (EL) in ETL or ELT.
- Connectors move data from source to destination.
- Raw data with minimal cleanup.
  - Absolutely essential
  - Rebuild in case of errors
  - Supports changes in future
- Use native loaders when possible.

Examples: Load CSV files on S3 into Redshift

Tools: Fivetran, Snowpipe, Meltano

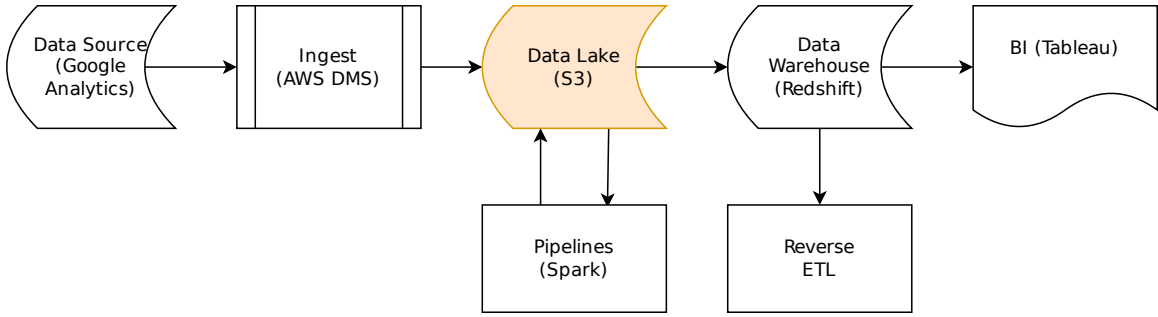


Data Lakes

A puddle of unstructured data.

- Landing zone for incoming raw data.
- Original source of truth for rebuilding.
- Source & destination for processing pipelines.
- Scratch space.

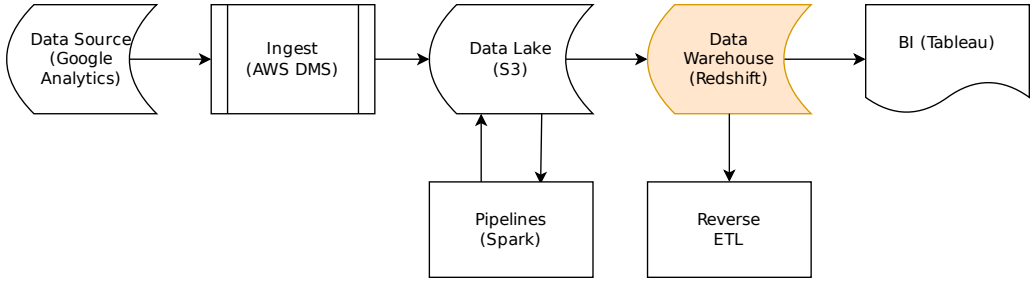
Example: CSV files on S3



Data Warehouses

- Workhorse of data platform.
- SQL-driven transformations.
- Transform layer for cleanup/normalization & analytics.
- Storage layer for most BI & dashboards.
- dbt: Templated SQL enables modularity and reusability. A game changer.

Examples: AWS Redshift, Google BigQuery, Snowflake



Transform Data with dbt

SQL without dbt:

```
select
order_id,
sum(case when payment_method = 'bank_transfer' then amount end) as bank_transfer_amount,
sum(case when payment_method = 'credit_card' then amount end) as credit_card_amount,
sum(case when payment_method = 'gift_card' then amount end) as gift_card_amount,
sum(amount) as total_amount
from raw_payments
group by 1
```

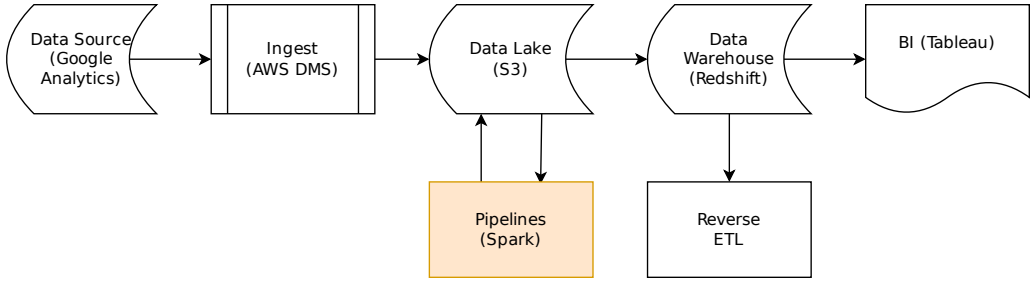
SQL with dbt: Use a 'for' loop in models for repeated statements:

```
select
order_id,
{% for payment_method in ["bank_transfer", "credit_card", "gift_card"] %}
sum(case when payment_method = '{{payment_method}}' then amount end) as {{payment_method}}_amount,
{% endfor %}
sum(amount) as total_amount
from {{ ref('raw_payments') }}
group by 1
```

Data Pipelines

- Catch-all for non-SQL processing.
- Python scripts, dataframe APIs.
- Data lake is typicaly used for storage.
- More complex data processing.
- Where Machine Learning happens.
    - Model inference & training.
- Frequently a result of productionizing of a Jupyter Notebook.

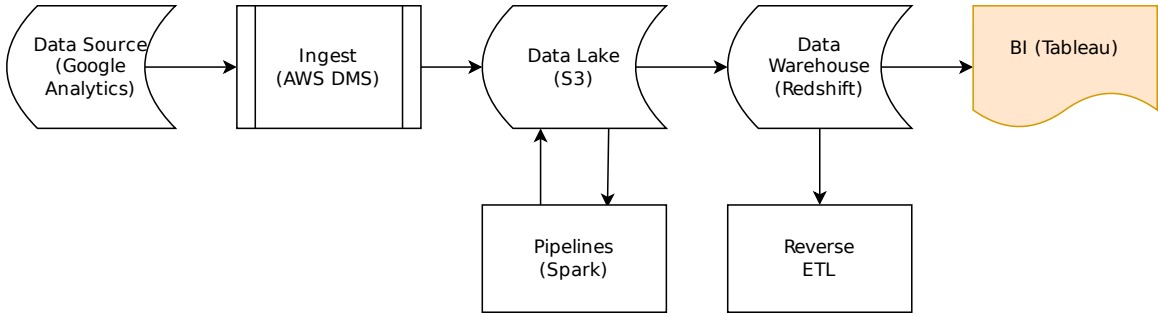
Examples: Spark, pandas, scikit-learn, R



Outputs

- Human-usable results of a data platform
- Static reports: PDF, Excel
- Dashboards
- Business Intelligence: interactive & exploratory
- Exports: CSV & JSON
- Machine learning models
- Metrics store

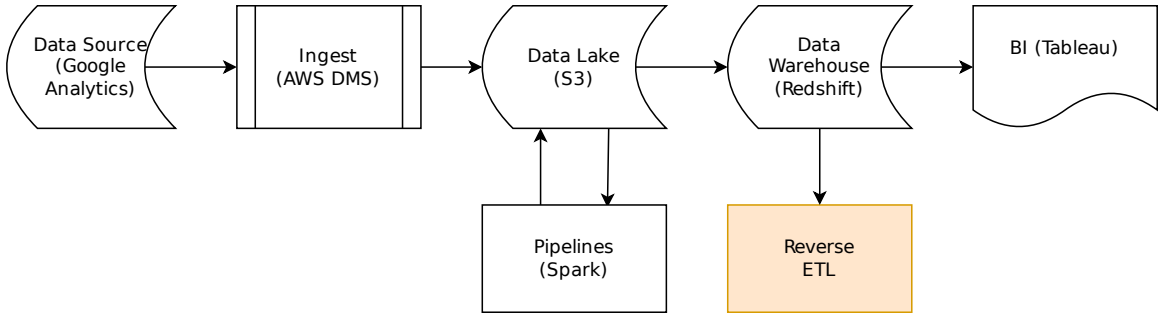
Examples: Tableau, Power BI, Grafana, Apache Superset



Reverse ETL

- Move beyond analytics to operationalize data.
- Close the loop.
- Examples:
  - Feed hot leads back to Salesforce.
  - Ad targeting.
- This is hard!

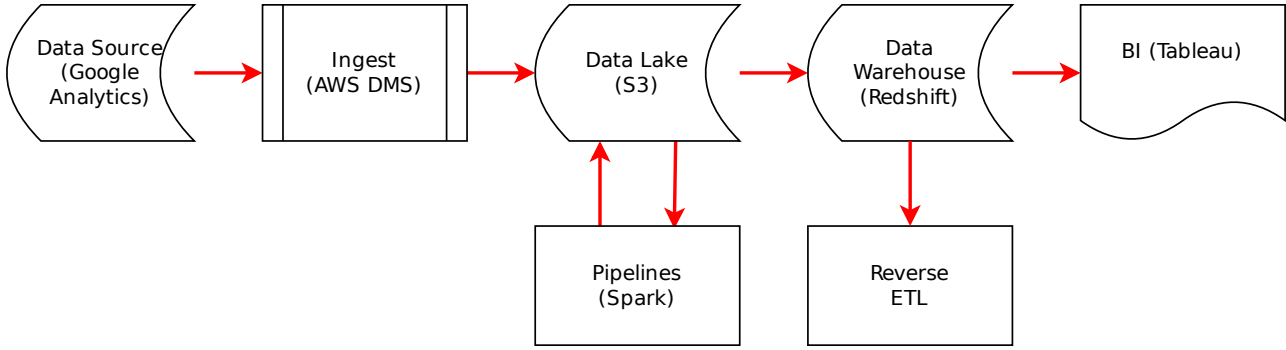
Solutions: Census, Grouparoo (defunct), hand-rolled Python



Orchestration

- How does anything happen?
- The arrows in a data diagram.
- Drives action in the system.
- Recipe for baking a data cake.
- Events that initiate jobs: passage of time, arrival of new data, manual runs.
- Avoid: Cron jobs, mess of lambdas, step functions.

Examples: Apache Airflow, Astronomer, Prefect, Dagster



Poll

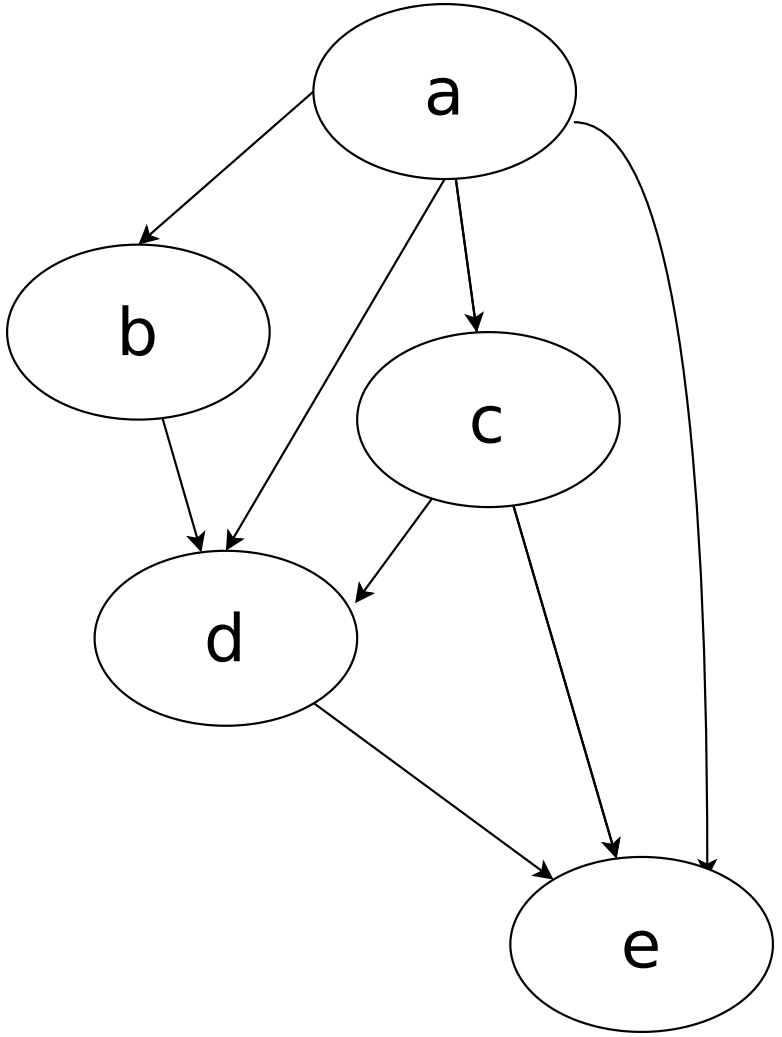
How do you run your data jobs?

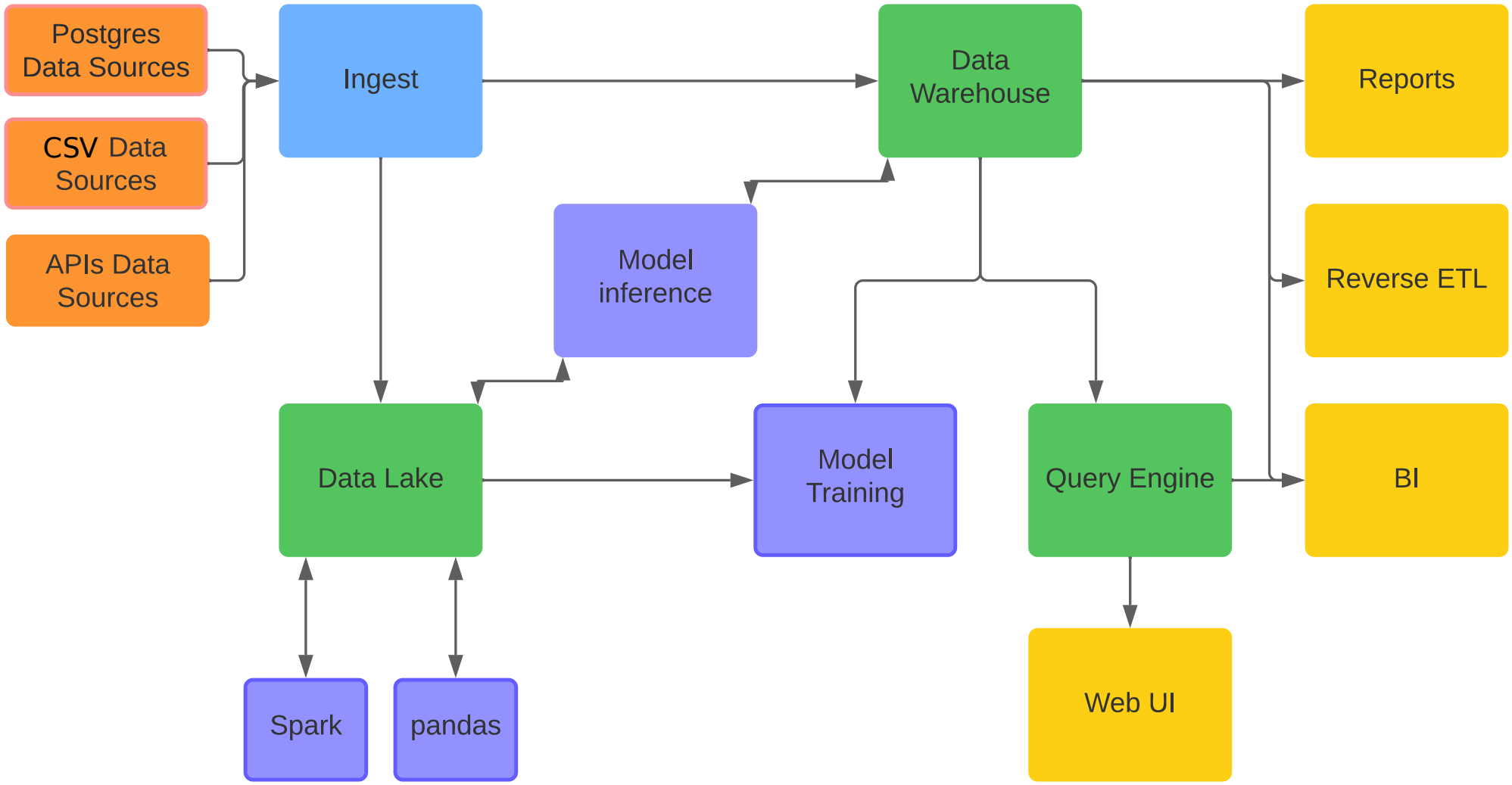
1. cron with scripts
2. By hand
3. Step functions and Lambdas
4. Airflow
5. Prefect
6. Other

What’s a DAG?

- Directed Acyclic Graph (DAG).
- Core organizational structure in data systems.
- Defines how data moves & the sequence of operations performed.
- Nodes & edges connected in a single graph without cycles (loops).

Examples: Supply Chain, dbt models, pip





Design Principles

- Rivers and refineries.
- Do the expensive things once.
- Data is immutable and immortal.
- Match queries to storage.

BREAK

Data Platform in Practice

- Quality & observability
- Testing
- Documentation
- Governance
- Debugging & optimization
- Operations

Quality & Observability

- Bad data is toxic.
- Quality: Predefined rules for correctness.
- Observability: Detecting outliers and weird stuff without judgements.
- Solution: checks after every processing step.
- Data service-level agreements (SLAs) for accuracy and timeliness.

Example: Great Expectations, dbt-expectations, re_data

Testing

- Ensure correctness when code changes.
- Unit testing is hard.
  - Complex stacks.
  - Large datasets.
  - No right answer.
- Fixtures: Presaved datasets for tests in CI/CD.
- Use data-diff to compare databases.

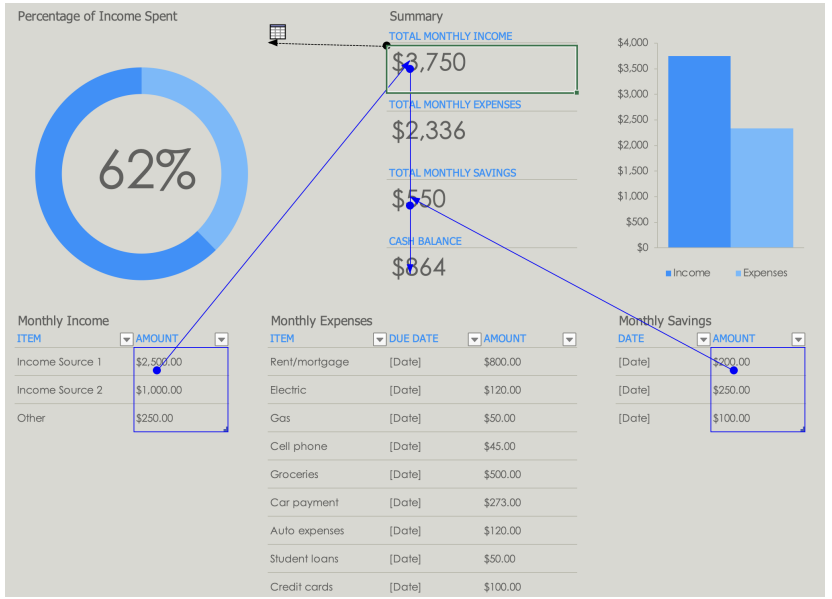
Poll

What is dbt-test used for?

1. Testing
2. Quality
3. Observability

Documentation

- Beyond data dictionaries.
- Catalogs enable discoverability.
- Lineage, a DAG to figure out where the information came from.
  - Sarbanes-Oxley accountability.
  - Plain language and ML explainability.
- Communicate data status in dashboards/BI.



Governance

- Who can see which rows and which columns.
- Role-based access control.
  - Users (∞ → 1) Groups
  - Permissions (∞ → 1) Roles
  - Roles GRANTed to Groups
- Permafrost: YAML-based permissions for Snowflake

Debugging & Optimization

- Debugging is hard.
- Benchmarking, cost estimation, capacity planning.
- When optimizing, be a scientist!

Operations: DevOps Not ClickOps!

- ClickOps: Devops by clicking around in AWS console.
- Error-prone and not reproducible.
- No support for multiple environments.
- "What's running in prod?"
- Solution: Automate Everything
  - *Everything* must be automated.
  - All code and config lives in Git.
  - Deploy with CI/CD on code pushes.

Tools: Terraform, GitHub Actions, dbt Cloud

Poll

How do you deploy/run your code?

1. Manually on cloud
2. On a laptop
3. dbt cloud
4. CI/CD

Dependency Pinning

- Reproducible installs.

`requirements.txt`

No !	Yes!
Tensorflow	Tensorflow==2.1.1
Pandas	Pandas==17.1
pytest	pytest==7.1.2

BREAK

Data Architecture

- Layouts
- The SQL deception
- Consistent data
- Modern modelling
- Warehouses compared

Layout: How to organize data

- You should have one.
- Put all data into a single bucket.
- Terminology in flux.
- Raw: Raw data.
- Core: Normalized & cleaned.
- Analysis: Derived data.

```
finance/                        marketing/
  raw/                            raw/
  core/                           core/
  analysis/                       analysis/
```

OLAP v. OLTP

OLAP (warehouses)	OLTP (relational)
No transactions	Transactional
Few writes, huge reads	Many writes and reads
Columnar oriented	Row oriented
Few users	Many users
Internal	External
Long running	Many short queries
Queries/BI	Web & Mobile Apps

Do not use data warehouse to drive public-facing apps!

Row v. Column

- Match format to query types.
- Maybe use both.

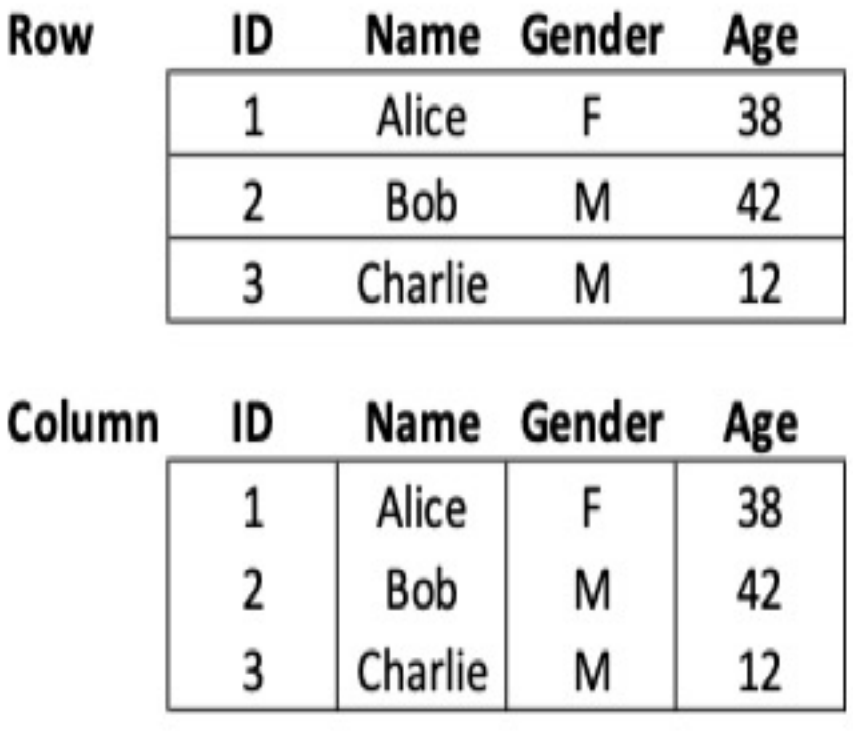
Row (CSV, PostgreSQL)

```
1,Alice,F,38,2,Bob,M,42,3,Charlie,M,12
```

Column (Parquet, Snowflake)

```
1,2,3,Alice,Bob,Charlie,F,M,M,38,42,12
```

Consequence: JOINs are slow on data warehouses.



Poll

What's the best format to: a) Calculate average age? b) Find names of men?

1. Row-oriented best for both.
2. Column-oriented best for both.
3. Columns for average age, rows for names.
4. Rows for average age, columns for names.

SQL Deception

- SQL is a family of declarative programming languages with many dialects and implementations.
- Imperative v. Declarative languages
  - Imperative: You write and run program to calculate data.
  - Declarative: Describe data you want and let database engine find it for you.
- All SQL is not the same!
  - Underlying execution models differ significantly.
  - Performance varies dramatically for the same query on different databases.
  - You **must** read and understand what it does in your database.

Consistency Without Transactions

- How do you present a correct & consistent view of data?
- Batch pipelines are long and jobs can fail midway through.
- Solution: Blue/Green deploys (borrowed from Web apps).
    i. Current production point to Blue environment.
    ii. Build Green environment.
    iii. Verify Green environment.
    iv. Point production to Green environment.
    v. Next run uses Blue: flip back and forth.
- CLONEs are lightweight copies for databases (Snowflake & BigQuery).

Pragmatic Modelling

"Premature optimization is the root of all evil." - Donald Knuth

- Model when needed.
- Model for business or technical reasons.
- Business
  - Logical entities that hang together.
  - Common foundation for reports, etc. across a company.
  - Metrics/KPIs.
  - Enable self serve by analytics engineers.
  - Make life easier for non-technical users.
- Technical
  - Extract models from common patterns in queries.
  - Code reuse.
  - Improve performance, reduce costs.

	Snowflake	BigQuery	Redshift
Overall	Amazing	Good	Poor
SQL dialect	Modern	Modern	Ancient
Pricing	Usage-based, expensive, and worth it.	Storage at Rest, data read.	Cluster size.
Scaling	Automatic	N/A	Difficult, slow
Workloads	Fully decoupled, manageable	Decoupled	Shared
Caching	Data	Queries	Limited, off by default

	Snowflake	BigQuery	Redshift
Partitioning	Automatic, excellent	Date time, Integer, or Ingest time	Any column, Auto available
Clustering	Automatic, excellent	Multiple column	Single column, Auto available
Ingest	Streaming or batch	Streaming or batch	Batch, limited streaming (slow commits)
External data	Excellent	Good, S3 supported	Slow, clunky

	Snowflake	BigQuery	Redshift
Clones	Yes	Beta	No
Indexes	Yes	Yes	No
Time travel	90 days, configurable	7 days, table only	No
Governance	Sophisticated: Row & column permissions, RBAC	IAM-based	Difficult
Administration	Everything in SQL	Some UI	UI required

Thanks to Datacoves for the preceding comparisons.

Good Old PostgreSQL

- Great warehouse for small to medium data (< 1B records).
- If you're not storing Web traffic or sensor data, this might be you.
- Operationally simple, familiar and powerful SQL dialect.
- Scale up with a really big single node.
- Cstore_fdw plugin for columnar store tables

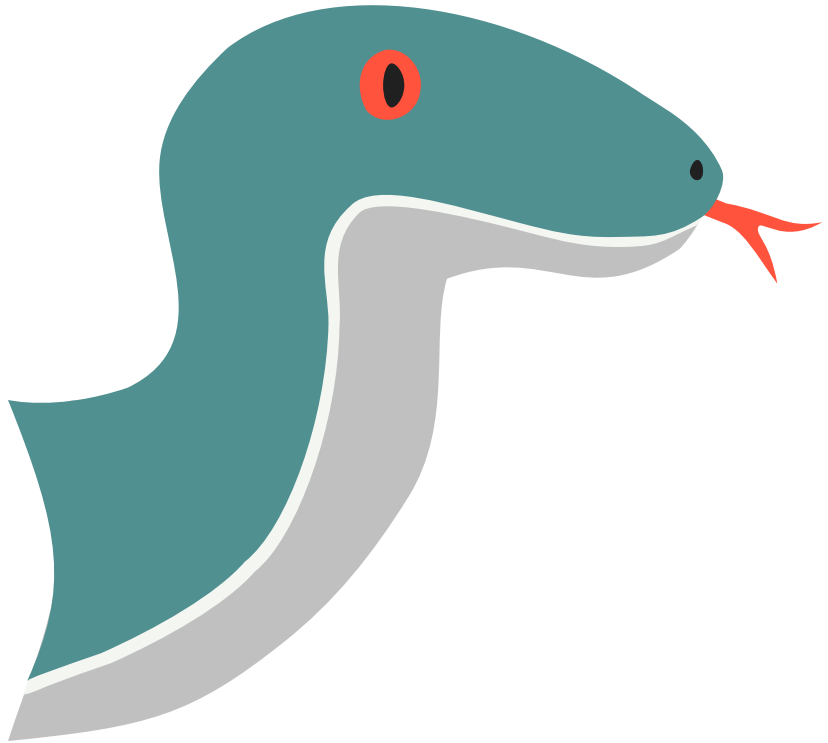
Itamar's Skepticism

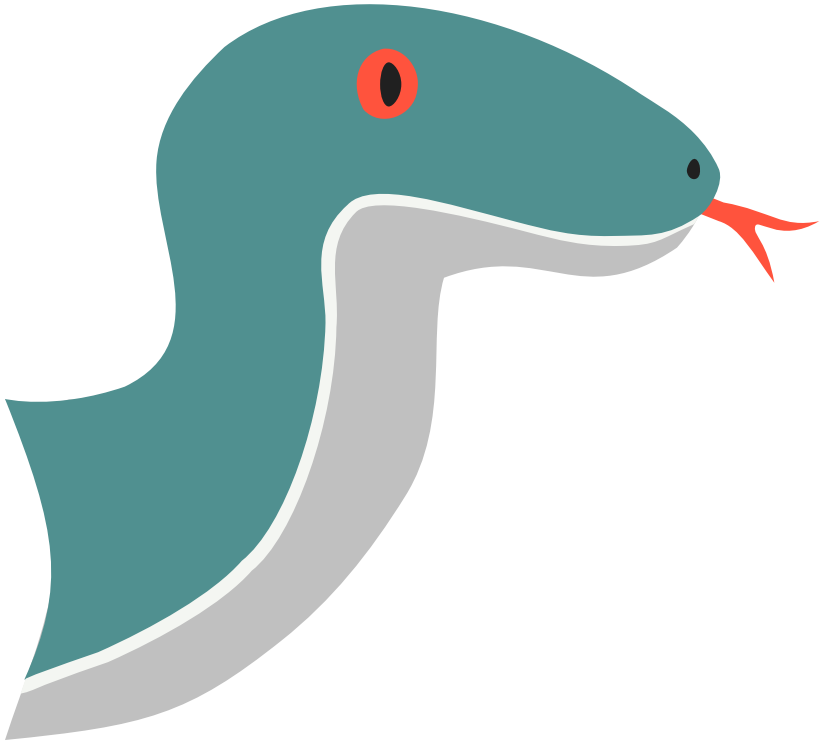
"You should use 1 computer or 100 computers, and be skeptical of anything in between."

- Itamar Turner-Trauring

Snakedev Workshops

- Acquisition & Ingest
- Data Warehouse Architecture
- Operational Data Platforms
- Orchestration with Airflow
- Data Quality & Testing
- Pipelines & Query Engines
- Notebooks in Production



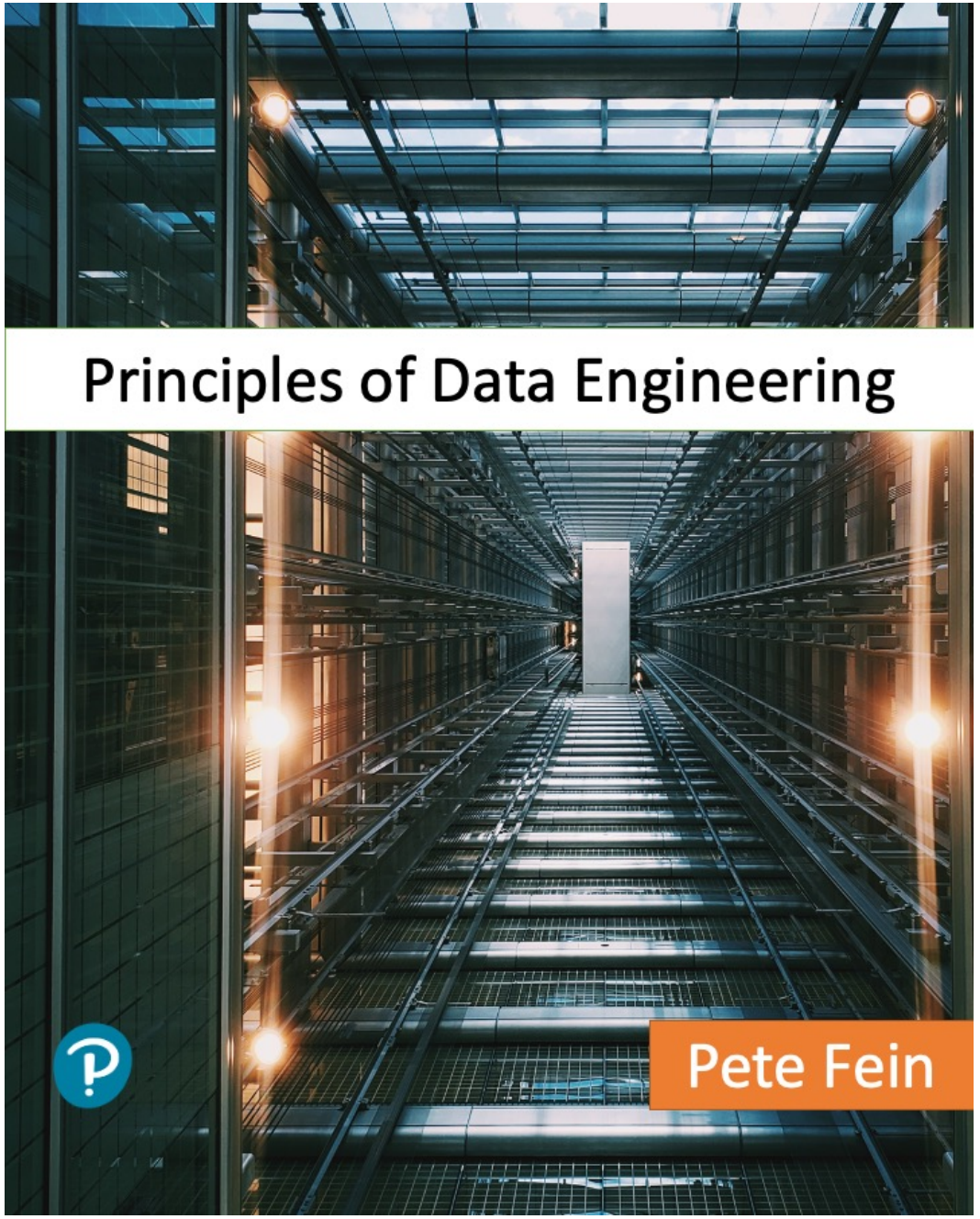


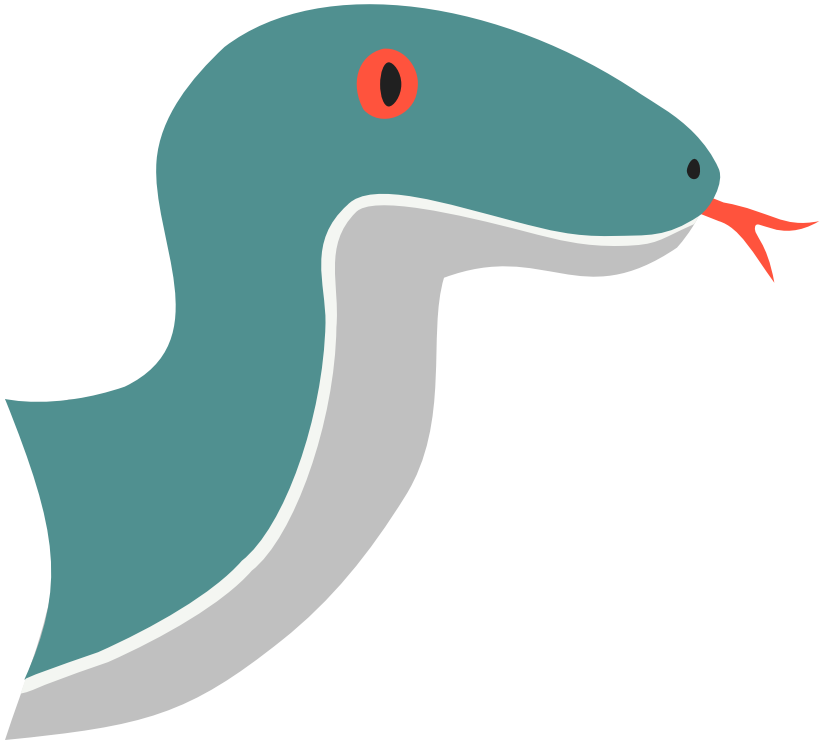
Snakedev Services

- Training
- Consulting
- Coaching
- Strategic Advisor
- Project Developer
- Optimization

Book!

- Principles of Data Engineering
- Published by Pearson (2023 publication date)





SNAKEDEV

All of Data Engineering in Three Hours

Pete@Snake.dev

https://snake.dev