The Modern Stack for ML Infrastructure

Ville Tuulos

outerbounds

The modern stack?

The stack?

The Evolution of Web Stacks

LAMP (1998)

The Evolution of Web Stacks

LAMP (1998)

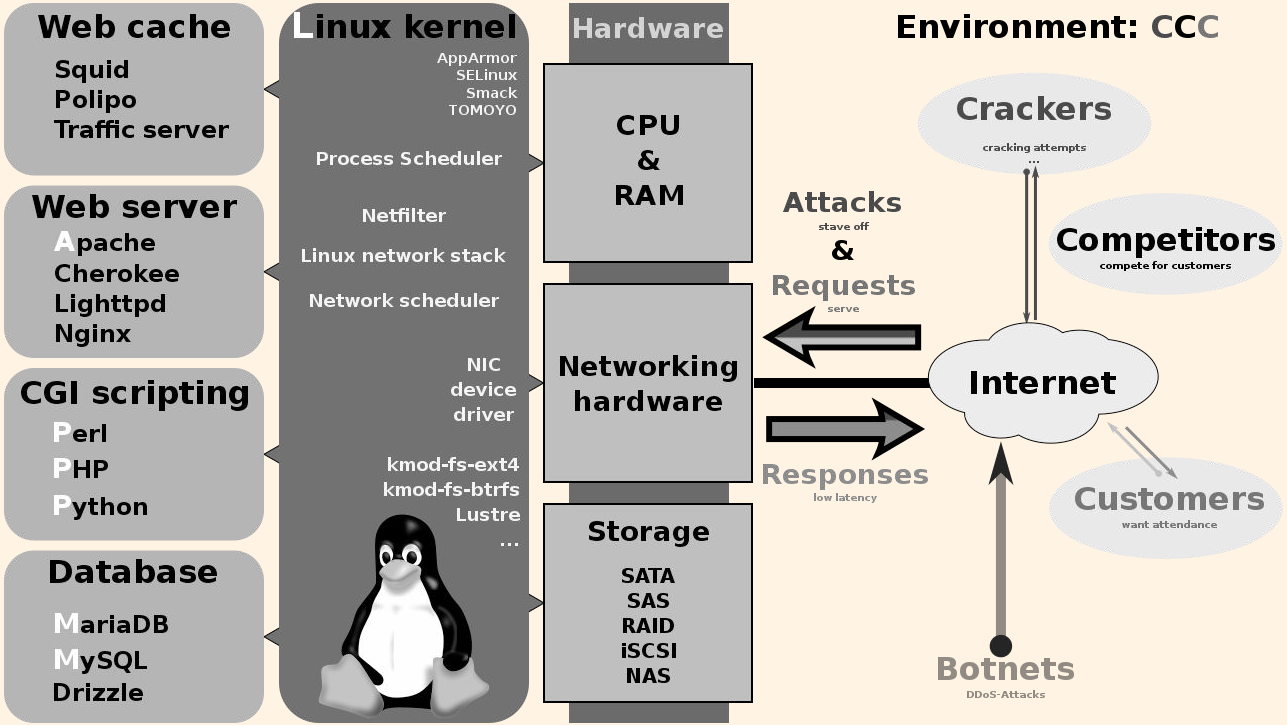


Figure by Shmuel Csaba Otto Traian / Wikipedia

The Evolution of Web Stacks

LAMP (1998)

MEAN (2013)

The Evolution of Web Stacks

LAMP (1998)

MEAN (2013)

JAM (2015)

The Evolution of Web Stacks

The stack becomes ***less technical, more human-centric*** 😊

LAMP (1998)

MEAN (2013)

Javascript

APIs

Markup

JAM (2015)

The Evolution of Web Stacks

*The stack becomes **simpler, more capable** over time* 💪

LAMP (1998)

MEAN (2013)

Javascript

APIs

Markup

JAM (2015)

The stack for ML infrastructure will become

simpler, more capable 💪

&

more human-centric 😊

The Evolution of ML Stack

The stack becomes ***less technical, more human-centric*** 😊

CLAM (1998)

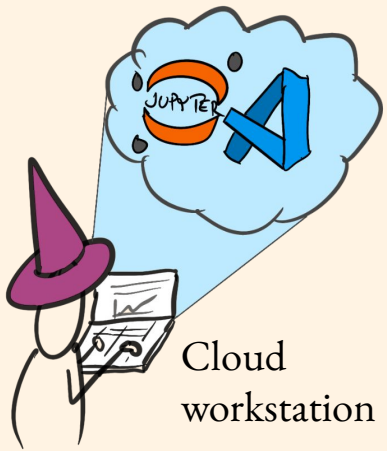
MLOps (2018)

Future?

Let’s design
a modern ML stack
from the ground up

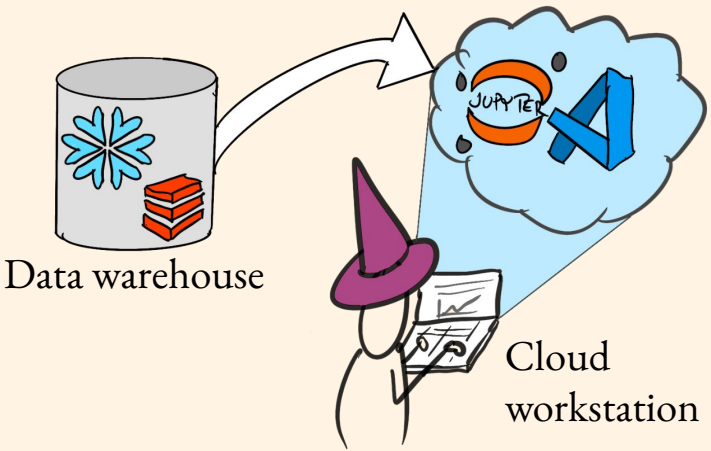
Here’s a data scientist

A modern data scientist uses a cloud workstation



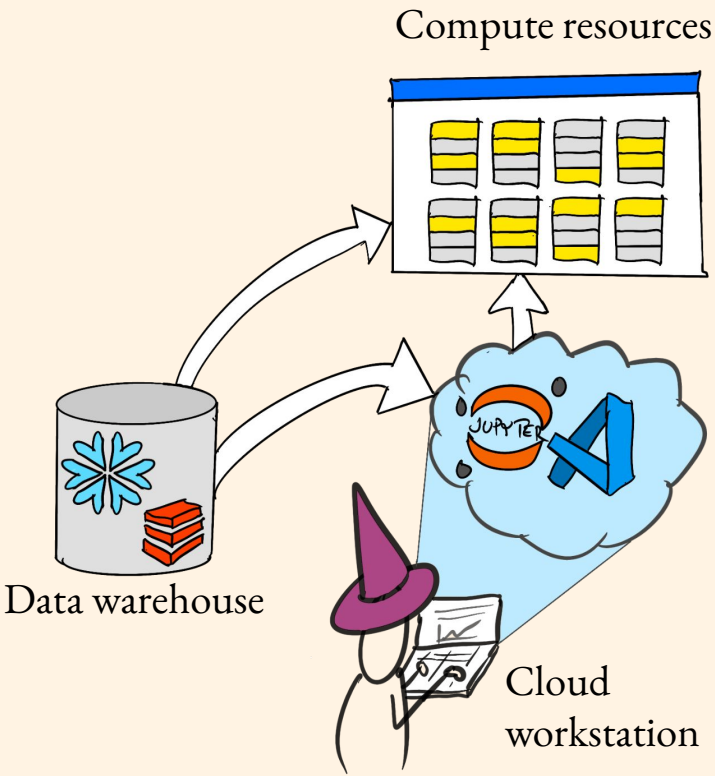
Cloud
workstation

Data flows seamlessly from the data warehouse to the workstation



Data

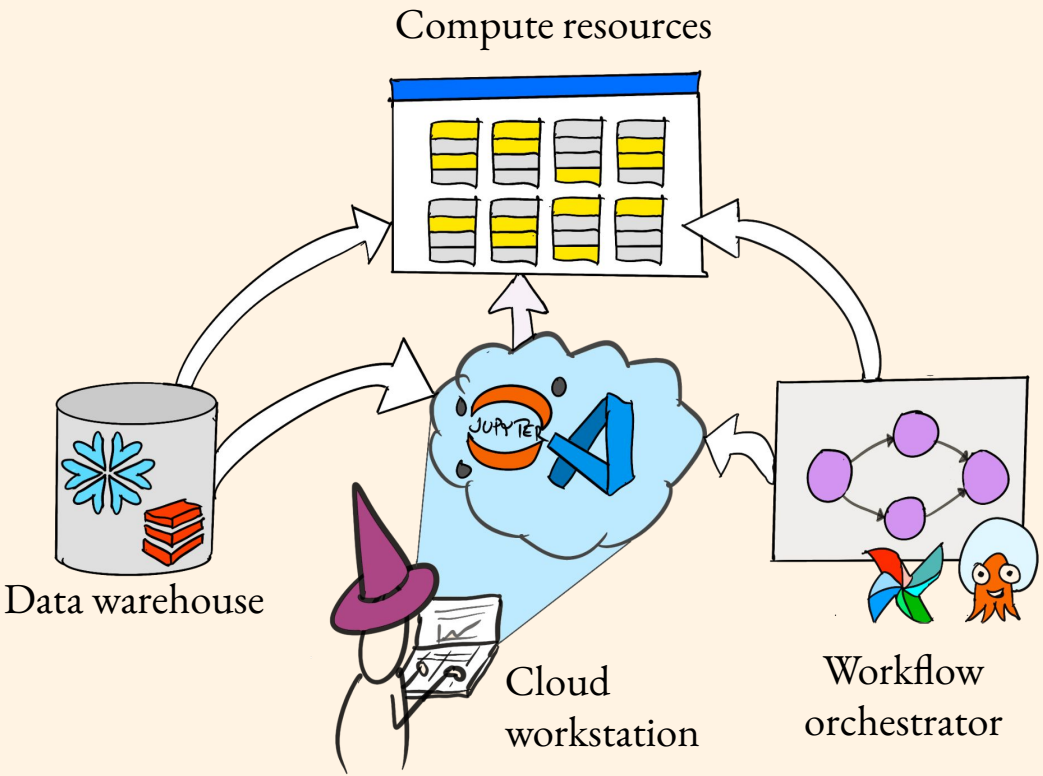
Experiments run at scale on a cloud-based compute cluster



Compute

Data

Complete workflows are developed and tested locally

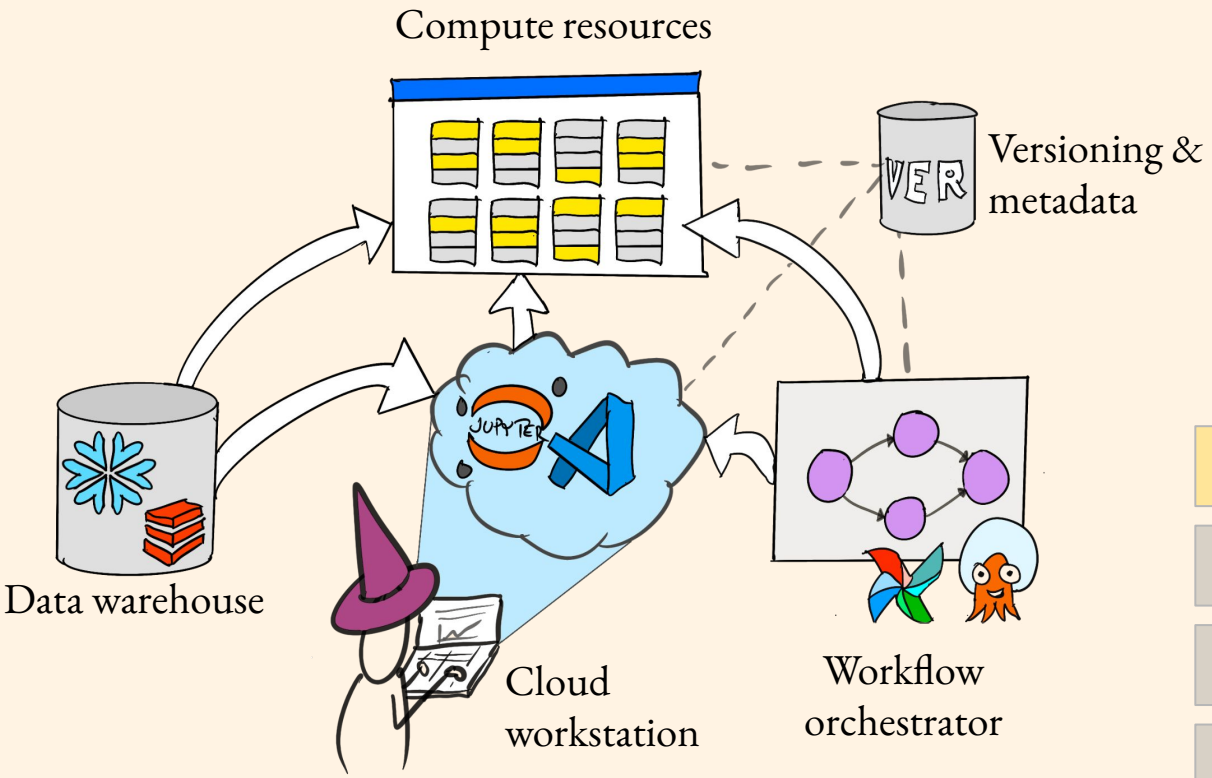


Orchestration

Compute

Data

Code, models, logs, and metrics gets stored and versioned automatically



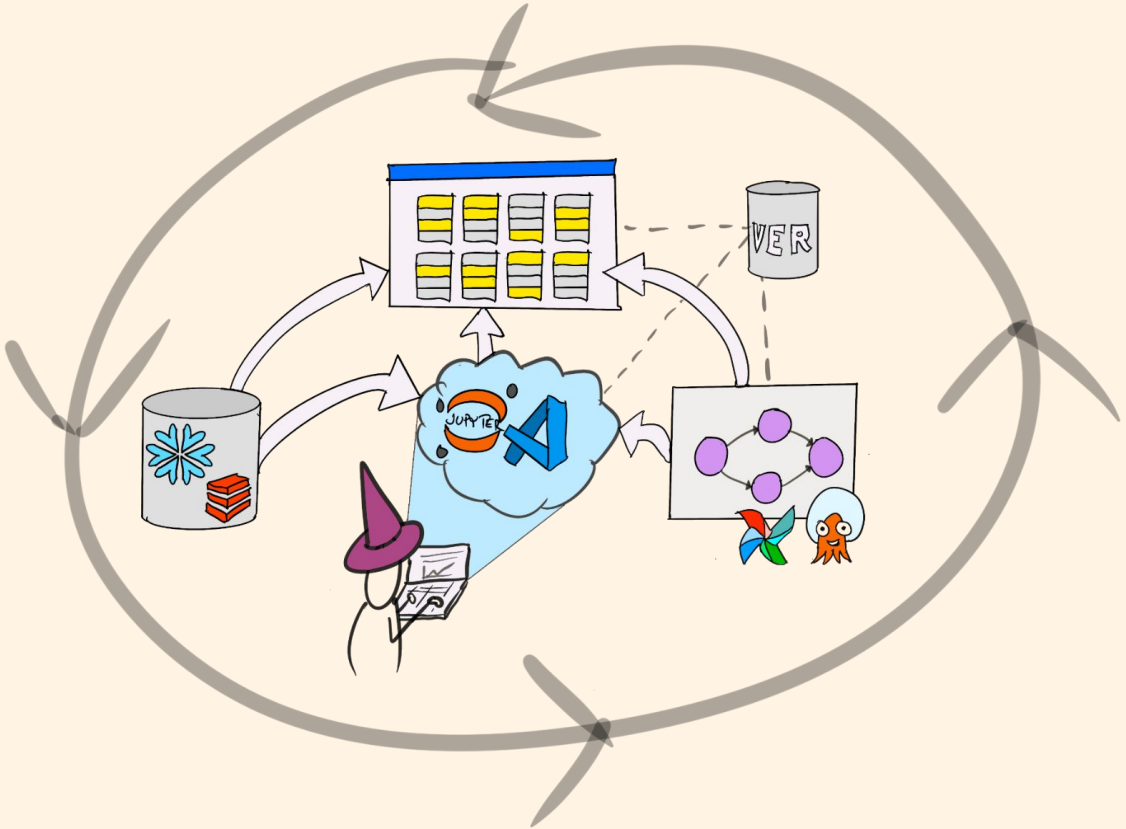
Versioning

Orchestration

Compute

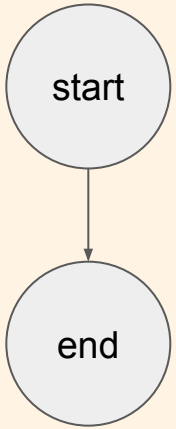
Data

Data Scientist can develop, test, and iterate on projects rapidly

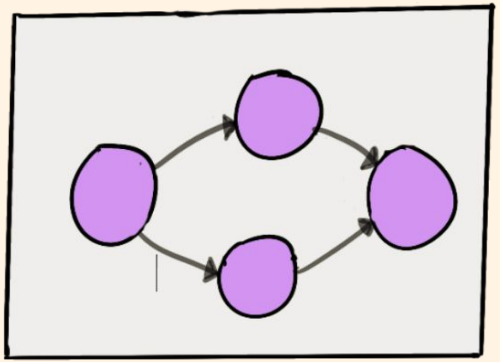


Example

Define workflows with a human-friendly syntax

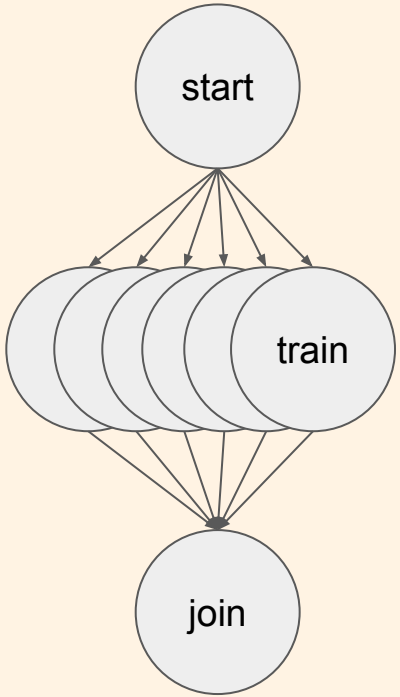



```python
class MyFlow(FlowSpec):

    @step
    def start(self):
        import pandas as pd
        pd.DataFrame(big_one)
        self.next(self.end)

    @step
    def end(self):
        pass
```

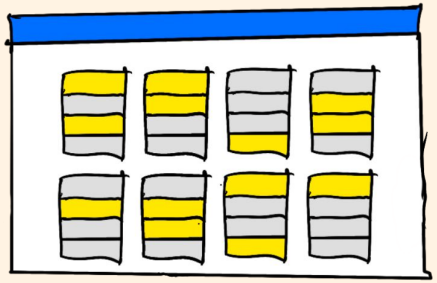


```
# python myflow.py run
```

Experiments run at scale on a cloud-based compute cluster

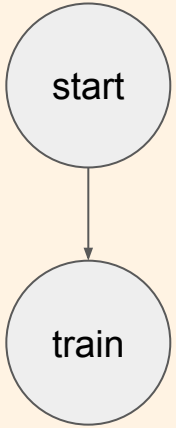



```
@step
def start(self):
    self.params = list(range(100))
    self.next(self.train, foreach='params')

@resources(memory=128000)
@step
def train(self):
    self.model = train(...)
    self.next(self.join)

@step
def join(self, inputs):
    ...
```



```
# python myflow.py run --with kubernetes
```

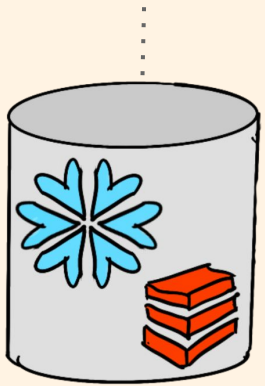
Everything gets versioned automatically



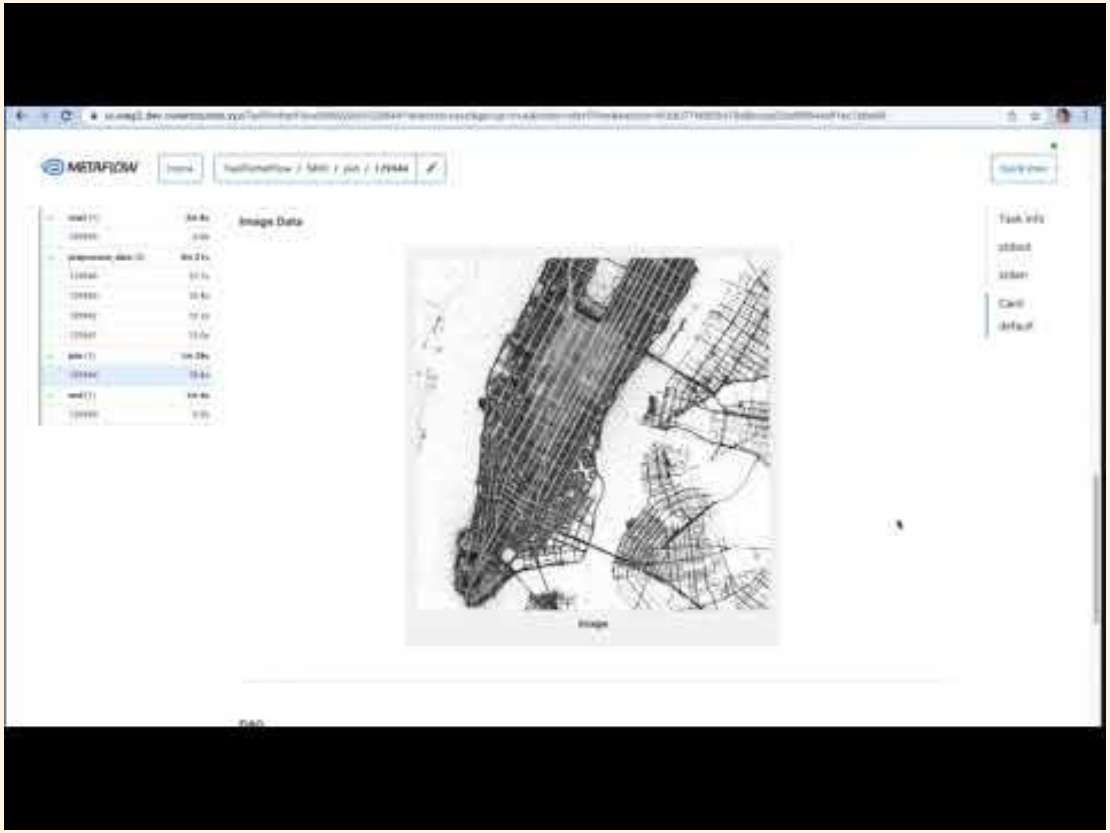

```python
class MyFlow(FlowSpec):

    @step
    def start(self):
        self.alpha = 0.5
        self.next(self.train)

    @step
    def train(self):
        self.model = train_model(self.alpha)
```

Comes with tools for fast data access

```
class QueryFlow(FlowSpec):
    @step
    def query(self):
        self.ctas = "CREATE TABLE %s AS %s" % (self.table, self.sql)
        query = wr.athena.start_query_execution(self.ctas)
        output = wr.athena.wait_query(query)
        loc = output['ResultConfiguration']['OutputLocation']
        with metaflow.S3() as s3:
            results = [obj.url for obj in s3.list_recursive([loc])
```



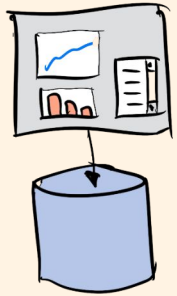
Data Scientist can develop, test, and iterate on projects rapidly



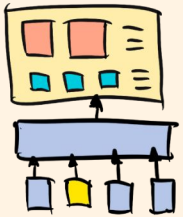
From prototype to Production

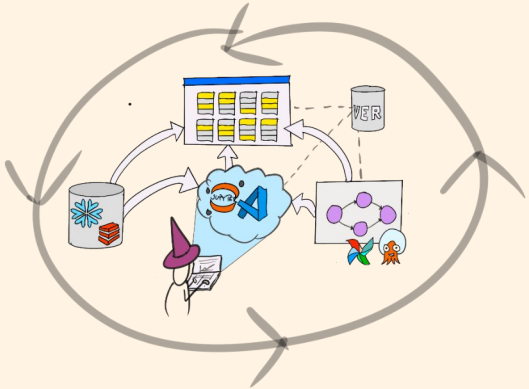
Real-world ML comes in many shapes and sizes

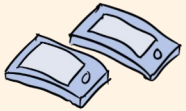
Decision-support systems



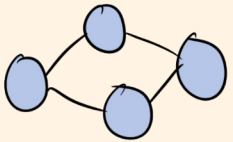
Product features







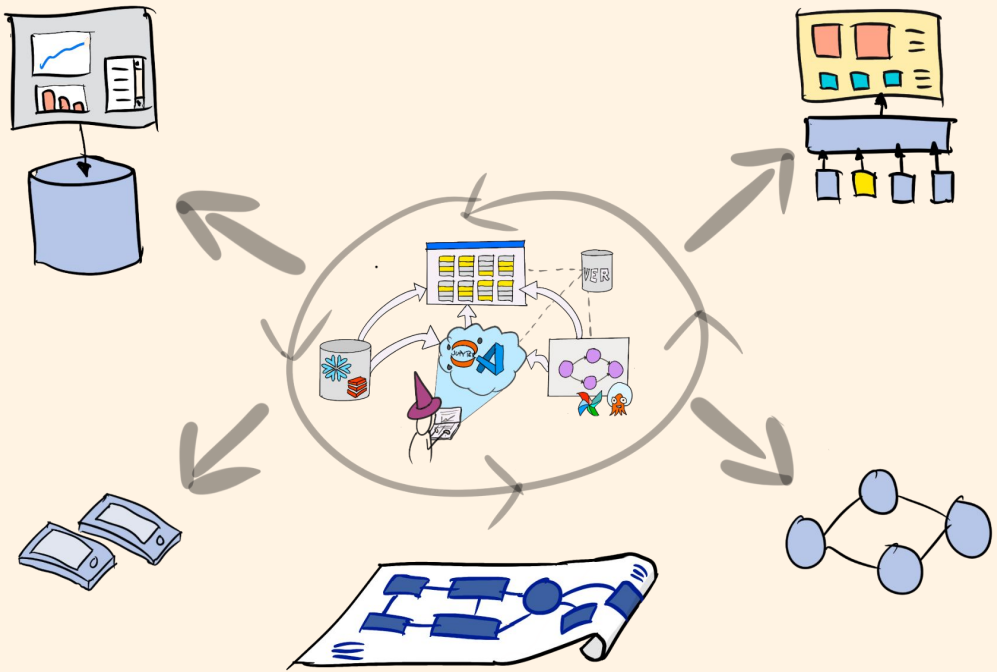
On-device ML



Data enrichment

There is not a single *production* but many

Provide architectural blueprints to support various deployment patterns

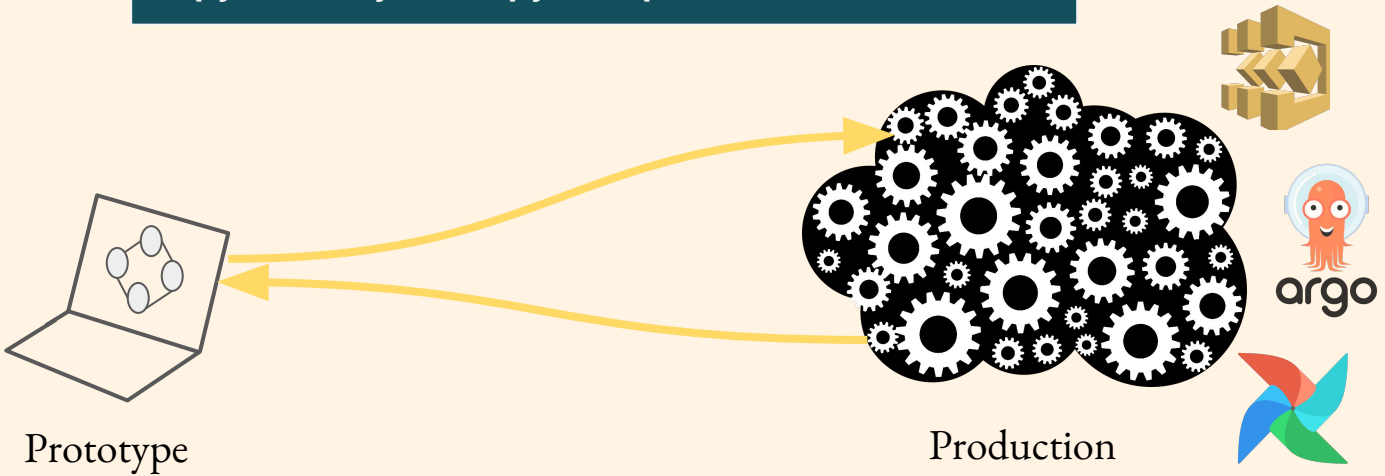


- Architecture
- Versioning
- Orchestration
- Compute
- Data

Metaflow Example

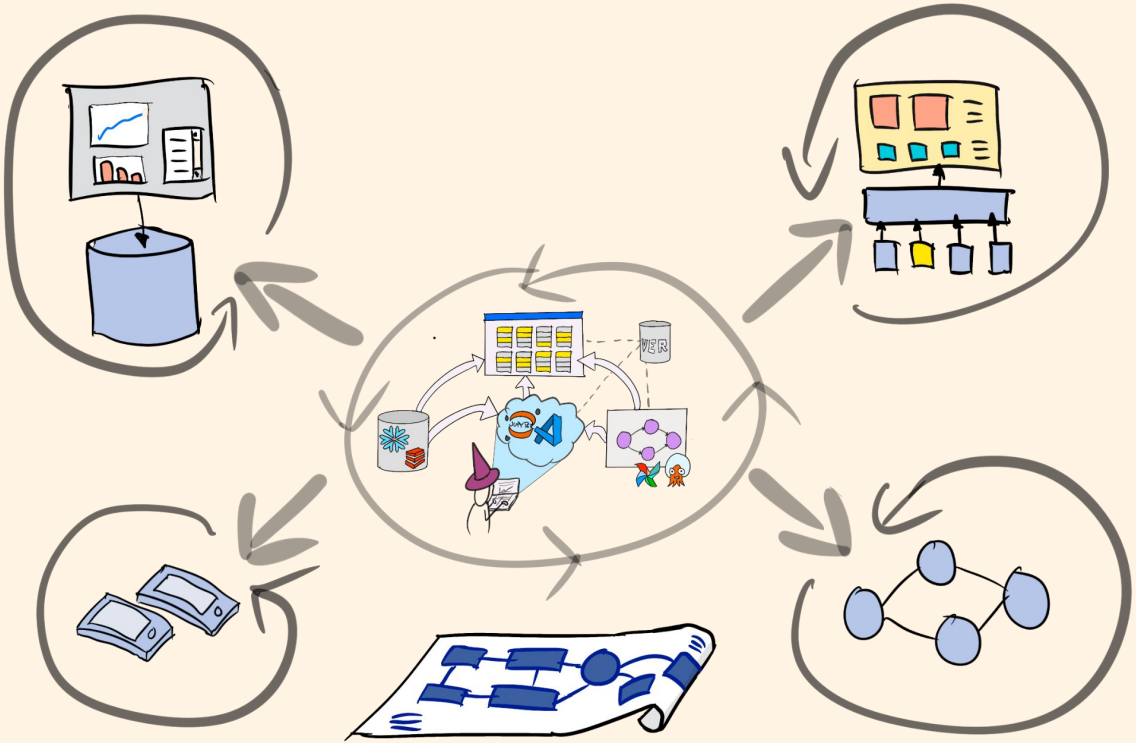
Single-click deployment (and back)

```
# python myflow.py step-functions create
```




```
# python myflow.py resume --origin-run-id sfn-199874
```

Continuous deployment, continuous experimentation



- Operations
- Architecture
- Versioning
- Orchestration
- Compute
- Data

Metaflow example

Deploy parallel models for A/B testing

Project: LTV

```
@project(name='LTV')
class TrainingFlow(FlowSpec):

    @step
    def start(self):
```



```
@project(name='LTV')
class PredictFlow(FlowSpec):

    @step
    def start(self):
```

```
# python myflow.py --branch a deploy
```

```
@project(name='LTV')
class TrainingFlow(FlowSpec):

    @step
    def start(self):
```



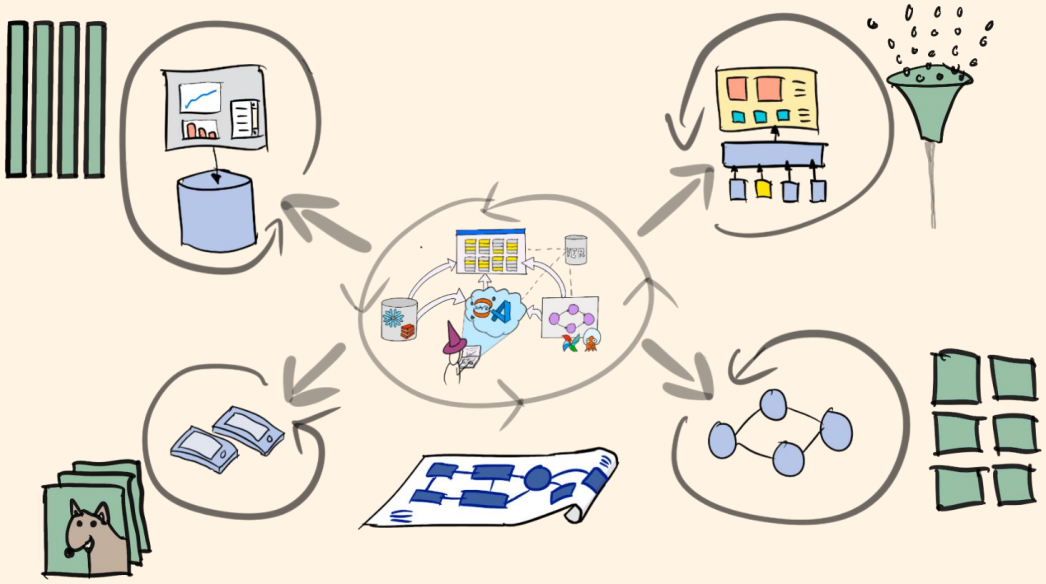
```
@project(name='LTV')
class PredictFlow(FlowSpec):

    @step
    def start(self):
```

```
# python myflow.py --branch b deploy
```

Data scientists can experiment with features flexibly…

Relational data

Streaming events



Images

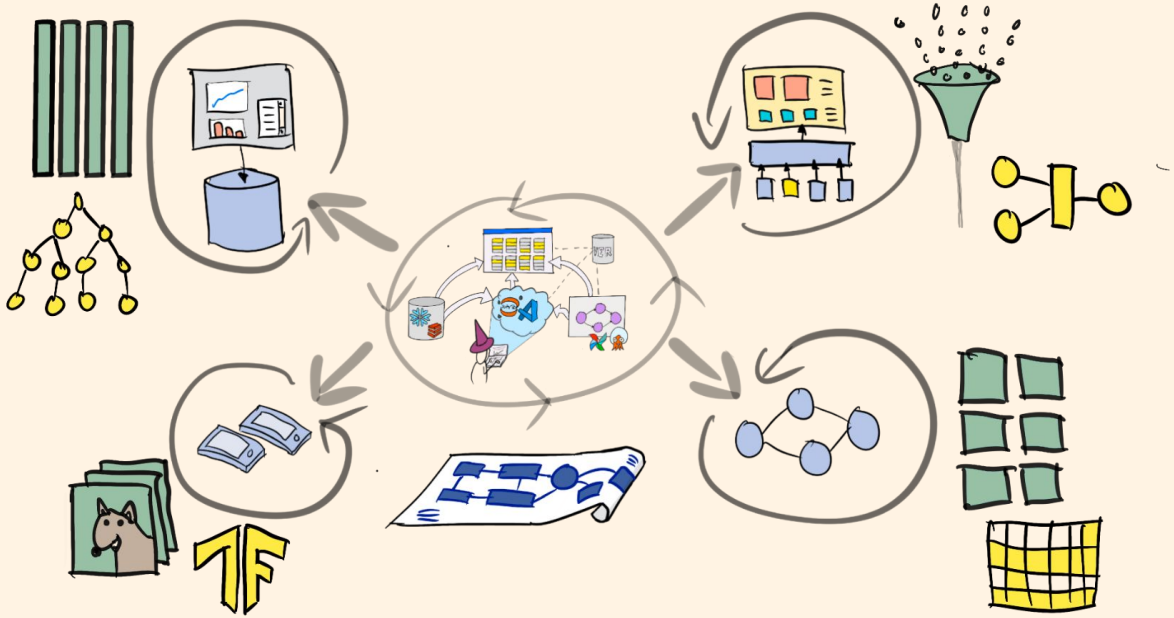
Semi-structured data

- Features
- Operations
- Architecture
- Versioning
- Orchestration
- Compute
- Data

...as well as iterate on various modeling approaches...

Decisions trees

Model Ensembles

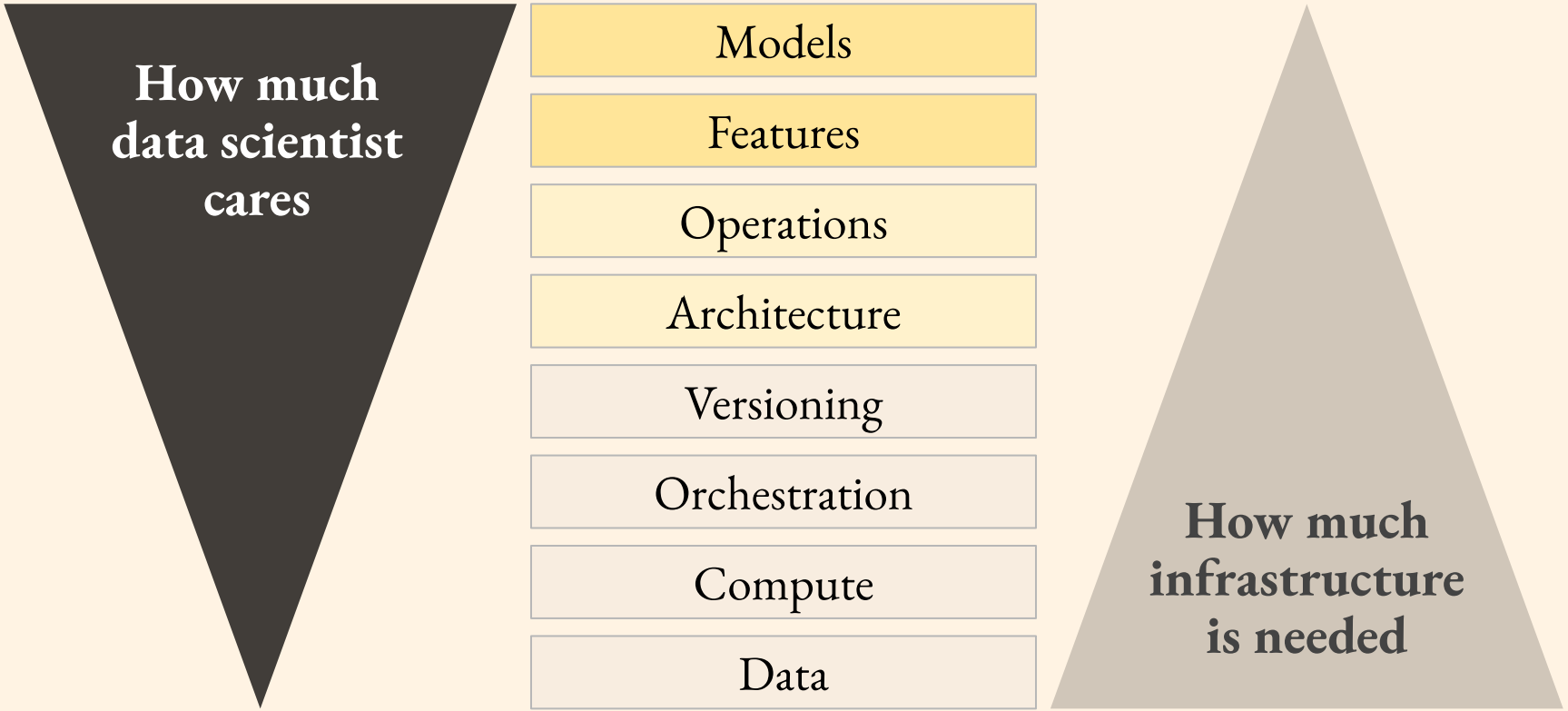


Deep neural networks

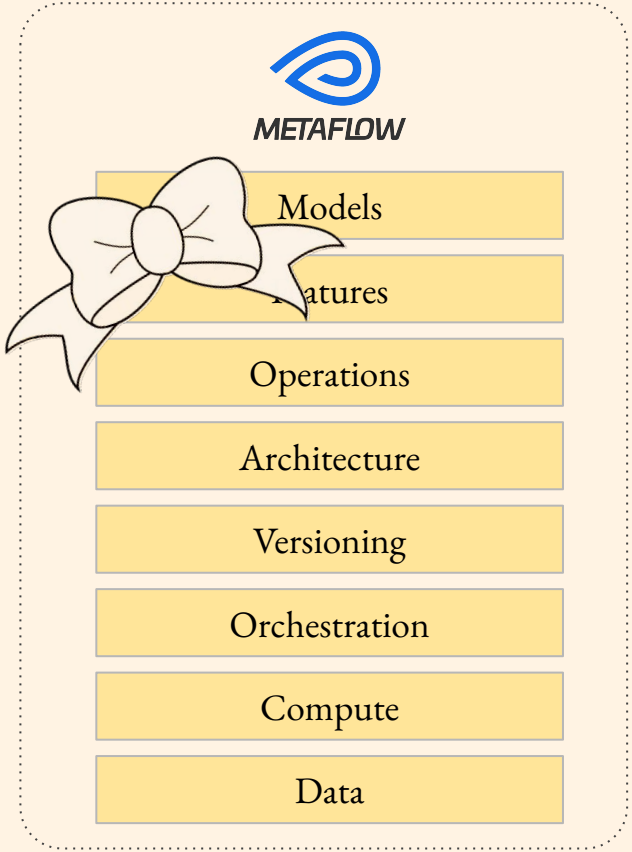
Embeddings

- Models
- Features
- Operations
- Architecture
- Versioning
- Orchestration
- Compute
- Data

because that's what data scientists are mostly supposed to do!



The full stack as a single, coherent, user-friendly package

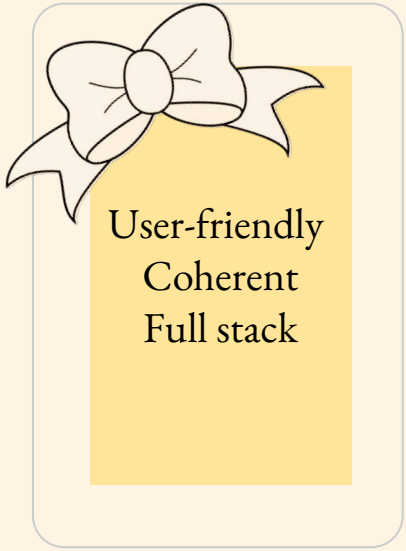


The Evolution of ML Stack

*The stack becomes **simpler, more capable** over time* 💪

CLAM (1998)

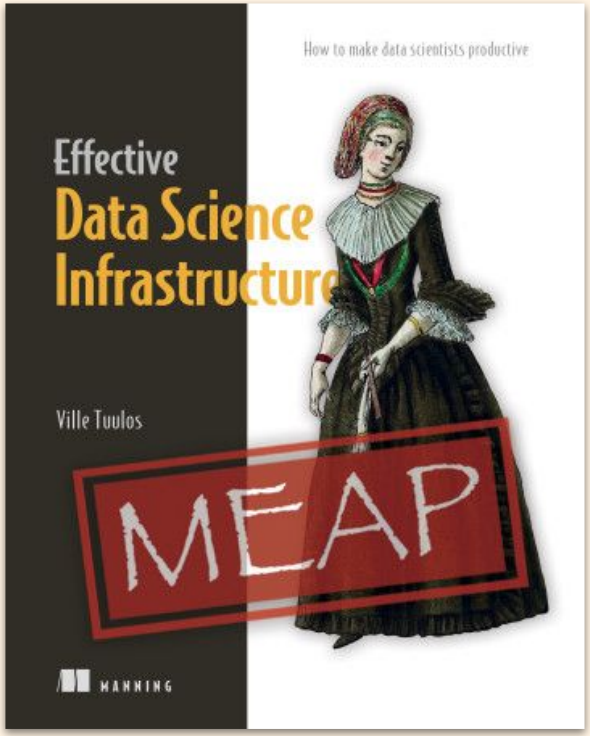
MLOps (2018)



Future!

Shameless plug: New book!

Effective Data Science Infrastructure



Thank you

Curious to learn more about **open-source Metaflow**?
Join 1000+ data scientists and engineers at

http://slack.outerbounds.co