

How Metaflow became a beloved Data Science framework at Netflix

Julie Amundson

Machine Learning Infrastructure Leader

My first day at Netflix!

The 2008 Netflix website features a red header with the Netflix logo and navigation links for 'Calvin Koo', 'Your Account', 'Buy / Redeem Gift', and 'Help'. Below the header is a navigation bar with tabs for 'Browse', 'Movies You'll Like', 'Friends', 'Queue', 'DVD Sale \$5.99+', and 'Watch Now'. A search bar is located on the right. The main content area is titled 'Watch Movies Instantly On Your PC' and includes a sub-header 'Instant Viewing • Full-length Movies and TV Series • Included in Your Membership'. The 'Suggestions For You' section displays four movie thumbnails: 'Shrek', 'America: Freedom to Fascism', 'Eron: The Smartest Guys in the Room', and 'Super Size Me', each with a 'Play' button and a star rating. The 'Your Video Quality' section shows a 'Good' rating and a link to 'How your internet speed affects video quality'. The 'Browse' section lists 'All Watch Now by:' with categories like 'Action & Adventure', 'Comedy', 'Drama', 'Independent', 'Sci-Fi & Fantasy', 'Thrillers', and 'Favorite Genres:'. Other genres listed include 'Anime & Animation', 'Children & Family', 'Classics', 'Documentary', and 'Gay & Lesbian'. The 'Recently Viewed' section shows 'Auschwitz: Inside the Nazi State: Episode 1: "Surprising Beginnings"' with a 'Play' button and a star rating. The 'From Your DVD Queue' section shows '12 Monkeys' and 'Black Rain' with 'Play' buttons and star ratings.

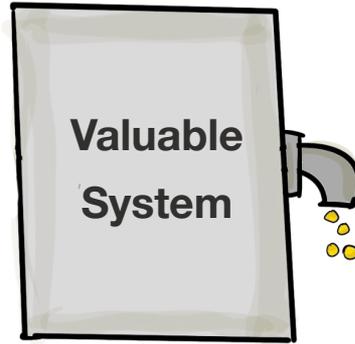
2008

vs.

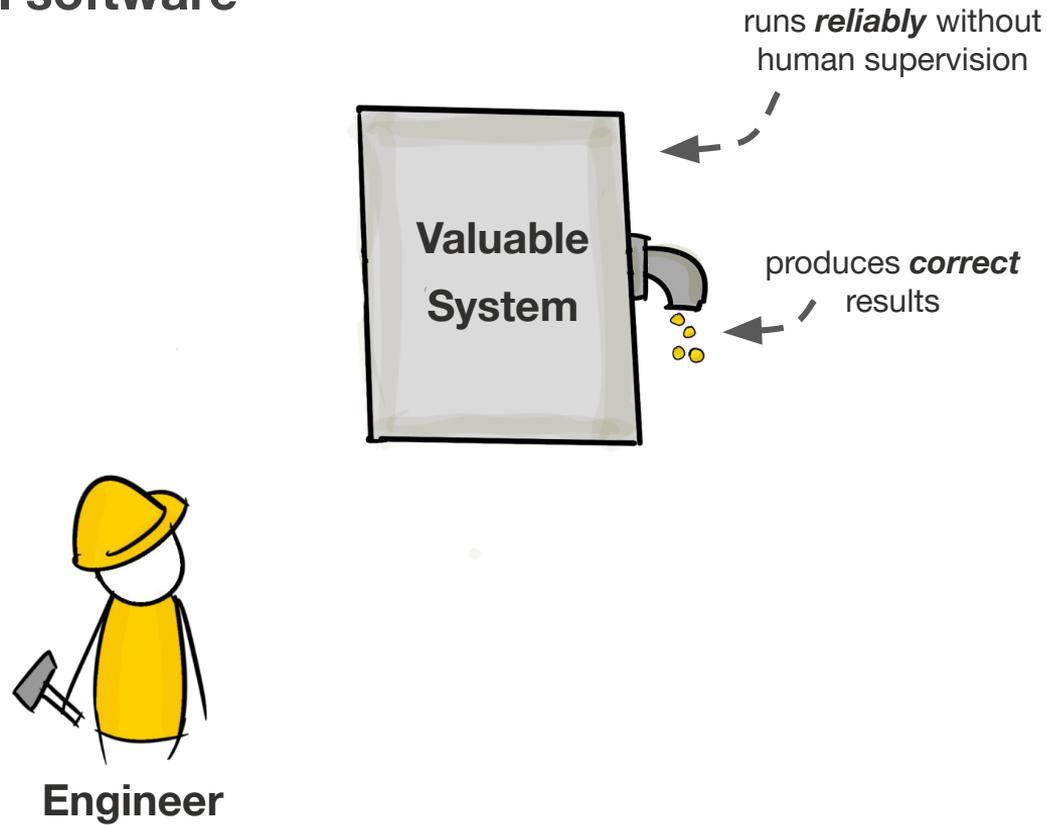
The 2018 Netflix website features a dark header with the Netflix logo and navigation links for 'Home', 'TV Programmes', 'Films', 'Originals', 'Recently Added', and 'My List'. The main content area is titled 'Netflix Originals >' and displays a row of four original series: 'Riverdale', 'Stranger Things', 'Black Mirror', and 'Star Trek: Discovery'. Below this is the 'Trending Now' section, which displays a row of five titles: 'Altered Carbon', 'Friends', 'Peaky Blinders', and 'The End of the F...'. The 'Popular on Netflix' section displays a row of five titles: 'Power', 'The Good Place', 'SpongeBob SquarePants', and 'The Crown'.

2018

2008: Traditional software



2008: Traditional software

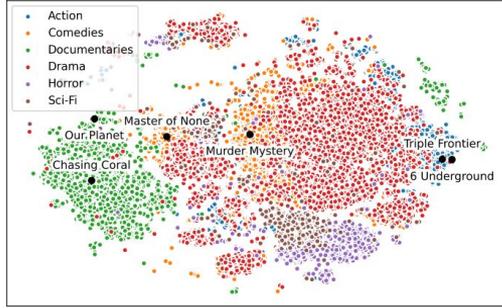


2008: I worked on the services behind the “play” button



Me

2018: Netflix rapidly expands ML investment



PRE-
PRODUCTION



PRODUCTION



POST
PRODUCTION



LOCALIZATION
& QC

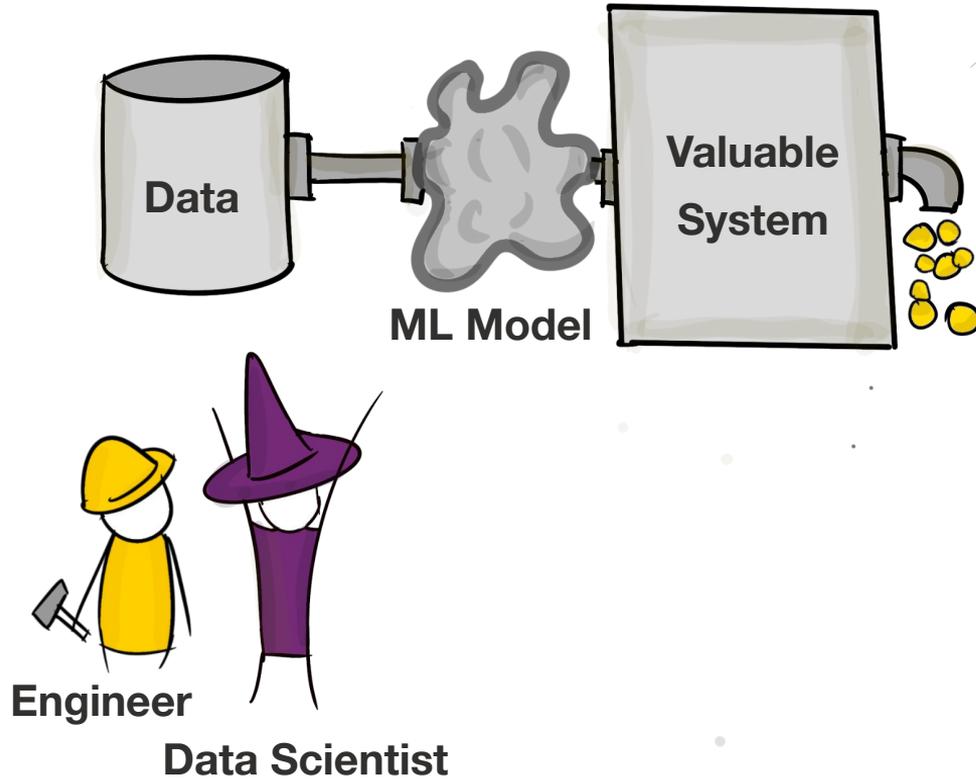


Content
\$12B budget
& growing

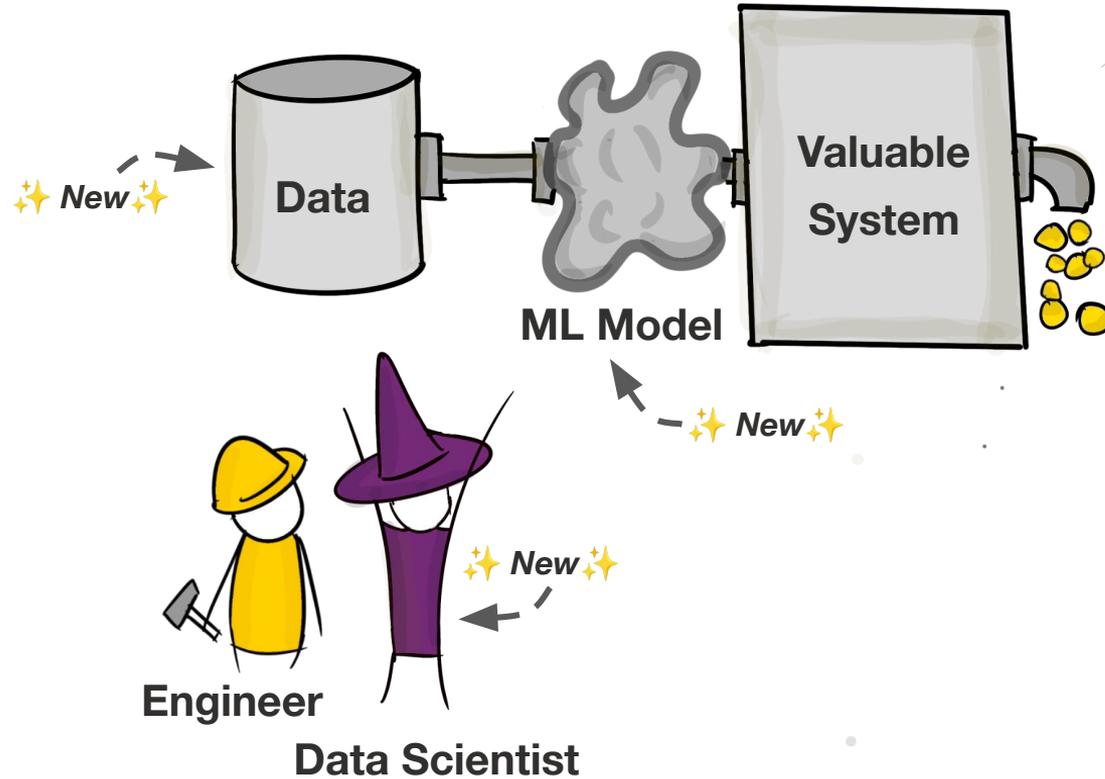
Studio
100s of concurrent
productions

Product
deliver the best
experience for **220M+**
members

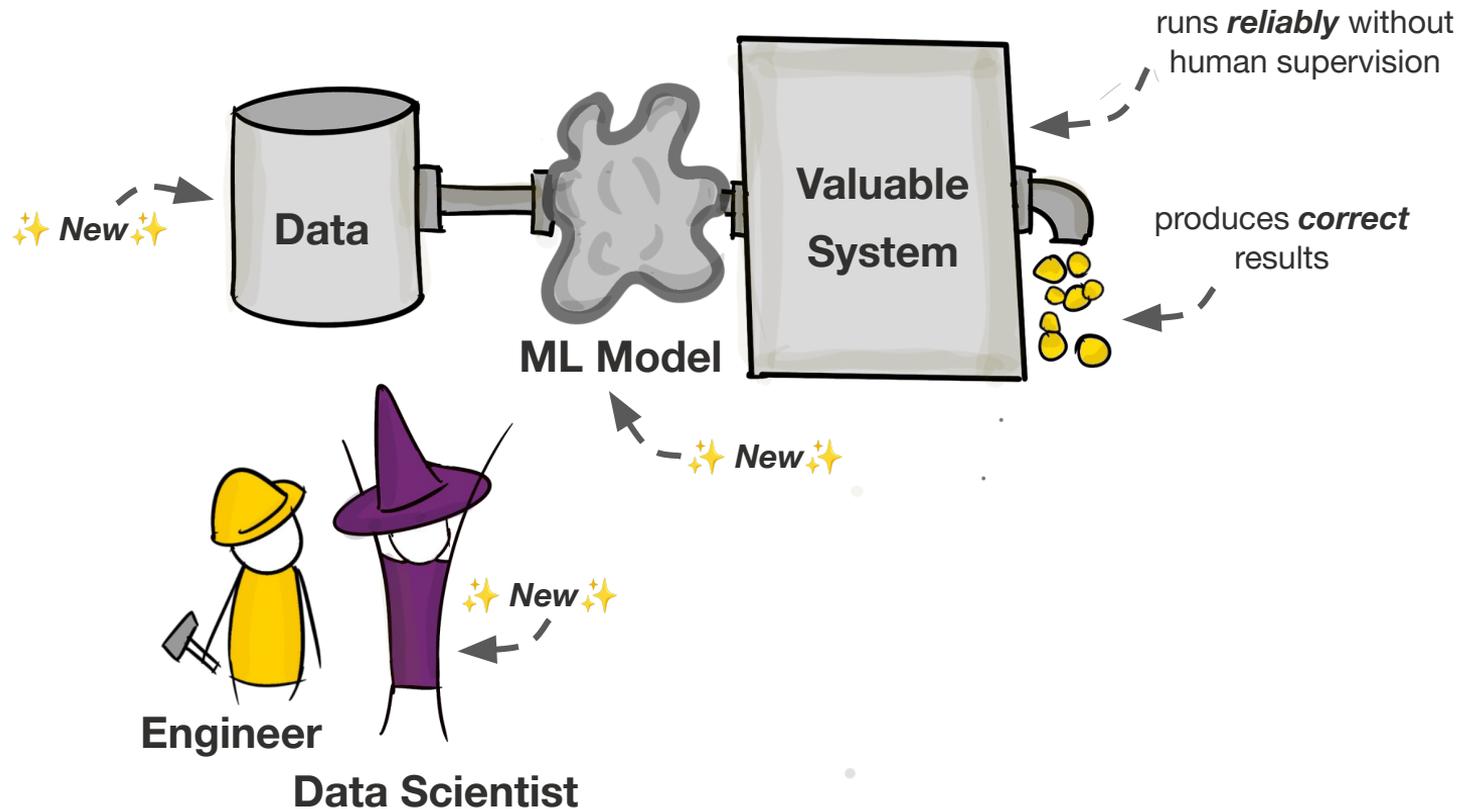
2018: ML-powered software



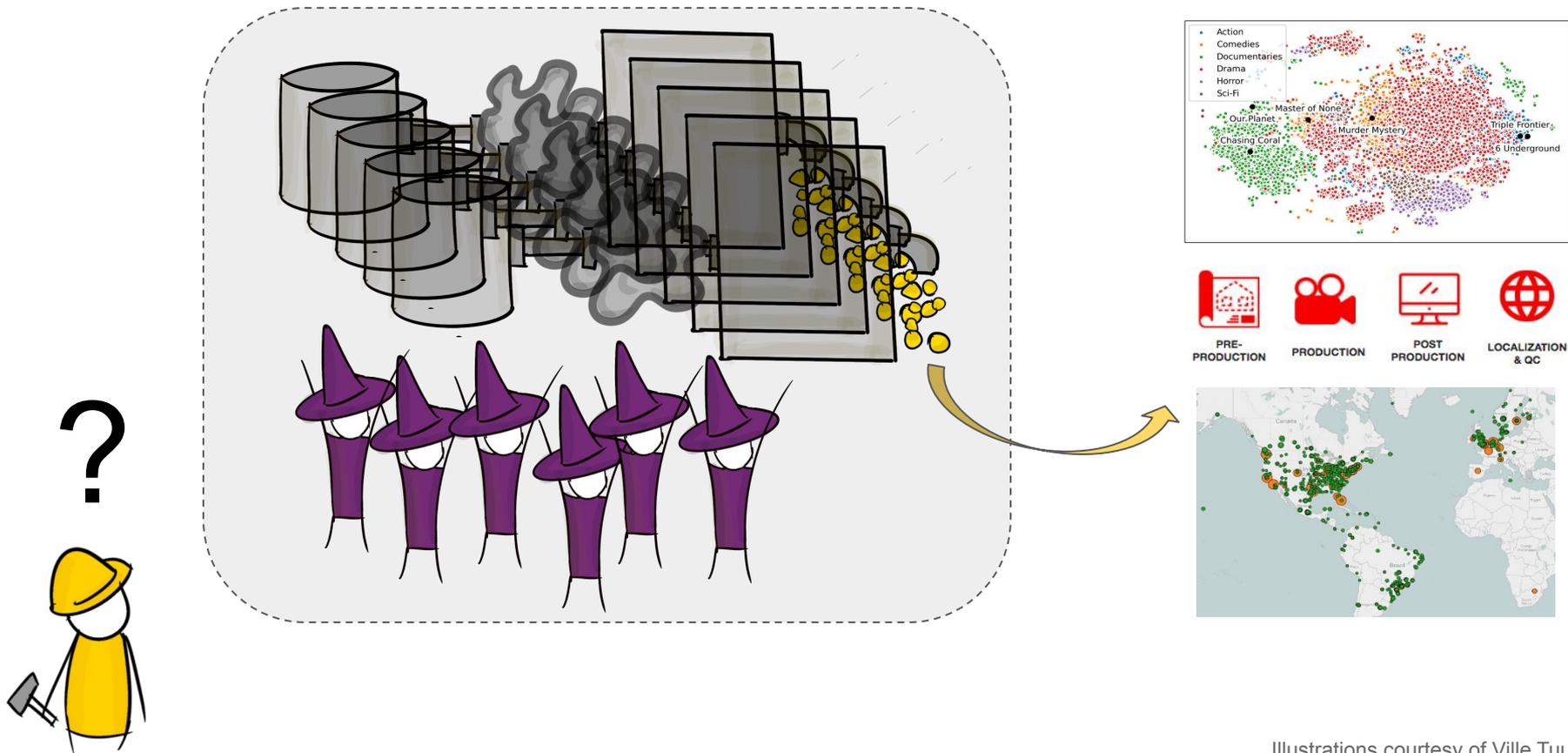
2018: ML-powered software



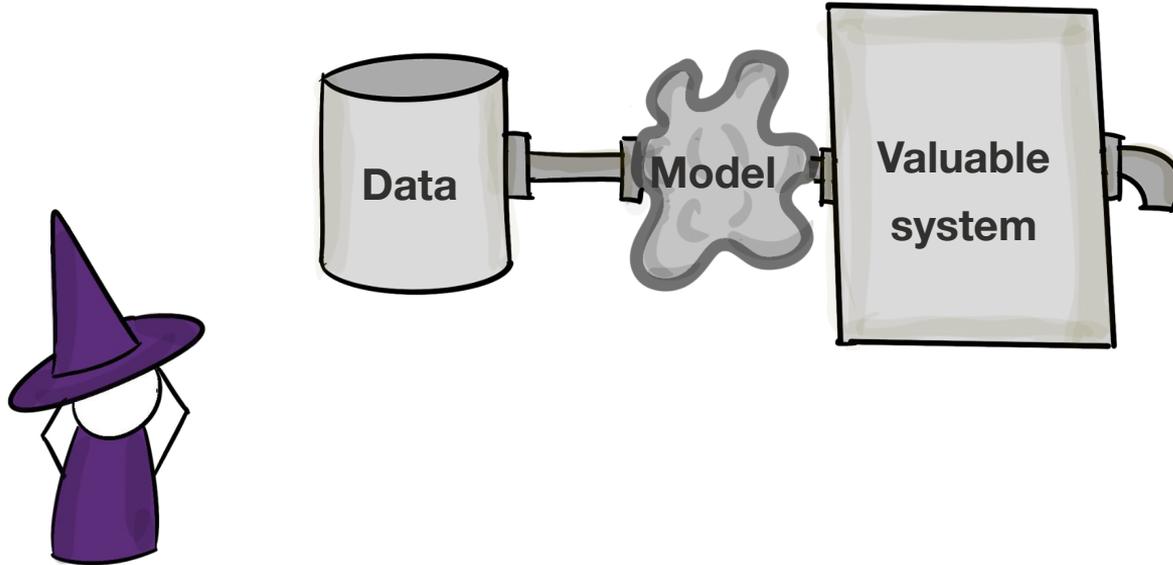
2018: ML-powered software



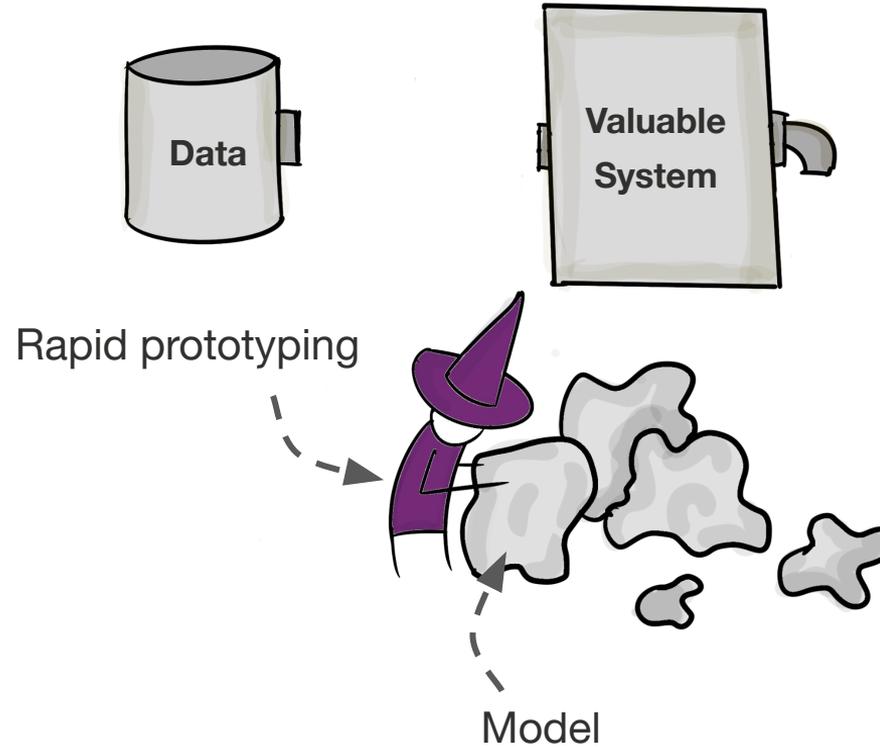
Netflix needed to produce many impactful ML-powered applications



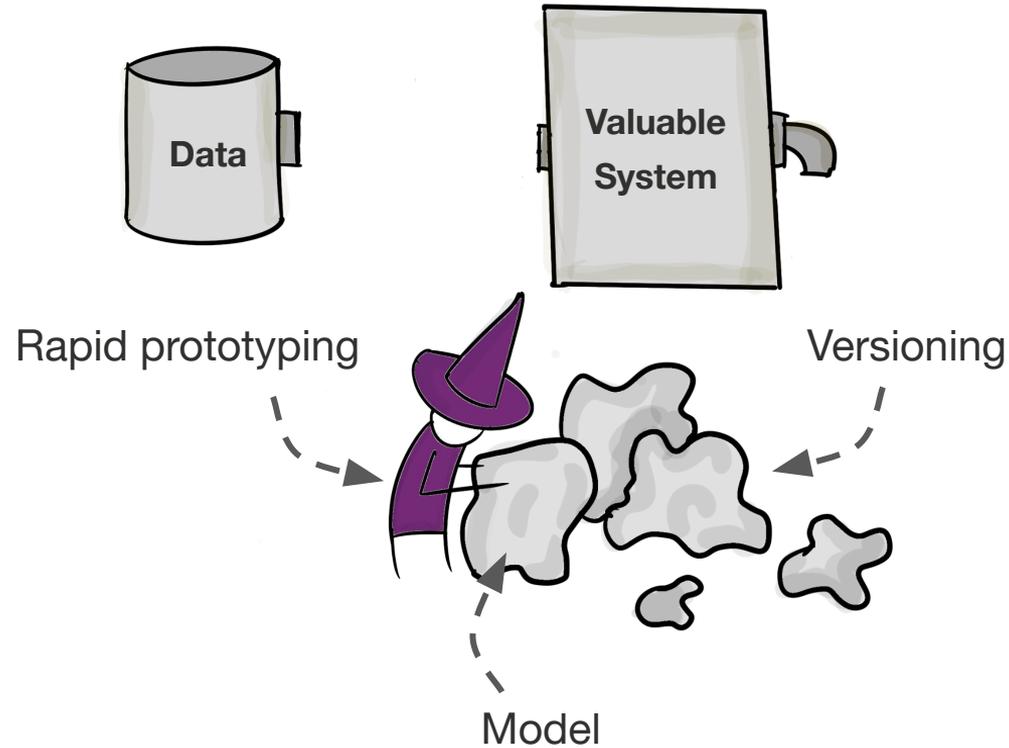
Sonia is building sentiment analysis models for Marketing



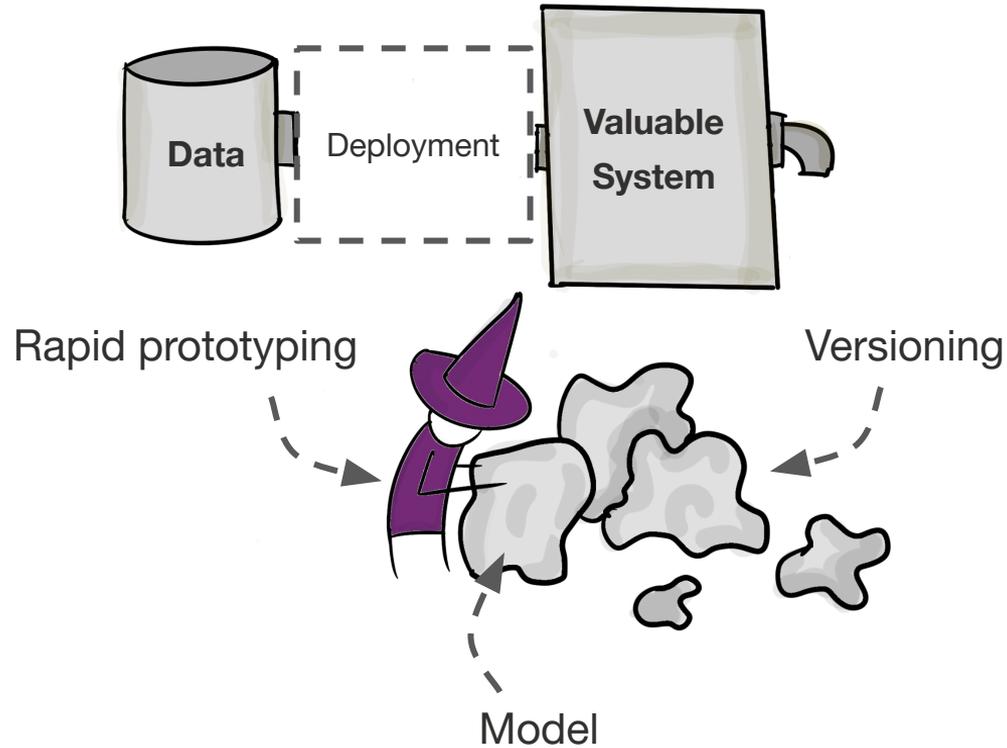
First, she needs to build a prototype



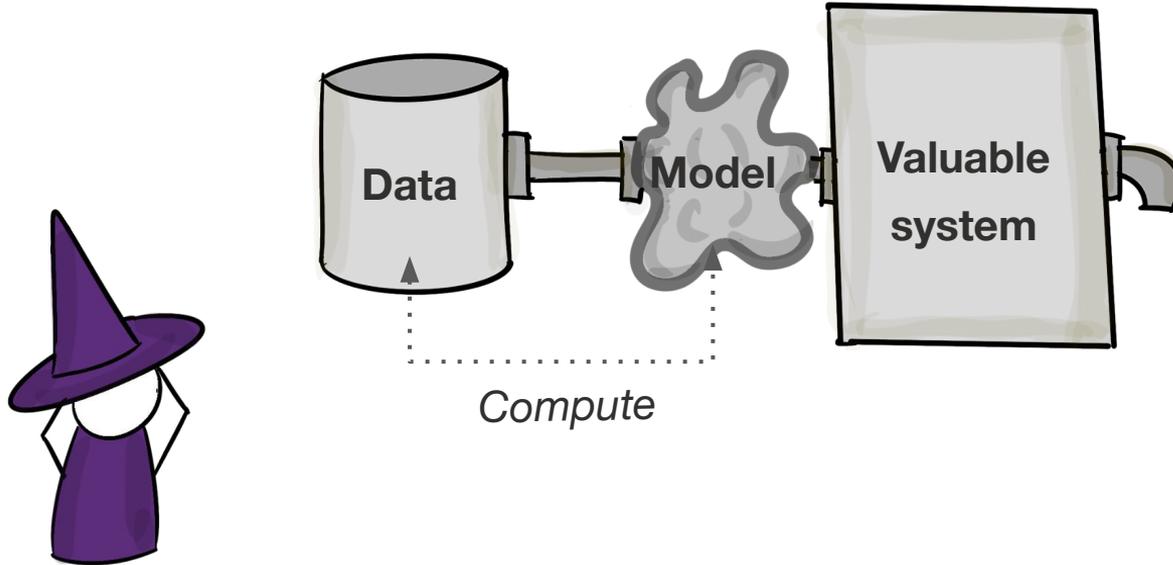
Sonia needs to iterate on many ideas to find a prototype that works



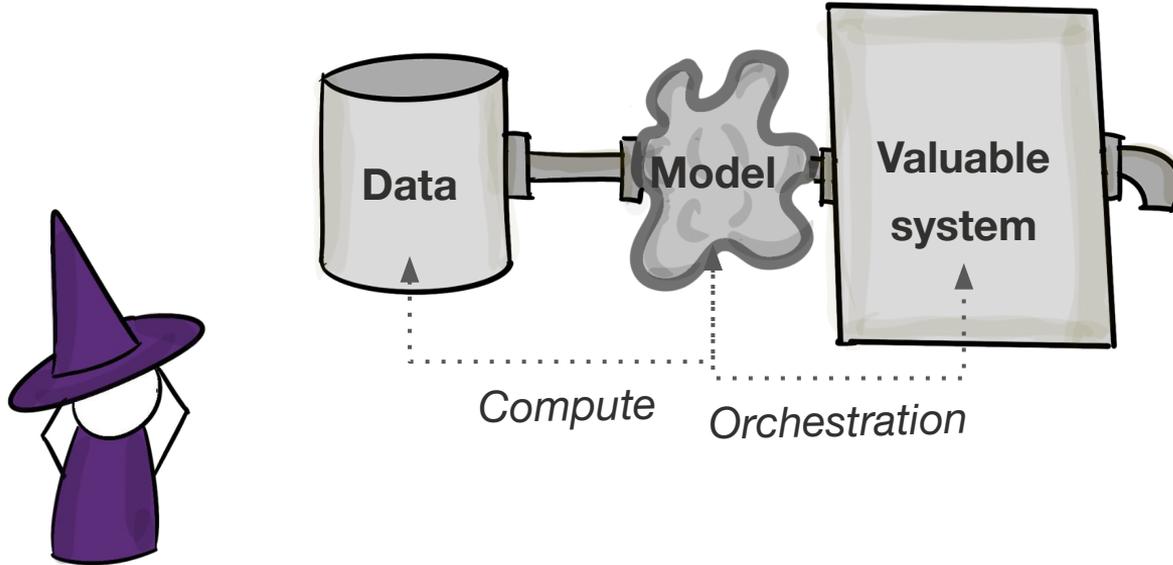
She needs a way to reliably deploy to production



Model training and scoring require compute

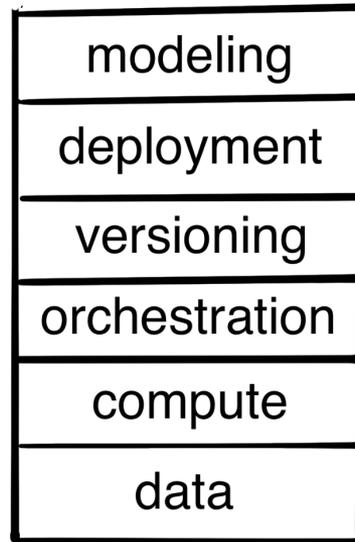


Sonia's modeling pipeline needs orchestration!



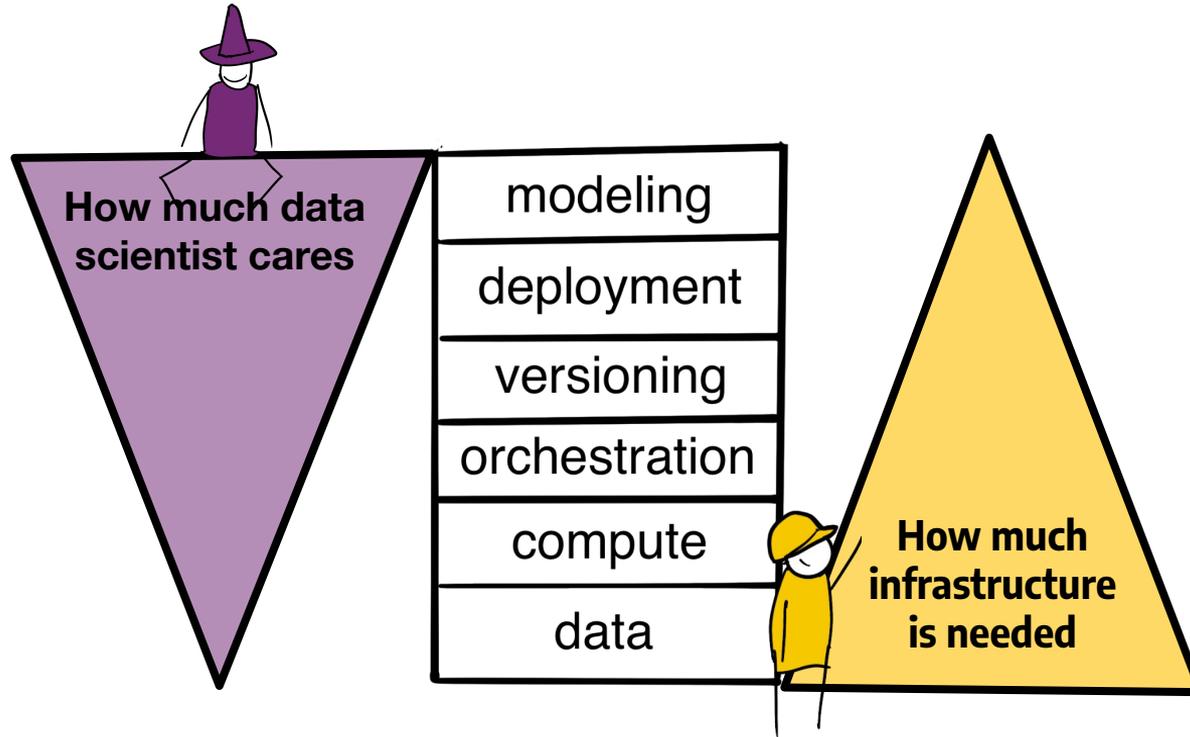
Putting the pieces together

The ML Infrastructure Stack



Putting the pieces together

The ML Infrastructure Stack

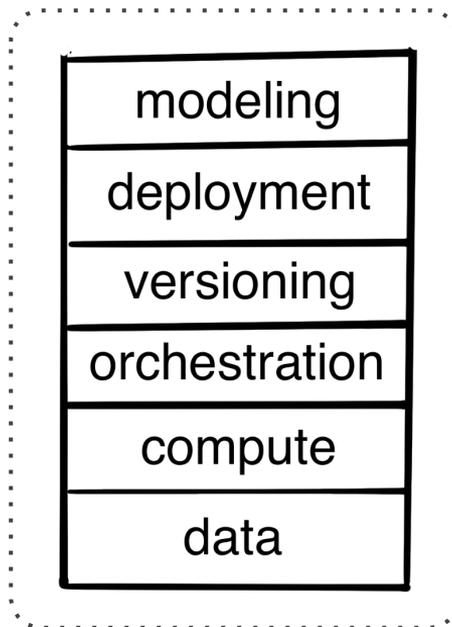


Metaflow

A full-stack, human-friendly framework for data science



METAFLOW



```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule
```

```
@conda_base(libraries={'scikit-learn': '1.1.2'})
```

```
@schedule(daily=True)
```

```
class HelloFlow(FlowSpec):
```

```
    @step
```

```
    def start(self):
```

```
        self.x = 1
```

```
        self.next(self.end)
```

```
    @kubernetes(memory=64000)
```

```
    @step
```

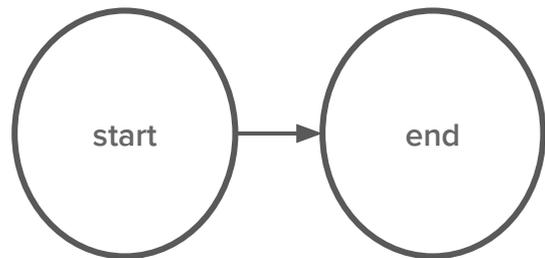
```
    def end(self):
```

```
        self.x += 1
```

```
        print("Hello world! The value of x is", self.x)
```

```
if __name__ == '__main__':
```

```
    HelloFlow()
```



```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule
```

```
@conda_base(libraries={'scikit-learn': '1.1.2'})
```

```
@schedule(daily=True)
```

```
class HelloFlow(FlowSpec):
```

```
    @step
```

```
    def start(self):
```

```
        self.x = 1
```

```
        self.next(self.end)
```

```
    @kubernetes(memory=64000)
```

```
    @step
```

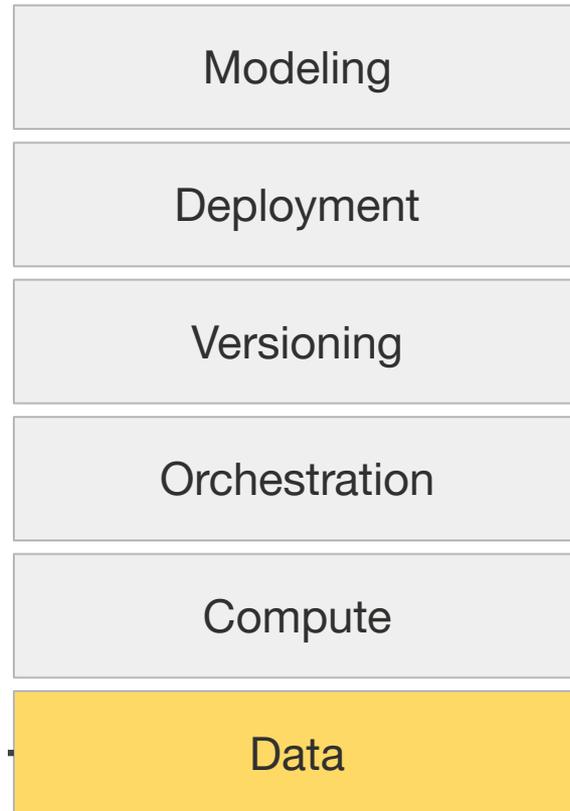
```
    def end(self):
```

```
        self.x += 1
```

```
        print("Hello world! The value of x is", self.x)
```

```
if __name__ == '__main__':
```

```
    HelloFlow()
```

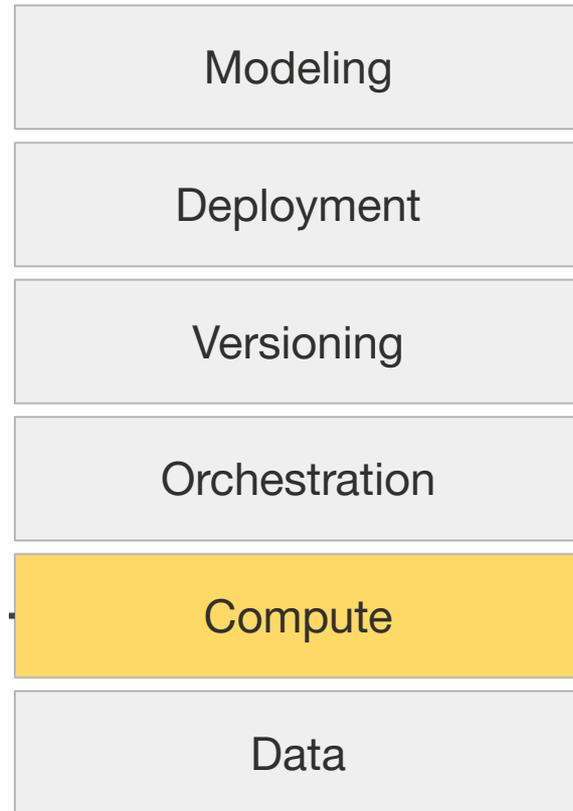


```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule

@conda_base(libraries={'scikit-learn': '1.1.2'})
@schedule(daily=True)
class HelloFlow(FlowSpec):
    @step
    def start(self):
        self.x = 1
        self.next(self.end)

    @kubernetes(memory=64000)
    @step
    def end(self):
        self.x += 1
        print("Hello world! The value of x is", self.x)

if __name__ == '__main__':
    HelloFlow()
```



```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule
```

```
@conda_base(libraries={'scikit-learn': '1.1.2'})
```

```
@schedule(daily=True)
```

```
class HelloFlow(FlowSpec):
```

```
    @step
```

```
    def start(self):
```

```
        self.x = 1
```

```
        self.next(self.end)
```

```
    @kubernetes(memory=64000)
```

```
    @step
```

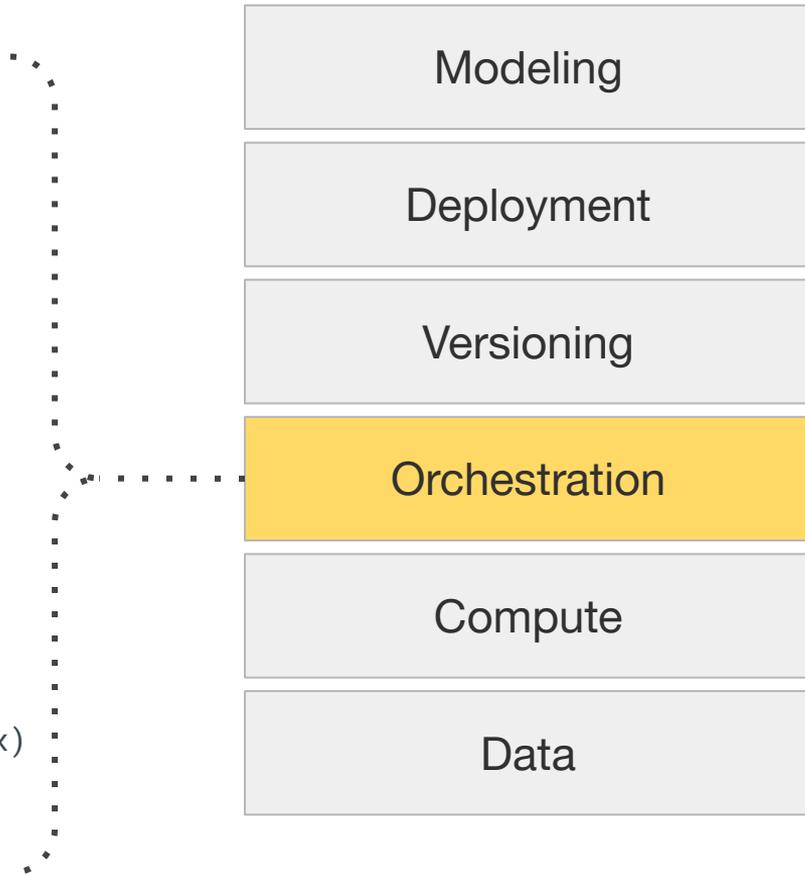
```
    def end(self):
```

```
        self.x += 1
```

```
        print("Hello world! The value of x is", self.x)
```

```
if __name__ == '__main__':
```

```
    HelloFlow()
```

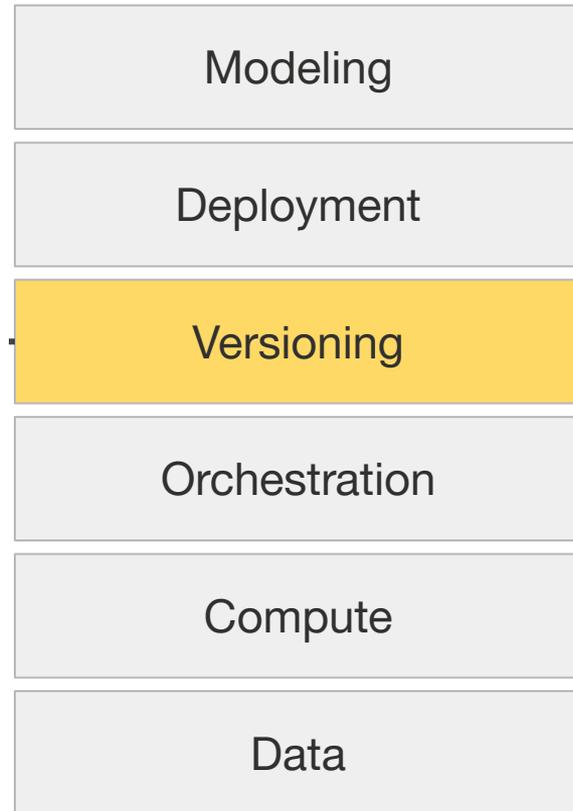


```
from metaflow import FlowSpec, step, conda_base, \
                    kubernetes, schedule

@conda_base(libraries={'scikit-learn': '1.1.2'})
@schedule(daily=True)
class HelloFlow(FlowSpec):
    @step
    def start(self):
        self.x = 1
        self.next(self.end)

    @kubernetes(memory=64000)
    @step
    def end(self):
        self.x += 1
        print("Hello world! The value of x is", self.x)

if __name__ == '__main__':
    HelloFlow()
```



```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule

@conda_base(libraries={'scikit-learn': '1.1.2'})
@schedule(daily=True)
class HelloFlow(FlowSpec):
    @step
    def start(self):
        self.x = 1
        self.next(self.end)

    @kubernetes(memory=64000)
    @step
    def end(self):
        self.x += 1
        print("Hello world! The value of x is", self.x)

if __name__ == '__main__':
    HelloFlow()
```

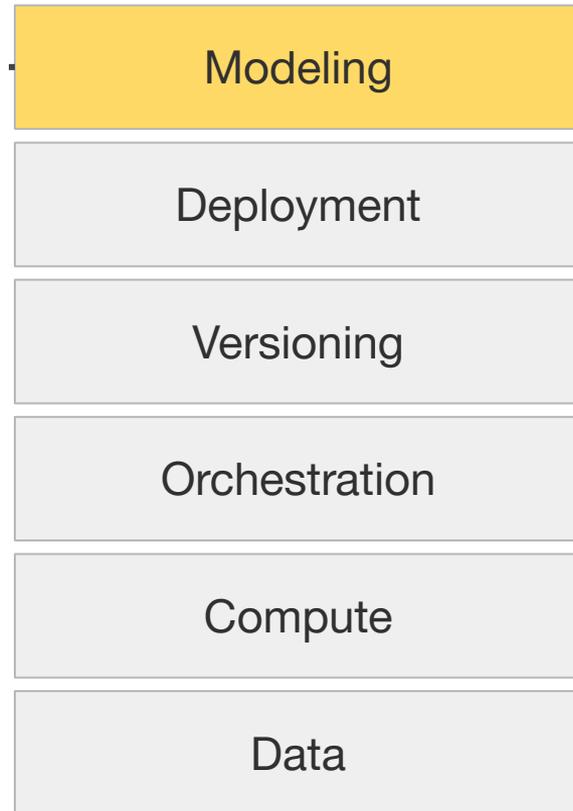


```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule

@conda_base(libraries={'scikit-learn': '1.1.2'})
@schedule(daily=True)
class HelloFlow(FlowSpec):
    @step
    def start(self):
        self.x = 1
        self.next(self.end)

    @kubernetes(memory=64000)
    @step
    def end(self):
        self.x += 1
        print("Hello world! The value of x is", self.x)

if __name__ == '__main__':
    HelloFlow()
```



```
from metaflow import FlowSpec, step, conda_base,\
                    kubernetes, schedule

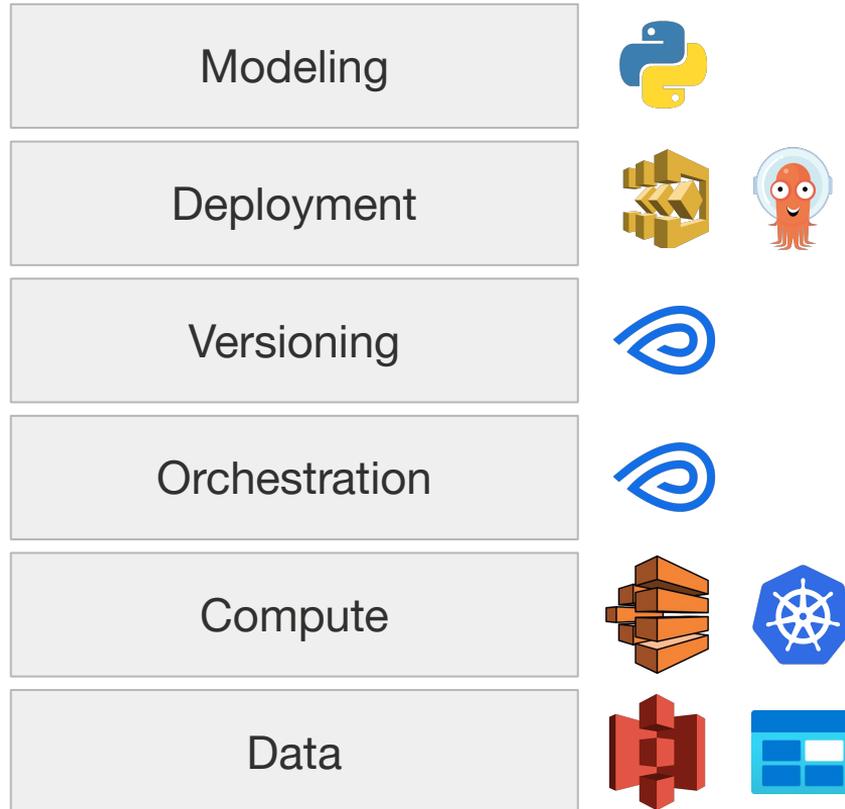
@conda_base(libraries={'scikit-learn': '1.1.2'})
@schedule(daily=True)
class HelloFlow(FlowSpec):
    @step
    def start(self):
        self.x = 1
        self.next(self.end)

    @kubernetes(memory=64000)
    @step
    def end(self):
        self.x += 1
        print("Hello world! The value of x is", self.x)

if __name__ == '__main__':
    HelloFlow()
```

**The code may look nice
but it doesn't produce value
by itself**

To produce real value, the code needs to integrate seamlessly *with the surrounding infrastructure*



Metaflow impact at Netflix



Velocity



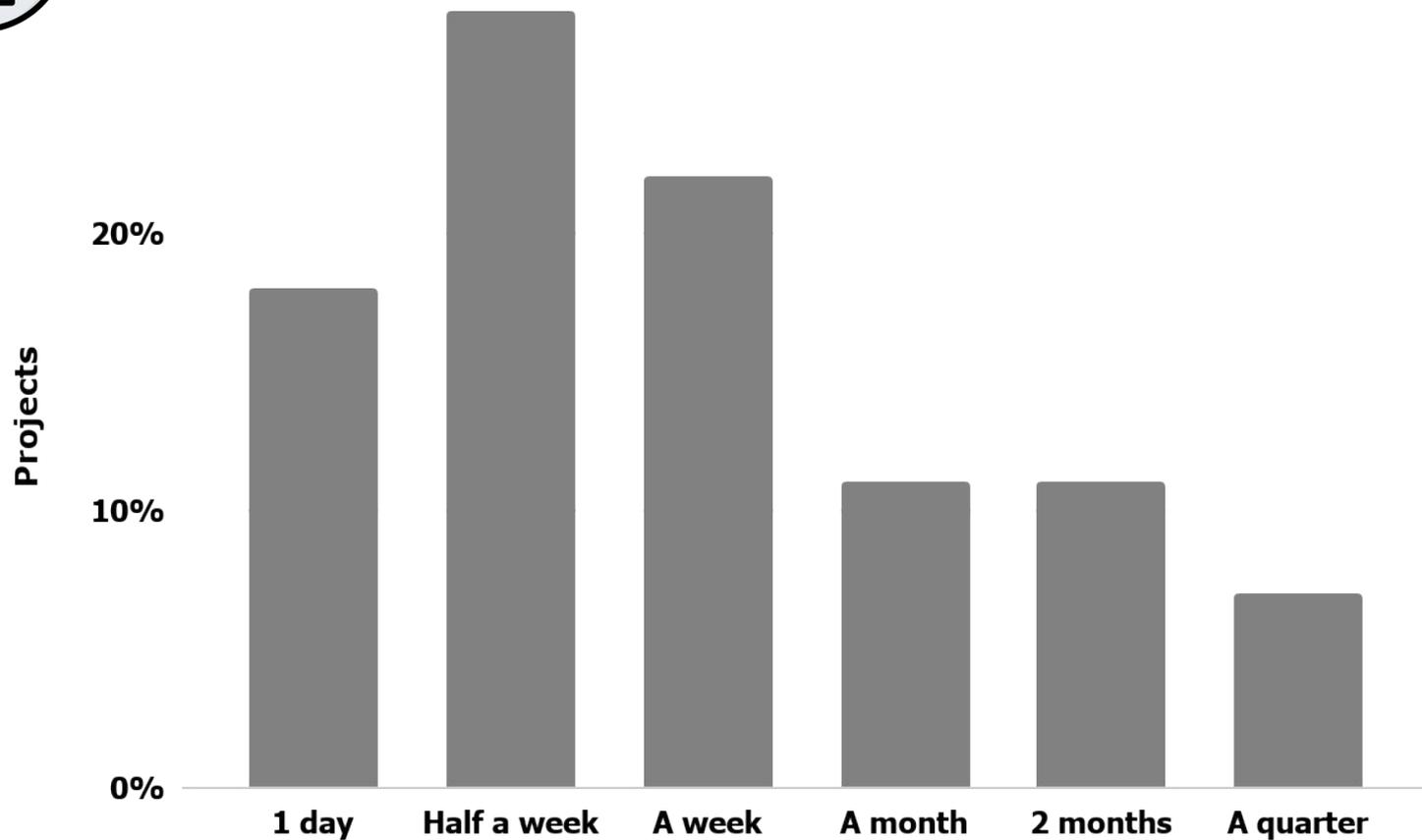
Volume



Variety

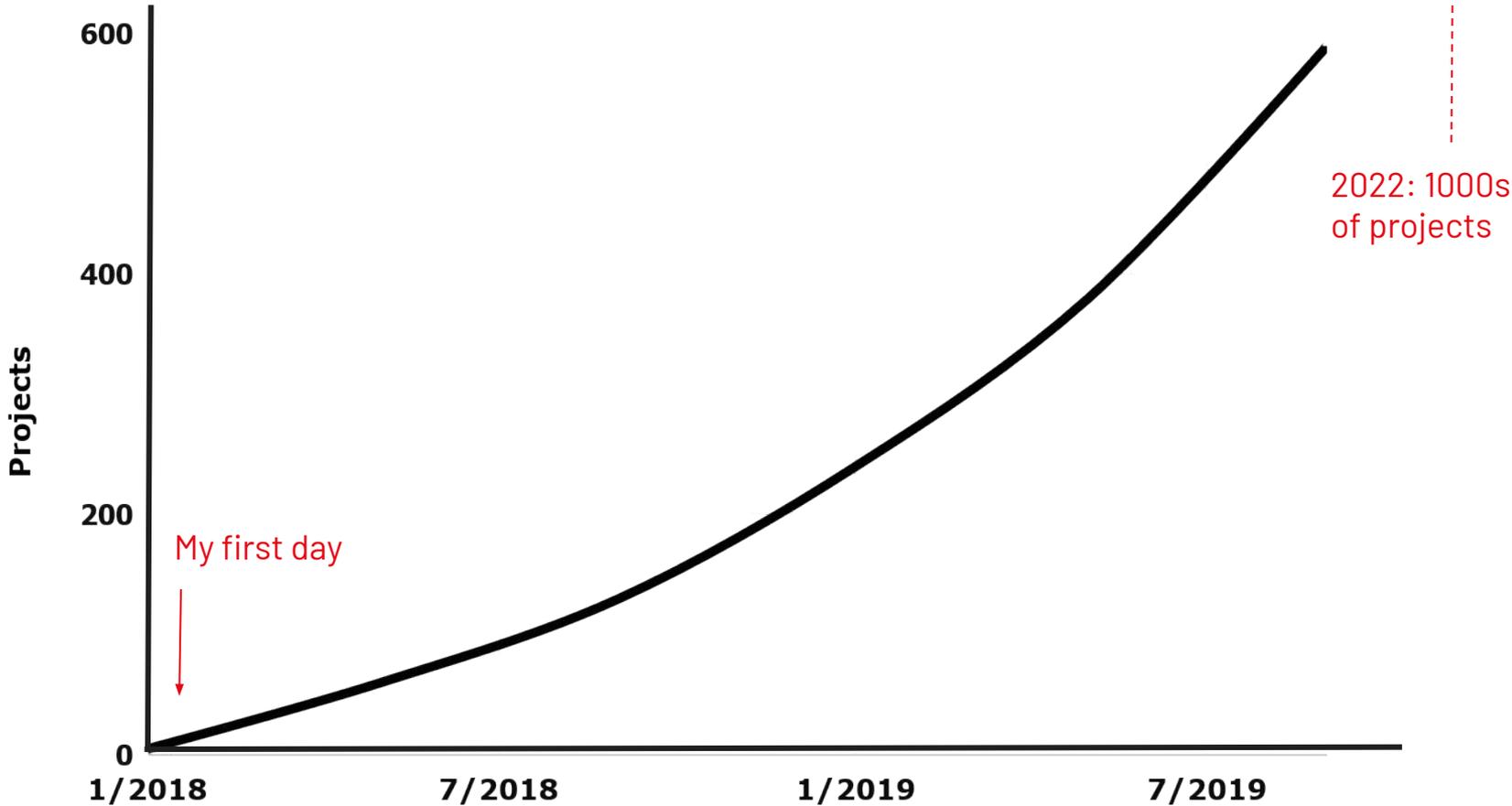


Velocity: Time to production





Volume: Metaflow Adoption





Beyond Variety: Netflix Open Source

 **1.8k+**

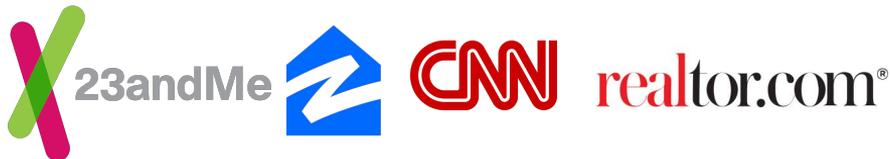
community
members

NETFLIX
OSS **7th**

most popular
project



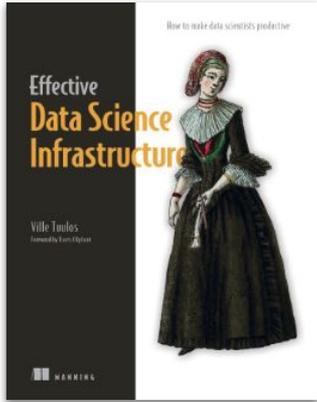
multiple cloud
integrations



adopted by hundreds
of companies

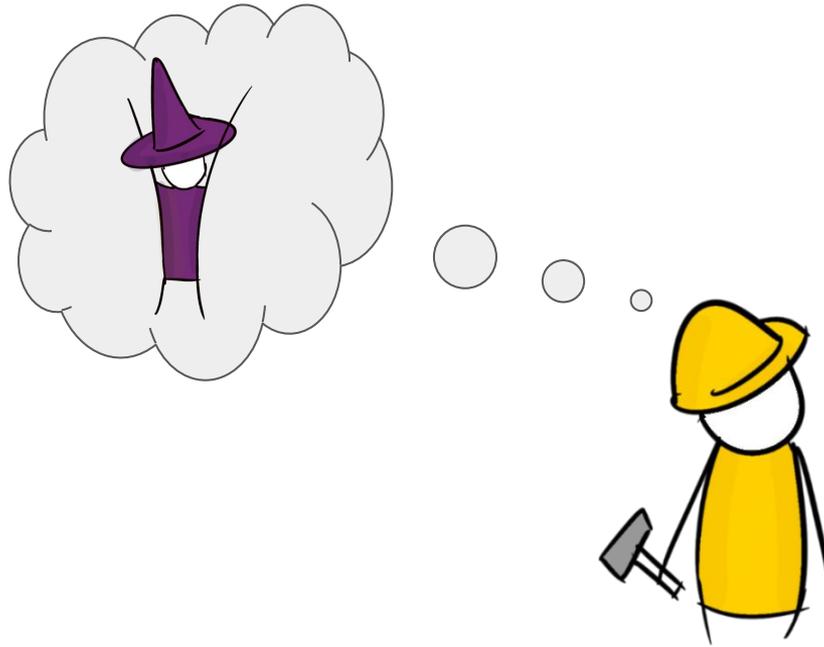
Want to learn more about Metaflow?

docs.metaflow.org
outerbounds.com



Effective Data Science Infrastructure
New book, by **Ville Tuulos**

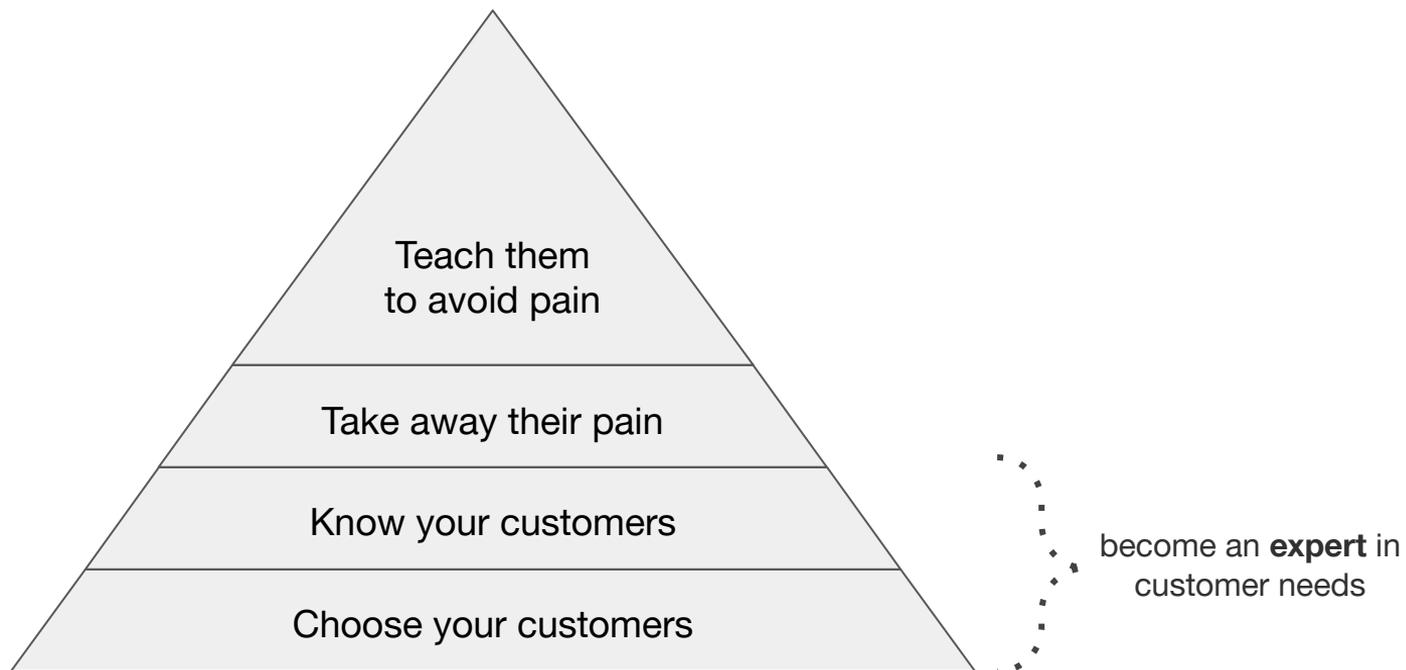
Metaflow's success began with customer obsession



Customer obsession in action



Customer obsession in action

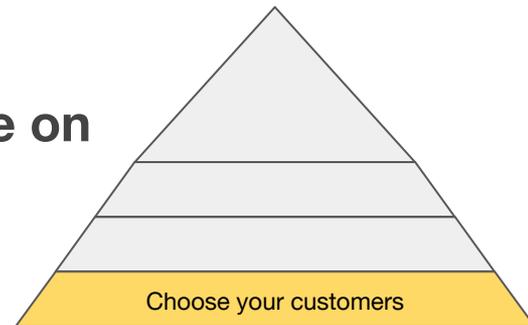


Customer obsession in action



equip your customers
with **superpowers**

Choosing Data Scientists meant we could concentrate on their needs



Customers

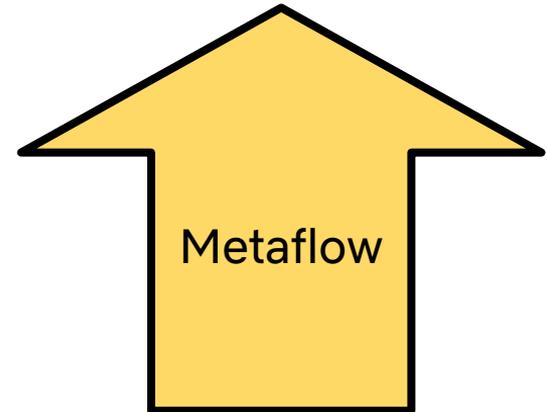
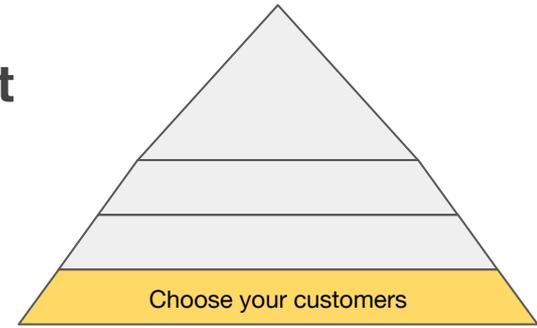
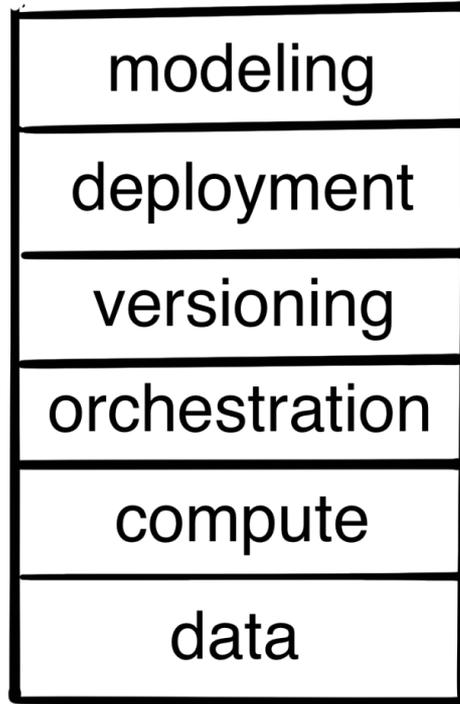
- Data Scientists
- ML Engineers

Non-Customers

- Algorithm Engineers
- Data Engineers
- Analytics Engineers
- Software Engineers

Choosing Data Scientists helped to clarify our product strategy

conventional wisdom



Choosing Data Scientists enabled us to focus

Non-features

- Generic feature store
- Training framework
- Generic Model registry
- ...etc...

modeling

deployment

versioning

orchestration

compute

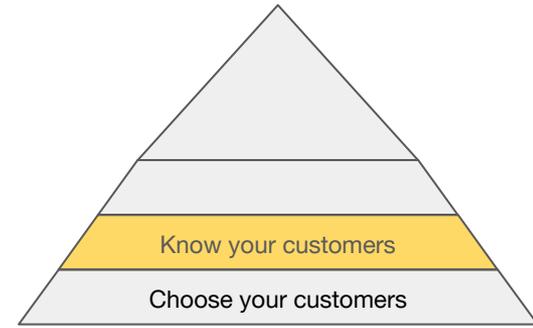
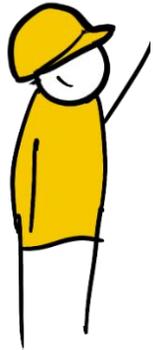
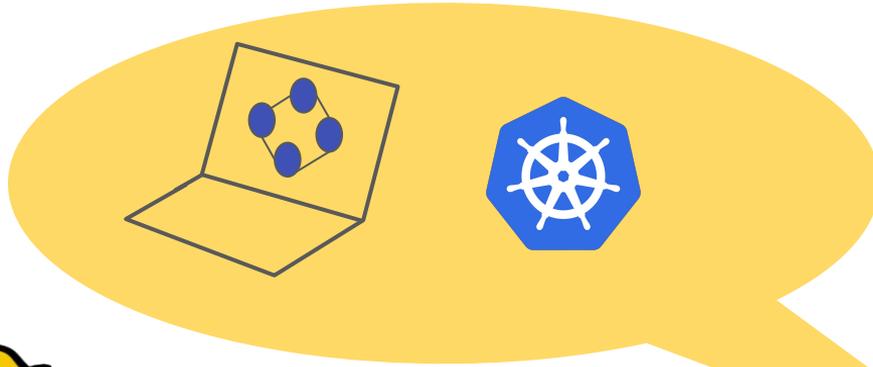
data

Choose your customers

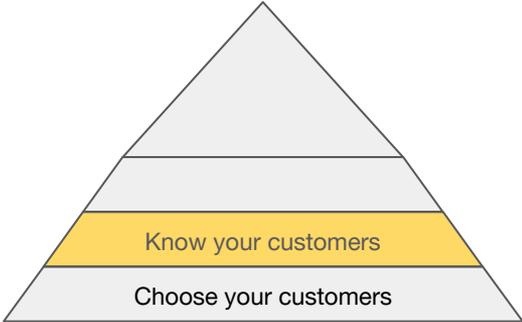
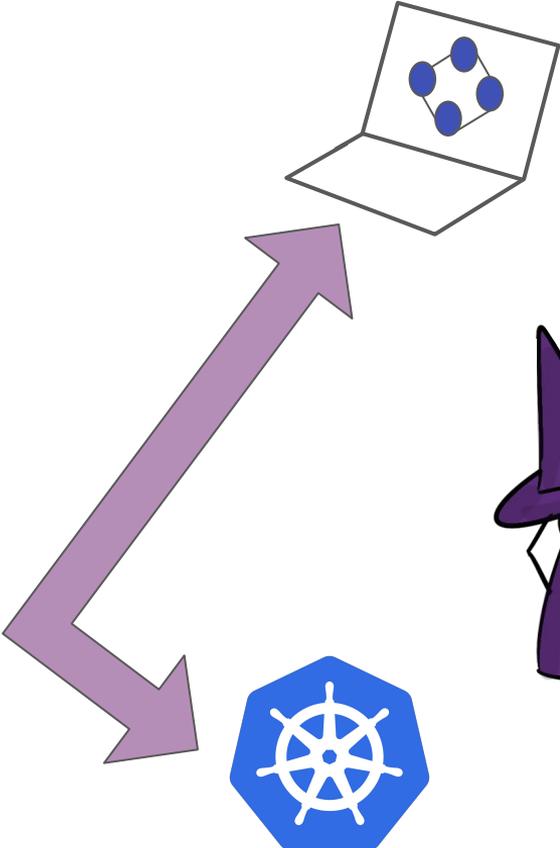
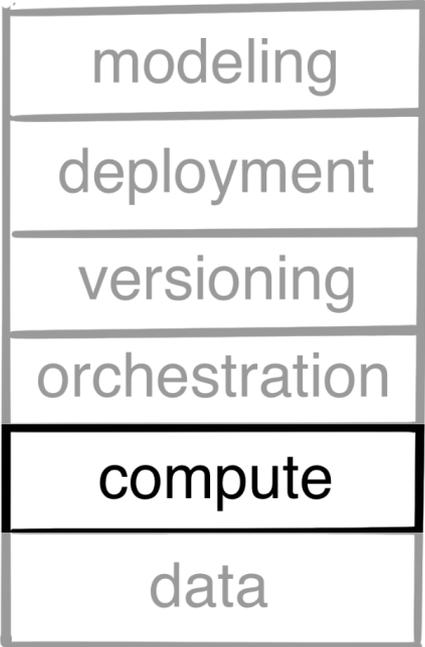
Features

- Data-parallel training
- Job scheduling
- Reproducibility
- ...etc...

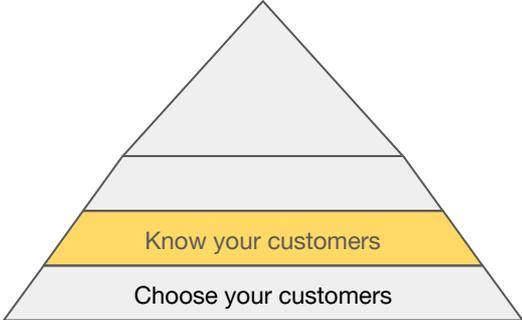
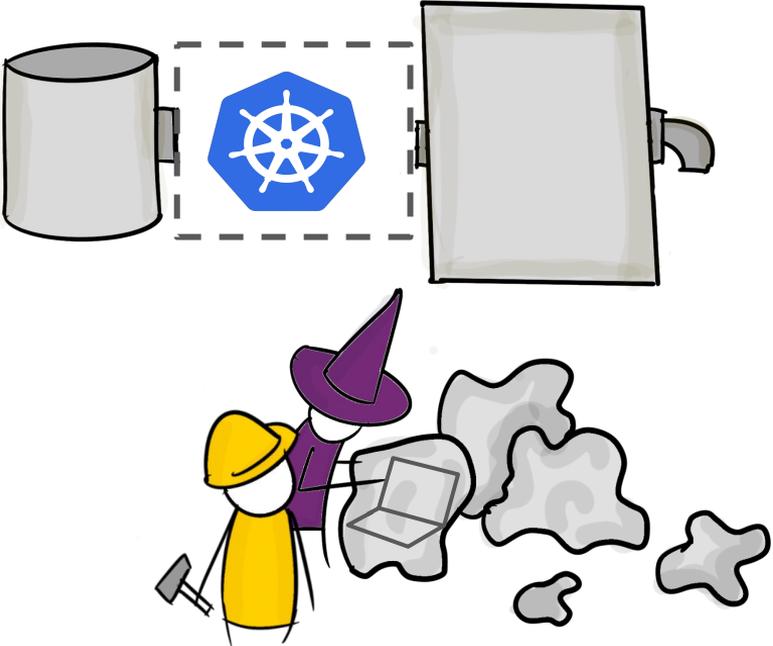
We spoke to many Data Scientists to learn about their daily lives, *in detail*



After many conversations, patterns emerged



Metaflow engineers paired with Data Scientists to experience their pains end-to-end

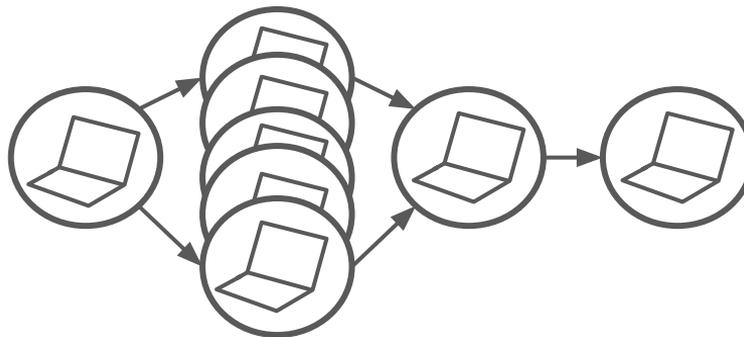


```
@step
def start(self):
    self.n_trees = [16, 32, 64]
    self.next(
        self.train, foreach='n_trees'
    )
```

```
@step
def train(self):
    n_trees = int(self.input)
    model, rmse = train_model(n_trees)
    self.rmse = rmse
    self.model = model
    self.next(self.join)
```

```
@step
def join(self, inputs):
    self.best_rmse = min(
        i.rmse for i in inputs
    )
    self.next(self.end)
```

The resulting features met Data Scientists where they were



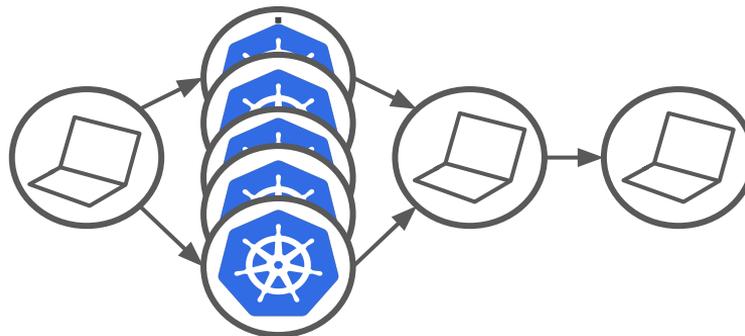
```
@step
def start(self):
    self.n_trees = [16, 32, 64]
    self.next(
        self.train, foreach='n_trees'
    )
```

Data Scientists could ask for compute resources any time they needed!



```
@step
@kubernetes(memory=64000)
def train(self):
    n_trees = int(self.input)
    model, rmse = train_model(n_trees)
    self.rmse = rmse
    self.model = model
    self.next(self.join)
```

```
@step
def join(self, inputs):
    self.best_rmse = min(
        i.rmse for i in inputs
    )
    self.next(self.end)
```



```
@step
def start(self):
    self.n_trees = [16, 32, 64]
    self.next(
        self.train, foreach='n_trees'
    )
```

...Without introducing new pain

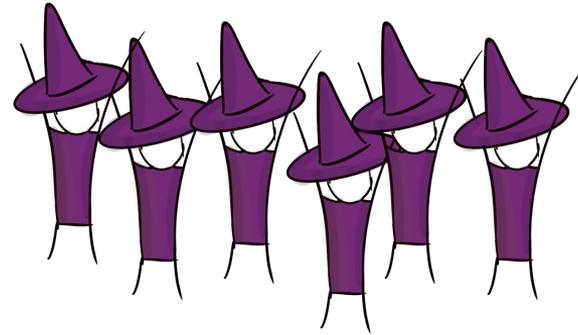
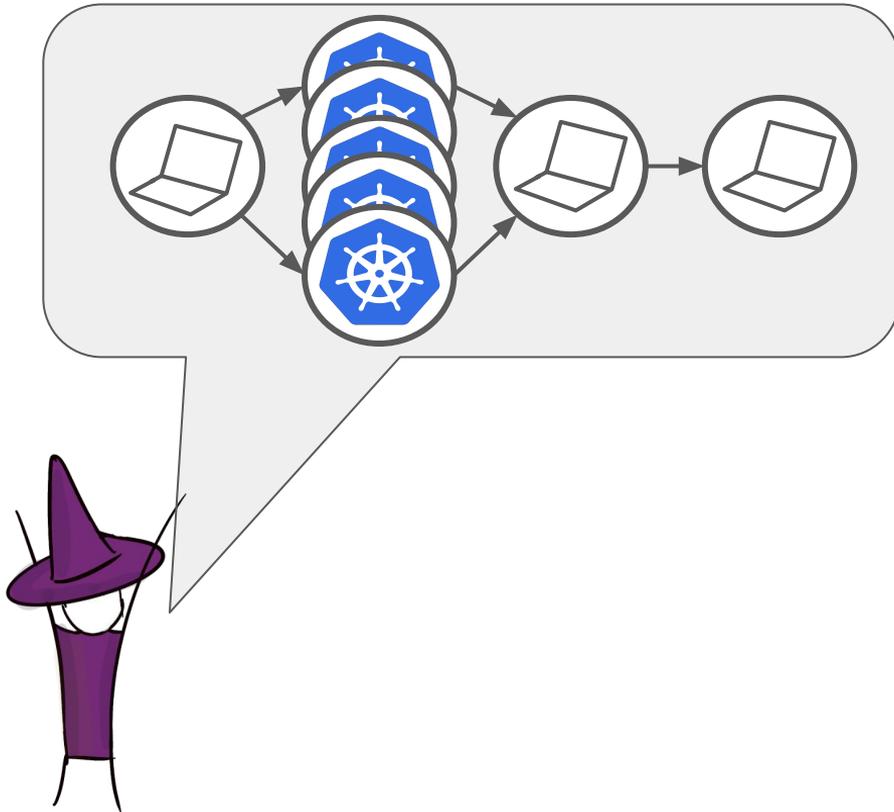
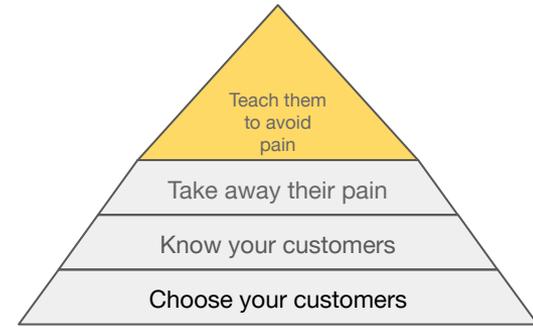
```
@step
@kubernetes(memory=64000)
def train(self):
    n_trees = int(self.input)
    model, rmse = train_model(n_trees)
    self.rmse = rmse
    self.model = model
    self.next(self.join)
```

```
@step
def join(self, inputs):
    self.best_rmse = min(
        i.rmse for i in inputs
    )
    self.next(self.end)
```

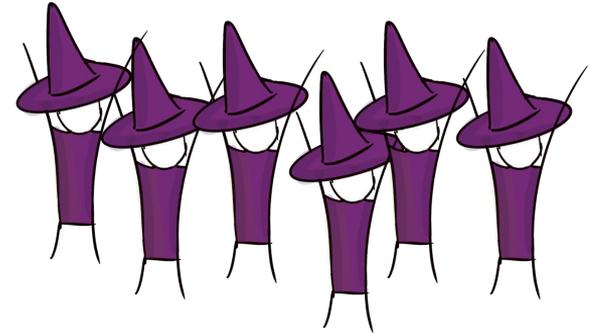
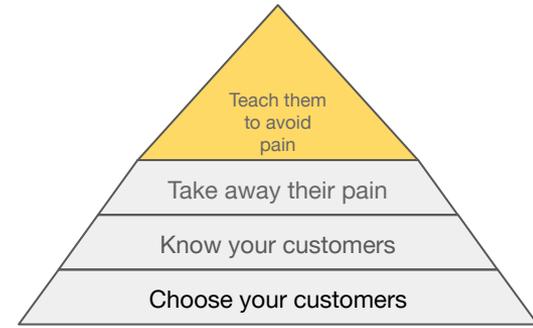
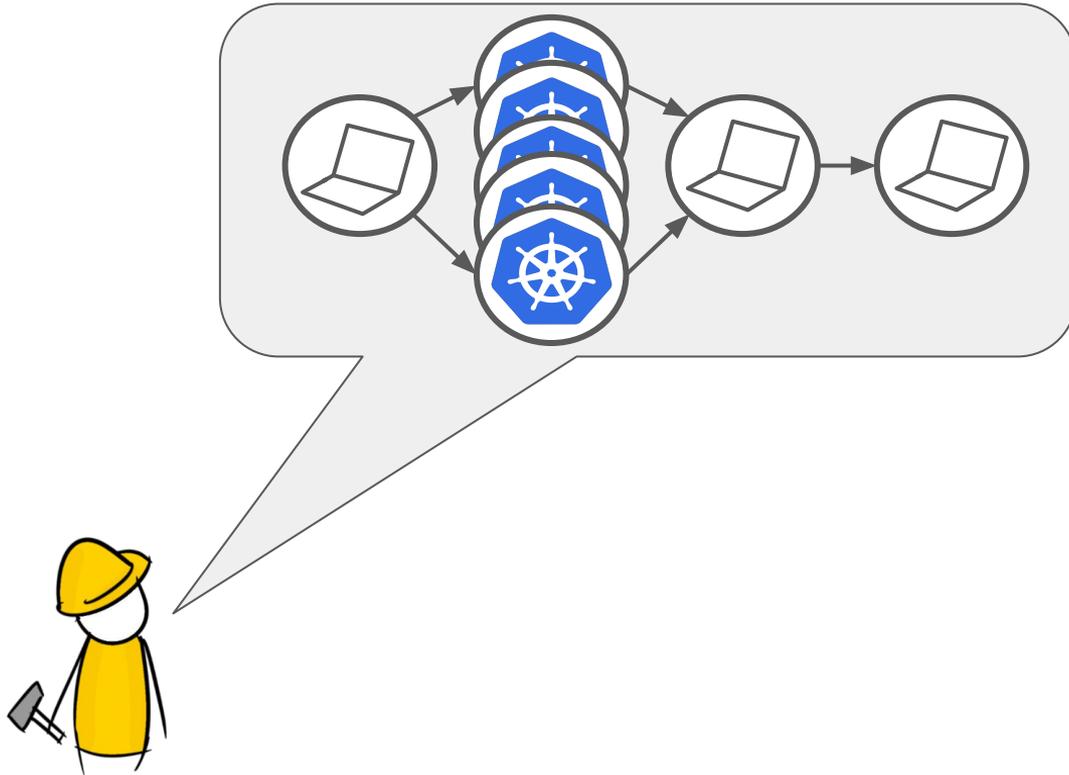


```
1  apiVersion: batch/v1
2  kind: Job
3  metadata:
4    name: pi
5  spec:
6    template:
7      spec:
8        containers:
9        - name: pi
10         image: perl
11         command: ["perl", "-Mbignum=bpi", "-wle", "print bpi(2000)"]
12         restartPolicy: Never
13     backoffLimit: 4
```

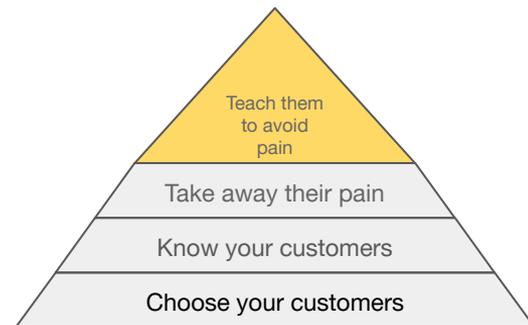
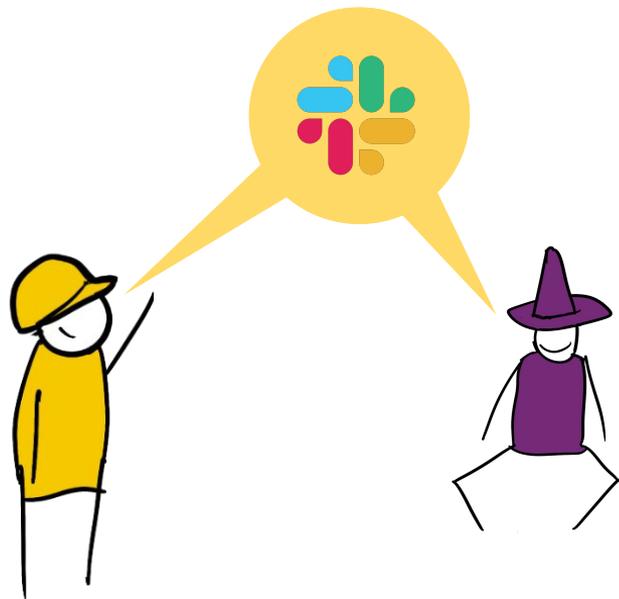
Data Scientists told their colleagues about the work they accomplished while using Metaflow



Metaflow engineers only marketed features that we could support forever!



The Metaflow team provided fanatical customer support



Nov 28th at 8:15 AM

Hi folks, I realize this question might be difficult to answer without more context but I'm looking for broad strokes... How do people generally organize their flows? Say you have one process performing some kind of ETL, another that is enhancing the artifact of that ETL with some meta-analysis, and another that's doing post-processing on some prior artifacts. Do you try to represent them all in one flow? Do you have several flows that are configured independently and then an over-arching "master" flow that can coordinate the whole thing when needed? If anyone has any insight, examples, or resources I could dig into I appreciate it. Thanks (edited)

✓ 1 🗨️

4 replies

 **ville** 2 days ago

I'd love to hear from other folks too, but I can quickly comment that it is very typical to have multiple flows. For instance, one flow to [train a model](#) and [another flow that uses it](#) to produce fresh predictions.

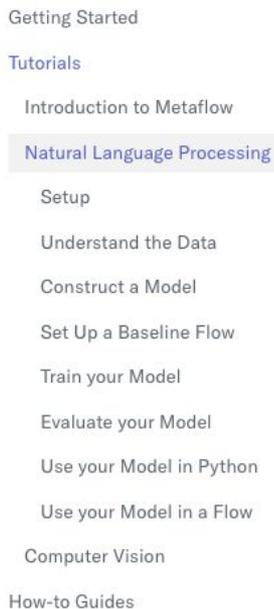
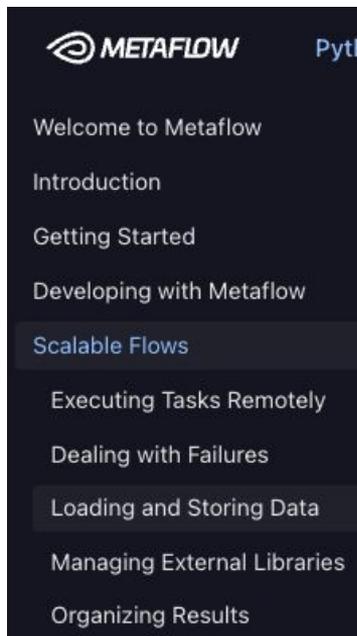
[classifier_train.py](#)
outerbounds/dsbook · Added by GitHub

[classifier_predict.py](#)
outerbounds/dsbook · Added by GitHub

👍 1 🗨️

Further reading: [Rackspace on Fanatical Customer Service](#)

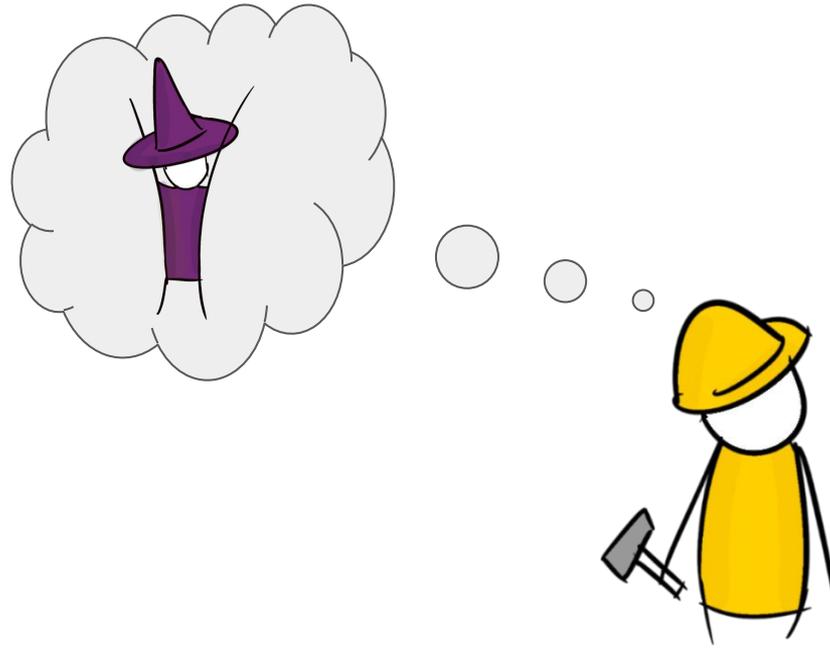
Metaflow engineers treated education as part of of the product



Documentation

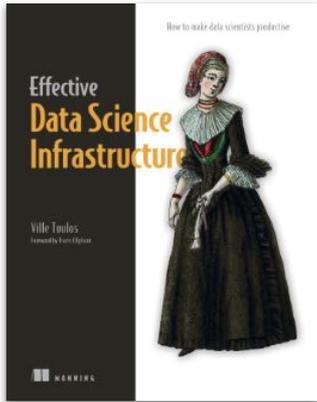
Tutorials

Classes



Want to learn more about Metaflow?

docs.metaflow.org
outerbounds.com



Effective Data Science Infrastructure
New book, by **Ville Tuulos**



Big thanks to Ville, who let me use many of his slides and illustrations in this deck!



I'm looking for my next adventure!

Find me after the talk, or visit my LinkedIn:

[linkedin.com/in/julieamundson](https://www.linkedin.com/in/julieamundson)

Questions?