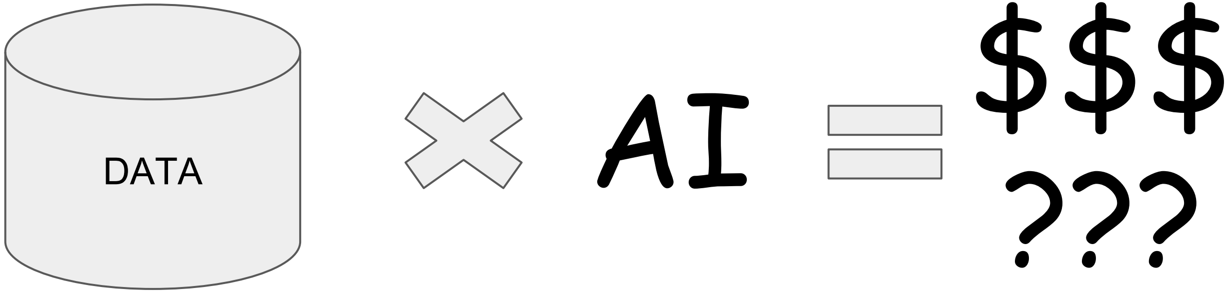
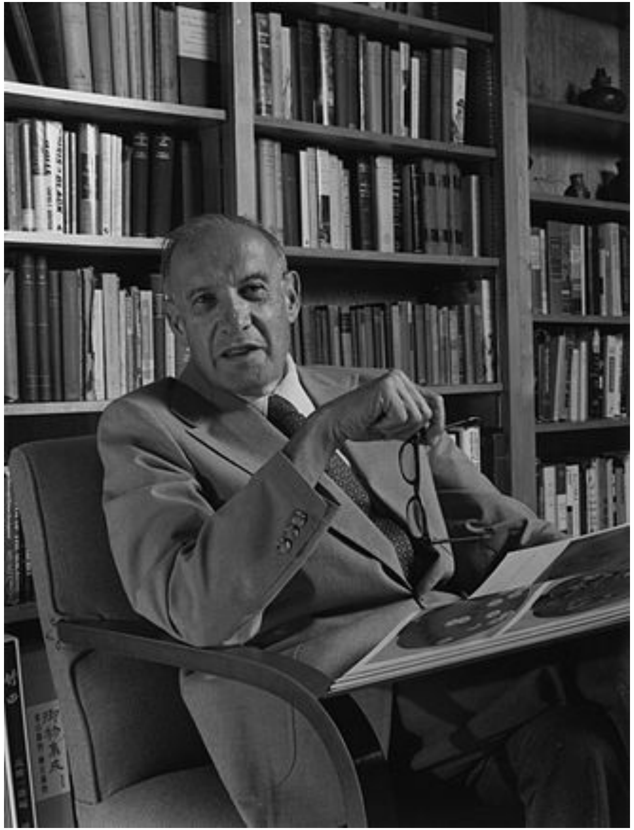
Asset Management for Machine Learning

Georg Hildebrand


DATA
AI
$$$
???

The Big Picture:

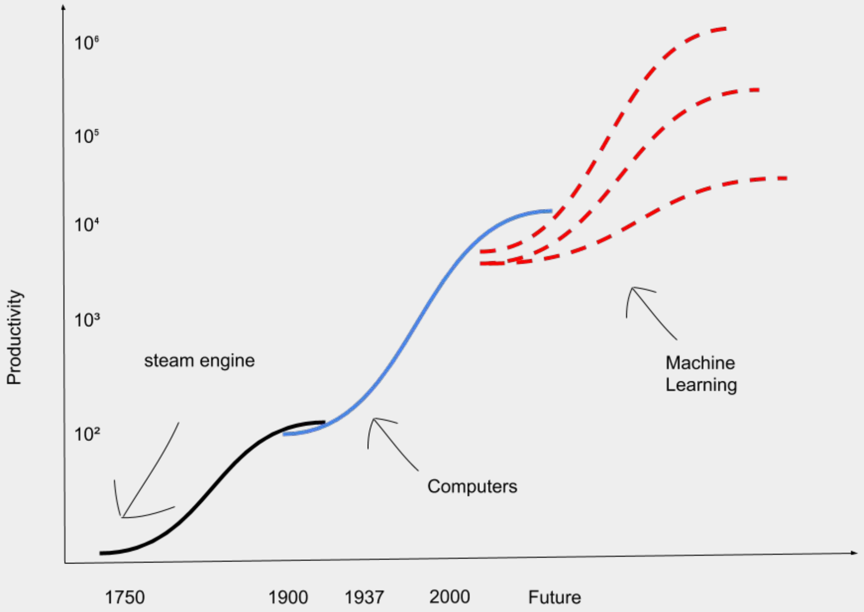
ML
=
Knowledge Workers Productivity
x 50 ??



Peter Drucker source

Increase of Performance

Acceleration through automatisation

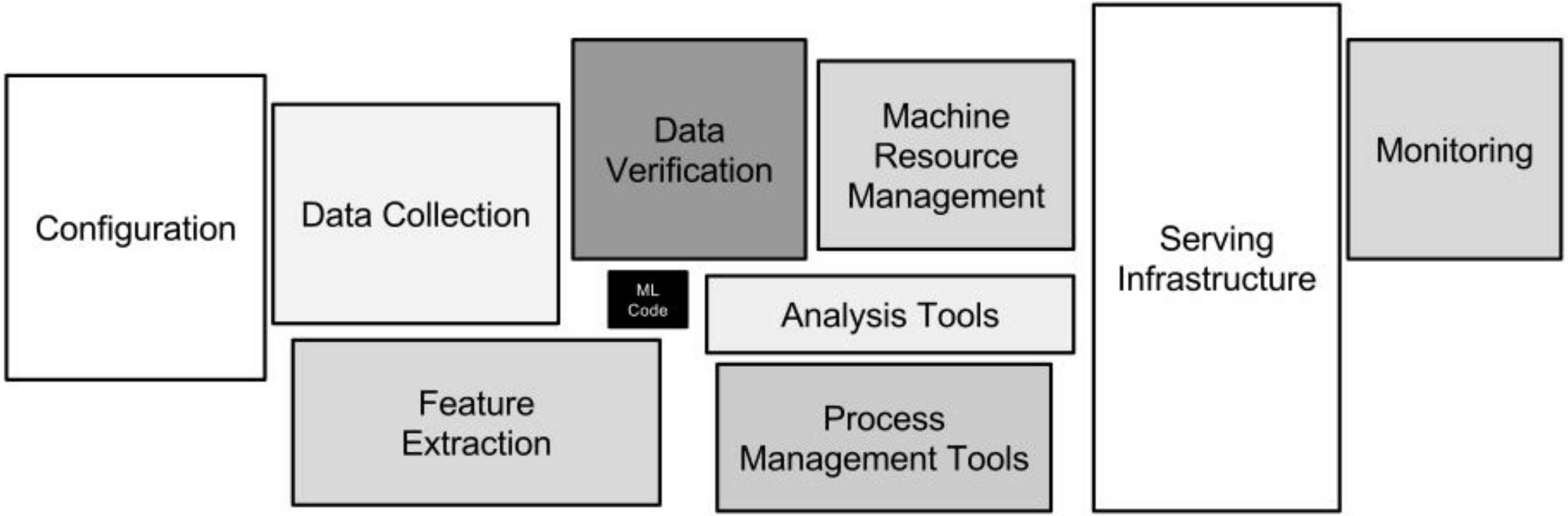


The area of knowledge workers

… the story of a model that was sent as email attachment …

… the story of features that were not available anymore …

Pick a complexity for your ML product:



*D. Sculley *et al.*, “Hidden technical debt in machine learning systems,” in *Advances in Neural Information Processing Systems*, 2015, pp. 2503–2511.

System of ML silos

Hidden depth of
machine learning

Productivity ...

source

How to reach

50 x productivity

?

… make models and features human and machine readable!

Store context

- How was the input data chosen?
- Who trained the model? Is there a project paper?
- How to run the model?
- Store log output of model training
- Framework used to build the model.
- **Chain model and feature preprocessing!**

Version and test

- Snapshot input data /features

- Version the model (eg. into S3 or a backend database)
- Link the container artifact.
- **Provide input and output tests**
- Store performance metrics
- …

Put it into practice

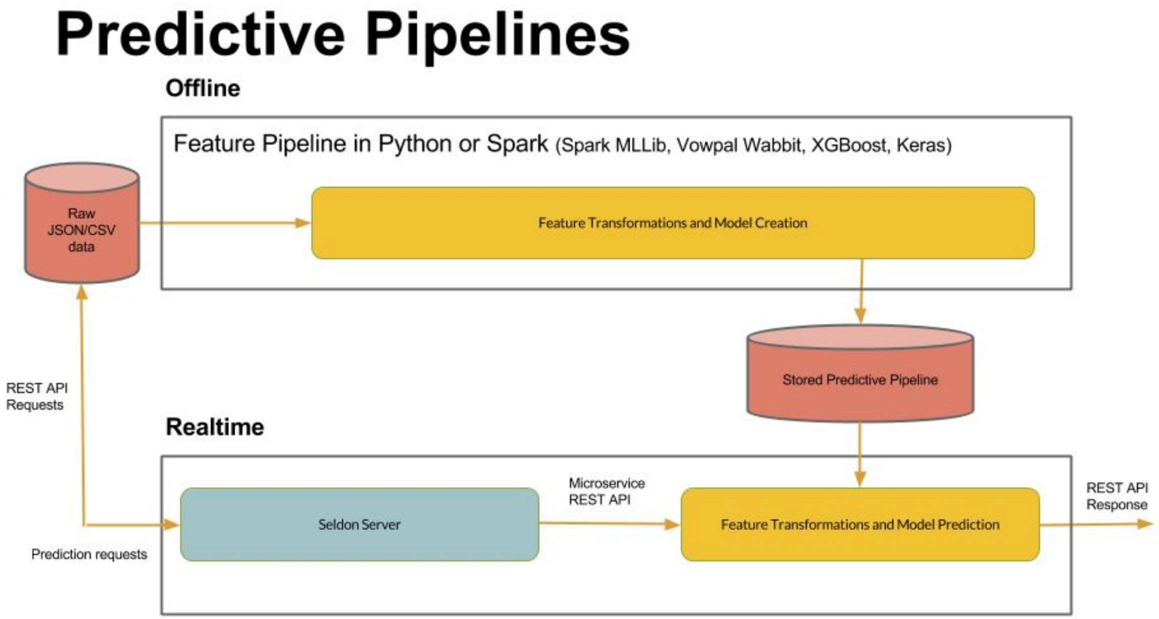
There is no one stop shop solution :-/

Manage ML Deployment pipeline:
Example: SeldonIO, serving and integrating ML
Purpose: End to end deployment for serving ML and more, very advanced.

https://github.com/SeldonIO/

Trys use the modularised way!

May require data to move around.

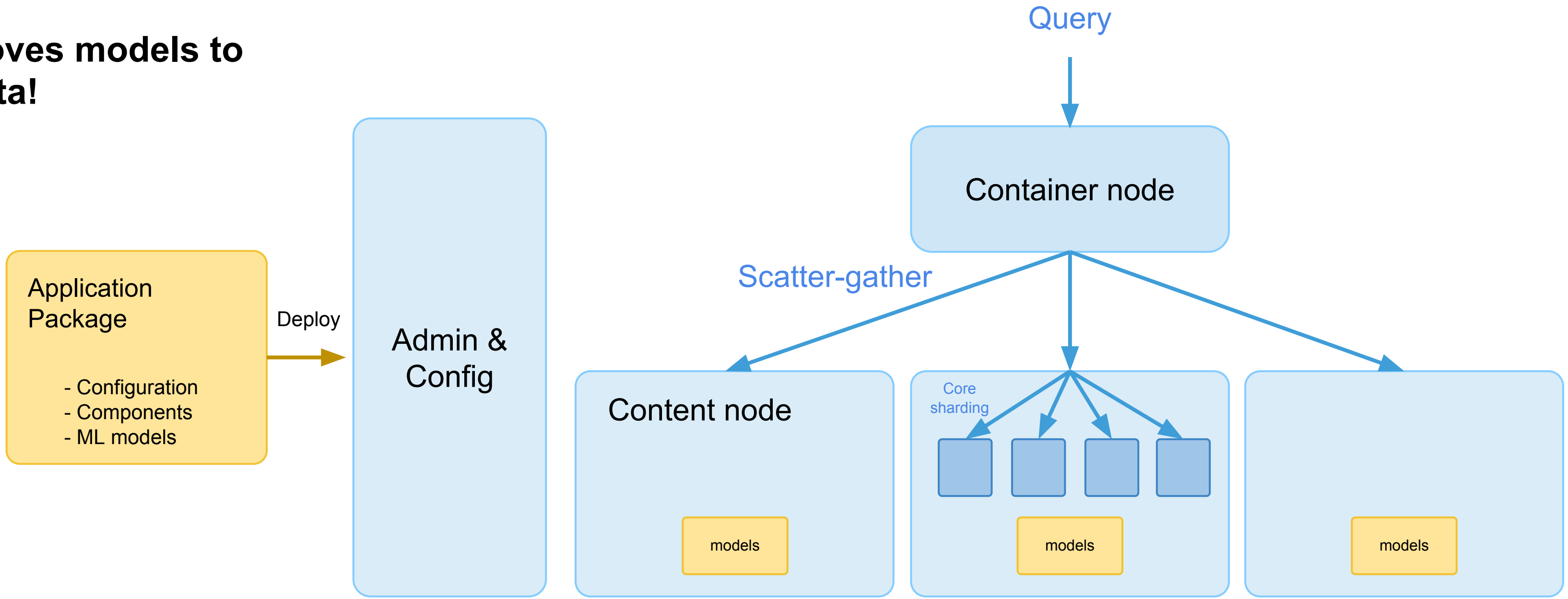


Manage Models + Data and serve low latency:

Example: Vespa, storing model together with the data
Purpose: eg. Low latency serving of ML

https://github.com/vespa-engine/vespa

Moves models to data!

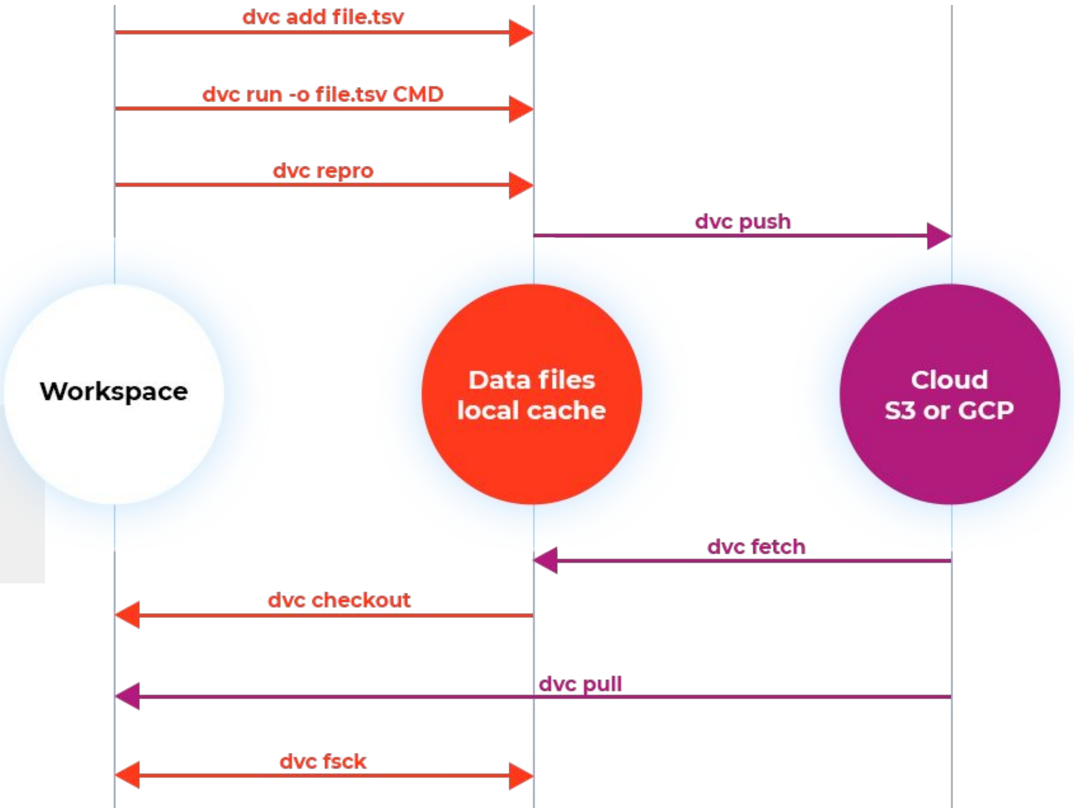


Manage Project, Code, Data and more:
Purpose: Git for data scientists - manage your code and data together

Example DVC (Data Version Control):
https://github.com/iterative/dvc

Input and output are tracked --> DAG !!!

```
$ dvc run python code/xml_to_tsv.py
data/Posts.xml data/Posts.tsv python
```



* source, why luigi and airflow might not be enough...

Model tracking, snapshotting and reproducibility:
Purpose: Git for data scientists - turn any repository into a trackable task record with reusable environments and metrics logging.

Example Datmo:
https://github.com/datmo/datmo

OS, but tight to paid service!!!

Normal Script

```python
# train.py
#
from sklearn import datasets
from sklearn import linear_model as lm
from sklearn import model_selection as ms
from sklearn import externals as ex
#
#
#
#
#
#
iris_dataset = datasets.load_iris()
X = iris_dataset.data
y = iris_dataset.target
data = ms.train_test_split(X, y)
X_train, X_test, y_train, y_test = data
#
model = lm.LogisticRegression(solver="newton-cg")
model.fit(X_train, y_train)
ex.joblib.dump(model, 'model.pkl')
#
train_acc = model.score(X_train, y_train)
test_acc = model.score(X_test, y_test)
#
print(train_acc)
print(test_acc)
#
#
#
#
#
#
#
#
#
```

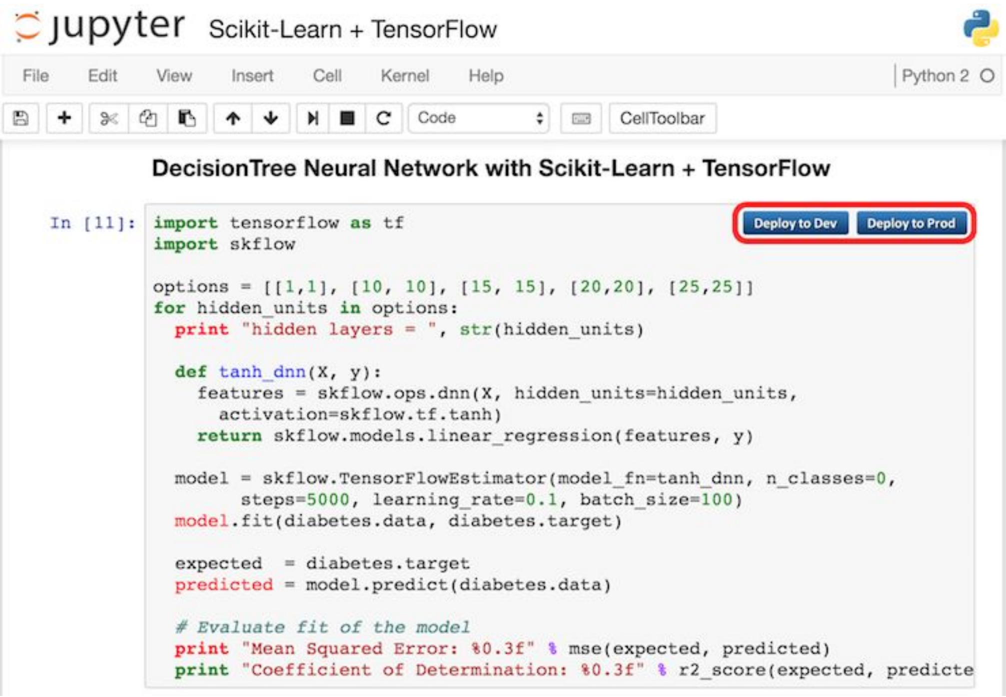
With Datmo

```python
# train.py
#
from sklearn import datasets
from sklearn import linear_model as lm
from sklearn import model_selection as ms
from sklearn import externals as ex
import datmo # extra line
#
config = {
    "solver": "newton-cg"
} # extra line
#
iris_dataset = datasets.load_iris()
X = iris_dataset.data
y = iris_dataset.target
data = ms.train_test_split(X, y)
X_train, X_test, y_train, y_test = data
#
model = lm.LogisticRegression(**config)
model.fit(X_train, y_train)
ex.joblib.dump(model, "model.pkl")
#
train_acc = model.score(X_train, y_train)
test_acc = model.score(X_test, y_test)
#
stats = {
    "train_accuracy": train_acc,
    "test_accuracy": test_acc
} # extra line
#
datmo.snapshot.create(
    message="my first snapshot",
    filepaths=["model.pkl"],
    config=config,
    stats=stats
) # extra line
```

Reproducible Model Pipelines + serving:
Purpose: Consistent, Immutable, Reproducible Model Runtimes

Example PipelineAI:
https://github.com/PipelineAI/pipeline

Models are stored in docker containers!!!



PipelineAI:
https://github.com/PipelineAI/pipeline

Open Source solutions for model / workflow management

Open Source Project	K8s 1st class support	# Active Maintainers	# Commits	1st Commit	Comp
Seldon.IO	Yes + Kubeflow	3	1131	2017-Feb	#6
H2o driverless-ai	Yes + Kubeflow	14	22967	2014-Mar	#12
PipelineAI	Yes. No Kubeflow	1	4846	2015-Jul	#273
Polyaxon	Yes. No Kubeflow	1	2732	2017-May	#88
Vespa-engine	No, but possilbe	14	18290	2016-Jun	#5854
IBM/FfDL	Yes. No Kubeflow	4	bulk-init	hidden	#77
RiseML	Yes. No Kubeflow	3	95	2017-Oct	#9
databricks/mlflow	No. No Kubeflow	5	bulk-init	hidden	#58
datmo	No, possible	4	bulk-init	2018-04	#213

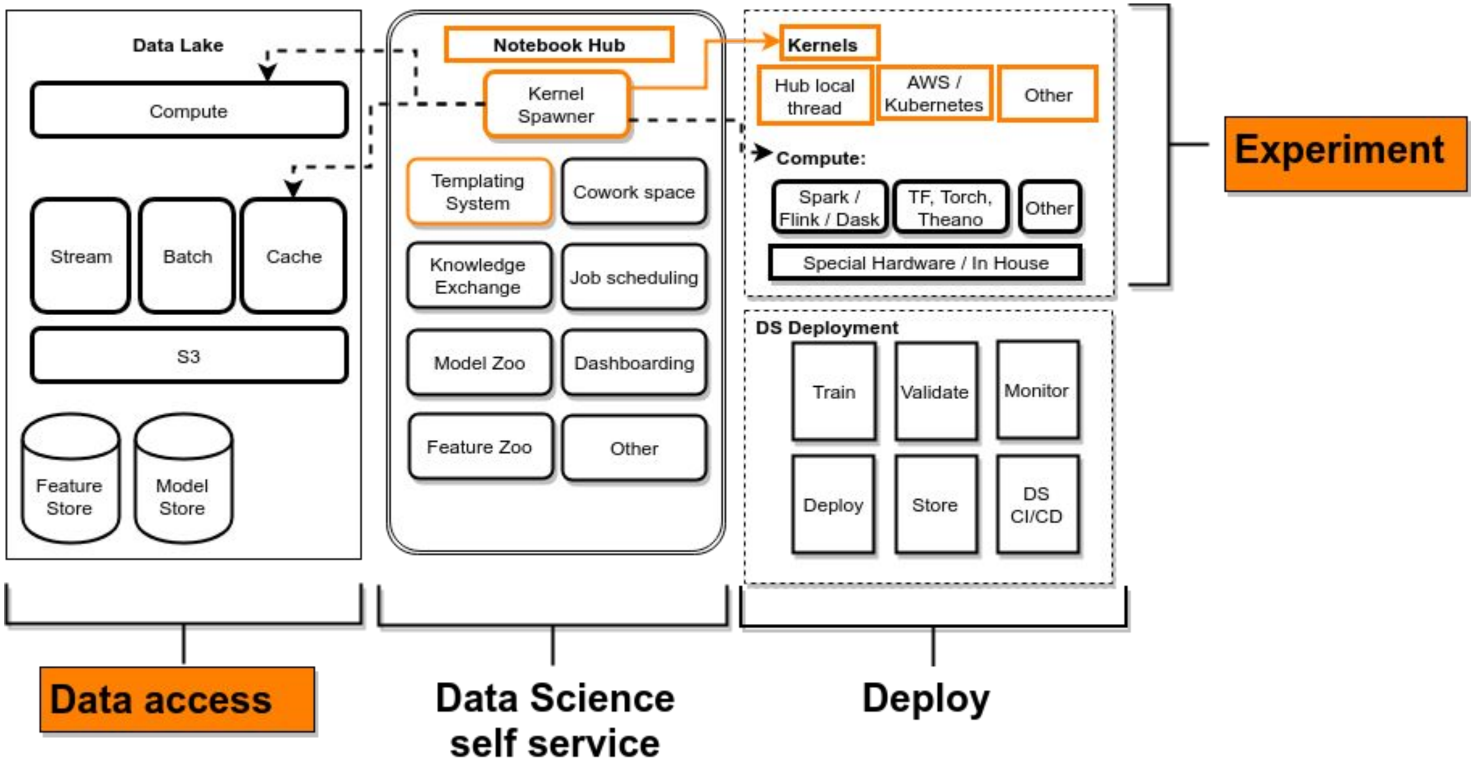
Data / Feature Management

Open Source Project	Note	# Active Maintainers	# Commits	1st Commit	Comp
Data Version Control (DVC)	project and data management	2	1364	2017-Mar	#readme
pachyderm	Data Pipeline, management	6	11652	2014-09	tbd
quiltdata/quilt	Data and package management	8	1177	2017-01	tbd

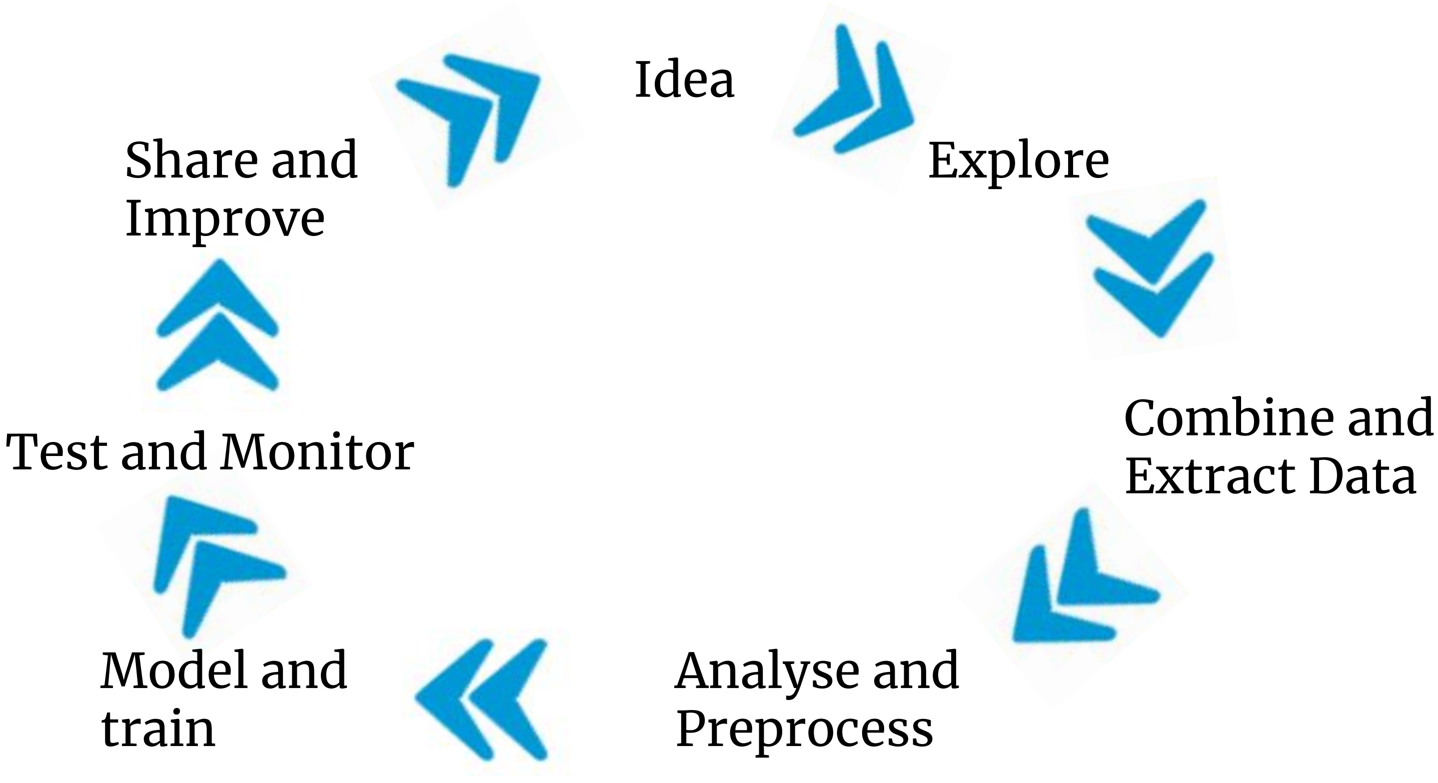
Questions?

Georg Hildebrand, zalando SE
https://github.com/georghildebrand
https://www.linkedin.com/in/georghildebrand/

Example Components



ML Journey


Idea
Explore
Combine and Extract Data
Analyse and Preprocess
Model and train
Test and Monitor
Share and Improve

There is even more hidden depth:

- Models erode boundaries of modularisation:
  context and preprocessing dependencies

- Data Dependencies Cost More than Code Dependencies (unstable data dependencies, data quality monitoring etc.)

- Complex hardware requirements (GPUs on kubernetes??)

- Feedback Loops (model influences its feature data)

- ML-System Anti-Patterns (glue code, pipeline jungles …)

* Own experience
** D. Sculley et al., “Hidden technical debt in machine learning systems,” in Advances in Neural Information Processing Systems, 2015, pp. 2503–2511.
*** D. Sculley et al., “Machine Learning: The High-Interest Credit Card of Technical Debt,” p. 9.