Дизайн систем машинного обучения

14. ML инфраструктура и платформы

Порядок защиты проектов

- Расписание объявим отдельно (возможно, два дня)
- 20 минут на проект.
  - 10 минут рассказ
  - 10 минут вопросы
- Баллы, максимум 40
  - проект работает 20
  - интересный рассказ 10
  - Крутая презентация с дополнительными материалами по теме 10
- Презентацию прислать в день показа чуть заранее
- Попробуем организовать трансляцию и запись

Что должно быть в презентации

- Публичная ссылка на работающий проект
- Команда проекта
- Что строили
- Сценарии использования
- Предположения, на которые опирались при дизайне системы
- Model Card (примеры в 13-й лекции)
  - Использованные данные
  - Оборудование и программное обеспечение
  - Самое важное о внутренностях системы
- С какими проблемами столкнулись и как их решали

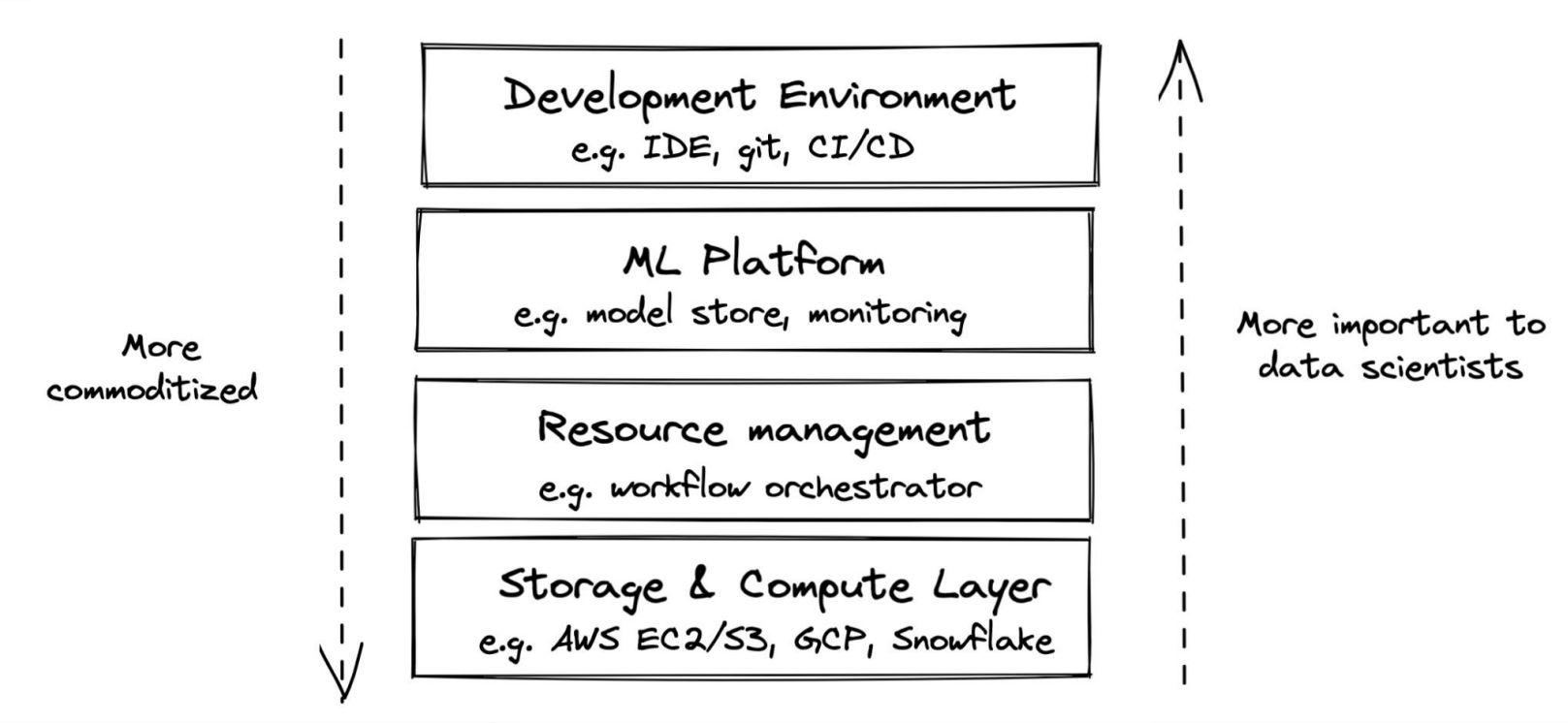
План курса

1) ~~Практическое применение машинного обучения~~
2) ~~Основы проектирования ML-систем~~
3) ~~Обучающие данные~~
4) ~~Подготовка и отбор признаков~~
5) ~~Выбор модели, разработка и обучение модели~~
6) ~~Оценка качества модели~~
7) ~~Развертывание~~
8) ~~Диагностика ошибок и отказов ML-систем~~
9) ~~Мониторинг и обучение на потоковых данных~~
10) ~~Жизненный цикл модели~~
11) ~~Отслеживание экспериментов и версионирование моделей~~
12) ~~Сложные модели: временные ряды, модели над графами~~
13) ~~Непредвзятость, безопасность, карточки моделей~~
14) **ML инфраструктура и платформы — Вы находитесь здесь**
15) Интеграция ML-систем в бизнес-процессы

ML инфраструктура и платформы

- Инфраструктура:
  комплекс взаимосвязанных обслуживающих структур
  или объектов, составляющих и обеспечивающих основу функционирования системы
- ML инфраструктура: набор обеспечивающих систем и инструментов для поддежки жизненного цикла системы

Составные части инфраструктуры



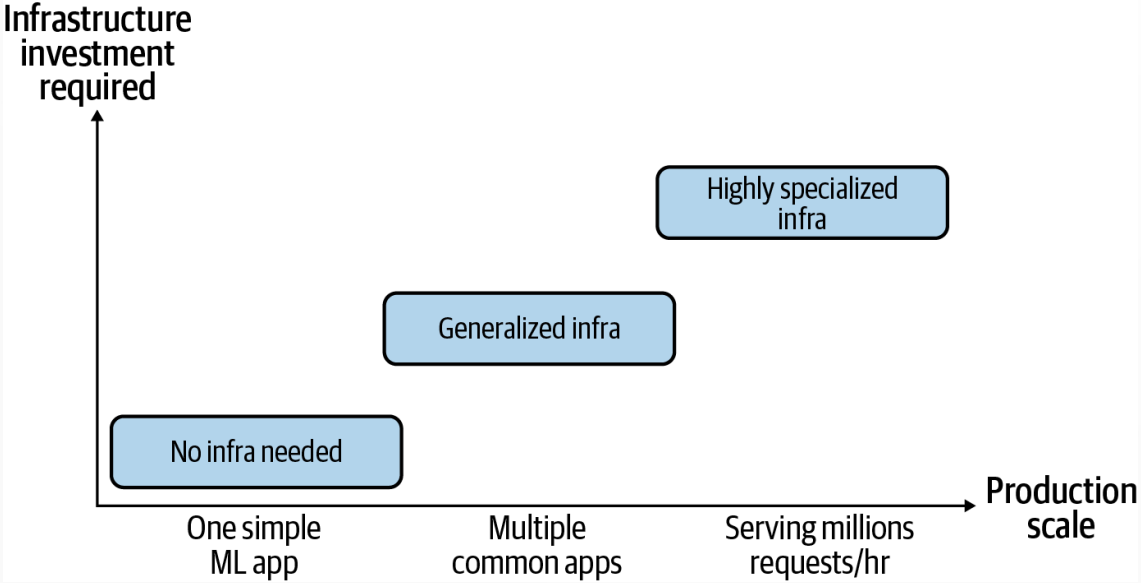
Chip Huyen CS 329

Какие проблемы решает инфраструктура

- Ускорение и воспроизводимость процессов
- Снижение порога входа для новых участников проекта
- Снижение затрат на разработку, развертывание и поддержку
- Экономия времени
- Повторное использование инженерных решений
- Единый набор подходов и инструментов

Разные потребности

- Большие компании разрабатывают свою инфраструктуру
- Маленькие стараются использовать готовую инфраструктуру



Chip Huyen CS 329

Уровень хранения

- Где хранятся данные
- HDD/SSD
- S3 / GS / Yandex Object Storage etc
- BigQuery
- Snowflake / RedShift

- Сравнительно дешевое место, дорогой трафик/обработка

Уровень вычислений

- CPU / GPU Baremetal servers
- vCPU / vGPU Virtual Servers
- App Runner / Cloud Run / Serverless Containers
- AWS Lambda / Cloud Functions /Yandex Cloud Functions

Вычисления: процессор

- Тактовая частота
- Архитектура
- Количество ядер
- Hyper Threading →
- Размер кеша L1 L2 L3

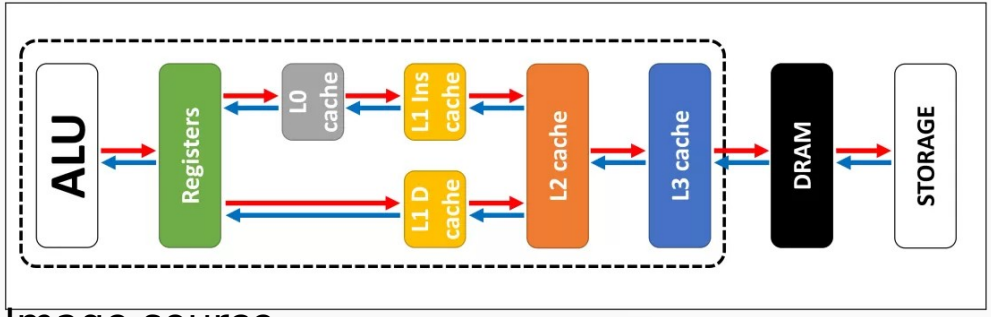


Image source
https://www.techspot.com/article/2066-cpu-l1-l2-l3-cache/

cat /proc/cpuinfo

```
processor       : 31
vendor_id       : GenuineIntel
cpu family      : 6
model           : 45
model name      : Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
stepping        : 6
microcode       : 0x61d
cpu MHz         : 2600.143
cache size      : 20480 KB
physical id     : 1
siblings        : 16
core id         : 7
cpu cores       : 8
apicid          : 47
initial apicid  : 47
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe s
yscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 mo
nitor ds_cpl vmx smx est tm2 ssse3 cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic popcnt tsc_deadline_timer aes xsave avx lahf_lm epb pti ssb
d ibrs ibpb stibp tpr_shadow vnmi flexpriority ept vpid xsaveopt dtherm ida arat pln pts flush_l1d
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf
bogomips        : 5202.05
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

Например: MongoDB требует AVX

x86_64

MongoDB requires the following minimum `x86_64` microarchitectures: [3]

- For Intel `x86_64`, MongoDB requires one of:
  - a *Sandy Bridge* or later Core processor, or
  - a *Tiger Lake* or later Celeron or Pentium processor.
- For AMD `x86_64`, MongoDB requires:
  - a *Bulldozer* or later processor.

Starting in MongoDB 5.0, `mongod`, `mongos`, and the legacy `mongo` shell no longer support `x86_64` platforms which do not meet this minimum microarchitecture requirement.

- MongoDB only supports Oracle Linux running the Red Hat Compatible Kernel (RHCK). MongoDB does **not** support the Unbreakable Enterprise Kernel (UEK).
- MongoDB 5.0 requires use of the AVX instruction set, available on select Intel and AMD processors.

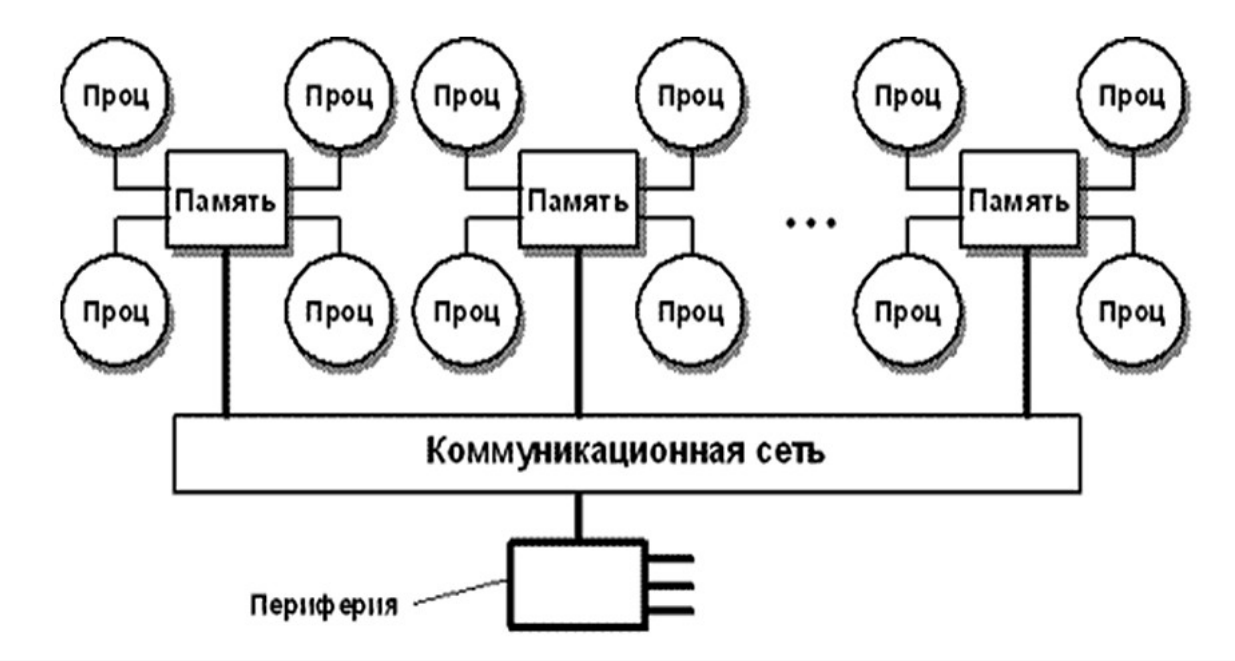
ARM64

MongoDB on `arm64` requires the *ARMv8.2-A* or later microarchitecture.

Starting in MongoDB 5.0, `mongod`, `mongos`, and the legacy `mongo` shell no longer support `arm64` platforms which do not meet this minimum microarchitecture requirement.

https://www.mongodb.com/docs/manual/administration/production-notes/

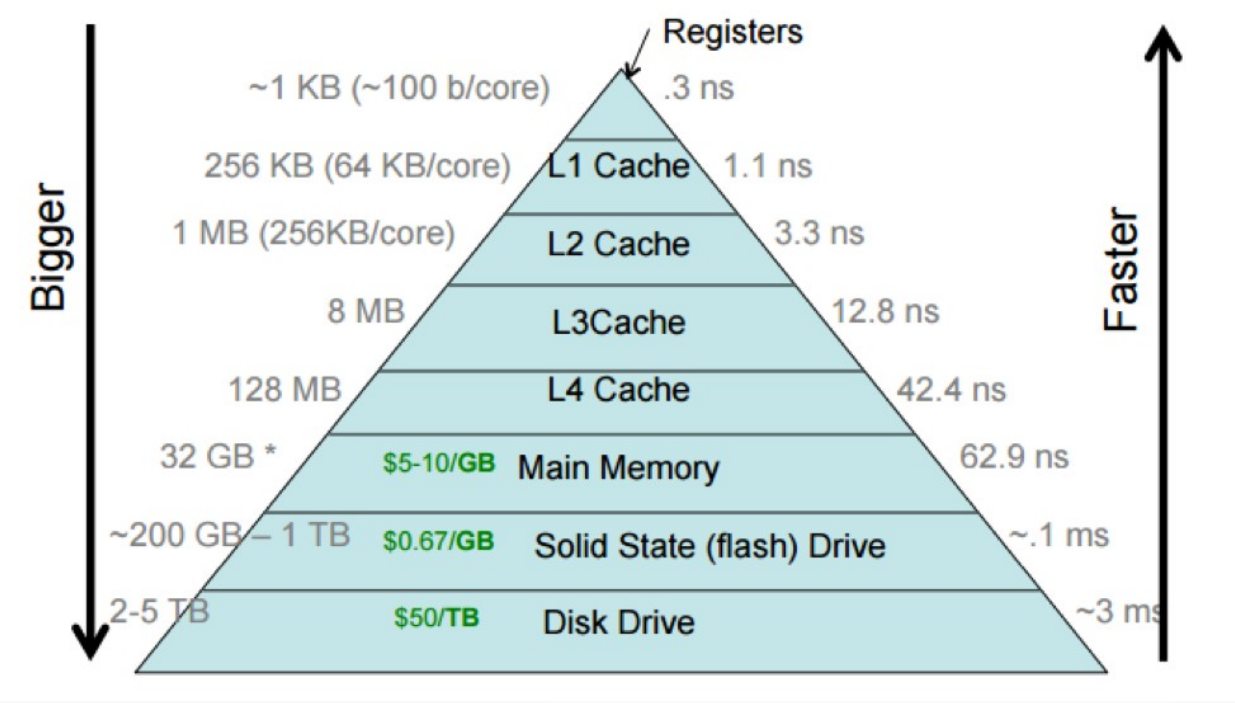
NUMA



https://ru.wikipedia.org/wiki/Non-Uniform_Memory_Access

https://www.mongodb.com/docs/manual/administration/production-notes/#mongodb-and-numa-hardware

Вычисления: память



https://cs61.seas.harvard.edu/site/2018/Storage2/

Память

- RAM
  - Размер доступной памяти
  - Пропускная способность
- GPU RAM
  - Хранит модель и батчи
  - Можно «объединять» память нескольких GPU
  - Медленный обмен с основной RAM
- GPU часто простаивает из-за медленной загрузки данных

Скорость обмена с памятью

Memory Bandwidth

From the previous section, we have seen that Tensor Cores are very fast. So fast, in fact, that they are idle most of the time as they are waiting for memory to arrive from global memory. For example, during BERT Large training, which uses huge matrices — the larger, the better for Tensor Cores — we have a Tensor Core TFLOPS utilization of about 30%, meaning that 70% of the time, Tensor Cores are idle.

This means that when comparing two GPUs with Tensor Cores, one of the single best indicators for each GPU's performance is their memory bandwidth. For example, The A100 GPU has 1,555 GB/s memory bandwidth vs the 900 GB/s of the V100. As such, a basic estimate of speedup of an A100 vs V100 is 1555/900 = 1.73x.

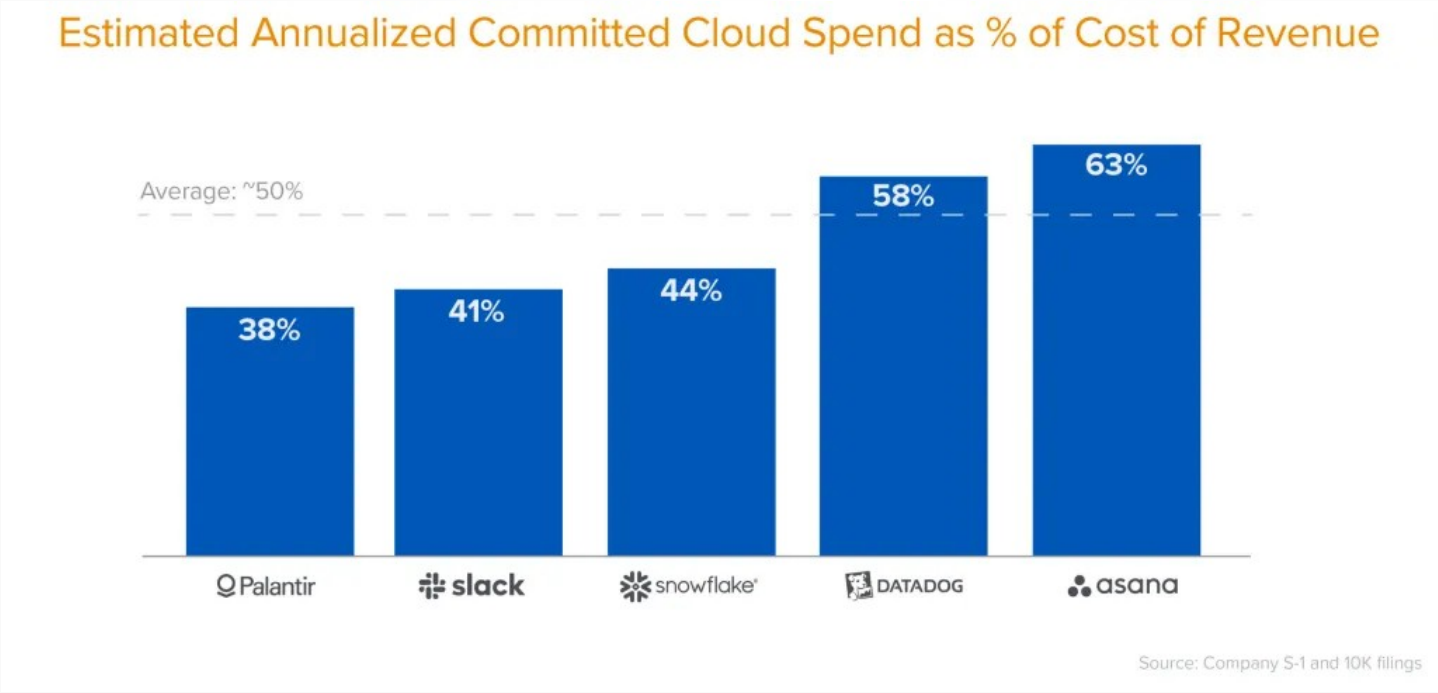
https://timdettmers.com/2020/09/07/which-gpu-for-deep-learning/

Свои сервера или облака

- Сервер в своей стойке на своей территории 1X
- Арендованный в стойке в датацентре 3X
- Арендованный в облаке 15x

- Облака — очень дорого
- Но свои сервера обычно сильно недогружены
- У маленьких организаций нет выбора, только облака.
- У быстрорастущих стартапов нет выбора — только облака
- Большие организации частично мигрируют из облаков

Затраты на облака



https://a16z.com/2021/05/27/cost-of-cloud-paradox-market-cap-cloud-lifecycle-scale-growth-repatriation-optimization/

Запуск задач

- Запуск по расписанию (cron)
  - не отслеживает взаимосвязи задач
- Планировщик (SLURM)
  - Решает, когда запускать задачу
- Оркестратор (Kubernetes)
  - Решает, где запускать задачу

Разделение условно, планировщики умеют в оркестрацию,
а оркестраторы умеют в планирование и расписание

Workflow management

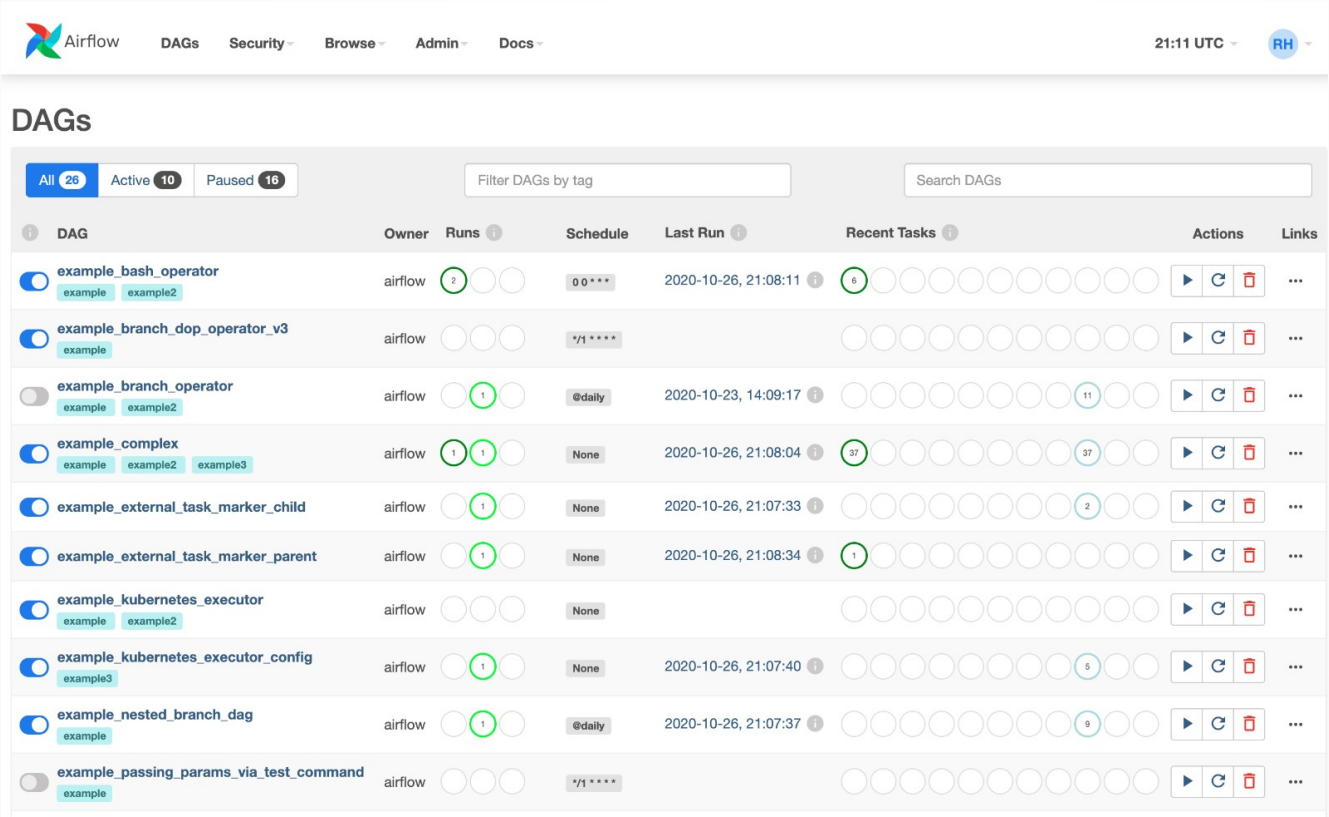
- Этап расчета — task
- Граф последовательного выполнения задач — DAG
- Одно выполнение графа — Flow
- Flow запускается вручную, по расписанию или по внешнему событию («сенсору»)
- Примеры — Airflow, Prefect, Dagster

Airflow

```python
from datetime import datetime

from airflow import DAG
from airflow.decorators import task
from airflow.operators.bash import BashOperator

# A DAG represents a workflow, a collection of tasks
with DAG(dag_id="demo", start_date=datetime(2022, 1, 1), schedule="0 0 * * *") as dag:

    # Tasks are represented as operators
    hello = BashOperator(task_id="hello", bash_command="echo hello")

    @task()
    def airflow():
        print("airflow")

    # Set dependencies between tasks
    hello >> airflow()
```

https://airflow.apache.org/docs/apache-airflow/stable/index.html

Airflow UI

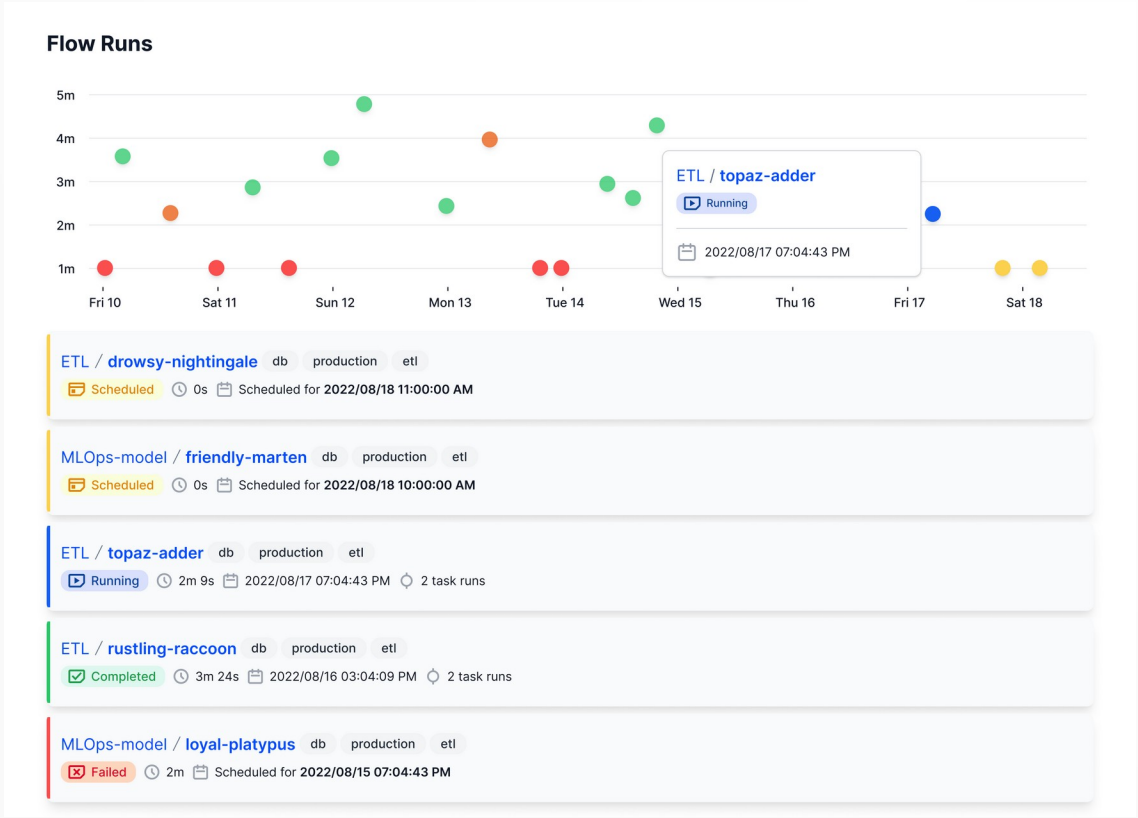

Airflow
DAGs
Security
Browse
Admin
Docs
21:11 UTC
RH
DAGs
All 26
Active 10
Paused 16
Filter DAGs by tag
Search DAGs
DAG
Owner
Runs
Schedule
Last Run
Recent Tasks
Actions
Links
example_bash_operator
example
example2
airflow
0 0 * * *
2020-10-26, 21:08:11
example_branch_dop_operator_v3
example
airflow
*/1 * * * *
example_branch_operator
example
example2
airflow
@daily
2020-10-23, 14:09:17
example_complex
example
example2
example3
airflow
None
2020-10-26, 21:08:04
example_external_task_marker_child
airflow
None
2020-10-26, 21:07:33
example_external_task_marker_parent
airflow
None
2020-10-26, 21:08:34
example_kubernetes_executor
example
example2
airflow
None
example_kubernetes_executor_config
example3
airflow
None
2020-10-26, 21:07:40
example_nested_branch_dag
example
airflow
@daily
2020-10-26, 21:07:37
example_passing_params_via_test_command
example
airflow
*/1 * * * *

Prefect.io

```python
from prefect import flow, task
from prefect.task_runners import SequentialTaskRunner

@task
def first_task(num):
    return num + num

@task
def second_task(num):
    return num * num

@flow(name="My Example Flow",
      task_runner=SequentialTaskRunner(),
)
def my_flow(num):
    plusnum = first_task.submit(num)
    sqnum = second_task.submit(plusnum)
    print(f"add: {plusnum.result()}, square: {sqnum.result()}")

my_flow(5)
```

https://docs.prefect.io/tutorials/flow-task-config/

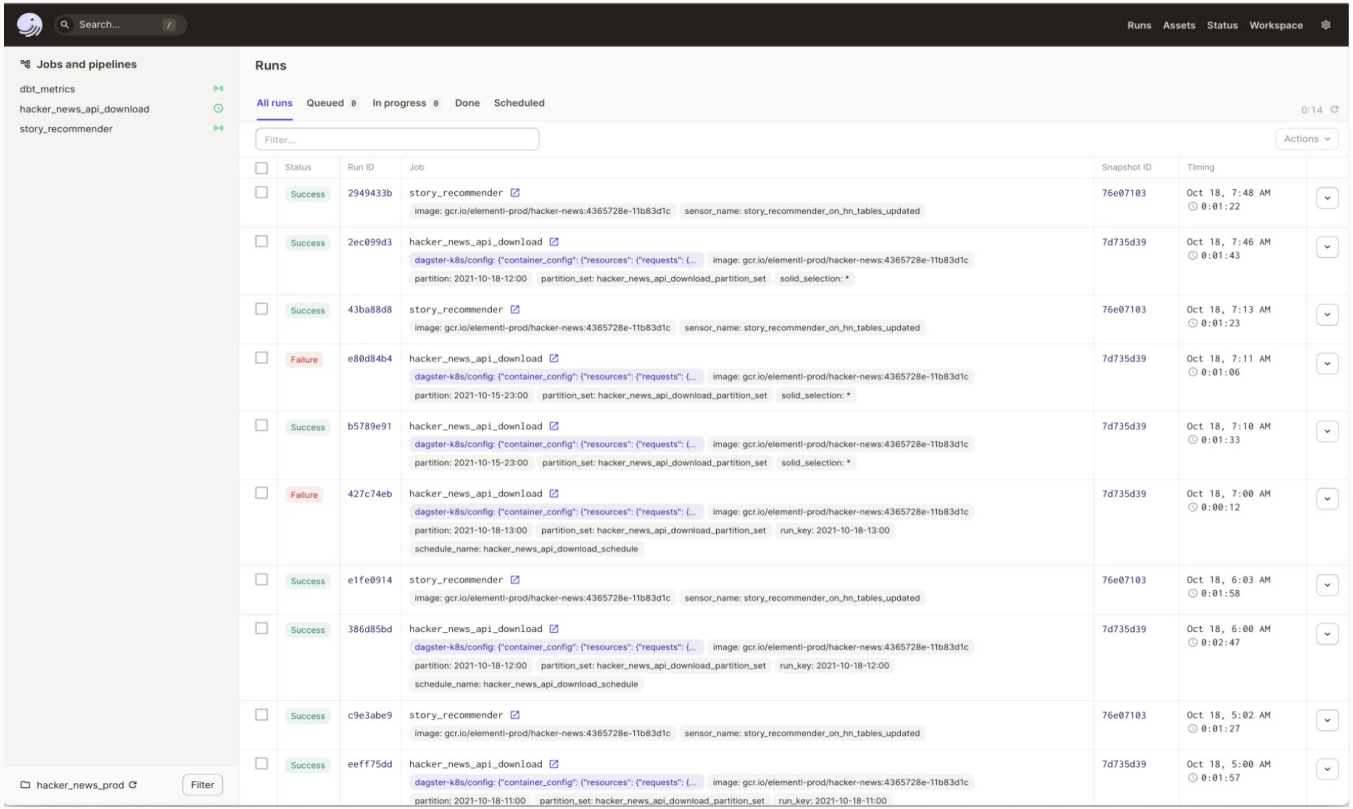
Prefect UI


Flow Runs
5m
4m
3m
2m
1m
Fri 10
Sat 11
Sun 12
Mon 13
Tue 14
Wed 15
Thu 16
Fri 17
Sat 18
ETL / topaz-adder
Running
2022/08/17 07:04:43 PM
ETL / drowsy-nightingale db production etl
Scheduled 0s Scheduled for 2022/08/18 11:00:00 AM
MLOps-model / friendly-marten db production etl
Scheduled 0s Scheduled for 2022/08/18 10:00:00 AM
ETL / topaz-adder db production etl
Running 2m 9s 2022/08/17 07:04:43 PM 2 task runs
ETL / rustling-raccoon db production etl
Completed 3m 24s 2022/08/16 03:04:09 PM 2 task runs
MLOps-model / loyal-platypus db production etl
Failed 2m Scheduled for 2022/08/15 07:04:43 PM

DAGster

```python
from dagster import asset
from pandas import DataFrame, read_html, get_dummies
from sklearn.linear_model import LinearRegression as Regression

@asset
def country_stats() -> DataFrame:
    df = read_html("https://tinyurl.com/mry64ebh")[0]
    df.columns = ["country", "continent", "pop_change"]
    df["pop_change"] = df["pop_change"].str.rstrip("%").astype("float")
    return df

@asset
def change_model(country_stats: DataFrame) -> Regression:
    data = country_stats.dropna(subset=["pop_change"])
    dummies = get_dummies(data[["continent"]])
    return Regression().fit(dummies, data["pop_change"])

@asset
def continent_stats(country_stats: DataFrame, change_model: Regression) -> DataFrame:
    result = country_stats.groupby("continent").sum()
    result["pop_change_factor"] = change_model.coef_
    return result
```

Dagit UI


Search...
Runs
Assets
Status
Workspace
Jobs and pipelines
dbt_metrics
hacker_news_api_download
story_recommender
Runs
All runs
Queued 0
In progress 0
Done
Scheduled
0:14
Filter...
Actions
Status
Run ID
Job
Snapshot ID
Timing
Success 2949433b story_recommender 76e07103 Oct 18, 7:48 AM 0:01:22
Success 2ec099d3 hacker_news_api_download 7d735d39 Oct 18, 7:46 AM 0:01:43
Success 43ba88d8 story_recommender 76e07103 Oct 18, 7:13 AM 0:01:23
Failure e80d84b4 hacker_news_api_download 7d735d39 Oct 18, 7:11 AM 0:01:06
Success b5789e91 hacker_news_api_download 7d735d39 Oct 18, 7:10 AM 0:01:33
Failure 427c74eb hacker_news_api_download 7d735d39 Oct 18, 7:00 AM 0:00:12
Success e1fe0914 story_recommender 76e07103 Oct 18, 6:03 AM 0:01:58
Success 386d85bd hacker_news_api_download 7d735d39 Oct 18, 6:00 AM 0:02:47
Success c9e3abe9 story_recommender 76e07103 Oct 18, 5:02 AM 0:01:27
Success eeff75dd hacker_news_api_download 7d735d39 Oct 18, 5:00 AM 0:01:57
hacker_news_prod
Filter

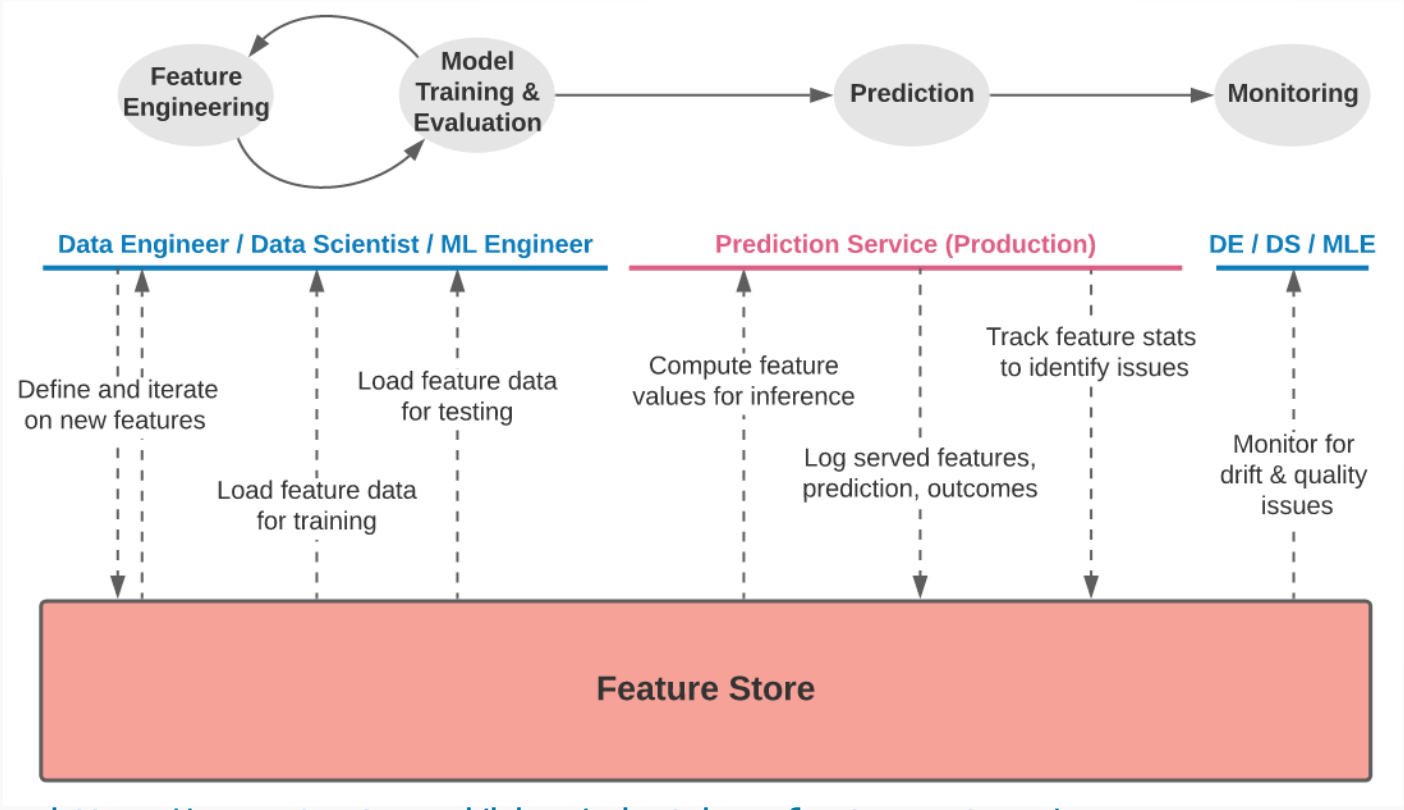
ML платформы

- Ключевые компоненты:
  - Развертывание
  - Хранилище моделей
  - Мониторинг
  - Отслеживание экспериментов
  - Метрики и дашборды
  - Feature Store

Проблемы с признаками

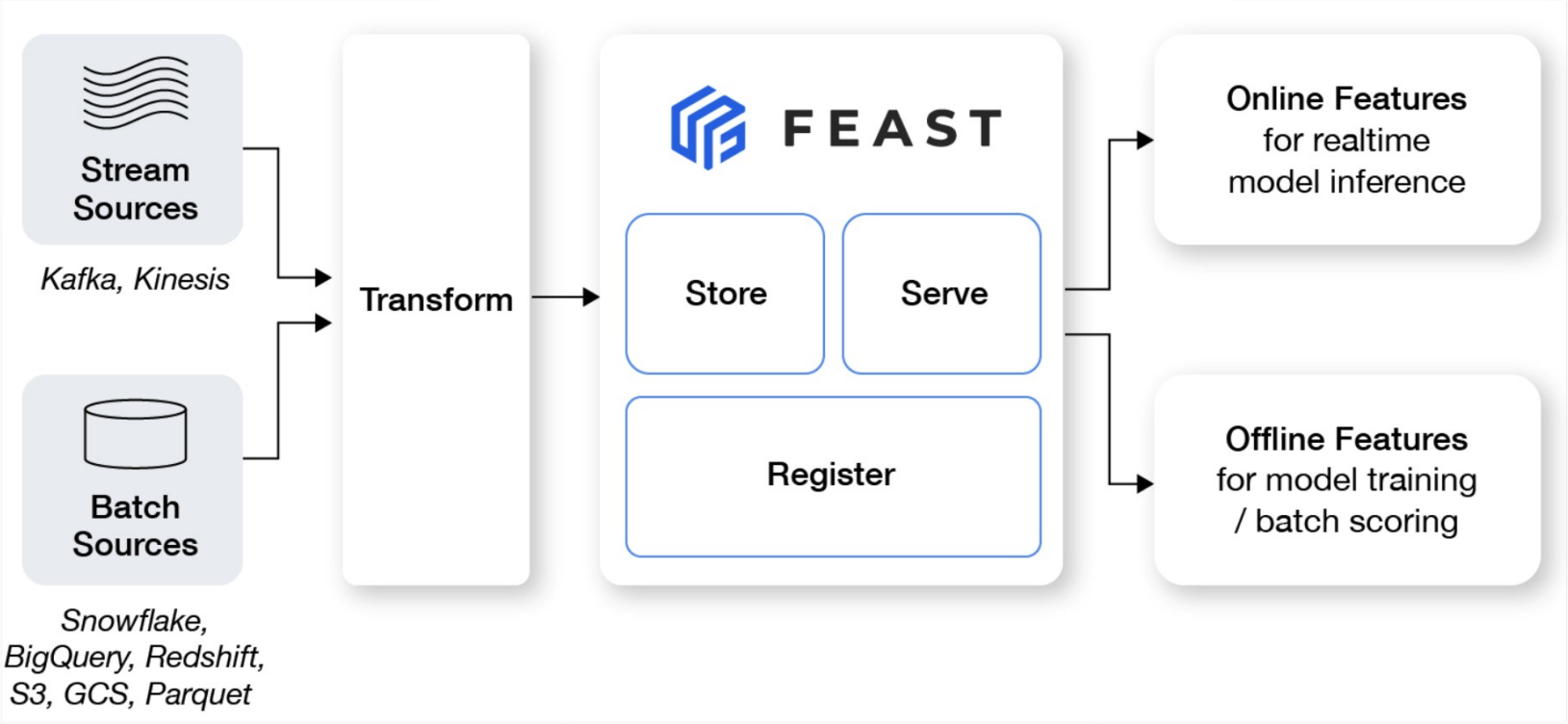
- Разные модели используют одни и те же признаки
- Посчитать один раз и использовать везде
- Снабдить данные документацией
- Дать разработчикам список доступных признаков
- Чтобы признаки у всех считались одинаково
- Чтобы признаки одинаково считались на трейне и на проде

Feature Store



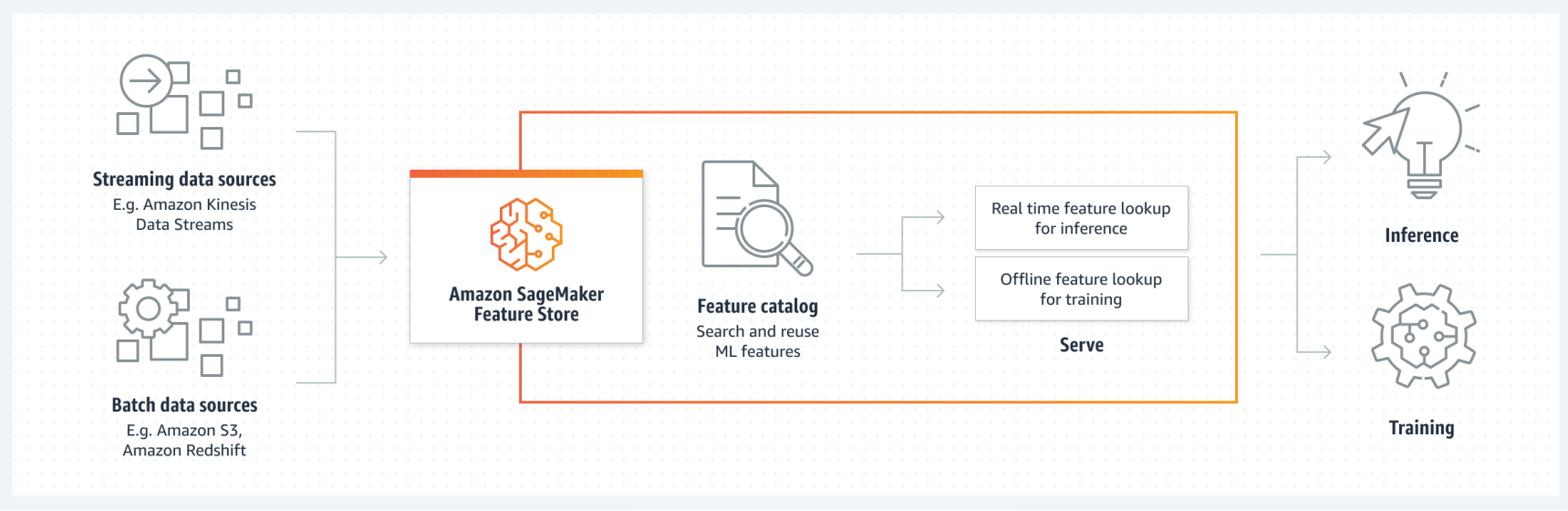
https://www.tecton.ai/blog/what-is-a-feature-store/

Feast



https://docs.feast.dev/

Amazon SageMaker



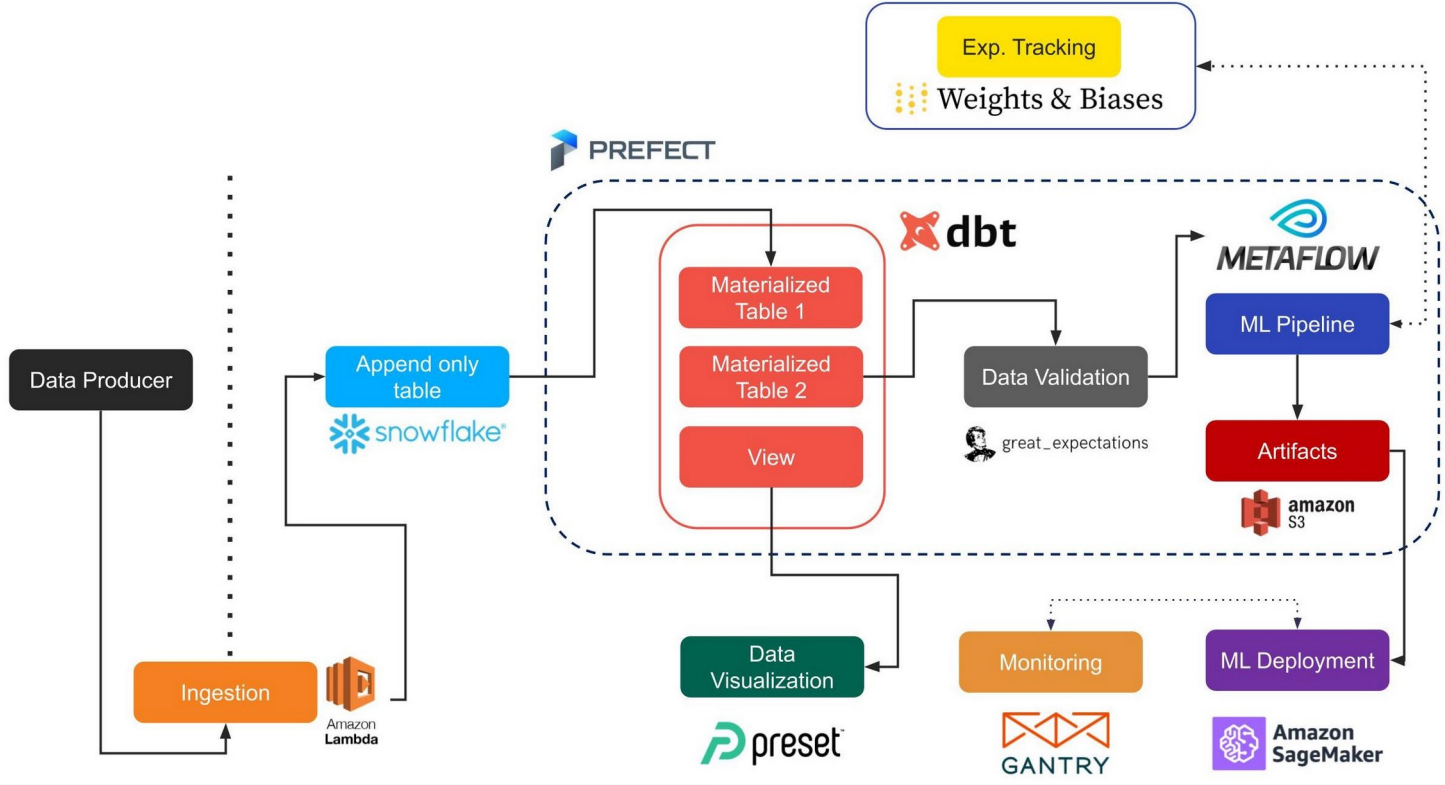
https://aws.amazon.com/sagemaker/feature-store/

И еще feature stores

- Hopsworks
- Google
- Databricks
- Featureform
- Tecton
- Continual

https://www.featurestore.org/

You Don't Need a Bigger Boat



https://github.com/jacopotagliabue/you-dont-need-a-bigger-boat

Дополнительные материалы

- You Do Not Need a Bigger Boat →
- The ML Test Score: A Rubric for ML Production Readiness and Technical Debt Reduction →
- Operationalizing Machine Learning Models: A Systematic Literature Review →
- Serverless on Machine Learning: A Systematic Mapping Study →
- DAG Card is the new Model Card →

Все будет в телеграм-канале