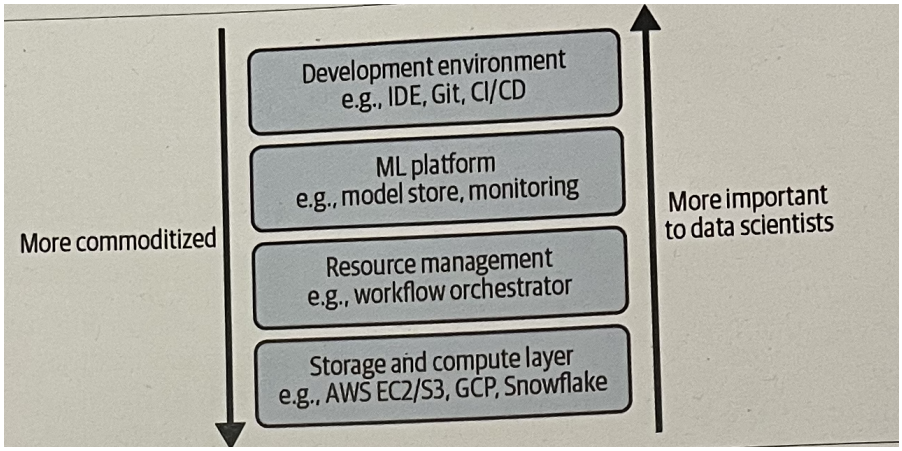
Machine Learning Systems

Created	@April 22, 2023 4:41 PM
Tags	

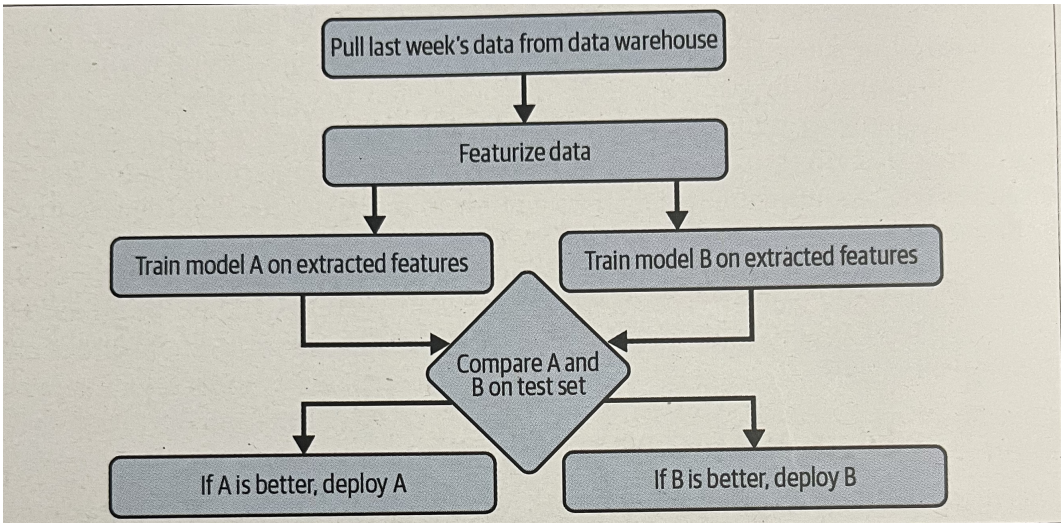
Chapter 10: Infrastructure and Tooling for MLOps

The book categorize ML infrastructure as 4 layers



The four layers

- Storage and compute The storage layer where data is collected and stored, providing the compute needed to run your ML workloads such as training a model. computing features, generating features
    - storage: can be hard drive disk in private data center, or cloud like Amazon S3, Snowflake
    - compute: a single CPU/GPU (for cloud, example includes AWS EC2 and GCP); or a computer cluster like K8S
        - Spark and Ray use “job” as their unit, Kubernetes uses “pod”.
        - while you can have multiple containers in a pod, you can’t independently start/stop different containers in the same pod
- Resource management: Resource management comprises tools to schedule and orchestrate your workloads to make the most out of your available computing resources.
    - cron: scheduling repetitive works
    - scheduler: cron programs that takes DAG (direct acyclic graph)
        - example: pull data, extract feature, train 2 models, and select the better one



- data science workflow management

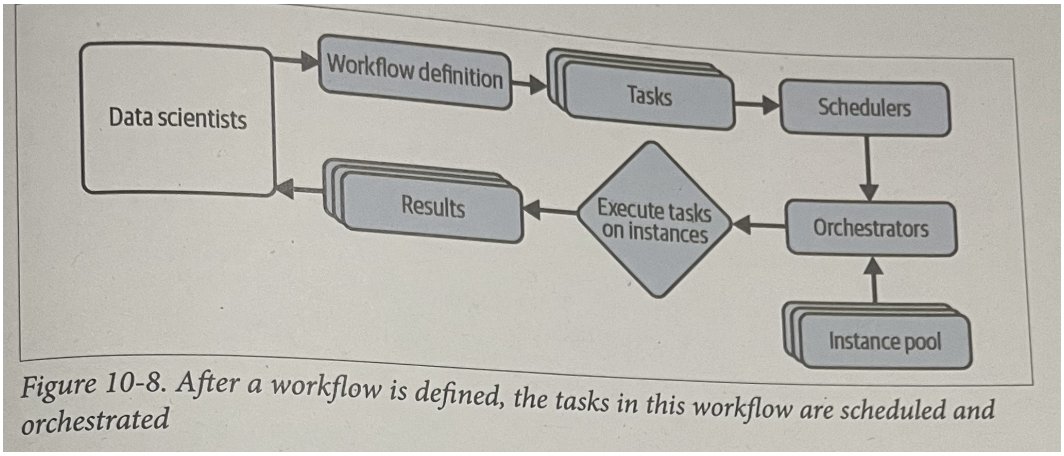


Figure 10-8. After a workflow is defined, the tasks in this workflow are scheduled and orchestrated

  - Tools, typically in python or configurations like YAML: Airflow, Argo, Prefect, Kuberflow, Metaflow

Metaflow example

```
# From the book
# Example: sketch of a recommender system that uses an ensemble of two models.
# Model A will be run on your local machine and model B will be run on AWS.
class RecysFlow(FlowSpec):
  @step
  def start(self):
    self.data = load data()
    self.next(self.fitA, self.fitB)
    # fitA requires a different version of NumPy compared to fitB

  @conda(libraries={"scikit-learn": "0.21.1","numpy": "1.13. 0"])
  @step
  def fitA(self):
    self.model = fit(self.data, model="A")
    self.next(self.ensemble)

  @conda(libraries={"numpy": "0.9.8"7)
  # Requires 2 GPU of 16GB memory
  @batch (gpu=2, memory=16000) @step
```

```
  def fitB(self) :
    self.model = fit(self. data, model="B")
    self.next(self.ensemble)

  @step
  def ensemble(self, inputs):
    self.outputs = (
    (inputs.fitA.model.predict(self.data) + inputs.fitB.model.predict(self.data)) / 2 for input in inputs
    )
    self.next(self.end)

  def end(self):
    print(self.outputs)
```

- ML platform: tools to aid the dev of ML applications such as model stores, feature stores, and monitoring tools
- dev environment: where code is written and experiments are run
  - need to have CI/CD pipeline setup, tools like GitHub Actions and CircleCI
- production environment
  - docker: to keep production environment the same
    - dockerfiles —build → docker images -run-time→ docker containers
  - docker compose: orchestrate different containers on a single host
  - Kubernetes (K8S): dynamically manage containers, allows resource sharing and easy communication
    More details: https://www.jeremyjordan.me/kubernetes/