From organically grown infrastructure, to a mature ML platform

OLIVIA STOICESCU, Sr. MLOps Engineer
23 Feb 2023

Agenda

- What is an ML Platform?
- ML Workflows
- From ML Workflows to a ML Platform
    - What is the Problem you are Solving?
    - Platform Interaction Modes
    - Platform Onboarding
    - Buy vs Build
    - The Path to Platform Enablement
- Conclusion

WHAT is a Platform?

Team Topologies book introduces the concept of **Thinnest Viable Platform (TVP)**:

A TVP is the smallest set of APIs, documentation, and tools needed to accelerate the teams developing modern software services and systems.

TVP

…. does this mean that ***anything*** can be considered a platform??

What is a ML Platform?

A platform which implements DevOps principles to ML workflows.

...however, MLOps is not DevOps:

- ML is experimental in nature
- Testing a ML system is more involved
- Production models might decay due to changing data profiles

ML Workflows

ML workflows include the following key assets: code, models and data.

… and the following stages: train, test, package, deploy, monitor and maintain models

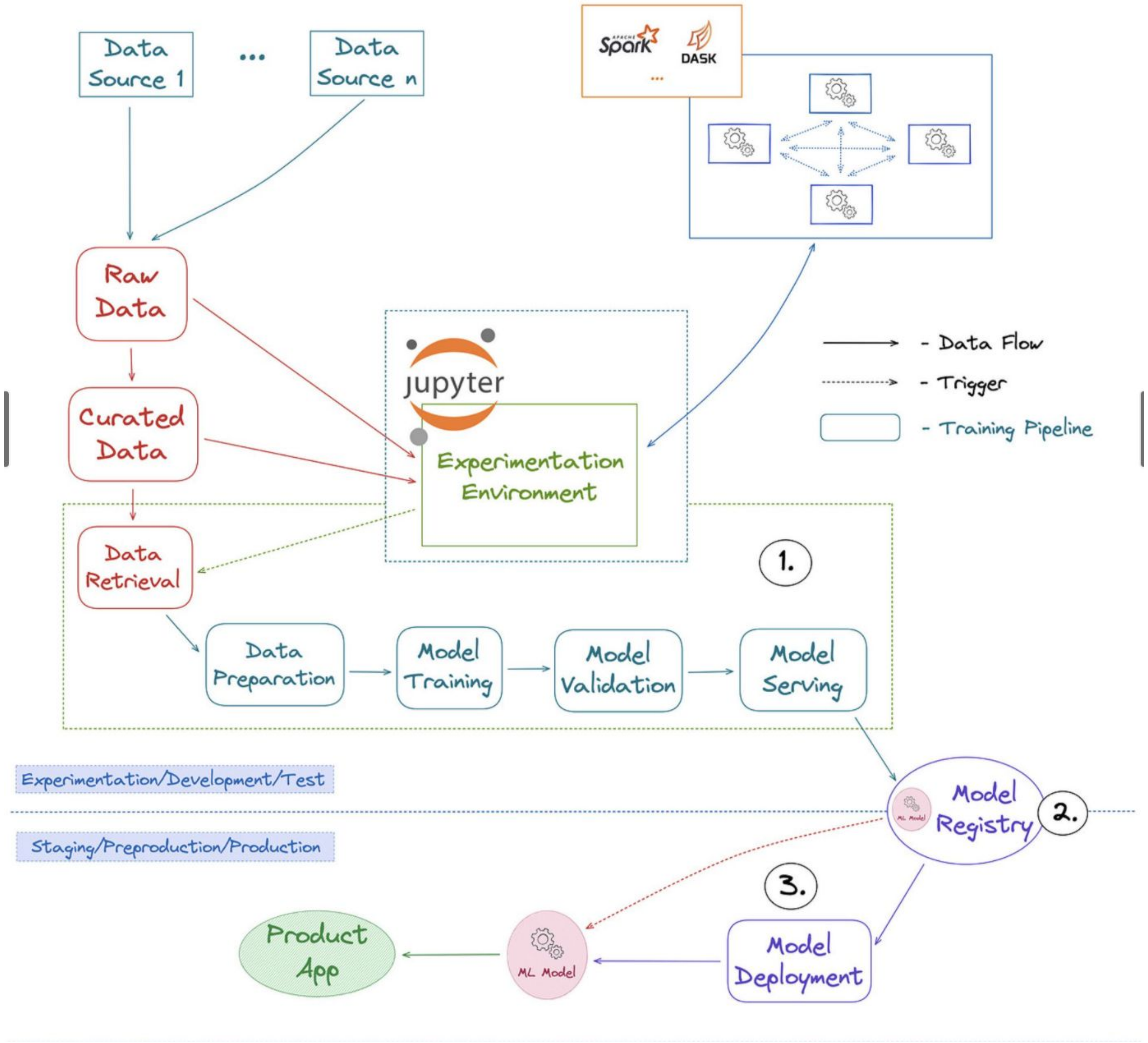
ML Workflow - no automation

1. The entire Machine Learning Pipeline is executed in the Experimentation Environment manually and on demand.

2. After the Model Artifact is created it is saved into a Model Registry

3. Model serving uses already present deployment procedures

Diagram source: https://www.newsletter.swirlai.com/p/sai-12-cap-theorem



ML Workflows - automation

Implementing ML in a production environment doesn't only mean **deploying your model** as an **API** for prediction.

It means deploying an **ML pipeline** that can automate the retraining and deployment of new models.

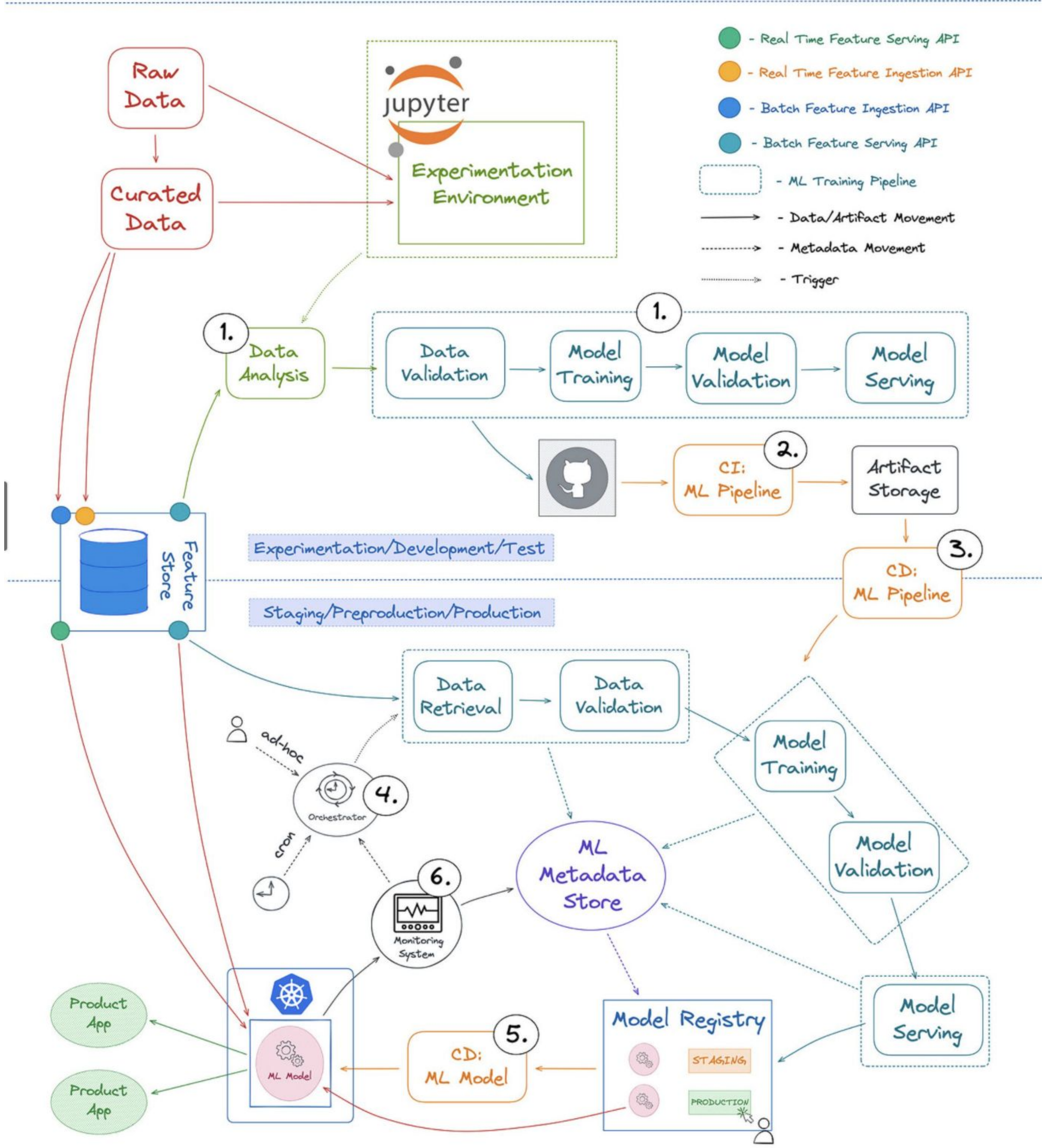
Setting up a **CI/CD system** enables you to automatically test and deploy new pipeline implementations.

- This system lets you cope with rapid changes in your data and business environment.
- You can gradually implement these practices to help improve the automation of your ML system development and production.

ML workflow - Mature

1. ML Pipelines are orchestrated but you can trigger them from the experimentation environment - Notebooks.
2. Continuous Integration Step for ML Pipelines
3. Continuous Delivery/ Deployment Step for ML Pipelines.
4. Automated triggering of the ML Pipeline.
5. ML Model Continuous Delivery.
6. Performance Monitoring.

Diagram source: https://www.newsletter.swirlai.com/p/sai-14-data-latency-in-ml-systems



From ML Workflows to an ML Platform

What problem are you solving?

👉 **Enable scalling** - allow the internal customers to become self-sufficient in their ML journey

👉 **Improving the ML output quality** - eg. moving from batch to real-time processing

👉 **Improve developer experience** - eg. automation, minimize cognitive load

Platform Interaction Modes

Understanding who is the key **internal customer** will help understand the optimal interaction mode:

`data-scientists, data analysts`

- can have have limited experience with the software life-cycle
- Lead the interaction via configuration

`(ML) engineers`

- more experienced with software lifecycle and tooling
- Build apis and cli tools

Platform Onboarding

The platform should aim to enable **fast onboarding**:

- Automate generating compliant projects (eg. via cookiecutter for python, archetypes for jvm)
- Standardize dependency management to scale fast to new projects (eg. via a parent pom for jvm, or using tools like poetry for python)
- Standardize CI/CD pipeline to scale fast to new projects (eg. via templates)
- Use gitOps to better support traceability and change management (eg. by using environment manifest repositories)

Buy vs Build: Open-source

Pros:

- Lots of free extensions, allowing to cover for missing functionality
- Flexibility in customization (you can decide which features to use or leave out)
- Community delivering case studies, tutorials, how-tos

Cons:

- Slower adoption (installation and initial setup can be a bumpy road)
- Maintenance (eg. kubeflow relies on k8s as base platform)

Buy vs Build: Managed

Pros:

Fast adoption

Effort-free features

Strategic partner (support)

Cons:

Vendor selection

Vendor lock-in

Buy vs Build

Most often, the discussion between managed and open-source comes down to **what is the most limited resource you have**.

- A managed MLOps platform minimizes the need for engineering resources but requires a certain amount of investment.
- While open source MLOps platforms are the other way around.

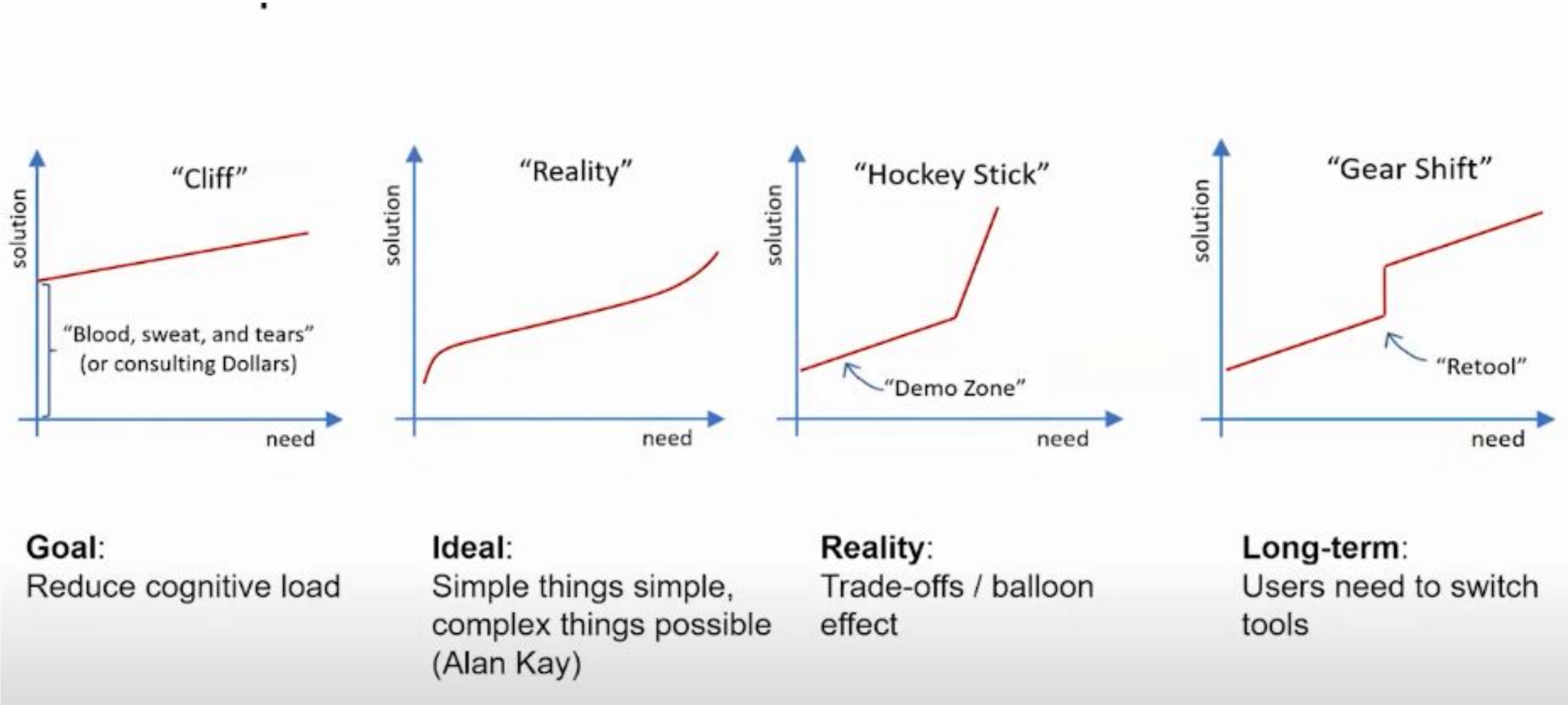
You can choose to apply a blend of the two and go for open-source on the components where in-house expertise is most aligned with the value streams

The Path to Platform Enablement

Typically, we start from some initial ML development setup.

At some point, this setup outgrows its initial purpose and needs to evolve.

Diagram source: https://www.youtube.com/watch?v=WaL3ZbLgMuI



Final thoughts

Favour incremental changes

- Changes are frequent and so they are expected

Favour loose coupling

- build extensible abstraction layers, in order to freely move the technologies behind the scenes

Favour testability of your platform components by design

- embedded visibility - if this changes, what/where is the impact?
- lack of visibility hinders incremental changes

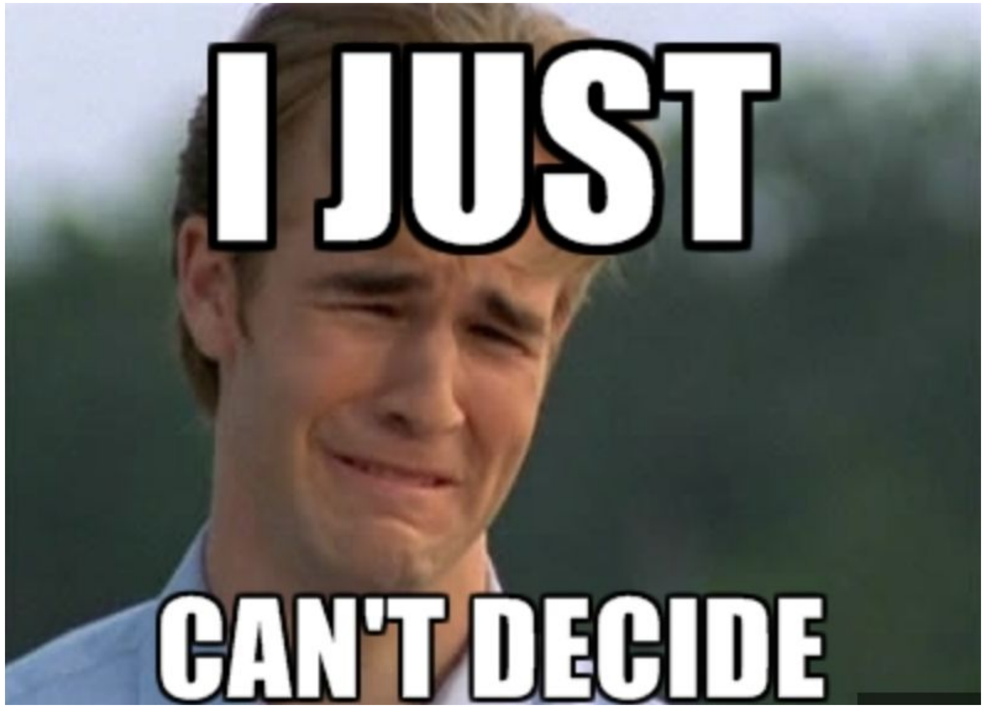
Conclusion

Keep in mind your main objective for creating or evolving a ML platform

Prioritize building for the key internal customer first

Minimize adoption friction by building appropriate interfacing tools

What got you here won’t get you there: skills and expertise that informed decisions at earlier stages, might not fit the current goals



pyladies amsterdam

```python
if questions:
    try:
        answer()
    except RuntimeError:
        pass
else:
    print("Thank You.")
```