ML Workflows and “ML Ops”

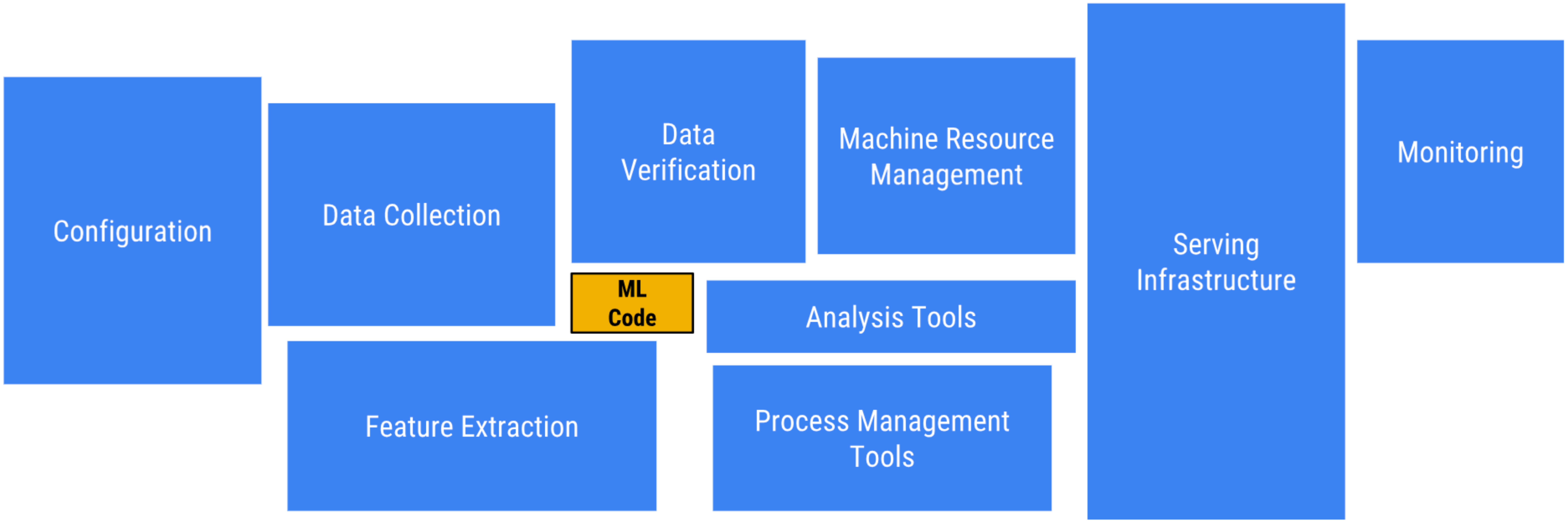
Amy Unruh

@amygdala

Launching the first proof-of-concept version of a machine learning system is pretty easy...

..but when you try to productionize and scale out, you notice all sorts of issues

- What was built as the prototype is only a very small piece of what you need to pay attention to
- Problems show up when you try to scale out, and keep a system in long-term continuous operation



From: *Hidden Technical Debt in Machine Learning Systems,* D. Sculley et al.

This is an important topic, that’s not often considered in theoretical ML courses…

Why do things become harder in a production system? (an incomplete list)

(... things we hear from customers...)

- data cleaning and processing is hard at scale
- scaling out & infrastructure issues
- training/serving skew
- unexpected interactions between components
  - data influences ML system behavior, which may erode abstraction boundaries
- data freshness requirements and model drift
- iteration, tracking/monitoring, and reproducibility requirements

(and lots more)

What are some things that can lead to trouble? (an incomplete list :)

- including proof-of-concept code in the production system
  - using notebook code directly
- building 'black box' components
- lack of data validation
- using biased or stale data
- lack of continuous monitoring

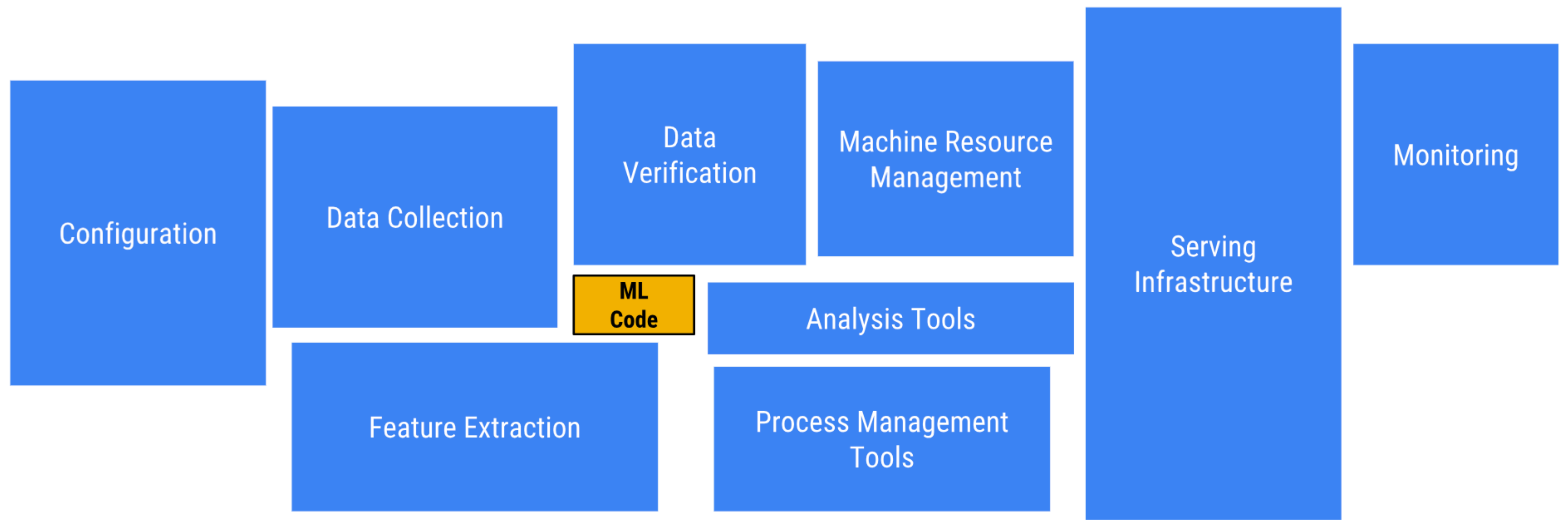
What can help?

(an incomplete list)

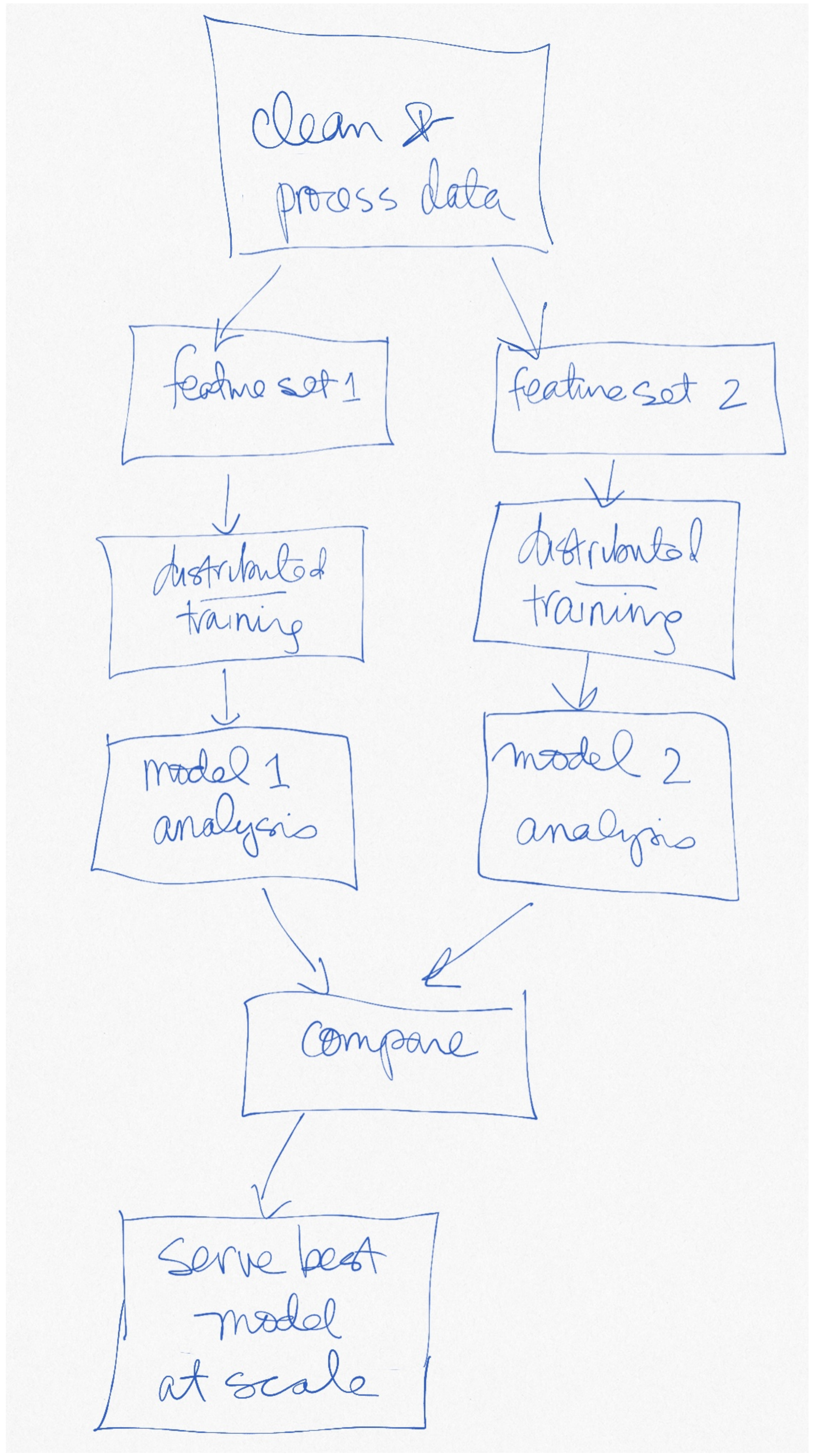
(“DevOps" —> "ML Ops”...)

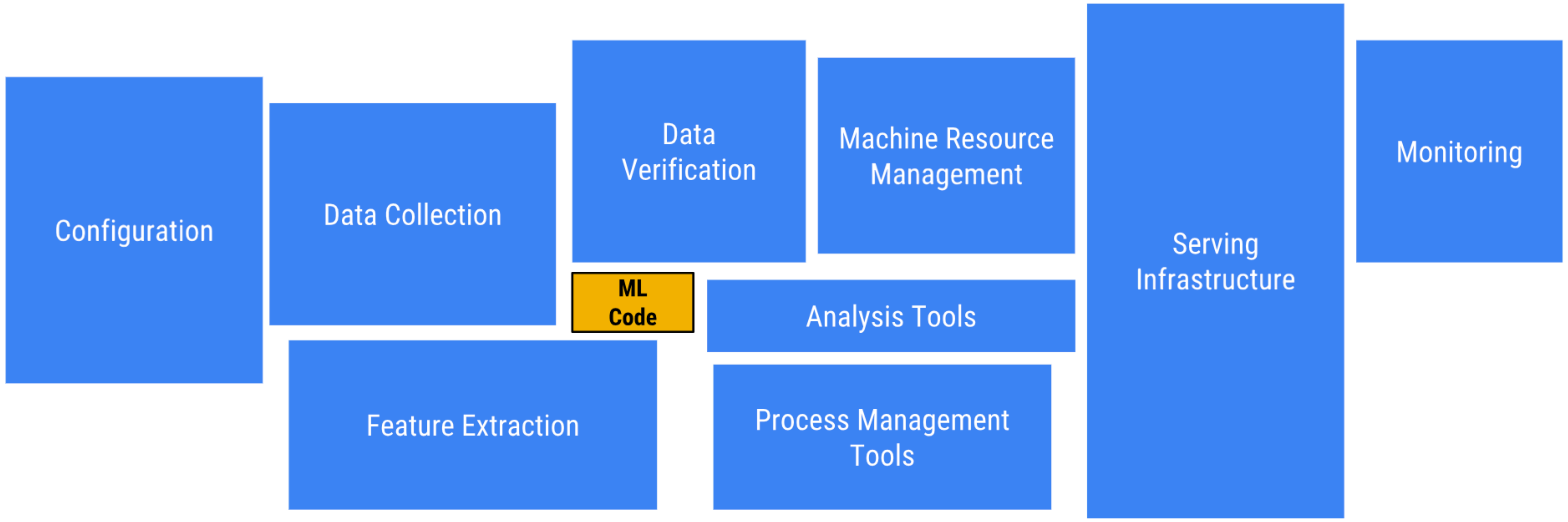
Lifecycle management: ML workflow frameworks

- reusable, composable, and scalable "building blocks"
- data and model version management
- support for controlled experimentation and model evaluation
- support for monitoring, auditing, checkpoints, and logging
- support for scheduled and triggered jobs, support for incremental learning
- support for collaboration



Lifecycle management: ML workflow frameworks





Lots of interesting work in this area right now end