Scalable ML with Kubernetes and Kubeflow

ODSC West - October 2019

About Me

John Tate

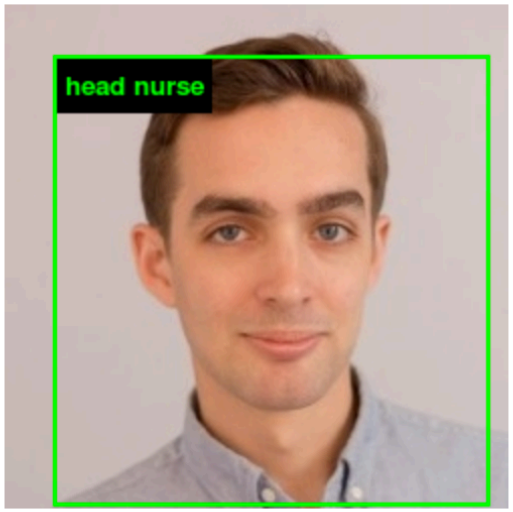
Senior Data Scientist

Metis Chicago

john.tate@thisismetis.com

@likelyjohntate

in/johnatate



3rd Annual

DEMYSTIFYING DATA SCIENCE™

A **FREE** Live Online Conference for Aspiring Data Scientists, Data-Focused Business Leaders and Practitioners

Agenda

1. Lab 1: Setup
2. Kubernetes and Kubeflow
3. Containers for Data Science
4. Lab 2: Kubeflow UI
5. Building Local
6. Lab 3: Fairing and Pipelines
7. Deploying Models with Kubeflow
8. Lab 4: SeldonCore
9. Next steps with Kubeflow

This is not a sales pitch for Kubernetes or Kubeflow

We’re here to talk about data science problems and how to solve them

GETTING WAY AHEAD OF OURSELVES:

Setting up K8s & Kubeflow

Lots of things to Install

For starters:

- github repo
- kubectl
- kfctl
- minikube

Our biggest challenge of the day:

Our biggest challenge of the day:

Conference WiFi

(or lack there of)

This is not lightweight stuff

- Kubernetes: ~5GB uncompressed
- Kubeflow: ~15GB uncompressed

The Normal Minikube Process

1. Minikube grabs an ISO + pulls k8s containers from a docker registry
2. We tell k8s what resources make up Kubeflow and how to set the up
3. k8s pulls those images from various docker registries

What we'll try today

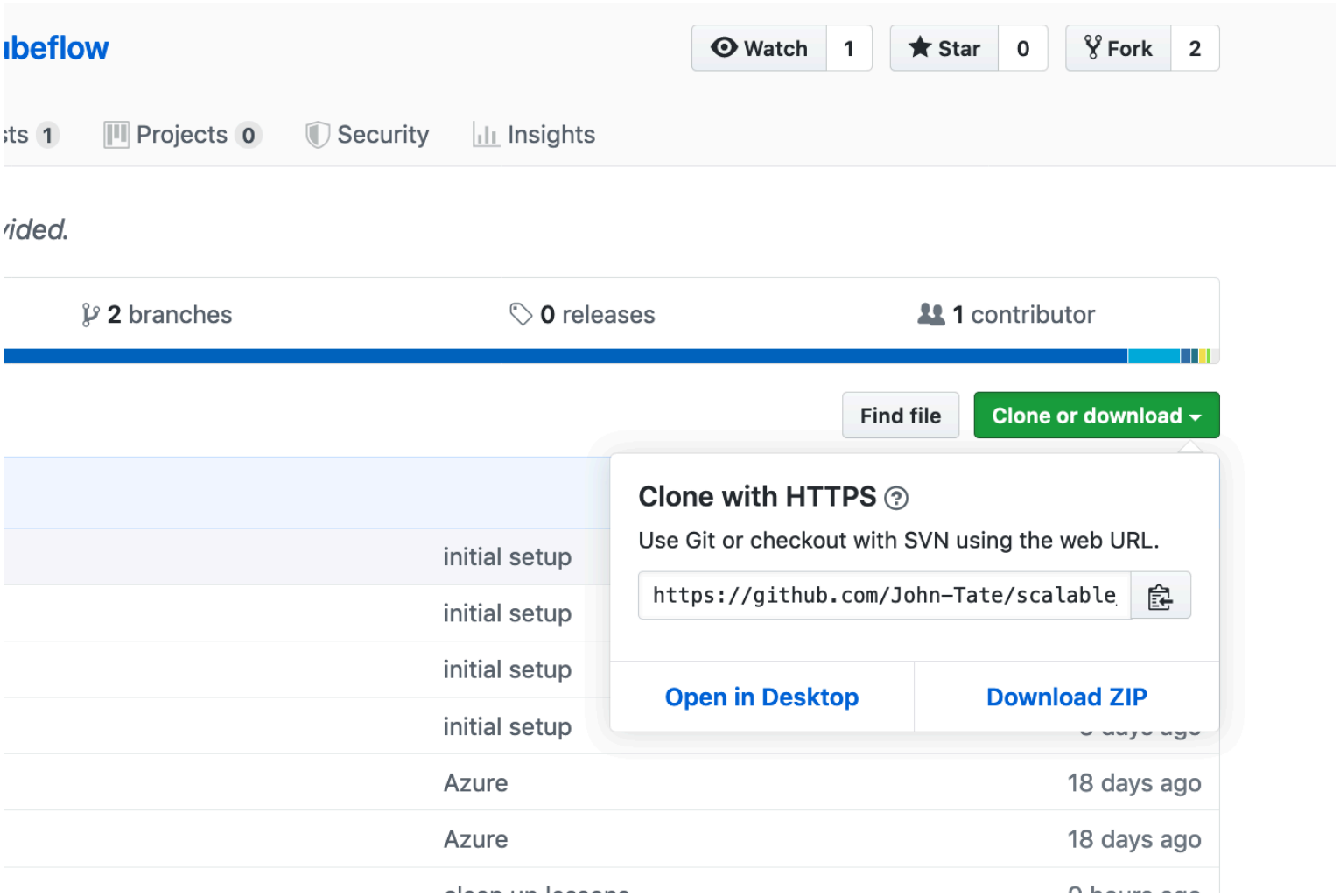
Strangers with ~~candy~~ flash drives

- Use Cached Images
- Let Minikube push them into the VM's local docker env
- Load from there

Grab the Git Repo

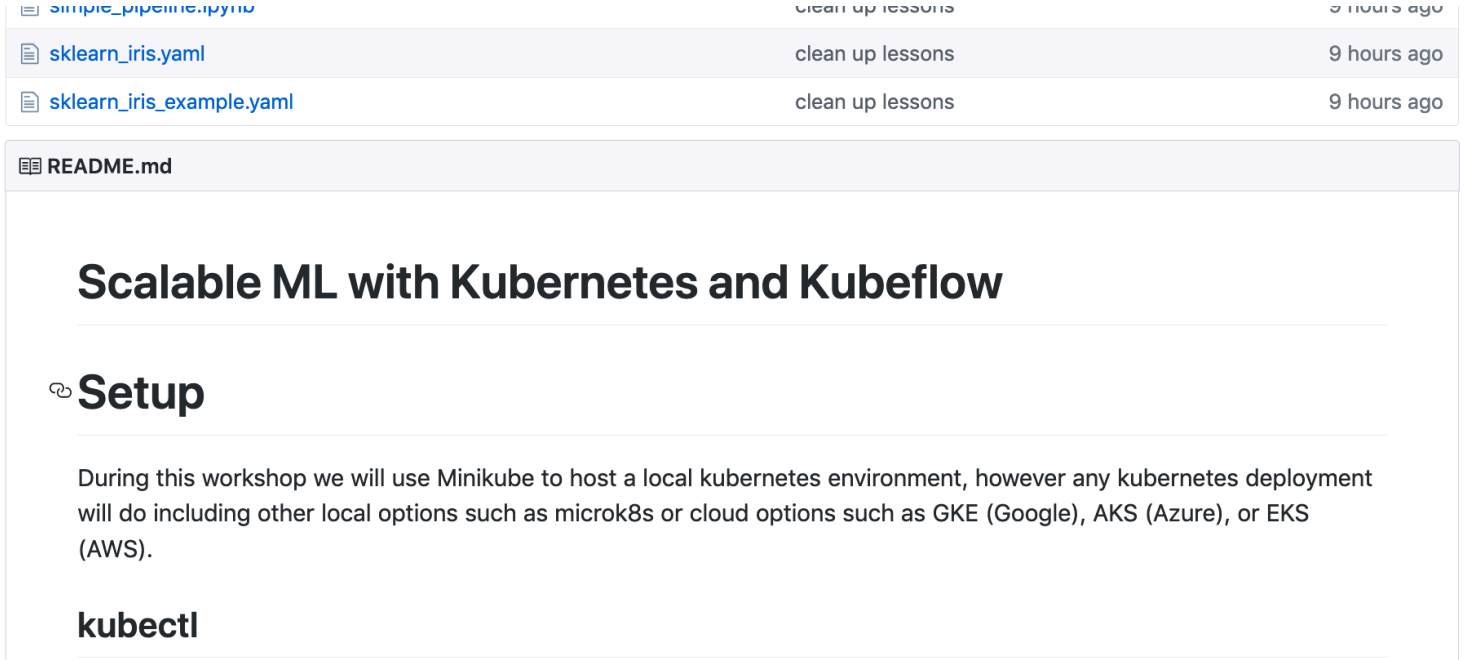
https://github.com/John-Tate/scalable_ml_with_kubeflow

http://odsc.johnatate.com


ıbeflow
Watch 1
Star 0
Fork 2
ts 1
Projects 0
Security
Insights
ıided.
2 branches
0 releases
1 contributor
Find file
Clone or download
Clone with HTTPS
Use Git or checkout with SVN using the web URL.
https://github.com/John-Tate/scalable
Open in Desktop
Download ZIP
initial setup
initial setup
initial setup
initial setup
Azure
18 days ago
Azure
18 days ago

```
$ cd ~
$ git clone https://github....
$ cd scalable_ml_with_kubeflow
```

All setup instructions / commands are in the README.md file in the github repo



simple_pipeline.ipynb	clean up lessons	9 hours ago
sklearn_iris.yaml	clean up lessons	9 hours ago
sklearn_iris_example.yaml	clean up lessons	9 hours ago

README.md

Scalable ML with Kubernetes and Kubeflow

Setup

During this workshop we will use Minikube to host a local kubernetes environment, however any kubernetes deployment will do including other local options such as microk8s or cloud options such as GKE (Google), AKS (Azure), or EKS (AWS).

kubectl

kubectl

- The CLI for Kubernetes
- Can interact with our k8s cluster no matter where it is:
  - Locally
  - In a VM
  - On a public cloud

kubectl

linux

```
$ gunzip ./utils/linux/kubectl.gz
$ chmod +x ./utils/linux/kubectl
$ sudo mv ./utils/linux/kubectl
/usr/local/bin/kubectl
$ kubectl version
```

macOS

```
$ gunzip ./utils/mac/kubectl.gz
$ chmod +x ./utils/mac/kubectl
$ sudo mv ./utils/mac/kubectl
/usr/local/bin/kubectl
$ kubectl version
```

kfctl

- A simple API for managing Kubeflow deployments
- Not many features
- More of a wrapper for kubectl

kfctl

linux

```
$ cd ./utils/linux/
$ tar -xvf kfctl_v0.6.2_linux.tar.gz
$ sudo mv kfctl /usr/local/bin/kfctl
```

macOS

```
$ cd ./utils/mac/
$ tar -xvf kfctl_v0.6.2_darwin.tar.gz
$ sudo mv kfctl /usr/local/bin/kfctl
```

Minikube

- Single node local k8s deployment
- Runs in a VM
- Has many utilities that make working with a local cluster & testing easier

Let’s look at the readme

Install Linux (Ubuntu)

1. Verify the output of this command is non-empty to check if your CPU supports virtualization:

```
grep -E --color 'vmx|svm' /proc/cpuinfo
```

2. Ensure you have a hypervisor installed. VirtualBox is recomended

3. Install Minikube

local file:

```
gunzip ./utils/linux/minikube.gz
chmod +x ./utils/linux/minikube
sudo install ./utils/linux/minikube /usr/local/bin
```

web:

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 \
  && chmod +x minikube

sudo install minikube /usr/local/bin/
```

Install macOS

1. Verify the output of this command is non-empty to check if your CPU supports virtualization:

```
sysctl -a | grep -E --color 'machdep.cpu.features|VMX'
```

2. Ensure you have a hypervisor installed. VirtualBox is recomended

3. Install Minikube

Homebrew:

```
brew cask install minikube
```

Curl:

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-darwin-amd64 \
  && chmod +x minikube

sudo mv minikube /usr/local/bin
```

Local File

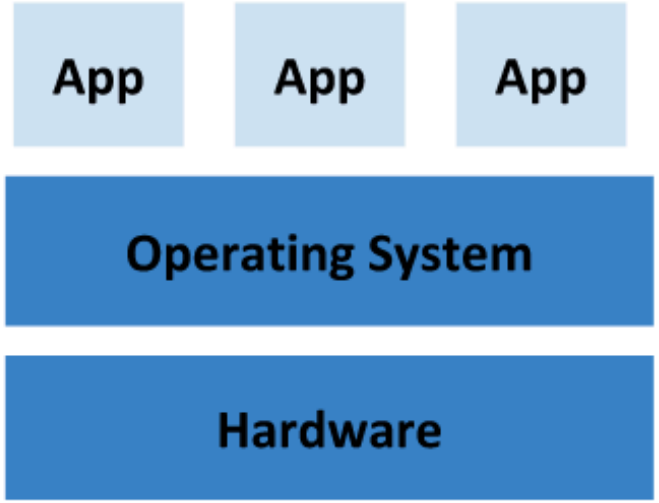
```
gunzip ./utils/mac/minikube.zip
chmod +x ./utils/mac/minikube
sudo install ./utils/mac/minikube /usr/local/bin
```

Start Minikube

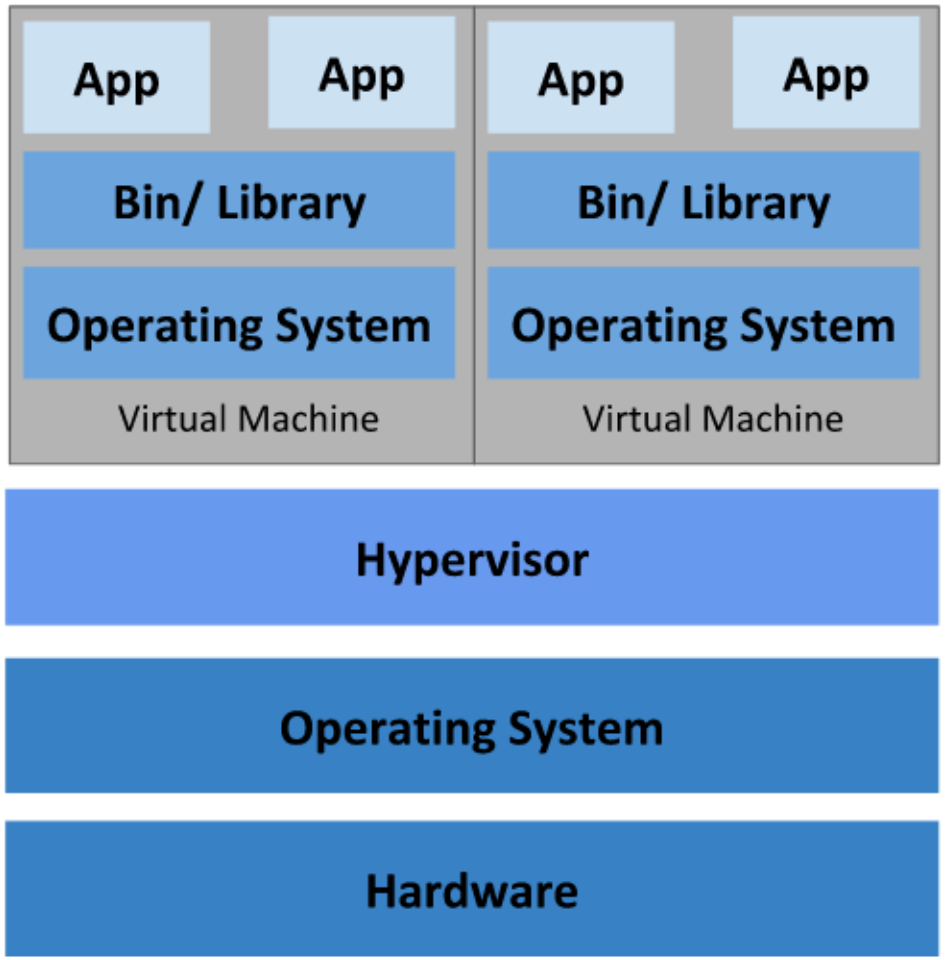
```
$ minikube start
--cpus 4
--memory 8096
--disk-size=40g
--kubernetes-version v1.14.0
--insecure-registry "10.0.0.0/24"
```

Let's talk about **Containers**

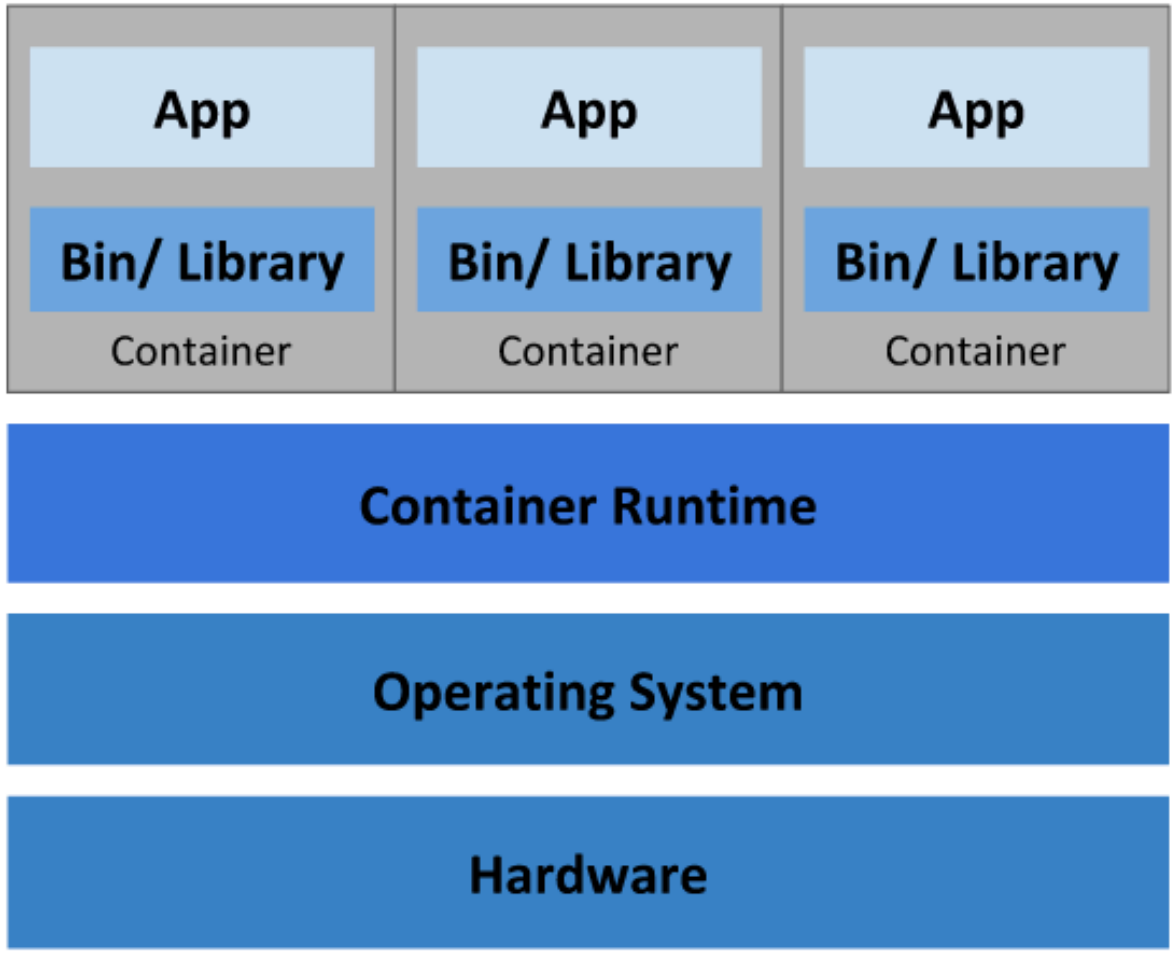
At least one image of shipping or logistics is required by law in any presentation about containers



Traditional Deployment



Virtualized Deployment



Container Deployment

Containers are not new

- FreeBSD *jail*
- Solaris Containers
- Linux Containers (LXC)
- Google LMCTFY

Containers are not enough

Non-trivial problems, including:

- Connecting containers to each other
- Managing what containers are active
- Managing who can access what resources
- Scaling resources

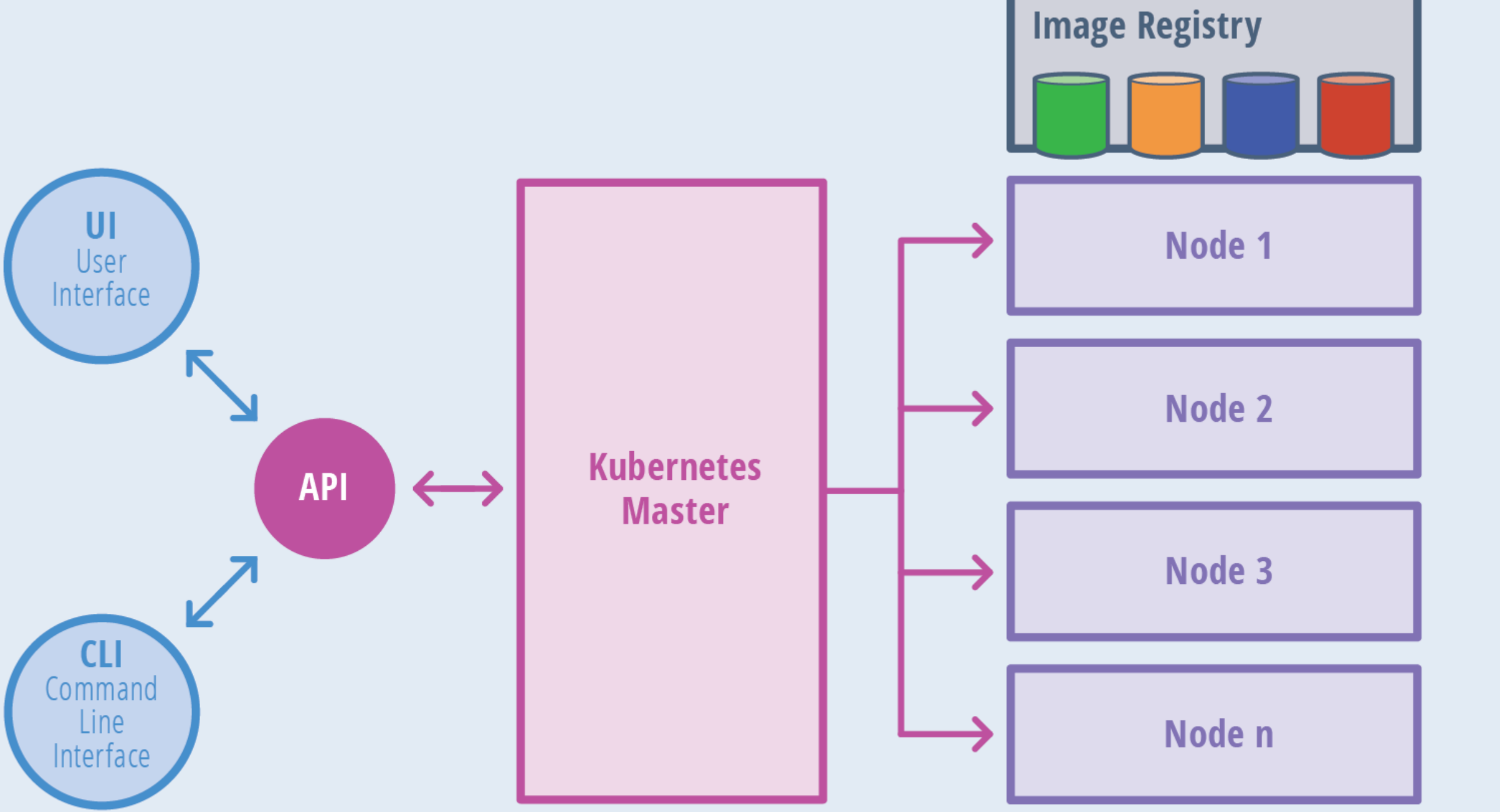
kubernetes

kubernetes

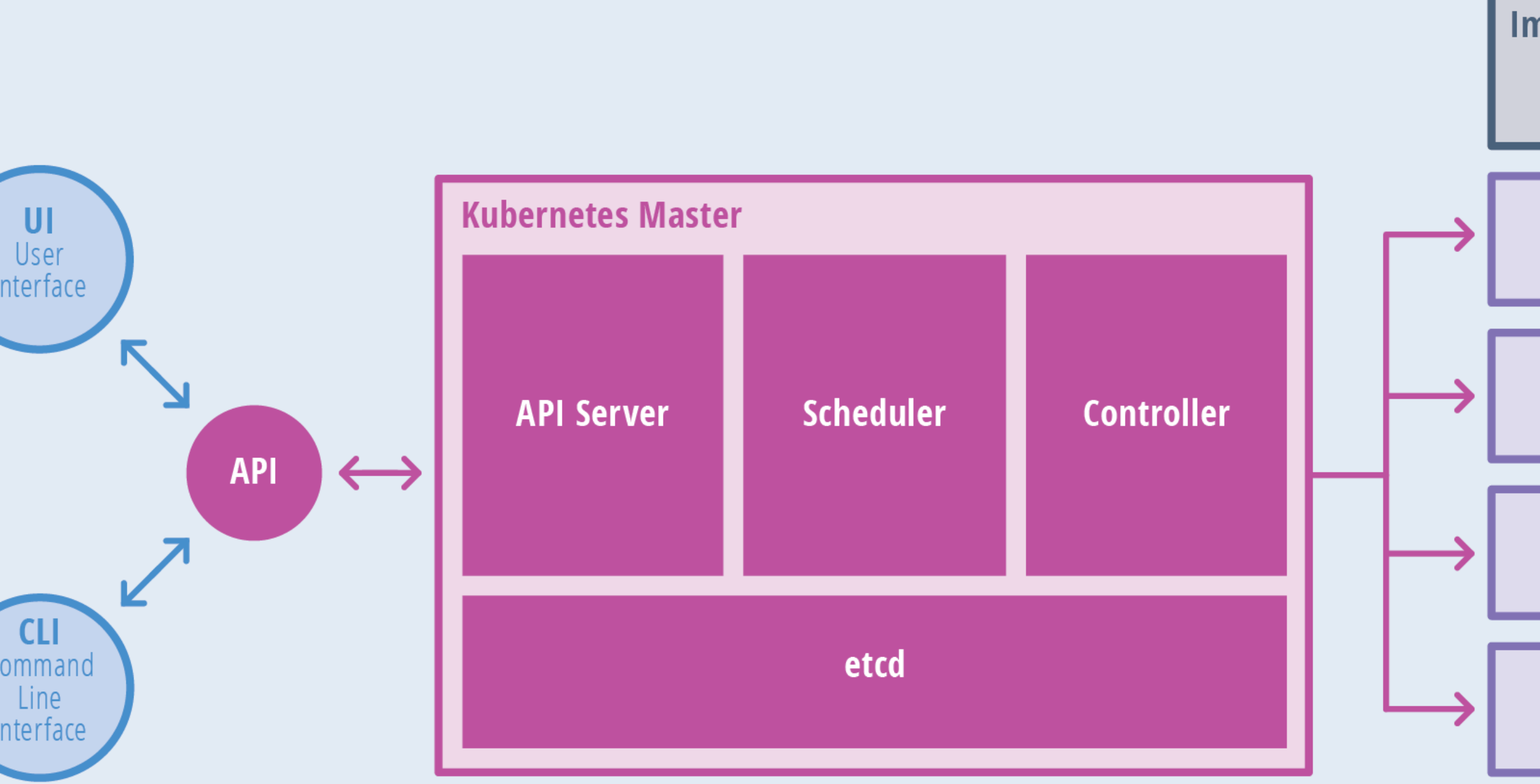
… for Data Scientists

Kubernetes Features

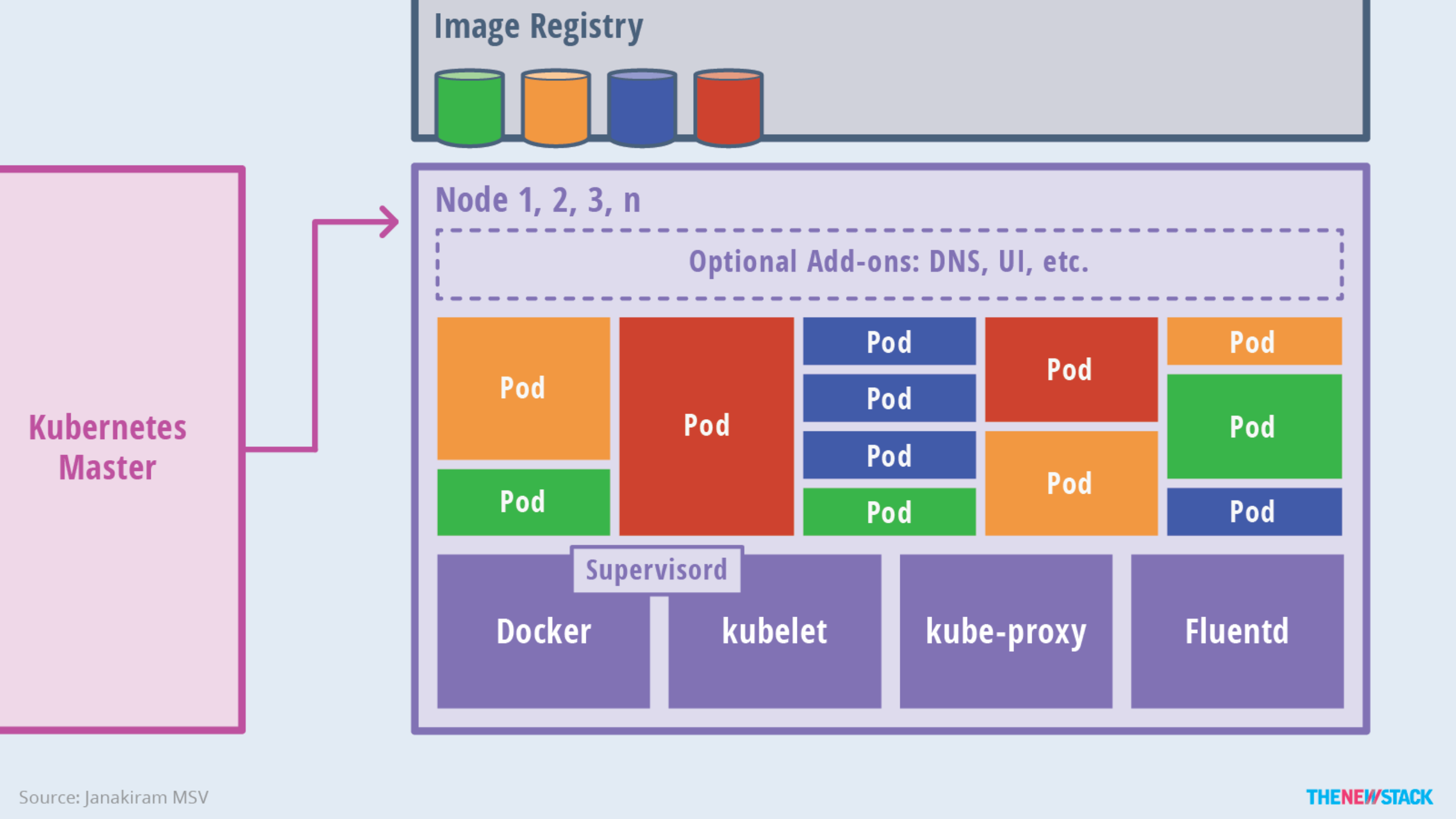
- Load balancing
- Service Discovery
- Ingress and Endpoints
- Orchestration
- Resource Management, and Utilization
- Service Health
- Credentials, Integration, and Security
- Horizontal Scaling



Source: Janakiram MSV



Source: Janakiram MSV



Source: Janakiram MSV

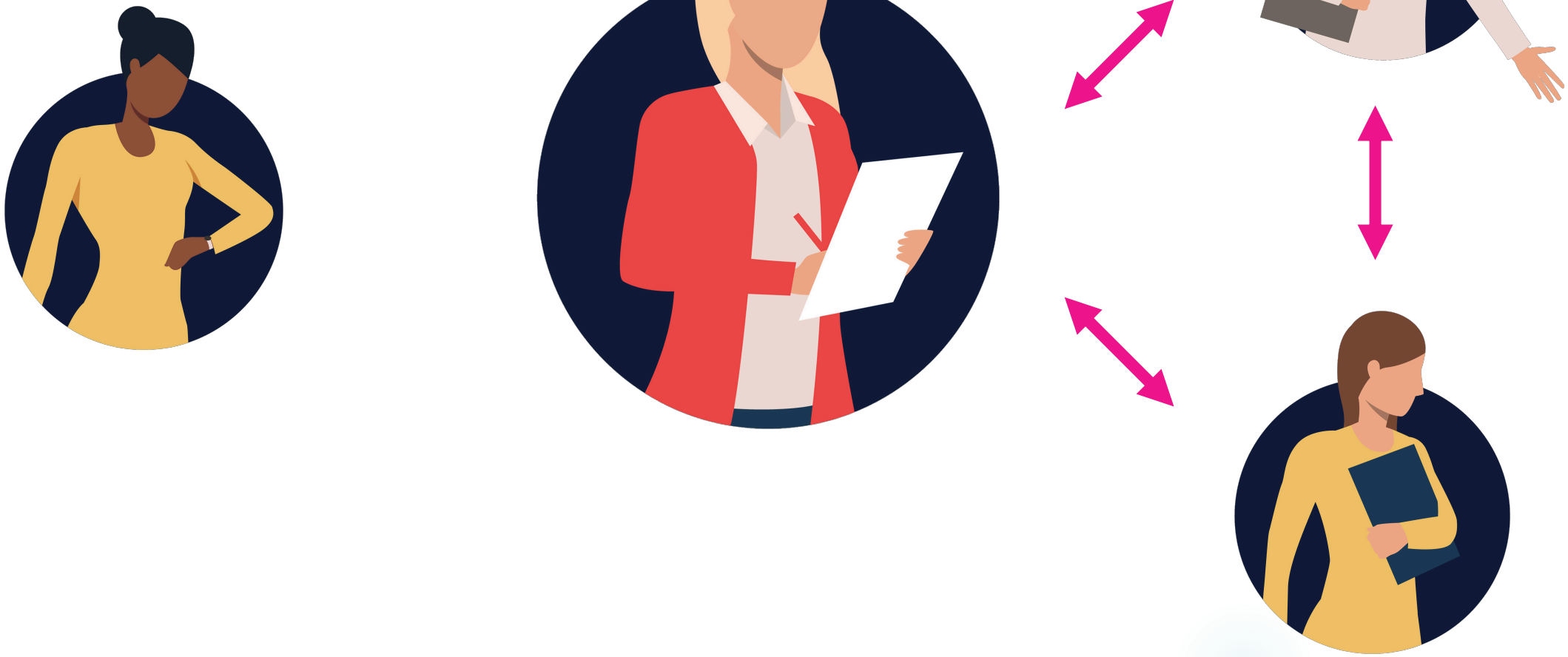
K8s Core Concepts

- Cluster – Master & Nodes
- Registry – Where Images & Containers are stored
- Nodes – Contain Pods
- Pods – Made up of Containers

That's great, but how does this help my data science team?

Machine Learning Isn't Our Biggest Problem


R
K
jupyter



Alice has problems

Sound Familiar?

- Building models is only one part of the puzzle
- Large data and models require *infrastructure*
- Collaboration is a challenge
- Managing Credentials, Integrations, and Environments is a job in and of itself
- Production systems and infrastructure can become a limiting factor in the model development story

Concept:

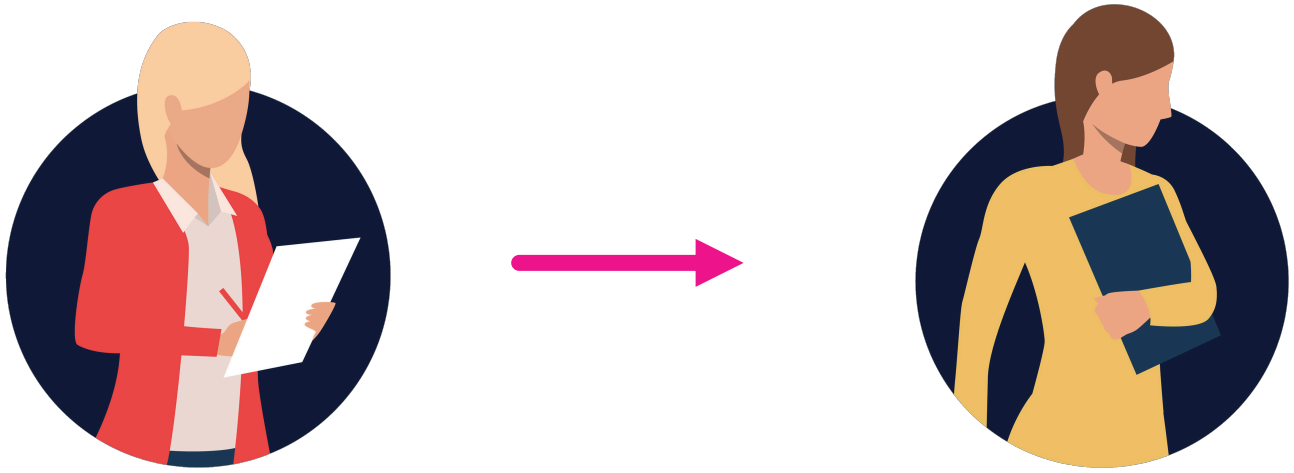
Deployments and infrastructure management should work with your preferred tools with minimal adjustment

Another Familiar Story

Another Familiar Story

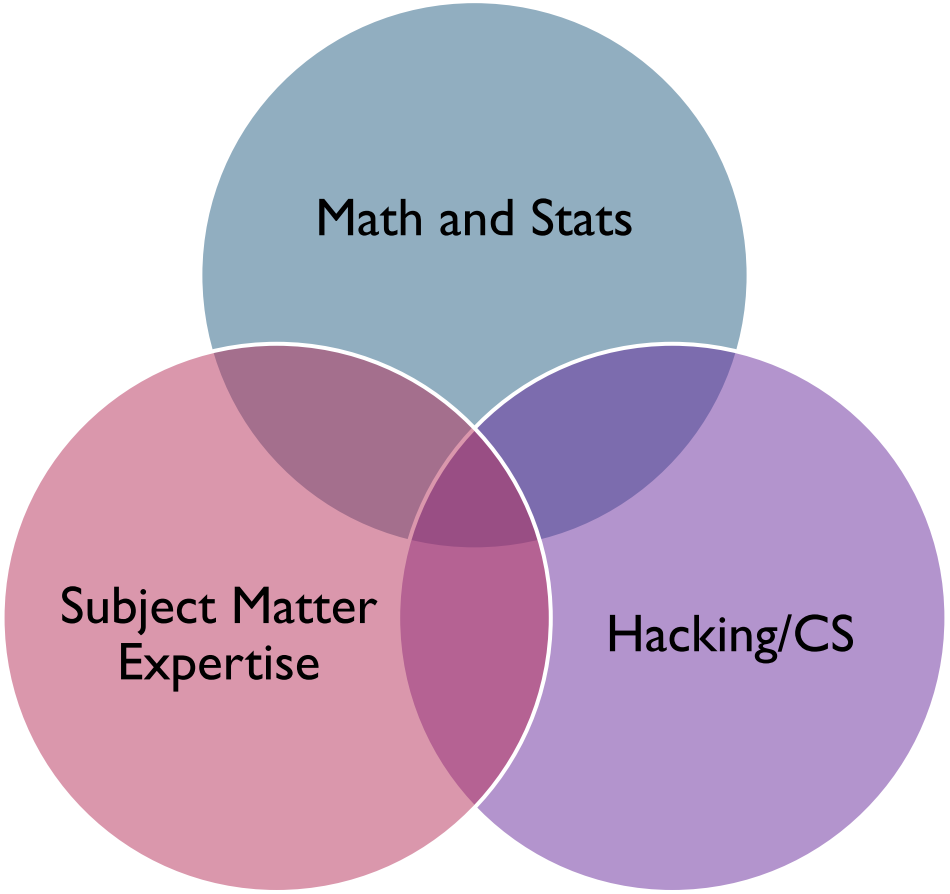
- Designs an Experiment
- Acquires Data
- Performs EDA
- Trains and Tunes a model

Another Familiar Story

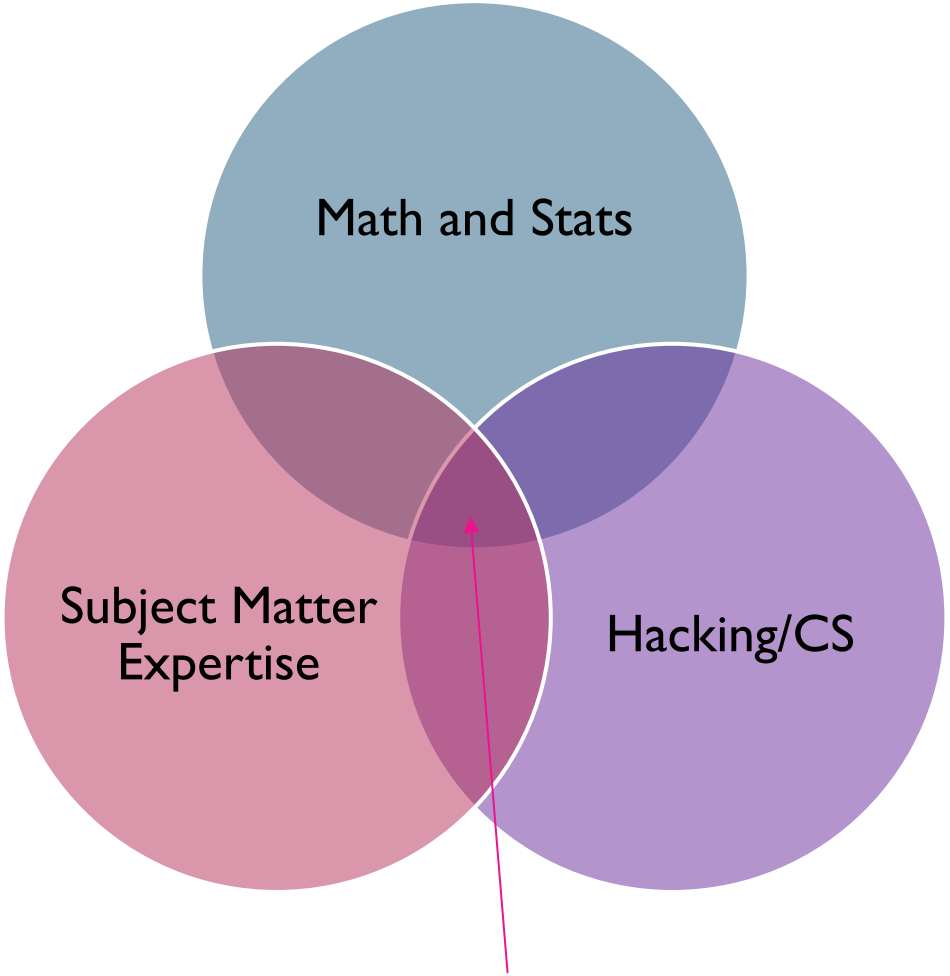


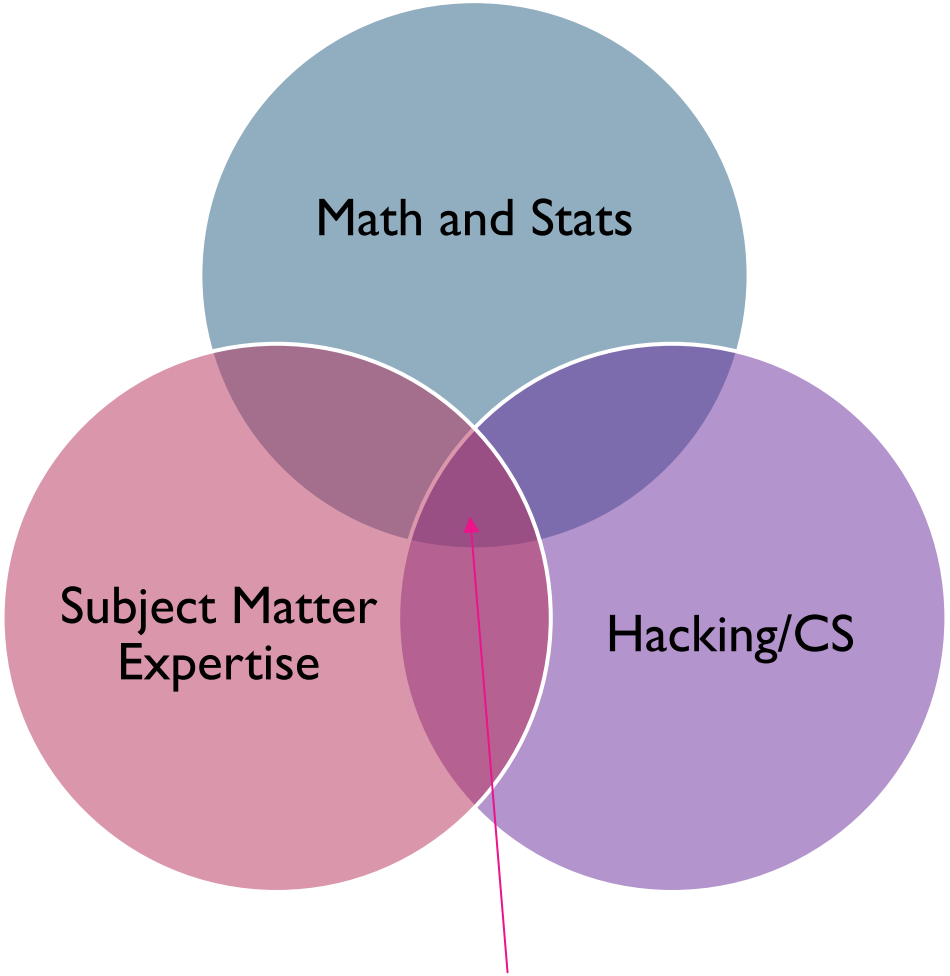
- Takes Model and Cleaning/FE code
- Re-engineers to work in serving system or production app

Another Familiar Story

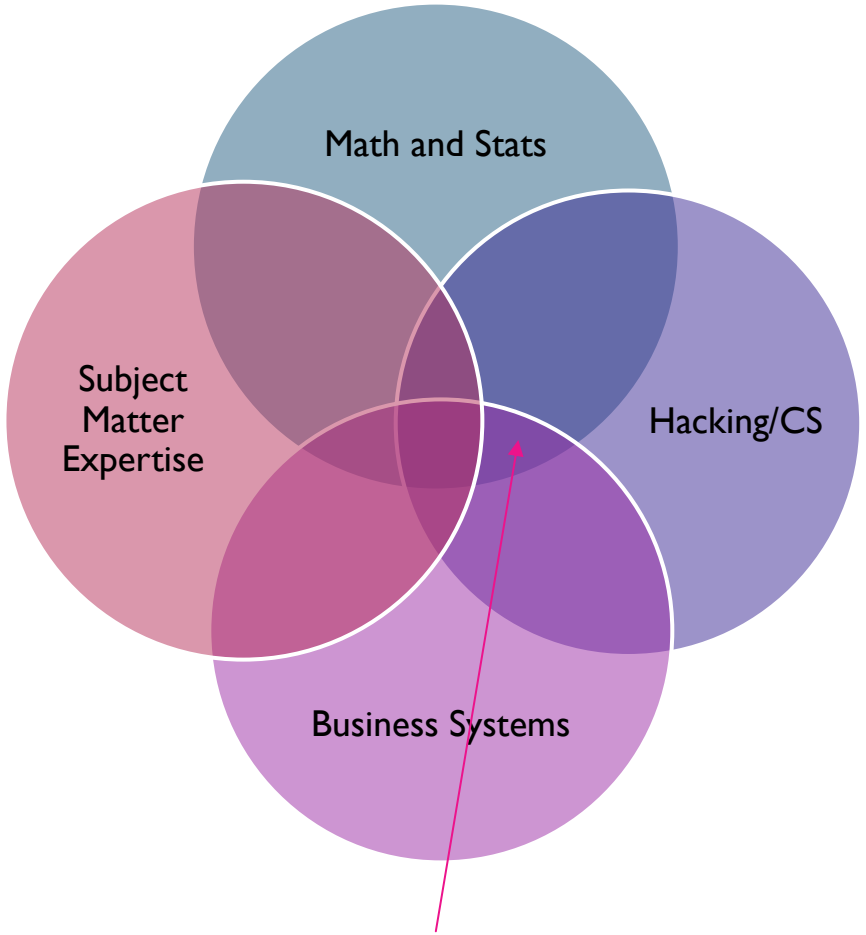


Data Scientist

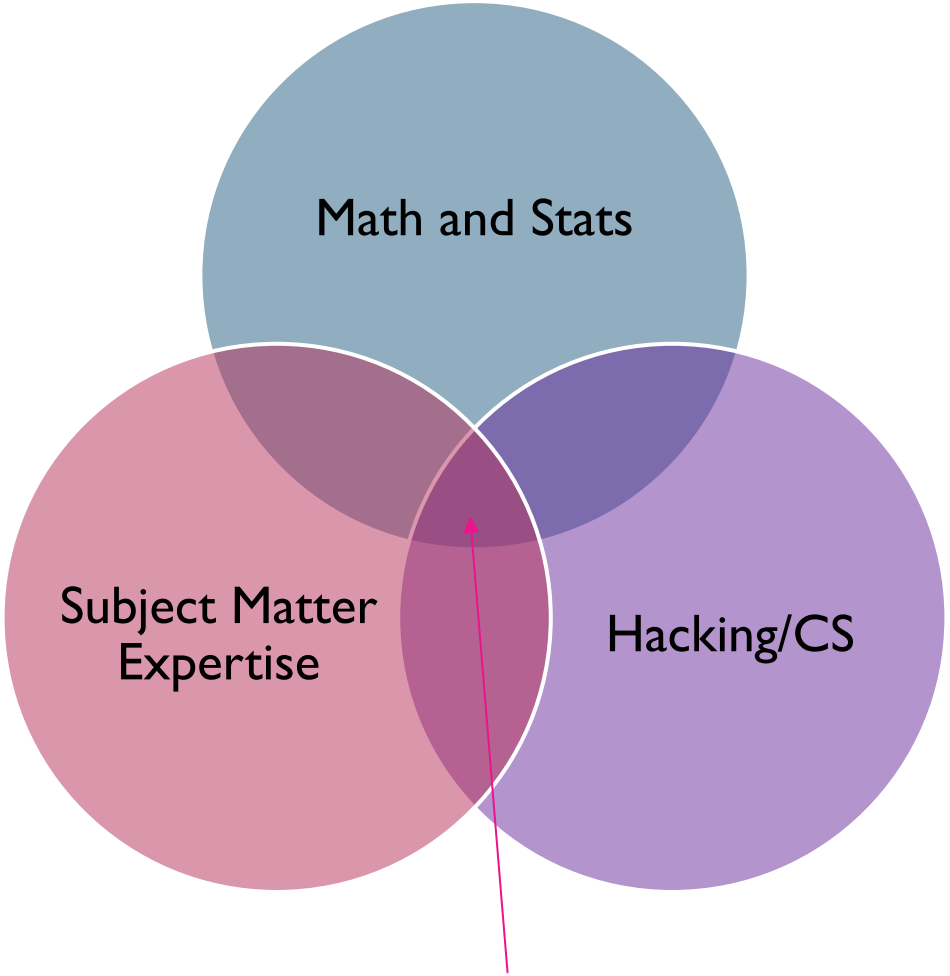

Math and Stats
Subject Matter Expertise
Hacking/CS
Data Scientist



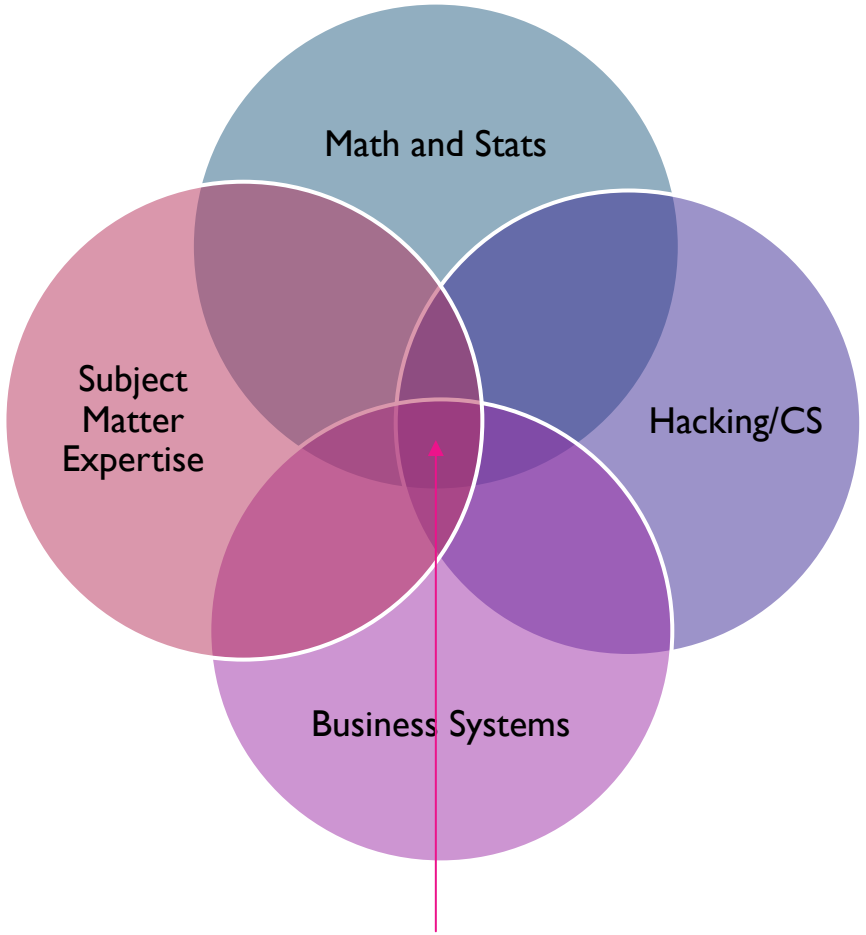
Data Scientist



Machine Learning Engineer?



Data Scientist



Unicorn?

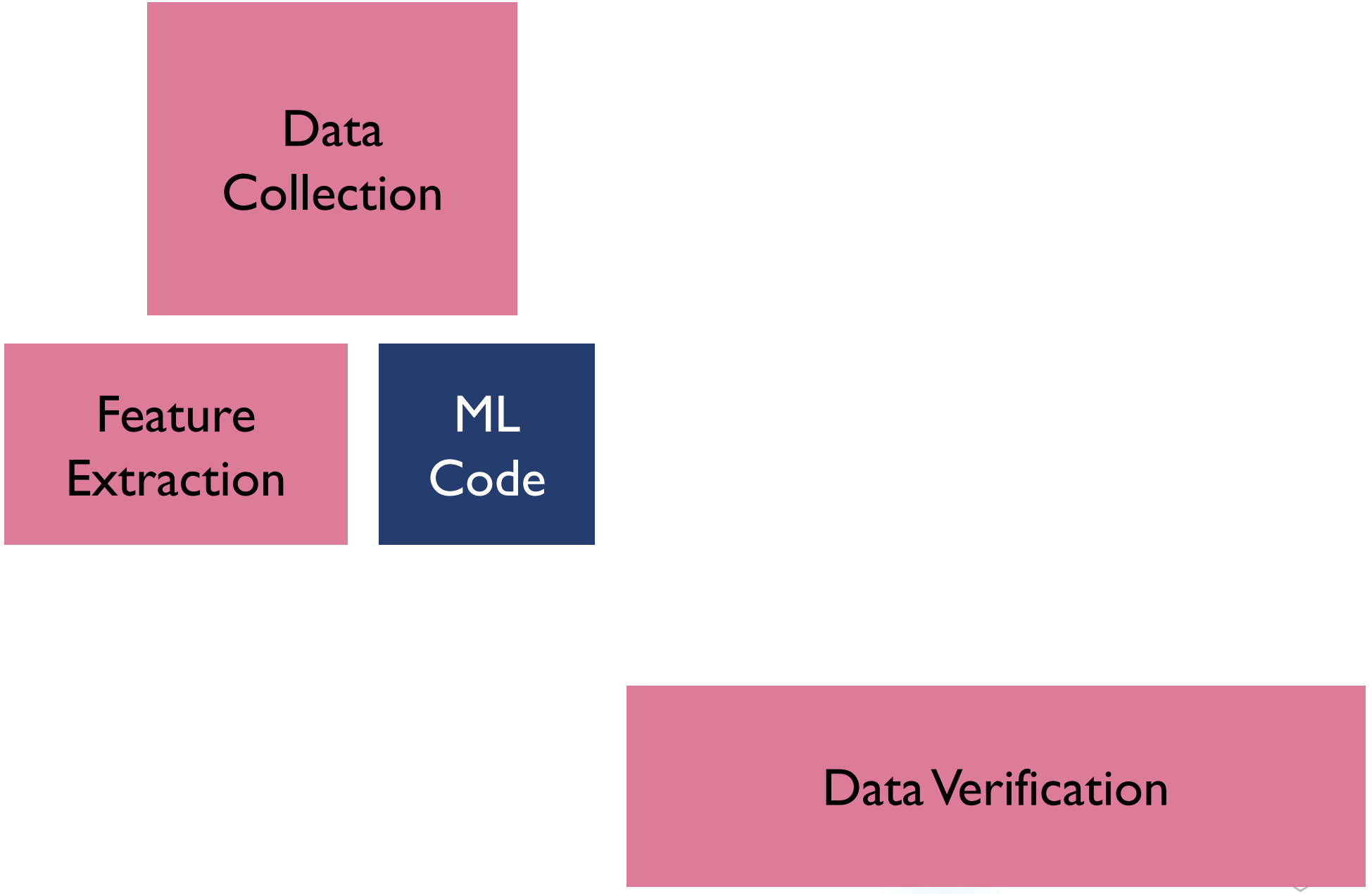
A Better Story

The Big Idea

Let data scientists build and deploy ML without additional developer resources

First, lets talk about your ML code

ML
Code


Data
Collection
Feature
Extraction
ML
Code
Data Verification

Monitoring

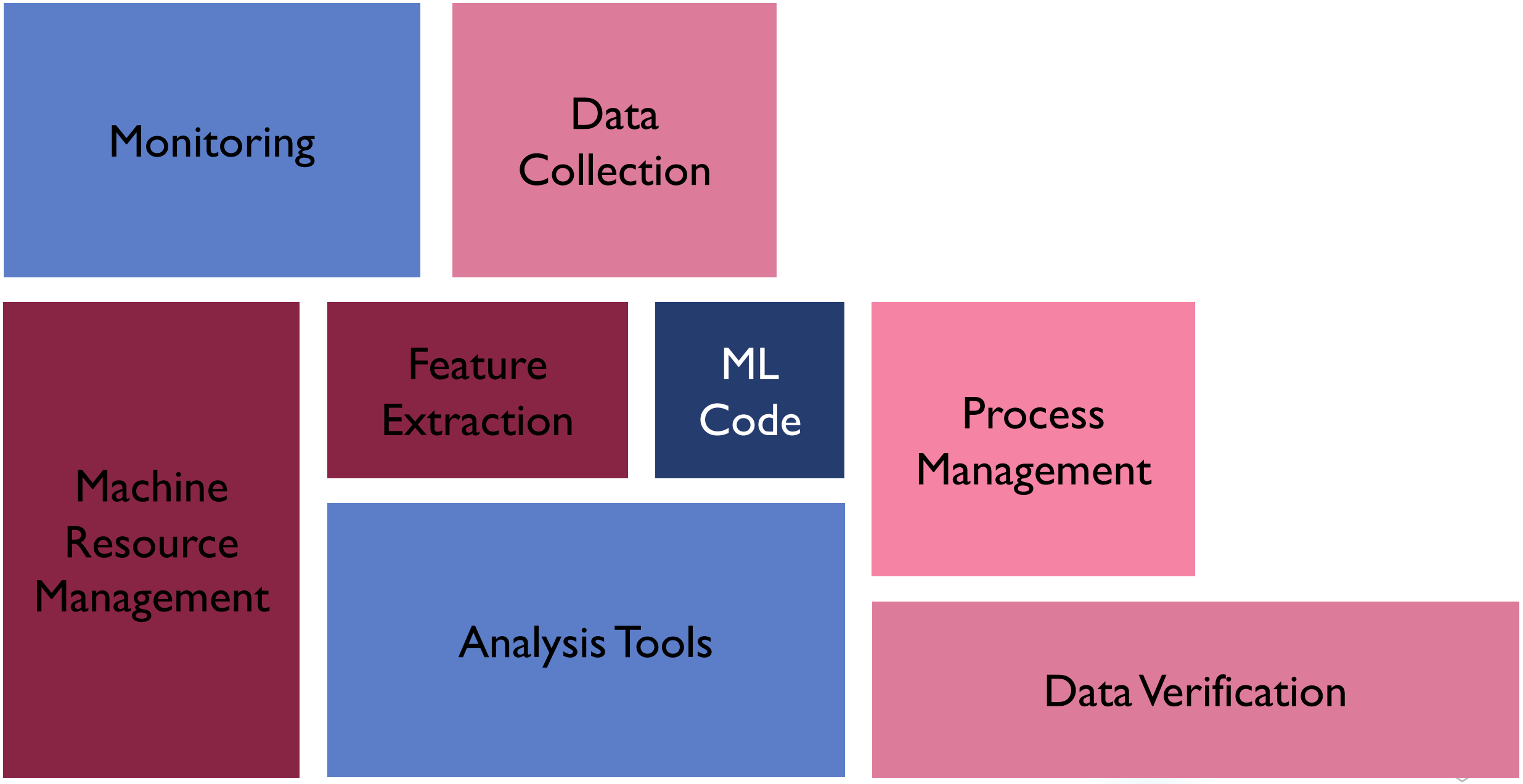
Data Collection

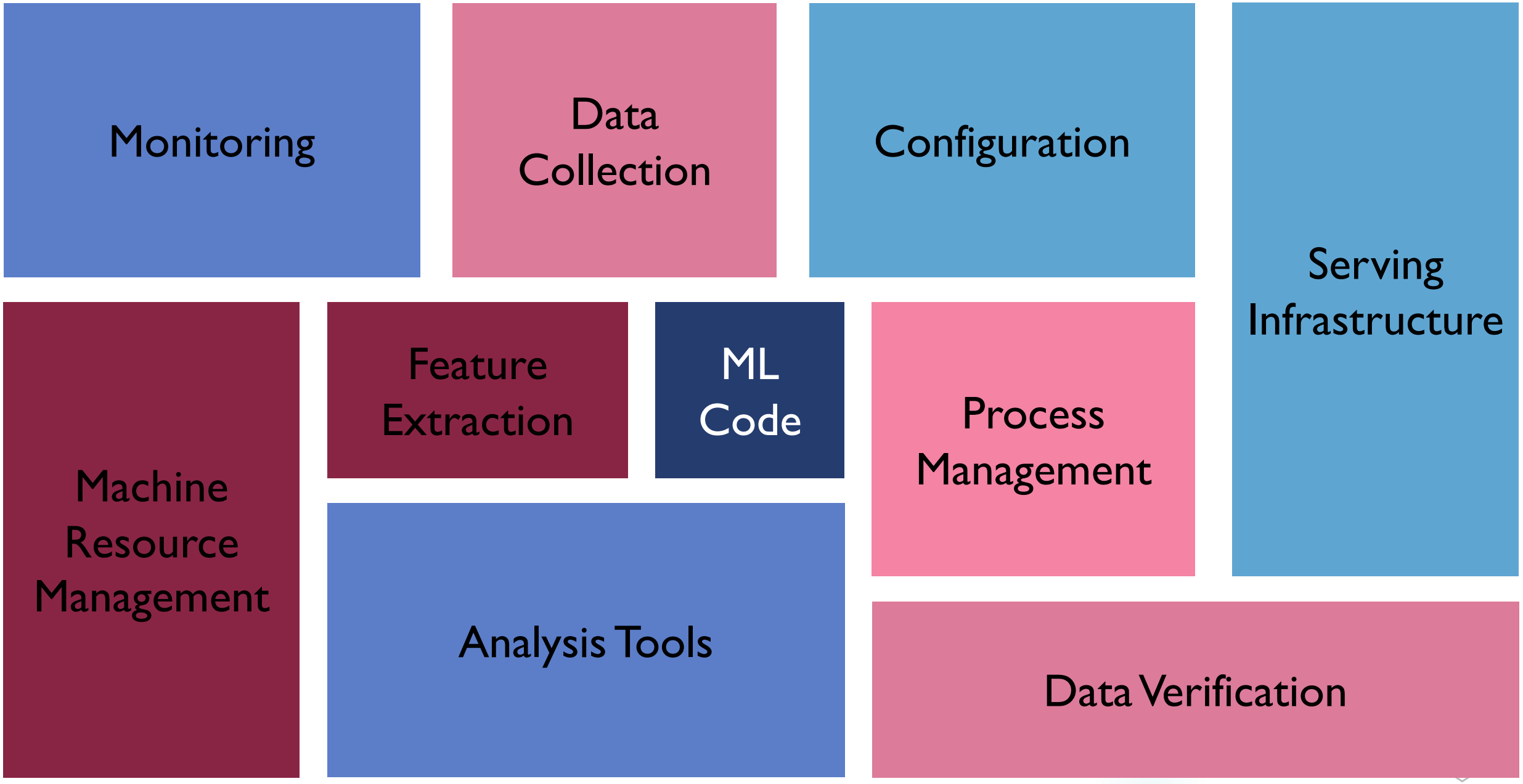
Feature Extraction

ML Code

Analysis Tools

Data Verification


Monitoring
Data Collection
Machine Resource Management
Feature Extraction
ML Code
Process Management
Analysis Tools
Data Verification


Monitoring
Data Collection
Configuration
Serving Infrastructure
Machine Resource Management
Feature Extraction
ML Code
Process Management
Analysis Tools
Data Verification

That’s a lot

Who's supporting all of this?

Who's supporting all of this?

If your answer is "the data scientists" we need to talk after class

Containers for Data Science

- Each project can have its own environment, dependencies, etc
- Enhanced portability
- Works the same* no matter where its run

Concept

Every difference between development, test, and production is a potential service outage waiting to happen

Kubeflow

Kubeflow is:

- Open Source
- A collection of tools and services

Kubeflow is not:

- A be-all-end all solution
- A replacement or recreation of other services

Kubeflow lets K8s do what its already great at:

- Easy, repeatable, portable deployments on a diverse infrastructure (laptop <-> ML rig <-> training cluster <-> production cluster)
- Deploying and managing loosely-coupled microservices
- Scaling based on demand

Setting up K8s and Kubeflow

1. Provision Kubernetes Cluster
2. Register Kubernetes Cluster with Kubectl
3. Create Kubeflow Configuration
4. Deploy Kubeflow to Kubernetes
    a. If needed, configure service accounts with credentials for private container registries, cloud services, data sources, etc

Kubeflow Componenets

Training

Serving

Notebooks

Pipelines

Model Tuning

…and more

Training

- Tensorflow fully supported through TFJob (including multi-GPU)
- PyTorch, MXNet, and Chainer supported as well
- Additional training options available through Fairing

Notebooks

- Deploy Notebooks from shared base containers
- Access shared data volumes and integrated systems
- Manage K8s resources and deploy models directly from KF Notebooks

LAB:
Accessing the Kubeflow UI

Kubeflow-Fairing

- Open Source
- Facilitates working with Kubernetes models from Jupyter notebooks and python scripts
- Quickly create docker containers from notebook environment
- Create training jobs, deploy models as endpoints
- Authenticate with GCP/AWS/Azure

```
!pip install kubeflow-fairing
```

```
from kubeflow import fairing
```

Pipelines

Developing complex workflows for k8s can be difficult

- Packaging code to containers
- Managing data flows
- Managing Outputs
- Managing Credentials

… hope you like YAML

Pipelines

What if you could define your entire container based pipeline directly from Python?

What if you could quickly wrap your Python functions to create lightweight, reusable components?

LAB:
Fairing & Pipelines with Jupyter

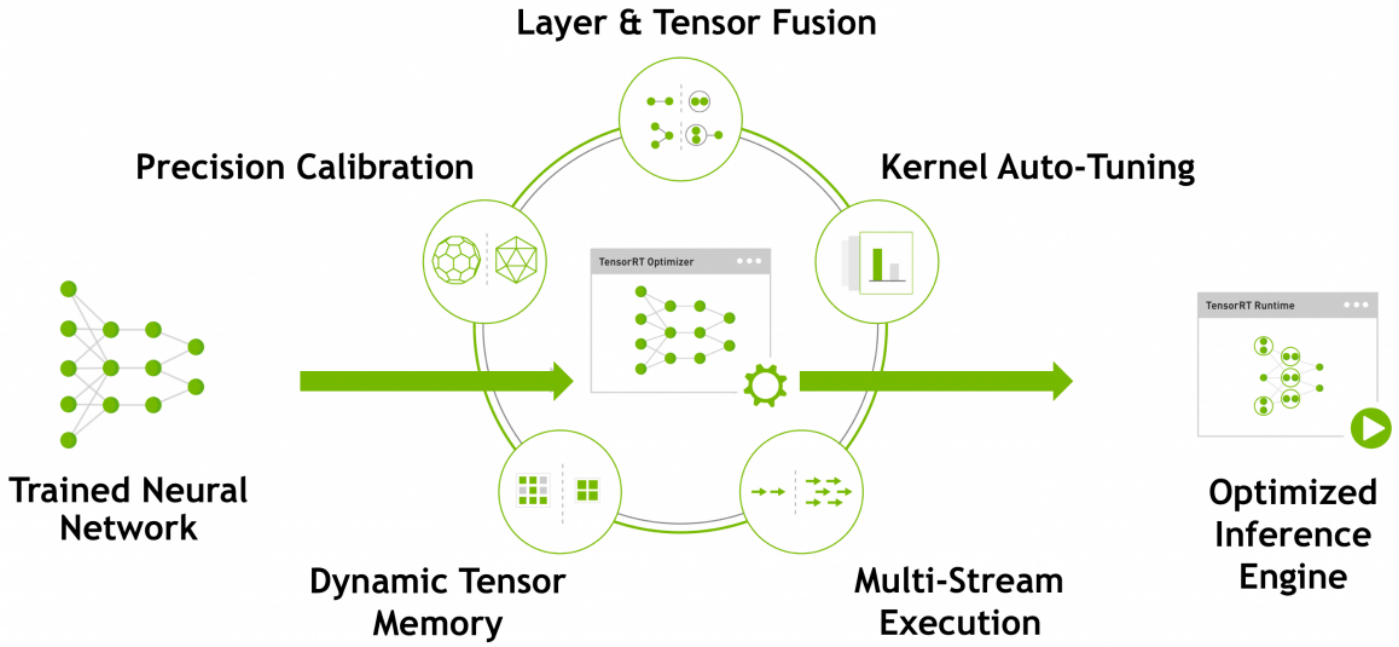
Serving Models

Serving

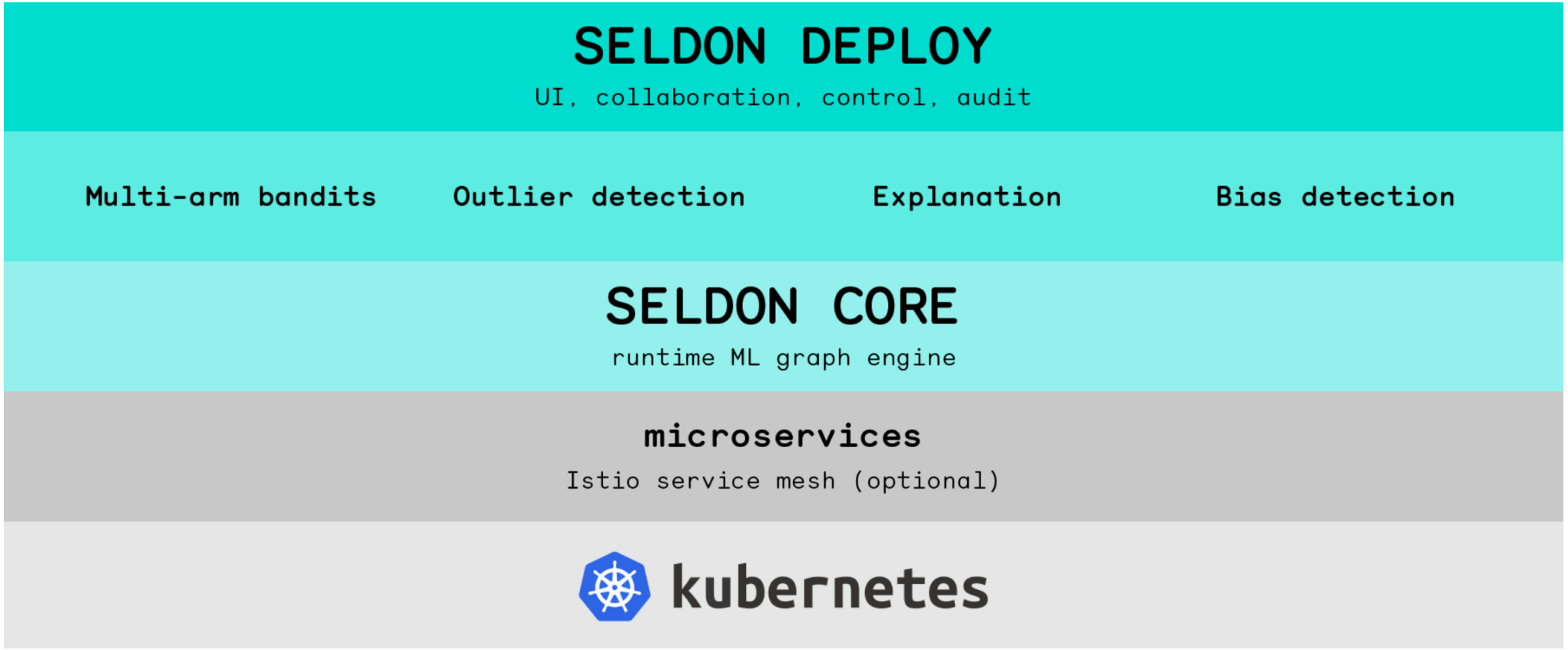
- Almost too many options:
  - TF Serving
  - NVIDIA TensorRT
  - SeldonCore
  - KFServing (new!)

NVIDIA TensorRT

- Speed up serving TF and Caffe models on GPUs (claim: up to 40x)



Seldon


SELDON DEPLOY
UI, collaboration, control, audit
Multi-arm bandits
Outlier detection
Explanation
Bias detection
SELDON CORE
runtime ML graph engine
microservices
Istio service mesh (optional)
kubernetes

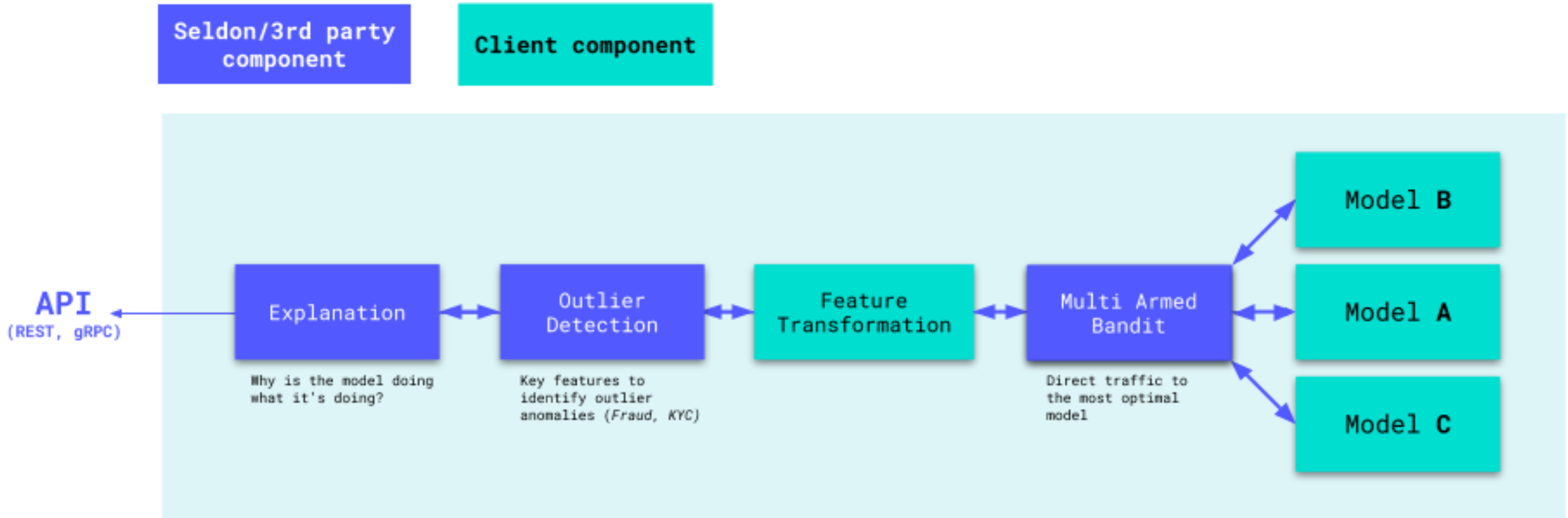
Seldon Core

- Plug and play model servers:
  - TensorFlow
  - Sklearn
  - XGBoost
  - MLFlow

Seldon Core

- Custom Model Wrappers:
  - Python
  - Java
  - R
  - NodeJS
  - …more

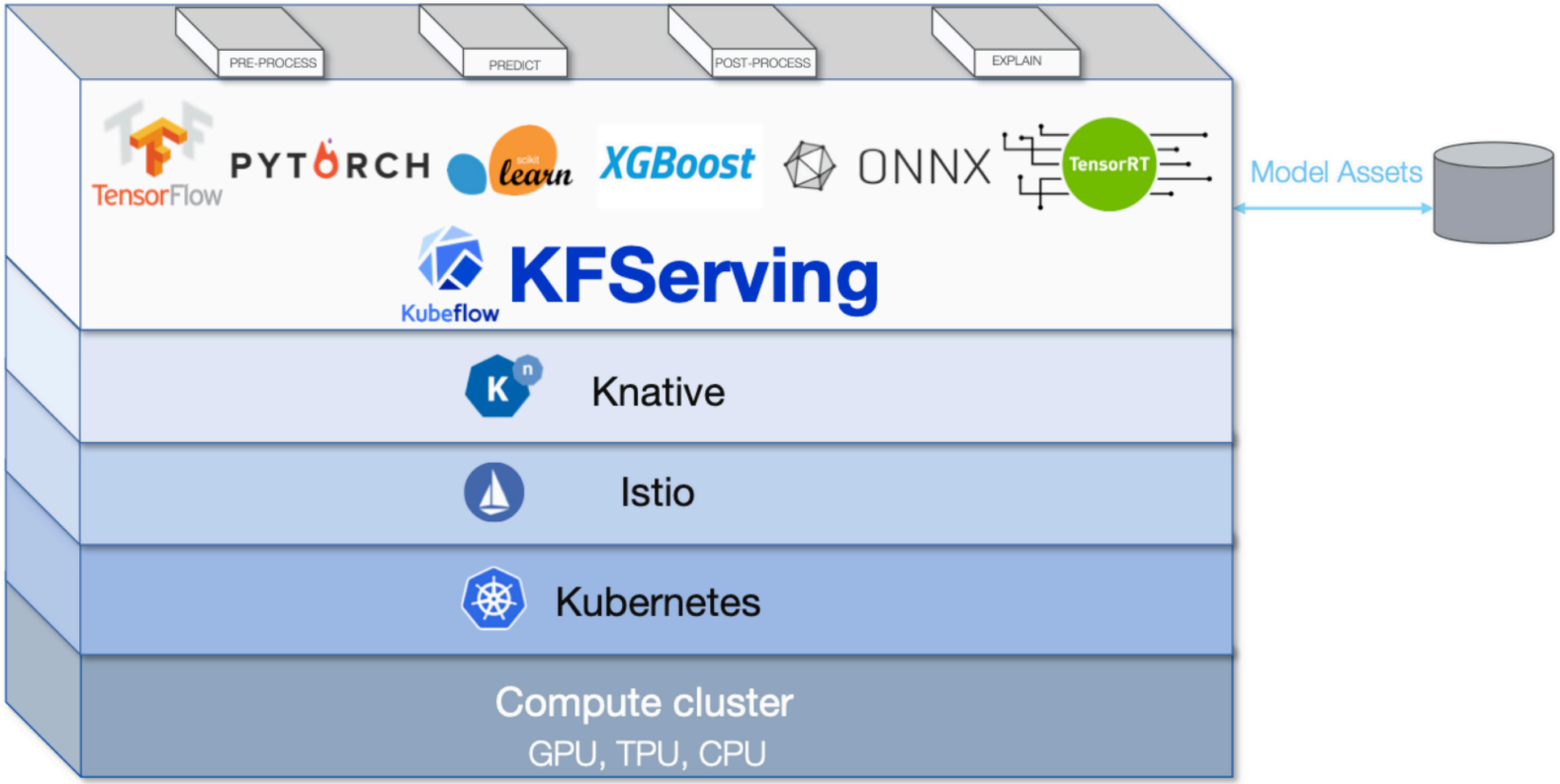
Seldon Core: Define Inference Pipeline



SELDON

LAB:
Simple Model Serving with SeldonCore

KFServing


PRE-PROCESS
PREDICT
POST-PROCESS
EXPLAIN
TensorFlow
PYTORCH
scikit learn
XGBoost
ONNX
TensorRT
Model Assets
Kubeflow
KFServing
Knative
Istio
Kubernetes
Compute cluster
GPU, TPU, CPU

Extras

- Hyper Parameter Tuning
- Neural Architecture Search
- Artifact Storage and Management

… and more to come