Choosing The Right MLOps Tools on Kubernetes

Ananda Dwi Rahmawati

Hello World!

- Cloud & DevOps Engineer, Singapore
- AWS Container Hero, Google Developer Expert Cloud - Modern Architecture, Open Source Enthusiast
- Master of Computer Science - University of Texas at Austin
- https://linktr.ee/misskecupbung

Agenda

- What is MLOps and Why Kubernetes?
- Key MLOps Stages & Challenges
- Kubernetes for MLOps: The Foundation
- Kubernetes MLOps Tooling Categories
- Data Versioning & Feature Stores
- Model Training & Experiment Tracking
- Model Deployment & Serving
- Monitoring & Observability
- Making the Right Choice: Evaluation Criteria
- Simple Use Case with Guide
- Q&A

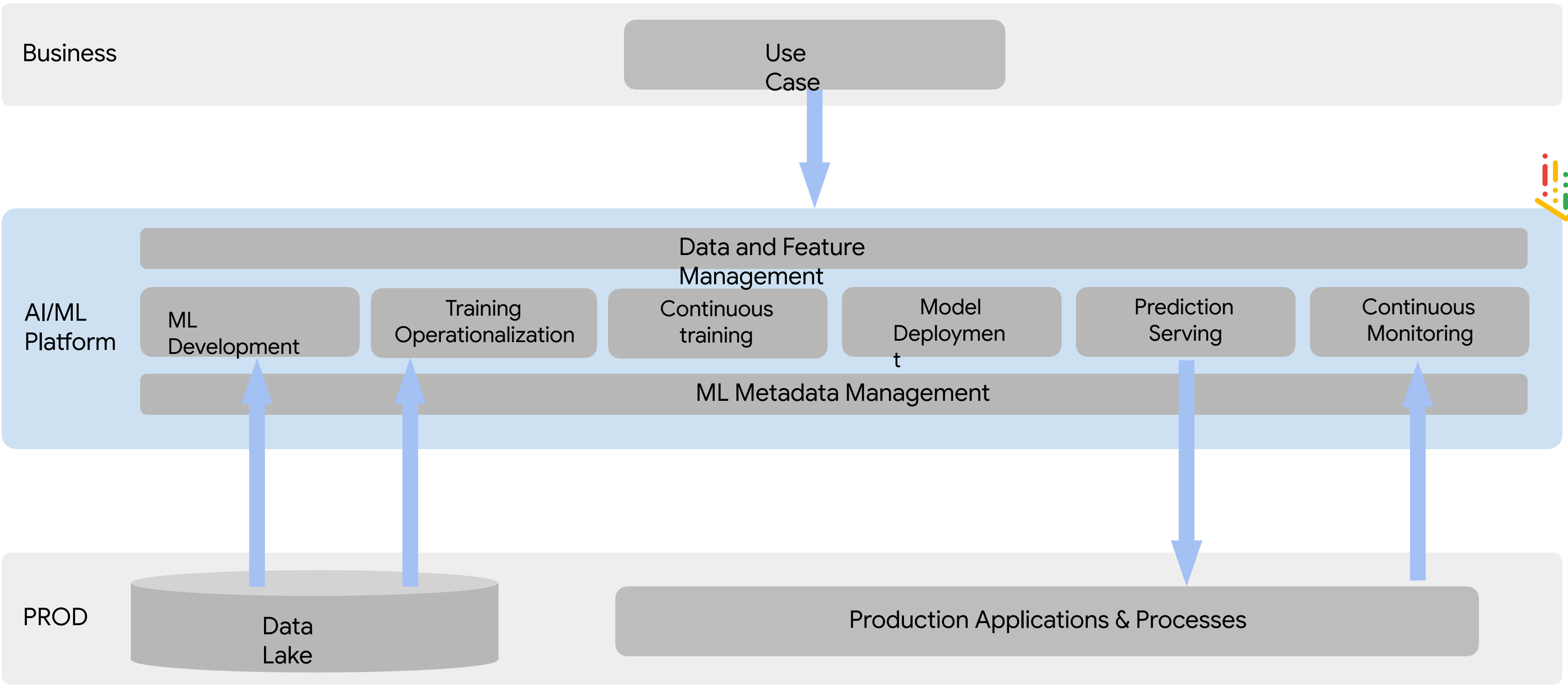
What is MLOps?

MLOps is a **set of practices** that aims to deploy and maintain ML models in production reliably and efficiently.

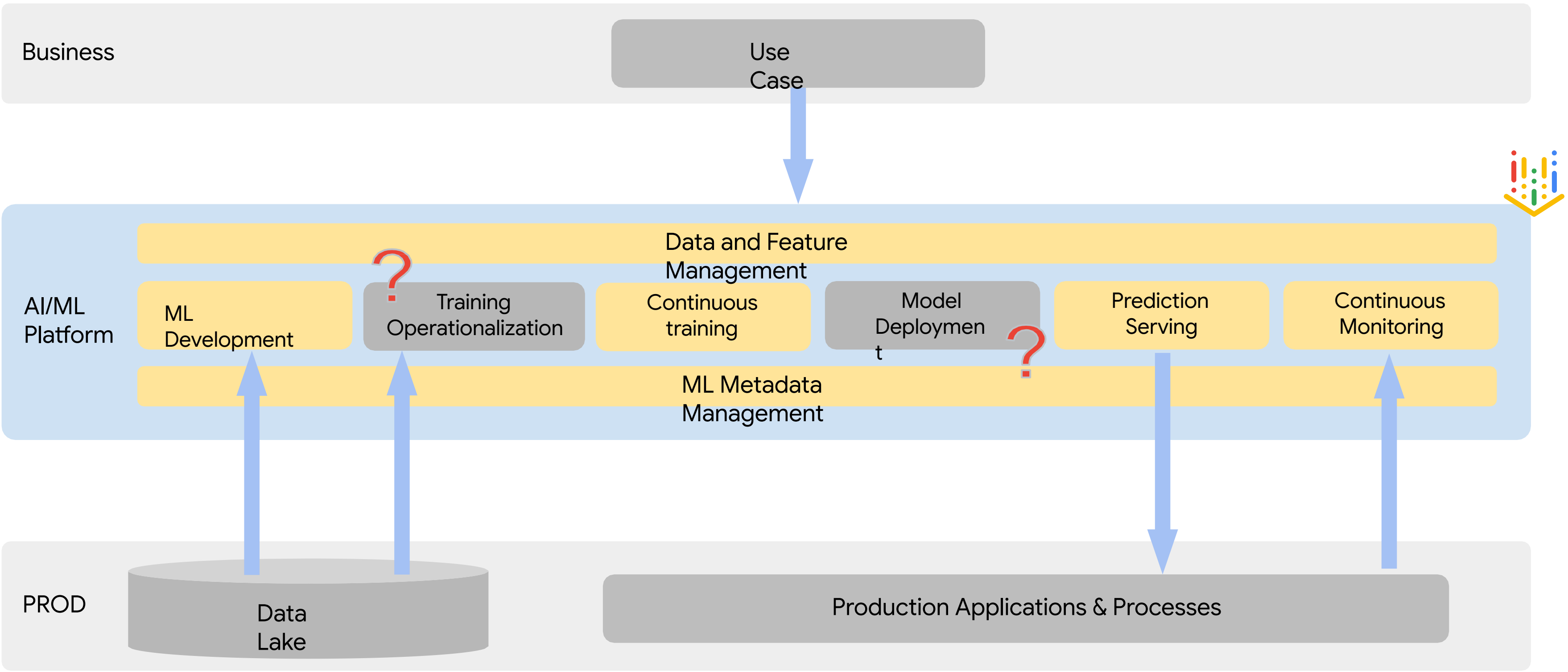
Goal: Bridge the gap between ML model development and operationalization.

Analogy: DevOps for Machine Learning.

MLOps: quick recap

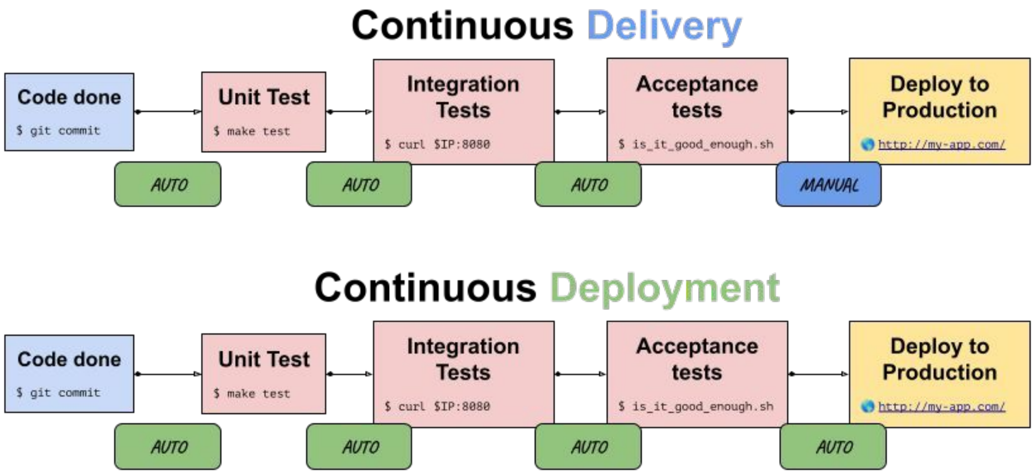


The “Ops” of MLOps

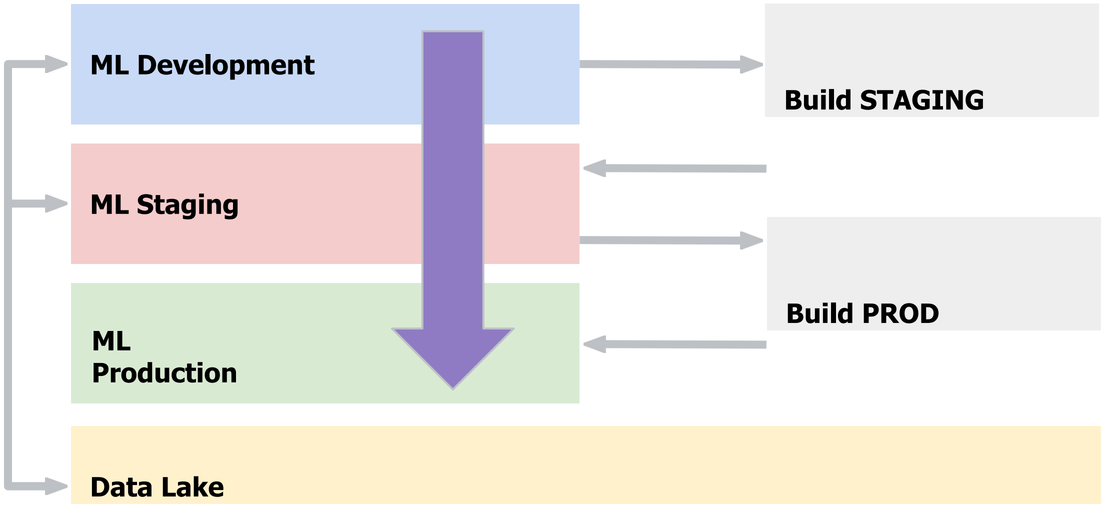


How would you implement continuous delivery* with ML today?

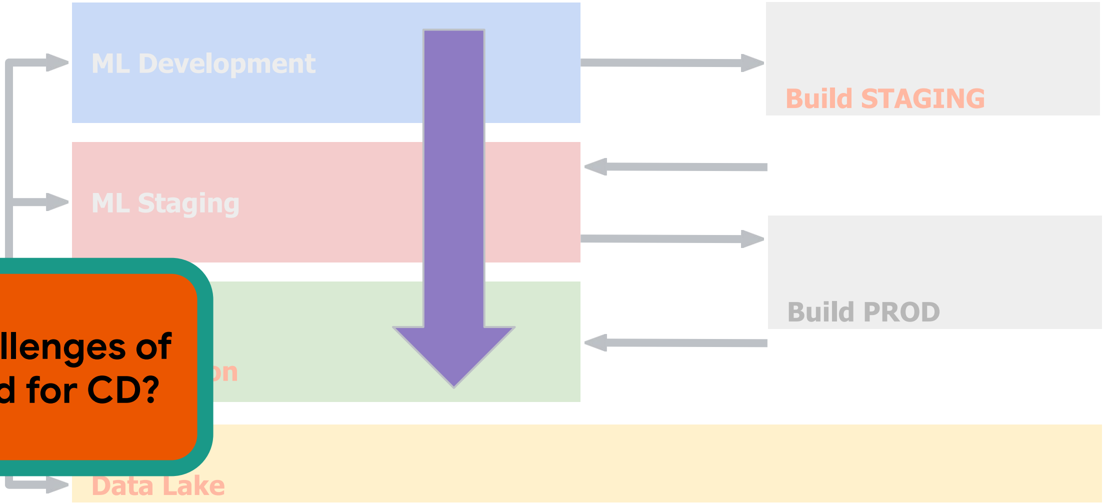
Some Terminology



Continuous delivery in ML



Continuous delivery in ML



What are the challenges of using Cloud Build for CD?

Opportunities

1. **Environment progression**. Ability to progress releases between [dev -> staging -> prod] environments
2. **Releases**. Releases should be **immutable**, and shall progress between environments
3. **Approval gates**. Approvals should be configurable before rolling out a release - per environment (e.g. prod)
4. **Rollback**. When a release fails, easily roll back to a previous one (eg, last stable)

MLOps Challenges

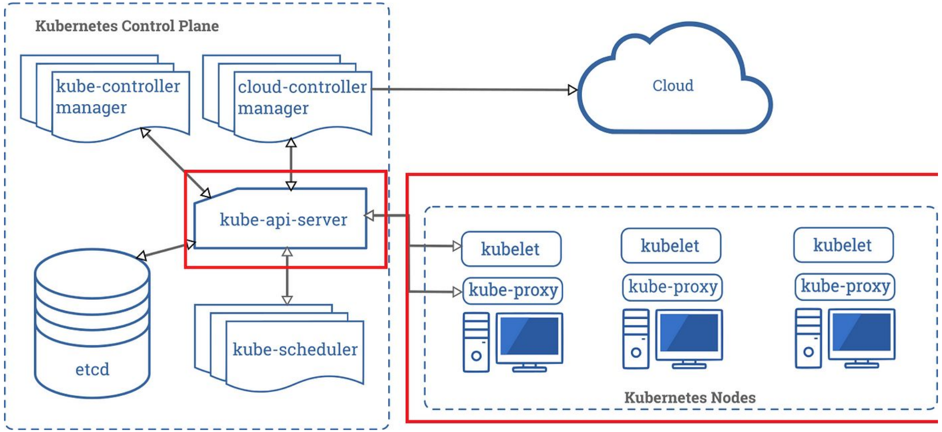
MLOps Challenges on Kubernetes

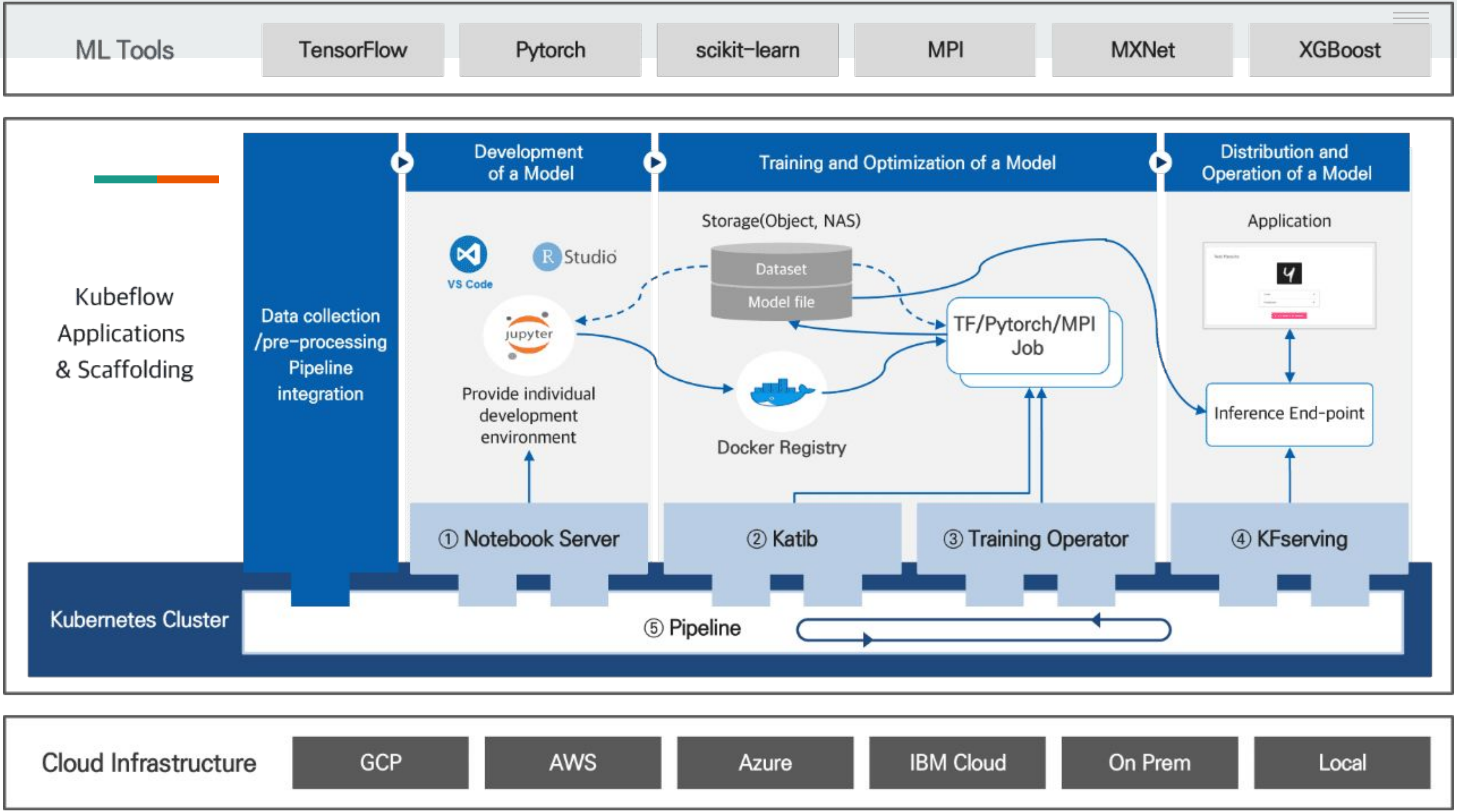
- **Complexity**: Kubernetes itself has a steep learning curve.
- **Resource Management**: Optimizing GPU usage, managing storage.
- **Data Management:** Large datasets, data versioning, feature stores.
- **Pipeline Orchestration**: Building robust, reproducible ML pipelines.
- **Model Lifecycle**: Tracking models from development to production.
- **Security**: Securing data, models, and infrastructure.

Why Kubernetes?

Why Kubernetes?

- **Portability**: Run ML workloads consistently across cloud and on-premise.
- **Scalability**: Easily scale resources for training and serving.
- **Resource Management**: Efficient allocation and isolation of compute, memory, and GPU.
- **Orchestration**: Automate deployment, scaling, and management of containers.
- **Ecosystem**: Rich set of tools and integrations.





ML Tools
TensorFlow
Pytorch
scikit-learn
MPI
MXNet
XGBoost
Kubeflow Applications & Scaffolding
Data collection /pre-processing Pipeline integration
Development of a Model
Training and Optimization of a Model
Distribution and Operation of a Model
VS Code
RStudio
jupyter
Provide individual development environment
Storage(Object, NAS)
Dataset
Model file
TF/Pytorch/MPI Job
Docker Registry
Application
Inference End-point
① Notebook Server
② Katib
③ Training Operator
④ KFserving
Kubernetes Cluster
⑤ Pipeline
Cloud Infrastructure
GCP
AWS
Azure
IBM Cloud
On Prem
Local

MLOps Tooling Categories

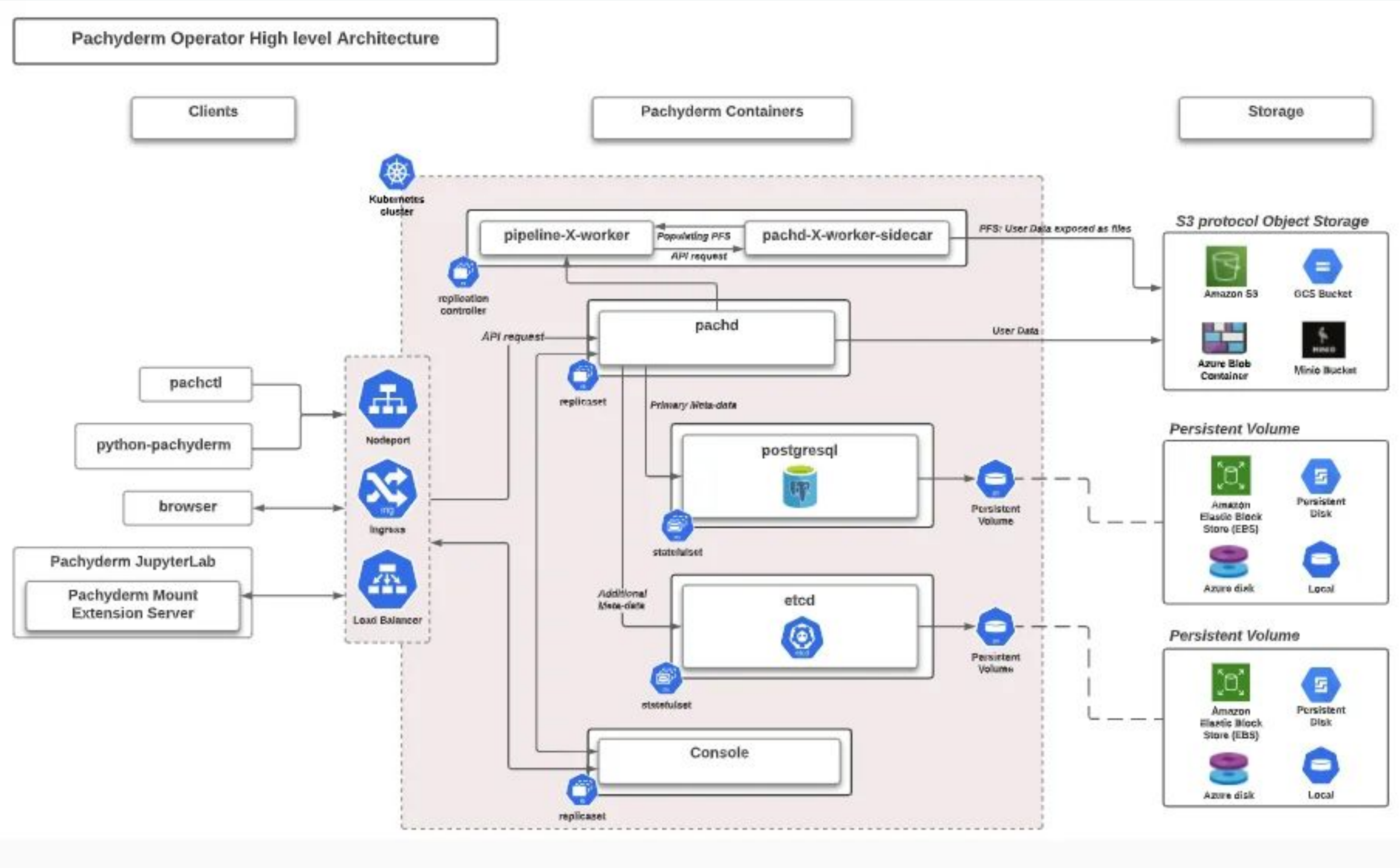
MLOps Tooling Categories

- Data Versioning & Feature Stores
- Model Training & Experiment Tracking
- Model Deployment & Serving
- Monitoring & Observability
- Pipeline Orchestration
- End-to-End Platforms
- … and many more https://github.com/kelvins/awesome-mlops

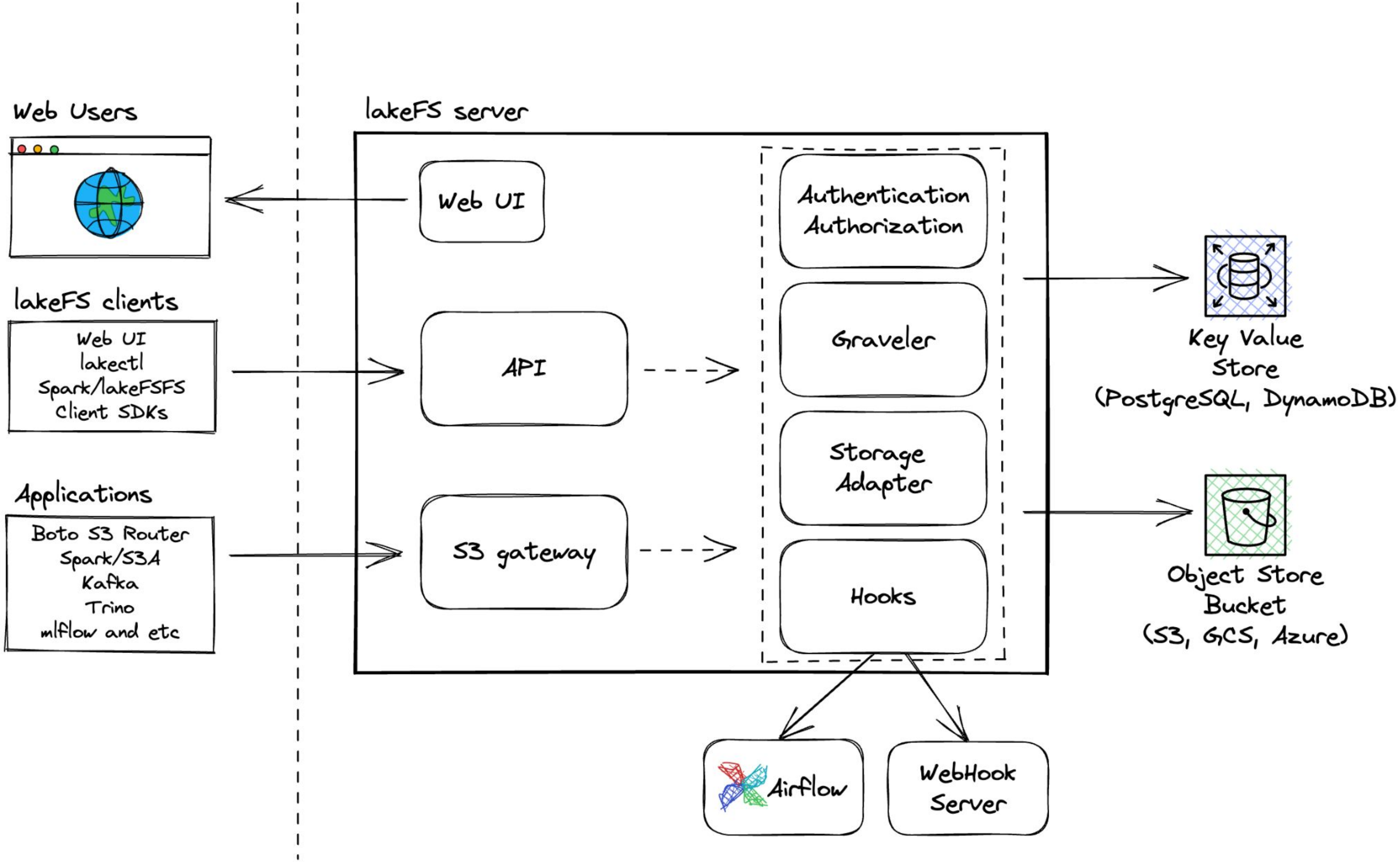
Data Versioning & Feature Stores

Data Versioning & Feature Stores

- Purpose: Manage **data changes**, ensure reproducibility, share features.
- Challenges: Large data volumes, schema evolution, **consistency**.
- Data Versioning
  - **DVC (Data Version Control)**: Git-like versioning for data and models. Integrates with S3, GCS, HDFS.
  - **Pachyderm**: Data versioning and data pipelines. Built on Kubernetes.
  - **LakeFS**: Git-like operations on data lakes.
- Feature Stores
  - **Feast**: Open-source feature store. Integrates with various data sources and serving layers.
  - **Hopsworks**: Enterprise feature store with a strong focus on MLOps.
  - Benefits: Feature reusability, consistency, reduced training-serving skew.



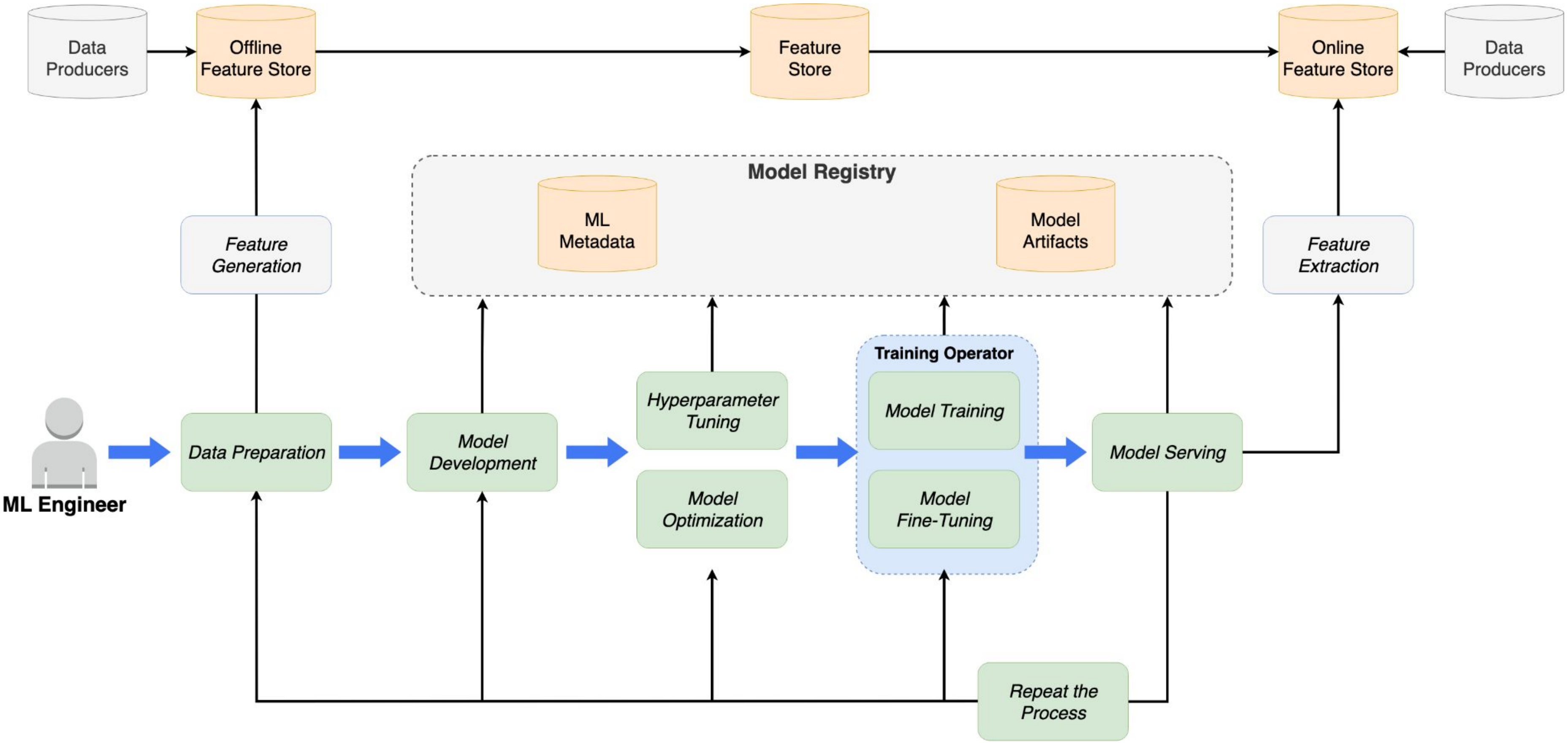
Pachyderm Operator High level Architecture
Clients
Pachyderm Containers
Storage
Kubernetes cluster
pipeline-X-worker
Populating PFS
API request
pachd-X-worker-sidecar
PFS: User Data exposed as files
S3 protocol Object Storage
Amazon S3
GCS Bucket
Azure Blob Container
Minio Bucket
replication controller
pachd
API request
User Data
replicaset
Primary Meta-data
pachctl
python-pachyderm
Nodeport
postgresql
Persistent Volume
Persistent Volume
Amazon Elastic Block Store (EBS)
Persistent Disk
Azure disk
Local
browser
Ingress
statefulset
Pachyderm JupyterLab
Pachyderm Mount Extension Server
Load Balancer
Additional Meta-data
etcd
Persistent Volume
statefulset
Console
replicaset


Web Users
lakeFS server
Web UI
Authentication
Authorization
lakeFS clients
Web UI
lakectl
Spark/lakeFSFS
Client SDKs
API
Graveler
Key Value
Store
(PostgreSQL, DynamoDB)
Storage
Adapter
Applications
Boto S3 Router
Spark/S3A
Kafka
Trino
mlflow and etc
S3 gateway
Hooks
Object Store
Bucket
(S3, GCS, Azure)
Airflow
WebHook
Server

Model Training & Experiment Tracking

Model Training & Experiment Tracking

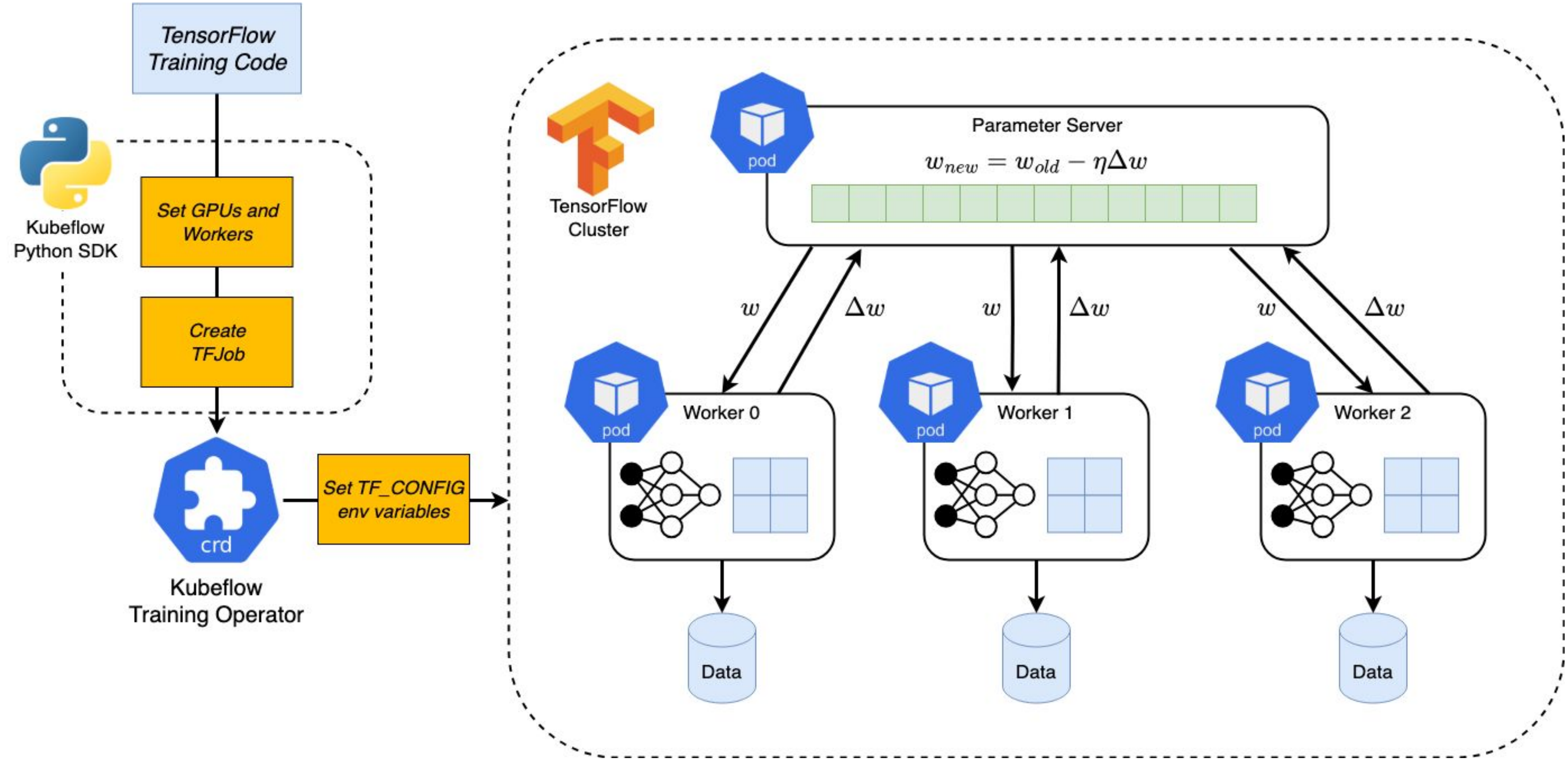
- **Purpose**: Efficiently train models, track experiments, manage hyperparameters.
- **Challenges**: Resource allocation, reproducibility, scaling training jobs.
- Model Training
  - **Kubeflow Training Operators** (TFJob, PyTorchJob, MPIJob): Run distributed training jobs natively on Kubernetes.
  - **Argo Workflows**: Can be used to orchestrate complex training workflows.
  - **Ray**: Unified framework for scaling AI and Python applications, including distributed training.
- Experiment Tracking
  - **MLflow**: Open-source platform for managing the ML lifecycle, including experiment tracking.
  - **Weights & Biases (W&B)**: Powerful experiment tracking, visualization, and collaboration platform.
  - **Neptune.ai**: Metadata store for MLOps, focusing on experiment tracking and model registry.


Data Producers
Offline Feature Store
Feature Store
Online Feature Store
Data Producers
Model Registry
ML Metadata
Model Artifacts
Feature Generation
Feature Extraction
Training Operator
Hyperparameter Tuning
Model Training
ML Engineer
Data Preparation
Model Development
Model Serving
Model Optimization
Model Fine-Tuning
Repeat the Process

Orchestration & Pipelines

Orchestration & Pipelines

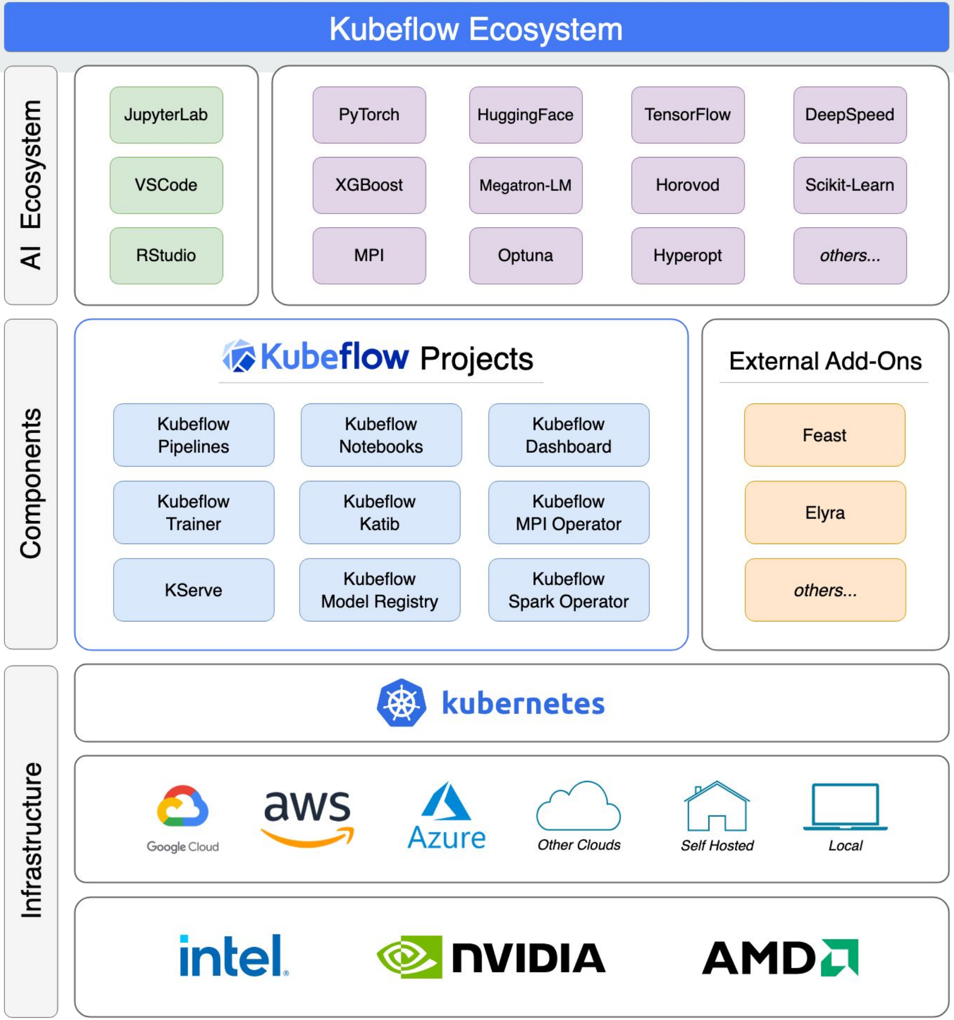
- Purpose: Automate the entire ML workflow, from data ingestion to model deployment.
- Challenges: Reproducibility, dependency management, error handling.
- Pipeline Orchestration Tools on Kubernetes
  - Kubeflow Pipelines: Component of Kubeflow, allows building and running reproducible ML pipelines.
  - Argo Workflows: Native Kubernetes workflow engine, highly flexible for ML pipelines.
  - Airflow on Kubernetes: Popular workflow orchestrator, can run tasks as Kubernetes Pods.

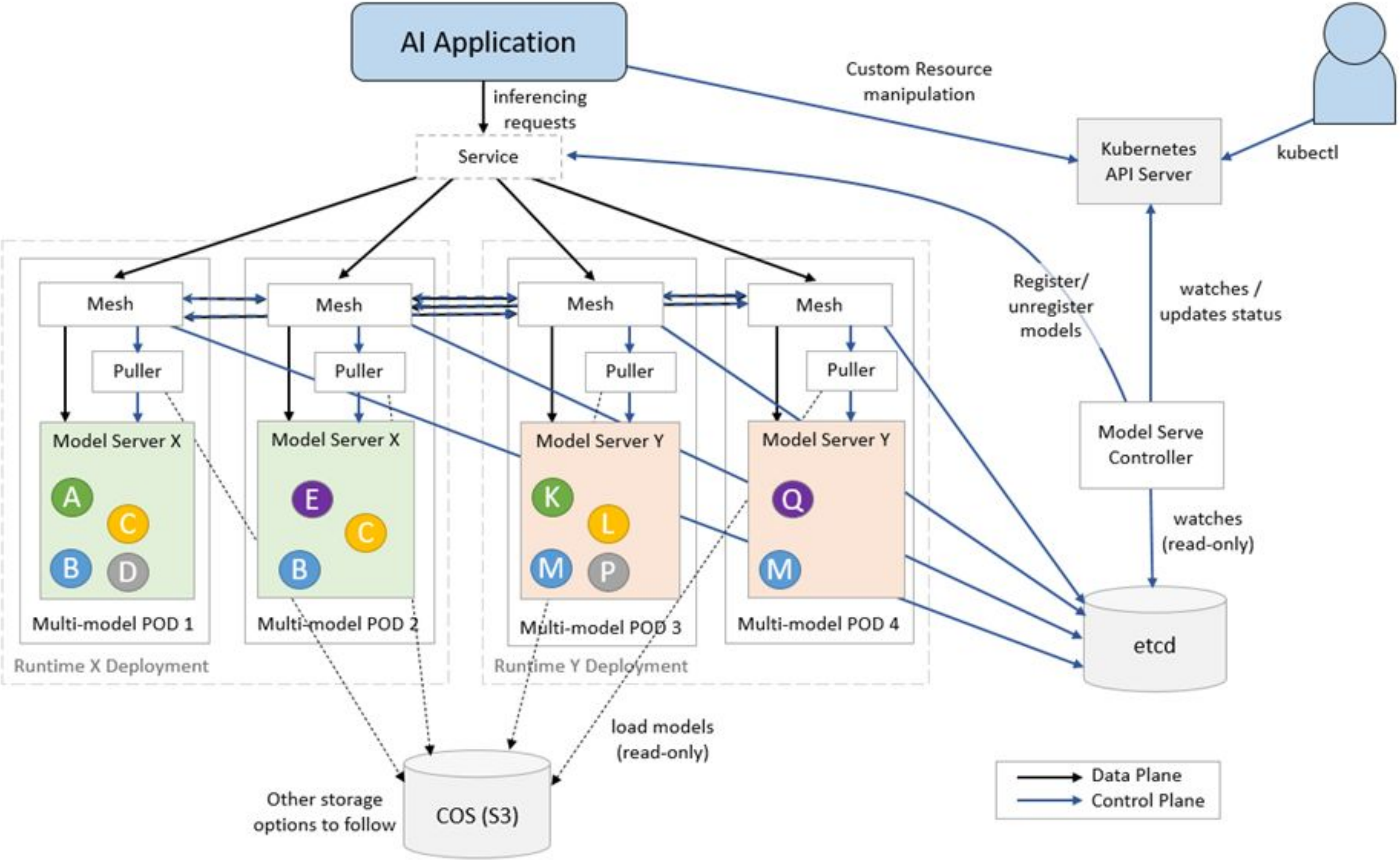

TensorFlow Training Code
Kubeflow Python SDK
Set GPUs and Workers
Create TFJob
Kubeflow Training Operator
crd
Set TF_CONFIG env variables
TensorFlow Cluster
pod
Parameter Server
$w_{new} = w_{old} - \eta \Delta w$
w
Δw
w
Δw
w
Δw
pod
Worker 0
pod
Worker 1
pod
Worker 2
Data
Data
Data

Model Deployment & Serving

Model Deployment & Serving

- Purpose: Expose trained models as APIs for inference.
- Challenges: Scalability, low latency, A/B testing, canary deployments.
- Model Serving Tools
  - KServe (formerly KFServing): Standardized model serving on Kubernetes. Supports various ML frameworks.
  - Seldon Core: Open-source platform for deploying ML models on Kubernetes. Advanced deployment strategies.
  - Triton Inference Server: NVIDIA's inference server for high-performance serving of deep learning models.
- Advanced Deployment Strategies on Kubernetes
  - Canary Deployments: Gradually shift traffic to new model versions.
  - A/B Testing: Route traffic to different model versions for comparison.
  - Blue/Green Deployments: Deploy new version alongside old, then switch traffic.
  - Tools: Istio, Linkerd (service mesh) can facilitate these strategies with KServe/Seldon.

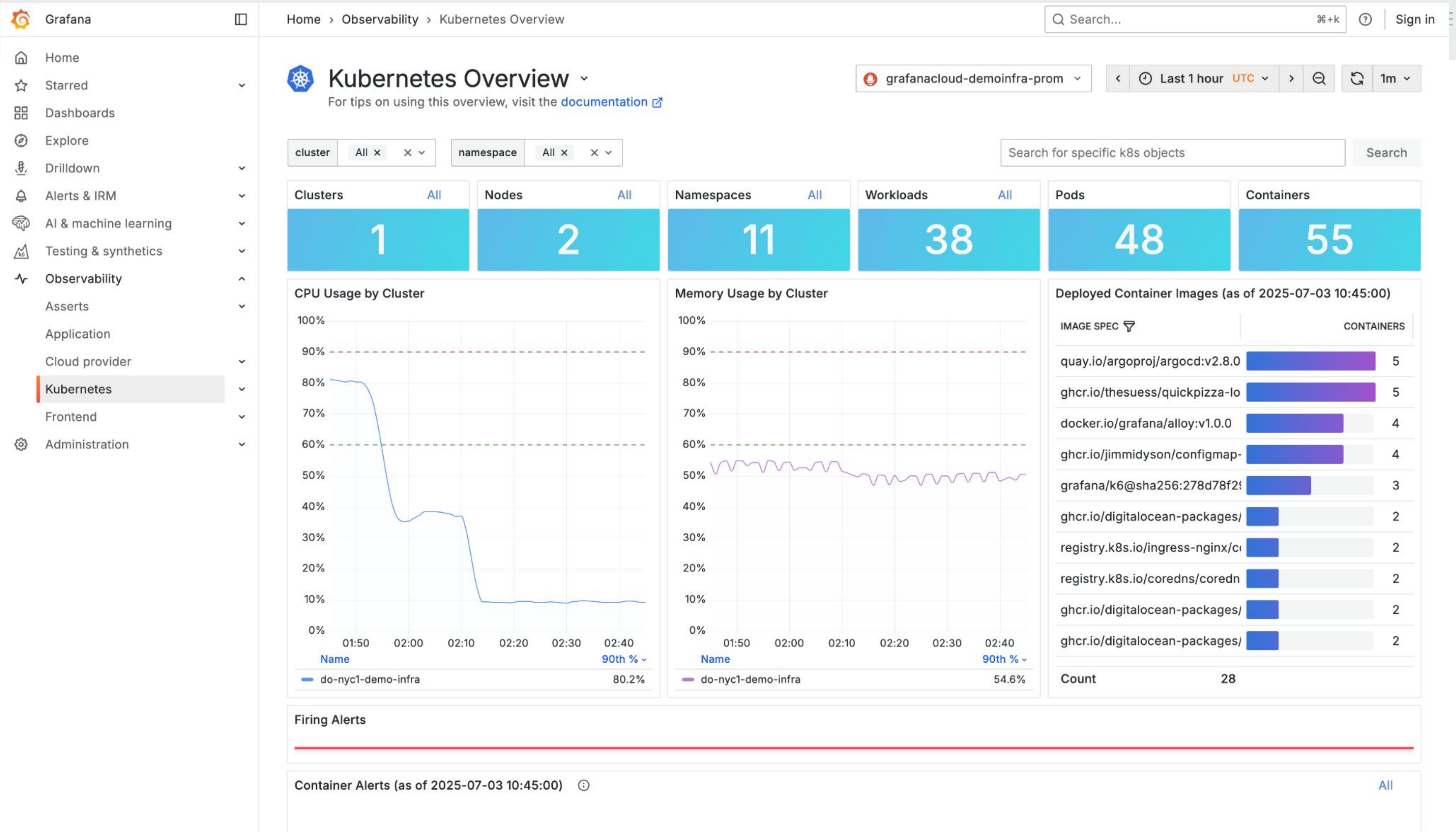

Kubeflow Ecosystem
AI Ecosystem
JupyterLab
VSCode
RStudio
PyTorch
HuggingFace
TensorFlow
DeepSpeed
XGBoost
Megatron-LM
Horovod
Scikit-Learn
MPI
Optuna
Hyperopt
others...
Components
Kubeflow Projects
Kubeflow Pipelines
Kubeflow Notebooks
Kubeflow Dashboard
Kubeflow Trainer
Kubeflow Katib
Kubeflow MPI Operator
KServe
Kubeflow Model Registry
Kubeflow Spark Operator
External Add-Ons
Feast
Elyra
others...
Infrastructure
kubernetes
Google Cloud
aws
Azure
Other Clouds
Self Hosted
Local
intel
NVIDIA
AMD


AI Application
inferencing requests
Custom Resource manipulation
kubectl
Kubernetes API Server
Service
Mesh
Mesh
Mesh
Mesh
Puller
Puller
Puller
Puller
Model Server X
Model Server X
Model Server Y
Model Server Y
A
C
B
D
E
C
B
K
L
M
P
Q
M
Multi-model POD 1
Multi-model POD 2
Multi-model POD 3
Multi-model POD 4
Runtime X Deployment
Runtime Y Deployment
Register/ unregister models
watches / updates status
Model Serve Controller
watches (read-only)
etcd
load models (read-only)
Other storage options to follow
COS (S3)
Data Plane
Control Plane

Monitoring & Observability

Monitoring & Observability

- Purpose: Track model performance, data drift, and infrastructure health in production.
- Challenges: Defining metrics, setting up alerts, visualizing performance.
- Monitoring
    - Prometheus & Grafana: Standard for infrastructure monitoring. Can be extended for model metrics.
    - Evidently AI: Open-source tool for data drift and model performance monitoring.
    - Fiddler AI / Arize AI: Commercial platforms for ML model monitoring, explainability, and debugging.
- Observability for MLOps
    - Logging: Centralized logging (e.g., Fluentd, Loki, ELK Stack).
    - Tracing: Distributed tracing for complex inference paths (e.g., Jaeger, OpenTelemetry).
    - Alerting: Integrate with PagerDuty, Slack, etc., for critical issues.


Grafana
Home › Observability › Kubernetes Overview
Search...
⌘+k
Sign in
Home
Starred
Dashboards
Explore
Drilldown
Alerts & IRM
AI & machine learning
Testing & synthetics
Observability
Asserts
Application
Cloud provider
Kubernetes
Frontend
Administration
Kubernetes Overview
For tips on using this overview, visit the documentation
grafanacloud-demoinfra-prom
Last 1 hour UTC
1m
cluster All
namespace All
Search for specific k8s objects
Search
Clusters All
1
Nodes All
2
Namespaces All
11
Workloads All
38
Pods
48
Containers
55
CPU Usage by Cluster
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
01:50 02:00 02:10 02:20 02:30 02:40
Name
90th %
do-nyc1-demo-infra 80.2%
Memory Usage by Cluster
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
01:50 02:00 02:10 02:20 02:30 02:40
Name
90th %
do-nyc1-demo-infra 54.6%
Deployed Container Images (as of 2025-07-03 10:45:00)
IMAGE SPEC
CONTAINERS
quay.io/argoproj/argocd:v2.8.0 5
ghcr.io/thesuess/quickpizza-lo 5
docker.io/grafana/alloy:v1.0.0 4
ghcr.io/jimmidyson/configmap- 4
grafana/k6@sha256:278d78f2 3
ghcr.io/digitalocean-packages/ 2
registry.k8s.io/ingress-nginx/c 2
registry.k8s.io/coredns/coredn 2
ghcr.io/digitalocean-packages/ 2
ghcr.io/digitalocean-packages/ 2
Count 28
Firing Alerts
Container Alerts (as of 2025-07-03 10:45:00)
All

Making the Right Choice: Evaluation Criteria

Making the Right Choice: Evaluation Criteria

- Open Source vs. Commercial: Cost, community support, features.
- Ease of Use & Learning Curve: For data scientists and engineers.
- Scalability & Performance: Can it handle your current and future needs?
- Integration with Existing Stack: Compatibility with your data sources, ML frameworks.
- Community Support & Documentation: Active development, helpful resources.
- Security & Compliance: Meets your organization's requirements.
- Vendor Lock-in: How easy is it to switch tools?

Simple Use Case with Guide

References

References

- https://atlan.com/pachyderm-data-lineage/
- https://github.com/AlexIoannides/kubernetes-mlops
- https://github.com/awesome-mlops/awesome-mlops-kubernetes
- https://play.grafana.org/
- https://medium.com/@craftworkai/utilizing-kubernetes-for-an-effective-mlops-platform-efc98325eaca
- https://www.datacamp.com/blog/top-mlops-tools
- https://github.com/kubeflow/pipelines/blob/master/developer_guide.md
- https://minikube.sigs.k8s.io/
- https://kubernetes.io/docs/tasks/tools/
- https://alexioannides.com/2019/01/10/deploying-python-ml-models-with-flask-docker-and-kubernetes/

Thank you.