MLOps

Index

- Assignment Brief
- System Overview
- Detailed Overview
- What can be improved
  - Short Term improvements

Assignment/Problem Statement

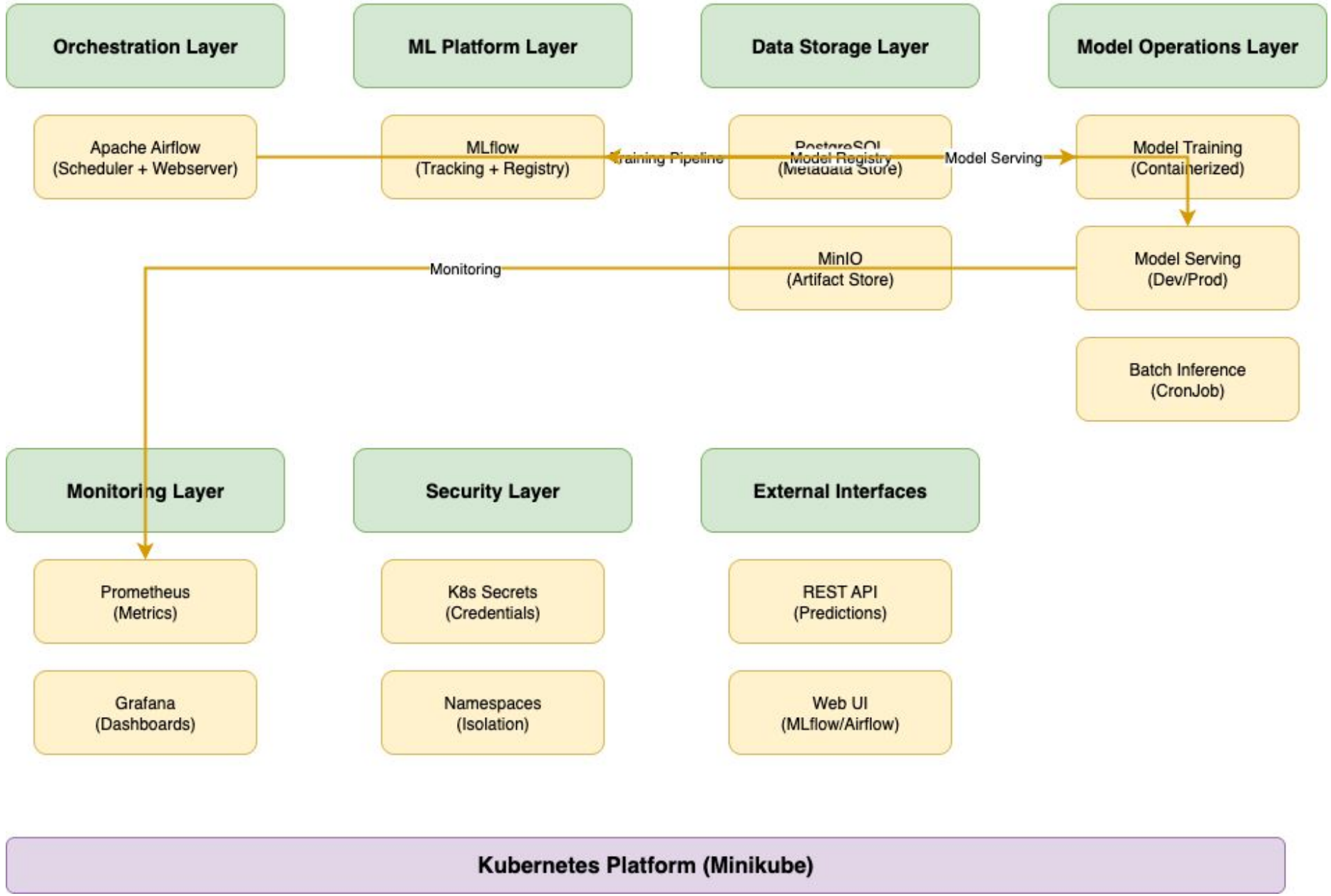
Design and implement a scalable machine learning model lifecycle management solution that demonstrates your MLOps expertise and architectural thinking.

- Functional Requirements
  - ~~Model Training~~**: Automated training pipeline with experiment tracking
  - ~~Model Deployment~~**: Multi-environment deployment with serving capabilities
  - ~~Model Monitoring~~**: Performance monitoring and drift detection
  - ~~Model Management~~**: Versioning, registry, and governance workflows
- Technical Requirements
  - ~~Scalability~~**: Support multiple models and concurrent users
  - ~~Reliability~~**: Production-ready with appropriate error handling
  - ~~Observability~~**: Monitoring, logging, and alerting capabilities
  - x Security**: Authentication and access controls
- Constraints
  - ~~Must be containerized and cloud-deployable~~
  - ~~Include both real-time and batch inference options~~
  - ~~Provide a user interface or API for interaction~~
  - ~~Support at least one popular ML framework~~
- ~~Bonus~~: Design a workflow orchestrator (using Airflow/Kubeflow). Integrate MLflow with workflow orchestration, effectively mimicking model training/retraining automation in a production environment

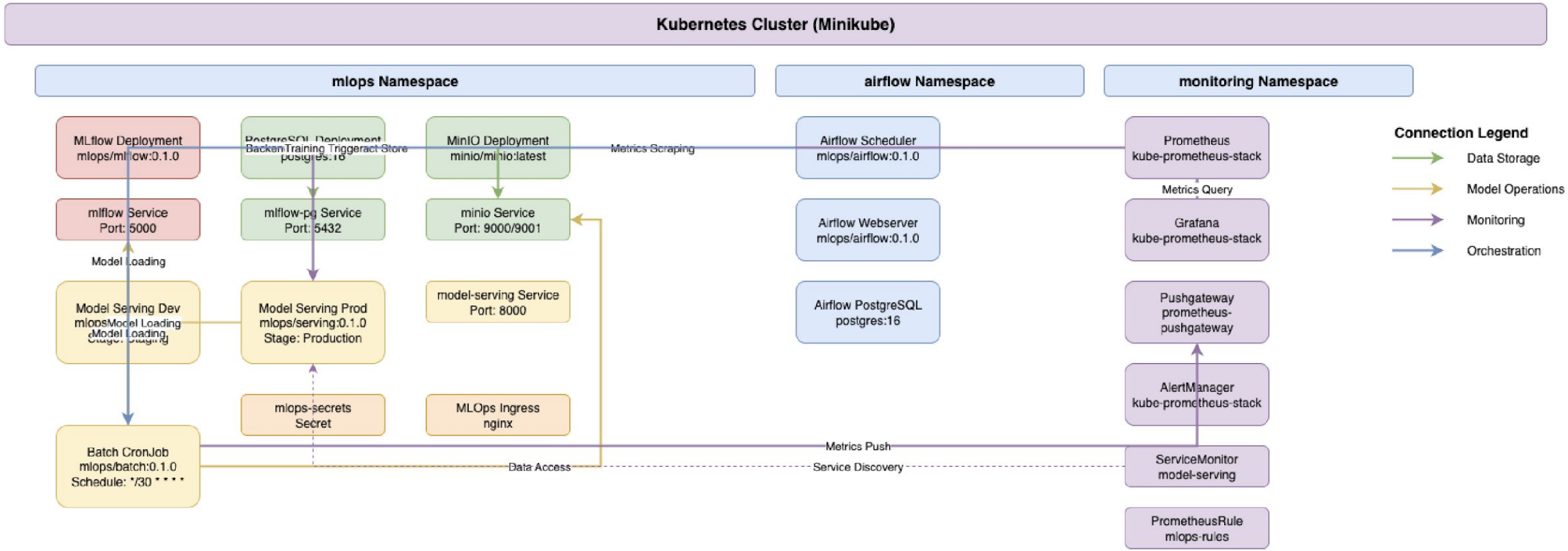
Key Decisions

- Orchestration: Airflow vs Kubeflow
- Registry: Mlflow
- Remote Storage: Minio (to mimic S3)
- Model Serving: FastApi
- Observability: kube-prometheus-stack + pushgateway

System Overview



Deployment Architecture



External Access

kubectl port-forward svc/mlflow 5000:5000 http://localhost:5000	kubectl port-forward svc/model-serving 8000:8000 http://localhost:8000	kubectl port-forward svc/kube-prom-grafana 3000:80 http://localhost:3000	kubectl port-forward svc/airflow-webserver 8080:8080 http://localhost:8080	kubectl port-forward svc/minio 9001:9001 http://localhost:9001

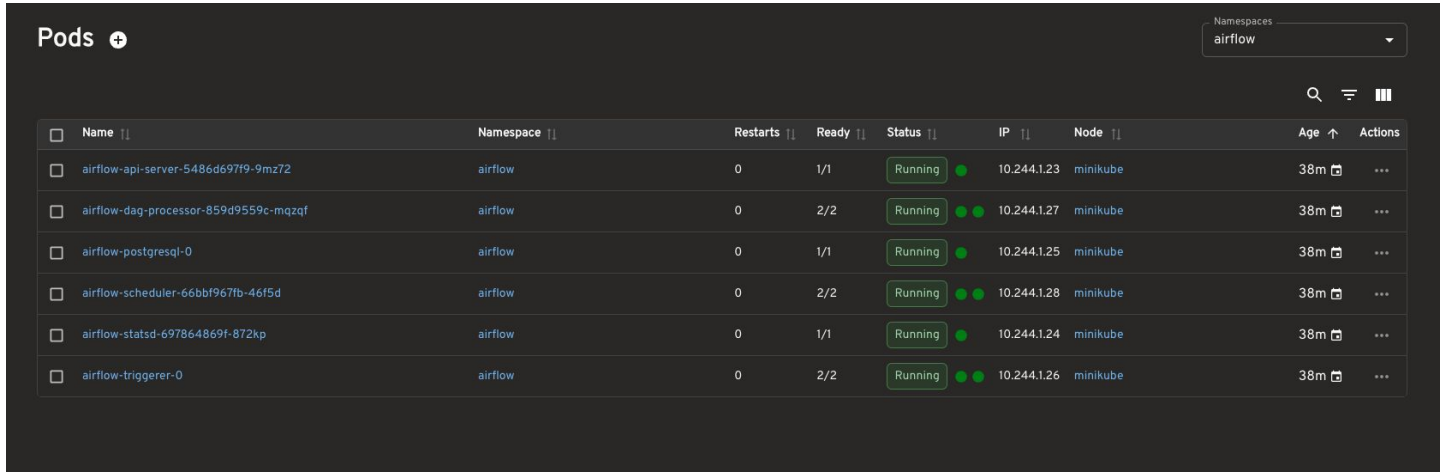
Key Deployment Details

- All images built with Makefile and loaded into Minikube
- Secrets managed via Kubernetes Secrets
- ServiceMonitors enable Prometheus auto-discovery
- CronJob runs batch inference every 30 minutes
- Blue/Green deployment via model stages (Staging/Production)
- Airflow DAG orchestrates training → registry → deployment
- MLflow tracks experiments and manages model versions
- MinIO stores artifacts and batch data
- PostgreSQL stores MLflow metadata and Airflow state
- Prometheus scrapes metrics from serving endpoints
- Grafana dashboards visualize metrics and alerts
- Pushgateway collects batch job metrics
- AlertManager sends notifications for drift/perf issues
- Ingress provides external access (demo setup)
- Namespaces provide logical separation

Demo Flow

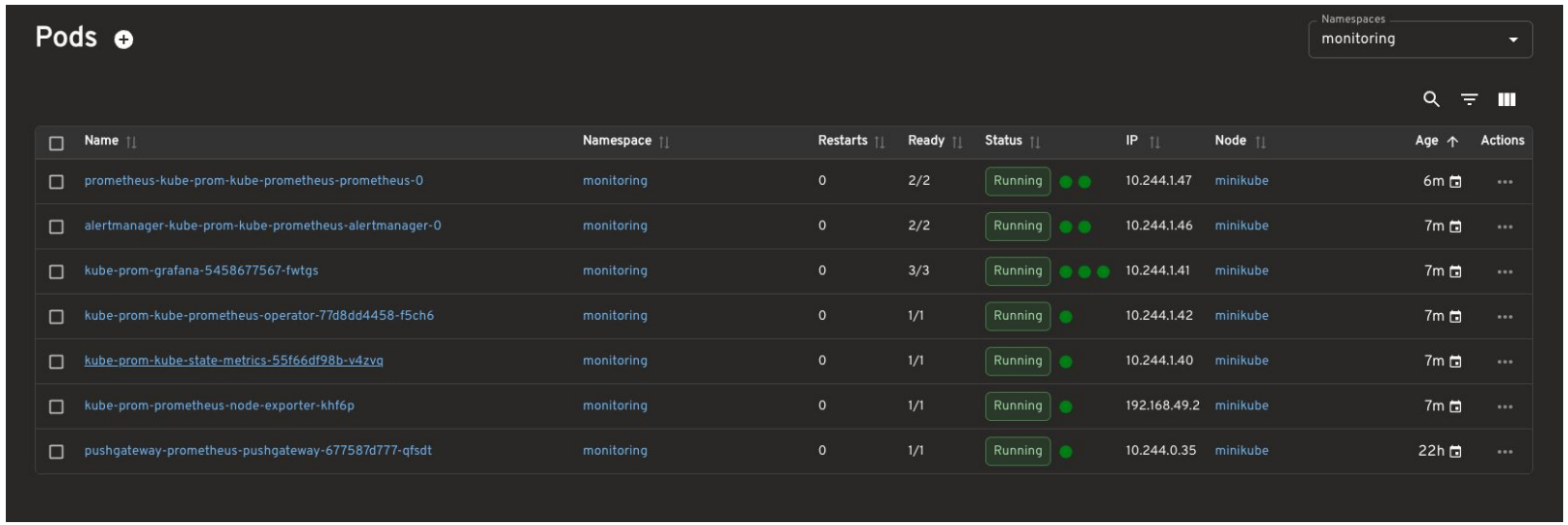
- Setup minikube
- Make images
- Setup core
    - namespaces
    - secrets
    - postgres for mlops
    - minio
        - Create bucket in minio
    - Mlflow
- Setup monitoring plane
    - kube-Prometheus stack
    - Pushgateway
    - Service Monitors – airflow, model serving, Prometheus alerts
- Deploy Airflow
- Deploy Model Serving
    - Leads to error as nothing exists so far
- Deploy one time staging model
    - Dev model comes up
    - Showcase curl request to this dev model
    - Showcase metrics in prometheus
    - Prod model will still be erroring out
- Run Airflow dag to showcase that once staging model is promoted to Prd, Prd model will come up
- Showcase Batch job for model monitoring
- Showcase alerts for model monitoring

Airflow NS



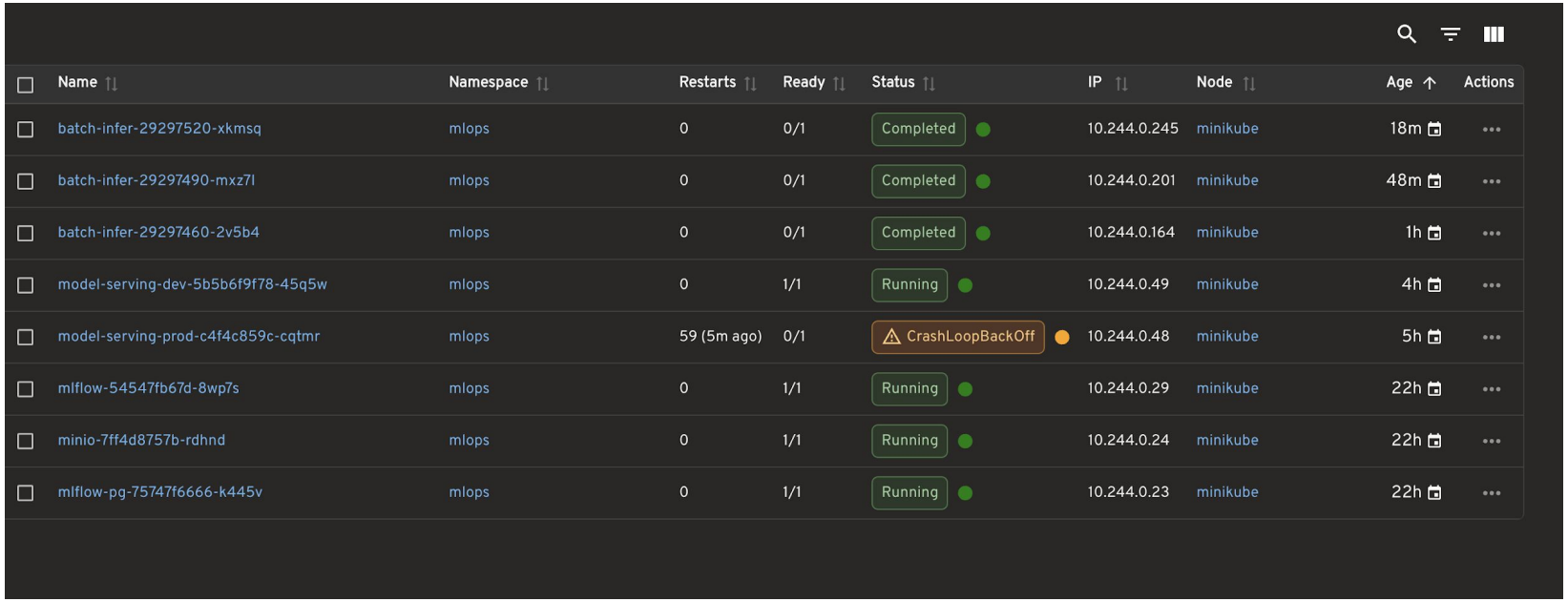
Name	Namespace	Restarts	Ready	Status	IP	Node	Age	Actions
airflow-api-server-5486d697f9-9mz72	airflow	0	1/1	Running	10.244.1.23	minikube	38m	...
airflow-dag-processor-859d9559c-mqzqf	airflow	0	2/2	Running	10.244.1.27	minikube	38m	...
airflow-postgresql-0	airflow	0	1/1	Running	10.244.1.25	minikube	38m	...
airflow-scheduler-66bbf967fb-46f5d	airflow	0	2/2	Running	10.244.1.28	minikube	38m	...
airflow-statsd-697864869f-872kp	airflow	0	1/1	Running	10.244.1.24	minikube	38m	...
airflow-triggerer-0	airflow	0	2/2	Running	10.244.1.26	minikube	38m	...

Monitoring NS



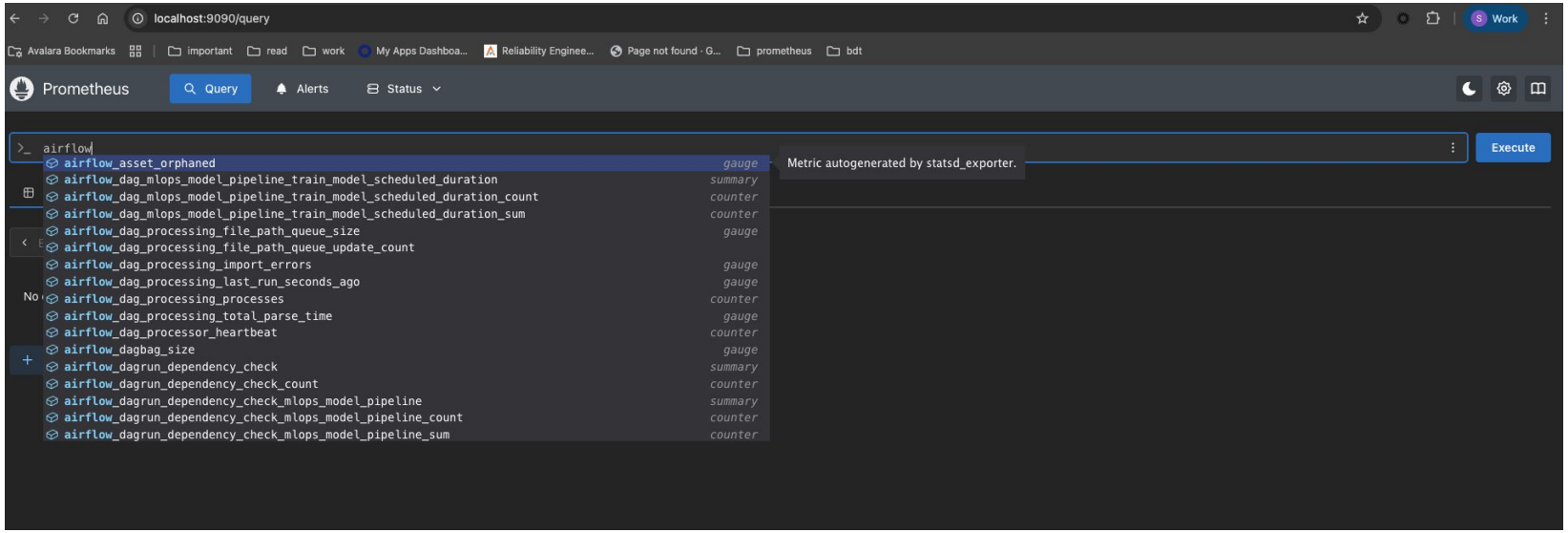
Name	Namespace	Restarts	Ready	Status	IP	Node	Age	Actions
prometheus-kube-prom-kube-prometheus-prometheus-0	monitoring	0	2/2	Running	10.244.1.47	minikube	6m	...
alertmanager-kube-prom-kube-prometheus-alertmanager-0	monitoring	0	2/2	Running	10.244.1.46	minikube	7m	...
kube-prom-grafana-5458677567-fwtgs	monitoring	0	3/3	Running	10.244.1.41	minikube	7m	...
kube-prom-kube-prometheus-operator-77d8dd4458-f5ch6	monitoring	0	1/1	Running	10.244.1.42	minikube	7m	...
kube-prom-kube-state-metrics-55f66df98b-v4zvq	monitoring	0	1/1	Running	10.244.1.40	minikube	7m	...
kube-prom-prometheus-node-exporter-khf6p	monitoring	0	1/1	Running	192.168.49.2	minikube	7m	...
pushgateway-prometheus-pushgateway-677587d777-qfsdt	monitoring	0	1/1	Running	10.244.0.35	minikube	22h	...

mlops NS

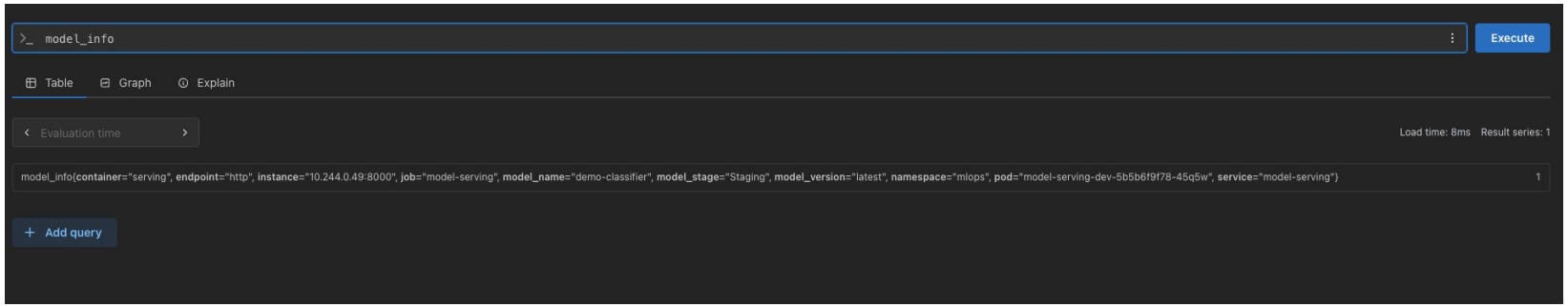


Name	Namespace	Restarts	Ready	Status	IP	Node	Age	Actions
batch-infer-29297520-xkmsq	mlops	0	0/1	Completed	10.244.0.245	minikube	18m	•••
batch-infer-29297490-mxz7l	mlops	0	0/1	Completed	10.244.0.201	minikube	48m	•••
batch-infer-29297460-2v5b4	mlops	0	0/1	Completed	10.244.0.164	minikube	1h	•••
model-serving-dev-5b5b6f9f78-45q5w	mlops	0	1/1	Running	10.244.0.49	minikube	4h	•••
model-serving-prod-c4f4c859c-cqtmr	mlops	59 (5m ago)	0/1	CrashLoopBackOff	10.244.0.48	minikube	5h	•••
mlflow-54547fb67d-8wp7s	mlops	0	1/1	Running	10.244.0.29	minikube	22h	•••
minio-7ff4d8757b-rdhnd	mlops	0	1/1	Running	10.244.0.24	minikube	22h	•••
mlflow-pg-75747f6666-k445v	mlops	0	1/1	Running	10.244.0.23	minikube	22h	•••

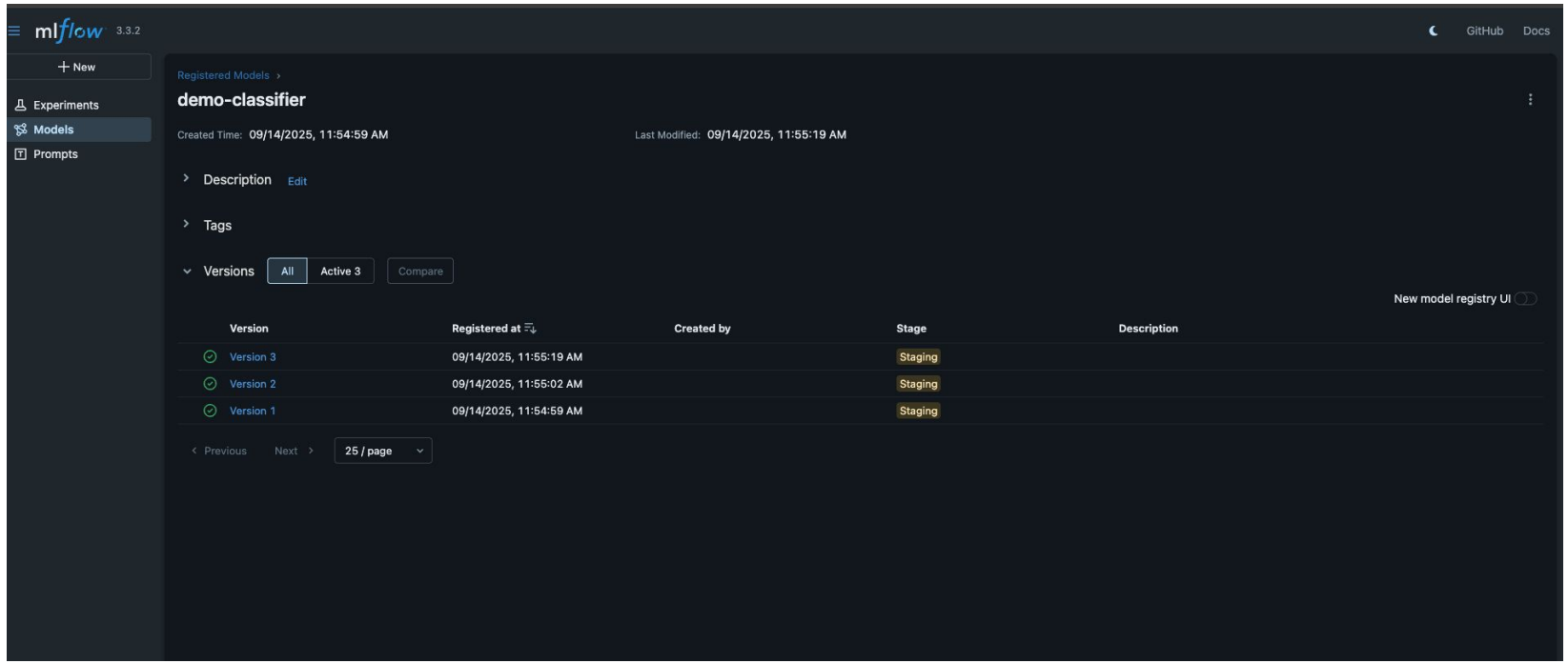
Airflow metrics in Prometheus captured from statsd via ServiceMonitor



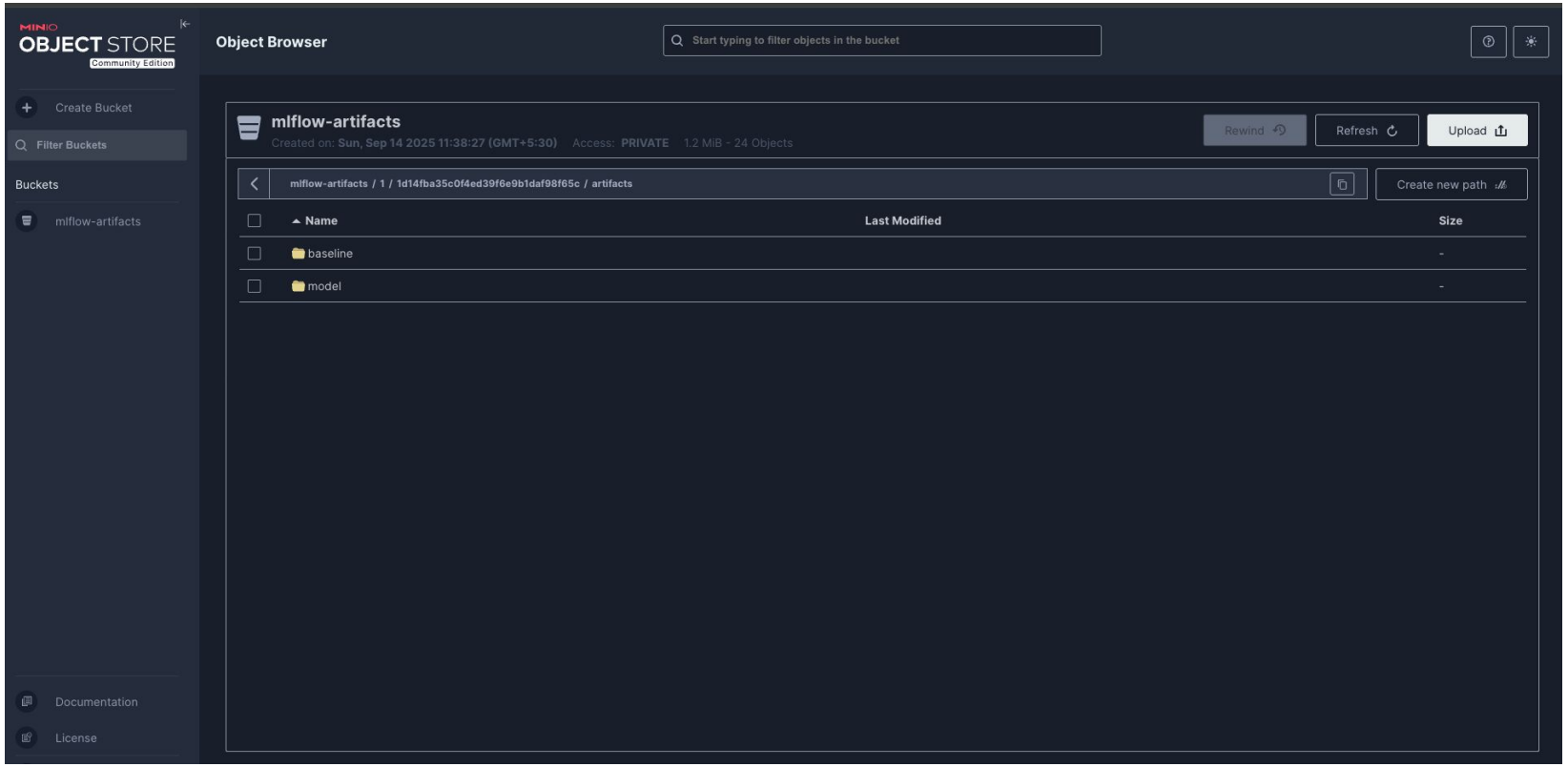
Model serving metrics via ServiceMonitor



Mlflow experiment recording



Minio object store - records both model and baseline psi



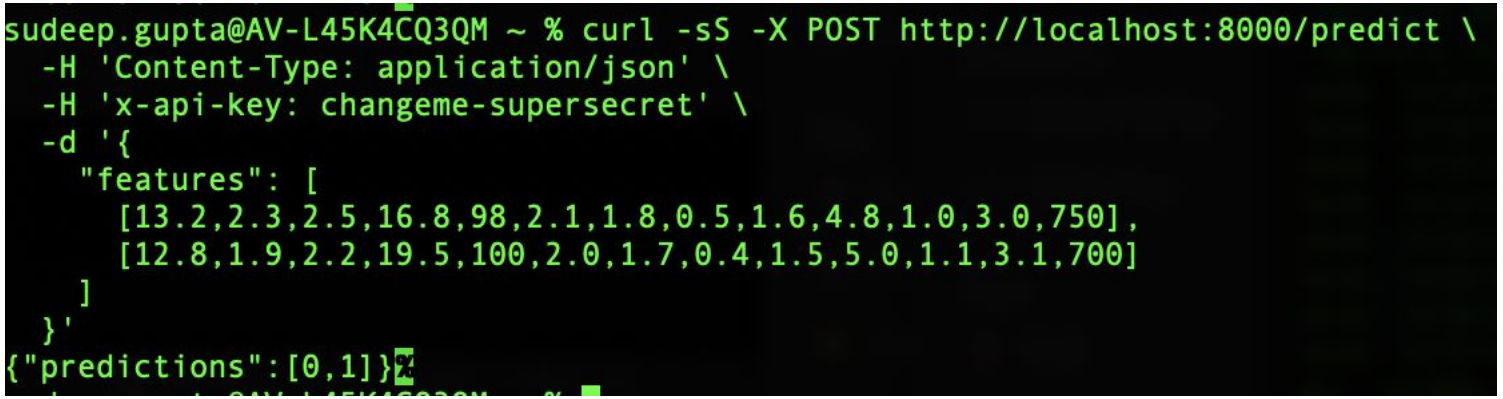
Drift detection job running which runs on schedule to get batch of recently predicted data and pushes psi metrics to Prometheus to check for deviations

```
2025-09-14T08:35:31.039542925Z Starting batch drift detection for model: demo-classifier/Production
2025-09-14T08:35:31.039557050Z No versions found in stage 'Production', using latest version
2025-09-14T08:35:31.039558133Z Using model version: 3 (stage: Staging)
2025-09-14T08:35:31.039558883Z Loaded baseline statistics for 13 features
2025-09-14T08:35:31.039559508Z Generating demo data for drift detection...
2025-09-14T08:35:31.039560300Z Calculating PSI for each feature...
2025-09-14T08:35:31.039561050Z PSI for alcohol: 0.0384
2025-09-14T08:35:31.039561675Z PSI for malic_acid: 0.0596
2025-09-14T08:35:31.039562383Z PSI for ash: 0.1000
2025-09-14T08:35:31.039563008Z PSI for alcalinity_of_ash: 0.0349
2025-09-14T08:35:31.039563675Z PSI for magnesium: 0.0643
2025-09-14T08:35:31.039564300Z PSI for total_phenols: 0.0241
2025-09-14T08:35:31.039565008Z PSI for flavanoids: 0.0452
2025-09-14T08:35:31.039565758Z PSI for nonflavanoid_phenols: 0.0172
2025-09-14T08:35:31.039566508Z PSI for proanthocyanins: 0.0261
2025-09-14T08:35:31.039567133Z PSI for color_intensity: 0.0354
2025-09-14T08:35:31.039567717Z PSI for hue: 0.0773
2025-09-14T08:35:31.039568300Z PSI for od280/od315_of_diluted_wines: 0.0338
2025-09-14T08:35:31.039568967Z PSI for proline: 0.0942
2025-09-14T08:35:31.039569550Z Average PSI: 0.0500
2025-09-14T08:35:31.039570175Z Successfully pushed PSI metric: 0.0500
2025-09-14T08:35:31.039571092Z Batch drift detection completed successfully
```

Model Serving

```
2025-09-14T08:03:21.565630941Z /usr/local/lib/python3.11/site-packages/sklearn/base.py:442: InconsistentVersionWarning: Trying to unpickle estim
orestClassifier from version 1.5.1 when using version 1.7.2. This might lead to breaking code or invalid results. Use at your own risk. For more
 refer to:
2025-09-14T08:03:21.565646816Z https://scikit-learn.org/stable/model_persistence.html#security-maintainability-limitations
2025-09-14T08:03:21.565648358Z   warnings.warn(
2025-09-14T08:03:21.582245864Z INFO:     Application startup complete.
2025-09-14T08:03:21.582451406Z INFO:     Uvicorn running on http://0.0.0.0:8000 (Press CTRL+C to quit)
2025-09-14T08:03:27.811458212Z Model loaded successfully: demo-classifier/Staging
2025-09-14T08:03:27.811539253Z INFO:     10.244.0.37:43474 - "GET /metrics HTTP/1.1" 200 OK
2025-09-14T08:03:42.800289064Z INFO:     10.244.0.37:38508 - "GET /metrics HTTP/1.1" 200 OK
2025-09-14T08:03:57.388543935Z INFO:     127.0.0.1:34720 - "POST /predict HTTP/1.1" 200 OK
2025-09-14T08:03:57.800286029Z INFO:     10.244.0.37:51924 - "GET /metrics HTTP/1.1" 200 OK
2025-09-14T08:04:12.801112321Z INFO:     10.244.0.37:53606 - "GET /metrics HTTP/1.1" 200 OK
2025-09-14T08:04:27.803167951Z INFO:     10.244.0.37:48126 - "GET /metrics HTTP/1.1" 200 OK
```

Model Prediction exposed via FastAPI



Shortcomings

- My development machine was a tad bit underpowered to do this all in Minikube
- Airflow – Debugging the Dag to launch the Prod training Pod in the mlops namespace became very tedious to debug, combined with things being slow to deploy
- Either of Prometheus or Airflow was randomly crashing on my machine in minikube. There was a small window when I was able to successfully get both working and take screenshots of performance, but was not able to proceed further as compute demand increased as I progressed.

Promoting Model from Staging to Production

- Once Staging model is deployed, split traffic between Staging and the actual model in Production
  - Istio can be used to provide fine grained control, but will need to toggle between two services
  - Argo Rollouts to make a process out of this, and manage this better
- Increase the traffic gradually to the Staging model
- Final Cutover, eventually

Managing Airflow (Infrastructure)

- Core Components (dedicated node pool)
  - Webserver
  - Scheduler
  - Triggerer
  - Dag Parser
  - Pgbouncer: replicated 2x, with node anti-affinity
- Task Pods - go to Spot Instance based node pools
- Monitoring Airflow: Apart from the usual container monitoring, resource utilization, and Airflow specific metrics etc, Airflow explicitly requires *Synthetic Monitoring*

Managing Airflow (Execution of Tasks)

Airflow as an Orchestrator tends to attract a lot of abuse

- Disjoint dags – (nothing theoretically wrong in Airflow parlance, mathematically a lot!) Leads to operational problems within Airflow for Operators (example: UI can sometimes break, or difficult to restart a chain/tree of processes because they are disjoint)
- No end date (no business process is meant to run till eternity)
- Lack of Dag metadata (difficult to figure out attribution)
- Too many tasks in the dag– fat dags imply a major failure or infra pressure down the line
- No task specific or priority queue : noisy neighbor problems down the line

https://airflow.apache.org/docs/apache-airflow/stable/administration-and-deployment/cluster-policies.html

Airflow Problems

- Lack of Multitenancy
  - Requires per team instance isolation
  - Fine grain access control can be tricky
- Does not play nice with Istio – not a caveat, per say but something to be aware of nonetheless

MLflow

- The most feature rich ML Tracking service lacking multitenancy
- Lack of Multitenancy can be overcome with custom solutioning with a K8s Operator
  - Deploy a per team registry
  - Shared Postgres instance with isolated schemas
  - Per team artifact roots
  - Endpoints can be secured with some additional overhead

How to do this entire setup for Production

- GitOps using ArgoCD for everything K8s
- IaaC: Terraform/OpenTofu to make infra modules for reusable deployments
- Possibly, Kubeflow instead of Airflow to get multitenancy via Pipeline
- Argo Rollouts for blue/green or canary deployments
- Istio for fine grain networking control and to work with Argo Rollout