MLOps at Snapchat: Continuous Machine Learning with Kubeflow & Spinnaker

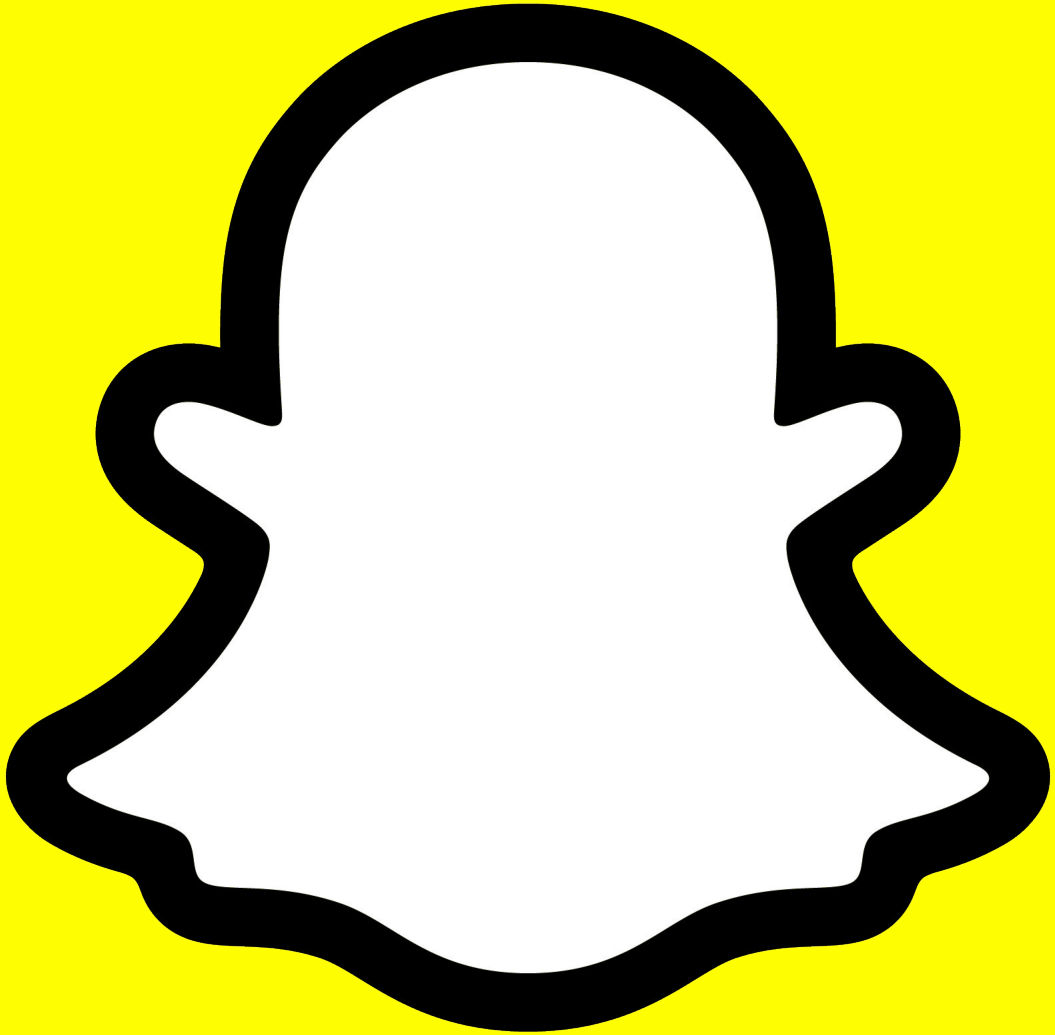
Kevin Dela Rosa

Kevin Dela Rosa

@perhaps_ai

Agenda

- Snapchat
- Anatomy of a production ML system
- MLOps best practices
- Journey to production grade ML



Unlocking lenses

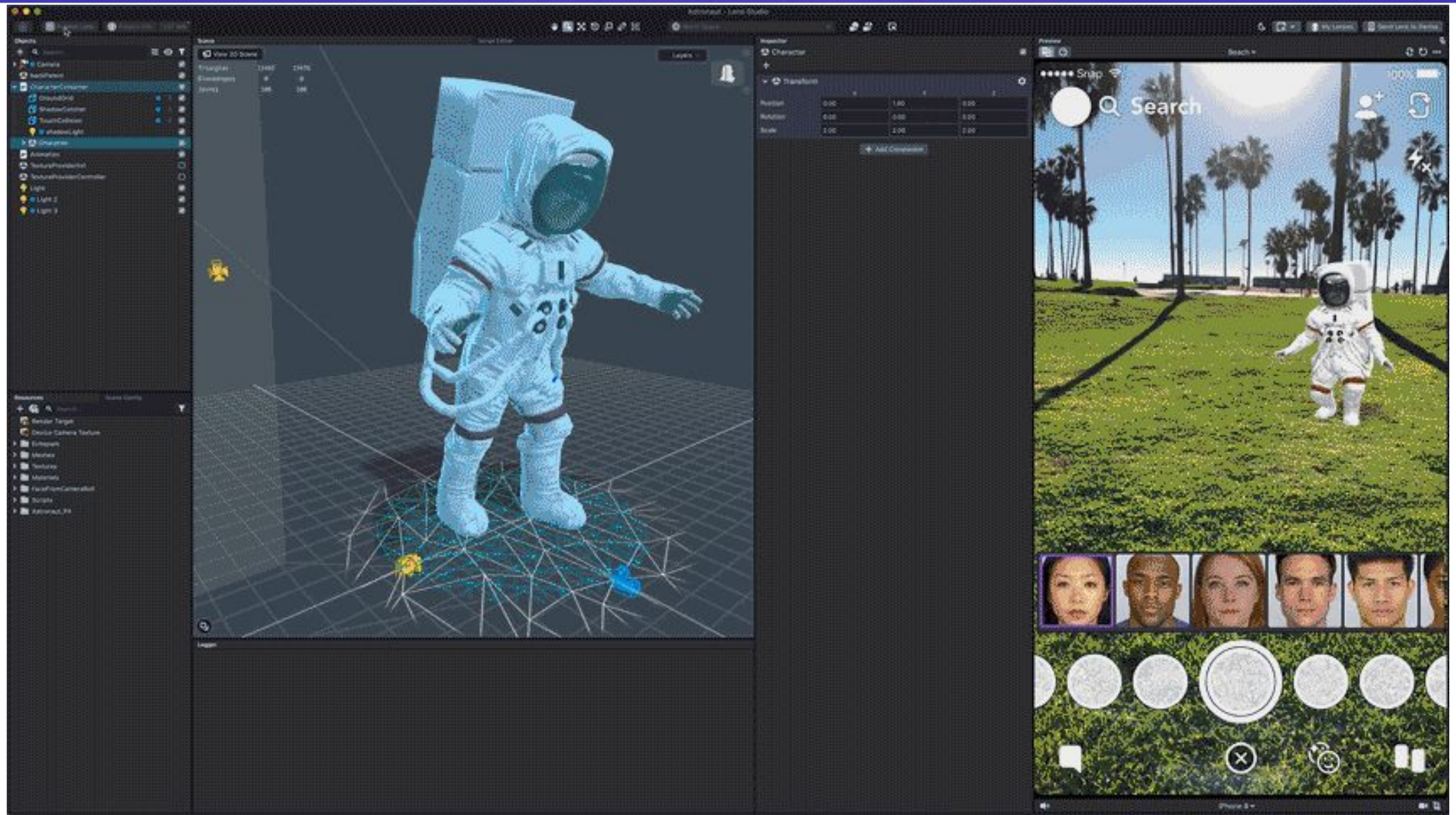
Traditional Mechanisms

- Snapcode
- Lens Link
- Lens Explorer

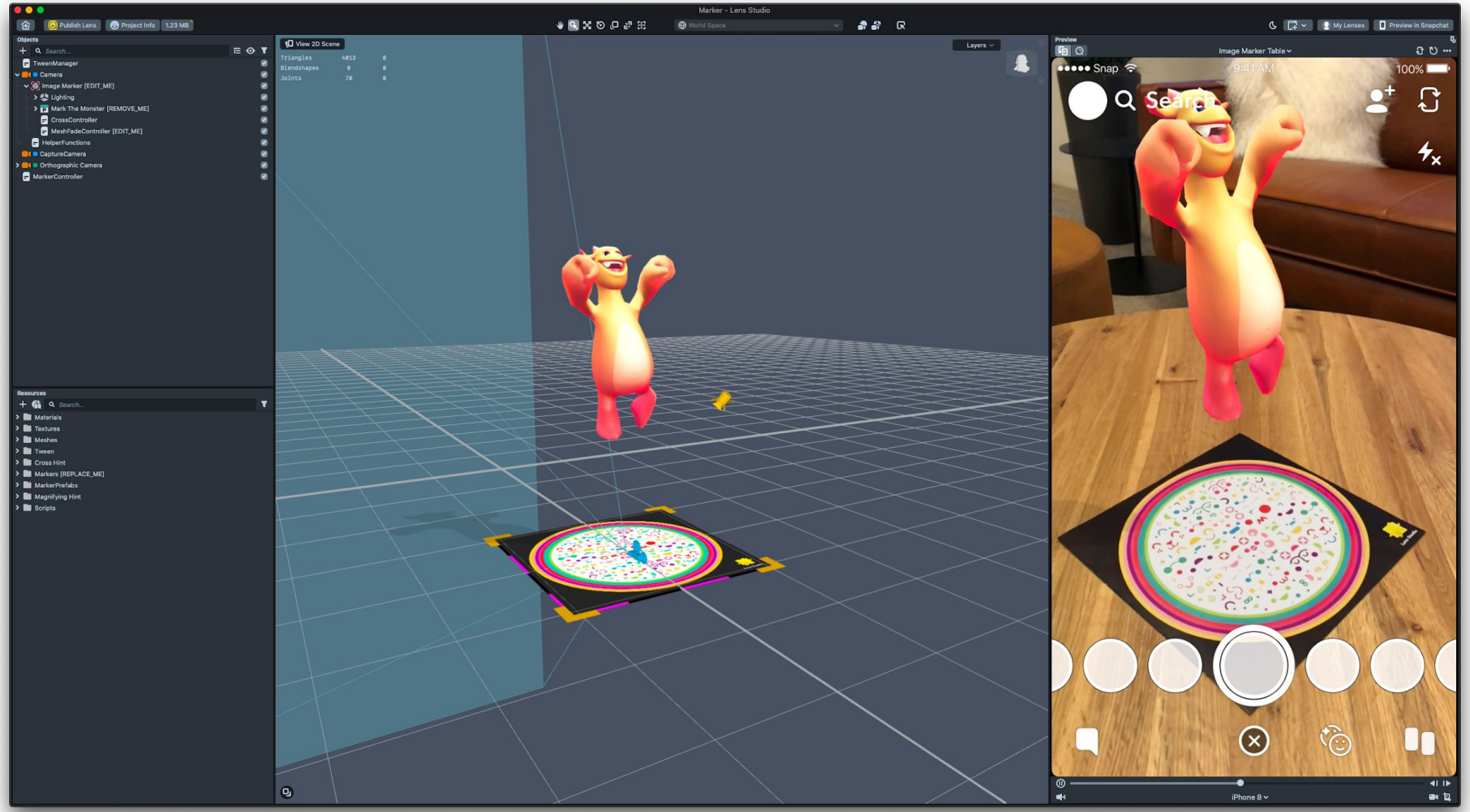
Discovery through Scan

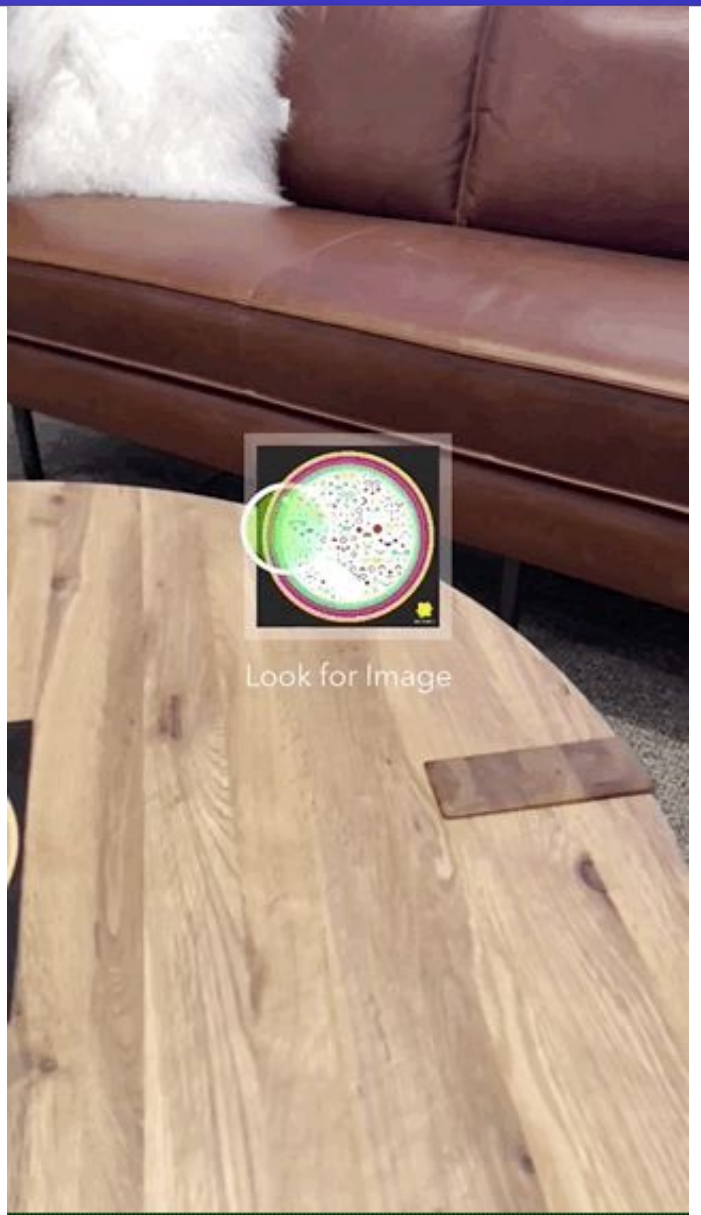
- Scan Triggers
- Marker Images

Scan triggers



Marker lenses





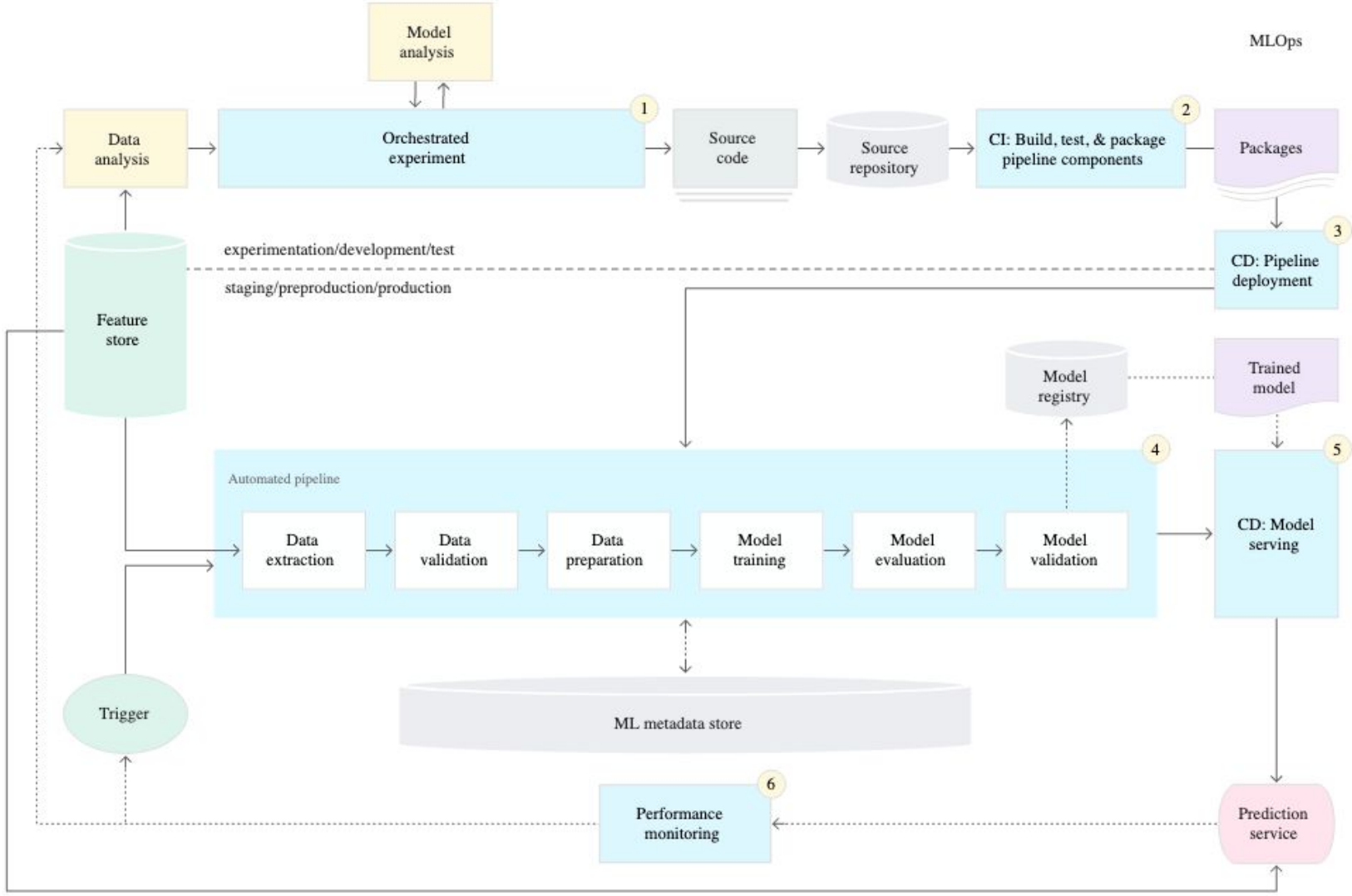
Powered by ML

Machine Learning Models

- Image Classification
- Object Detection
- Semantic Segmentation
- Content Based Information Retrieval
- Nearest Neighbor Search
- Ranking

and more…

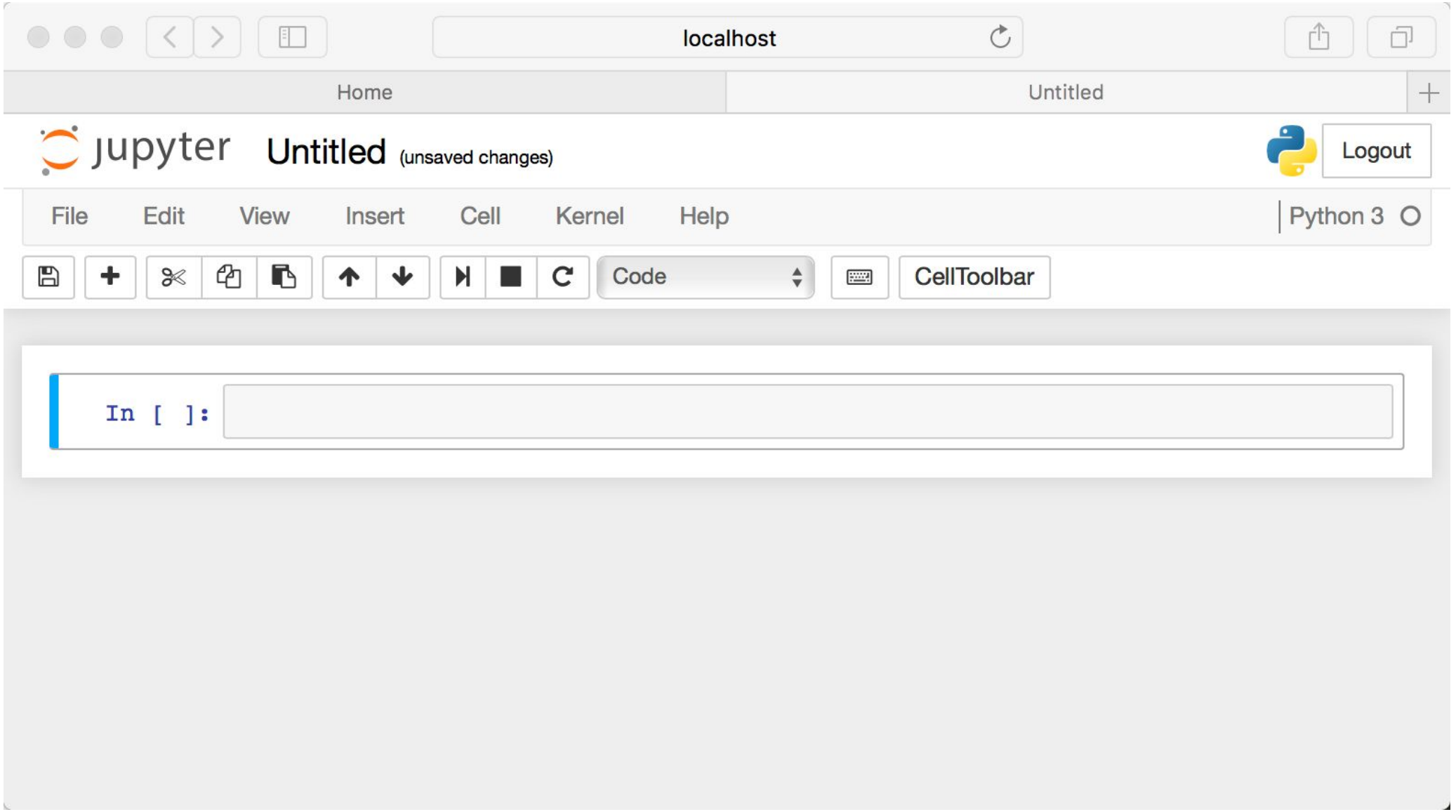
MLOps Setup: CI/CD Automation



From - https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning

Everybody has to start somewhere

KubeCon CloudNativeCon North America 2020 Virtual



This is the way

MLOps = DevOps + ML

Special considerations

- ML is Model + Code + Data
- Data changes

Unique to ML systems

- Data pipelines / ETL
- Feature store
- ML Pipeline

MLOps Principles[1]

- Versioning
- Testing
- Automation
- Reproducibility
- Monitoring

1 - https://ml-ops.org/content/mlops-principles

Journey to production grade ML

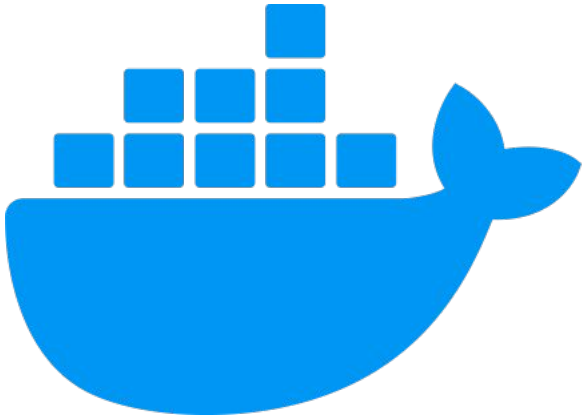
Incremental process, level of automation will increase as your system matures

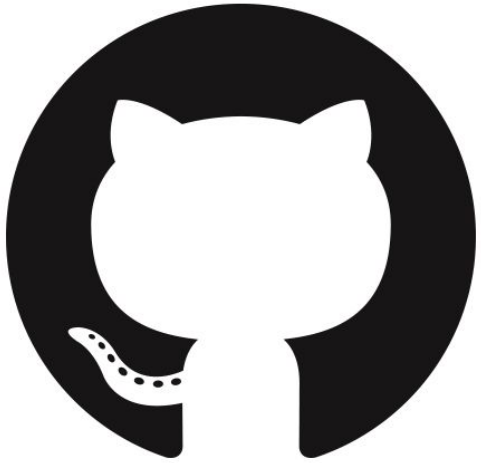
Steps

1. Notebook decomposition, source control
2. ML pipeline automation
3. Continuous Integration
4. Continuous delivery of ML pipeline
5. Continuous training, triggering of ML pipeline
6. Continuous delivery of model / serving
7. Monitoring

1: Notebooks to Containers

Break up monolithic Jupyter notebooks into modular code / containerized programs. Add to source control.





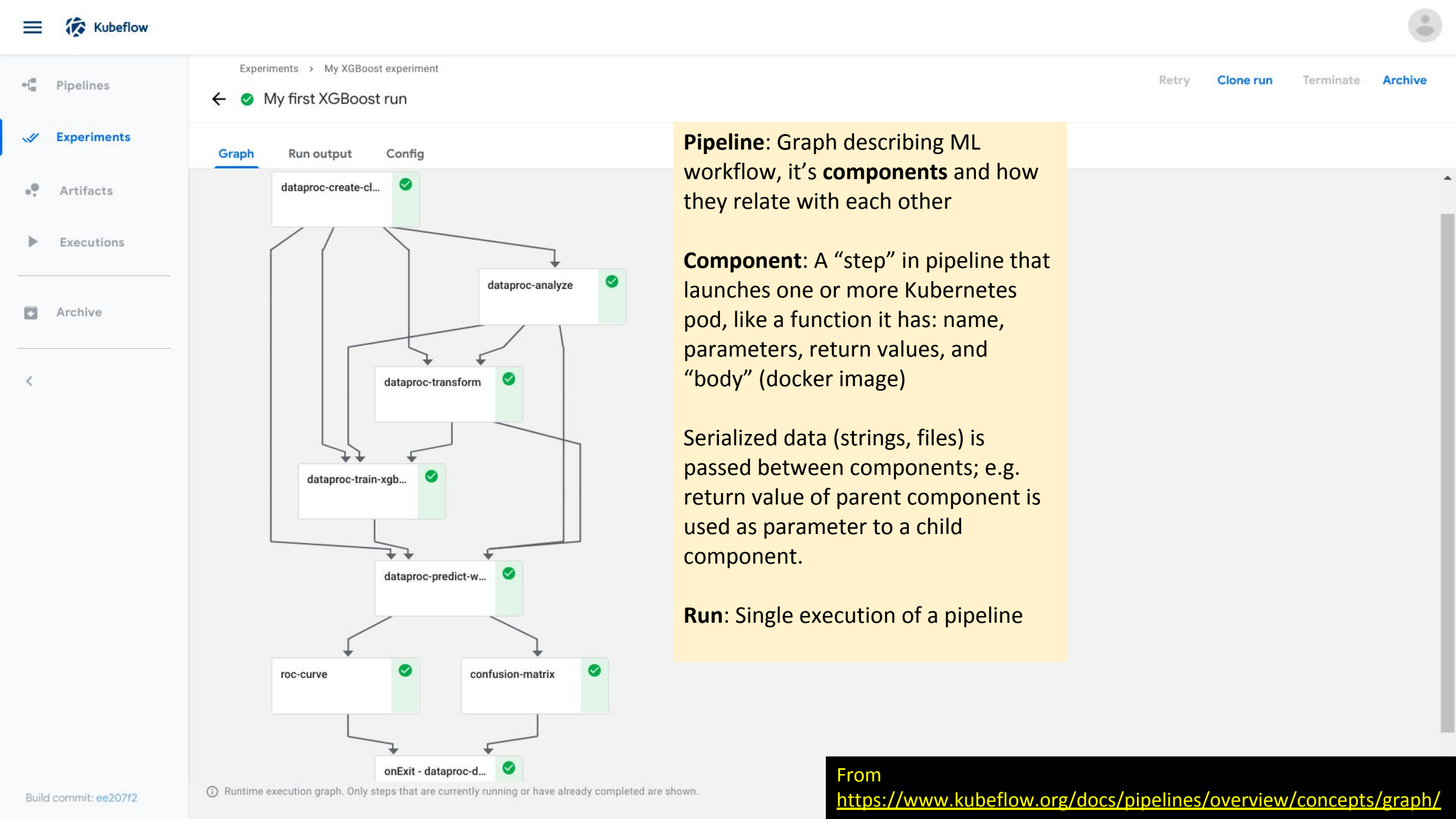
Characteristics

Versioning – track changes to algorithms & transform code
Testing – modular code = easier unit testing
Reproducibility – environment and package dependencies

2: Kubeflow for ML pipeline

Connect Docker images via Kubeflow pipelines to automate the ML pipeline.

Kubeflow



Pipeline: Graph describing ML workflow, it's **components** and how they relate with each other

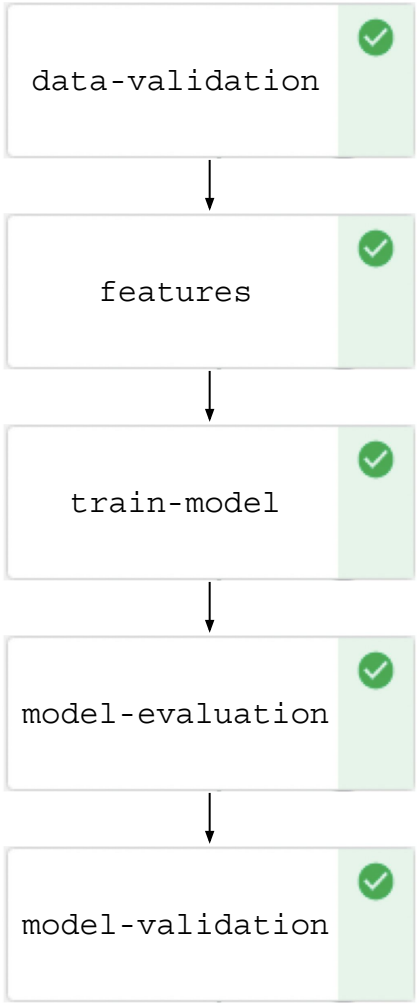
Component: A "step" in pipeline that launches one or more Kubernetes pod, like a function it has: name, parameters, return values, and "body" (docker image)

Serialized data (strings, files) is passed between components; e.g. return value of parent component is used as parameter to a child component.

Run: Single execution of a pipeline

From https://www.kubeflow.org/docs/pipelines/overview/concepts/graph/

2: Kubeflow for ML pipeline



2: Kubeflow for ML pipeline

Characteristics

Automation – automate the steps in your ML process with a pipeline(s)
Reproducibility – store metadata and model information
Versioning – track changes to your ML process, pipeline parameters, hyper parameters & training configuration
Testing – data validation, model evaluation/validation, model spec unit tested, integration test ML pipeline
Monitoring – log model performance on holdout validation set

3: Continuous integration

Automate the building of code, unit tests, and integration tests.

Publish artifacts: docker images, compiled Kubeflow pipeline YAML, Kubernetes configuration

3: Continuous integration

Characteristics

Versioning – publish docker images, Kubeflow component and pipeline specification YAML

Automation – automated unit and integration testing of docker images and ML pipelines

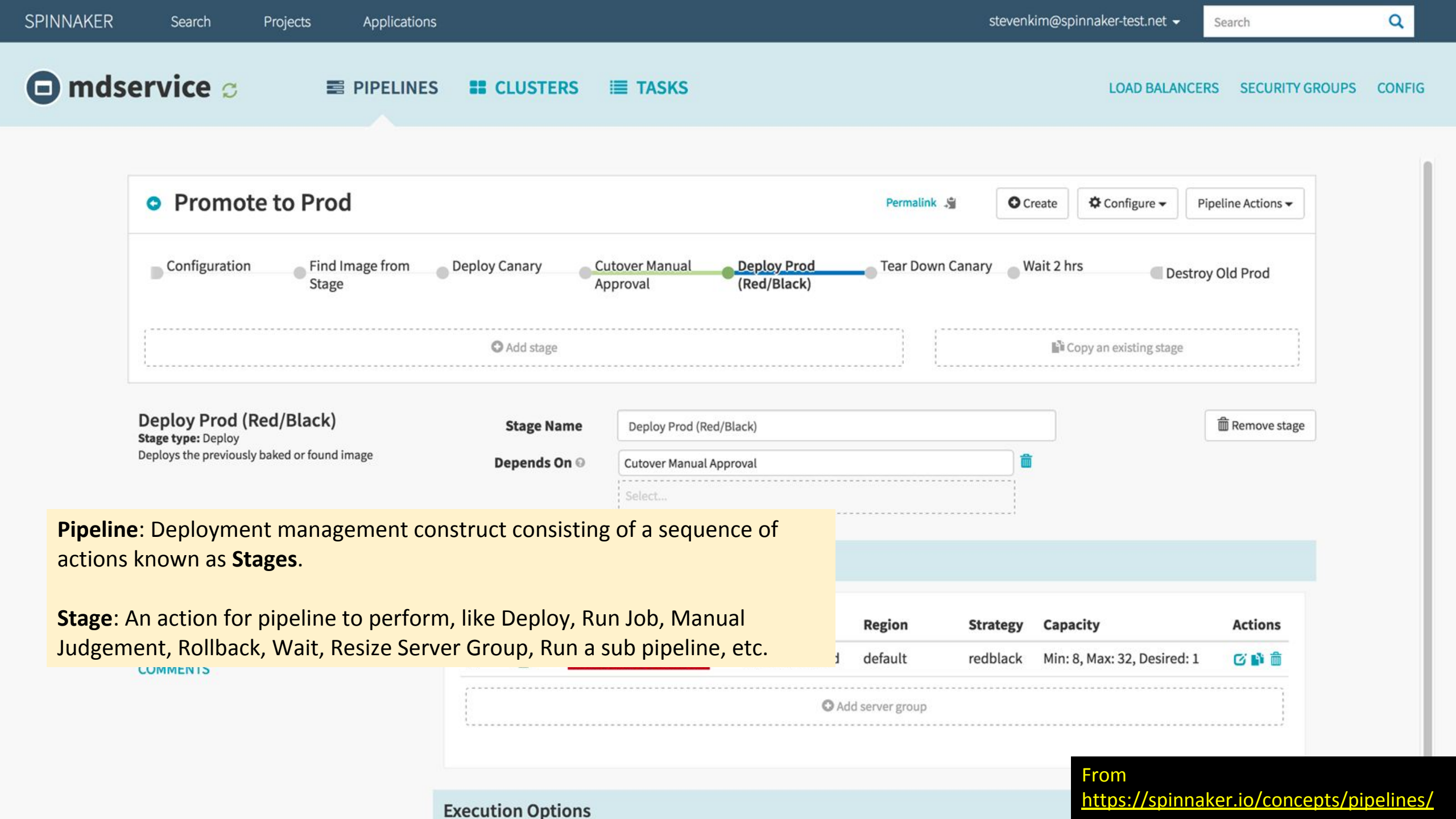
4: Spinnaker for CD of ML Pipeline

Orchestrate the deployment of ML pipeline(s) with a continuous delivery system.

Characteristics

Automation – trigger from CI to automatically react to changes in training process / ML pipeline

Reproducibility – deployment of ML pipeline in repeatable fashion, rollback to previous versions



Pipeline: Deployment management construct consisting of a sequence of actions known as **Stages**.

Stage: An action for pipeline to perform, like Deploy, Run Job, Manual Judgement, Rollback, Wait, Resize Server Group, Run a sub pipeline, etc.

From https://spinnaker.io/concepts/pipelines/

4: Spinnaker for CD of ML Pipeline

Artifacts
- ML pipeline KFP YAML
- Pipeline parameters ConfigMap

Run Job
- Submit KFP job
- Wait for success
- Produce artifact / pass reference to artifact for trained model

More on this in stage 6

5: Continuous training

Trigger ML Pipeline Deployment on a set schedule (e.g. daily, weekly, monthly, etc.) or upon availability of new data.

Characteristics

Automation – training/new model generation process reacts to new data

6: Spinnaker for model delivery

Trigger ML model server deployment upon availability of new model or changes to serving configuration.

Characteristics

Automation – continuous delivery/deployment of new models and/or serving infrastructure

Reproducibility – deployment of model in repeatable fashion, rollback to previous versions

6: Spinnaker for model delivery

Trigger

- Detect new model artifact or server deployment

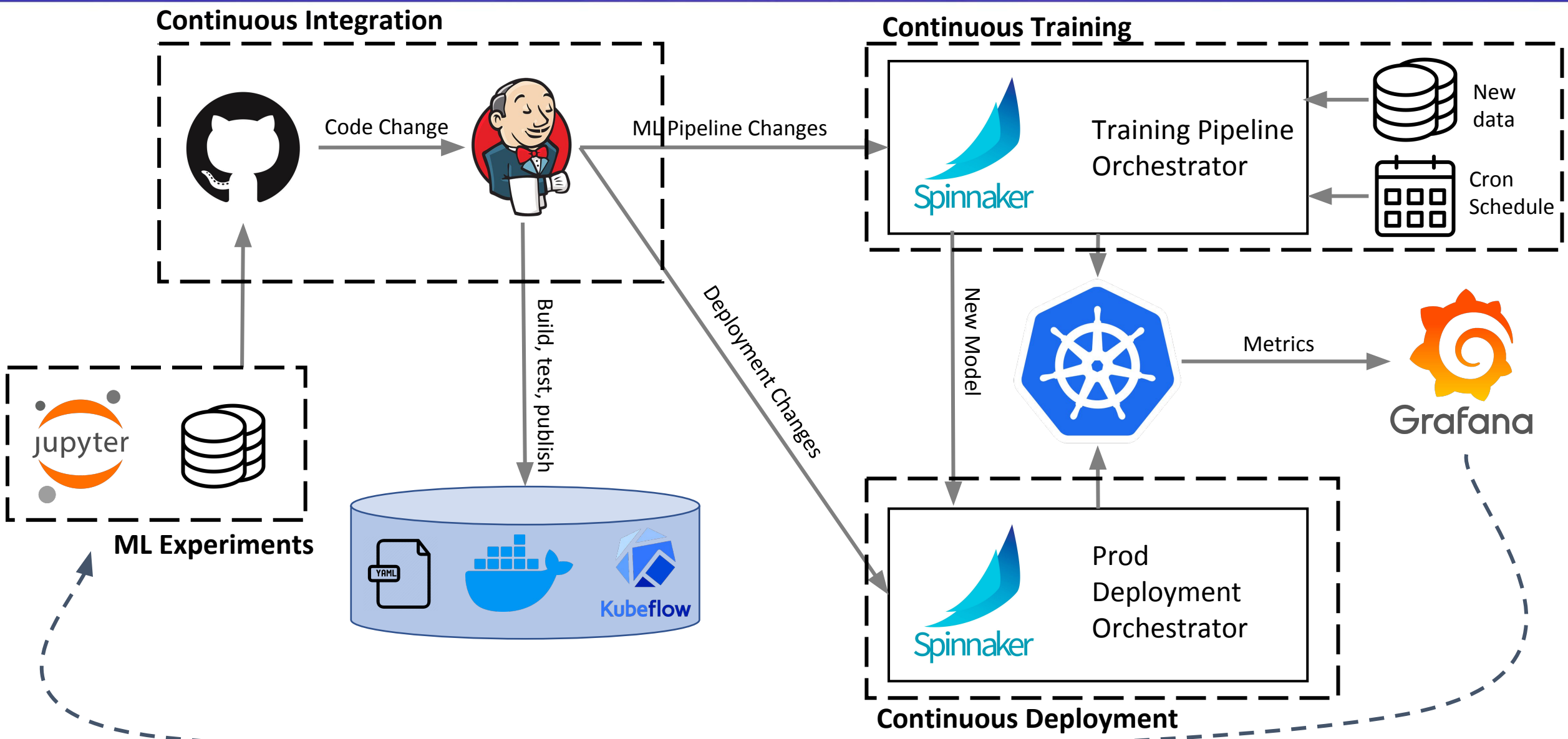
7: Monitoring

Log telemetry about server and model performance. In addition to typical server metrics (latency, traffic, errors, saturation), monitor and alert on unexpected data/prediction changes.

Characteristics

Monitoring – unexpected prediction changes can signal change in user behavior or bad model behavior

“Continuous Machine Learning”



Links

Snapchat

- https://lensstudio.snapchat.com/guides/sharing/scan/
- https://lensstudio.snapchat.com/templates/marker/

MLOps Resources

- https://ml-ops.org/
- https://cloud.google.com/solutions/machine-learning/mlops-continuous-delivery-and-automation-pipelines-in-machine-learning
- https://github.com/cdfoundation/sig-mlops

Tools

- https://www.kubeflow.org/
- https://spinnaker.io/

Kevin Dela Rosa
@perhaps_ai

Thanks for listening!


KEEP CLOUD NATIVE
EVERYWHERE
KubeCon
CloudNativeCon
North America 2020
Virtual