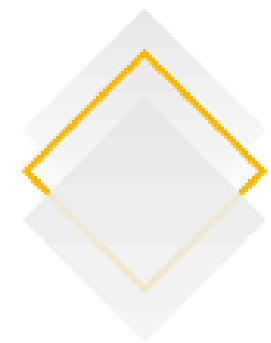
Kedro and MLFlow

---

https://www.youtube.com/watch?v=ZPxuohy5SoU

MLOps • 2022

Kedro



KEDRO

What is

- Kedro is a bridge between machine learning and software engineering.
- Kedro is a template for new data engineering and data science projects

KEY TERMS

Data Catalog

- It is a YAML API for reference datasets. the data sets can be in cloud storage or can be local.
- It makes the datasets declarative, rather than imperative. So all the informations related to a dataset are highly organized

Node

- It is a Python function that accepts input and optionally provides outputs.

Pipeline

- It is a collection of nodes. It is a DAG (Directed acyclic graph).

KEDRO-VIZ

It is a interactive visualization of the entire pipeline. It is a tool that can be very helpful for explaining what you're doing to people.

Experiment Tracking

MLFLOW

- MLFlow can be very helpful in terms of tracking metrics over time. We can visualize that and communicate what is the progress over time.
- MLFlow centralize all of these metrics and also the models generates. So we can have a defines workspace or a share workspace, and we are be able to upload models, artifacts, residuals...

MLFLOW COMPONENTS

There are 3 API

1. mlflow Tracking.
2. mlflow Projects
3. mlflow Models.

Mlflow centralize location for all of the metrics and artifacts.
Models are more on the deployment approach, it's a way to store models and then serve them afterwards

Kedro and MLflow

COMPLEMENTARY AND NOT CONFLICTING

Kedro is the foundation of your data science and data engineering project.
mlflow create that centralized repository of metrics and progress over time .

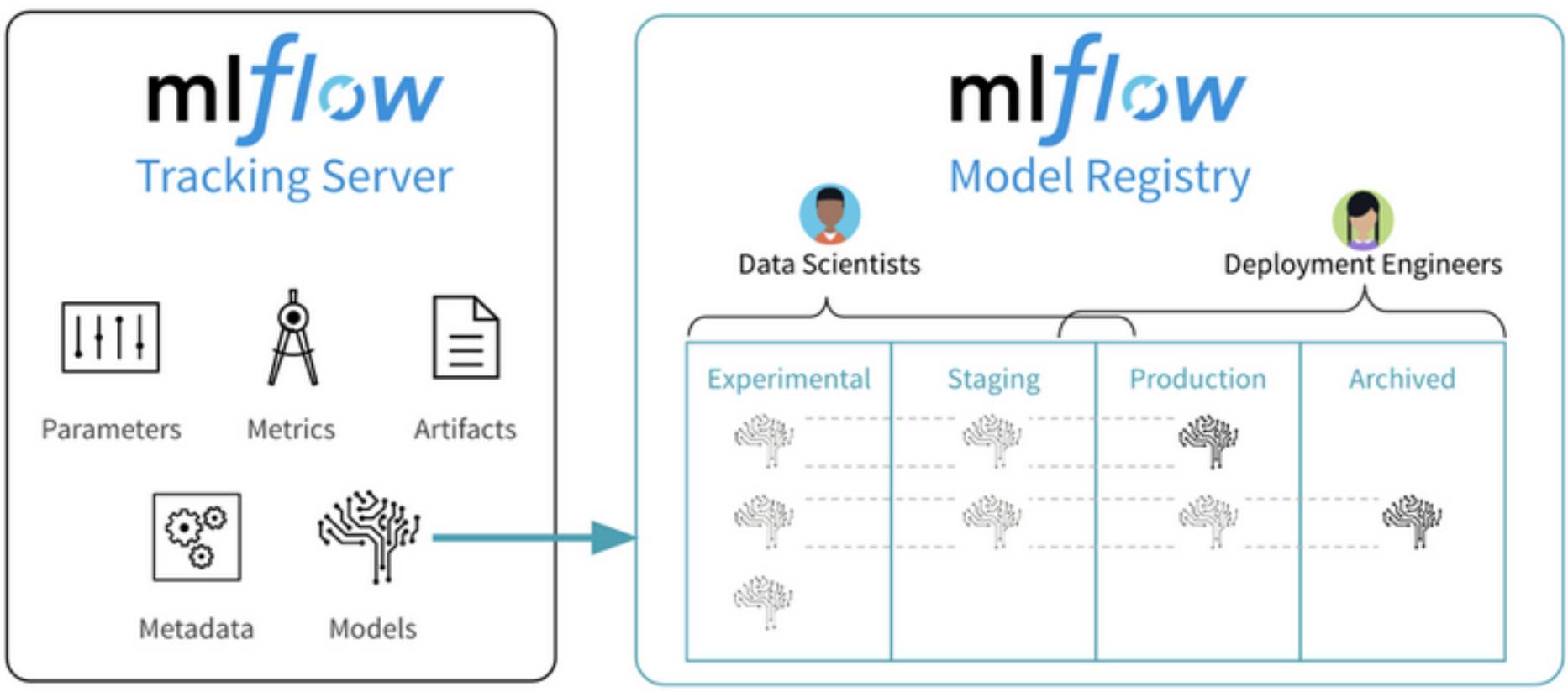
Feature	Kedro	MLflow
Artifact Versioning	Yes	Yes
Metric Tracking	No	Yes
Parameter Versioning	Yes	Yes
Experiment Comparison	No	Yes
Code Organisation	Yes	No
Pipeline Construction	Yes	No
Pipeline Visualisation	Yes	No
Data Abstraction	Yes	No
Deployment	Yes	Yes

MLflow

MODEL REGISTRY

MLflow provides many features including model lineage, model versioning, production to deployment transitions, and annotations.

As Model Registry there are three principal tools: MLflow, Neptune and Azure Machine Learning.
In details: https://neptune.ai/blog/model-registry-makes-mlops-work



Kedro and MLflow

MLFLOW CODES

https://github.com/tgoldenberg/kedro-mlflow-example

- kedro jupyter notebook
    - context.io.list()
    - df = context.io.load('example_iris_data')

https://www.youtube.com/watch?v=fCWGevB366g

https://github.com/mbloem/kedro-mlflow-demo
https://medium.com/quantumblack/deploying-and-versioning-data-pipelines-at-scale-942b1d81b5f5

https://www.youtube.com/watch?v=6z0_n8kxh-g

- mlflow models build-docker

https://www.youtube.com/watch?v=r0do1KVEGqM

https://github.com/TripathiAshutosh/mlflow/blob/main/artifacts/1/6e94445d51104bc89b2aeda4ffbba10f/artifacts/model/python_env.yaml

mlFlow and Bentoml

MLOps • 2022

Bentoml

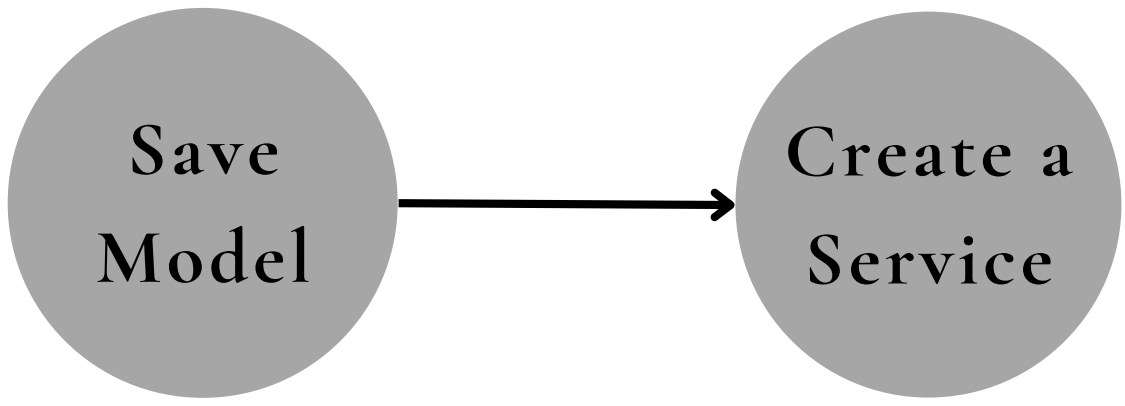
BENTOML

BentoML, on the other hand, focuses on ML in production. By design, BentoML is agnostic to the experimentation platform and the model development environment.
BentoML stores all packaged model files under the ~/bentoml/repository/{service_name}/{service_version} directory by default. The BentoML packaged model format contains all the code, files, and configs required to run and deploy the model.

MODEL MANAGEMENT

By default, save() will save all the BentoService saved bundle files under ~/bentoml/repository/ directory, following by the service name and service version as sub-directory name. And all the metadata of saved BentoService are stored in a local SQLite database file at ~/bentoml/storage.db.

- bentoml list
- bentoml get <name_model>
- bentoml get <name_model>:<num_version>



mlflow and Bentoml

DIFFERENCES

MLFlow provides components that work great for experimentation management, ML project management. BentoML only focuses on serving and deploying trained models.

Main differences:

- MLFlow focuses on loading and running a model, while BentoML provides an abstraction to build a prediction service, which includes the necessary pre-processing and post-processing logic in addition to the model itself.
- BentoML is more feature-rich in terms of serving, it supports many essential model serving features that are missing in MLFlow, including multi-model inference, API server dockerization, built-in Prometheus metrics endpoint and many more
- MLFlow API server requires the user to also use MLFlow's own "MLFlow Project" framework, while BentoML works with any model development and model training workflow - users can use BentoML with MLFlow, Kubeflow, Floydhub, AWS SageMaker, local jupyter notebook, etc

Bento Management API

GET

```
import bentoml
bento = bentoml.get("iris_classifier:latest")
print(bento.tag)
print(bento.path)
print(bento.info.to_dict())
```

LIST

```
import bentoml
bentos = bentoml.list()
```

IMPORT/EXPORT

Bentos can be exported to or import from **AWS S3**, **GCS**, **FTP**, **Dropbox**, etc. For example:
`bentoml.export_bento('my_bento:latest', 's3://my_bucket/folder')`

```
import bentoml
bentoml.export_bento('my_bento:latest',
'/path/to/folder/my_bento.bento')
bentoml.import_bento('/path/to/folder/my_bento.be
nto')
```

Deploying Bentos

OVERVIEW

What is a Bento?
Bento is a file archive with all the source code, models, data files and dependency configurations required for running a user-defined bentoml.Service, packaged into a standardized format.

The three most common deployment options with BentoML are:

- Generate container images from Bento for custom docker deployment
- Yatai: Model Deployment at scale on Kubernetes
- bentoctl: Fast model deployment on any cloud platform

https://docs.bentoml.org/en/latest/concepts/deploy.html

CONTAINERIZE BENTOS

Containerizing bentos as Docker images allows users to easily distribute and deploy bentos.

Steps:

- Run **bentoml list <name_bento>:latest** to view available bentos in the store.
- Run **bentoml containerize** to start the containerization process.
- Run the generated docker image: example
  - **docker run -p 3000:3000 iris_classifier:ejwnswg5kw6qnuqj**

Deploying Bentos

DEPLOY WITH YATAI

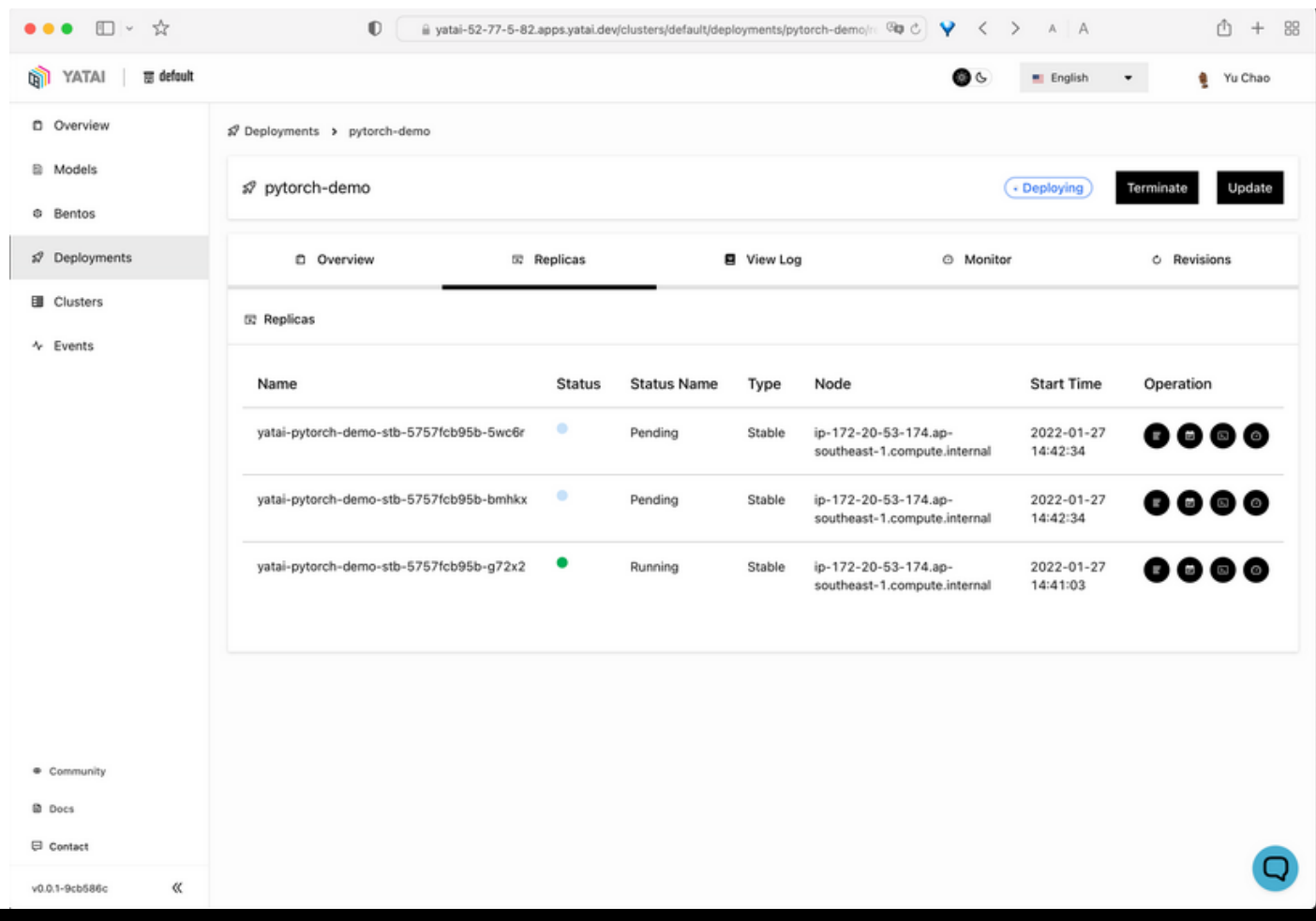
Yatai helps ML teams to deploy large scale model serving workloads on **Kubernetes**. It standardizes BentoML deployment on Kubernetes, provides UI and APIs for managing all your ML models and deployments in one place.

Steps:

- Get an API token from Yatai Web UI and login from your bentoml CLI command:
  - **`bentoml yatai login --api-token {YOUR_TOKEN_GOES_HERE} --endpoint http://yatai.127.0.0.1.sslip.io`**
- Push your local Bentos to yatai:
  - **`bentoml push iris_classifier:latest`**

DEPLOY VIA WEB UI

Yatai offers an easy-to-use web UI for quickly creating deployments.



Deploying Bentos

DEPLOY WITH BENTOCTL

bentoctl is a CLI tool for deploying Bentos to run on any cloud platform. It supports all major cloud providers, including AWS, Azure, Google Cloud, and many more.

Steps:

- Install aws-lambda plugin for bentoctl (for example)
  - **`bentoctl operator install aws-lambda`**
- Initialize a bentoctl project.
  - **`bentoctl init`**
- build the deployable artifacts required for this deployment.
  - **`bentoctl build -b iris_classifier:btzv5wfv665trhcu -f ./deployment_config.yaml`**
- use terraform CLI command to apply the generated deployment configs to AWS.
  - **`terraform init`**
  - **`terraform apply -var-file=bentoctl.tfvars --auto-approve`**

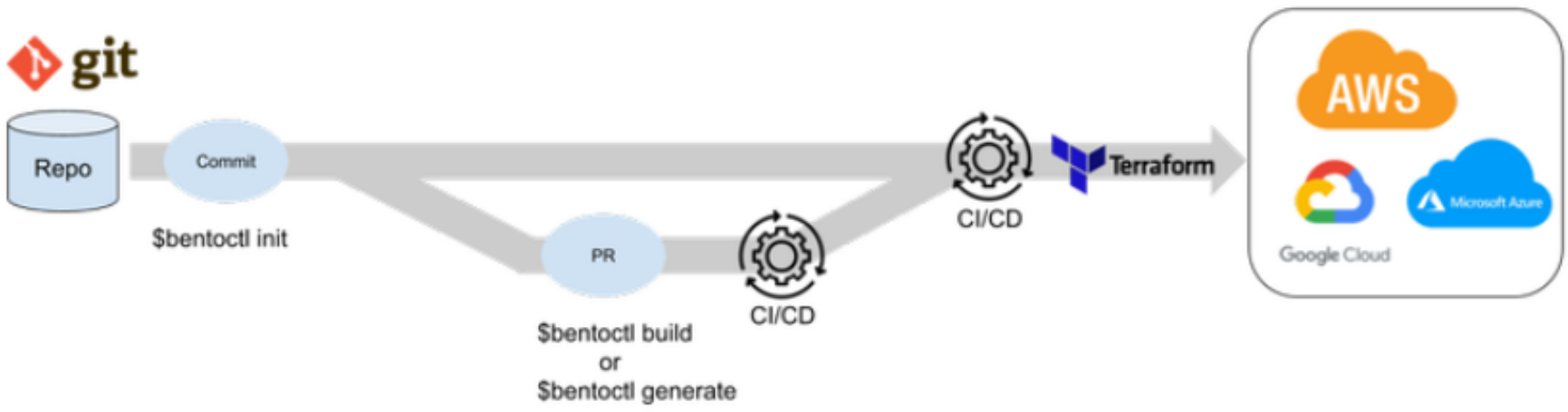
Bentoctl and GitOps workflows

FOR GITOPS WORKFLOWS

- Getting started with Bentoctl is as easy as running the command `**bentoctl init**`.
    - This will kick off an interactive process that generates the standard configuration and necessary Terraform project files.
- Once you've generated your Terraform scaffolding, the `**bentoctl build**` command packages your prediction service according to the location you're deploying to.
    - This step automatically updates the Terraform's variable file with the newly-built image information.
- If you ever need to make adjustments to deployment resources, you can easily update the Bentoctl configuration file and run `**bentoctl generate**`

The bentoctl and Yatai deployment workflow is optimized for CI/CD and GitOps.

https://modelserving.com/blog/introduction-to-bentoctl



References

Bentoml:
https://docs.bentoml.org/en/latest/integrations/mlflow.html

Differences:
https://docs.bentoml.org/en/0.13-lts/faq.html

Together:
https://docs.bentoml.org/en/latest/integrations/mlflow.html
https://github.com/bentoml/BentoML/tree/main/examples/mlflow/pytorch

Deploying:
https://docs.bentoml.org/en/latest/concepts/deploy.html