DVC vs Kedro

2022 • GIORGIA BERTACCHINI

MLOps • 2022

DVC (Data Version Control) and kedro

DATA VERSION CONTROL

What is

- It takes on a Git-like model to provide management and versioning of datasets and machine learning models. DVC is a simple command-line tool that makes machine learning projects shareable and reproducible.

KEDRO

What is

- Kedro is an open-source Python framework for creating reproducible, maintainable and modular data science code.

DATA

data/data.xml.dvc

- DVC stores information about the added file in a special .dvc file named data/data.xml.dvc, this metadata file is a placeholder for the original data.

DATA

conf/base/catalog.yml

- Data Catalog, which is the registry of all data sources available for use by the project.

directory /data

- where the data are divided during project.
- where are saved also models, plot and other created.

DVC (Data Version Control) and Kedro

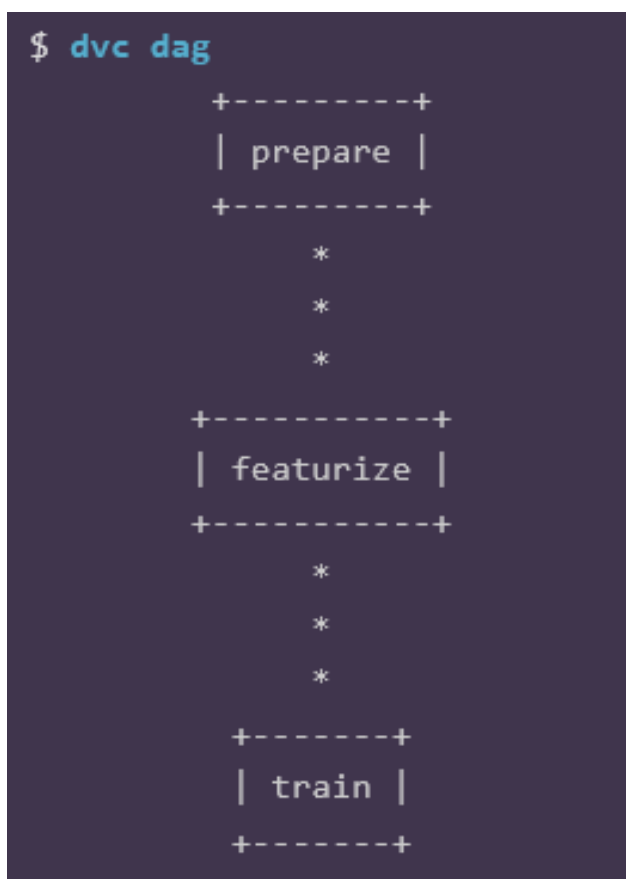
PIPELINE

dvc.yaml file

- It includes information about the steps of pipeline, with dependencies and outputs, and concatenate the nodes od pipeline

command: dvc dag

- to visualize the pipeline structure.



PIPELINE

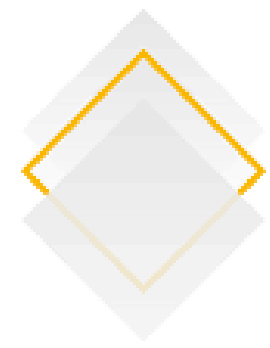
src/name_kedro_project/pipelines/name_pipeline/pipeline.py file

- It includes information about the steps of pipeline, with dependencies and outputs, and concatenate the nodes od pipeline

command kedro viz

- this command should open up a visualisation in your browser
- to visualize the pipeline structure and other informations

DVC (Data Version Control) and Kedro



METRICS

DVC makes it easy to track metrics, and visualize performance with plots.

command: dvc run

- specifing node of pipeline and dependencies, create in output plots and a file with metrics.

command show diff

- show difference through metrics different, for example metrics of different branches

METRICS

command kedro viz

- to visualize same data, for example MetricsDataSet and PlotlyDataSet and other informations

EXPERIMENTS

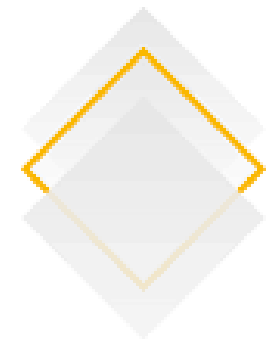
DVC can track the experiments, list and compare their most relevant metrics, parameters.

EXPERIMENTS

command kedro viz

- Experiment tracking in Kedro-Viz also supports the display of plots, such as Plotly and Matplotlib, and other results from all experiments.

DVC (Data Version Control) and Kedro



PARAMETERS

params.yaml

- DVC can track parameters, that can be any values used inside your code to influence the results.

command: dvc params diff

- Show changes in dvc params between commits in the DVC repository

PARAMETERS

parameters/name_pipeline.yaml

- where are write parameters, that can be any values used inside your code to influence the results.

PLOTS

DVC have a set of commands to create, visualize and compare data sets.

PLOTS

command kedro viz

- Kedro-Viz show the plot of data in output of PlotlyDataSet and Matplotlib nodes.

DVC (Data Version Control) and Kedro

DVC ON VS CODE

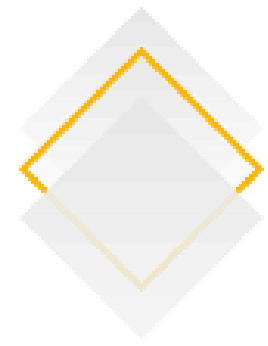
There are a DVC extension, which brings a full machine learning experimentation platform to Visual Studio Code. with this extension in VisualCode can have Interactive plots, Live tracking and Experiment bookkeeping.

More: https://iterative.ai/blog/DVC-VS-Code-extension

KEDRO-VIZ

The same feature of DVC extension for Visual Studio Code are also in the browser opened by command "kedro viz".

DVC (Data Version Control) and Kedro



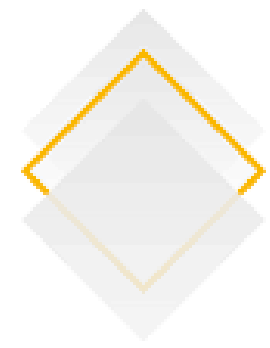
CONCLUSION

DVC and Kedro are two tools very similar.

But

- DVC work very well with GitHub actions, because more of features are based on command-lines. This allows easy comparisons between branches of a GitHub project.
- Kedro-Viz open a browser page with all pipeline, that include node and input/output data. For all these is write the corrispective path and command-line for show or run. Kedro-Viz show in a easy way also plot, metrics and show experiments history.

DVC (Data Version Control) and Kedro



From: https://medium.com/y-data-stories/creating-reproducible-data-science-workflows-with-dvc-3bf058e9797b

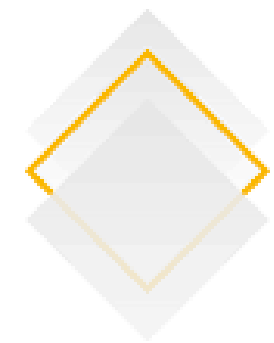
> DVC is not the only tool for the job. It works best for small to middle-sized projects and solves the problem without adding too much complexity. However, depending on your needs, project size and deployment considerations you may find Kedro or other tools more suitable. We will cover some of them in future tutorials.

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 .

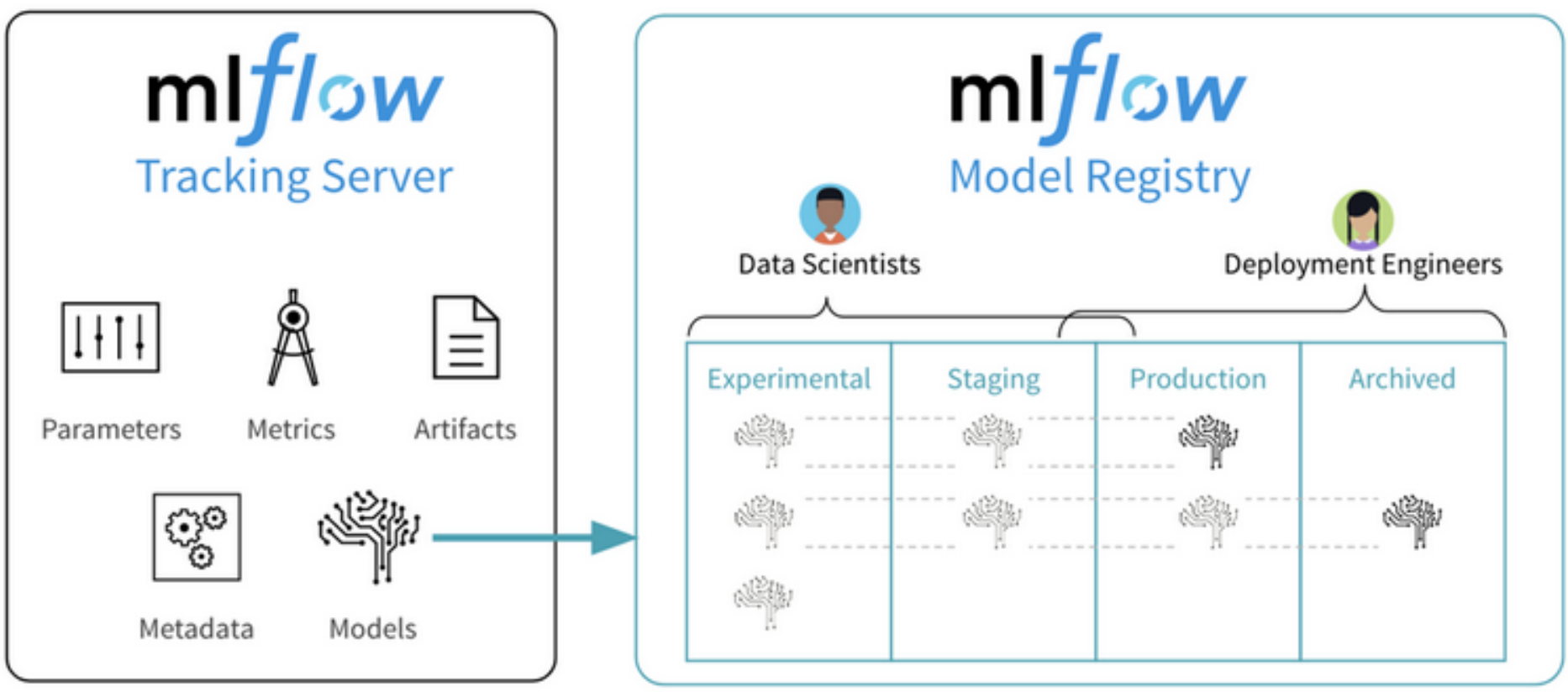
Feature	Kedro	MLflow
Artifact Versioning	Yes	Yes
Metric Tracking	No	Yes
Parameter Versioning	Yes	Yes
Experiment Comparison	No	Yes
Code Organisation	Yes	No
Pipeline Construction	Yes	No
Pipeline Visualisation	Yes	No
Data Abstraction	Yes	No
Deployment	Yes	Yes

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

For save mlflow experiments in a specific folder or database:

- https://www.mlflow.org/docs/latest/python_api/mlflow.html
- https://github.com/dmatrix/google-colab/blob/master/mlflow_issue_3317.ipynb

MLflow and Bentoml

MLOps • 2022

Bentoml

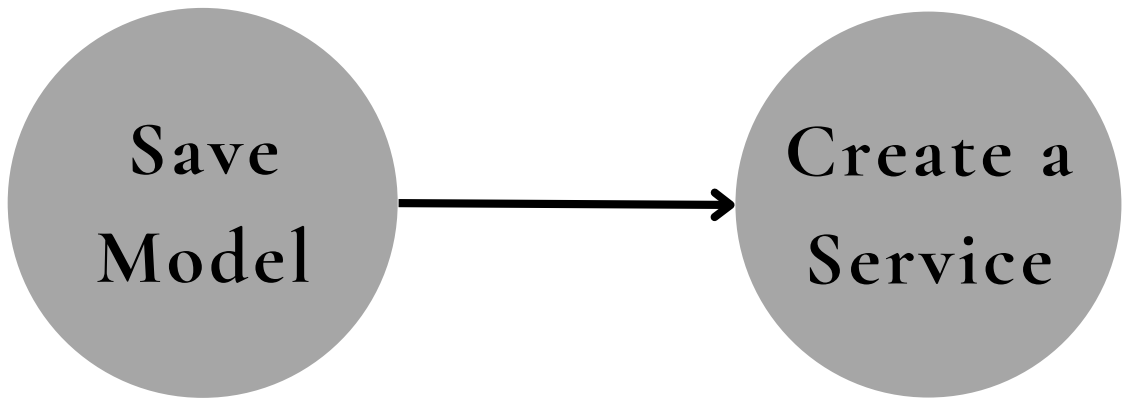
BENTOML

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 <name_bento>:latest**
- to start the containerization process.
- Run the generated docker image: example
  - **docker run -p 3000:3000 iris_classifier:ejwnswg5kw6qnuqj**

Deploying Bentos

DEPLOY WITH YATAI

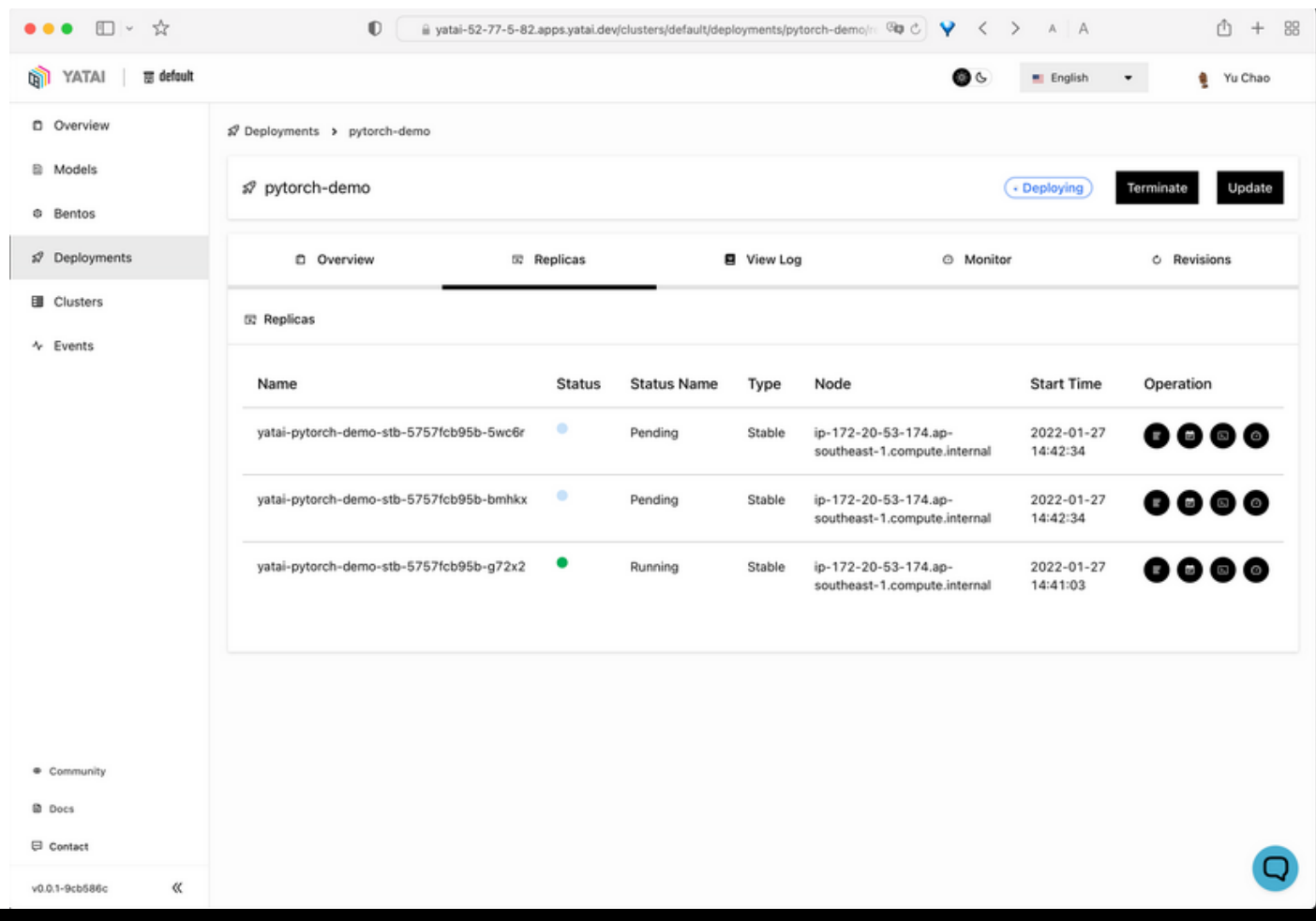
Yatai helps ML teams to deploy large scale model serving workloads on **Kubernetes**. It standardizes BentoML deployment on Kubernetes, provides UI and APIs for managing all your ML models and deployments in one place.

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`**

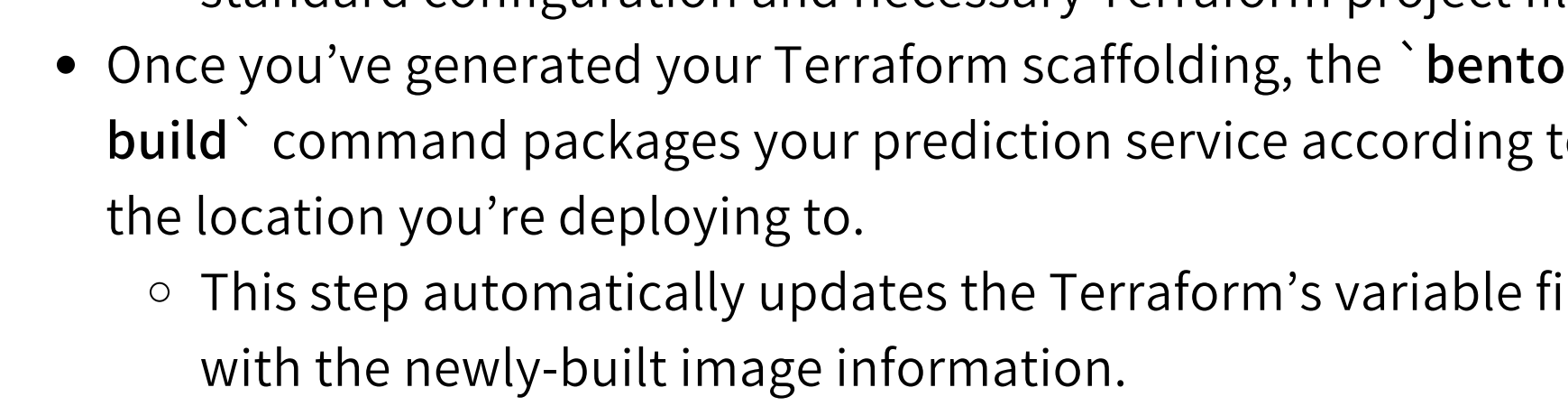
Bentoctl and GitOps workflows

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

- https://kedro.readthedocs.io/en/stable/index.html
- https://dvc.org/doc/start
- https://iterative.ai/blog/DVC-VS-Code-extension

- https://docs.bentoml.org/en/latest/integrations/mlflow.html
- https://docs.bentoml.org/en/0.13-lts/faq.html

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

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

Deploy Kedro

2022 • GIORGIA BERTACCHINI

MLOps • 2022

Deploy Kedro

DEPLOY KEDRO ON A PRODUCTION SERVER

There are three alternative methods to deploy your Kedro pipelines:

- Container-based using **Kedro-Docker**
- Package-based using **kedro package**
- CLI-based using the **Kedro CLI**

1. CONTAINER-BASED

Step:

- Use **Kedro-Docker plugin** (https://github.com/kedro-org/kedro-plugins/tree/main/kedro-docker) to streamline the process
- After you've built the Docker image for your project locally, transfer the image to the production server. You can do this as follows:
  - `docker tag <image-name> <DockerID>/<image-name>`
  - `docker push <DockerID>/<image-name>`
  - `docker pull <DockerID>/<image-name>`

Deploy Kedro

---

2. PACKAGE-BASED

Step:

- `kedro package`
    - Kedro builds the package into the **dist**/ folder of your project, and creates one **.egg** file and one .**whl** file, which are Python packaging formats for binary distribution.
    - The resulting package only contains the Python source code of your **Kedro pipeline**, not any of the **conf**/, **data**/ and **logs**/ subfolders
- After having installed your project on the remote server, run the Kedro project as follows from the root of the project:
    - `python -m project_name`

3. CLI-BASED

Step:

- Cloning your project codebase to the server.
    - `git clone <repository>`
- Install the project's dependencies, by running the following in the project's root directory:
    - `pip install -r src/requirements.txt`
- And run the pipeline:
    - `kedro run`

References

- https://kedro.readthedocs.io/en/stable/deployment/single_machine.html
- https://github.com/kedro-org/kedro-plugins/tree/main/kedro-docker