Transforming Machine Learning Workflows with ZenML

Karan Sharma
06 August, 2024

Agenda

- MLops
- ZenML
- ZenML for MLops Platform Engineer
- ZenML for Data Scientist
- Core Components of ZenML
- ZenML with other tools
- Demo
- Conclusion and Q&A

What is MLops?

- **Definition**: MLOps integrates ML system development and operations to deploy and maintain machine learning models in production reliably and efficiently.

Benefits

- Reproducibility
- Scalability
- Collaboration
- Compliance

ZenML

- Pipeline based MLops framework for developing and deploying as production level.
- Separate Infrastructure code from development code which eases collaboration from different teams.

ZenML for MLops Platform Engineer

Define, Deploy, and Manage Production Environments

- ZenML enables MLOps experts to create sophisticated production environments.
- Shareable environments facilitate collaboration within teams.

```
# Deploy ZenML to any cloud
zenml deploy --provider aws

# Connect cloud resources with a simple wizard
zenml stack register <STACK_NAME> --provider aws

# Deploy entire MLOps stacks at once
zenml stack deploy  --provider gcp
```

Self-Hosted Deployment

- **Cloud Flexibility**: Deploy ZenML on any cloud provider.
- **Terraform Utilities**: Deploy additional MLOps tools or entire stacks.

ZenML for MLops Platform Engineer

Standardize MLOps Infrastructure

- Register staging and production environments as ZenML stacks.
- Colleagues can run ML workflows on standardized stacks.

Avoid Vendor Lock-In

- Infrastructure is decoupled from code, allowing for easy transitions.
- Switch tooling stacks to suit performance and pricing needs.

ZenML for Data Scientist

Develop Locally

- **Flexibility**: Develop ML models in any environment using favorite tools.
- **Seamless Transition**: Easily switch to production environments without code changes.

```
python run.py  # develop your code locally with all your favorite tools
zenml stack set production
python run.py  # run on production infrastructure without any code changes
```

ZenML for Data Scientist

ZenML's Pythonic SDK

Unintrusive Integration

- **Simple Decorators**: Use `@step` or `@pipeline` decorators to convert Python functions into ZenML pipelines.
- **Minimal Changes**: Easily integrate ZenML into existing codebases.

Automatic Metadata Tracking with ZenML

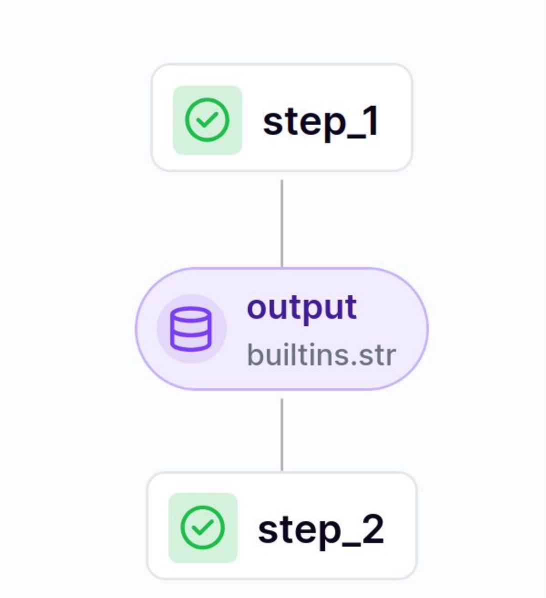
Comprehensive Tracking

- **Automatic Metadata**: Tracks all run metadata, datasets, and models.
- **Version Control**: Saves and versions datasets and models automatically.

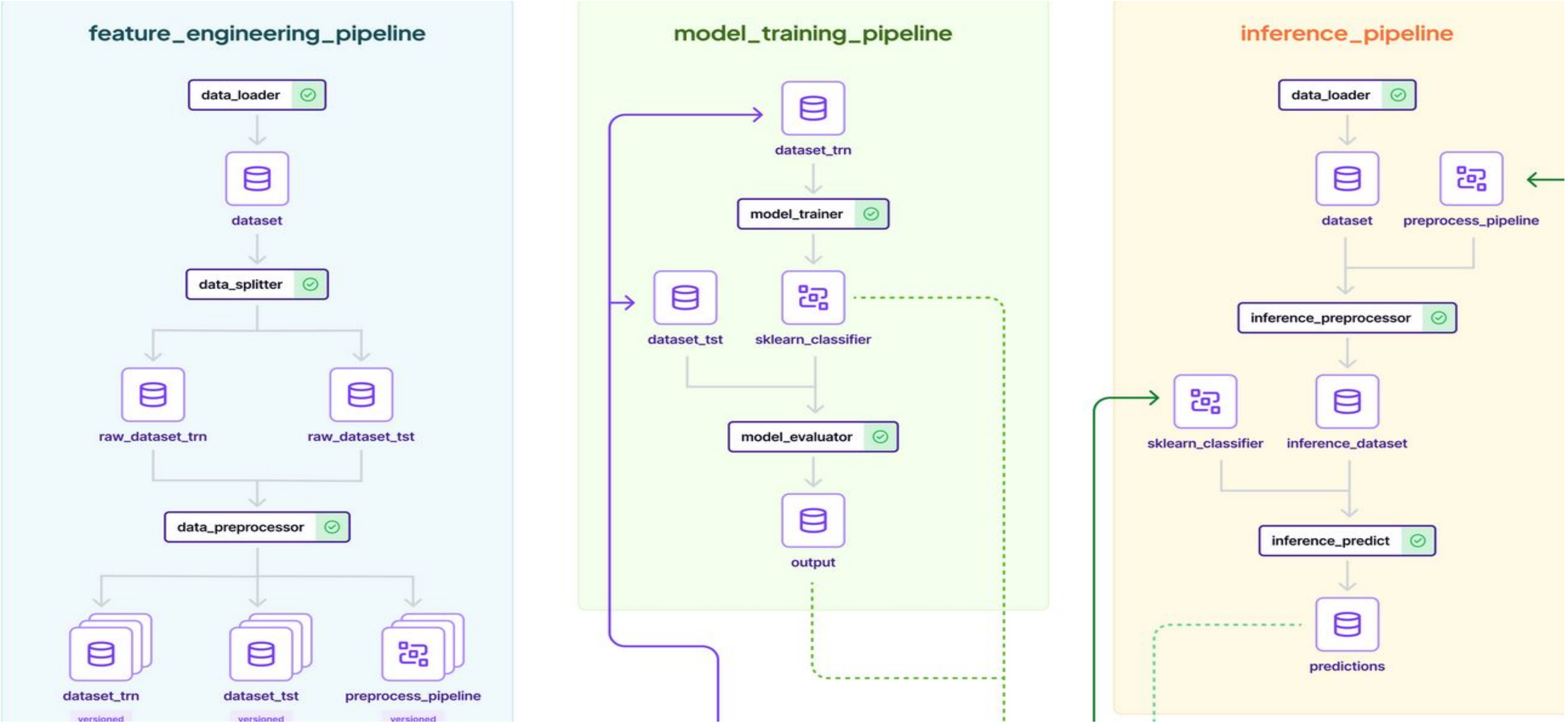
ZenML for Data Scientist

```
from zenml import pipeline, step

@step
def step_1() -> str:
  return "world"

@step
def step_2(input_one: str, input_two: str) -> None:
  combined_str = input_one + ' ' + input_two
  print(combined_str)

@pipeline
def my_pipeline():
  output_step_one = step_1()
  step_2(input_one="hello", input_two=output_step_one)

my_pipeline()
```

My pipeline()



ZenML Sample code project workflow



ZenML Stack component

Data Versioning Component

- Tracking changes in data, ensuring data availability, and maintaining data integrity.
- **Tools**:
  - DVC (Data Version Control)
  - Neptune
  - Concepts like Feature Store (e.g., Feast)

Feature Store

- Manages the transformation of raw data into features for training models.
- Can inherently include data versioning capabilities so we can use Feature Store Container in place of Data versioning component.

ZenML Stack component

Experiment Tracker

- Tracks hyperparameters, model versions, outputs, and experiment configurations.
- **Tools**:
  - MLflow
  - Weights and Biases

Compute Environment Component

- Provides the necessary computational resources (CPU, GPU) for model training.
- **Tools**:
  - Kubernetes clusters
  - Azure ML
  - AWS SageMaker

ZenML Stack component

Model Deployer

- Responsible for serving the model in real-time or batch mode.
- Supports pre and post-processing and integrates with model registries.
- **Tools**:
  - Seldon Core
  - KFServing (KServe)

Monitoring Component

- Monitors the deployed models for data quality, infrastructure health, and model performance.
- **Tools**:
  - Prometheus
  - Grafana
  - TensorBoard

ZenML Stack component

Artifact Store

- Central storage for all artifacts (models, datasets, experiment results) used in the MLOps workflow.
- Can be implemented using cloud storage solutions like S3, Azure Blob Storage, Google Cloud Storage.

Orchestrator

- Manages the execution and coordination of different components in the MLOps pipeline.
- Can be run locally or on the cloud (e.g., Kubernetes).

ZenML with Feast (feature store)

- Feast (Feature Store) is an operational data system for managing and serving machine learning features to models in production.
- There are two core functions that feature stores enable:
  - access to data from an offline / batch store for training.
  - access to online data at inference time.

```
zenml integration install feast
```

```
zenml feature-store register feast_store --flavor=feast
zenml stack register ... -f feast_store
```

ZenML with MLflow (experiment tracker)

- Automatically track experiments in your experiment tracker with **MLflow**.

```
@step(experiment_tracker="mlflow")
def read_data_from_snowflake(config: pydantic.BaseModel) -> pd.DataFrame:
  df = read_data(client.get_secret("snowflake_credentials")
  mlflow.log_metric("data_shape", df.shape)
  return df
```

ZenML with MLflow

- Track model metadata and lineage with **MLflow**.

```
@step(
  settings={"resources": ResourceSettings(memory="240Gb") }
  model=Model(name="my_model", model_registry="mlflow")
)
def my_trainer(df: pd.DataFrame) -> transformers.AutoModel:
  tokenizer, model = train_model(df)
```

Demo

Conclusion

- ZenML is pipeline based MLops workflow framework.
- ZenML integrates seamlessly with many popular open-source tools.
- ZenML separates infrastructure code with ML development code.

Discussion/Q&A