Quick Guide to ZenML Orchestration Setup

# What is ZenML Orchestration?

ZenML Orchestration automates the workflow of ML pipelines, managing how and where each step, such as data preprocessing and training, is executed.

# Why Use Orchestration?

Using Orchestration provides several benefits:

**- Manage Complexity:** Easier handling of complex ML workflows. **- Scalability:** Efficiently scales pipeline execution for larger datasets or computations. **- Reproducibility:** Ensures consistent execution for repeatable experiments. **- Resource Optimization:** Better use of computational resources, crucial for cloud or on-premise environments.

# Setting Up ZenML Orchestration

## Prerequisites

- Python: Ensure Python is installed on your system.

## Installation & Initialization

1. Install ZenML & MLflow:

- Open your terminal or command prompt.

- Run the following commands:

**pip install zenml  
 pip install mlflow**  
  
 Purpose: Installs ZenML for pipeline management and MLflow for experiment tracking.

2. Initialize ZenML:

- Open your terminal or command prompt.

- Run the following command:

**zenml init**  
  
 Purpose: Prepares your working directory for ZenML projects.

## Configuration Steps

1. Install MLflow Integration:

- Open your terminal or command prompt.

- Run the following command:

**zenml integration install mlflow -y**  
  
 Purpose: Integrates MLflow into ZenML for enhanced experiment tracking.

2. Register an Orchestrator:

- Open your terminal or command prompt.

- Run the following command, replacing placeholders as needed:

**zenml orchestrator register <orchestrator\_name> --flavor=<flavor>**  
Certainly! Here are some examples of `<orchestrator\_name>` when registering orchestrators with different flavors:

1. Registering a local orchestrator:

**zenml orchestrator register local\_orchestrator --flavor=local**

2. Registering an orchestrator for Apache Airflow:

zenml orchestrator register airflow\_orchestrator --flavor=airflow

3. Registering a Kubernetes-based orchestrator for Kubeflow:

zenml orchestrator register kubeflow\_orchestrator --flavor=kubeflow

4. Registering an orchestrator for AWS SageMaker:

zenml orchestrator register sagemaker\_orchestrator --flavor=sagemaker

5. Registering an orchestrator for Google Cloud Vertex AI:

zenml orchestrator register vertex\_orchestrator --flavor=vertex

You can replace `<orchestrator\_name>` with any name that makes sense for your use case, as long as it is unique and descriptive of the orchestrator's purpose or environment.  
 Replace '<orchestrator\_name>' with your chosen name (e.g., 'local\_orchestrator').  
 Replace '<flavor>' with an orchestrator flavor like 'local', 'airflow', or 'kubeflow'.  
  
 Purpose: Defines how and where your pipeline runs.

3. Register an Artifact Store:

- Open your terminal or command prompt.

- Run the following command, replacing placeholders as needed:

**zenml artifact-store register <store\_name> --flavor=local --path=<path>**

Certainly! Here are some examples of `<store\_name>` when registering an artifact store with different orchestrators:

1. Registering a local artifact store for the Kubeflow orchestrator:

```

zenml artifact-store register local\_store\_for\_kubeflow --flavor=local --path=/path/to/local/artifacts

```

2. Registering a Google Cloud Storage (GCS) artifact store for the Kubeflow orchestrator:

```

zenml artifact-store register gcs\_store\_for\_kubeflow --flavor=gcs --path=gs://your-bucket-name/path/to/artifacts

```

3. Registering an Amazon S3 artifact store for the Kubeflow orchestrator:

zenml artifact-store register s3\_store\_for\_kubeflow --flavor=s3 --path=s3://your-bucket-name/path/to/artifacts

4. Registering a Minio artifact store for the Kubeflow orchestrator:

zenml artifact-store register minio\_store\_for\_kubeflow --flavor=minio --path=http://minio-server:9000/path/to/artifacts

5. Registering a local artifact store for the Airflow orchestrator:

zenml artifact-store register local\_store\_for\_airflow --flavor=local --path=/path/to/airflow/artifacts  
 Replace '<store\_name>' and '<path>' appropriately.  
  
 Purpose: Specifies where pipeline artifacts (outputs) are stored.

4. Register MLflow Tracker:

- Open your terminal or command prompt.

- Run the following command, replacing placeholders as needed:

**zenml experiment-tracker register <tracker\_name> --flavor=mlflow**  
  
 Replace '<tracker\_name>' as desired.  
  
 Purpose: Sets up MLflow for logging and tracking experiments.

5. Create and Set a ZenML Stack:

- Open your terminal or command prompt.

- Run the following command, replacing placeholders as needed:

**zenml stack register <stack\_name> -m default -a <artifact\_store\_name> -o <orchestrator\_name> -e <tracker\_name> --set**  
  
 Replace '<stack\_name>', '<artifact\_store\_name>', '<orchestrator\_name>', and '<tracker\_name>' with your chosen names.  
  
 Purpose: Combines all components (orchestrator, artifact store, tracker) into a single stack for running ML pipelines.

## Example: Local Setup

1. Install and Initialize:

- Open your terminal or command prompt.

- Run the following commands:

**pip install zenml  
 pip install mlflow  
 zenml init**  
  
 Purpose: Install ZenML, MLflow, and initialize ZenML in your working directory.

2. Configure Components:

- Open your terminal or command prompt.

- Run the following commands:

**zenml integration install mlflow -y  
 zenml orchestrator register local\_orchestrator --flavor=local  
 zenml artifact-store register local\_artifacts --flavor=local --path=/path/to/local/artifacts  
 zenml experiment-tracker register local\_mlflow --flavor=mlflow**   
 **zenml stack register my\_stack -a local\_store\_for\_kubeflow -o kubeflow\_orchestrator -e local\_mlflow --set** Purpose: Configure ZenML components for local setup.

**Multi-Tenant Kubeflow Deployment**

If you're working with a multi-tenant Kubeflow deployment, you need to set the **kubeflow\_hostname** attribute in the orchestrator configuration. This hostname is the address of your Kubeflow dashboard.

**Single-Tenant Kubeflow Deployment**

For a single-tenant deployment, you have two options:

## Additional Notes

- Orchestrator Flavors: Different flavors allow for different environments. For example, 'airflow' for Apache Airflow, 'kubeflow' for Kubernetes-based workflows, etc.

- Cloud & On-Premise Setups: For cloud or on-premise, change the orchestrator flavor and artifact store path accordingly. For instance, use 'sagemaker' for AWS or 'vertex' for Google Cloud.

## Final Thoughts

- Flexibility: ZenML supports various environments, making it versatile for different needs.

- Ease of Use: Once set up, ZenML simplifies running and managing ML pipelines.

- Scalability: Easily scalable from local development to cloud-based production environments.

**Command I ran personally**   
zenml integration install mlflow -y (.venv) (base) yatin@Yatins-Air poc2\_mlops % zenml orchestrator register kubeflow\_orchestrator --flavor=kubeflow Successfully registered orchestrator `kubeflow\_orchestrator`. Dashboard URL: http://127.0.0.1:8237/workspaces/default/components/orchestrator/5ad03a6b-b90f-4f0d-b1e9-e20afe91427c/configuration (.venv) (base) yatin@Yatins-Air poc2\_mlops % zenml artifact-store register local\_store\_for\_kubeflow --flavor=local --path=/Users/yatin/Desktop/CGI/Example Successfully registered artifact\_store `local\_store\_for\_kubeflow`. Dashboard URL: http://127.0.0.1:8237/workspaces/default/components/artifact\_store/8f9a71ba-e44c-4d39-a0c2-3f480a6497e3/configuration (.venv) (base) yatin@Yatins-Air poc2\_mlops % zenml experiment-tracker register local\_mlflow --flavor=mlflow Successfully registered experiment\_tracker `local\_mlflow`. Dashboard URL: http://127.0.0.1:8237/workspaces/default/components/experiment\_tracker/664b4e3e-cb2c-4e83-86c5-39b6cd6d7f8e/configuration (.venv) (base) yatin@Yatins-Air poc2\_mlops % zenml stack register my\_stack -m default -a local\_store\_for\_kubeflow -o kubeflow\_orchestrator -e local\_mlflow --set Usage: zenml stack register [OPTIONS] STACK\_NAME Try 'zenml stack register --help' for help. Error: Got unexpected extra arguments (-m default) (.venv) (base) yatin@Yatins-Air poc2\_mlops % zenml stack register my\_stack -a local\_store\_for\_kubeflow -o kubeflow\_orchestrator -e local\_mlflow --set ⠋ Registering stack 'my\_stack'... You are configuring a stack that is composed of components that are relying on local resources (artifact\_store: local\_store\_for\_kubeflow, experiment\_tracker: local\_mlflow) as well as components that are running remotely (orchestrator: kubeflow\_orchestrator). This is not recommended as it can lead to unexpected behavior, especially if the remote components need to access the local resources. Please make sure that your stack is configured correctly, or try to use component flavors or configurations that Stack 'my\_stack' successfully registered! Active repository stack set to:'my\_stack' Dashboard URL: http://127.0.0.1:8237/workspaces/default/stacks/01123573-a8db-421b-982f-840a0ee01eb3/configuration

Further commands I ran

docker run -d -p 5001:5000 --restart=always --name registry registry:2

zenml container-registry register local\_registry --flavor=default --uri=localhost:5001

zenml stack update my\_stack -c local\_registry