🐸FrogID ML - Databricks MLOps Naming Conventions

Context & Assumptions

- The aus_museum_dbx_dev workspace may include **multiple projects**, not limited to FrogID.
- There is **no need to isolate** intermediate data from other FrogID-related streams (e.g., server-side development), since they primarily provide input to or consume output from the ML workflow.
- Currently, only a single Databricks workspace is available. Different environments are simulated as separate catalogs within the development workspace. Once the project is production-ready, a dedicated production workspace will be provisioned.

General Naming Principles

- Use **lowercase** with **underscores** (e.g., frog_data, not FrogData)
- Use only _ as a separator (no dashes or special characters)
- Be **descriptive but concise** (e.g., gold_species_summary, not species_final_report_v1)

Naming Convention Formats

In Databricks Unity Catalog(UC), all metadata is registered in a metastore. The hierarchy of database objects in any Unity Catalog metastore is divided into three levels, represented as a three-level namespace (catalog.schema.table-etc) when you reference tables, views, volumes, models, and functions.

Catalog Format

```
Unset
<organisation_unit>_<env>
```

Examples:

- aus_museum_dbx_dev
- aus_museum_dbx_staging

- aus_museum_dbx_prod

Schema Format

```
Unset
<project/team_sandbox>
```

Example:

- frogid_ml

Table Format

```
Unset
<stage>_<subject>_[<purpose>]
```

Component	Description	Example
stage	Processing stage (bronze, silver, gold)	silver, gold
subject	Main data topic/entity	cane_toad, frog_call
purpose	*(Optional)* Usage or role	features, labels, prediction

Example Table Names

Catalog	Schema	Table Name	Description
aus_museum_dbx_dev	frogid_ml	bronze_cane_toad	Raw ingestion of cane toad data

aus_museum_dbx_staging	frogid_ml	silver_cane_toad_features	Feature-engineered training dataset
aus_museum_dbx_staging	frogid_ml	silver_cane_toad_labels	Labeled outcomes for model training
aus_museum_dbx_prod	frogid_ml	gold_cane_toad_prediction	Final model prediction outputs for serving

Volume Usage & Directory Structure

Volumes are used to store unstructured and intermediate data assets such as raw audio files, processed feature sets, model artifacts, and predictions.

- **External volumes(frogid_files_s3)** will be used to read audio files directly from an S3 bucket (read-only) to avoid data duplication and reduce storage costs.
- **Internal volumes(frogid_files)** will be used to store temporary or generated files such as preprocessed datasets, inference outputs, and model binaries.

Internal Volume Directory Structure

To standardize usage, each internal volume will follow a structured layout with the following directories:

Directory	Purpose	Example File
input/	Raw or ingested data	input/sample.csv, input/sample.wav
staging/	Intermediate, preprocessed files	staging/processed.parquet
output/	Inference results or final outputs	output/predict.parquet
model/	Serialized model artifacts or metadata. Models will eventually be	model/frog_id.pkl

	registered in the catalog—use this directory only for exporting models temporarily for analysis or validation purposes.	

External Volume Set up

External volume is created pointing at *s3://austmus-frogid/* to read .wav/.acc files that was collected and transformed from upstream jobs.

MLFlow Experiment naming convention

There are 3 areas we need a naming convention for:

- Experiments
- Runs
- Models

Experiments naming convention:

Experiments should be grouping a collection of different experiment runs where the metric results are comparable. If what is being predicted differs, or the data used differs, then the results could not be compared.

Experiment naming convention suggestion:

- **problem**: The specific problem being solved (e.g., species_detection, call_classification)
- **target**: What you're trying to predict (e.g., single_species, multi_species, 15species)
- **data_selection**: Characteristics of input data (e.g., weak_label, strong_label, spectrograms)

{problem}_{target}_{data_selection}

E.g.
Problem: frogidml
Target: binary_single_species_weak_labels
Data: 30sec_canetoads_vs_other

frogidml_binary_single_species_weak_labels_30sec_canetoads_vs_other

Run naming convention:

It is helpful to name runs under an experiment to convey what was different about that trial. A recommended naming convention is:

{model_family}_{description}_{timestamp}

- **model_family**: The type/architecture of model (e.g., cnn_resnet50, transformer_wav2vec, logistic)
- **description**: Brief description of what's unique about this run (e.g., augmented, denoised, hyperparameter_batch32)
- **timestamp**: Automatic timestamp or date (e.g., 20250512)

Model naming convention:

The model name can get very long if we simply concatenate the experiment_name x run_name, so better to simply pick a generally descriptive name with automatic version number

frogidml_{target}_{model_type}_{version}

- **target**: What the model predicts (e.g., species_15, cane_toad)
- **model_type**: Type of model (e.g., classifier, detector)

version: MLflow will assign a version number automatically