DB2 Evidence Loader

Python tool to load YAML evidence files into a DB2 database.

NOTES:

- With minor changes, it should also be applicable for other database providers.
- In the guide we use Db2 for Linux, UNIX, with minor changes it should also be applicable for Db2 for z/OS.
- The Python source code, DB2 schema, and configuration files are mentioned solely as example use cases. They should not be interpreted as any form of design commitment.
- This this sample code that may contains issues. Not to use in production.

Table of Contents

Architecture

The project uses a **Template Method** pattern with:

- `DB2EvidenceLoaderBase` : Abstract class containing all business logic
- `DB2EvidenceLoaderIBM` : Implementation using `ibm_db` driver
- `DB2EvidenceLoaderJDBC` : Implementation using `JayDeBeApi` (JDBC) driver

Artifact Uniqueness

Important: Artifacts are identified by their **PATH** (unique constraint):

- Same artifact path = same artifact (reused across multiple steps)
- Many-to-many relationship between STEP and ARTIFACT via `STEP_ARTIFACT` junction table
- When an artifact with the same path is encountered, the existing artifact is linked instead of creating a duplicate
- The `ARTIFACT_HASH` field has been removed from the schema

File Structure

```
project/
│
├── db2_config.yaml          # Configuration file (REQUIRED)
├── db2_config.py            # Configuration loader
├── db2_evidence_base.py     # Abstract base class
├── db2_evidence_ibm.py      # IBM DB implementation
├── db2_evidence_jdbc.py     # JDBC implementation
├── db2_schema_ddl.sql       # Database schema DDL
└── README.md                # This file
```

Database Schema

The database uses the following structure:

```
DEPLOY (1) ──┬──→ ACTIVITY (N) ──┬──→ ACTION (N) ──┬──→ STEP (N)
             │                   │                 │ │
             └──→ PROPERTIES     └──→ PROPERTIES   │ └──→ STEP_ARTIFACT (junction)
                                                   └──→ PROPERTIES  │
                                                                    └──→ ARTIFACT (unique by PATH)
                                                                          └──→ PROPERTIES
```

Key Points:

- **ARTIFACT** table stores unique artifacts identified by `ARTIFACT_PATH`
- **STEP_ARTIFACT** junction table creates many-to-many relationship
- Same artifact can be used by multiple steps (no duplication)
- Constraint: `UNIQUE (ARTIFACT_PATH)` ensures artifact uniqueness

Database Setup

Before using the evidence loader, you must create the database and schema.

Step 1: Create Database (if needed)

On Linux/Unix where DB2 is installed:

```
# Switch to DB2 instance owner
su - db2inst1

# Create database
db2 create database DEPLOY using codeset UTF-8 territory en PAGESIZE 8192

# Verify database was created
db2 list database directory | grep DEPLOY
```

Step 2: Create Schema

Create the schema that will contain all tables:

```
-- Connect to database
db2 connect to DEPLOY user db2inst1

-- Verify connection
db2 "SELECT CURRENT SERVER FROM SYSIBM.SYSDUMMY1"

-- Create schema
db2 "CREATE SCHEMA DEPLOYZ AUTHORIZATION db2inst1"

-- Verify schema was created
db2 "SELECT SCHEMANAME FROM SYSCAT.SCHEMATA WHERE SCHEMANAME = 'DEPLOYZ'"
```

Step 3: Create Tables

Run the DDL script to create all tables:

```
# Execute the schema creation script
db2 -tvf db2_schema_ddl.sql
```

Step 4: Verify Tables

Check that all tables were created successfully:

```
-- List all tables in DEPLOYZ schema
db2 "SELECT TABNAME FROM SYSCAT.TABLES WHERE TABSCHEMA = 'DEPLOYZ' ORDER BY TABNAME"

-- Expected output:
-- TABNAME
--------------------------
-- ACTION
-- ACTIVITY
-- ARTIFACT
-- DEPLOY
-- PROPERTIES
-- STEP
-- STEP_ARTIFACT
-- STEP_RESULT_DETAIL
-- V_ARTIFACT_USAGE
-- V_DEPLOYMENT_HIERARCHY
-- V_STEP_ARTIFACTS
```

Installation and Configuration

Option 1: IBM DB Driver (ibm_db)

Installation

```
pip install ibm_db
```

Configuration (Windows)

The IBM DB driver requires DB2 CLI libraries on Windows:

1. **Download DB2 CLI Driver**
   - Download from: IBM Data Server Driver Package
   - Extract to `C:\clidriver` (or another directory)

Useful Links

- **Official Documentation**: https://github.com/ibmdb/python-ibmdb
- **API Reference**: https://github.com/ibmdb/python-ibmdb/wiki/APIs
- **Installation Guide**: https://github.com/ibmdb/python-ibmdb#installation
- **Troubleshooting** (usefull for z/OS DB2 support): https://github.com/ibmdb/python-ibmdb/blob/master/README.md#troubleshooting

Prerequisites

- **Python**: 3.6+

Option 2: JDBC Driver (JayDeBeApi)

Installation

```
pip install JayDeBeApi
pip install JPype1  # Required for JayDeBeApi
```

Configuration

1. **Download DB2 JDBC Driver**
   - Download `db2jcc4.jar` from: IBM Data Server Driver for JDBC
   - Or extract from DB2 installation: `<DB2_HOME>/java/db2jcc4.jar`
   - Save to an accessible directory (e.g., `/opt/drivers/db2jcc4.jar`)

Useful Links

- **JayDeBeApi Documentation**: https://github.com/baztian/jaydebeapi
- **JPype1 Documentation**: https://jpype.readthedocs.io/
- **DB2 JDBC Driver**: https://www.ibm.com/support/pages/db2-jdbc-driver-versions-and-downloads
- **DB2 JDBC Developer Guide**: https://www.ibm.com/docs/en/db2/12.1.x?topic=apis-installing-data-server-driver-jdbc-sqlj

- **JayDeBeApi Examples**: https://github.com/baztian/jaydebeapi/blob/master/README.rst

Prerequisites

- **Java JDK**: 8 or higher (JRE is sufficient)
- **Python**: 3.6+
- **JAR File**: `db2jcc4.jar` or `db2jcc.jar`

Usage

Configuration File

Update the `db2_config.yaml` file with your database connection settings:

```yaml
# Driver selection: ibm_db or jdbc
driver: ibm_db

# Database schema
schema: DEPLOYZ

# IBM DB configuration (if driver: ibm_db)
ibm_db:
  database: DEPLOY
  hostname: localhost
  port: 50000
  protocol: TCPIP
  uid: ${DB2_USER}
  pwd:  ${DB2_PASSWORD}

# JDBC configuration (if driver: jdbc)
jdbc:
  url: jdbc:db2://localhost:50000/DEPLOY
  username: ${DB2_USER}
  password:  ${DB2_PASSWORD}
  driver_path: /opt/drivers/db2jcc4.jar
  driver_class: com.ibm.db2.jcc.DB2Driver

# Logging
logging:
  level: INFO
  format: '%(asctime)s - %(levelname)s - %(message)s'

# Options
options:
  verbose: true
  stop_on_error: false
```

Using Environment Variables in Configuration

You can use environment variables in your `db2_config.yaml`:

```yaml
jdbc:
  url: jdbc:db2://${DB_HOST}:${DB_PORT}/${DB_NAME}
  username: ${DB_USER}
  password: ${DB_PASSWORD}
  driver_path: ${JDBC_DRIVER_PATH}
```

Then set the environment variables:

```
# Unix
export DB_HOST=localhost
export DB_PORT=50000
export DB_NAME=DEPLOY
export DB_USER=db2inst1
export DB_PASSWORD=secret
export JDBC_DRIVER_PATH=/opt/drivers/db2jcc4.jar

# Windows
set DB_HOST=localhost
set DB_PORT=50000
set DB_NAME=DEPLOY
set DB_USER=db2inst1
set DB_PASSWORD=secret
set JDBC_DRIVER_PATH=C:\drivers\db2jcc4.jar
```

Using IBM DB Driver

Command Line

```
# Set credentials
set DB2_USER=YOUR_DB2_USER
set DB2_PASSWORD=YOUR_DB2_PASSWORD

# Using default db2_config.yaml
python.exe db2_evidence_ibm.py evidences.yml

# Using custom config file
python.exe db2_evidence_ibm.py evidences.yml my_config.yaml
```

Using JDBC Driver

Command Line

```
# Set credentials
export DB2_USER=YOUR_DB2_USER
export DB2_PASSWORD=YOUR_DB2_PASSWORD

# Using default db2_config.yaml
python3 db2_evidence_jdbc.py evidences.yml

# Using custom config file
python3 db2_evidence_jdbc.py evidences.yml my_config.yaml
```

Querying Samples

```
-- List all deployments in a specific environment
SELECT DEPLOY_ID, ENVIRONMENT_NAME, DEPLOY_TIMESTAMP, STATUS
FROM DEPLOYZ.DEPLOY
WHERE ENVIRONMENT_NAME = 'PROD'
ORDER BY DEPLOY_TIMESTAMP DESC;

-- List all artifacts deployed in a specific environment
SELECT DISTINCT art.ARTIFACT_ID,
                art.ARTIFACT_NAME,
                art.ARTIFACT_TYPE,
                art.ARTIFACT_PATH
FROM DEPLOYZ.DEPLOY d
JOIN DEPLOYZ.ACTIVITY act ON d.DEPLOY_ID = act.DEPLOY_ID
JOIN DEPLOYZ.ACTION a ON act.ACTIVITY_ID = a.ACTIVITY_ID
JOIN DEPLOYZ.STEP s ON a.ACTION_ID = s.ACTION_ID
JOIN DEPLOYZ.STEP_ARTIFACT sa ON s.STEP_ID = sa.STEP_ID
JOIN DEPLOYZ.STEP_ARTIFACT sa ON s.STEP_ID = sa.STEP_ID
JOIN DEPLOYZ.ARTIFACT art ON sa.ARTIFACT_ID = art.ARTIFACT_ID
WHERE d.ENVIRONMENT_NAME = 'PROD'
ORDER BY art.ARTIFACT_NAME;

-- In which environments is the artifact named "LGACDB02" deployed?
SELECT DISTINCT
       d.ENVIRONMENT_NAME,
       d.DEPLOY_TIMESTAMP,
       act.ACTIVITY_NAME,
       a.ACTION_NAME,
       s.STEP_NAME,
       art.ARTIFACT_TYPE
FROM DEPLOYZ.DEPLOY d
JOIN DEPLOYZ.ACTIVITY act ON d.DEPLOY_ID = act.DEPLOY_ID
JOIN DEPLOYZ.ACTION a ON act.ACTIVITY_ID = a.ACTIVITY_ID
JOIN DEPLOYZ.STEP s ON a.ACTION_ID = s.ACTION_ID
JOIN DEPLOYZ.STEP_ARTIFACT sa ON s.STEP_ID = sa.STEP_ID
JOIN DEPLOYZ.ARTIFACT art ON sa.ARTIFACT_ID = art.ARTIFACT_ID
WHERE art.ARTIFACT_NAME = 'LGACDB02'
ORDER BY d.ENVIRONMENT_NAME, d.DEPLOY_TIMESTAMP

-- In which environments is the artifact named "LGACDB02" of type "CICSLOAD" deployed?
SELECT DISTINCT
       d.ENVIRONMENT_NAME,
       d.DEPLOY_TIMESTAMP,
       act.ACTIVITY_NAME,
       a.ACTION_NAME,
       s.STEP_NAME,
       art.ARTIFACT_TYPE
FROM DEPLOYZ.DEPLOY d
JOIN DEPLOYZ.ACTIVITY act ON d.DEPLOY_ID = act.DEPLOY_ID
JOIN DEPLOYZ.ACTION a ON act.ACTIVITY_ID = a.ACTIVITY_ID
JOIN DEPLOYZ.STEP s ON a.ACTION_ID = s.ACTION_ID
JOIN DEPLOYZ.STEP_ARTIFACT sa ON s.STEP_ID = sa.STEP_ID
JOIN DEPLOYZ.ARTIFACT art ON sa.ARTIFACT_ID = art.ARTIFACT_ID
WHERE art.ARTIFACT_NAME = 'LGACDB02' and art.ARTIFACT_TYPE = 'CICSLOAD'
ORDER BY d.ENVIRONMENT_NAME, d.DEPLOY_TIMESTAMP

-- List the properties (PROPERTIES) of an activity for a deployment with a specific DEPLOY_TIMESTAMP
SELECT
    act.ACTIVITY_ID,
    act.ACTIVITY_NAME,
    p.PROPERTY_KEY,
    p.PROPERTY_VALUE
FROM DEPLOYZ.DEPLOY d
JOIN DEPLOYZ.ACTIVITY act ON d.DEPLOY_ID = act.DEPLOY_ID
JOIN DEPLOYZ.PROPERTIES p
    ON p.ENTITY_TYPE = 'ACTIVITY'
   AND p.ENTITY_ID = act.ACTIVITY_ID
WHERE d.DEPLOY_TIMESTAMP = TIMESTAMP('2025-11-27 12:26:21.0')
ORDER BY act.ACTIVITY_ID, p.PROPERTY_KEY;

-- List the properties of a specific artifact (by name and type) for a given deployment timestamp
SELECT
    d.DEPLOY_ID,
    d.ENVIRONMENT_NAME,
    d.DEPLOY_TIMESTAMP,
    art.ARTIFACT_ID,
    art.ARTIFACT_NAME,
    art.ARTIFACT_TYPE,
    p.PROPERTY_KEY,
    p.PROPERTY_VALUE
FROM DEPLOYZ.DEPLOY d
JOIN DEPLOYZ.ARTIFACT art
    ON EXISTS (
        SELECT 1
        FROM DEPLOYZ.STEP_ARTIFACT sa
        JOIN DEPLOYZ.STEP s ON sa.STEP_ID = s.STEP_ID
        JOIN DEPLOYZ.ACTION a ON s.ACTION_ID = a.ACTION_ID
        JOIN DEPLOYZ.ACTIVITY act ON a.ACTIVITY_ID = act.ACTIVITY_ID
        WHERE sa.ARTIFACT_ID = art.ARTIFACT_ID
          AND act.DEPLOY_ID = d.DEPLOY_ID
    )
JOIN DEPLOYZ.PROPERTIES p
    ON p.ENTITY_TYPE = 'ARTIFACT'
   AND p.ENTITY_ID = art.ARTIFACT_ID
   AND p.DEPLOY_ID = d.DEPLOY_ID
WHERE d.DEPLOY_TIMESTAMP = TIMESTAMP('2025-11-27 12:26:21.0') and art.ARTIFACT_NAME = 'LGACDB02'
ORDER BY art.ARTIFACT_ID, p.PROPERTY_KEY;
```

Driver Comparison

Feature	IBM DB (`ibm_db`)	JDBC (`JayDeBeApi`)
Performance	□□□□□ Excellent	□□□□ Very Good
Installation	□□□ Medium (requires CLI)	□□□□ Easy (just JAR)
Portability	□□□ OS-dependent	□□□□□ Cross-platform
Compatibility	□□□□□ Native DB2	□□□□ Standard JDBC
Windows	□□□ Complex setup	□□□□□ Simple
Linux	□□□□□ Native	□□□□ Requires Java
z/OS	□□□□□ Native For DB2 on z/OS	□□□□ Requires Java

When to Use Which Driver?

Use `ibm_db` if:

- DB2 CLI is already installed
- Maximum performance required
- Running on Linux/AIX
- Running on z/OS to access DB2 on z/OS
- Need advanced DB2 features

Use `JayDeBeApi` if:

- Simplified installation desired
- Cross-database portability needed
- Docker/containerized deployment

Additional Resources

DB2 Documentation

- DB2 Knowledge Center

License