Managing Community Catalogs via catalogs-contrib

1. Motivation

As the Apache Gravitino community grows, we see an increasing demand for supporting various catalogs, particularly those that are vendor-specific or used in niche scenarios.

We are currently facing a dilemma:

- Maintenance Burden: Directly integrating every requested catalog into the main catalogs/ directory places a heavy burden on coremaintainers to guarantee stability and integration test coverage.
- Community Engagement: We want to welcome more contributors. If we set the bar too high or require too much maintenance for every new catalog, it might scare away developers and stop Gravitino from supporting more data sources.
- Repository Fragmentation: Moving these catalogs to a separate repository (e.g., gravitino-contrib) makes them get less attention and spot. Users tend to trust and contribute to the main Apache repository more than external ones.
- Distribution Size: Including too many catalogs makes the release binary excessively large, which may conflict with ASF's policy for distribution package sizes.

To balance community enthusiasm with maintainability, we propose introducing a catalogs-contrib module within the main repository.

2. Proposed Changes

2.1 Directory Structure Restructuring

We will create a new top-level directory named catalogs-contrib. Catalogs that are vendor-specific, experimental, or have a smaller user base will be placed here. Core catalogs (Hive, Lakehouse, standard JDBC) will remain in catalogs/.

Current Structure:

```
gravitino/
├── catalogs/
│   ├── catalog-hive
│   ├── catalog-lakehouse-iceberg
│   ├── catalog-jdbc-mysql
│   └── catalog-jdbc-oceanbase  <-- (To be moved)
```

New Structure:

```
gravitino/
├── catalogs/              <-- Core Catalogs (High standard, strict CI)
│   ├── catalog-hive
│   ├── catalog-lakehouse-iceberg
│   └── catalog-jdbc-mysql
│
└── catalogs-contrib/      <-- Community Catalogs (Relaxed CI requirements)
    └── catalog-jdbc-oceanbase
```

2.2 CI/CD Policy

To ensure the stability of the core CI pipeline and reduce resource consumption:

- Unit Tests (UT): Required for all modules, including those in catalogs-contrib.
- Integration Tests (IT):
  - Core Catalogs: Must pass all integration tests in the CI pipeline.
  - Contrib Catalogs: CI will NOT run integration tests by default. This avoids CI flakiness caused by unstable Docker images or resource limitations.

2.3 Distribution Strategy

We will update the Gradle build scripts to produce two types of binary distribution packages:

- Standard Package (gravitino-<version>-bin.tar.gz):
  - Contains the Gravitino server and core catalogs only.
  - Lightweight and stable.

- Extended Package (gravitino-<version>-bin-extend.tar.gz):
  - Contains everything in the standard package plus all modules under catalogs-contrib.

- "Batteries included" for users who need specific vendor support.
- This package will not be released, and users can build it by themselves.

2.4 Implementation Plan

- Refactor Build Script: Update settings.gradle.kts and build.gradle.kts to handle the new directory structure.
- Update CI Workflows: Modify GitHub Actions (e.g., backend-integration-test-action.yml) to exclude catalogs-contrib from heavy integration tests.
- Release Scripts: Update release scripts to generate and sign both binary packages.
- Documentation: Update the "Getting Started" guide to explain the difference between the two binary packages