MEP 14 -- Offer Milvus 2.0 JAVA SDK

Current state: Accepted

ISSUE: https://github.com/milvus-io/milvus-sdk-java/issues/188, https://github.com/milvus-io/milvus-sdk-java/issues/183

PRs:

Keywords: JAVA SDK

Released: Together with Milvus 2.0 stable release

Authors: trovwu, chinamcafee, Xiaofan, weilong

Summary(required)

We are gonna to deliver a JAVA SDK with full functionality for Milvus 2.0.

Motivation(required)

We've seen many users demands for JAVA SDK, it is probably the most popular SDKs except for pymilvus.

Public Interfaces(optional)

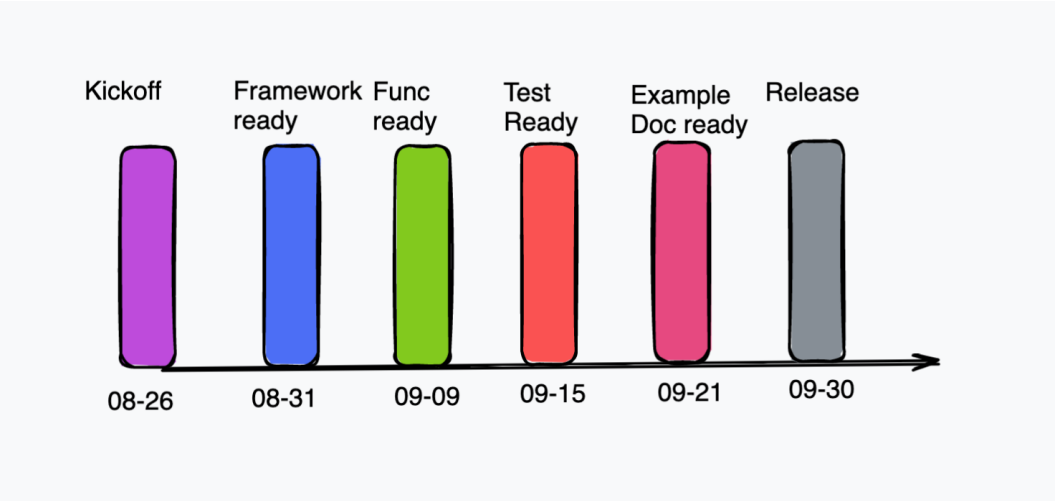
```
syntax = "proto3";

import "common.proto";
import "schema.proto";

option java_multiple_files = true;
option java_package = "io.milvus.grpc";
option java_outer_classname = "MilvusProto";
option java_generate_equals_and_hash = true;

package milvus.proto.milvus;

service MilvusService {
    //collection related
    rpc CreateCollection(CreateCollectionRequest) returns (common.Status) {}
    rpc DropCollection(DropCollectionRequest) returns (common.Status) {}
    rpc HasCollection(HasCollectionRequest) returns (BoolResponse) {}
    rpc LoadCollection(LoadCollectionRequest) returns (common.Status) {}
    rpc ReleaseCollection(ReleaseCollectionRequest) returns (common.Status) {}
    rpc DescribeCollection(DescribeCollectionRequest) returns (DescribeCollectionResponse) {}
    rpc GetCollectionStatistics(GetCollectionStatisticsRequest) returns (GetCollectionStatisticsResponse) {}
    rpc ShowCollections(ShowCollectionsRequest) returns (ShowCollectionsResponse) {}

    //partition related
    rpc CreatePartition(CreatePartitionRequest) returns (common.Status) {}
    rpc DropPartition(DropPartitionRequest) returns (common.Status) {}
    rpc HasPartition(HasPartitionRequest) returns (BoolResponse) {}
    rpc LoadPartitions(LoadPartitionsRequest) returns (common.Status) {}
    rpc ReleasePartitions(ReleasePartitionsRequest) returns (common.Status) {}
    rpc GetPartitionStatistics(GetPartitionStatisticsRequest) returns (GetPartitionStatisticsResponse) {}
    rpc ShowPartitions(ShowPartitionsRequest) returns (ShowPartitionsResponse) {}

    //index related
    rpc CreateIndex(CreateIndexRequest) returns (common.Status) {}
    rpc DescribeIndex(DescribeIndexRequest) returns (DescribeIndexResponse) {}
    rpc GetIndexState(GetIndexStateRequest) returns (GetIndexStateResponse) {}
    rpc GetIndexBuildProgress(GetIndexBuildProgressRequest) returns (GetIndexBuildProgressResponse) {}
    rpc DropIndex(DropIndexRequest) returns (common.Status) {}

    //dml,dql
    rpc Insert(InsertRequest) returns (MutationResult) {}
    rpc Delete(DeleteRequest) returns (MutationResult) {}
    rpc Search(SearchRequest) returns (SearchResults) {}
    rpc Flush(FlushRequest) returns (FlushResponse) {}
    rpc Query(QueryRequest) returns (QueryResults) {}
    rpc CalcDistance(CalcDistanceRequest) returns (CalcDistanceResults) {}

    //management
    rpc GetPersistentSegmentInfo(GetPersistentSegmentInfoRequest) returns (GetPersistentSegmentInfoResponse) {}
    rpc GetQuerySegmentInfo(GetQuerySegmentInfoRequest) returns (GetQuerySegmentInfoResponse) {}
    rpc Dummy(DummyRequest) returns (DummyResponse) {}
    // https://wiki.lfaidata.foundation/display/MIL/MEP+8+--+Add+metrics+for+proxy
    rpc GetMetrics(GetMetricsRequest) returns (GetMetricsResponse) {}
}
```

Design Details(required)

TODO lists

1. API Design 3 Reference, GO SDK, Java 1.1 SDK, Pymilvus SDK
2. Initial version of the code, finish all the frameworks, including how the sdk communicate with milvus cluster, how we do unit test.
3. Finish DDL, DML , Control API implementation
4. Unit Test
5. Usage emamples
6. API Document
7. CI Test

Time Lines



Which JDK version are we gonna to support?

Milvus java SDK 2.0 is gonna to support JDK >= 1.8. The Reasons are:

1) Milvus java SDK 1.0 follows the same rule

2) JDK 1.8 offers many grammer sugar which are very useful

3) We don't foresee many old systems that really need milvus 2.0

Which style of API JAVA JDK is gonna to support?

We decide to support low level APIs in our first version, Higher level Apis can be implement on top of low level apis.

How do we test java sdk?

1) bring up a standalone milvus cluster by test containers.

Compatibility, Deprecation, and Migration Plan(optional)

- Milvus 2.0 JAVA SDK won't be compatible to Milvus 0.x Server or 1.x Server, User pick different SDKs based on their server version.

Test Plan(required)

1. Unit test
    a. JAVA SDK will implement a mock milvus for basic testing,
    b. Start a standalone milvus complicated test.
2. CI test
    a. do we need to setup basic ci test for further improvement?
3. Examples
    a. finish all the examples in user guide and make sure it works https://milvus.io/docs/v2.0.0/example_code.md

Rejected Alternatives(optional)

If there are alternative ways of accomplishing the same thing, what were they? The purpose of this section is to motivate why the design is the way it is and not some other way.

References(optional)

Briefly list all references