KIP-1007: Introduce Remote Storage Not Ready Exception

Status

Current state: *Accepted*

Discussion thread: *here*

JIRA: *KAFKA-15876*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

When tiered storage is enabled on the cluster, Kafka broker has to build the remote log metadata for all the partitions that it is either leader/follower on node restart. The remote log metadata is built in asynchronous fashion and does not interfere with the broker startup path. Once the broker becomes online, it cannot handle the client requests (FETCH and LIST_OFFSETS) to access remote storage until the metadata gets built for those partitions. Currently, we are returning a ReplicaNotAvailable exception back to the client so that it will retry after sometime.

ReplicaNotAvailableException is applicable when there is a reassignment is in-progress and kind of deprecated with the NotLeaderOrFollowerException (PR#8979). It's good to introduce an appropriate retriable exception for remote storage errors to denote that it is not ready to accept the client requests yet.

Public Interfaces

New Exception class:

RemoteStorageNotReadyException

```
package org.apache.kafka.common.errors;

/**
 * An exception that indicates remote storage is not ready to receive the requests yet.
 */
public class RemoteStorageNotReadyException extends RetriableException {

    private static final long serialVersionUID = 1L;

    public RemoteStorageNotReadyException(String message) {
        super(message);
    }

    public RemoteStorageNotReadyException(String message, Throwable cause) {
        super(message, cause);
    }

    public RemoteStorageNotReadyException(Throwable cause) {
        super(cause);
    }

}
```

Proposed Changes

When the metadata is not ready, instead of returning ReplicaNotAvailableException in RemotePartitionMetadataStore, we will return the new RemoteStorageNotReadyException.

The consumer can read the local data as long as it knows the offset from where to fetch the data from. When there is no initial offset, the consumer decides the offset based on the below config:

```
auto.offset.reset = earliest / latest / none
```

- For `earliest` offset policy and any offset that lies in the remote storage, the consumer (FETCH request) cannot be able to make progress until the remote log metadata gets synced.
- In a FETCH request, when there are multiple partitions where a subset of them are consuming from local and others from remote, then only the partitions which are consuming from the remote cannot make progress and the partitions that fetch data from local storage should be able to make progress.
- In a FETCH request, when the fetch-offset for a partition is within the local-storage, then it should be able to consume the messages.
- All the calls to LIST_OFFETS will fail until the remote log metadata gets synced.

Compatibility, Deprecation, and Migration Plan

- Both ReplicaNotAvailable and RemoteStorageNotReady exceptions extend RetriableException class.
- If the client implementation explicitly checks for ReplicaNotAvailableException instead of RetriableException, then it will break the clients. Tiered storage was not production ready in 3.6, so we don't expect any specific client side changes.

Test Plan

Describe in few sentences how the KIP will be tested. We are mostly interested in system tests (since unit-tests are specific to implementation details). How will we know that the implementation works as expected? How will we know nothing broke?

- Will cover the patch with unit tests.

Rejected Alternatives

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.