KIP-619: Add internal topic creation support

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: Not created yet

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

Motivation

Kafka and its upstream applications treat internal topics differently from non-internal topics. For example:

- Kafka does not allow user-defined internal topics.
- Internal topic partitions cannot be added to a transaction
- Internal topic records cannot be deleted
- Producing to internal topics might get rejected

Clients and upstream applications may define their own internal topics. For example, Kafka Connect defines `connect-configs`, `connect-offsets`, and `connect-statuses`. Clients are fetching the internal topics by sending the MetadataRequest (ApiKeys.METADATA).

However, clients and upstream application cannot register their own internal topics in servers. As a result, servers have no knowledge about client-defined internal topics. They can only test if a given topic is internal or not simply by checking against a static set of internal topic string, which consists of two internal topic names `__consumer_offsets` and `__transaction_state`. As a result, MetadataRequest cannot provide any information about the client created internal topics.

To solve the pain point, I'm proposing support for clients to register and query their own internal topics.

Public Interfaces

1. TopicConfig will have a new topic config `internal`, which indicates if the topic is internal or not.

Proposed Changes

Server-side:

Internal topic metadata propagation

The static internal topic testing is defined in Topic.java. At server-side, it's dependents are:

1. KafkaApis
2. MetadataCache
3. ReplicaManager

Both ReplicaManager and KafkaApi keep the mutable MetadataCache, which provides the metadata for each topic partition. Instead of testing internal topics against a hard-coded set of string, the server should refer the cached metadata for the internal topic testing. To achieve this, UpdateMetadataPartitionState will keep the information if the topic is internal.

Below will be the workflow about how the internal topic information gets propagated to all servers.

1. When a topic gets created, clients will pass the topic config `internal`. Zookeeper will be aware of the client-created internal topics.

2. KafkaZkClient will be able to query all the internal topics.
3. KafkaController can then utilize the information provided by KafkaZkClient and wrap a boolean field in its UpdateMetadataRequests for propagating the internal topic information to all servers.

After this process, all servers will be aware of the latest set of internal topics and can cache internal topics in MedatadaCache. Thus, that KafkaApi can construct the metadata response with the information of all clients created internal topics by referring MetadataCache.

Internal topic behaviors

Let's denote

- topics `__consumer_offsets` and `__transaction_state` as system-defined internal topics,
- topics whose config contain `internal=true` as user-defined internal topics.

ACL:

Below will be the default allowed operations for internal topics. Cluster admin might want to add restrictions using ACLs on user-defined internal topics depending on the actual user application logic.

	system-defined internal topics	**user-defined internal topics**
Topic creation (ApiKeys. CREATE_TOPICS)	allowed	allowed
Topic deletion (ApiKeys. DELETE_TOPICS)	forbidden	allowed
Produce (ApiKeys.PRODUCE)	forbidden	allowed
Add to transaction (ApiKeys. ADD_PARTITIONS_TO_TXN)	allowed	allowed

Metadata:

For the metadata operation, user-defined internal topics will be treated in the same way as system-defined internal topics. For example, in the metadata request (`ApiKeys.METADATA`), broker will mark both user-defined internal topics and system-defined internal topics as `internal`.

Client-side:

To get the internal topic information, instead of using the static internal topic testing or implementing their own logic, clients can utilize KafkaAdminClients and make a MetadataRequest (ApiKey.METADATA).

Post ZK world

KIP-500 proposed metadata quorum. Since the changes proposed in this KIP interact directly with KafkaApi instead of Zookeeper and modifies the cached metadata, it should be easily migrated in Post ZK world.

Compatibility, Deprecation, and Migration Plan

In the current version, users might create a topic with the configuration internal = true. After the changes in this proposal got adopted, the semantic of this configuration would change and might break the expected behavior. The upstream application must change the topic config key from `internal` to something else and change the application logic if necessary.

Rejected Alternatives

1. Specify a naming convention that all internal topic should start with the prefix `_`.
    a. It's hard to make all clients adjust their topic names.
2. Change several public APIs to make the clients pass a flag indicating if the topic is internal or not when it creates a topic. Add a new ZK path such as `topics/internal`.
    a. May require a new flag in TopicCommand