KIP-981: Manage Connect topics with custom implementation of Admin

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here*

Motivation

Similiar to KIP-787: MM2 manage Kafka resources with custom Admin implementation motivations. This KIP motivation is to make sure connect can use a custom admin to create topics.

At the moment Connect can either create topics automatically or using Admin client when topic.creation.enable enabled set to true. This setup assume that Kafka cluster either has auto.create.topics.enable or that the connect client has admin access to create all needed topics.

While the current approach simplifies creating topics in connect, it created a problem for any organization that runs any sort of resource management or federated solutions where these systems are usually the only application allowed to create topics in Kafka's ecosystem to control capacity and budget planning.

The downside of connect using AdminClient directly is that Connect create topics using AdminClient whenever `topic.creation.enable` is true ignoring any capacity safeguard set by the ecosystem. This increase the disk usage on the cluster. The team that runs the cluster will only notice the capacity issue when their disk usage hits the threshold of their alerts.

Currently there are 3 ways to deal with this issue:

- Drop connect related topics from the centralized/federated system and accept the risk of not tracing these topics. This direction avoids a conflict between connect topics and a centralized system however it will create capacity and budget issues if connect cluster started to handle more usecases that require creating more topics in Kafka.
- Creating Connect topics upfront before setting up new connectors. This approach requires a lot of manual work.
- Run a watcher that sync Kafka topics back to the centralized system. The problem with this approach that we update the centralized system after the fact. So any capacity safeguard in this system wouldn't be applied as the topics have been created already on Kafka cluster.

This KIP proposes to reuse the `org.apache.kafka.clients.admin.ForwardingAdminClient` class to allow users of Connect to easily integrate Connect with their ecosystem.

Public Interfaces

The KIP proposes adding flexibility to how connect manages Kafka topics. By **allowing connectors and workers to load a custom implementation of Admin interface.**
To make it easier for users to provide their custom implementation of Admin, the KIP will re-use the `ForwardingAdmin`class which delegates to `KafkaAdminClient.`
The implemented class can be overridden using the following configurations.

- forwarding.admin.class default value will be set to ForwardingAdmin with Map<String, Object> config to configure KafkaAdminClient as delegate.
- The provided implementation must have a contract that accepts Map<String, Object> config to configure KafkaAdminClient and any custom resource management clients

The configuration for custom resource management client and/or KafkaAdminClient can be passed using `admin.*` as a prefix

Proposed Changes

- Introduce a new TopicAdmin's constructor that accepts a client factory

```
public TopicAdmin(Map<String, Object> adminConfig, Function<Map<String, Object>, Admin> adminFactory) {
        this(adminConfig, adminFactory.apply(adminConfig));
    }
```

- Update `SharedTopicAdmin` to support the new interface of `TopicAdmin` .
    - `SharedTopicAdmin` already have an interface that accept client factory currently it's private and hidden behind the default constructor. This KIP will make this constructor the default instead.
- Initialise `` `SharedTopicAdmin` `` using the new default constructor.
- As well as updating `DeadLetterQueueReporter` to use ForwardingAdmin instead of `KafkaAdminClient`

Connector Configuration Properties

Properties common to the Connectors(s) and Worker(s):

Config name	Default	Description
`forwarding.admin.class`	org.apache.kafka.clients.admin.ForwardingAdminClient	The fully qualified name of class that extend ForwardingAdminClient. The class must have a contractor that accept configuration (Map<String, Object> config) to configure KafkaAdminClient and any other needed clients.

Compatibility, Deprecation, and Migration Plan

Connectors and workers will still use `KafkaAdminClient` behind the scenes with the default config for **`forwarding.admin.class`**

Rejected Alternatives

Similar to the ones listed in KIP-787

Manage creating and modifying Kafka topics outside Connect by building a separate tool that monitors the same set of topics created by Connect and create/modify topics once it detects configuration changes. The downsides with this are

- This will be a duplicate effort as Connect already has all this logic implemented; it only needs to use a different client than AdminClient.
- Connect still bypasses any capacity safeguards the ecosystem would have in place.
- Any downtime with this tool will cause conflict between Connect resources and management resource system.