KIP-37 - Add Namespaces to Kafka

- Status
- Motivation
- Public Interfaces
- Proposed Changes
    - Possible Delimiters
    - Multi-tiered Namespaces
        - Create
        - List
        - Delete
        - Move
    - Changes in Storage Layouts
    - Namespace Acls
    - Namespace Configs
    - Backwards Compatibility
    - Handling Rolling Upgrade
    - Handling Regexes in Topic Subscription
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives

Status

Current state: *"Under Discussion"*

Discussion thread: *here*

JIRA: *KAFKA-2630*

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

Motivation

Apache Kafka is rapidly finding its place in data heavy organizations as a fault-tolerant message bus. One of the goals of Kafka is data integration, which makes it important to support many users in one Kafka system. With increasing adoption and user community, support for multi-tenancy is becoming a popular demand. There have been a few discussions on Apache Kafka's mailing lists regarding the same, indicating importance of the feature. Namespaces will allow/ enable many functionalities that require logical grouping of topics. If you think topic as a SQL table, then namespace is a SQL database that lets you group tables together. Following are a few use cases.

- Namespaces will allow users to create topics with same name as long as they are part of different namespaces.
- No need to set configs for each topic individually
- Allow bootstrapping any new entity in a namespace with some default configs, which is set for that particular namespace, and then letting each entity override parts of that config.
- Similar to configs, acls can be set at namespace level.
- When Kafka decides to support at rest encryption, having namespaces at logs level will allow encrypting different namespaces with different keys.

Public Interfaces

Briefly list any new interfaces that will be introduced as part of this proposal or any existing interfaces that will be removed or changed. The purpose of this section is to concisely call out the public contract that will come along with this feature.

A public interface is any change to the following:

- *Binary log format*
    - No changes expected
- *The network protocol and api behavior*
    - No changes expected
- Any class in the public packages under clientsConfiguration, especially client configuration
    - org/apache/kafka/common/serialization
        - No changes expected
    - org/apache/kafka/common
        - No changes expected
    - org/apache/kafka/common/errors

        - InvalidNamespaceException will be added to indicate the namespace user is trying to use does not exist.
    - org/apache/kafka/clients/producer
        - No changes expected
    - org/apache/kafka/clients/consumer (eventually, once stable)
        - No changes expected
- *Monitoring*
- *Command line tools and arguments*
    - Add create, list, move and delete for namespaces to *kafka-topics* and *AdminUtils*.
    - An optional *"namespace"* argument will be added to *kafka-topics, kafka-configs and kafka-acls*. If *namespace* argument is not provided, a default namespace of "" will be used. This will help in keeping the current behavior of Kafka and cli tools intact.
- *Anything else that will likely break existing users in some way when they upgrade*
    - None

Proposed Changes

After considering a few approaches, listed in Rejected Alternatives (not really rejected, up for discussion) section, below is what we think is the least obtrusive approach to support namespaces in Kafka. We suggest to represent namespaces at storage layer, i.e., storage layout of Zookeeper entities and logs on disk. Internal and public APIs can pass around namespaces, as part of, prepended to, topic names. However, we need to separate namespace and topic while interacting with storage layers. This can be done by using a delimiter that is not allowed in Kafka topics. Currently, Kafka allows a topic name to contain characters only in [a-zA-Z0-9\\._\\-]. That gives us a few options to decide on the delimiting char. We suggest to have ":" as the delimiting char, but it can be any of the following.

Possible Delimiters

1. <namespace>:<topic>
2. <namespace>|<topic>
3. <namespace>#<topic>
4. <namespace>%<topic>
5. <namespace>@<topic>
6. <namespace>*<topic>
7. <namespace>~<topic>
8. <namespace>$<topic>
9. <namespace>^<topic>
10. <namespace>&<topic>
11. <namespace>><topic>

Multi-tiered Namespaces

Namespace can have any char in [a-zA-Z0-9\\.\\-]. The ":" in namespaces will be used to separate tiers. For instance, a namespace "org:apache:kafka" will translate to "org/apache/kafka" in storage layouts.

Create

kafka-topics will be modified to support creating namespaces, as shown below.

```
kafka-topics.sh --zookeeper <ZK_CONNECTION_STRING> --create --namespace <NAMESPACE>
```

Creating namespace will be **required** before it can be used. There is no plan of supporting auto namespace creation. If a multi-tiered namespace is created, it will create all the tiers specified in the namespace. For instance, the command below will create three namespaces, i.e., org, org:apache and org:apache:kafka.

```
kafka-topics.sh --zookeeper <ZK_CONNECTION_STRING> --create --namespace org:apache:kafka
```

List

kafka-topics will be modified to support listing namespaces, as shown below.

```
kafka-topics.sh --zookeeper <ZK_CONNECTION_STRING> --list --namespace <NAMESPACE>
```

The above command will list all the namespaces under the namespace specified, <NAMESPACE>.

Delete

kafka-topics will be modified to support deleting a namespace, as shown below.

```
kafka-topics.sh --zookeeper <ZK_CONNECTION_STRING> --delete --namespace <NAMESPACE>
```

The above command will delete all topics under <NAMESPACE>. However, if there is any namespace under <NAMESPACE>, then the command will throw a warning that there are namespaces within <NAMESPACE> and die. However, if the above command is run with a *recursive* option, as shown below, then all namespaces within will be deleted along with current <NAMESPACE>.

```
kafka-topics.sh --zookeeper <ZK_CONNECTION_STRING> --delete --namespace <NAMESPACE> --recursive
```

Note that the delete will essentially mark topics and namespaces for deletion, the same logic that exists today for deletion.

Move

kafka-topics will be modified to add support for moving a topic from one namespace to another namespace, as shown below.

```
kafka-topics.sh --zookeeper <ZK_CONNECTION_STRING> --move --from-namespace <NAMESPACE> --to-namespace
<NAMESPACE>
```

Changes in Storage Layouts

Following are the proposed changes in layout structures persisted on ZK and disks.

Type	Existing layout	Proposed layout
ZK	*/brokers/topics/<topic>*	*/brokers/topics/<namespace>/<topic>*
ZK	*/configs/<entity_name>/<entity>*	*/configs/<namespace>/<entity_name>/<entity>*
ZK	*/admin/delete_topics/<topic>*	*/admin/delete_topics/<namespace>/<topic>*
ZK	*/kafka_acls/<entity_name>/<entity>*	*/kafka_acls/<namespace>/<entity_name>/<entity>*
Disk	*/<log_dir>/<topic>_<partition>*	*/<log_dir>/<namespace>/<topic>_<partition>*

By default, namespace will be empty string. All existing entities will be part of default namespace and the current storage layouts will be in accordance with the proposed storage layouts.

Namespace Acls

kafka-acls will be modified to support namespace parameter, which if specified will let users specify acls for a namespace. Prior to checking acls for any entity in a namespace, permission to access the particular namespace will be checked.

Namespace Configs

Currently, Kafka lets you provide cluster and topic level configs. However, for a multi-tenant Kafka cluster default configs for different namespaces could be different. We propose to allow configs be specified at namespace level. If a particular config is not specified at a topic level, but is specified at a namespace level, config value from namespace level should be honored. *kafka-configs* will need to be modified to support that.

Backwards Compatibility

The goal here is to make sure that any existing topic, producers and consumers, without any namespace, continues to work as expected. All topics under /broker/topics will be part of default namespace, i.e., "". Any topic created without specifying a namespace will be part of the default namespace. As long as users do not specify namespaces in their request or cli commands, things should work just as before.

Handling Rolling Upgrade

We need a way to be able to stop users from creating a namespace after upgrading a subset of brokers, not all brokers. This will probably lead to inconsistent behavior. One easy way to handle this is to doc it.

Handling Regexes in Topic Subscription

Regexes should work fine, however users might have to modify their existing regexes based on topics and namespaces that exist on the cluster. If someone, like mirror-maker, is subscribing to ".*", then this will just work fine and nothing has to be changed. However, if user has regex like "bla*" and the user later creates a namespace "bla", then they will get subscribed to topics in namespace "bla" that they were probably not expecting.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*

    - Handling regexes in topic subscription, as explained in previous section.
- *If we are changing behavior how will we phase out the older behavior?*
    - No need to phase out old behavior.
- *If we need special migration tools, describe them here.*
    - NA
- *When will we remove the existing behavior?*
    - NA

Rejected Alternatives

Following are a few alternatives that we considered.

1. Just prepend namespace to topic names, even at storage layer, instead of creating a hierarchy of directories to represent namespaces.
    a. Will not enable encrypting namespaces with different keys.
    b. Namespace level configs or acls won't be possible without creating yet another znode in Kafka's chroot in ZK. I do not think having */kafka-namespaces/<namespace>* will help in alleviating the confusion caused by passing namespace and topic name together in APIs, as we will still have to represent multi-tiered namespaces under */kafka-namespaces* as dirs.
2. Modify request/ response formats to take namespace specifically.
    a. Solves the issue of delimiting string required in proposed approach and the issue with existing regex consumers.
    b. This definitely is the cleanest approach. However, will require lots of API and protocol changes. This is listed in rejected alternatives, but we should totally consider this as an option.
3. Manage namespaces separately.
    a. This will still have the issue of topic name collisions even if they belong to separate namespaces.
4. Add namespace to session object.
    a. Will avoid each request and response to have namespace with topic name, however this probably is violating separation of concerns.
5. To have delimiter char configurable.
    a. Will add yet another config, without any clear gain.