

KIP-115: Enforce `offsets.topic.replication.factor` upon `__consumer_offsets` auto topic creation

- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Rejected Alternatives](#)

Status

Current state: Accepted

Discussion thread: [here](#)

JIRA: [here](#)

Released: 0.10.3.0

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


Motivation

This KIP aims to enforce `offsets.topic.replication.factor` upon `__consumer_offsets` auto topic creation.

Kafka brokers have a config called `offsets.topic.replication.factor` that specify the replication factor for the `__consumer_offsets` topic. The problem is that this config isn't being enforced upon auto topic creation. If an attempt to auto create the internal topic is made when there are fewer brokers than `offsets.topic.replication.factor`, the topic ends up getting created anyway with the current number of live brokers. The current behavior is pretty surprising when you have clients or tooling running as the cluster is getting setup. Even if your cluster ends up being huge, you'll find out much later that `__consumer_offsets` was setup with no replication.

The cluster not meeting the `offsets.topic.replication.factor` requirement on the internal topic is another way of saying the cluster isn't fully setup yet. The right behavior should be for `offsets.topic.replication.factor` to be enforced.

Rationale for the prior behavior can be found in

 Unable to render Jira issues macro, execution error.

Public Interfaces

Set the `offsets.topic.replication.factor` to 1 in `config/server.properties` to maintain existing single-broker quickstart behavior. Note that this does not change the default `offsets.topic.replication.factor` value of 3 in `KafkaConfig`.

Proposed Changes

Internal topic creation can happen in five paths:

1. By a broker upon `GroupCoordinatorRequest`.
2. By a broker from `MetadataRequest` querying the internal topic even if `auto.create.topics.enable` is false.
3. By a user when using `AdminUtils`.
4. By a user when running `kafka-topics.sh` (which calls `AdminUtils`).
5. By a broker through `AdminManager` (which calls `AdminUtils`) handling `CreateTopicsRequest`.
6. By a user directly writing to the topic znode in zookeeper.

Consequences of this KIP:

1. will now fail topic creation of the internal topic with `GROUP_COORDINATOR_NOT_AVAILABLE` until the `offsets.topic.replication.factor` requirement is met.
2. will now fail topic creation of the internal topic and retain existing behavior of failing topic creation with `INVALID_REPLICATION_FACTOR` until the `offsets.topic.replication.factor` requirement is met.
3. will retain existing behavior. `AdminUtils` compares cluster size vs. replication factor comparison and throws an `InvalidReplicationFactorException` if the manually specified replication factor isn't met. If the replication factor is met, it creates the topic, ignoring the broker's `offsets.topic.replication.factor` config.

4. Same as 3.
5. Same as 3. `CreateTopicsResponse` including an internal topic will return `INVALID_REPLICATION_FACTOR` if the manually specified replication factor isn't met.
6. is unrelated, as the zookeeper write will not receive any error from kafka.

Compatibility, Deprecation, and Migration Plan

This is a bug fix KIP impacting the setup of new clusters. Users setting up a cluster should keep in mind that the `__consumer_offsets` topic will not be created until their cluster satisfies the `offsets.topic.replication.factor`.

Rejected Alternatives

One rejected alternative was to push `__consumer_offsets` topic creation logic out of the brokers and into the `KafkaController`. Since the `KafkaController` can detect broker membership changes through zookeeper, it can create the `__consumer_offsets` topic as soon as the `offsets.topic.replication.factor` is met. While doable, it is more complicated as it would add even more logic to keep track of in the already complicated `KafkaController` and would additionally require `KafkaApis` to instead lookup cluster readiness based on its `MetadataCache` when responding to `GroupCoordinatorRequests`.