KIP-885: Expose Broker's Name and Version to Clients

- Status
- Motivation
- Public Interfaces
  - DescribeCluster API
  - AdminClient API
- Proposed Changes
  - Validation
- Compatibility, Deprecation, and Migration Plan
- Rejected Alternatives
  - Put BrokerSoftwareName and BrokerSoftwareVersion in the ResponseHeader
  - Put BrokerSoftwareName and BrokerSoftwareVersion in the ApiVersions response
  - Return the name and version per broker in the DescribeCluster response
  - Put BrokerSoftwareName and BrokerSoftwareVersion in the ResponseHeader but send it only once
  - Add a new request to communicate the broker metadata to the client
  - Have BrokerSoftwareName and BrokerSoftwareVersion be tags
  - Have BrokerSoftwarename and BrokerSoftwareVersion be tags, and add an IncludeSoftwareNameVersion tag to ApiVersionsRequest

Status

Current state: *Under Discussion*

Discussion thread: *here*

JIRA: *here*

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

Motivation

A Kafka broker has never had the ability to tell clients what version of Kafka it is. Today, at best, a client can use ApiVersions to determine the minimum and maximum supported version for all supported requests per broker. Since Kafka 0.8.0, every release, at least one request key was added or one request's version changed. Clients could use the response from ApiVersions to "guess" the version of Kafka they are talking to. So far for Kafka 3.4, no new request has been added and no existing request has bumped its version. If released today, clients will no longer be able to correctly guess which major/minor version of Kafka they are talking to. As well, for clients talking to other Kafka-like replacements (AWS Kinesis, Microsoft Event Hubs, Redpanda), it has been impossible to accurately guess the version because these systems support different sets of requests and request versions.

End users have found this version guessing very useful. For example, any UX management layer on top of Kafka that does not hook into Kafka metrics directly (AKHQ, Burrow, CMAK, Conduktor, Kafdrop, etc.), can use the version guess to display each broker's version (or cluster version, which could be the minimum or maximum of all brokers) in the UX. Clients can also use the broker version in log lines so that end users can quickly see if a version looks about right or if something is seriously broken.

The version is useful even if the API surface area does not change. End users can quickly check *from a client* if the cluster is up to date, or if they may be exposed to bugs / problems from older broker versions. This is yet another area that users can ensure their systems are what they expect them to be.

This KIP is the broker-side complement of KIP-511.

Public Interfaces

DescribeCluster API

Bump DescribeClusterRequest and DescribeClusterResponse to v1 and add two new required fields, BrokerSoftwareName and BrokerSoftwareVersion, to the response.

ApiVersionsResponse.json

```
{
  "apiKey": 60,
  "type": "response",
  "name": "DescribeClusterResponse",
  "validVersions": "0-1", // BUMPED TO VERSION 1
  "flexibleVersions": "0+",
  "fields": [
    { "name": "ThrottleTimeMs", "type": "int32", "versions": "0+",
      "about": "The duration in milliseconds for which the request was throttled due to a quota violation, or
zero if the request did not violate any quota." },
    { "name": "ErrorCode", "type": "int16", "versions": "0+",
      "about": "The top-level error code, or 0 if there was no error" },
    { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+", "default": "null",
      "about": "The top-level error message, or null if there was no error." },
    { "name": "ClusterId", "type": "string", "versions": "0+",
      "about": "The cluster ID that responding broker belongs to." },
    { "name": "ControllerId", "type": "int32", "versions": "0+", "default": "-1", "entityType": "brokerId",
      "about": "The ID of the controller broker." },
    { "name": "Brokers", "type": "[]DescribeClusterBroker", "versions": "0+",
      "about": "Each broker in the response.", "fields": [
      { "name": "BrokerId", "type": "int32", "versions": "0+", "mapKey": true, "entityType": "brokerId",
        "about": "The broker ID." },
      { "name": "Host", "type": "string", "versions": "0+",
        "about": "The broker hostname." },
      { "name": "Port", "type": "int32", "versions": "0+",
        "about": "The broker port." },
      { "name": "Rack", "type": "string", "versions": "0+", "nullableVersions": "0+", "default": "null",
        "about": "The rack of the broker, or null if it has not been assigned to a rack." }
    ]},
    // -------------- START NEW FIELD --------------
    { "name": "BrokerSoftwareName", "type": "string", "versions": "1+", "nullableVersions": "1+", "ignorable":
true,
      "about": "The name of the broker replying to this request." },
    { "name": "BrokerSoftwareVersion", "type": "string", "versions": "1+", "nullableVersions": "1+",
"ignorable": true,
      "about": "The version of the broker replying to this request." }
    // -------------- END NEW FIELD --------------
    { "name": "ClusterAuthorizedOperations", "type": "int32", "versions": "0+", "default": "-2147483648",
      "about": "32-bit bitfield to represent authorized operations for this cluster." }
  ]
}
```

AdminClient API

The DescribeClusterResponse class will be updated to add two methods:

```
public class DescribeClusterResponse extends AbstractResponse {
  // ...

  public String brokerSoftwareName() {
    return data.brokerSoftwareName();
  }

  public String brokerSoftwareVersion() {
    return data.brokerSoftwareVersion();
  }
}
```

Proposed Changes

This KIP proposes two new required fields in the DescribeCluster response to permanently remove the need to "guess" a version based on the ApiVersions response, and to add the ability to determine the name of the software the client is talking to. For Kafka, the idea is to return "Kafka" for the BrokerSoftwareName, and CURRENT_BROKER_VERSION for the BrokerSoftwareVersion. For other Kafka-like replacements, they can return something like "Kinesis" and ${kinesis_version}, or "Redpanda" and ${redpanda_version}.

Validation

Each system may name their software differently. Kafka, Kinesis, and Redpanda all use one word, while Event Hubs uses two. Optionally, Kinesis could be displayed as "Amazon Kinesis", and Event Hubs could be displayed as "Microsoft Event Hubs". This proposal does not prescribe any validation to the name field.

Each system may version their software differently. Kafka uses \d.\d, with the patch version largely being invisible to end users. Redpanda uses vYY.\d, with the patch version somewhat being invisible to end users (though it is relatively more important). It's unclear what versioning scheme Kinesis or Event Hubs uses, if any. Because each system may version uniquely, this proposal does not prescribe any validation to the version field.

Compatibility, Deprecation, and Migration Plan

N/A, only new clients will use these new fields. This does not change the behavior of existing clients.

Rejected Alternatives

Put BrokerSoftwareName and BrokerSoftwareVersion in the ResponseHeader

One alternative is mentioned to parallel KIP-511, but there is even less reason to include the broker name and version in every response. Clients are not interested in these pieces of information on *every* response, so we should not require it in every response. The DescribeCluster response is the perfect place to put these fields: only interested clients will ask for the cluster metadata. Clients never need to know the broker name nor version otherwise.

Put BrokerSoftwareName and BrokerSoftwareVersion in the ApiVersions response

This KIP originally proposed placing these two new fields in ApiVersions. After discussion, we want to avoid the risk of the BrokerSoftwareVersion being misused by clients which opt into or out of behavior. The easiest way to avoid misuse is to simply not make misuse possible. As well, the average producer & consumer is not interested in the broker name nor version. We should only return these fields in a less-frequently-used, more admin request.

Return the name and version per broker in the DescribeCluster response

The broker software name and broker software version are not available centrally. These are going to be per-broker fields. A single broker cannot return the software name and version for all other brokers; if a client is interested in the name and version for all brokers, the client must request all brokers.

Put BrokerSoftwareName and BrokerSoftwareVersion in the ResponseHeader but send it only once

Again paralleling KIP-511, these fields could be sent only in the first ResponseHeader to save bytes in subsequent requests. Lik KIP-511, it would be weird to have the information in random requests and it would make clients inconsistent.

Add a new request to communicate the broker metadata to the client

A new separate request/response could be used for the purpose. This is a valid option. This KIP considers the extra bit of information small enough, and the ApiVersions request infrequently enough, such that it's worth it to add these fields to ApiVersions.

Have BrokerSoftwareName and BrokerSoftwareVersion be tags

This would add no overhead if a broker does not want to tell the client its version. However, if a broker does not want to tell the client of its version, this defeats the purpose of the KIP. The broker should always tell the client of its version, thus there is no reason for these fields to be tags.

Have BrokerSoftwarename and BrokerSoftwareVersion be tags, and add an IncludeSoftwareNameVersion tag to ApiVersionsRequest

This solves the prior point: the response can be optional, but if the client asks, the broker should return the fields. Tag support is up to the broker, so it is possible that a client will ask and will not receive a response. As well, from a client implementation perspective, it is easier to just always ask for these fields, and it does not add much to the response. I worry that brokers will look like they support the full protocol, but then will not support this tag and it will be impossible for clients to determine what version of broker they are talking to.