KIP-900: KRaft kafka-storage.sh API additions to support SCRAM for Kafka Brokers

Status

Current state: *Accepted*

Discussion thread: DISCUSS, DISCUSS+VOTE

JIRA: Unable to render Jira issues macro, execution error. Unable to render Jira issues macro, execution error.

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

Motivation

Today, we can use SCRAM authentication for Kafka Brokers when the cluster uses ZooKeeper (ZK) for the quorum servers. This is possible to bootstrap by first setting up the ZK servers and setting the inter-broker communication password by directly updating them on the ZK before the Kafka Brokers are started. See Configuring SCRAM for details. We wish to implement some similar mechanism for storing the Kafka Broker authentication credentials for SCRAM when the cluster uses KRaft for the quorum servers. We want these credentials to be stored before the Kafka cluster starts for the first time.

Proposed Changes

We will update the `kafka-storage` tool to take SCRAM credentials and store them when we format each node.

The `kafka-storage` tool is used to initialize the storage space for each broker and controller. One of the files created is the `bootstrap.checkpoint` which contains a set of `ApiMessageAndVersion` records that are needed for the bootstrap process of the cluster. I am proposing we add an interface to the kafka-storage tool so we can add the needed SCRAM credential updates, `UserScramCredentialsRecord` which is an `ApiMessageAndVersion` record, when the storage is formatted. With this, when the initial start of the cluster happens, each broker will have a copy of the SCRAM server side credential needed so they can communicate with each other.

I propose we add an option `--add-scram` to the `kafka-storage` tool that will add an `ApiMessageAndVersion` records directly into the `__cluster_metadata` topic. (See KIP-801 for details on `__cluster_metadata`). The option can be applied multiple times to the format command so that multiple SCRAM records can be added to bootstrap the cluster. Below is the new updates `kafka-storage` command.

```
./bin/kafka-storage.sh format -h
usage: kafka-storage format [-h] --config CONFIG --cluster-id CLUSTER_ID [--add-scram SCRAM_CREDENTIAL] [--
release-version RELEASE_VERSION] [--ignore-formatted]

optional arguments:
  -h, --help            show this help message and exit
  --config CONFIG, -c CONFIG
                        The Kafka configuration file to use.
  --cluster-id CLUSTER_ID, -t CLUSTER_ID
                        The cluster ID to use.
  --add-scram SCRAM_CREDENTIAL, -S SCRAM_CREDENTIAL
                        A SCRAM_CREDENTIAL to add to the __cluster_metadata log e.g.
                        'SCRAM-SHA-256=[user=alice,password=alice-secret]'
                                             'SCRAM-SHA-512=[user=alice,iterations=8192,salt="
MWx2NHBkbnc0ZndxN25vdGN4bTB5eTFrN3E=",saltedpassword="mT0yyUUxnlJaC99HXgRTSYlbuqa4FSGtJCJfTMvjYCE="]'
  --release-version RELEASE_VERSION, -r RELEASE_VERSION
                        A KRaft release version to use for the initial metadata version.
  --ignore-formatted, -g
```

I propose the SCRAM_CREDENTIAL argument will be a key value pair where the key is one of the SCRAM mechanisms supported, either SCRAM-SHA-256 or SCRAM-SHA-512, and the value is a set of key value pairs of parameters to populate the `UserScramCredentialsRecord`. The SCRAM_CREDENTIAL argument is very similar to the argument passed to the `kafka-config` tool for configuring SCRAM in a ZK cluster. See Configuring SCRAM for details.

The subarguments for the SCRAM_CREDENTIAL require a "user" key and either a "password" key or a "saltedpassword" key. If using the "saltedpassword" key you must also supply an "iteration" key and a "salt" key. The "iteration" and "salt" key are otherwise optional. However if they are not supplied, "iteration" count will default to 4096 and the "salt" will be randomly generated. The value for "salt" and "saltedpassword" is base64 encoding of binary data.

I propose to also add support for the argument parsing to include taking a file of arguments. This is a standard Argparse4j feature and will make it easier to bootstrap brokers and controllers with multiple SCRAM_CREDENTIALS. I propose to use the '@' character to proceed the filename argument which contains additional arguments. See the Argparse4j fromfileprefix manual entry for details.

Compatibility, Deprecation, and Migration Plan

These changes are additions to the `kafka-storage` tool only. There are no backwards compatibility issues with this change.

Test Plan

These changes are to the bootstrap of a cluster and have no impact of existing clusters at all.

Rejected Alternatives

1. The original proposal was for adding raw records to the __cluster_metadata. These records would be describe as a single key value where the key is the name of the ApiMessageAndVersion record and the value a JSON encoding of the fields for that record. An example would be the following for a SCRAM credential record.

   ```
   UserScramCredentialsRecord={"name":"alice","mechanism":1,"salt":"MWx2NHBkbnc0ZndxN25vdGN4bTB5eTFrN3E=","
   SaltedPassword":"mT0yyUUxnlJaC99HXgRTSYlbuqa4FSGtJCJfTMvjYCE=","iterations":8192}'
   ```

   This proposal is not very customer friendly as the customer now needs to know what the fields of the underlying `UserScramCredentialsRecord` are. It also has the issue that the underlying record format could change in the future requiring the command line to change. It is desired that even if the record format changes, the argument parsing shouldn't be affected.
2. Update `kafka-config` to take a format directory option and use the same arguments for altering SCRAM credentials to add them to the `__cluster_metadata` topic for bootstrap. The issues with this is that it requires multiple commands to format each node in a cluster. It also has the problem of adding a whole new block of code to `kafka-config` just to handle the `bootstrap.checkpoint` file and it would need logic to understand if the bootstrap had completed.
3. Update `kafka-storage` to append records to bootstrap.checkpoint with multiple invocations of the tool. This would allow the use of the same command line arguments from `kafka-config` to be used. It was deemed a requirement that a single invocation of `kafka-storage` format all the records for bootstrap.