KIP-554: Add Broker-side SCRAM Config API

Status

Current state: *Accepted*

Discussion thread: here

JIRA: 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

The *kafka-configs.sh* command supports changing the SCRAM settings for users. For example:

```
bin/kafka-configs --zookeeper localhost:2181 \
    --alter \
    --entity-type users \
    --entity-name alice \
    --add-config 'SCRAM-SHA-256=[iterations=8192,password=alice-secret],SCRAM-SHA-512=[password=alice-secret]'
```

However, this command depends on Apache ZooKeeper. There is no broker-based API to change these settings.

As part of the KIP-500 effort to remove Kafka's ZooKeeper dependency, we need a broker-side API to alter these settings.

Public Interfaces

describeUserScramCredentials

describeUserScramCredentials will describe the currently configured SCRAM user credentials. For security reasons, it will not list their passwords.

```
public enum ScramMechanism {
    UNKNOWN(0),
    SCRAM_SHA_256(1),
    SCRAM_SHA_512(2);

    private final byte type;
    private final String mechanismName;

    public static ScramMechanism fromType(byte type) {
        // etc...
    }

    public static ScramMechanism fromMechanismName(String mechanismName) {
        // etc...
    }

    public String mechanismName() {
        // etc...
    }

    public byte type() {
        // etc...
    }

    private ScramMechanism(byte type) {
        // etc...
    }
}

public class ScramCredentialInfo {
    private final ScramMechanism mechanism;
    private final int iterations;
}

public class UserScramCredentialsDescription {
    private final String name;
    private final List<ScramCredentialInfo> infos;
}

public class DescribeScramUserCredentialsOptions extends AbstractOptions<DescribeScramUserCredentialsOptions> {
}

// interface Admin: Describe all users.
default DescribeScramUserCredentialsResult describeScramUserCredentials() {
    return describeScramUserCredentials(null);
}

// interface Admin: Describe indicated users (null or empty implies all).
default DescribeScramUserCredentialsResult describeScramUserCredentials(List<String> users) {
    return describeScramUserCredentials(users, new DescribeScramUserCredentialsOptions());
}

// interface Admin
DescribeScramUserCredentialsResult describeScramUserCredentials(List<String> users,
DescribeScramUserCredentialsOptions options);

public class DescribeScramUserCredentialsResult {
    // completes successfully only if users() does and every described user does
    public KafkaFuture<Map<String, UserScramCredentialsDescription>> all();

    // completes successfully if request was authorized and the list of users described is determined
    public KafkaFuture<List<String>> users();

    // completes successfully if the individual user is described successfully
    public KafkaFuture<UserScramCredentialsDescription> description(String userName)
}
```

describeScramUserCredentials will be implemented by a new RPC.

```
{
  "apiKey": 50,
  "type": "request",
  "name": "DescribeUserScramCredentialsRequest",
  "validVersions": "0",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "Users", "type": "[]UserName", "versions": "0+", "nullableVersions": "0+",
      "about": "The users to describe, or null/empty to describe all users.", "fields": [
      { "name": "Name", "type": "string", "versions": "0+",
        "about": "The user name." }
    ]}
  ]
}

{
  "apiKey": 50,
  "type": "response",
  "name": "DescribeUserScramCredentialsResponse",
  "validVersions": "0",
  "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 message-level error code, 0 except for user authorization or infrastructure issues." },
    { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+",
      "about": "The message-level error message, if any." },
    { "name": "Results", "type": "[]DescribeUserScramCredentialsResult", "versions": "0+",
      "about": "The results for descriptions, one per user.", "fields": [
      { "name": "User", "type": "string", "versions": "0+",
        "about": "The user name." },
      { "name": "ErrorCode", "type": "int16", "versions": "0+",
        "about": "The user-level error code." },
      { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+",
        "about": "The user-level error message, if any." },
      { "name": "CredentialInfos", "type": "[]CredentialInfo", "versions": "0+",
        "about": "The mechanism and related information associated with the user's SCRAM credentials.",
"fields": [
        { "name": "Mechanism", "type": "int8", "versions": "0+",
          "about": "The SCRAM mechanism." },
        { "name": "Iterations", "type": "int32", "versions": "0+",
          "about": "The number of iterations used in the SCRAM credential." }]}
    ]}
  ]
}
```

It will require DESCRIBE permissions on the CLUSTER resource. It will return CLUSTER_AUTHORIZATION_FAILED if the user has insufficient permissions.

It will return RESOURCE_NOT_FOUND if a user is requested to be described but that user does not exist/has no credentials. Note that DescribeScramUserCredentialsResult **will not** consider such a username to be part of its `users()` result list, and this RESOURCE_NOT_FOUND error by itself **will not** prevent the future returned by `all()` from completing successfully.

It will return DUPLICATE_RESOURCE if a user is requested to be described twice. Note that DescribeScramUserCredentialsResult **will** consider such a username to be part of its `users()` result list, and this **will** cause the future returned by `all()` to complete exceptionally.

AlterScramUserCredentials

alterScramUserCredentials will create or change SCRAM user credentials.

Alterations will create the given user credential if it doesn't exist, or alter it if it does.

```
public abstract class UserScramCredentialAlteration {
    private final String user;
}

public class UserScramCredentialUpsertion extends UserScramCredentialAlteration {
    private final ScramCredentialInfo info;
    private final byte[] salt;
    private final byte[] password;

    // There will be one constructor that randomly generates a salt, and one that accepts a pre-defined salt.
}

public class UserScramCredentialDeletion extends UserScramCredentialAlteration {
    private final ScramMechanism mechanism;
}

public class AlterScramUserCredentialsOptions extends AbstractOptions<AlterScramUserCredentialsOptions> {}

// interface Admin
default AlterScramUserCredentialsResult alterScramUserCredentials(List<UserScramCredentialAlteration>
alterations) {
    return alterScramUserCredentials( alterations, new AlterScramUserCredentialsOptions());
}

// interface Admin
AlterScramUserCredentialsResult alterScramUserCredentials(List<UserScramCredentialAlteration> alterations,
                                    AlterScramUserCredentialsOptions options);

public class AlterUserScramCredentialsResult {
    public Map<String, KafkaFuture<Void>> values();
    public KafkaFuture<Void> all(); // completes successfully only if everything in values() does
}
```

If any of the operations associated with a single user can't be done, none of the operations will be done. On the other hand, operations could succeed for one user but fail for another, different user.

If a user doesn't exist and we add a credential for them, they will be created. If a user exists and we remove their last credential, they will be deleted.

The AlterScramUserCredentialsRequest and AlterScramUserCredentialsResponse implement the new API.

```
{
  "apiKey": 51,
  "type": "request",
  "name": "AlterUserScramCredentialsRequest",
  "validVersions": "0",
  "flexibleVersions": "0+",
  "fields": [
    { "name": "Deletions", "type": "[]ScramCredentialDeletion", "versions": "0+",
      "about": "The SCRAM credentials to remove.", "fields": [
      { "name": "Name", "type": "string", "versions": "0+",
        "about": "The user name." },
      { "name": "Mechanism", "type": "int8", "versions": "0+",
        "about": "The SCRAM mechanism." }
    ]},
    { "name": "Upsertions", "type": "[]ScramCredentialUpsertion", "versions": "0+",
      "about": "The SCRAM credentials to update/insert.", "fields": [
      { "name": "Name", "type": "string", "versions": "0+",
        "about": "The user name." },
      { "name": "Mechanism", "type": "int8", "versions": "0+",
        "about": "The SCRAM mechanism." },
      { "name": "Iterations", "type": "int32", "versions": "0+",
        "about": "The number of iterations." },
      { "name": "Salt", "type": "bytes", "versions": "0+",
        "about": "A random salt generated by the client." },
      { "name": "SaltedPassword", "type": "bytes", "versions": "0+",
        "about": "The salted password." }
    ]}
  ]
}

{
  "apiKey": 51,
  "type": "response",
  "name": "AlterUserScramCredentialsResponse",
  "validVersions": "0",
  "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": "Results", "type": "[]AlterUserScramCredentialsResult", "versions": "0+",
      "about": "The results for deletions and alterations, one per affected user.", "fields": [
      { "name": "User", "type": "string", "versions": "0+",
        "about": "The user name." },
      { "name": "ErrorCode", "type": "int16", "versions": "0+",
        "about": "The error code." },
      { "name": "ErrorMessage", "type": "string", "versions": "0+", "nullableVersions": "0+",
        "about": "The error message, if any." }
    ]}
  ]
}
```

An addition will return UNACCEPTABLE_CREDENTIAL if an empty user name or an invalid number of iterations (less than the minimum required for the mechanism or more than a hard-coded max of 16,384) is passed.

DUPLICATE_RESOURCE will be returned if a user appears as both an upsertion and a deletion in the same request.

UNSUPPORTED_SASL_MECHANISM will be returned if the broker does not recognize the requested SASL mechanism.

A removal will return an error code, RESOURCE_NOT_FOUND, if it was instructed to delete a credential that did not exist.

The RPC will require ALTER on CLUSTER. It will return CLUSTER_AUTHORIZATION_FAILED if the user has insufficient permissions.

It will be will be sent to the controller and will return NOT_CONTROLLER if the receiving broker is not the controller.

Command-Line Changes

We will extend the *kafka-configs.sh* command to so that it is possible to set a SCRAM configuration without using *--zookeeper*. The command-line syntax will be unchanged, except for the fact that users will now be able to pass *--bootstrap-server* instead of *--zookeeper*.

As mentioned earlier, this API does not return secrets. Therefore, the salt, salted password, and so on will not be returned by a kafka-configs.sh --describe operation. The describe operation will return only the presence of the user plus the algorithm and number of iterations used. For example:

```
$ bin/kafka-configs.sh --bootstrap-server localhost:9020 \
    --alter \
    --entity-type users \
    --entity-name alice \
    --add-config 'SCRAM-SHA-256=[iterations=8192,password=alice-secret],SCRAM-SHA-512=[password=alice-secret]'

Completed updating config for entity: user-principal 'alice'.

$ bin/kafka-configs.sh --bootstrap-server localhost:9020 --entity-type users --entity-name alice --describe

Configs for user-principal 'alice' are SCRAM-SHA-512=iterations=8192
```

Compatibility, Deprecation, and Migration Plan

There is no impact on compatibility, since this adds a new API that didn't exist before.

Rejected Alternatives

Extend IncrementalAlterConfigs to handle SCRAM

Rather than creating new RPCs, we could have extended the IncrementalAlterConfigs RPC to support changing SCRAM configurations. However, this would involve some fairly complex string manipulation for clients that wanted to use the API. It would also be more cumbersome to list the SCRAM users.