KIP-50 - Move Authorizer to o.a.k.common package

Status

Current state: *Voted on, but never adopted. Superseded by KIP-504*

Discussion thread: *here [Change the link from the KIP proposal email archive to your own email thread]*

JIRA: *KAFKA-3186*

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

Motivation

Kafka supports pluggable authorization. Third party authorizer implementations allow existing authorization systems like, Apache Sentry, Apache Ranger, etc to extend authorization to Kafka as well. Implementing Kafka's authorizer interface requires depending on kafka's core, which is huge. This has been already raised as a concern by Sentry, Ranger and Kafka community. Even Kafka clients require duplication of authorization related classes, like Resource, Operation, etc, for adding ACLs CRUD APIs.

Kafka authorizer is agnostic of principal types it supports, so are the acls CRUD methods in Authorizer interface. The intent behind is to keep Kafka principal types pluggable, which is really great. However, this leads to Acls CRUD methods not performing any check on validity of acls, as they are not aware of what principal types Authorizer implementation supports. This opens up space for lots of user errors, KAFKA-3097 is an instance.

Public Interfaces

Move Authorizer interfaces and related classes to common module.

Following interface and classes will be moved to `org.apache.kafka.common.security.auth`.

1. `Authorizer`
2. `Acl`
3. `Operation`
4. `PermissionType`
5. `Resource`
6. `ResourceType`
7. `Session`

Add method to get authorizer implementation specific description to `Authorizer` interface.

```
/**
* description of authorizer implementation, like, valid principal types.
* @return description of authorizer implementation.
*/
public String description()
```

Add exceptions related to Authorizer.

Following exceptions will be added to `org.apache.kafka.common.errors`.

InvalidAclException

```
/**
 * Throw when an invalid Acl is being added or removed.
 */
public class InvalidAclException extends AuthorizationException {

    private static final long serialVersionUID = 1L;

    public InvalidAclException(String message) {
        super(message);
    }

    public InvalidAclException(String message, Throwable cause) {
        super(message, cause);
    }

}
```

InvalidOperationException

```
/**
 * Throw when an invalid operation is being performed on a resource.
 */
public class InvalidOperationException extends AuthorizationException {

    private static final long serialVersionUID = 1L;

    public InvalidOperationException(String message) {
        super(message);
    }

    public InvalidOperationException(String message, Throwable cause) {
        super(message, cause);
    }

}
```

InvalidPrincipalException

```
/**
 * Throw when an invalid principal is provided by user.
 */
public class InvalidPrincipalException extends AuthorizationException {

    private static final long serialVersionUID = 1L;

    public InvalidPrincipalException(String message) {
        super(message);
    }

    public InvalidPrincipalException(String message, Throwable cause) {
        super(message, cause);
    }

}
```

InvalidResourceException

```
/**
 * Throw when an invalid resource is accessed.
 */
public class InvalidResourceException extends AuthorizationException {

    private static final long serialVersionUID = 1L;

    public InvalidResourceException(String message) {
        super(message);
    }

    public InvalidResourceException(String message, Throwable cause) {
        super(message, cause);
    }

}
```

Update Authorizer interface to get rid of getter naming convention.

```
public interface Authorizer extends Configurable {

    /**
     * @param session   the session being authorized
     * @param operation type of operation client is trying to perform on resource
     * @param resource  resource the client is trying to access
     * @return true if the session is authorized to perform the operation on the resource, else false
     *
     * @throws org.apache.kafka.common.errors.InvalidResourceException if resource does not exist
     * @throws org.apache.kafka.common.errors.InvalidOperationException if requested operation is not
     *          supported on the resource
     */
    public boolean authorize(Session session, Operation operation, Resource resource);

    /**
     * Implementation specific description, like, supported principal types.
     *
     * @return implementation specific description
     */
    public String description();

    /**
     * Add the acls to resource, this is an additive operation so existing acls will not be overwritten,
instead these new
     * acls will be added to existing acls.
     *
     * @param acls     set of acls to add to existing acls
     * @param resource the resource to which these acls should be attached
     *
     * @throws org.apache.kafka.common.errors.AuthorizationException if not authorized to add acls for the
resource
     * @throws org.apache.kafka.common.errors.InvalidResourceException if resource does not exist
     * @throws org.apache.kafka.common.errors.InvalidAclException if an invalid acl is being added
     */
    public void addAcls(Set<Acl> acls, Resource resource);

    /**
     * Remove these acls from the resource.
     *
     * @param acls     set of acls to be removed
     * @param resource resource from which the acls should be removed
     * @return true if some acl got removed, false if no acl was removed
     *
     * @throws org.apache.kafka.common.errors.AuthorizationException if not authorized to remove acls for the
resource
     * @throws org.apache.kafka.common.errors.InvalidResourceException if resource does not exist
     * @throws org.apache.kafka.common.errors.InvalidAclException if an invalid acl is being removed
     */
    public boolean removeAcls(Set<Acl> acls, Resource resource);

    /**
     * Remove a resource along with all of its acls from acl store.
     *
     * @param resource  resource that is to be removed
     * @return true if resource existed and got deleted, else false
     *
     * @throws org.apache.kafka.common.errors.AuthorizationException if not authorized to remove acls for the
resource
     * @throws org.apache.kafka.common.errors.InvalidResourceException if resource does not exist
     */
    public boolean removeAcls(Resource resource);

    /**
     * Get set of acls for this resource.
     *
     * @param resource resource whose acls are to be returned
     * @return empty set if no acls are found, otherwise the acls for the resource
     *
     * @throws org.apache.kafka.common.errors.AuthorizationException if not authorized to access acls for the
resource
     * @throws org.apache.kafka.common.errors.InvalidResourceException if resource does not exist
     */
    public Set<Acl> acls(Resource resource);

    /**
     * Get acls for this principal.
     *
     * @param principal principal whole acls are to be returned
     * @return empty Map if no acls exist for this principal, otherwise a map of resource -> acls for the
principal
     *
     * @throws org.apache.kafka.common.errors.AuthorizationException if not authorized to access acls for the
principal
     * @throws org.apache.kafka.common.errors.InvalidPrincipalException if principal is invalid
     */
    public Map<Resource, Set<Acl>> acls(KafkaPrincipal principal);

    /**
     * Gets map of resource to acls for all resources.
     */
    public Map<Resource, Set<Acl>> acls();

    /**
     * Closes this instance.
     */
    public void close();

}
```

Proposed Changes

The KIP proposes to move authorizer interface and all related classes, i.e., `Acl, Operation, PermissionType, Resource, ResourceType, and Session`, to a separate package, `org.apache.kafka.common.security.auth.` Third-party authorizer implementations and `core` can depend on `clients` module to access `Authorizer` interface and related classes. Only change made to default authorizer, `SimpleAclAuthorizer,` will be the interface it extends.

Authorizer interface will be updated to remove getter naming convention and expected exceptions will be added.

`description()` will be added to Authorizer interface. Each authorizer implementation can override this method to provide info on implementation specific aspects of authorizer, for instance, Principal Types it supports. The description will be provided by `kafka-acls.sh` CLI when --help is specified. The KIP suggests that acls should be validated in authorizer implementations.

The changes are backwards incompatible and authorizer implementations will have to be updated to use the new interface and related classes. Apache Ranger and Apache Sentry, only known third party authorizer implementations, are OK with making this backwards incompatible change.

Compatibility, Deprecation, and Migration Plan

- *What impact (if any) will there be on existing users?*
    - Users will have to upgrade third party authorizer implementations along with broker upgrade, else Kafka broker will fail to start.
- *If we are changing behavior how will we phase out the older behavior?*
    - This will require a breaking change for third party implementations. Apache Ranger and Apache Sentry, only known third party authorizer implementations, are OK with making this backwards incompatible change.
- *If we need special migration tools, describe them here.*
    - None.
- *When will we remove the existing behavior?*
    - Existing Authorizer interface and related classes will be removed as part of the changes made as per this KIP.

Rejected Alternatives

- Add getSupportedPrincipalTypes in authorizer interface.
- Add validation at Authorizer level.
- An alternative of providing supported Principal types via interface is via a config option.
- Add description() method to Authorizer.
- Create a new module for authorizer, so that third party implementations will not have to depend on clients or core. However, to be consistent with current code org, this is rejected.