simplesteph / **kafka-security-manager**

Manage your Kafka ACL at scale https://hub.docker.com/r/simplesteph/...

#kafka #acl #ksm #zookeeper #docker #acl-changes #broker #security

78 commits | **3** branches | **5** releases | **4** contributors | MIT

Branch: master | New pull request | Find file | Clone or download

simplesteph refactored AclParser to have proper class instantiantion (#37) ... Latest commit `13fd92d` 19 days ago

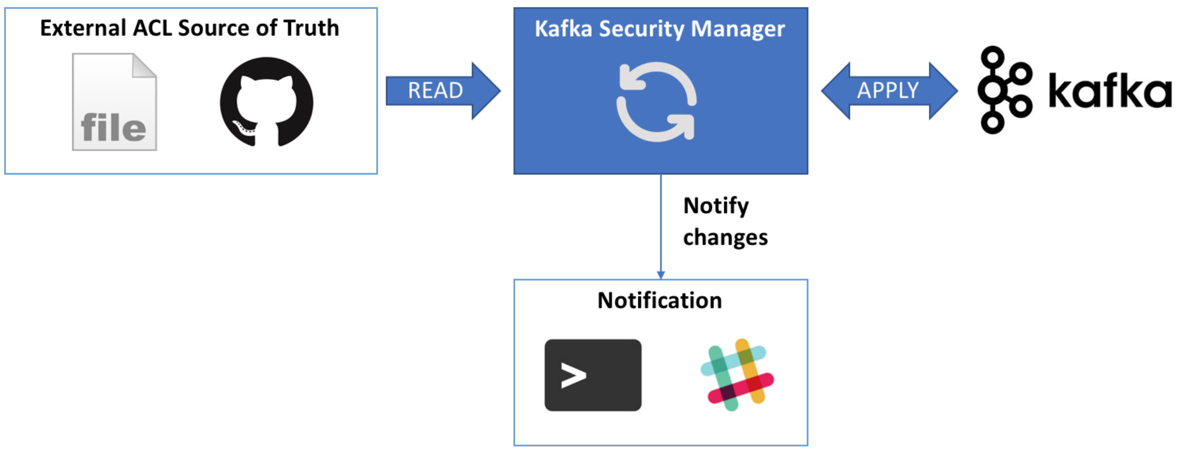
example	Kafka 2.0.0 (#30)	2 months ago
project	publish artifacts to sonatype and maven central (#25)	5 months ago
src	refactored AclParser to have proper class instantiantion (#37)	19 days ago
travis	publish artifacts to sonatype and maven central (#25)	5 months ago
.gitignore	Add support for AWS S3 as ACL Source (#27)	2 months ago
.travis.yml	sonatype.sbt NPE fixes	4 months ago
CHANGELOG.md	refactored AclParser to have proper class instantiantion (#37)	19 days ago
LICENSE.txt	added license, this fixes #7	8 months ago
README.md	refactored AclParser to have proper class instantiantion (#37)	19 days ago
build.sbt	0.5-SNAPSHOT	a month ago
docker-compose.yml	0.5-SNAPSHOT	a month ago
docker_build.sh	removed the double tests	8 months ago
docker_push.sh	fixed permissions and script for deploy	8 months ago
sonatype.sbt	sonatype.sbt NPE fixes	4 months ago
test.sh	refactoring for configuration	8 months ago

README.md

build passing

Kafka Security Manager

Kafka Security Manager (KSM) allows you to manage your Kafka ACLs at scale by leveraging an external source as the source of truth. Zookeeper just contains a copy of the ACLs instead of being the source.



There are several advantages to this:

- **Kafka administration is done outside of Kafka:** anyone with access to the external ACL source can manage Kafka Security
- **Prevents intruders:** if someone were to add ACLs to Kafka using the CLI, they would be reverted by KSM within 10 seconds.
- **Full auditability:** KSM provides the guarantee that ACLs in Kafka are those in the external source. Additionally, if for example your external source is GitHub, then PRs, PR approvals and commit history will provide Audit the full log of who did what to the ACLs and when
- **Notifications**: KSM can notify external channels (such as Slack) in order to give feedback to admins when ACLs are changed. This is particularly useful to ensure that 1) ACL changes are correctly applied 2) ACL are not changed in Kafka directly.

Your role is to ensure that Kafka Security Manager is never down, as it is now a custodian of your ACL.

A sample CSV to manage ACL is:

```
KafkaPrincipal,ResourceType,PatternType,ResourceName,Operation,PermissionType,Host
User:alice,Topic,LITERAL,foo,Read,Allow,*
User:bob,Group,bar,PREFIXED,Write,Deny,12.34.56.78
User:peter,Cluster,LITERAL,kafka-cluster,Create,Allow,*
```

Important Note: As of KSM 0.4, a new column `PatternType` has been added to match the changes that happened in Kafka 2.0. This enables KSM to manage `LITERAL` and `PREFIXED` ACLs. See #28

Building

```
sbt clean test
sbt universal:stage
```

This is a Scala app and therefore should run on the JVM like any other application

Artifacts

By using the JAR dependency, you can create your own `SourceAcl`.

SNAPSHOTS artifacts are deployed to Sonatype

RELEASES artifacts are deployed to Maven Central:

`build.sbt` (see Maven Central for the latest `version`)

```
libraryDependencies += "com.github.simplesteph" %% "kafka-security-manager" % "version"
```

Configuration

Security configuration

Make sure the app is using a property file and launch options similar to your broker so that it can

1. Authenticate to Zookeeper using secure credentials (usually done with JAAS)
2. Apply Zookeeper ACL if enabled

Kafka Security Manager does not connect to Kafka.

Sample run for a typical SASL Setup:

```
target/universal/stage/bin/kafka-security-manager -Djava.security.auth.login.config=conf/jaas.conf
```

Where `conf/jaas.conf` contains something like:

```
Client {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="/etc/kafka/secrets/zkclient1.keytab"
    principal="zkclient/example.com@EXAMPLE.COM";
};
```

Configuration file

For a list of configuration see application.conf. You can customise them using environment variables or create your own `application.conf` file and pass it at runtime doing:

```
target/universal/stage/bin/kafka-security-manager -Dconfig.file=path/to/config-file.conf
```

Overall we use the lightbend config library to configure this project.

Environment variables

The default configurations can be overwritten using the following environment variables:

- `KSM_READONLY=false` : enables KSM to synchronize from an External ACL source. The default value is `true`, which prevents KSM from altering ACLs in Zookeeper
- `KSM_EXTRACT=true` : enable extract mode (get all the ACLs from Kafka formatted as a CSV)
- `KSM_REFRESH_FREQUENCY_MS=10000` : how often to check for changes in ACLs in Kafka and in the Source. 10000 ms by default
- `AUTHORIZER_CLASS` : override the authorizer class if you're not using the `SimpleAclAuthorizer`
- `AUTHORIZER_ZOOKEEPER_CONNECT` : zookeeper connection string
- `AUTHORIZER_ZOOKEEPER_SET_ACL=true` (default `false`): set to true if you want your ACLs in Zookeeper to be secure (you probably do want them to be secure) - when in doubt set as the same as your Kafka brokers.
- `SOURCE_CLASS` : Source class. Valid values include
  - `com.github.simplesteph.ksm.source.NoSourceAcl` (default): No source for the ACLs. Only use with `KSM_READONLY=true`
  - `com.github.simplesteph.ksm.source.FileSourceAcl` : get the ACL source from a file on disk. Good for POC
  - `com.github.simplesteph.ksm.source.GitHubSourceAcl` : get the ACL from GitHub. Great to get started quickly and store the ACL securely under version control.
  - `com.github.simplesteph.ksm.source.S3SourceAcl` : get the ACL from S3. Good for when you have a S3 bucket managed by Terraform or Cloudformation. This requires `region` , `bucketname` and `objectkey` . See Access credentials for credentials management.
    - `SOURCE_S3_REGION` AWS S3 Region
    - `SOURCE_S3_BUCKETNAME` AWS S3 Bucket name
    - `SOURCE_S3_OBJECTKEY` The Object containing the ACL CSV in S3
- `NOTIFICATION_CLASS` : Class for notification in case of ACL changes in Kafka.

- `com.github.simplesteph.ksm.notification.ConsoleNotification` (default): Print changes to the console. Useful for logging
- `com.github.simplesteph.ksm.notification.SlackNotification` : Send notifications to a Slack channel (useful for devops / admin team)

- `ACL_PARSER_CSV_DELIMITER` : Change the delimiter character for the CSV Parser (useful when you have SSL)

Running on Docker

Building the image

```
./build-docker.sh
```

Docker Hub

Alternatively, you can get the automatically built Docker images on Docker Hub

Running

(read above for configuration details)

Then apply to the docker run using for example (in EXTRACT mode):

```
docker run -it -e AUTHORIZER_ZOOKEEPER_CONNECT="zookeeper-url:2181" -e EXTRACT=true \
            simplesteph/kafka-security-manager:latest
```

Any of the environment variables described above can be used by the docker run command with the `-e` options.

Example

```
docker-compose up -d
docker-compose logs kafka-security-manager
# view the logs, have fun changing example/acls.csv
docker-compose down
```

For full usage of the docker-compose file see kafka-stack-docker-compose

Extracting ACLs

You can initially extract all your existing ACL in Kafka by running the program with the config `extract=true` or `export EXTRACT=true`

Output should look like:

```
[2018-03-06 21:49:44,704] INFO Running ACL Extraction mode (ExtractAcl)
[2018-03-06 21:49:44,704] INFO Getting ACLs from Kafka (ExtractAcl)
[2018-03-06 21:49:44,704] INFO Closing Authorizer (ExtractAcl)

KafkaPrincipal,ResourceType,PatternType,ResourceName,Operation,PermissionType,Host
User:bob,Group,PREFIXED,bar,Write,Deny,12.34.56.78
User:alice,Topic,LITERAL,foo,Read,Allow,*
User:peter,Cluster,LITERAL,kafka-cluster,Create,Allow,*
```

You can then use place this CSV anywhere and use it as your source of truth.

External API

To activate this feature, set `FEATURE_GRPC=true`.

gRPC Endpoint

Kafka Security Manager exposes a GRPC endpoint to be consumed by external systems. The use case is to build a UI on top of KSM or some level of automation.

By default KSM binds to port 50051, but you can configure this using the `GRPC_PORT` environment variable.

gRPC Gateway (REST Endpoint)

By default KSM binds to port 50052, but you can configure this using the `GRPC_GATEWAY_PORT` environment variable.

This provides a REST API to consume data from KSM. Swagger definition is provided at src/main/resources/specs/KsmService.yml

Service Definition

The API is defined according to the proto file in src/main/protobuf/

Upgrade Notes

TODO: Mention to look for inter broker protocol version before doing this

Compatibility

KSM Version	Kafka Version	Notes
0.5-SNAPSHOT	2.0.0	
0.4	2.0.0	important change: added column 'PatternType' in CSV
0.3	1.1.x	
0.2	1.1.x	upgrade to 0.3 recommended
0.1	1.0.x	might work for earlier versions

Contributing

You can break the API / configs as long as we haven't reached 1.0. Each API break would introduce a new version number.

PRs are welcome, especially with the following:

- Code refactoring / cleanup / renaming
- External Sources for ACLs (JDBC, Microsoft AD, etc...)
- Notification Channels (Email, etc...)

Please open an issue before opening a PR.