KIP-252 - Extend ACLs to allow filtering based on ip ranges and subnets

Status

Current state: *"Under Discussion"*

Discussion thread: here - *pre-KIP discussion*

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

Related Jiras: ⚠ **Unable to render Jira issues macro, execution error.**

⚠ **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

The current ACL functionality in Kafka is a bit limited concerning host based rules when specifying multiple hosts. A common scenario for this would be that if have a YARN cluster running Spark jobs that access Kafka and want to create ACLs based on the ip addresses of the cluster nodes.
Currently kafka-acls only allows to specify individual ips, so creating a rule to allow access from multiple hosts would look like follows:

```
./kafka-acls --add --producer \
--topic test --authorizer-properties zookeeper.connect=localhost:2181 \
--allow-principal User:spark \
--allow-host 10.0.0.10 \
--allow-host 10.0.0.11 \
--allow-host ...
```

This can get unwieldy fast if you have a large cluster with many hundred nodes. Internally this command would not create a single ACL with multiple host entries, but rather one ACL per host that was specified on the command line, which can make the ACL listing very confusing when multiple principals and multiple topics need to be specified for all hosts.

Implementing functionality that allows to specify entire subnets, or custom ip address ranges would help to keep the amount of rules small, which has multiple benefits:

- less data needs to be stored in Zookeeper
- less rules to be kept in memory and evaluated by broker
- potentially slightly better performance due to range comparison instead of string comparison per host
- much better visibility of current rules for admins

Public Interfaces

The changes proposed here do not affect public interfaces or otherwise interfere with backward compatibility. All changes outlined below have been designed in a fashion that works with ACLs that were previously created and will not change the behavior of the kafka-acl.sh script in any way.

The only change that will be outwardly visible is the number of rules that are generated internally when multiple hosts are specified on the command line. In case anybody automatically generates ACLs from another system and has processes in place that check whether all expected ACLs were generated this should not unduly affect anybody.

Proposed Changes

I propose to extend the current SimpleACLAuthorizer as well as the internal ACL classes to allow specifying IP addresses in the following formats:

- **single IP address:** same as in the current version
- **IP address range:** start and end address separated by a minus sign (for example: 10.0.0.1-10.0.0.10 or 192.168.0.1-192.168.10.255)
- **Subnet in CIDR notation**: see wikipedia (for example: 10.0.0.0/24)

The same options will be accepted for IPv6 deployments:

- **single IP address**: same as IPv4, a single address (for example: fd7d:4b9e:a5ce:ffff::100)
- IP address range: start and end address separated by a minus sign (for example: fd7d:4b9e:a5ce:ffff::100-fd7d:4b9e:a5ce:ffff::200)
- **Network Range**: An IPv6 range specification - see wikipedia (for example: fd7d:4b9e:a5ce::/48)

The kafka-acl tool will accept a comma separated list containing any combination of the above, which allows for flexible specification of multiple ranges, if necessary. So for example the following would be possible: *10.0.0.1,192.168.0.0/24,10.10.0.1-10.10.0.150*

Mixing IPv4 and IPv6 addresses will also be acceptable.

To ensure backwards compatibility, it will remain possible to specify multiple --allow-host or --deny-host options on the command line, internally all of these would be concatenated together with a comma as separating character and then be treated just like a single parameter. So the following two calls are functionally identical:

```
./kafka-acls --add --producer \
--topic test --authorizer-properties zookeeper.connect=localhost:2181 \
--allow-principal User:spark \
--allow-host 10.0.0.10-10.0.0.100 \
--allow-host 10.0.0.0/24 \
--allow-host 192.168.0.1


./kafka-acls --add --producer \
--topic test --authorizer-properties zookeeper.connect=localhost:2181 \
--allow-principal User:spark \
--allow-host 10.0.0.10-10.0.0.100,10.0.0.0/24,192.168.0.1
```

⚠ Unable to render Jira issues macro, execution error.

has a patch available, but would currently only add interpretation of CIDR notation, no specific ranges, which I think could easily be added.

I have looked at the ACL and Authorizer code a bit and created a very brief proof of concept implementation, just to convince myself that this can be done in a fashion that doesn't change/break too much. I've pushed the commit in case anybody is interested to take a look - this is nowhere near finished or properly tested, but gives an idea of how this could be done.

Compatibility, Deprecation, and Migration Plan

Backwards compatibility to ACLs that were generated with older versions is given, as the data format in which ACLs are stored in Zookeeper is not changed and can still be loaded with later implementations of the code.

Source compatibility is also given, due to the fact that parsing of the strings would need to happen internally in the ACL class and the current constructor would not be changed, thus all code creating instances of the class still works.

Changes to the SimpleACLAuthorizer are necessary, in order to implement checking of the ip ranges, instead of performing a string comparison.

The only area where current code might be affected are custom authorizers that use ACLs from Zookeeper and check ip addresses. These would need to be adapted to call a new method on the acl object instead of directly comparing the address, which is a fairly minor change.

Rejected Alternatives

KIP-7 discussed something similar to this, but was rejected. My guess is, that this was swept up in the larger security initiative and the simply got left out in the end.