Configurable Token Management

for Spark Running on YARN

Background

Current Spark on YARN supports accessing kerberized cluster if security is enable, also it is transparent for end user, Spark on YARN automatically gets the token from system to be accessed. This significantly enhance the usability for production users, but still it has several problems:

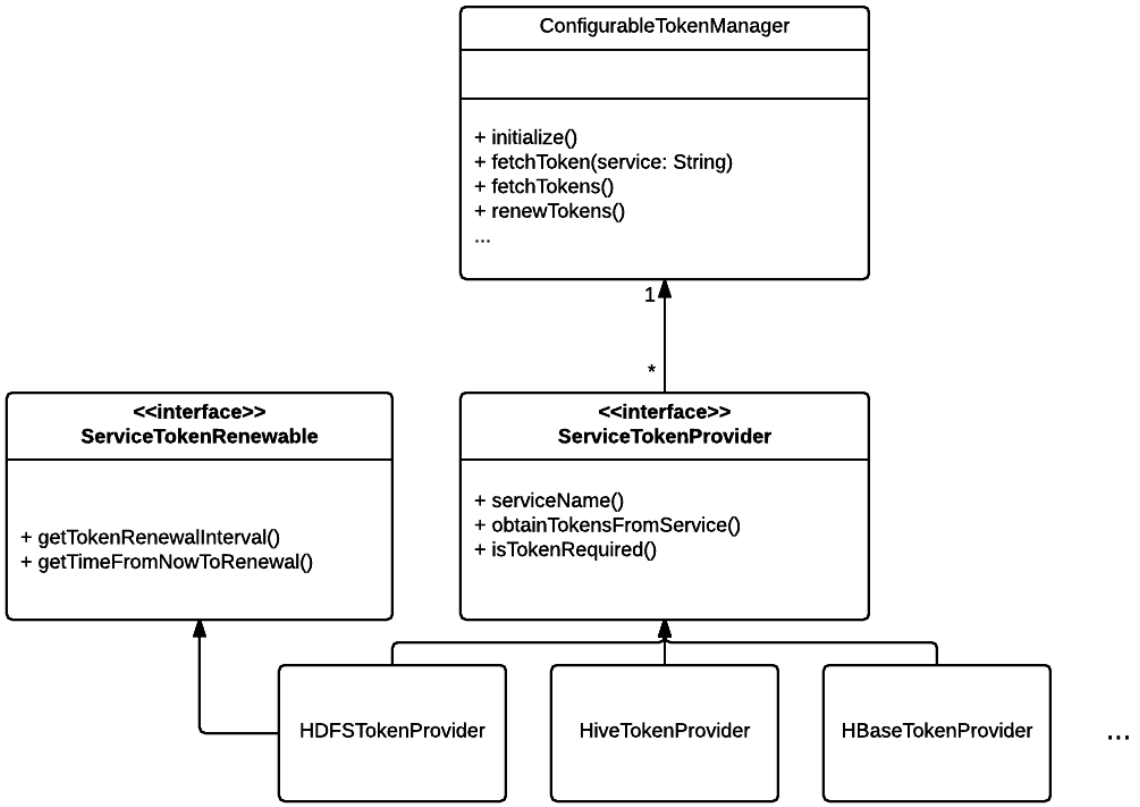
1. Supported service is hard-coded, only HDFS, Hive and HBase are supported for token fetching. For other third-party services which need to be communicated with Spark in Kerberos way, currently the only way is to modify Spark code.
2. Current token renewal and update mechanism is also hard-coded, which means other third-party services cannot be benefited from this system and will be failed when token is expired.
3. Also In the code level, current token obtain and update codes are placed in several different places without elegant structured, which makes it hard to maintain and extend.

Target

So here propose a new configurable token management class to address all the issues mentioned above:

- Make service pluggable and configurable, this will allow users to add their own service token fetching mechanism (not only limit to HDFS, Hive and HBase).
- Add configurable token management class to manage all the services, including token fetching, renewing and updating.
- Provide enough flexibility for end users to satisfy different scenarios (batch processing or long running service, dynamically token updating...)

Solution



ServiceTokenProvider

ServiceTokenProvider is responsible for obtaining token from specific service, any service that need to be accessed through Kerberos has to implement its own *ServiceTokenProvider* to offer a way to get the tokens. Also *getTokenRenewalInterval* should be implemented to offer the token renewal interval for timely token renewal.

By default *HDFSTokenProvider*, *HiveTokenProvider* and *HBaseTokenProvider* will be implemented internally. Also users could implement their own *ServiceTokenProvider* and registered into *ConfigurableTokenManager*.
Two configurations to control whether this *ServiceTokenProvider* will be loaded or not:

1. spark.yarn.security.tokens.${service}.enabled
   This configuration controls whether this service will be loaded or not, this configuration keeps the same as current one, by default hdfs, hbase and hive token provider will be loaded. For any other service which needs to be loaded should set this configuration to true.

Also the **service** here should keep the same name as provided in *ServiceTokenProvider*.

2. spark.yarn.security.tokens.${service}.class

   This configuration specifies the full-qualified class name of token provider. By default hdfs, hbase and hive class will be loaded in automatically. For other service needs to be loaded should specify the class name.

ServiceTokenRenewable

For any service token Provider in which token is renewable should implement this interface, this will be used for periodical token renewal to avoid token expiration.

ConfigurableTokenManager

ConfigurableTokenManager manages all the registered token providers and provide APIs for other modules to call. This module will be lied in yarn/client, driver, AM and executors as a singleton. In the yarn/client it will fetch the tokens and add into Credentials and UGI, in the AM side it will periodically renew the tokens, in the driver/executor side it will update the new tokens periodically.