MEP 27 -- Support Basic Authentication

Current state: Under Discussion

Summary

To support authentication by username/password when accessing milvus instance.

Motivation

There is no basic security model for milvus instances currently. Users can access any milvus instance once they have the address by any milvus sdk. This project aims to support basic authentication with username/password. Clients need to provide username and password when accessing the milvus instance.

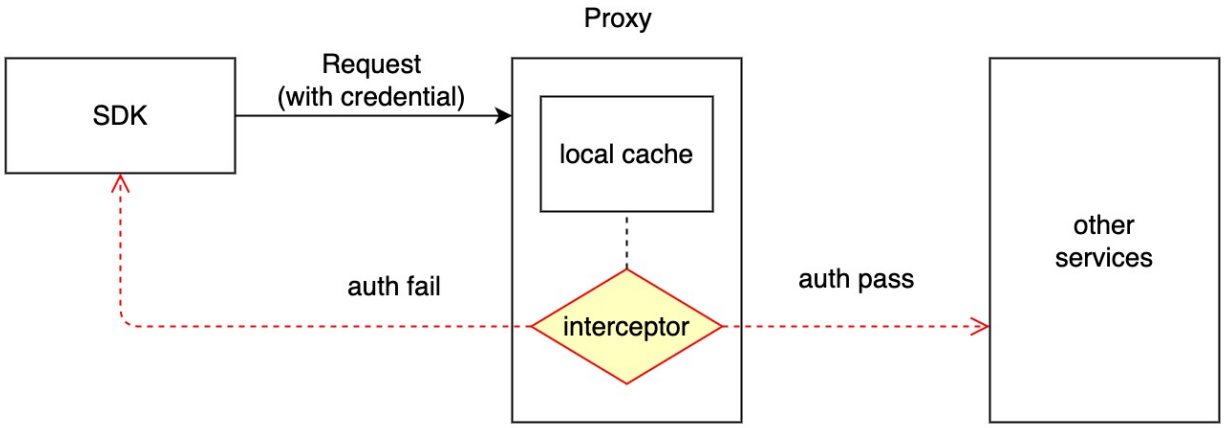
Design Details

Prerequisite

- SDK clients MUST encrypt password when connecting to milvus service.
- Milvus create default user root as an initial user to create other users.

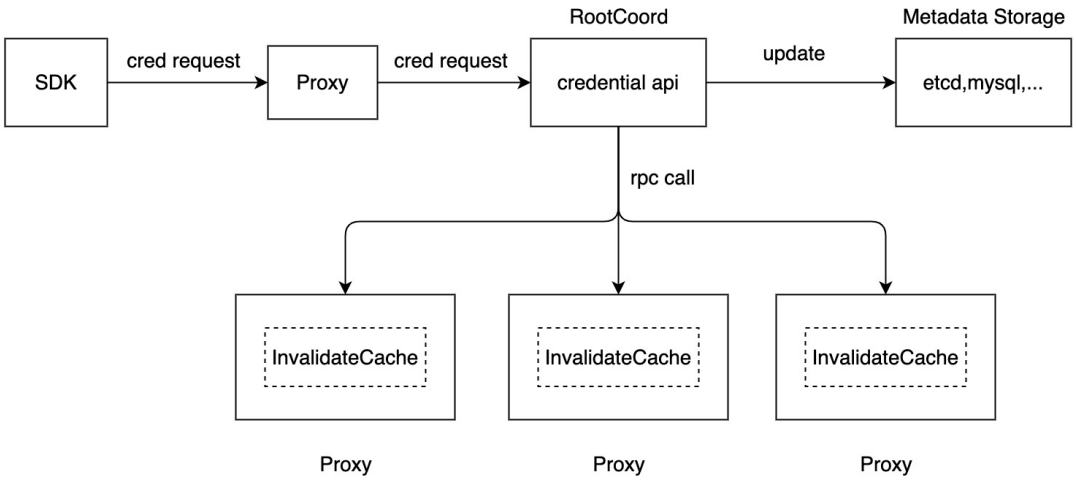
Authentication Workflow
Since grpc requests all handled by proxy, we will do the authentication in the proxy component. Logging on to the milvus instance will follow the processes below

1. Create credential for each milvus instance and store encrypted password in etcd. Here we use bcrypt which implements Provos and Mazières's ad aptive hashing algorithm.
2. On the client side, SDK sends ciphertext when connecting milvus service. The ciphertext is base64(<username>:<passwd>) and attached to the metadata with the key "authorization".
3. Milvus proxy component intercepts the request and verify the credential.
4. Credentials are cached locally on Proxy component.



Cache Update Workflow

1. Credential apis (insert/query/delete credentials) are implemented by RootCoord.
2. Credential modification apis persist credentials on etcd (or other storage like mysql), and call each proxy's api to invalidate local caches in all proxy components.
3. Auth interceptor in proxy component will firstly find credential records from local cache. If the cache missed, it will trigger rpc call to fetch record from RootCoord and update the local cache.



Etcd model for credentials

```
Key: ${prefix}/credentials/users/${username}
Value: {"password": ${encrypted_password}, ...}
```

Interface of credentials apis

```
struct Credential {
        username string,
        password string
}

func NewCredential(cred Credential) (bool,error)
func ListUsers() []string
func UpdateCredential(cred Credential) (bool,error)
func DeleteCredential(username string) (bool,error)
```

Compatibility

To be compatible with preview version, Milvus will use a toggle for it. If the toggle is on, it will check the credentials for each grpc call, otherwise it acts like the non-authenticate mode.

Test Plan

Case 1: create credentials for milvus

1. Access with correct credentials should succeed
2. Access with incorrect credentials should fail
3. Access without credentials should fail

Case 2: no credentials created for milvus

1. Access without credentials should succeed
2. Access with credentials should succeed (just ignore the input credential)

Future work

SSL/TLS transportation
Authorization on RBAC control