AQUAQ
Analytics

AquaQuarantine:

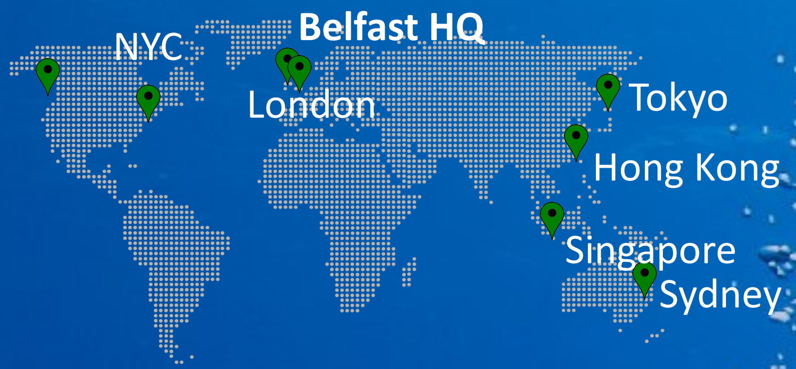
Securing kdb+

Experts in fast data solutions

for demanding environments

AQUAQ Analytics

- Established in 2011
- Headquarters in Belfast, N.Ireland
- Headcount of 160 staff
- 2016 US Subsidiary launched
- 2018 Singapore subsidiary launch
- 2020 Hong Kong subsidiary launch



What do we do?

Technology Consultancy Services

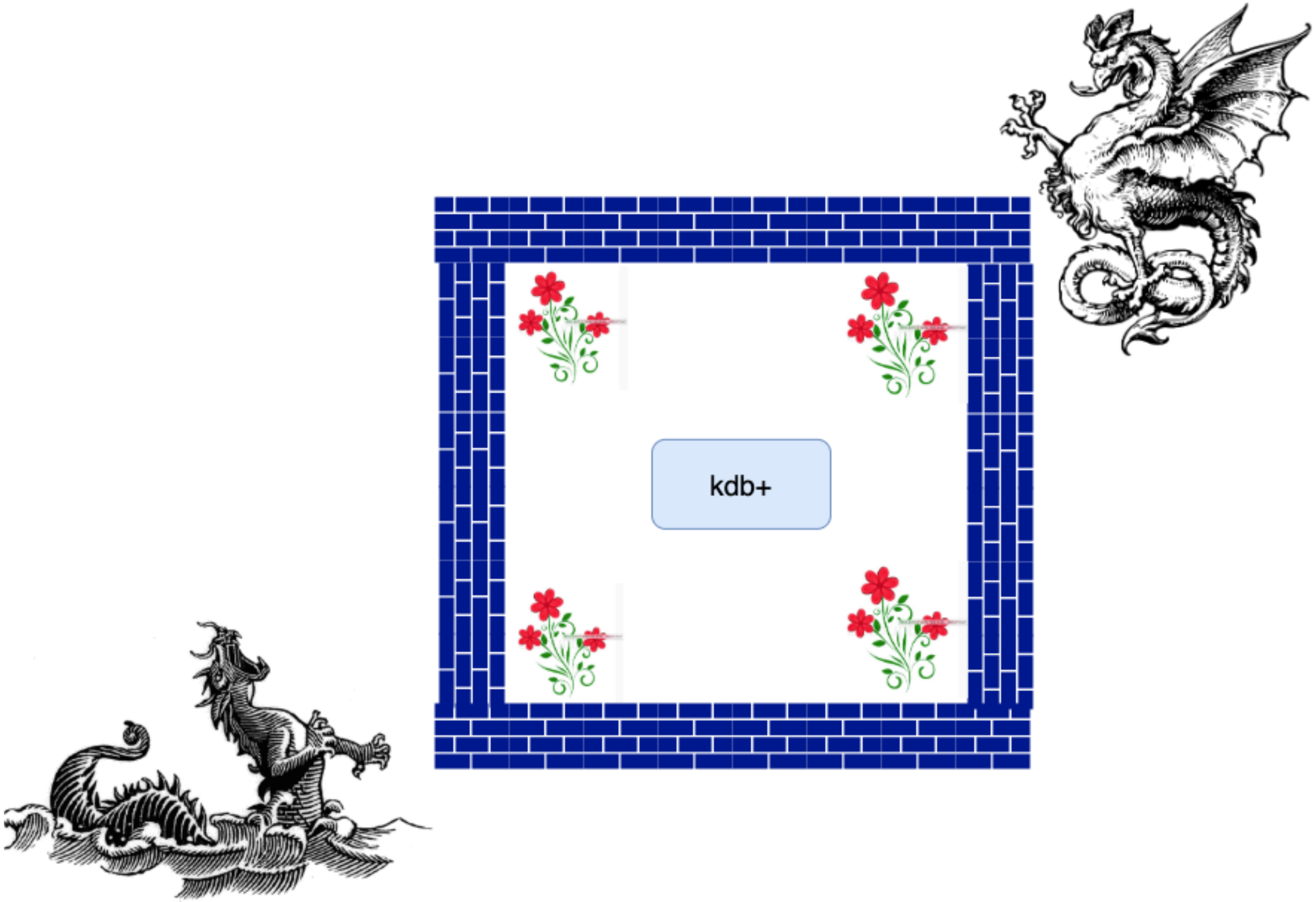
Altair Panopticon Professional Services

Remote (24/7) Support Centre of Excellence

- Authentication
- Encryption
- Entitlements

- kdb+ has added security features over the last few years
- Security scrutiny for internal applications has increased
- Authentication and encryption now critical requirements, especially for cloud

- I’m not a security expert!


kdb+

```
q –p 5000
```

```
http://localhost:5000/?system%22rm%20-r%20*%22
```

Authentication

```
-u  / username and password restrictions
    / file system access restrictions

-U  / as u, but no access restrictions
```

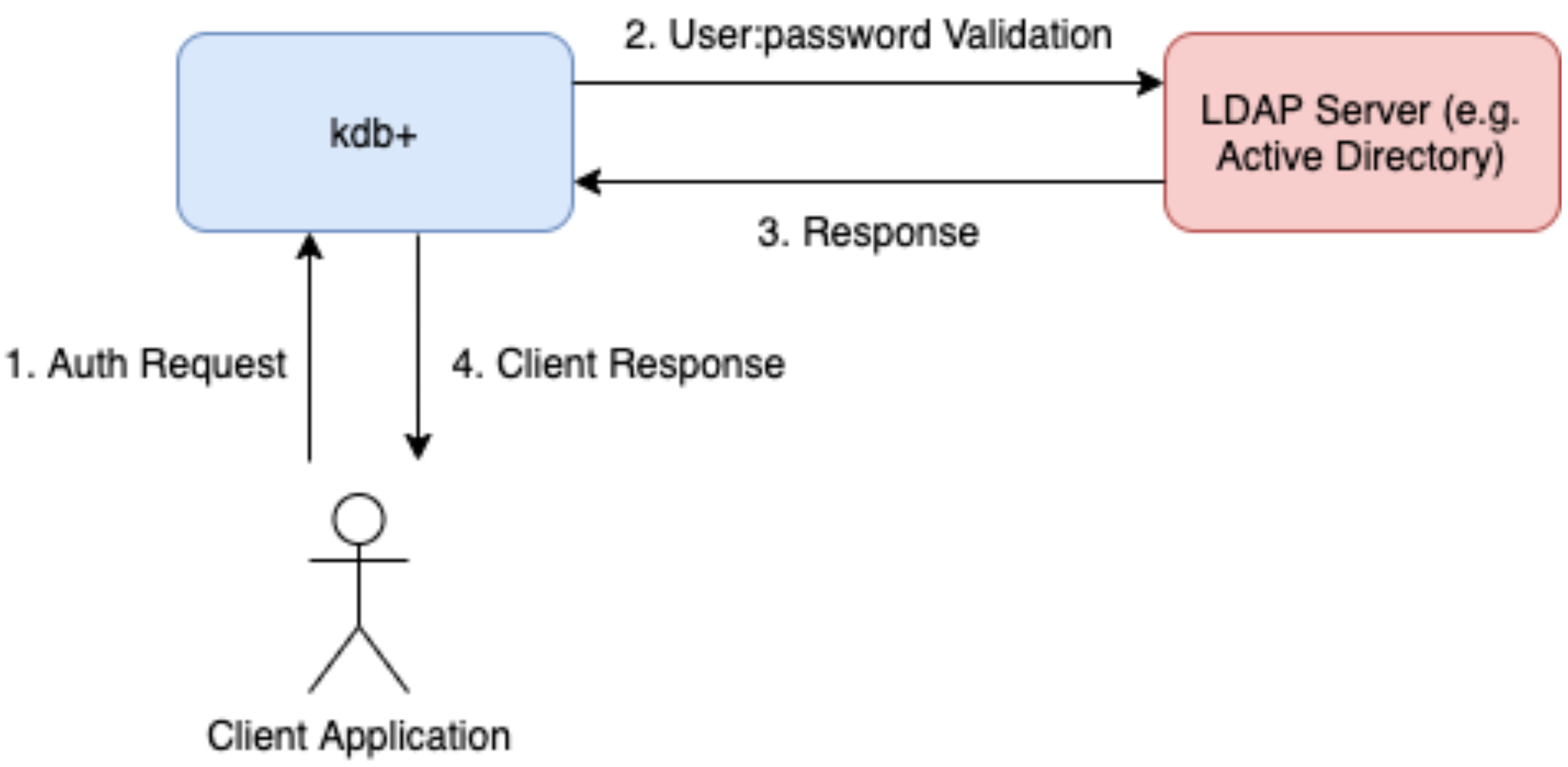
Password file format is

```
user1:pass1
user2:pass2
```

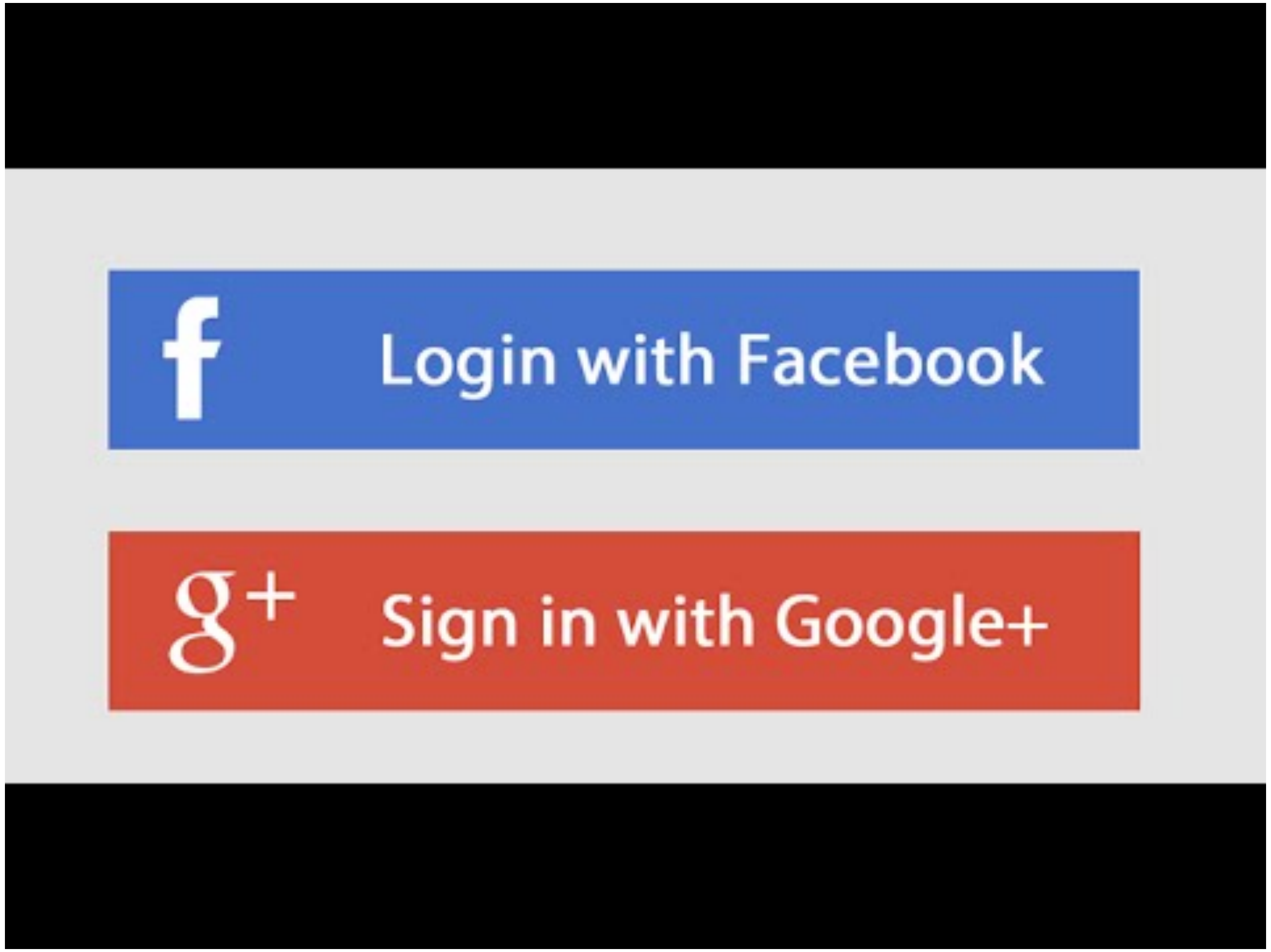
Passwords in password file can be hashed

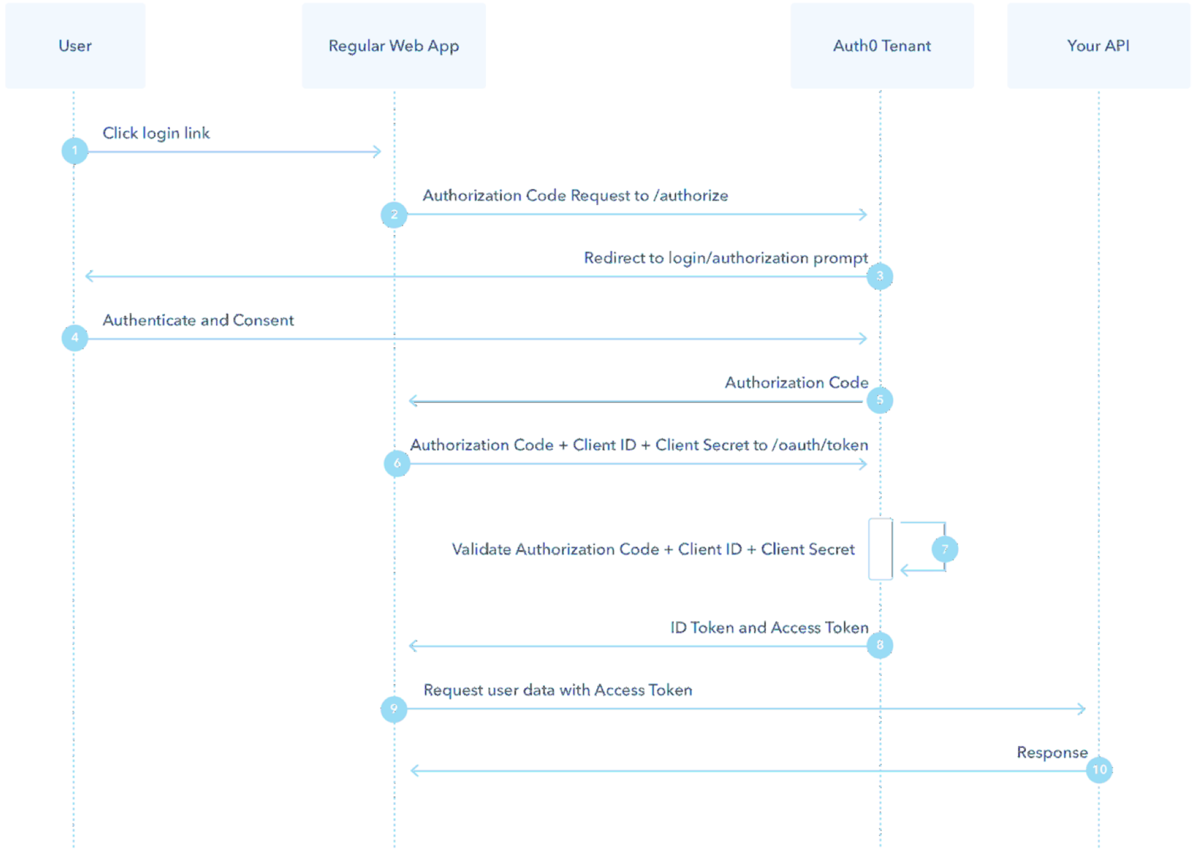
```
md5   / md5
-33! / sha1 (kdb+ 4.0)
```

```
.z.pw:{[u;p]
 if[u in `jim`bob`anne; :1b];
 :doComplexAuth[u;p]}
```



- Users+passwords are centrally managed
- Groups can be used to manage access


Login with Facebook
Sign in with Google+



User

Regular Web App

Auth0 Tenant

Your API

1 Click login link

2 Authorization Code Request to /authorize

3 Redirect to login/authorization prompt

4 Authenticate and Consent

5 Authorization Code

6 Authorization Code + Client ID + Client Secret to /oauth/token

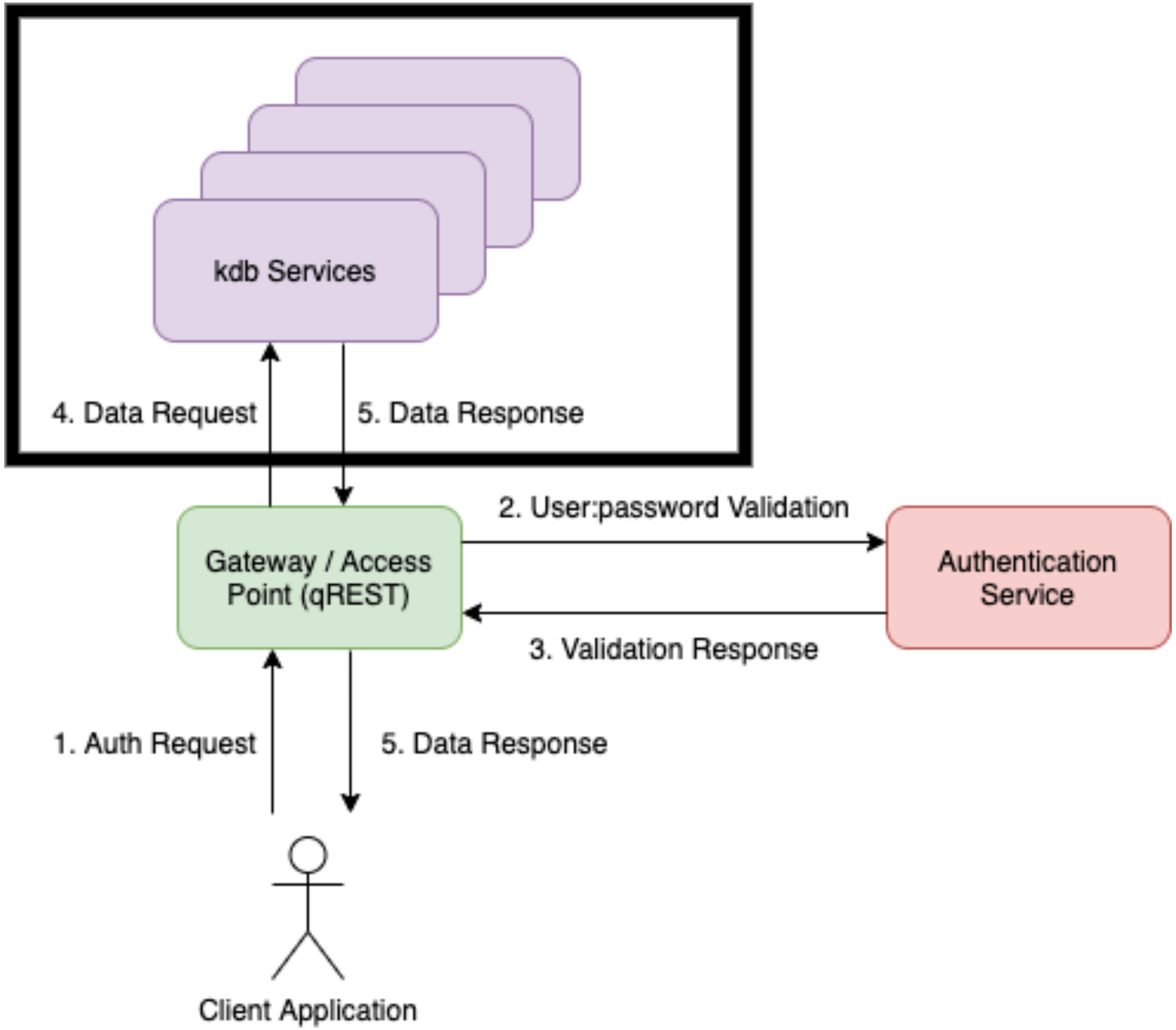
7 Validate Authorization Code + Client ID + Client Secret

8 ID Token and Access Token

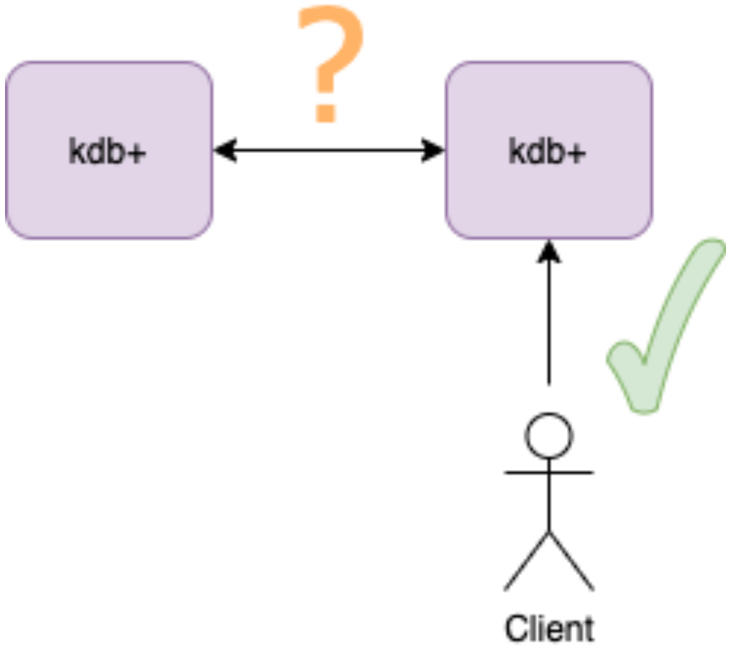
9 Request user data with Access Token

10 Response

- Single Sign On, Token based e.g. oauth2, SAML
- kdb+ never sees password (only token)
- Easier to extend e.g. multifactor authentication


kdb Services
4. Data Request
5. Data Response
2. User:password Validation
Gateway / Access Point (qREST)
Authentication Service
3. Validation Response
1. Auth Request
5. Data Response
Client Application

- q process <-> q process – how / where to store the secrets for outbound connection?
- Shouldn’t store secrets in version control
- Can inject secrets to codebase on build step and compile
- Can store secrets externally from codebase and decrypt
- IP whitelists (.z.a)
- Must restrict to API access only



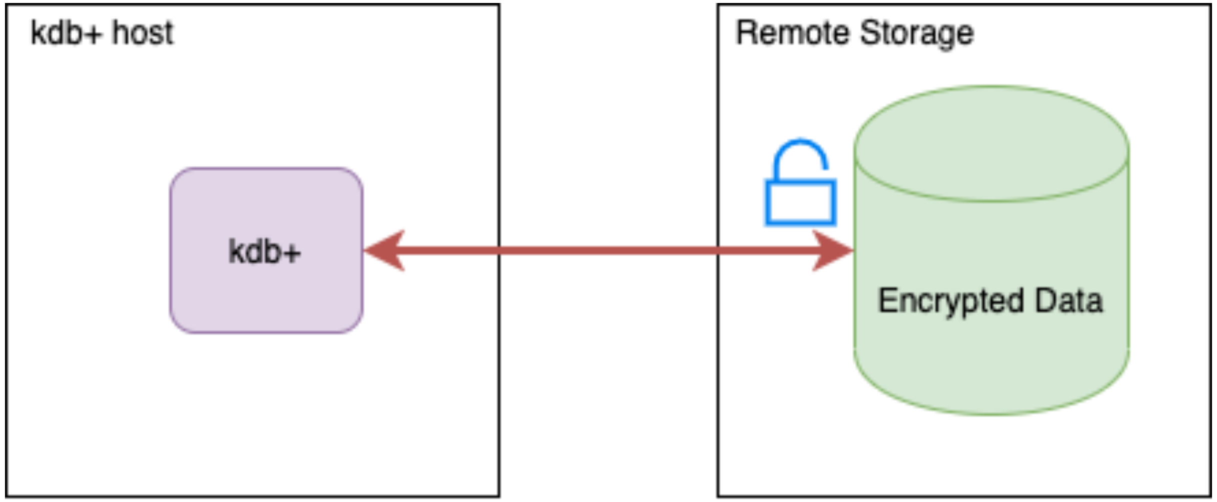
Encryption

- Since 3.4, kdb+ has supported TLS for in-transit encryption
- Some data may not be considered sensitive (market data)
- Execution data is sensitive
- Any data transiting externally must be encrypted
- We see more and more requirements for securing internal connections

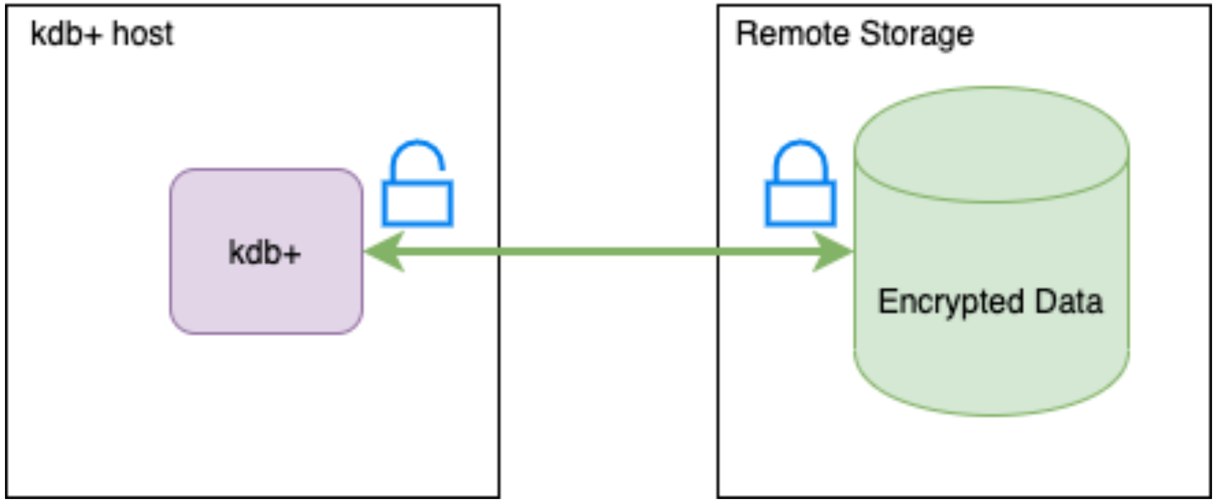
- TLS setup requires certificates on both client and server
- kdb+ can be set to accept both plain and encrypted connections
- Encryption has an overhead
  - 40-50x slower on hopen
  - 1.5x slower on data transfer
- Do all connections need to be encrypted?
- Does all data?
- Could the architecture be modified to reduce the number of encrypted connections?
- Localhost connections don’t need encryption

- Data is considered to “rest” both in memory and on-disk
- File System encryption can be employed (Full Disk Encryption)
- Transparent Disk Encryption (TDE) is available in kdb+ 4.0
- With TDE, kdb+ does the decryption
  - Selective encryption
  - If storage device is remote data is transferred encrypted
  - Data is portable without decryption/encryption cycle
  - Platform agnostic
  - Separation of responsibilities

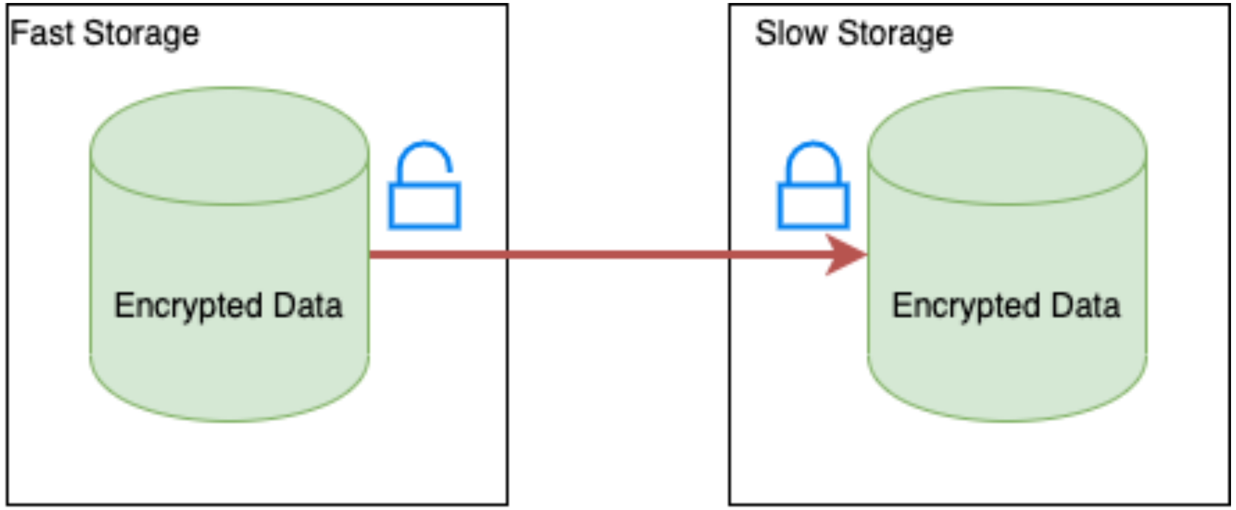
FDE


kdb+ host
kdb+
Remote Storage
Encrypted Data

TDE


kdb+ host
kdb+
Remote Storage
Encrypted Data

FDE



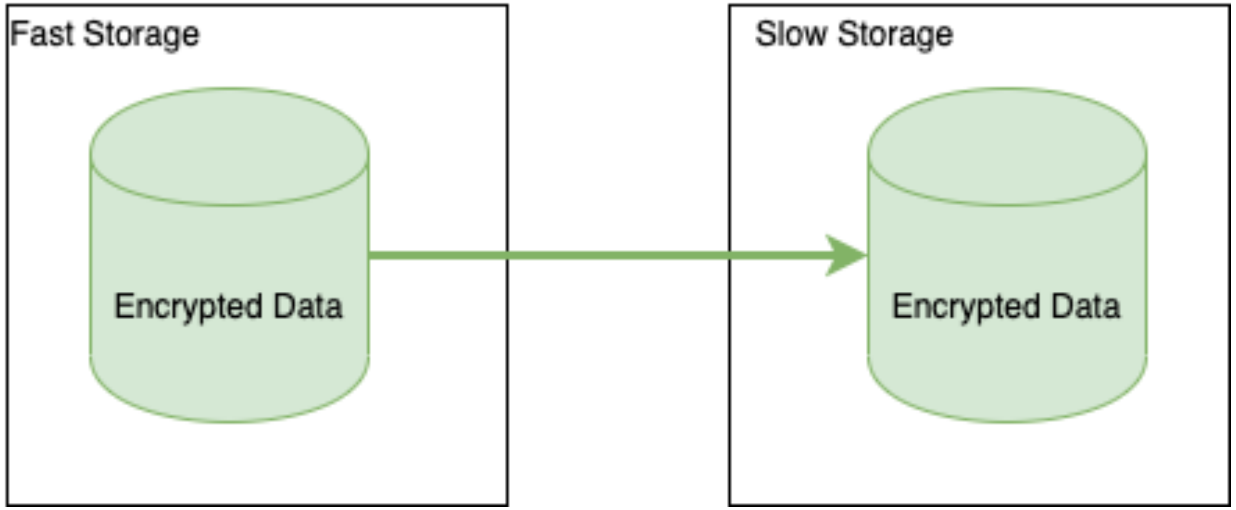
Fast Storage

Encrypted Data

Slow Storage

Encrypted Data

TDE



Fast Storage

Encrypted Data

Slow Storage

Encrypted Data

- Data-at-rest encryption has a performance overhead
- Less overhead than compression, minimal additional overhead when added on top of compression
- Requires more modern chipsets for performance (AES-NI support)

- Data-at-rest in-memory encryption not possible currently

Entitlements

- All incoming requests can be interrogated in .z.[ps|pg|ph|pi|ws]
- IP address, user, handle available (.z.a, .z.u, .z.w)
- The best and easiest way to control a system is to restrict to pre-defined function calls

```
.z.pg:{
 if[not (type[x] in 0 11h) and type[first x] in -11 10h;
  '`$"not a pre-defined function call"];
 // do stuff here
 }
```

```
-b / blocked (read only access)

.z.pg:{
 $[.z.u in superusers;
   value x;
   reval(value;enlist x)]}
```

```
.z.ph:{.h.he["no chance"]}
```

https://code.kx.com/q/kb/dare/

https://code.kx.com/q/kb/ssl

https://code.kx.com/q/kb/firewalling/

https://code.kx.com/q/ref/eval/

https://code.kx.com/q/basics/cmdline/

http://aquaqanalytics.github.io/TorQ/handlers/#permissionsq

Thanks!

Q+A

30th April: TorQ

7th May: Gateway Design Principles

14th May: Grafana and kdb+

21st May: kdb+ 4.0

28th May: TBC