SphereEx | linking data and services simply

Dynamic data masking & encryption for MySQL/PostgreSQL with no code changes

Trista Pan, SphereEx CTO & Co-founder

panjuan@apache.org

Content

- Data life cycle management
- Technologies help with data security
- How to perform these technologies
- Apache ShardingSphere
- Solution introduction
- A hands-on practice

About me

SphereEx Co-Founder & CTO

Apache Member & Incubator mentor

AWS Data Hero

Tencent Cloud TVP

Apache ShardingSphere PMC

Apache brpc & Apache AGE & Apache HugeGraph (Incubating) mentor

China Mulan Community Mentor

Bio: https://tristazero.github.io

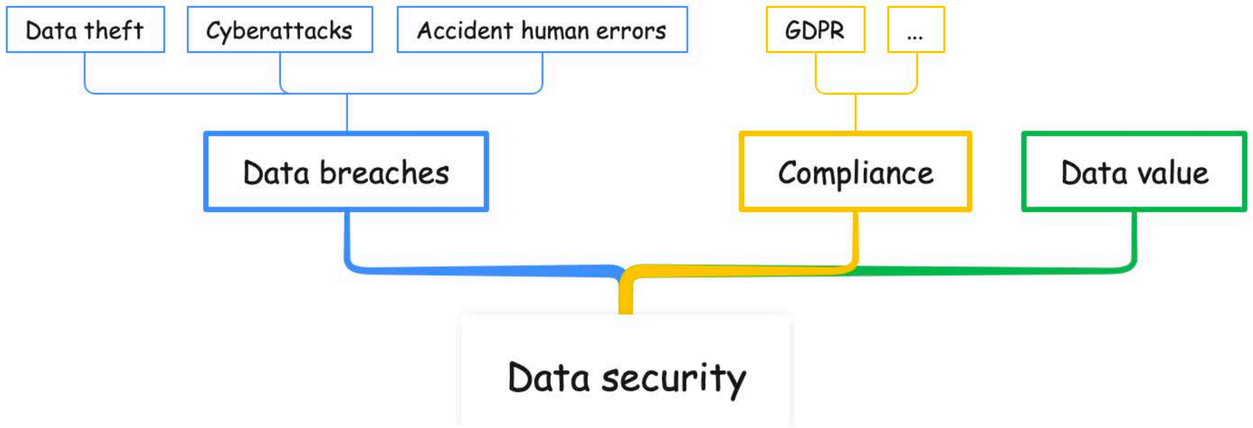
LinkedIn: https://www.linkedin.com/in/panjuan

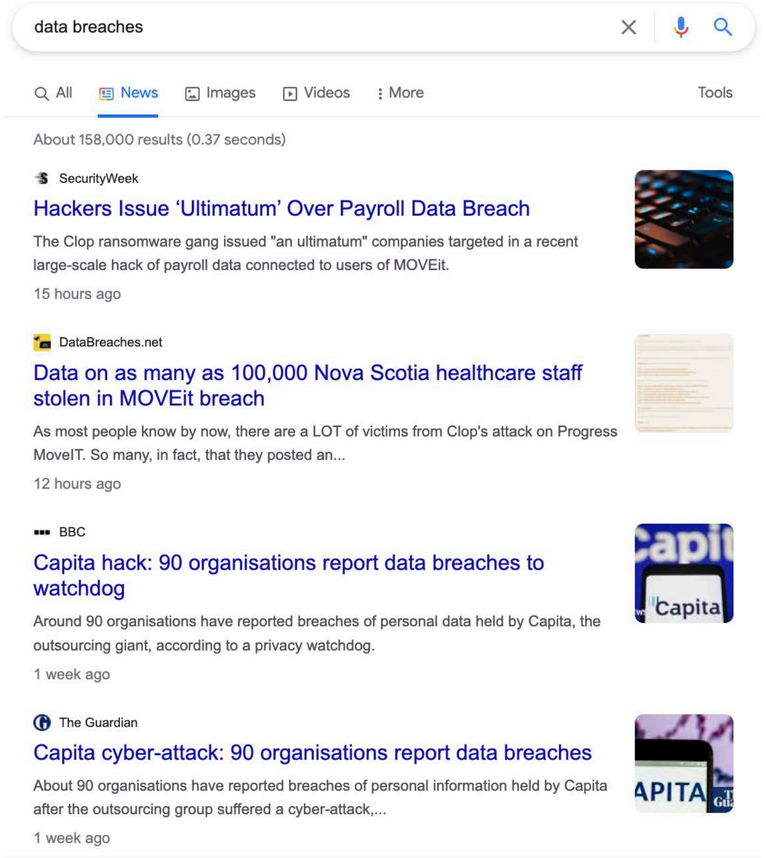
GitHub: https://github.com/tristaZero

Twitter: @tristaZero

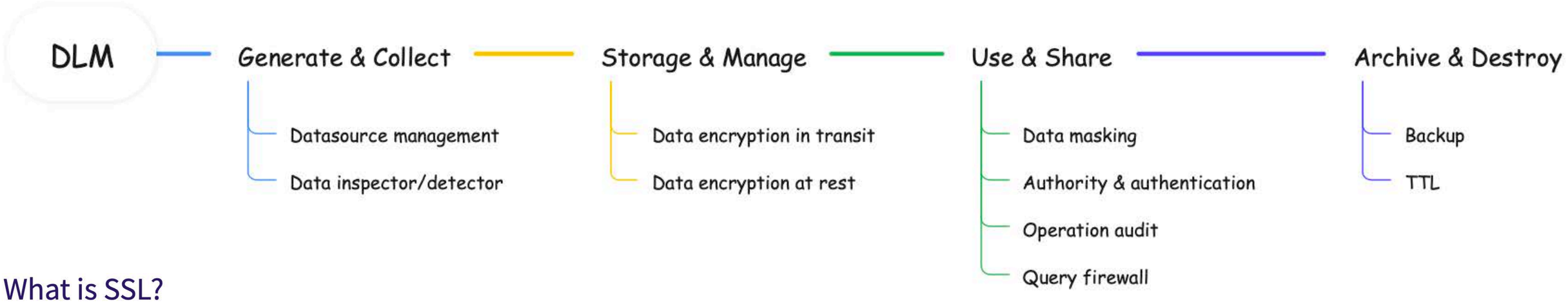
Project Twitter: @ShardingSphere

Data security





Data life cycle management



What is SSL?

SSL stands for Secure Sockets Layer. It is a protocol that Netscape developed in the 1990s as a way to secure communications over the internet. Today, its primary function is to prevent security flaws in communications by encrypting data sent between two parties. SSL is used in various applications, including email, web browsing, and file transfer.

SSL Protocol

Netscape developed the SSL protocol in the 1990s. It is a proprietary protocol that is not subject to public scrutiny. TLS has superseded SSL certificates, but SSL is still used in some applications.

What is TLS?

Transport Layer Security, or TLS, provides the same security features as SSL but with some enhancements. The Internet Engineering Task Force (IETF) created TLS to standardize security protocols across the internet.

Using Time to Live

Time to Live (TTL) is a mechanism that allows you to automatically expire table rows. TTL is expressed as the amount of time data is allowed to live in the store. Data which has reached its expiration timeout value can no longer be retrieved, and will not appear in any store statistics. Whether the data is physically removed from the store is determined by an internal mechanism that is not user-controllable.

Data life cycle management

- DLM
  - Generate & Collect
    - Datasource management
    - Data inspector/detector
  - Storage & Manage
    - Data encryption in transit
    - Data encryption at rest
  - Use & Share
    - Data masking
    - Authority & authentication
    - Operation audit
    - Query firewall
  - Archive & Destroy
    - Backup
    - TTL

What is Data Encryption?

Data encryption is a method of protecting data by encoding it in such a way that it can only be decrypted or accessed by an individual who holds the correct encryption key. When a person or entity accesses encrypted data without permission, it appears scrambled or unreadable.

Data encryption is the process of converting data from a readable format to a scrambled piece of information. This is done to prevent prying eyes from reading confidential data in transit. Encryption can be applied to documents, files, messages, or any other form of communication over a network.

user_address_cipher	user_address_plain
cebM4llQQIvKk3wbjCG8NQ==	123 Main St, NY
KUSrSjZcwGKET+ZDvkBnOJiOEClgm1K0kDlafjO/+uk=	456 Smith Ave, CA
tRbooaDDWghOI3B6PUWi7w==	789 Park Rd, TX
bBGl4bOAVevf/FyIdeRPMg==	1010 Elm St, IL
rfSp9MxNlBozIlMTBta+eAq2ZLhWcqfQ8/EQnIqMx+g=	555 Broadway, NY
NK55Nym5MNLmn0eHjKkNbA==	777 Oak Ln, CA
MOn3XXmSC6Zcx1/Y/EYVZAq2ZLhWcqfQ8/EQnIqMx+g=	999 Maple Rd, IL
TNfMDNA/47h5aXjn9GzSPw==	333 Pine Dr, FL
ND3LcNpVd+QYdqpww8gi4Aq2ZLhWcqfQ8/EQnIqMx+g=	444 Cedar St, TX
HSuHGZpidkNHrnUNXWH6WxGPK+n6ssFMRDGfbua8gng=	888 Beach Blvd, FL

What is an Encryption Algorithm?

Encryption algorithms are used to convert data into ciphertext. By using the encryption key, an algorithm can alter data in a predictable manner, resulting in the encrypted data appearing random, but it can be converted back into plaintext by using the decryption key.

Best Encryption Algorithms

There's a host of different encryption algorithms available today. Here are five of the more common ones.

- AES. The Advanced Encryption Standard (AES) is the trusted standard algorithm used by the United States government, as well as other organizations. Although extremely efficient in the 128-bit form, AES also uses 192- and 256-bit keys for very demanding encryption purposes. AES is widely considered invulnerable to all attacks except for brute force. Regardless, many internet security experts believe AES will eventually be regarded as the go-to standard for encrypting data in the private sector.
- Triple DES. Triple DES is the successor to the original Data Encryption Standard (DES) algorithm, created in response to hackers who figured out how to breach DES. It's symmetric encryption that was once the most widely used symmetric algorithm in the industry, though it's being gradually phased out. TripleDES applies the DES algorithm three times to every data block and is commonly used to encrypt UNIX passwords and ATM PINs.
- RSA. RSA is a public-key encryption asymmetric algorithm and the standard for encrypting information transmitted via the internet. RSA encryption is robust and reliable because it creates a massive bunch of gibberish that frustrates would-be hackers, causing them to expend a lot of time and energy to crack into systems.

What is Data Masking?

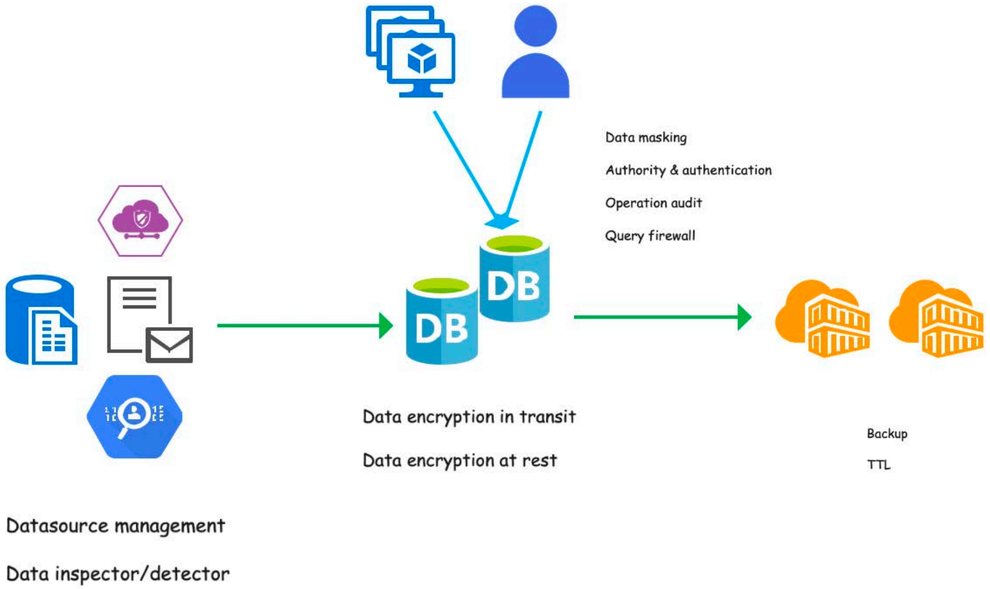
Data masking is a data security technique that scrambles data to create an inauthentic copy for various non-production purposes. Data masking retains the characteristics and integrity of the original production data and helps organizations minimize data security issues while utilizing data in a non-production environment. This masked data can be used for analytics, training, or testing.

last_name	first_name	ssn	gender	state
Smith	Bob	123-45-6789	M	CA
Doe	Jane	098-76-5432	F	PA
King	Stephen	888-67-5309	M	WI
Savage	Randal;	135-24-6789	M	FL
Downer	Debbie	918-55-4680	F	NC

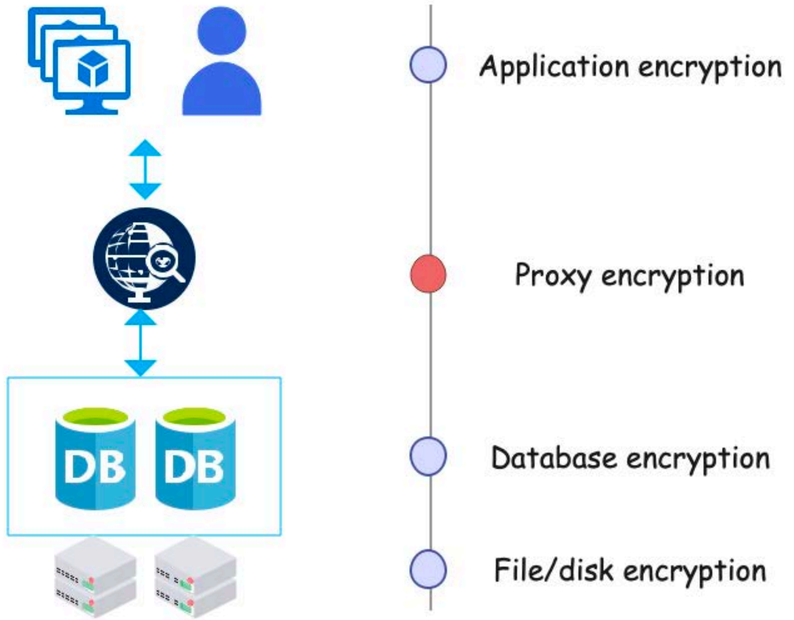
→

last_name	first_name	ssn	gender	state
Smith	Bob	xxx-xx-xxxx	M	CA
Doe	Jane	xxx-xx-xxxx	F	PA
King	Stephen	xxx-xx-xxxx	M	WI
Savage	Randy	xxx-xx-xxxx	M	FL
Downer	Debbie	xxx-xx-xxxx	F	NC

Data life cycle management



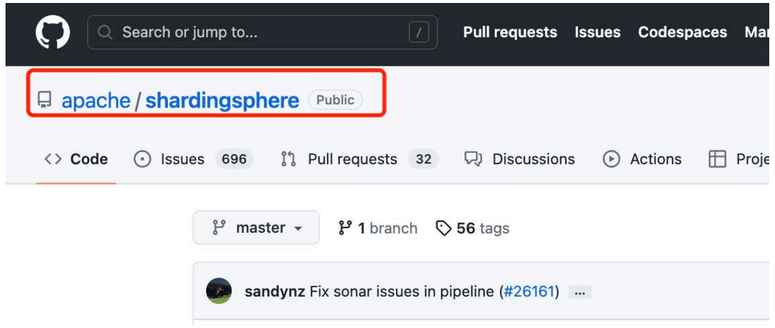
- ✓ Data encryption in transit
- ✓ Data encryption at test
- ✓ Dynamic data masking
- ✓ Authentication

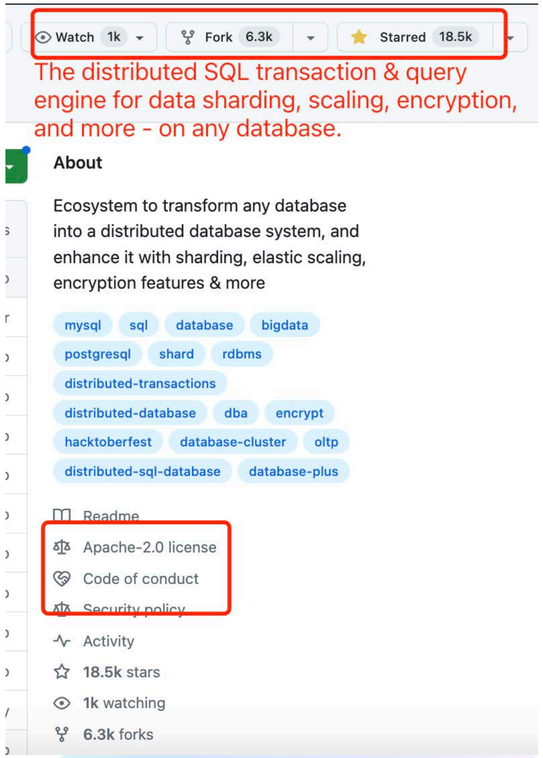


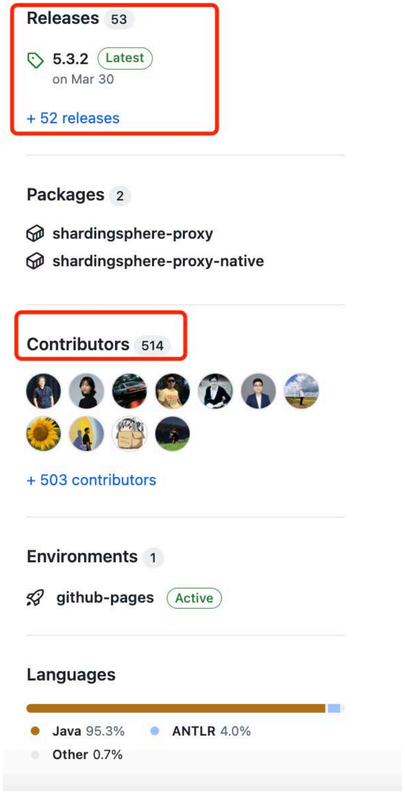
Apache ShardingSphere

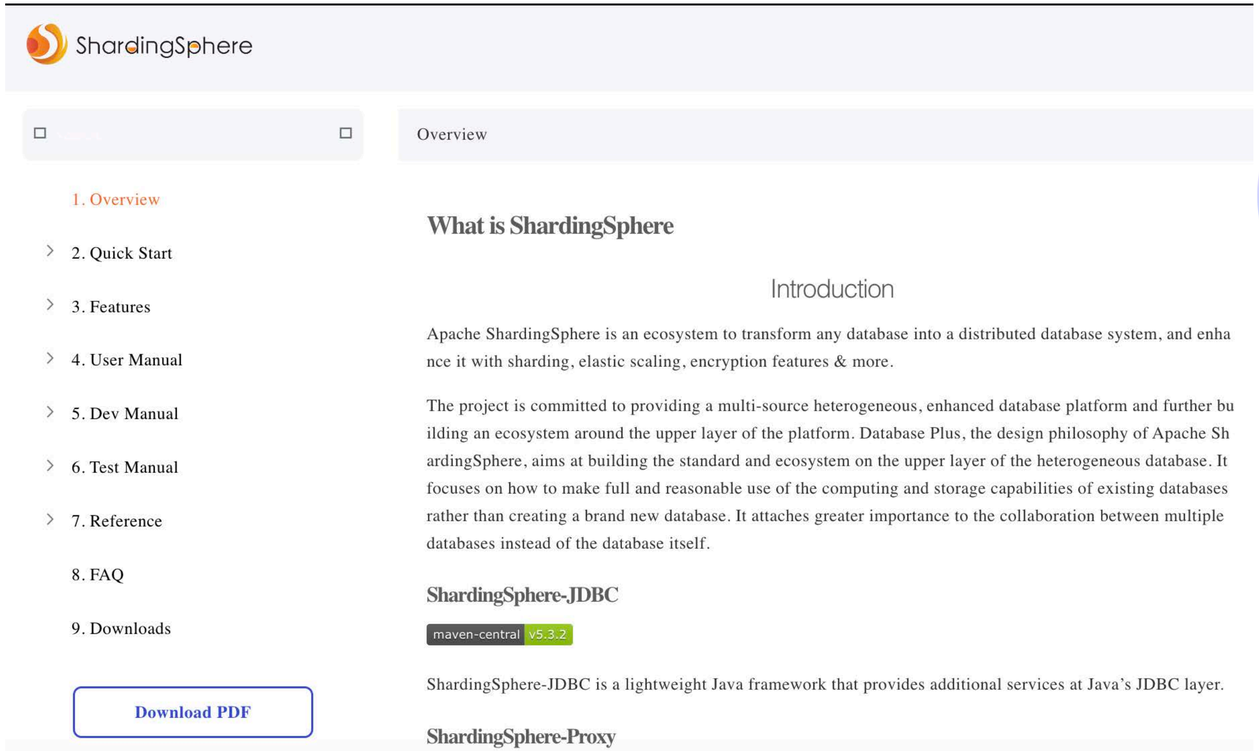
What's ShardingSphere?

The distributed SQL transaction & query engine for data sharding, scaling, encryption, and more - on any database.

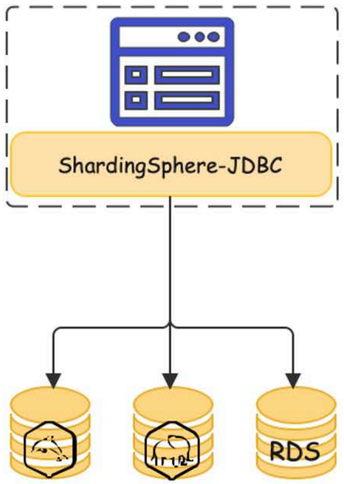


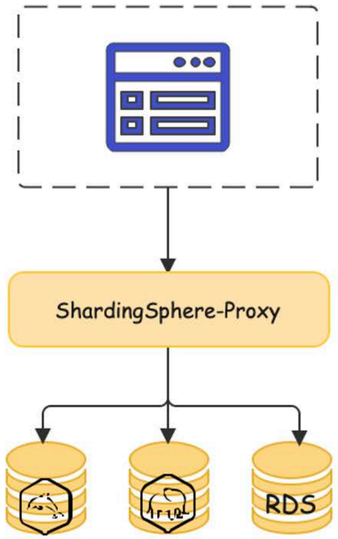


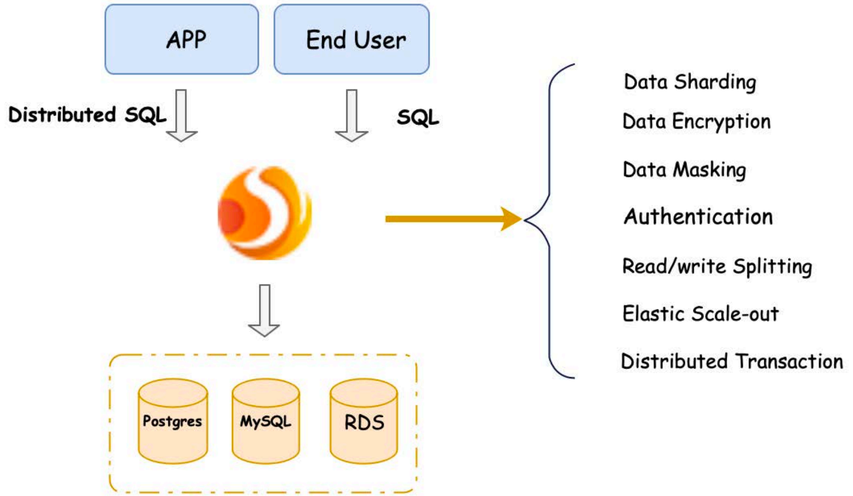




Apache ShardingSphere







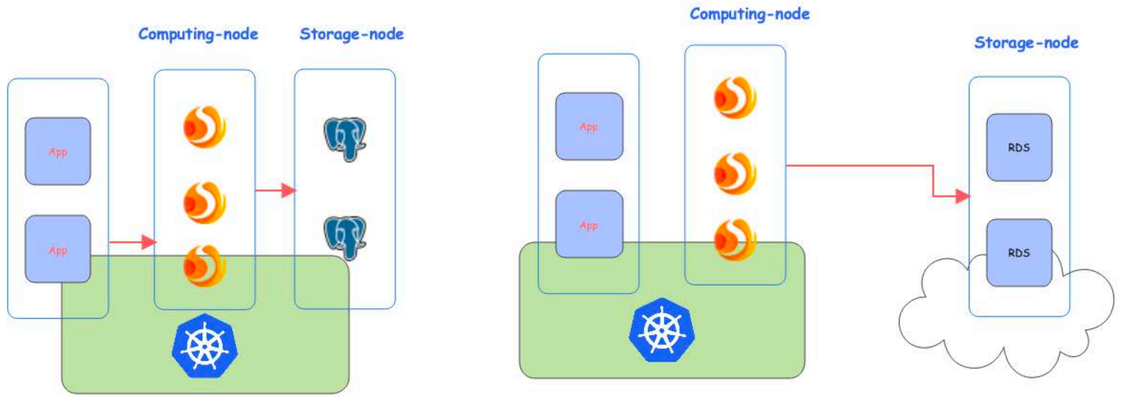
Setup on Kubernetes by on-click

ShardingSphere-on-Cloud

Take Apache ShardingSphere to the cloud

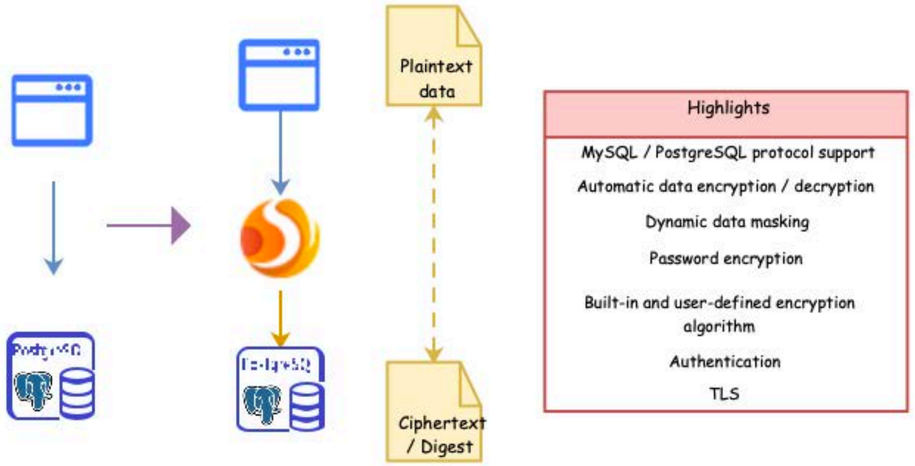
A collection of tools & best practices including automated deployment scripts to virtual machines in AWS, Google Cloud Platform, Alibaba Cloud, CloudFormation Stack templates, and Terraform one-click deployment scripts.

Helm Charts, Operators, automatic horizontal scaling, and other tools for the Kubernetes cloud-native environment are also included.

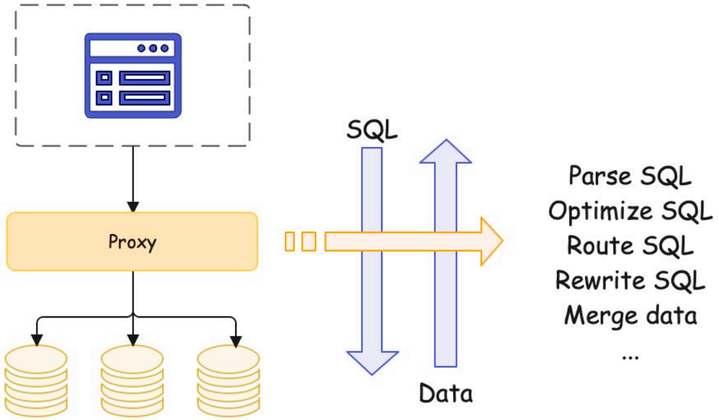


https://shardingsphere.apache.org/oncloud/

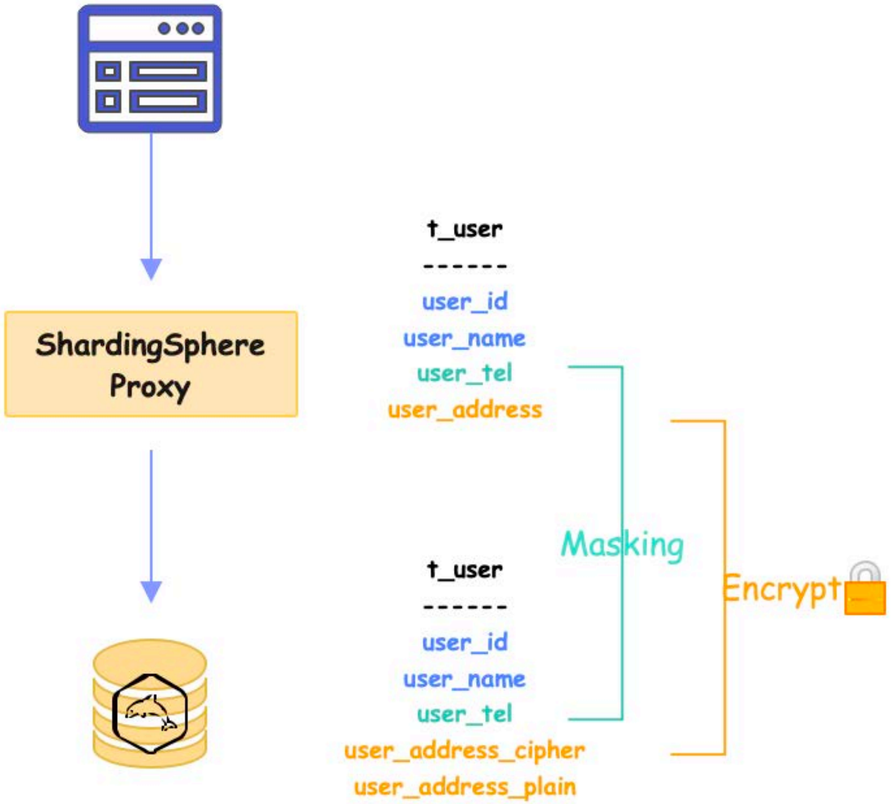
The process



Highlights
MySQL / PostgreSQL protocol support
Automatic data encryption / decryption
Dynamic data masking
Password encryption
Built-in and user-defined encryption algorithm
Authentication
TLS



Distributed SQL to define encrypted table and columns



This chapter will introduce the detailed syntax of DistSQL.

Definition

DistSQL (Distributed SQL) is Apache ShardingSphere's specific SQL, providing additional operation capabilities compared to standard SQL.

Flexible rule configuration and resource management & control capabilities are one of the characteristics of Apache ShardingSphere.

Rule Operation

- Create encrypt rule

```
CREATE ENCRYPT RULE t_encrypt (
    COLUMNS(
        (NAME=user_id,CIPHER=user_cipher,ENCRYPT_ALGORITHM(TYPE(NAME='AES',PROPERTIES('aes-key-value'='123456abc')))),
        (NAME=order_id,CIPHER =order_cipher,ENCRYPT_ALGORITHM(TYPE(NAME='RC4',PROPERTIES('rc4-key-value'='123456abc'))))
));
```

- Create encrypt table

```
CREATE TABLE `t_encrypt` (
    `id` int(11) NOT NULL,
    `user_id` varchar(45) DEFAULT NULL,
    `order_id` varchar(45) DEFAULT NULL,
    PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

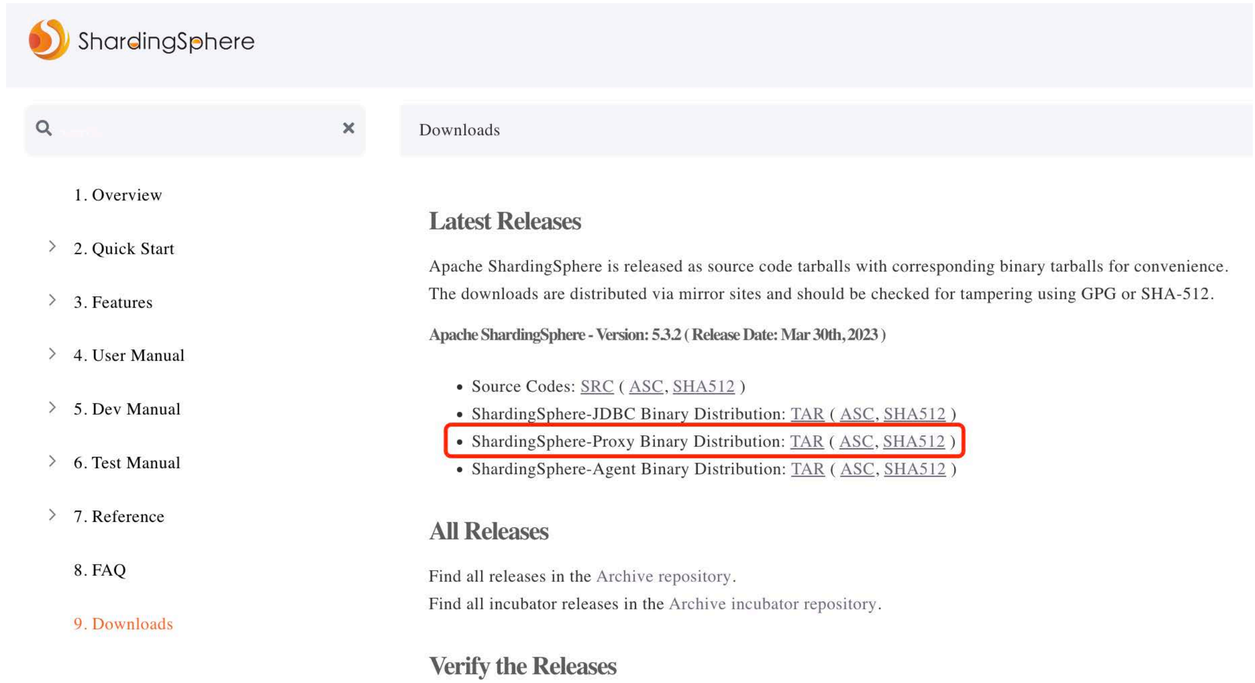
https://shardingsphere.apache.org/document/current/en/user-manual/shardingsphere-proxy/distsql/usage/mask-rule/

Demo show

1. Deploy a MySQL instance
2. Deploy a ShardingSphere-Proxy
3. Execute DistributedSQL to let ShardingSphere-Proxy recognize this MySQL instance
4. Create a table t_user on ShardingSphere-Proxy with encrypted rule and data masking rule
5. Show the description of this table
6. INSERT data for test on ShardingSphere-Proxy
7. Execute SELECT query to do test

Demo show

Download and install ShardingSphere-Proxy and start a MySQL instance



https://shardingsphere.apache.org/document/5.3.2/en/downloads/

Demo show

Download and install ShardingSphere-Proxy and start a MySQL instance

```
create database demo_db;
```

```
mysql> create database demo_db;
Query OK, 1 row affected (0.04 sec)

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| demo_db            |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.07 sec)
```

```
REGISTER STORAGE UNIT ds_0 (
    HOST="192.168.10.23",
    PORT=3306,
    DB="demo_db",
    USER="test",
    PASSWORD="test"
);
```

```
mysql> REGISTER STORAGE UNIT ds_0 (
    ->     HOST="192.168.10.23",
    ->     PORT=3306,
    ->     DB="demo_db",
    ->     USER="root",
    ->     PASSWORD="sphereEx@2021"
    -> );
Query OK, 0 rows affected (0.40 sec)
```

Demo show

Create encrypted and data masking rule for table t_user

```
mysql> CREATE MASK RULE t_user (
    -> COLUMNS(
    -> (NAME=user_telphone,TYPE(NAME='KEEP_FIRST_N_LAST_M',PROPERTIES("first-n"=3,"last-m"=4,"replace-char"="*")))
    -> ));
Query OK, 0 rows affected (0.07 sec)

mysql>
mysql>
mysql> CREATE ENCRYPT RULE t_user (
    ->   COLUMNS(
    ->     (
    ->       NAME = user_address,
    ->       PLAIN = user_address_plain,
    ->       CIPHER = user_address_cipher,
    ->       ENCRYPT_ALGORITHM(
    ->         TYPE(
    ->           NAME = 'AES',
    ->           PROPERTIES('aes-key-value' = '123456abc')
    ->         )
    ->       )
    ->     )
    ->   ),
    ->   QUERY_WITH_CIPHER_COLUMN = true
    -> );
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> SHOW ENCRYPT TABLE RULE t_user\G
*************************** 1. row ***************************
                  table: t_user
           logic_column: user_address
          cipher_column: user_address_cipher
           plain_column: user_address_plain
   assisted_query_column:
       like_query_column:
          encryptor_type: AES
         encryptor_props: aes-key-value=123456abc
     assisted_query_type:
    assisted_query_props:
         like_query_type:
        like_query_props:
query_with_cipher_column: true
1 row in set (0.00 sec)
```

```
mysql> SHOW MASK TABLE RULE t_user\G
*************************** 1. row ***************************
          table: t_user
         column: user_telphone
 algorithm_type: KEEP_FIRST_N_LAST_M
algorithm_props: first-n=3,last-m=4,replace-char=*
1 row in set (0.01 sec)
```

```
1 CREATE TABLE t_user (
2     user_id INT,
3     user_name VARCHAR(50),
4     user_telphone VARCHAR(50),
5     user_address VARCHAR(200)
6 );
```

```
mysql> CREATE TABLE t_user (
    ->     user_id INT,
    ->     user_name VARCHAR(50),
    ->     user_telphone VARCHAR(50),
    ->     user_address VARCHAR(200)
    -> );
Query OK, 0 rows affected (0.03 sec)
```

Demo show

Insert test data and query to test the function

```
mysql> INSERT INTO t_user (user_id, user_name, user_telphone, user_address)
    -> VALUES (1, 'Olivia','111-123-4567','123 Main St, NY' ),
    ->        (2, 'Ethan ','222-234-5678','456 Smith Ave, CA' ),
    ->        (3, 'Ava   ','333-345-6789','789 Park Rd, TX' ),
    ->        (4, 'Noah  ','333-456-7890','1010 Elm St, IL' ),
    ->        (5, 'Emma  ','444-567-8901','555 Broadway, NY' ),
    ->        (6, 'Mason ','555-678-9012','777 Oak Ln, CA' ),
    ->        (7, 'Mia   ','666-789-0123','999 Maple Rd, IL' ),
    ->        (8, 'Liam  ','777-890-1234','333 Pine Dr, FL' ),
    ->        (9, 'Harper','888-901-2345','444 Cedar St, TX' ),
    ->        (10,'Lucas ','999-012-3456','888 Beach Blvd, FL' );
Query OK, 10 rows affected (0.01 sec)
```

```
mysql> select * from t_user;
+---------+-----------+---------------+-------------------+
| user_id | user_name | user_telphone | user_address      |
+---------+-----------+---------------+-------------------+
|       1 | Olivia    | 111*****4567  | 123 Main St, NY   |
|       2 | Ethan     | 222*****5678  | 456 Smith Ave, CA |
|       3 | Ava       | 333*****6789  | 789 Park Rd, TX   |
|       4 | Noah      | 333*****7890  | 1010 Elm St, IL   |
|       5 | Emma      | 444*****8901  | 555 Broadway, NY  |
|       6 | Mason     | 555*****9012  | 777 Oak Ln, CA    |
|       7 | Mia       | 666*****0123  | 999 Maple Rd, IL  |
|       8 | Liam      | 777*****1234  | 333 Pine Dr, FL   |
|       9 | Harper    | 888*****2345  | 444 Cedar St, TX  |
|      10 | Lucas     | 999*****3456  | 888 Beach Blvd, FL |
+---------+-----------+---------------+-------------------+
10 rows in set (0.01 sec)
```

VS

```
mysql> SHOW CREATE TABLE t_user;
+--------+------------------------------------------
| Table  | Create Table
                                                    |
+--------+------------------------------------------
| t_user | CREATE TABLE `t_user` (
  `user_id` int(11) DEFAULT NULL,
  `user_name` varchar(50) DEFAULT NULL,
  `user_telphone` varchar(50) DEFAULT NULL,
  `user_address_cipher` varchar(200) DEFAULT NULL,
  `user_address_plain` varchar(200) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 |
+--------+------------------------------------------
1 row in set (0.00 sec)

mysql>
mysql> SELECT * FROM t_user;
+---------+-----------+---------------+----------------------------------------------+--------------------+
| user_id | user_name | user_telphone | user_address_cipher                          | user_address_plain |
+---------+-----------+---------------+----------------------------------------------+--------------------+
|       1 | Olivia    | 111-123-4567  | cebM4llQQIvKk3wbjCG8NQ==                     | 123 Main St, NY    |
|       2 | Ethan     | 222-234-5678  | KUSrSjZcwGKET+ZDvkBnOJiOEClgm1K0kDlafjO/+uk= | 456 Smith Ave, CA  |
|       3 | Ava       | 333-345-6789  | tRbooaDDWghOI3B6PUWi7w==                     | 789 Park Rd, TX    |
|       4 | Noah      | 333-456-7890  | bBGl4bOAVevf/FyIdeRPMg==                     | 1010 Elm St, IL    |
|       5 | Emma      | 444-567-8901  | rfSp9MxNlBozIlMTBta+eAq2ZLhWcqfQ8/EQnIqMx+g= | 555 Broadway, NY   |
|       6 | Mason     | 555-678-9012  | NK5SNym5MNLmn0eHjKkNbA==                     | 777 Oak Ln, CA     |
|       7 | Mia       | 666-789-0123  | MOn3XXmSC6Zcx1/Y/EYVZAq2ZLhWcqfQ8/EQnIqMx+g= | 999 Maple Rd, IL   |
|       8 | Liam      | 777-890-1234  | TNfMDNA/47h5aXjn9GzSPw==                     | 333 Pine Dr, FL    |
|       9 | Harper    | 888-901-2345  | ND3LcNpVd+QYdqpww8gi4Aq2ZLhWcqfQ8/EQnIqMx+g= | 444 Cedar St, TX   |
|      10 | Lucas     | 999-012-3456  | HSuHGZpidkNHrnUNXWH6WxGPK+n6ssFMRDGfbua8gng= | 888 Beach Blvd, FL |
+---------+-----------+---------------+----------------------------------------------+--------------------+
10 rows in set (0.00 sec)
```

Query from ShardingSphere-Proxy

Query from MySQL

THAKNS

SphereEx | SphereEx, linking data and services simply

Bio: https://tristazero.github.io

LinkedIn: https://www.linkedin.com/in/panjuan

GitHub: https://github.com/tristaZero

Twitter: @tristaZero

Project Twitter: @ShardingSphere