SingleStoreDB Cloud Security White Paper

Table of Contents

Security at SingleStore

At SingleStore, data security is a top priority and focus area for our SingleStoreDB Cloud offering. Backed by cybersecurity professionals with decades of experience and industry leading certifications in cloud security, security is embedded into everything we do — from software development lifecycle processes to operations of our SingleStoreDB cloud on AWS, Azure or GCP.

Our secure by design approach lays a foundation of security providing a defence in depth posture for both on-premise and cloud managed deployments of SingleStore products. Through continuous evaluation of the security properties of confidentiality, integrity, and availability, paired with validation of authentication, authorization and auditing controls, we maintain the highest levels of security assurance for SingleStore products and services.

Overview

SingleStoreDB Cloud offers the world's first fully-managed unified cloud database bringing together transactions and analytics, ingest and query performance, scalability, and resiliency while autonomously handling installation, operations, and management-related details. SingleStoreDB Cloud is designed and architected to handle and recover from failures automatically. To maintain each cluster's availability, SingleStoreDB Cloud runs on a highly resilient software cloud infrastructure with built-in high availability (HA).

The SingleStoreDB cloud Service is composed of two distinct components; the **Control Plane** and the **Data Plane**. The Control Plane is an online portal that provides a user with a unified view of all SingleStoreDB Cloud deployments in one place; it provides insights into performance metrics, resource utilization, cluster health, security, and access control. The control plane is also responsible for the entire orchestration and deployments of the customers cluster resources. On the other hand the Data Plane refers to the compute and the storage within the services that includes the database clusters. It can be deployed in the cloud across a number of cloud providers, regions, or on premises in a customer datacenter.

SingleStoreDB Cloud Editions

SingleStore provides several editions of SingleStoreDB Cloud to meet the unique needs of a variety of customer workloads. These include - Standard, Premium, and Dedicated - each tailored to specific customer workloads and feature needs. Each edition shares the Control Plane infrastructure, but may have differences in the way the Data Plane is deployed and managed to meet unique constraints around compliance and data governance.

Control Plane

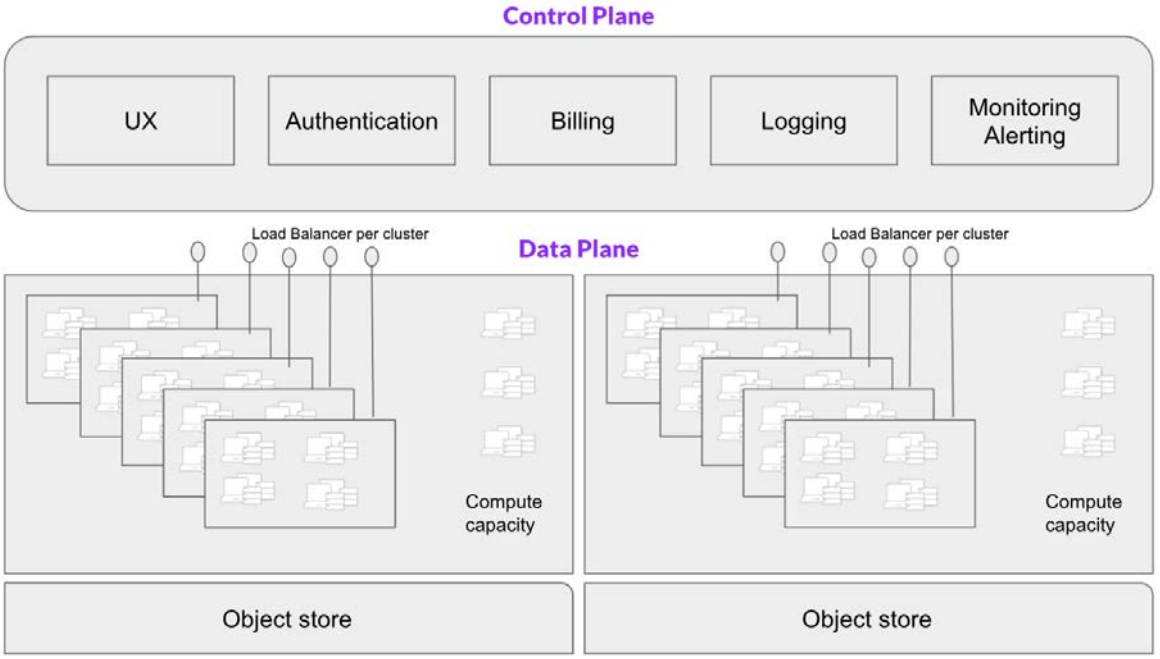
The control plane for SingleStoreDB Cloud consists of various services which talk to each other via HTTPS APIs. All internal connectivity uses TLS 1.2. These services interact with the data plane via the Kubernetes REST API.

The primary service exposes an HTTPS API to the administrative portal (frontend). The customer-facing portal runs in a web browser (portal.singlestore.com) and the static assets are stored on AWS S3 and served worldwide via AWS Cloudfront. Comparable approaches are available in GCP and Azure.

The HTTPS API used between the control plane and the data plane is secured using signed JSON Web Tokens (JWT tokens) with HTTP Authentication. These JWT tokens are generated by our identity provider and identify a single user of the SingleSToreDB cloud. The mapping between users and their respective organizations is managed by the control plane.

The control plane has a data store which is used to store organizations, metadata about all clusters, billing data, quotas, etc.

Figure 1. SingleStoreDB Cloud Control Plane / Data Plane Architecture



Data Plane

The Data Plane is implemented as one or more containerized deployments per region. Each containerized deployment manages the compute, storage, and load balancers used to host customer SingleStoreDB clusters. Compute (memory, CPU and local disk) is isolated for each customer using a container running on the Host VM. Clusters share an object store bucket, however, they are segregated by keys (similar to path or subdirectory). So most resources are isolated but things like network bandwidth are shared. Each deployment is in a single cloud provider account.

SingleStoreDB Cloud leverages the power of containerization to automate functionality and provide a seamless user experience. Container architecture is used for the Data Plane shown in *Figure 1*. This is coupled with the control plane for managing the overall cluster, and the data plane to provide capacity such as CPU, memory, network, and storage so that the containers can run and connect to a network.

The control plane communicates with the data plane using a finely-scoped, well-defined API. This connection is secured by TLS, secure certificate and access key authentication, over a private non-routable network. A custom "SingleStore Cluster" type is created in the deployment, and the control plane manages these resources. An in-cluster process ("operator") reads those custom resources and acts against clusters in the data plane.

If a customer wishes to delete their cluster, a request is issued through the administrative portal. The control-plane will issue an API request to the data plane to deallocate the cluster. The cluster is shutdown and billing charges will stop accruing at this point. Encrypted data from terminated clusters is automatically and securely purged from all systems on termination. Hence, it is not possible to recover data from a terminated deployment. If you need to retain data post termination, it is recommended that you backup your data to an external object storage bucket.

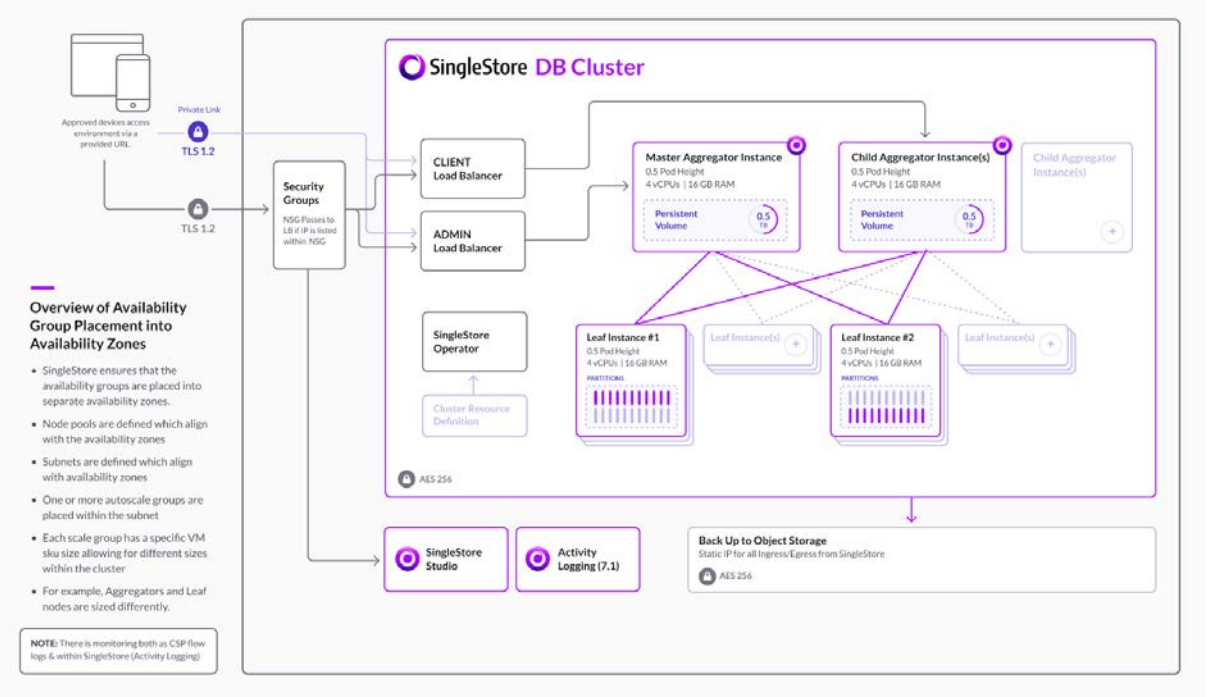


Figure 2. SingleStoreDB Cloud Data Plane Architecture

Data Plane Architecture for Product Editions

Standard, Premium

The data plane for the Standard and Premium editions is deployed to a single region in AWS, Azure, or GCP, and clusters are isolated at the k8s namespace. Multiple customer clusters are provisioned in the shared data plane but have isolated local storage. Use of containers ensures memory and CPU resources are aligned to specific customers. Data plane resources needed for operation such as orchestration, logging, job scheduling, and data security are also aligned on a per cluster basis.

Dedicated

The Dedicated offering is structured as follows: a dedicated instance of the data plane aligned to a single customer, giving them further isolation beyond what customers of Standard and Premium get. i.e. all host machines contain only compute nodes from their organisation/workloads. These dedicated deployments show up as "private regions" in our portal but only available for users from that company. Users can do exactly the same operations as they can with a standard or premium edition. The dedicated offering can be implemented in any cloud vendor.

The region hosting data is a dedicated environment, and is not shared with any other customer. The keys used to encrypt the data are backed by KMS and the Encryption Keys are encrypted and stored within the deployment. This dedicated deployment supports connectivity between environments (AWS PrivateLink, Azure Private Link, GCP Private Service Connect) which enables private network access from customer IP addresses. The authentication and entitlement is performed by resources in the deployment with support for federated authentication. Each deployment, as shown in *Figure 2.*, is in a dedicated cloud account; it has a dedicated deployment, with uniquely addressed networks. ACLs are in place to allow management only through designated bastion hosts, to which access requires a connection to a certificate-based VPN.

Datacenters

SingleStoreDB Cloud runs on the cloud infrastructure of the top three public cloud providers: Amazon Web Services (AWS), Google Cloud, and Microsoft Azure. Customer data is stored in SingleStoreDB clusters and customers can choose which SingleStoreDB cluster edition they would like to run between Standard, Premium and Dedicated editions. All the clusters are running on Virtual machines managed in a Kubernetes environment.

Public cloud provider's data centers follow strict compliance guidelines that SingleStore inherits as part of the service orchestration. Along with the key security capabilities that SingleStore builds on. For additional information, please refer to the compliance page of your selected cloud provider:

- AWS compliance
- Azure compliance
- Google Cloud Compliance

Information Security and Privacy Certifications

SingleStore has undergone an 18 month project to transform the way the business approaches information security and data privacy. We have now secured industry-leading security certifications including ISO 27001 and SOC 2 Type 2. SingleStore is also fully compliant to the requirements of HIPAA, CCPA and GDPR.

As we grow, SingleStore continues to mature its information security posture and is passionate about meeting the security and compliance requirements of our customers.

Connectivity

At SingleStore we empower our customers to help secure their data from unauthorised users. We use a layered approach to security, starting with IP allowlisting to ensure only devices you trust, and have given access to, can access your cluster or your data. We then ensure that the data passing between your trusted devices and SingleStoreDB Cloud is encrypted with TLS 1.2 to protect it from being intercepted during transit.

SingleStoreDB Cloud supports private connectivity between VPCs (AWS PrivateLink, Azure Private Link, and GCP Private Service Connect), which ensures data does not leave a customer's environment and is not exposed to the wider internet.

Access to the service can be achieved either directly via the internet, through private connectivity, or via VPC endpoints. When using VPC endpoints, connectivity is bound between the endpoint in the dedicated service and your environment. These must reside within the same region.

Support and development teams are provided with read-only access to telemetry and observability data. The data here does not contain any personal identifiable information (PII) data or query text from the customers. Administrative access to the control plane is limited to the SingleStore SRE team only. A list of administrators is maintained and regularly reviewed during hire, annual review process and at termination. Access requires connectivity to a certificate-based VPN, and additional connection to a bastion host. Administrative actions require additional short-lived authentication and authorization to connect to API endpoints. Additionally, SingleStore also supports multi-person approval workflow for Just In-time (JIT) requests for access to customers' environments. These requests require both internal and customer approvals before access to the infrastructure is granted. Access to these services is logged for audit purposes.

For cloud services IAM roles there are dedicated secrets in each environment. These credentials are stored in an encrypted database and rotated on an as-needed basis.

The SingleStore SRE team can update security groups; however these actions are logged in the portal and in logging systems. Our SRE team can also access leaf nodes which are logged in the audit trail. This is granted based on least privilege. The SREs are trained on their responsibilities annually and access is reviewed quarterly and removed when no longer appropriate/required or when an employee leaves SingleStore.

Cloud Native Connectivity

SingleStoreDB Cloud supports cloud native connectivity by exposing a cluster endpoint for clients and applications to connect to. A SingleStore cluster endpoint looks like this:

[`mysql -u admin -h svc-ee432f9e-37a2-4236-a8fa-05936b6e292d-ddl.aws-ireland-1.svc.singlestore.com -P 3306 --default-auth=mysql_native_password -p`]

Endpoints are accessible only from IP addresses which have been allowlisted in SingleStore's firewall rules, and connections are authenticated and secured using TLS/SSL. Complete details on connecting to your cluster from a command line or from client tools can be found here.

In addition to securing and encrypting all connections to the SingleStoreDB cluster, the service also supports private network routing across all major platforms, including AWS PrivateLink, Azure Private Link and Google Private Service Connect. Please contact us for more information on connecting to your deployment from AWS, GCP or Azure.

Configure the SingleStoreDB cloud Connection

SingleStore can be connected to a number of clients including:

- Collibra Data Governance
- DBeaver
- JetBrains DataGrip
- MySQL
- SQL Developer
- SQL Editor
- Sequel Pro
- SQL Workbench
- SQuirrel SQL

Information on how to connect via each client is available on our documentation here.

Authentication

SingleStoreDB Cloud portal authenticates with your SingleStore account using token based secure authentication. This is then used across the SingleStoreDB Cloud Portal.

SingleStoreDB Cloud provides a customer admin with the power to provision and control access within their organisation and to take responsibility for who can see what and when. Authentication is based on username and password. Password complexity can be set by the customer, however as a standard it must meet the requirements of NIST 800-63B. All connections must be explicitly allowlisted via IP address. Credentials are encrypted and managed by SingleStore within its secure credentials store.

Federated authentication for the management portal supports SAML and OIDC delegation to a customer's IDP.

In order to combat authentication token reply from unauthorised endpoints our SAML IDP will act as an intermediary to your IDP for our SP (i.e. portal.) The login process makes use of a temporary code during the exchange that is invalidated once used, thus preventing reuse. In addition, access to the endpoints is protected by the aforementioned IP allowlisting.

IP Allowlisting

There are multiple ways that users can ensure only a select number of IPs can communicate with their SingleStoreDB cloud cluster.

First and foremost, users get the option to add a new IP or their current IP during the cluster creation process. Users also get the opportunity to open access to any IP, but this is not the recommended approach. If a user chooses not to make IP selections during cluster creation or wants to modify the settings, they can do so after the fact. Contact SingleStore support if you'd like more information on configuring IP allowlisting.

SAML2 Authentication (SSO)

SingleStoreDB Cloud supports use of cloud-native Identity Providers (i.e. Okta, Ping, Azure AD) for SSO that support the SAML protocol to authenticate users to the SingleStore portal. Refer to the details here.

JWT based authentication

A JSON Web Token (JWT) is an open, industry standard (RFC 7519) that defines a compact and self-contained method for securely transmitting information between parties as a JSON object, which can be verified and trusted as it is digitally signed.

JWTs are useful for both authorization (the most common scenario for using a JWTs) and information exchange (where information can be securely transmitted between parties). JWTs can be created by both the SingleStore Cloud portal and by customer-run identity providers and used to authenticate users to database clusters.

Password Policies

SingleStoreDB Cloud enables the customer to control complexity, lifetime and reuse of passwords used by end users to connect to database clusters.

The following parameters are configurable in relation to password use.

Variable	**Description**
password_expiration_seconds	The time in seconds before a password expires. The default value is 0, which indicates that passwords will never expire.
expire_root_password	Specifies whether the root password can expire. The default value is OFF. When set to OFF, the password_expiration_seconds duration does not apply to the root password. If set to ON, the root password will expire after the password_expiration_seconds duration is reached.
password_history_count	Restricts the reuse of previous user passwords. This variable is the number of previous passwords per user that SingleStoreDB will store and disallow from reuse. SingleStoreDB will disallow setting a user account's password to one of the last password_history_count number of passwords for that user. The count includes the current password. For example, if set to 2, setting a user's password to its current password or the last password before the current password is disallowed. The default value is 0, which indicates that any previous password can be reused. The maximum is 10.
password_min_length	The minimum number of characters required. These variables control password complexity requirements. The default value is 0 (disabled) and the accepted values are integers from 0 to 100.
password_min_uppercase_chars	The minimum number of uppercase characters required. The default value is 0 (disabled) and the accepted values are integers from 0 to 100.

password_min_lowercase_chars	The minimum number of lowercase characters required. The default value is 0 (disabled) and the accepted values are integers from 0 to 100.
password_min_numeric_chars	The minimum number of numeric digit characters required. The default value is 0 (disabled) and the accepted values are integers from 0 to 100.
password_min_special_chars	The minimum number of special (non-alphanumeric) characters required. The default value is 0 (disabled) and the accepted values are integers from 0 to 100.
password_max_consec_sequential_chars	The maximum number of consecutive characters allowed. For example, if set to 3, passwords with a 4-letter sequence or longer (e.g “1234” or “abcd”) are disallowed.
password_max_consec_repeat_chars	The maximum number of consecutive repeated characters allowed. For example, if set to 3, passwords with 4 or more consecutive repeated characters (e.g., “aaaa” or “1111”) are disallowed.

When a user enters a new password, if the password does not meet the complexity policy, the following error message is returned:

Error: password does not meet the requirements specified for <variable> in your password complexity policy. Password not changed.

Example Password Complexity Usage

The following stored procedure (you can also use individual SET statements in the command line) will create a password complexity policy where passwords must:

- be at least 12 characters long
- include at least one uppercase character
- include at least one lowercase character
- include at least one numeric character
- include at least one special character

```
CREATE DATABASE db_security
USE db_security

DELIMITER //
CREATE OR REPLACE PROCEDURE set_password_complexity_policy() AS
BEGIN
        SET GLOBAL password_min_length=12;
        SET GLOBAL password_min_uppercase_chars=1;
        SET GLOBAL password_min_lowercase_chars=1;
        SET GLOBAL password_min_numeric_chars=1;
        SET GLOBAL password_min_special_chars=1;
END
//
DELIMITER ;
```

CALL set_password_complexity_policy();

Administration

Role Based Access Control (RBAC)

SingleStore utilises a role based access model in the database with the following roles:

Role	Description
Compliance Officer	Management for roles and schema authorizations.
Security Officer	Full authority to view, modify, and create users and groups. Manages user passwords.
Database Administrator	This role cannot execute backups, nor can it read any of the data within the database. Responsible for creating and removing databases. Ability to restore backups.
Backup Operator	Authorization to perform cluster backups. ***Note** - all backups created are encrypted at rest based on the customer security requirements.*
Application Schema Owner	Dedicated, per-application role, authorized to execute, create, alter, and delete DDL statements. Cannot view application data.
Application Service Account	Dedicated, per-application role, authorized to execute select, update, insert, delete DML.

SingleStore recommends the following roles be used as a starting point for all use of the RBAC functionality. It is strongly suggested that these commands be kept in a separate, version-controlled, file and loaded into SingleStoreDB. These scripts should be executed on all nodes where users will connect, typically all aggregators and on all leaves if the application's design requires it to bypass the aggregators. Below is an example of how a role is configured:

```
CREATE ROLE 'compliance_role';
GRANT USAGE on *.* to ROLE 'compliance_role' WITH GRANT OPTION;

CREATE GROUP 'compliance';
GRANT ROLE 'compliance_role' to 'compliance';
```

Users can access a database and execute their functions and responsibilities through the creation of users, roles and groups, and granting of correct permissions (privileges) and the relationship between them. The relationship between these entities are summarised in the list below:

- A role can have multiple privileges.
- A group can have multiple roles.
- A group can have multiple users.
- A user can have multiple roles.
- A user can be assigned to multiple groups.
- Users inherit the permissions, roles of the groups they are assigned to.

Row Level Security

Row-level security in SingleStoreDB is achieved by creating a view on a table with a special roles column.

For a table to be used with row-level security, it must have a VARBINARY column where row entry in the column contains a comma separated list of roles which have access to that row. There are special formatting constraints for the roles columns which are discussed below.

Consider the following table containing 4 rows:

ACCESS_ROLES	DATA_1	DATA_2	DATA_3
,ROLE_A,ROLE_B,	xxxxxx	xxxxxx	xxxxxx
,ROLE_A,ROLE_C,	xxxxxx	xxxxxx	xxxxxx
,ROLE_D	xxxxxx	xxxxxx	xxxxxx
,	xxxxxx	xxxxxx	xxxxxx

For a given role, the ACCESS_ROLES field will be used to specify which roles have access to that row. The DATA_1 through DATA_3 columns are data stored in a table.

It is important that each role name in ACCESS_ROLES be surrounded by a preceding and trailing comma.

To update an existing table to work with row-level security, use the following command:

```
ALTER TABLE <table> ADD COLUMN ACCESS_ROLES VARBINARY(<SIZE>) DEFAULT ",";
```

Once again it is important to set the default value of the ACCESS_ROLES column to a comma (",") for row-level security control to function correctly.

Encryption

Encryption encapsulates the processes and controls used to ensure data remains inaccessible to unauthorised users and to protect data between the end user, client apps, and servers involved. In accordance with industry guidelines and best practice SingleStoreDB Cloud applies both encryption to data in transit and data at rest.

Encryption at Rest

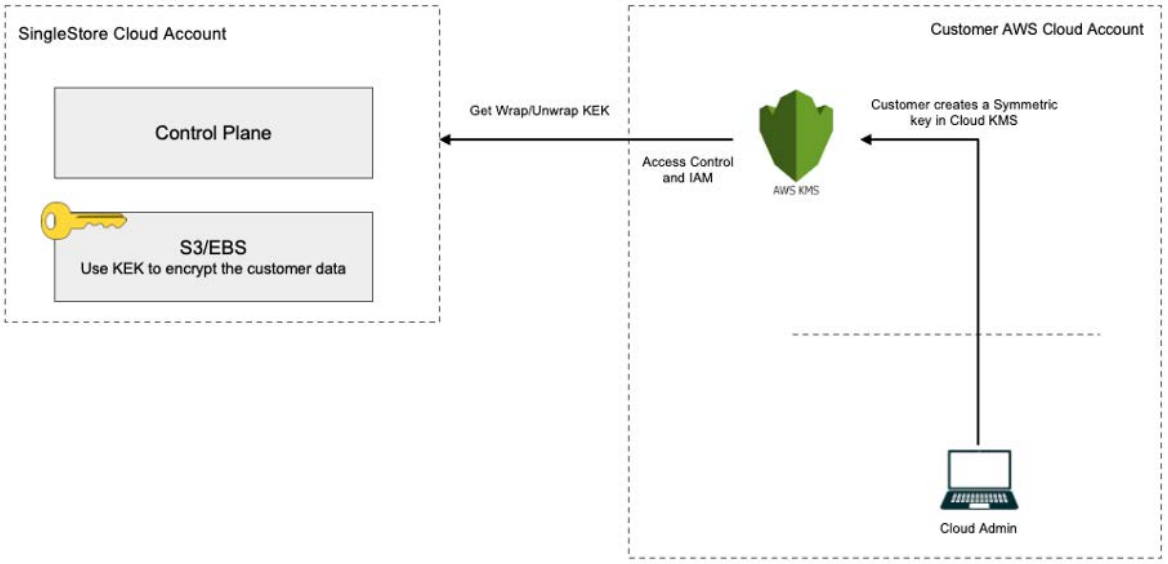
Data at Rest - SingleStoreDB Cloud utilises the best practice solution provided by the cloud hosting partner, which is AES-256 for AWS, Azure, and Google Cloud. This is an encryption algorithm using a 256-bit key length and is currently the strongest encryption algorithm(supported by the cloud provider) available from our cloud hosting partners. In a standard configuration, the cloud-managed KMS key is used to encrypt persistent-storage (i.e., EBS & S3) at rest.

Encryption at rest using Customer managed encryption key

Customers who require additional control of data at rest encryption may configure SingleStore to use an encryption key that they manage through the Cloud providers key vault (i.e., AWS KMS, Azure Key Vault, GCP KMS). This allows customers to import their own encryption key and control access to it. If a customer chooses to revoke access to this key, or delete the key, encrypted data will be unable to be decrypted by SingleStore, and anyone else, without the key. This support helps customers meet their corporate policy requirements and align with Zero Trust principles.

Customer managed encryption keys control access to data through their configuration as Key Encrypting Keys (KEK). KEKs are used to encrypt the data keys that actually encrypt and decrypt data at rest. When data at rest is encrypted using this method, access to the KEK is needed to unwrap the data key(s) that will be used to decrypt the data at rest. If the KEK is not available the data keys can not be unwrapped and the data can not be decrypted. Details of the encryption method available here.

Refer to the layout of the customer managed encryption for data protection at rest.



The customer should use the cloud provider's key management service to generate or import an encryption key that they have control over. The customer should grant permissions to SingleStoreDB Cloud to use this key to protect EC2 volumes and the S3 bucket. If at any time the customer wants to prevent decryption of their data in the cloud they can simply revoke access to the customer's managed key or remove the key.

Currently this is only supported for the Dedicated edition. Using the cloud KMS if the key is withdrawn (unshared) then SingleStore will no longer be able to read persistent storage providing customer control of the data.

Note: SingleStoreDB Cloud will be unable to decrypt any data at rest that is protected by a customer managed key if access to the key is removed or if the key is deleted from the environment. Additionally, deleted customer managed keys are the responsibility of the customer and can not be recovered by SingleStore. Also, refer to the best practices and control available for using AWS

managed customer managed key encryption here. The key provided by the customer is used to protect all the clusters in the account.

Encryption of data in Motion/Transit

Data in transit - For all connections to the database SingleStoreDB Cloud supports TLS 1.2. Transport Layer Security (TLS) uses a combination of symmetric and asymmetric encryption focusing on the uses of key pairs, a public key and a private key.

To ensure a secure connection to SingleStoreDB Cloud, SQL clients must be properly configured to both require a secure connection and to verify the supplied server certificate. If a customer has REQUIRE SSL configured, then users will not be able to connect without SSL configured. Not having TLS configured can compromise security and lead to man-in-the-middle attacks, where a would-be attacker can impersonate a server when SSL is disabled or create a secure connection by impersonating a server using an illegitimate server certificate.

Additionally, for data movement within the clusters including the distributed join operations across leaf nodes, data transmission between the cluster and the blob stores (AWS S3, Azure blob store or Google cloud storage) always uses TLS 1.2 for encryption of data in motion.

Logging and Monitoring

With data being today's currency it's important to know who can access it, who has viewed it, and why. SingleStoreDB Cloud currently has access to all internal logs ensuring there is a full audit trail.

The internal telemetry feed is available to SingleStore site reliability engineers and only aggregated usage statistics are available to the product team. Observability data stored in the customer tenant is currently kept indefinitely, this will be limited to 30-day retention.

SingleStoreDB Cloud provides cluster monitoring to customers through cluster reports. This enables you to proactively understand the activity within your cluster and react to abnormal behaviours.

Audit Logging

SingleStore captures and manages all log data within a customer cluster. By default, a cluster is set to ADMIN-ONLY-INCLUDING-PARSE-FAILS which captures data on all valid and invalid statements and queries.

A valid statement or query is one that can be successfully parsed by SingleStoreDB. Invalid statements or queries include those with misspellings or improper syntax. Note that user credentials and Personal identifiable information (**PII)** information contained in statements and queries is obfuscated in audit logs.

The ADMIN-ONLY-INCLUDING-PARSE-FAILS will only log statements that require administrator permissions, namely DDL operations such as CREATE, DROP, ALTER, etc. Additionally, if a query contains passwords (such as SET PASSWORD), the password's value will be omitted from the log. It will also log invalid statements that fail to parse. These invalid statements may include sensitive information that would normally be obfuscated in a log entry.

Log data is available to customers on request and will be shared in the form of a text file as shown below. For more details refer to the documentation here.

```
auditlog_node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-0-3306_2021-07-29_16-21-
20.log  auditlog_node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-0-3306_2021-07-29_17-21-
21.log
bash-4.2$ more auditlog_node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-0-3306_2021-07-
29_16-21-20.log
0,2021-07-29 16:21:20.763,INFO: Log opened on MemSQL startup. Format version 2. Log level initiated
at ADMIN-ONLY-INCLUDING-PARSE-FAILS
1,2021-07-29 16:21:22.781,UTC,node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-
0:3306,leaf,USER_LOGIN,100000,root,localhost,root@%,mysql_native_password,SUCCESS
2,2021-07-29 16:21:22.819,UTC,node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-
0:3306,leaf,USER_LOGIN,100000,root,localhost,root@%,mysql_native_password,SUCCESS
3,2021-07-29 16:21:22.843,UTC,node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-
0:3306,leaf,USER_LOGIN,100000,root,localhost,root@%,mysql_native_password,SUCCESS
4,2021-07-29 16:21:22.843,UTC,node-ab242b2d-0fc8-4243-8388-dddff7de255f-master-
0:3306,leaf,18446744073709551615,100000,root,information_schema,,425139562662118040,GRANT
ALL ON *.* TO root@'%' IDENTIFIED BY '<pas
sword>'
```

Conclusion

Data security is a core fundamental at SingleStoreDB Cloud, and we align with industry guidelines and best practices to ensure data is protected throughout its lifecycle. In addition to the systems and practices we have already implemented for data security, we have continued investment in measures and certifications to ensure SingleStore stays on the cutting edge of security.

If you have further questions regarding security on SingleStoreDB Cloud, then please reach out to us at security@SingleStore.com.