

Secrets Manager

Best Practices

Product Documentation



Copyright Notice

©2013-2019 Tencent Cloud. All rights reserved.

Copyright in this document is exclusively owned by Tencent Cloud. You must not reproduce, modify, copy or distribute in any way, in whole or in part, the contents of this document without Tencent Cloud's the prior written consent.

Trademark Notice



All trademarks associated with Tencent Cloud and its services are owned by Tencent Cloud Computing (Beijing) Company Limited and its affiliated companies. Trademarks of third parties referred to in this document are owned by their respective proprietors.

Service Statement

This document is intended to provide users with general information about Tencent Cloud's products and services only and does not form part of Tencent Cloud's terms and conditions. Tencent Cloud's products or services are subject to change. Specific products and services and the standards applicable to them are exclusively provided for in Tencent Cloud's applicable terms and conditions.

Contents

Best Practices

Hosting and Using Secrets

Rotating Hosted Secrets

Best Practices

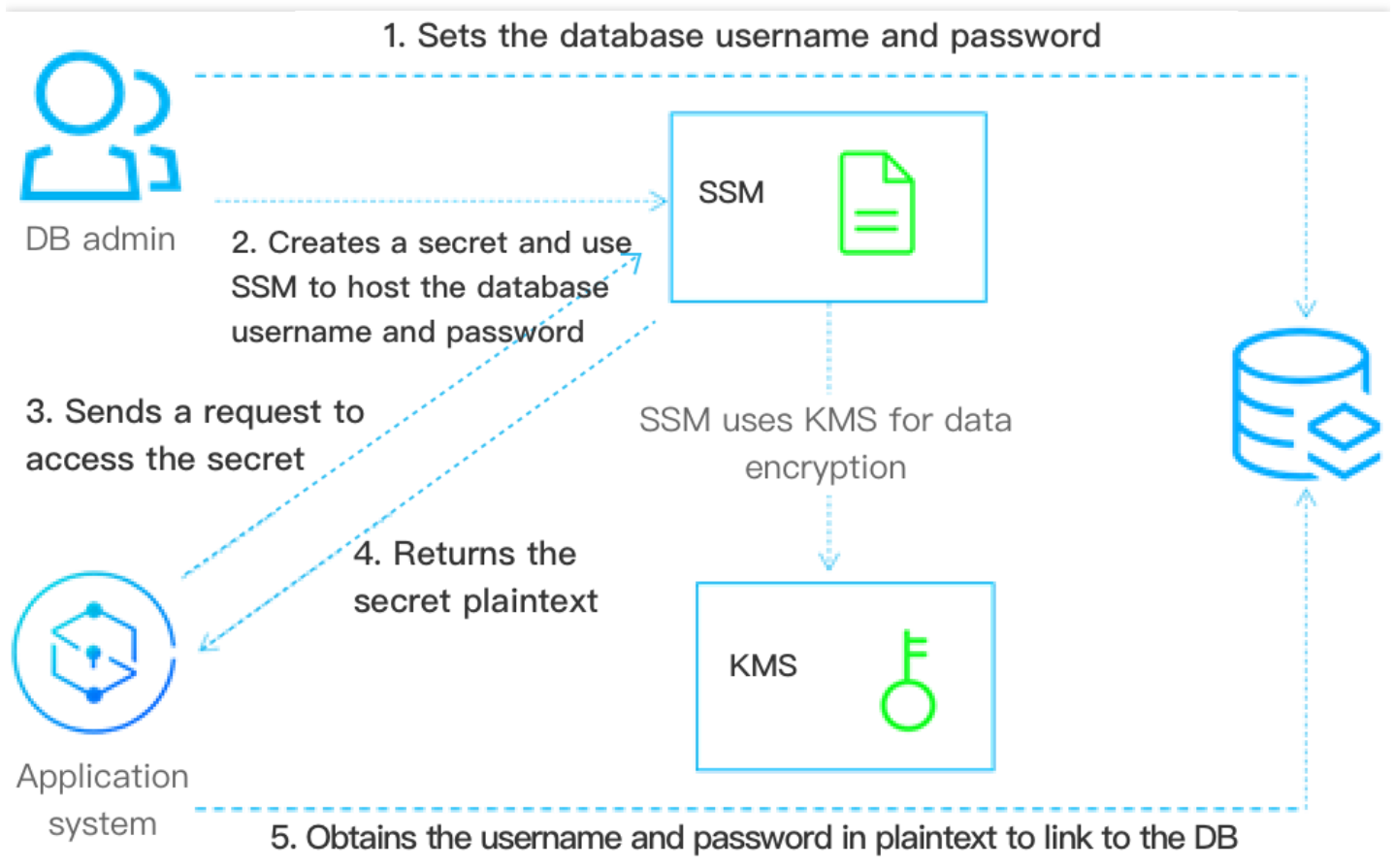
Hosting and Using Secrets

Last updated : 2020-11-16 14:10:11

Normally, various types of verification information (i.e., passwords, tokens, SSH keys, and API keys) for identity verification are embedded in the configuration file of the application as plaintext, which offers lower security. You can use SSM to encrypt and store sensitive information to avoid risks caused by the plaintext coding of sensitive secrets.

Directions

The following uses the hosted username and password of a database as an example to introduce the basic use cases of secret hosting.



1. DB admin sets the database username and password.
2. DB admin creates a secret object in SSM. The secret object is used to store the encrypted username and password obtained in Step 1.

3. When the application needs to access the database, it needs to send a request to SSM to access the secret.
4. After SSM receives the secret plaintext, it decrypts the secret and sends the secret plaintext to the application over HTTPS.
5. The application reads and parses the secret plaintext returned by SSM to obtain the username and password for accessing the target database.
6. DB admin can create multiple versions for a secret. Also, it can update the version content to implement configuration synchronization, version management, and secret rotation.

Application Effects

The application system can call the SSM APIs or SDK to obtain the sensitive secret plaintext, avoiding leakage risks caused by coding secret as plaintext in the application or configuration file. The calling comparison is as follows:

- The following are examples of storing the username and password of a database as plaintext in the local configuration or the code file, which brings a higher risk of sensitive secret leakage.
 - Sample code of obtaining the secret plaintext:

```
func GetDBConfig() string {
    dbConnStr := "user:password@tcp(127.0.0.1:3306)/test"
    return dbConnStr
}
```

- Sample code of using the secret plaintext:

```
conn, err := sql.Open("mysql", GetDBConfig())
if err != nil {
    // error handler
}
```

- The following are examples of using SSM to store the username and password for connecting to the database. It avoids storing the username and password as plaintext in the code or local configuration file.
 - Sample code of obtaining the secret plaintext:

```
func GetDBConfig(secretName, version *string) string {
    credential := common.NewCredential(
```

```
secretId,  
secretKey,  
)  
cpf := profile.NewClientProfile()  
cpf.HttpProfile.Endpoint = endpoint  
client, _ := ssm.NewClient(credential, region, cpf)  
  
request := ssm.NewGetSecretValueRequest()  
request.SecretName = secretName  
request.VersionId = version  
  
resp, err := client.GetSecretValue(request)  
if err != nil {  
    // error handler  
}  
return *resp.Response.SecretString  
}
```

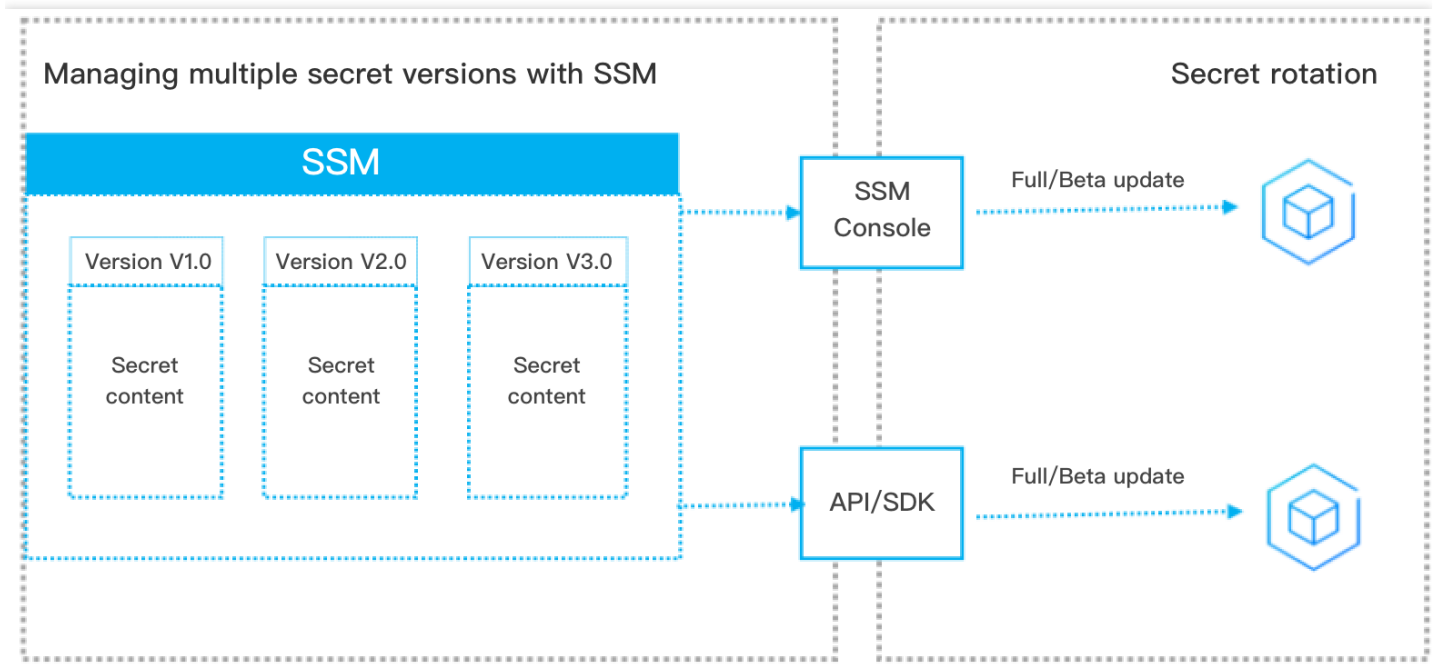
- Sample code of using the secret plaintext:

```
secretName := "MySecret1"  
version := "MyVersion1"  
conn, err := sql.Open("mysql", GetDBConfig(&secretName, &version))  
if err != nil {  
    // error handler  
}
```

Rotating Hosted Secrets

Last updated : 2021-08-18 14:21:42

To improve system security, the update of a secret needs to be synced across multiple applications and configurations. For a multi-application scenario, if secrets are stored in local files, an application might be missed, running a risk of application interruptions. SSM enables you to apply a secret update to all dependent applications. You can also create multiple versions for a secret to implement beta updates and rotation.



You can rotate secrets using either of the following ways:

- **Method 1:** add a secret version. The server can implement beta rotation by updating the secret version.

Credential Management + Add

Version Number	Operation
1.0.0	View Change Delete

- **Method 2:** update the content of the current secret. When the server calls an API to obtain the secret, the secret content is updated automatically. For more information, please see [Examples of Secret API Calls](#).