凭据管理系统

最佳实践

产品文档

腾讯云

文档目录

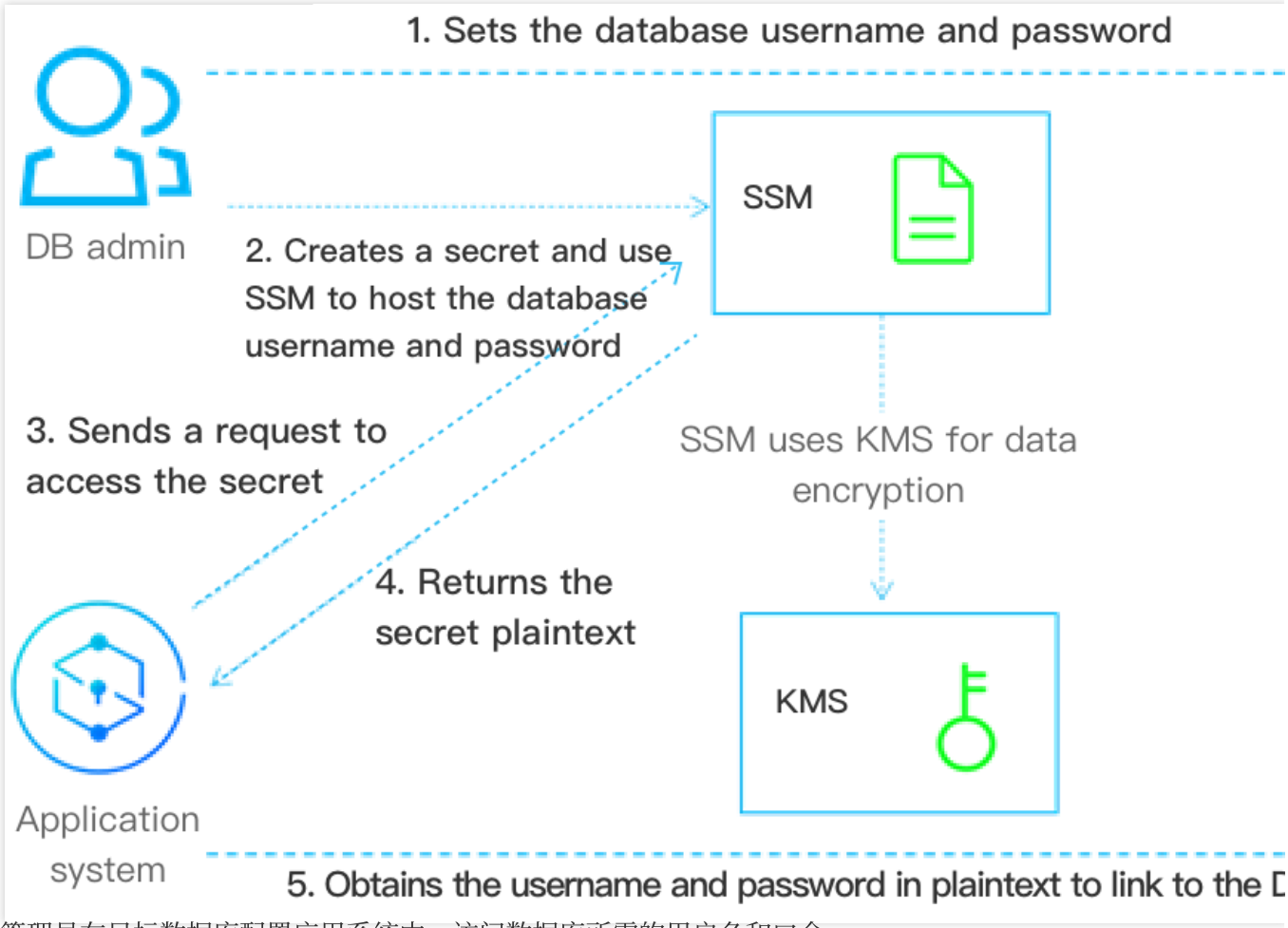
最佳实践
凭据托管和使用

最近更新时间：2024-01-02 15:13:39

应用程序或服务中，用于身份验证的各种认证信息，如口令、令牌、证书、SSH 密钥或 API 密钥等，通常情况下直接明文保存在应用程序的配置文件中，安全性较低。借助凭据管理系统将这些敏感认证信息加密存储，可有效避免敏感凭据明文编码带来的风险问题。

操作流程

以数据库用户名和口令的托管为例，介绍基本的凭据托管和使用场景。

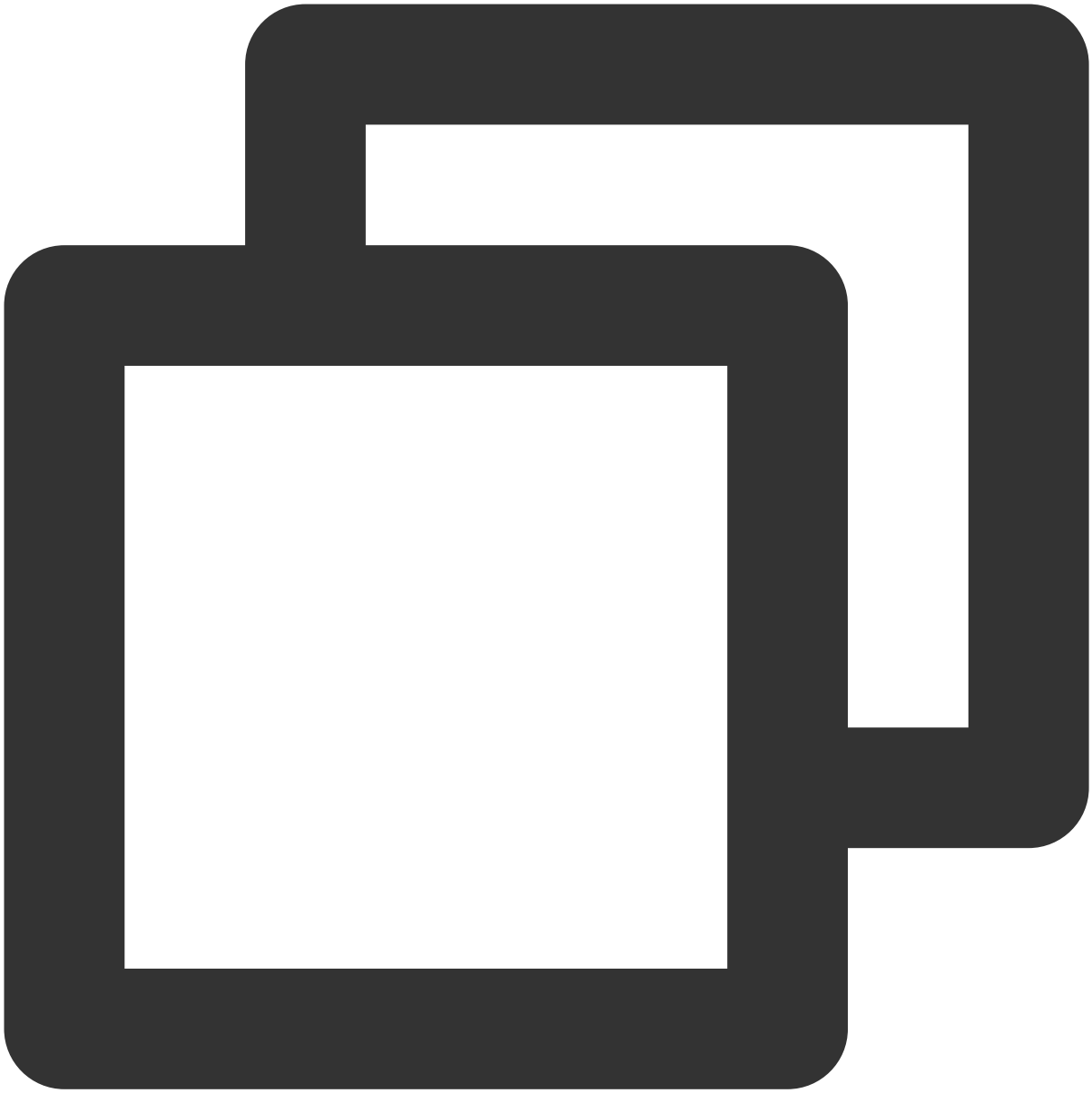


1. DB 管理员在目标数据库配置应用系统中，访问数据库所需的用户名和口令。
2. DB 管理员在 SSM 凭据管理系统中创建一个凭据对象，用来加密存储步骤1中获取的用户名和口令。
3. 应用系统需要访问数据库时，需要向 SSM 凭据管理系统请求访问凭据。
4. SSM 凭据管理系统获取到存储的凭据密文，解密后将凭据明文通过 HTTPS 返回给应用系统。

5. 应用系统读取并解析 SSM 凭据管理系统返回的凭据明文，从而获取用户名和口令，并可使用该账号访问目标数据库。
6. DB 管理员可为凭据创建多个版本内容，也可更新凭据版本内容，实现配置同步、版本管理、凭据轮换。

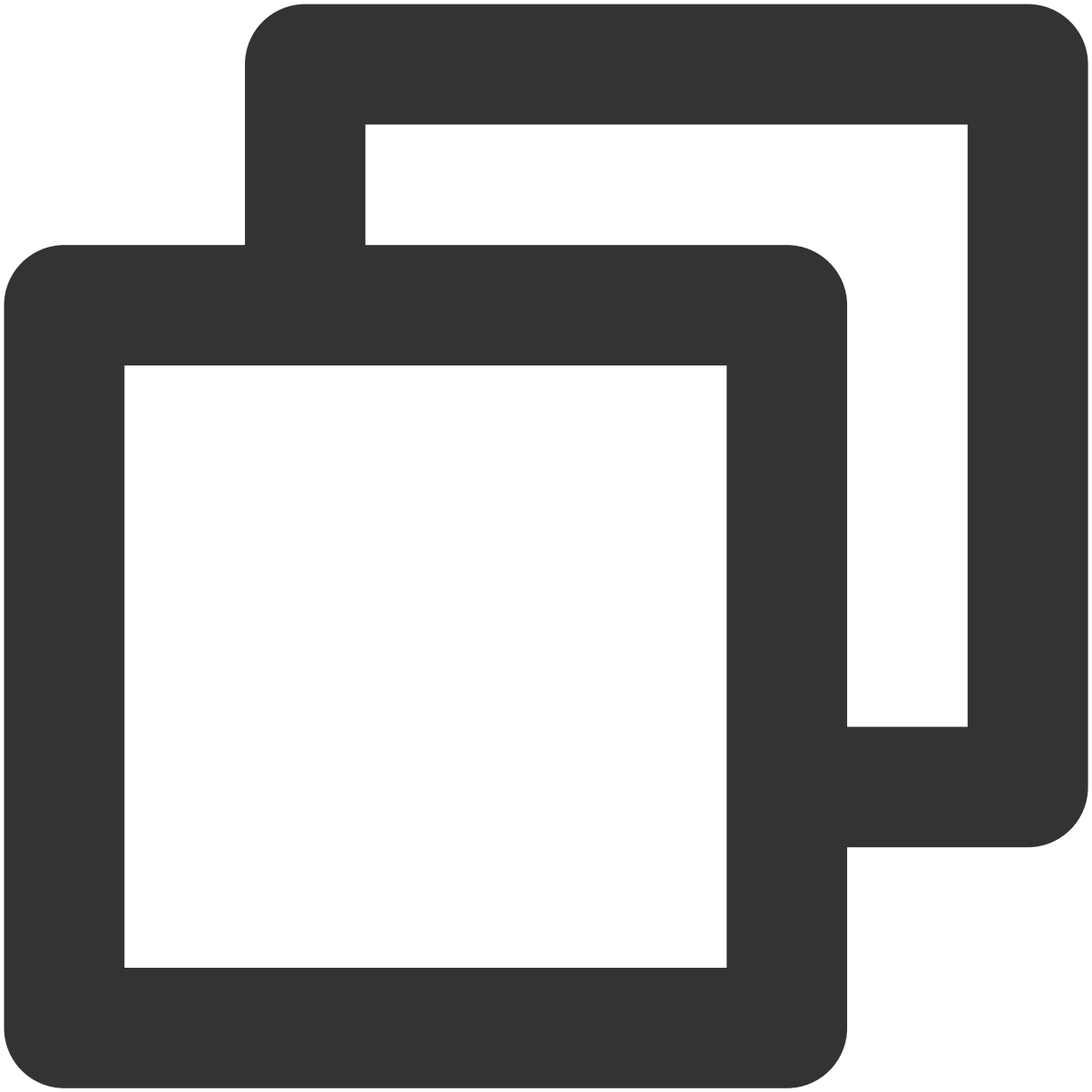
应用效果

对应用系统而言，通过调用 SSM 凭据管理系统的 API 或 SDK 来获取敏感的凭据明文，可避免在程序或配置中，明文编码凭据带来的信息泄露风险，调用对比如下：
使用本地存储数据库连接信息，连接信息明文保存在本地配置或者代码文件中，敏感凭据易泄露。
获取凭据明文示例代码：



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

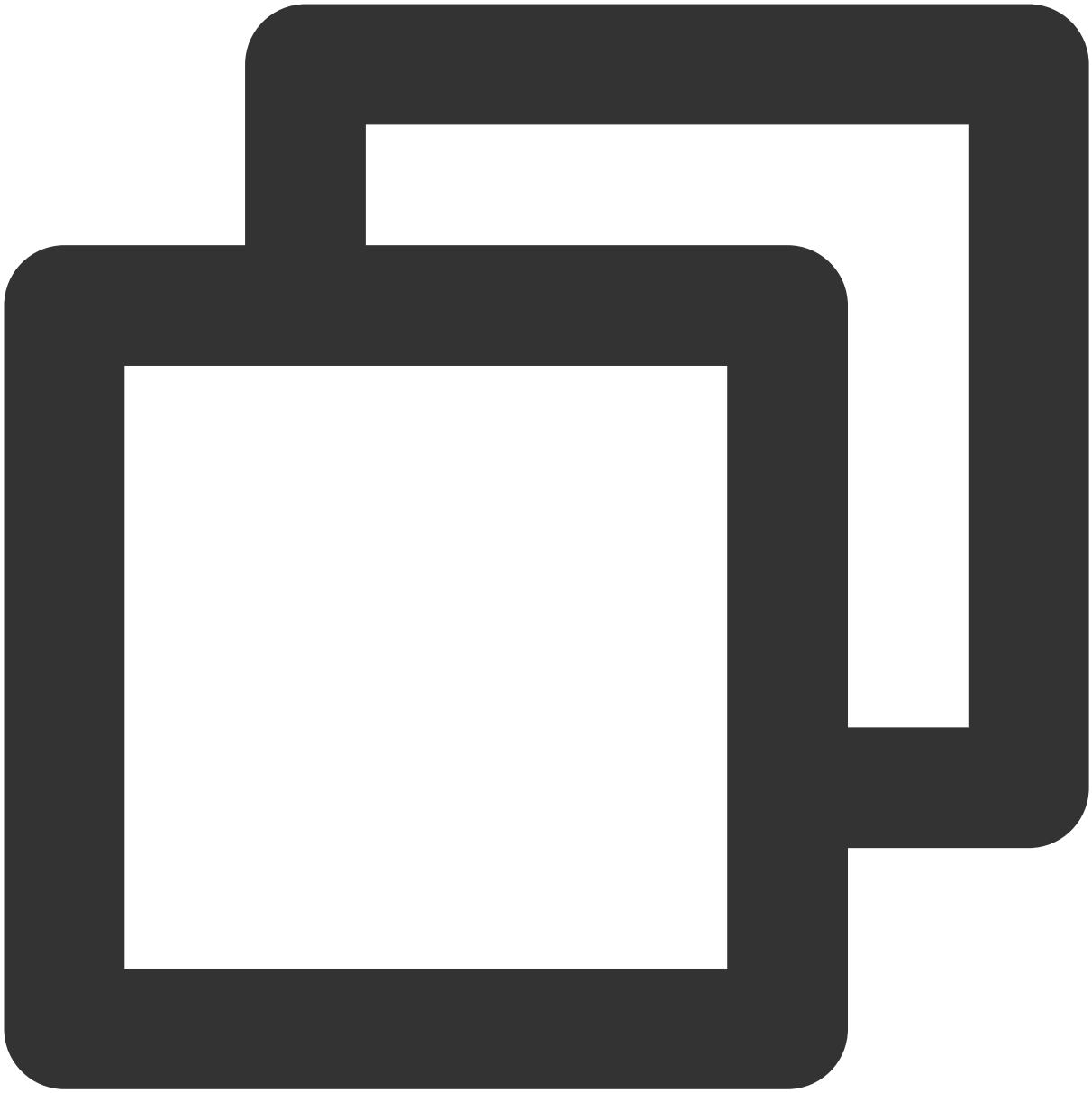
使用凭据明文示例代码：



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

使用 SSM 凭据管理系统连接数据库 DB 时，代码和本地配置中无需明文存储 DB 的连接信息。

获取凭据明文示例代码：



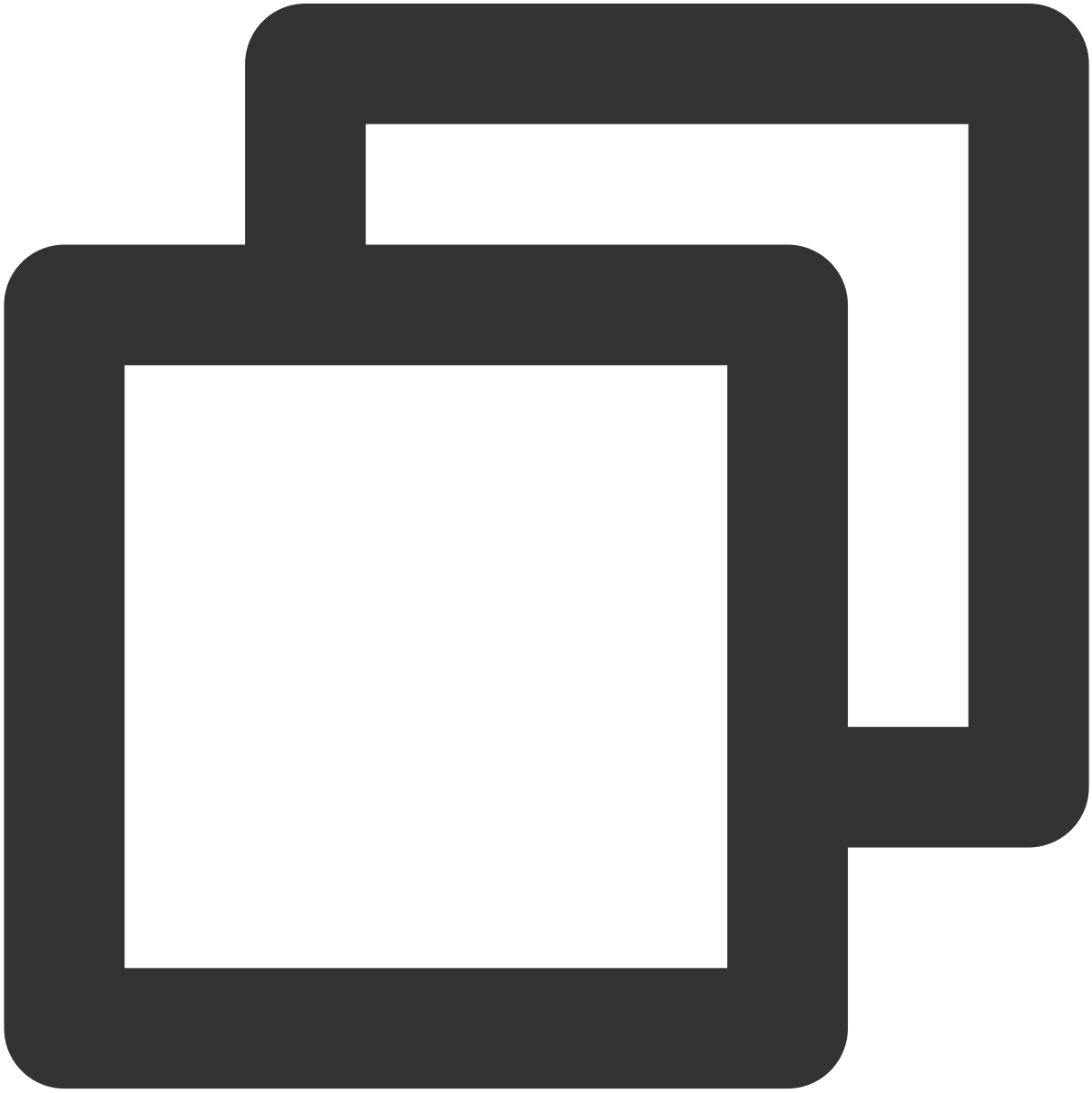
```
func GetDBConfig(secretName, version *string) string {
        credential := common.NewCredential(
                    secretId,
                    secretKey,
        )
        cpf := profile.NewClientProfile()
        cpf.HttpProfile.Endpoint = endpoint
        client, _ := ssm.NewClient(credential, 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
}
```

使用凭据明文示例代码：



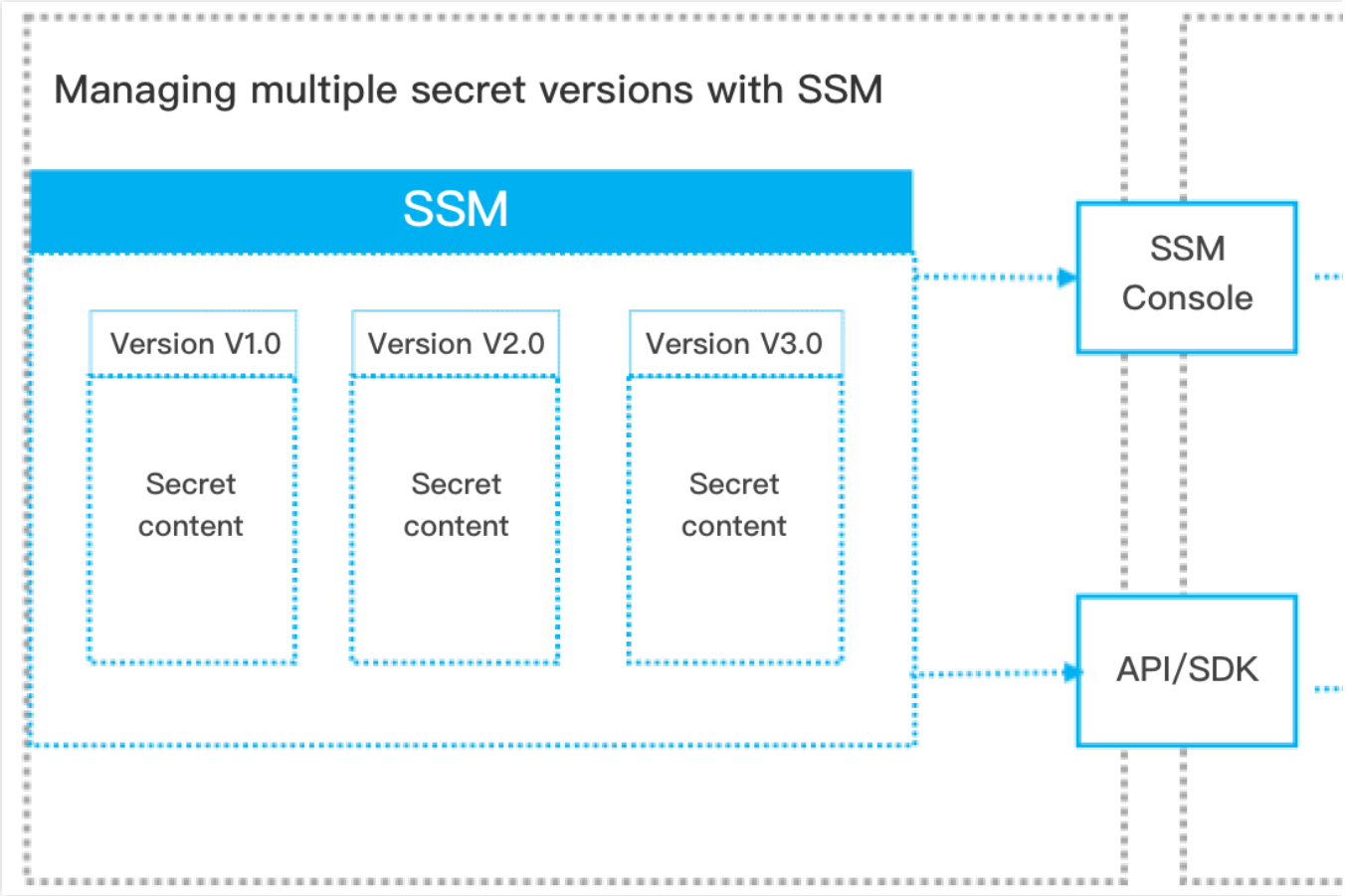
```
secretName := "MySecret1"
```

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

自定义凭据的轮换

最近更新时间：2024-01-02 15:13:40

为提升系统安全性，要求对目标凭据具备依赖性的应用配置同步更新。当多种应用系统在本地存储凭据内容时，在凭据更新时容易遗漏，从而带来应用中断风险。使用凭据管理系统，可以实现凭据管理系统内一处凭据更新，处处生效。此外，还可以为 凭据创建配置多个版本 ，实现凭据的灰度更新和轮换。



可以使用以下两种方式进行凭据轮换：

方式一：增加新的凭据版本，业务侧通过更新获取凭据的版本号实现灰度轮换。

方式二：直接修改当前使用凭据的内容，业务侧下次调用接口获取凭据时，会自动更新凭据内容，详情可以参见 凭据相关调用示例。