PowerShell Script Documentation: IT Industry Standard

This document provides an in-depth explanation of each function within the PowerShell script that interacts with HashiCorp Vault, MongoDB Atlas, and SQL Server. The script is designed to fetch API keys from Vault, retrieve MongoDB server and performance data, and insert the data into a SQL Server database efficiently. Each function in the script is documented below, highlighting its purpose, parameters, and process in detail.

# 1. Function: Get-VaultVariablesFromSQL

This function retrieves Vault configuration details stored in a SQL Server database. The details fetched include Vault address, namespace, path, and certificate information. These details are essential for establishing a secure connection to HashiCorp Vault.

## Parameters:

- sqlConnectionString (string): This is the connection string used to connect to the SQL Server instance.

## Process Flow:

1. A connection is established to the SQL Server using the provided connection string.  
2. The function executes a SQL query that fetches Vault-related details from the vault.cert\_table.  
3. The fetched details (Vault address, namespace, path, and certificate details) are stored in a hashtable.  
4. Error handling is included to capture any issues with SQL Server connections or queries.  
5. The SQL connection is closed once the process is complete, and the hashtable with Vault variables is returned.

# 2. Function: Insert-ServerDataIntoSqlServer

This function is responsible for inserting MongoDB server data into a SQL Server table. It uses the SqlBulkCopy class to insert large amounts of data efficiently, minimizing performance overhead.

## Parameters:

- sqlConnectionString (string): The connection string for the SQL Server instance where data will be inserted.  
- data (object): An object containing the server data that needs to be inserted into the SQL Server table.

## Process Flow:

1. The function establishes a connection to the SQL Server using the provided connection string.  
2. A DataTable is created to hold the server data. Columns such as hostname\_port, projectId, replicaSetName, nodeType, version, and clusterCreated are added to the DataTable.  
3. The server data from the input object is looped through, and each entry is added to the DataTable as a new row.  
4. The SqlBulkCopy class is used to perform the bulk insertion of the DataTable into the MongoDB.MongoServers\_STG table.  
5. Error handling captures any exceptions, and the SQL connection is closed after the process completes.

# 3. Function: ConvertAndInsertPerformanceData

This function is responsible for converting MongoDB Atlas performance metrics data (retrieved in JSON format) into a format that can be inserted into SQL Server using bulk copy operations.

## Parameters:

- data (psobject): The performance data retrieved from MongoDB Atlas in JSON format.  
- sqlServer (string): The name of the SQL Server where the data will be inserted.  
- database (string): The database name within SQL Server where the data will be stored.

## Process Flow:

1. A DataTable is created with columns such as groupId, hostId, processId, name, timestamp, value, and units.  
2. The function loops through the measurements property of the input data and extracts performance metrics (e.g., CPU usage, memory).  
3. Each performance data point is added to the DataTable with the relevant information.  
4. The SqlBulkCopy class is used to insert the populated DataTable into the MongoDB.PerformanceMetrics\_STG table in SQL Server.  
5. Error handling is included to ensure that any issues encountered during data insertion are properly logged.

# 4. Function: Get-ApiKeysFromVault

This function authenticates with HashiCorp Vault using a TLS certificate and retrieves API keys (public and private keys) from a specified Vault path.

## Parameters:

- vaultAddress (string): The address of the Vault server.  
- vaultNamespace (string): The namespace in Vault to use.  
- vaultPath (string): The path within Vault where the secrets (API keys) are stored.  
- certPath (string): The path to the TLS certificate used for authentication.  
- certPassword (string): The password for the TLS certificate.

## Process Flow:

1. The TLS certificate is imported using the certificate path and password provided.  
2. The function performs a REST API call to Vault to authenticate using the certificate and retrieve a client token.  
3. Using the client token, a second API call is made to fetch the secrets (API keys) from the specified Vault path.  
4. The function returns a list of API keys (both public and private) that are extracted from the Vault data.  
5. Error handling ensures that any authentication or data retrieval issues are captured and logged.

# 5. Function: Fetch-And-Insert-MongoDBServerData

This function retrieves server data for MongoDB Atlas from the Atlas API and inserts the retrieved information into a SQL Server table.

## Parameters:

- credential (PSCredential): The credentials used to authenticate with MongoDB Atlas (typically API keys).  
- groupId (string): The MongoDB Atlas project ID (group ID) for which server data needs to be fetched.  
- sqlConnectionString (string): The connection string for SQL Server where the data will be inserted.

## Process Flow:

1. The function calls the MongoDB Atlas API using the provided credentials and group ID to retrieve server details (hostname, replica set).  
2. The retrieved server data is passed to the Insert-ServerDataIntoSqlServer function to insert it into the SQL Server table.  
3. Error handling ensures that any API or database issues are logged.

# 6. Function: Get-ProjectIDs

This function fetches all project IDs (group IDs) for a given MongoDB Atlas organization. These project IDs are used to fetch server and performance data for each project.

## Parameters:

- credential (PSCredential): The credentials to authenticate with MongoDB Atlas.

## Process Flow:

1. A REST API call is made to MongoDB Atlas to retrieve all projects associated with the authenticated organization.  
2. The function extracts the project IDs from the API response and returns them as a list.  
3. Error handling ensures that any API issues are captured and logged.

# 7. Function: Get-ServerInfo

This function retrieves server information from a SQL Server table for a specific MongoDB project (group).

## Parameters:

- SQLConn (SqlConnection): The SQL connection object used to query the database.  
- groupId (string): The MongoDB project ID (group ID) for which server information is needed.

## Process Flow:

1. The function executes a SQL query to fetch server details (hostname, port) for the specified MongoDB project.  
2. The server details are returned as a list for further processing (e.g., retrieving performance data).  
3. The function ensures that the SQL connection is properly closed after execution.

# 8. Function: Invoke-MongoDBAtlasAPI

This function calls the MongoDB Atlas API to retrieve performance data for a specific server within a specified time range (e.g., the last 24 hours).

## Parameters:

- GroupId (string): The MongoDB project ID (group ID) for which performance data is needed.  
- SourceServer (string): The server for which performance data is being retrieved.  
- credential (PSCredential): The credentials to authenticate with MongoDB Atlas.

## Process Flow:

1. The function constructs a time range (usually the last 24 hours) and sends an API request to MongoDB Atlas.  
2. The API returns performance metrics data for the specified server, which is then returned by the function.  
3. Error handling is in place to capture any issues during the API call and data retrieval.

# 9. Function: Write-ErrorLog

This function logs error messages into a SQL Server table. It captures errors that occur during the execution of various functions within the script, ensuring that issues are properly documented for later analysis.

## Parameters:

- ErrMsg (string): The error message to be logged.  
- TargetServer (string): The server where the error occurred.  
- CollectorName (string): The name of the process or collector that generated the error.

## Process Flow:

1. The function constructs a SQL insert query that logs the error message, server name, and other relevant information into a table.  
2. The function executes the SQL query using a trusted connection to SQL Server.  
3. Error handling is also included to capture any issues that occur during the error logging process itself.