**Documentação Detalhada do Script de Processamento de Arquivos**

**Descrição**

Este script Python é utilizado para automatizar o processamento de arquivos ZIP dentro de um ambiente AWS. Ele verifica a integridade dos arquivos, evita duplicatas e move os arquivos processados entre diferentes buckets do S3, utilizando o DynamoDB para controle.

**Componentes do Script**

**Classe cp (Configuração de Parâmetros)**

Esta classe armazena as configurações estáticas necessárias para o funcionamento do script. Inclui caminhos de diretórios, informações de banco de dados e e-mail, e tipos de arquivo.

* FILES\_FOLDER: Nome do bucket S3 onde os arquivos ZIP são inicialmente armazenados.
* AIRLINES\_FOLDER: Lista dos caminhos nos quais o script irá operar.
* FAILED\_FOLDER: Nome do bucket S3 para onde os arquivos com falhas serão movidos.
* PROCESSING\_FOLDER: Nome do bucket S3 para onde os arquivos serão movidos para processamento.
* RAW\_FOLDER: Nome do bucket S3 onde os arquivos processados finais serão armazenados.
* TEMP\_FILES\_FOLDER: Caminho local para armazenamento temporário de arquivos.
* PROCESS\_ZIP\_FOLDER: Subdiretório dentro dos buckets onde os arquivos ZIP serão processados.
* ACCESS\_TYPE: Define se o acesso será por CSV ou por banco de dados.
* FILE\_HASH\_NAME e FILE\_LOG\_NAME: Nomes dos arquivos CSV para controle de hash e log.

**Classe zf (Funções Genéricas)**

Contém métodos para manipulação de datas, números aleatórios, hash de dados, conexões com o banco de dados, e funções relacionadas ao tratamento de arquivos ZIP.

* getDateTime(): Retorna a data e a hora atual formatada.
* getRandomNumber(): Gera um número aleatório com a quantidade de dígitos especificada.
* getHashFromData(): Calcula e retorna o hash SHA-512 de um conjunto de dados.
* getDBConnection(): Estabelece e retorna uma conexão com o serviço especificado da AWS.
* getDBRows(): Retorna o número de linhas encontradas no DynamoDB que correspondem a uma condição específica.
* getDBData(): Retorna os dados de uma consulta ao DynamoDB.
* query\_item(): Realiza uma consulta ao DynamoDB e retorna os itens encontrados.
* setDBRow(): Insere dados em uma tabela do DynamoDB.
* descompactar\_em\_outro\_bucket(): Descompacta um arquivo ZIP de um bucket S3 e carrega os arquivos descompactados em outro bucket.
* validar\_zip(): Valida a integridade de um arquivo ZIP.
* append\_to\_csv\_s3(): Anexa dados a um arquivo CSV armazenado em um bucket S3.
* search\_string\_in\_s3\_file(): Procura por uma string em um arquivo armazenado no S3 e retorna a linha em que foi encontrada.

**Funções de Processamento**

* verificar\_pasta\_de\_arquivos(): Verifica os arquivos em uma pasta específica e chama as funções apropriadas para processar arquivos ZIP ou outros tipos de arquivo.
* verificar\_arquivo(), verificar\_arquivo\_csv(), verificar\_arquivo\_zip(), verificar\_arquivo\_zip\_csv(): Estas funções verificam se um arquivo já foi processado com base em seu hash, validam sua integridade e, se apropriado, descompactam e movem para o próximo estágio de processamento ou para o bucket de falhas.

**Função Principal main()**

* Inicializa a execução do script, imprime mensagens de início e término no console e itera sobre os caminhos especificados para processar os arquivos.

**Execução do Script**

Para executar o script, assegure-se de que todas as configurações da classe cp estejam corretas e correspondam à estrutura do seu ambiente AWS. As funções da classe zf são chamadas conforme necessário durante o processamento dos arquivos.

O script pode ser executado diretamente a partir da linha de comando ou ser agendado como um job em um sistema de automação.

**Manutenção e Monitoramento**

Os logs de operações e falhas são essenciais para a manutenção e monitoramento do processo de transferência de arquivos. O script registra todas as operações importantes e as falhas nos arquivos CSV ou no DynamoDB, o que facilita o rastreamento e a esolução de problemas. Acompanhar esses registros é crucial para garantir a integridade e a confiabilidade do processo de transferência de arquivos.

**Segurança e Conformidade**

O script utiliza o hash SHA-512 para garantir a integridade dos dados, evitando assim a manipulação não autorizada de arquivos durante o processo de transferência. É essencial que as políticas de segurança da AWS, como o IAM (Identity and Access Management), estejam corretamente configuradas para permitir o acesso apenas aos usuários e serviços autorizados.

**Troubleshooting**

Em caso de falhas, o script irá registrar os erros correspondentes no arquivo de log ou na tabela DynamoDB, dependendo da configuração escolhida. Estes registros incluem informações detalhadas sobre o erro, facilitando a identificação e correção do problema.

**Customização**

Embora o script venha com configurações predefinidas, ele pode ser customizado para atender a diferentes necessidades e requisitos de negócios. As constantes e variáveis na classe cp podem ser modificadas para se adaptarem a diferentes estruturas de diretórios, políticas de nomenclatura e estratégias de backup.

**Considerações Finais**

Antes de colocar o script em produção, é recomendável realizar testes em um ambiente de desenvolvimento ou de teste para assegurar que o script funciona como esperado e que todas as interações com os serviços AWS estão corretas e seguras.

Certifique-se de que a documentação esteja acessível à equipe de operações ou a qualquer pessoa responsável pela execução e manutenção do script, para que possam referenciar e entender claramente o processo e a lógica implementada.

Nota: Este documento foi elaborado com base no código fornecido e contém informações que devem ser validadas e ajustadas conforme o ambiente específico em que o script será executado.