Atividade PBL - Projeto de Armazenamento Distribuído para uma Plataforma de Streaming

Problema:

Uma empresa de streaming de vídeos deseja armazenar e distribuir vídeos para usuários globalmente. O sistema de armazenamento deve ser altamente disponível, tolerante a falhas e capaz de escalar conforme a demanda cresce. Além disso, a empresa deseja manter uma latência baixa para garantir que os vídeos sejam transmitidos rapidamente.

Desafio:

Escolher um sistema de armazenamento distribuído adequado para a plataforma de streaming (ex.: S3, Ceph, Cassandra). Projetar a arquitetura de armazenamento, considerando replicação, consistência e escalabilidade. Implementar um protótipo simples que simule o upload e a distribuição de vídeos para múltiplos nós distribuídos.

Objetivo:

Aplicar os conceitos de armazenamento distribuído para projetar e implementar um sistema escalável e eficiente, garantindo disponibilidade e baixa latência.

Código do script de upload do arquivo local para o S3, e de geração do link com o CloudFront já configurado -

```
import boto3
import logging
from botocore.exceptions import ClientError
```

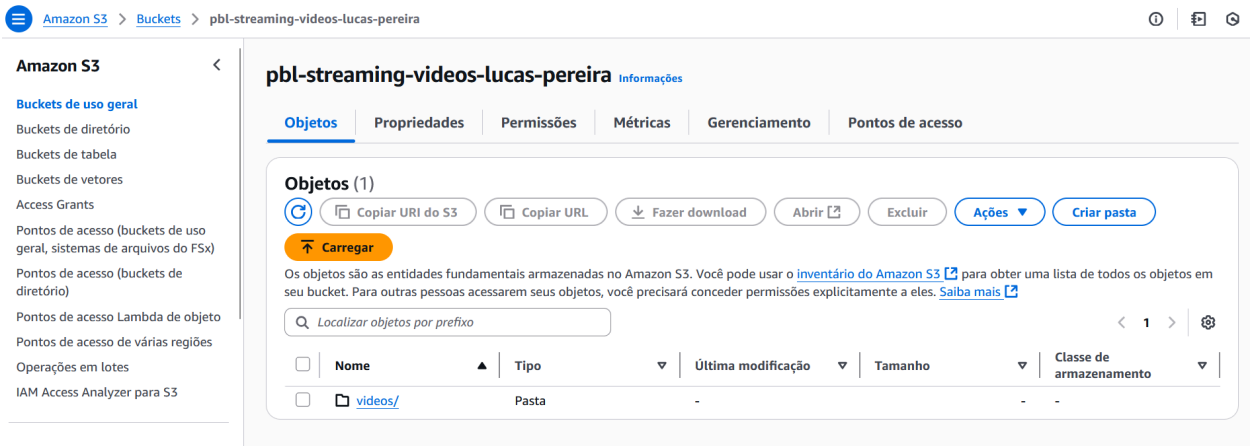
```
import os

# --- Configuração ---
AWS_REGION = "sa-east-1"  # Mude para a sua região, se necessário
BUCKET_NAME = "pbl-streaming-videos-lucas-pereira" # <<< COLOQUE O N
OME DO SEU BUCKET AQUI
PROFILE_NAME = "default" # Perfil de credenciais da AWS, se não for 'defaul
t', altere aqui.

# Configura o logging para ver informações detalhadas
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)
s - %(message)s')

# Cria uma sessão usando um perfil específico (opcional)
# session = boto3.Session(profile_name=PROFILE_NAME)
# s3_client = session.client("s3", region_name=AWS_REGION)

# Usa credenciais do ambiente padrão
s3_client = boto3.client("s3", region_name=AWS_REGION)
video_path = ""
def upload_video(file_path: str , object_name: str = None):
    """
    Faz o upload de um arquivo de vídeo para o S3.

    :param file_path: Caminho para o arquivo local.
    :param object_name: Nome do objeto no S3. Se não for especificado, usa o
nome do arquivo.
    :return: True se o upload for bem-sucedido, False caso contrário.
    """
    if object_name is None:
        object_name = os.path.basename(file_path)

    logging.info(f"Iniciando upload do arquivo '{file_path}' para o bucket '{BUC
KET_NAME}' como '{object_name}'...")

    try:
```

```
        # O método upload_file gerencia arquivos grandes de forma eficiente (m
ultipart upload)
        s3_client.upload_file(file_path, BUCKET_NAME, object_name)
        logging.info("Upload concluído com sucesso!")
    except FileNotFoundError:
        logging.error(f"O arquivo '{file_path}' não foi encontrado.")
        return False
    except ClientError as e:
        logging.error(f"Erro do cliente S3: {e}")
        return False
    return True

def get_distribution_url(object_name: str, expiration_seconds: int = 3600):
    """
    Gera uma URL pré-assinada que instrui o navegador a exibir o vídeo (inlin
e).
    Esta URL concede acesso temporário ao objeto.

    :param object_name: Nome do objeto no S3.
    :param expiration_seconds: Tempo em segundos até a URL expirar.
    :return: URL pré-assinada ou None em caso de erro.
    """
    logging.info(f"Gerando URL de visualização para '{object_name}' com valid
ade de {expiration_seconds} segundos...")

    try:
        url = s3_client.generate_presigned_url(
            'get_object',
            Params={
                'Bucket': BUCKET_NAME,
                'Key': object_name,
                'ResponseContentDisposition': 'inline'  # ← ESTA É A LINHA QUE FA
Z A MÁGICA
            },
            ExpiresIn=expiration_seconds
        )
```

```
        logging.info(f"URL gerada com sucesso!")
        return url
    except ClientError as e:
        logging.error(f"Erro ao gerar a URL: {e}")
        return None

if __name__ == "__main__":
    # --- Simulação ---

    # 1. Crie um arquivo de vídeo de teste, ex: 'meu_video_teste.mp4'
    # Coloque este arquivo na mesma pasta do script ou especifique o caminh
o completo.
    video_file_to_upload = 'pastor-taser.mp4'
    s3_object_key = 'videos/yt-shorts.mp4'

    # Criar um arquivo de teste se não existir
    if not os.path.exists(video_file_to_upload):
        logging.warning(f"Arquivo '{video_file_to_upload}' não encontrado. Crian
do um arquivo de teste.")
        with open(video_file_to_upload, 'wb') as f:
            f.write(os.urandom(1024 * 1024)) # Cria um arquivo de 1MB

    # 2. Simular o upload do vídeo
    if upload_video(video_file_to_upload, s3_object_key):

        # 3. Simular a obtenção da URL para distribuição (o que o backend faria
para o cliente)
        # Em um sistema real, esta URL seria retornada pela API para o player de
vídeo.
        distribution_link = get_distribution_url(s3_object_key)

        if distribution_link:
            print("\n--- Simulação de Distribuição ---")
            print("O backend entregaria esta URL para o player do usuário:")
            print(f"\n{distribution_link}\n")
            print("Copie e cole esta URL no seu navegador para baixar o vídeo.")
```

```
            print("(A URL é válida por 1 hora)")


# ... (início do script com a função de upload) ...

CLOUDFRONT_DOMAIN = "dkrygojknq0nx.cloudfront.net" # <<< COLOQUE S
EU DOMÍNIO DO CLOUDFRONT AQUI

def get_cdn_url(object_name: str):
    """
    Constrói a URL pública e permanente do vídeo através da CDN CloudFront.
    """
    return f"https://{CLOUDFRONT_DOMAIN}/{object_name}"

if __name__ == "__main__":
    # ... (código de upload) ...

    # 2. Simular o upload do vídeo
    if upload_video(video_file_to_upload, s3_object_key):

        # 3. Obter a URL de distribuição da CDN
        # Esta é a URL que seu backend entregaria para o aplicativo cliente.
        cdn_link = get_cdn_url(s3_object_key)

        if cdn_link:
            print("\n--- Distribuição via CDN CloudFront ---")
            print("A URL do vídeo para o player é:")
            print(f"\n{cdn_link}\n")
            print("Esta URL é pública e otimizada para entrega global.")
```

AWS S3(armazenamento distribuído)-



AWS CloudFront (disponibilidade e redução de latência)-