```
import os
import zlib
import boto3
import logging
import logging.handlers
from splunk_handler import SplunkHandler
from splunk_handler import force_flush
import time


def lambda_handler(event, context):
    # Lets get started
    print('Loading function')
    print("Log stream name:", context.log_stream_name)

    # Read in the environment and set the splunk logger variables
    host = os.environ['host']
    token = os.environ['token']
    port = os.environ['port']
    index = os.environ['index']
    sourcetype = os.environ['sourcetype']
    verify = os.environ['verify']
    debug = os.environ['debug']

    if debug.lower() == 'true':
        debug = True
    else:
        debug = False

    if verify.lower() == 'true':
        verify = True
    else:
        verify = False

    retry_count = os.environ['retry_count']
    source = os.environ['source']
    retry_backoff = os.environ['retry_backoff']
    timeout = os.environ['timeout']
    flush_interval = os.environ['flush_interval']
    queue_size = os.environ['queue_size']

    # Instantiate the splunk logger
    splunk = SplunkHandler(host=host, port=port, token=token,
index=index, sourcetype=sourcetype, debug=debug,
                           verify=verify,
retry_count=int(retry_count), source=source,
                           retry_backoff=float(retry_backoff),
                           timeout=int(timeout),
flush_interval=float(flush_interval), queue_size=int(queue_size))
```

```
    # Create the custom logger
    mylog = logging.getLogger("MyLogger")
    mylog.setLevel(logging.INFO)
    mylog.addHandler(splunk)

    # Get the information that was passed in from S3 PUT trigger
    for record in event['Records']:
        print(type(record))
        key = record['s3']['object']['key']
        bucket = record['s3']['bucket']['name']

    # We got the file and bucket info, lets go get the object
    print('Go try to get the file %s in bucket %s' % (key, bucket))
    s3_client = boto3.client('s3')

    # Get the object instead of downloading the file
    obj = s3_client.get_object(Bucket=bucket, Key=key)

    # Read in the Body element data, this is the raw file data; S3
Object is a JSON object
    file_content = obj['Body'].read().decode("latin-1")

    # Split the contents of the file, header info from SIEM data
    file_split_date = file_content.split("|==|\n")
[1].encode('latin-1')

    # Decompress the SIEM data
    uncompressed_file_content =
zlib.decompressobj().decompress(file_split_date)

    # Log the data for debugging
    if debug:
        print(uncompressed_file_content.decode())

    # Send the data to Splunk via HTTP Event Collector
    print("Let's try to send this data to %s." % host)
    for msg in uncompressed_file_content.decode().splitlines():
        if msg != '':
            mylog.info(msg)
            print("Queue: %s" % msg)
            print("Queue size = %s" % splunk.queue.qsize())
            if splunk.queue.full():
                # Force the flush
                print("Flush the splunk logger!!!")
                splunk.force_flush()

    # Force the flush
    print("Final flush the splunk logger!!!")
    splunk.force_flush()
    print("Final Queue size = %s" % splunk.queue.qsize())
```

```
    print("Finished")
    # Put the newly decompressed log in a new bucket for archiving
    print("Put the newly decompressed log in a new bucket for
archiving.")
    s3_client.put_object(Bucket='logs-decompressed', Key=key,
Body=uncompressed_file_content)
    print("All done!!")
```