from datetime import datetime, timedelta

import requests

from chronicle import ingest

from chronicle.utils import get\_env\_var

# Fetch logs from API, process it and ingest

def get\_and\_ingest\_audit\_logs():

# Calc start time, end time will be 'now'

min\_time = datetime.utcnow() - timedelta(minutes=int(FUNCTION\_MINUTE\_INTERVAL))

min\_time\_str = min\_time.strftime("%Y-%m-%dT%H:%M:%SZ")

min\_time = int((min\_time - datetime(1970, 1, 1)).total\_seconds())

print(

"Retrieving "

+ str(FUNCTION\_MINUTE\_INTERVAL)

+ " mins of logs since: "

+ min\_time\_str

)

print("Processing logs...")

url = f"https://api.slack.com/audit/v1/logs?oldest={min\_time}"

headers = {

"Accept": "application/json",

"Content-Type": "application/json",

"Authorization": f"Bearer {SLACK\_ADMIN\_TOKEN}",

}

# Iterate through all the pages if pagination

# available and add data to data\_list

while True:

data\_list = []

if DEBUG:

print(f"Processing set of results with mintime: {min\_time}")

r = requests.get(url=url, headers=headers)

r.raise\_for\_status()

try:

r = r.json()

except (TypeError, ValueError) as e:

print(

"ERROR: Unexpected data format received while collecting \

audit logs"

)

raise(e)

log\_count = len(r.get("entries", []))

if DEBUG:

print(f"Retrieved {log\_count} audit logs from the API call")

# No need to ingest logs for empty response

if log\_count == 0:

break

data\_list.extend(iter(r["entries"]))

# Ingest list of data to chronicle

ingest(data\_list, CHRONICLE\_DATA\_TYPE)

# update the url if next cursor is available

if r["response\_metadata"]["next\_cursor"]:

url = "https://api.slack.com/audit/v1/logs?oldest={} \

&cursor={}".format(

min\_time, r["response\_metadata"]["next\_cursor"]

)

print('More records expected so re-running! (processed '+str(log\_count)+' records)')

else:

print("Logs processed successfully.")

break

def main():

global CHRONICLE\_DATA\_TYPE, DEBUG

global SLACK\_ADMIN\_TOKEN, FUNCTION\_MINUTE\_INTERVAL

CHRONICLE\_DATA\_TYPE = "SLACK\_AUDIT"

FUNCTION\_MINUTE\_INTERVAL = get\_env\_var("FUNCTION\_MINUTE\_INTERVAL", 60)

# Slack admin token

SLACK\_ADMIN\_TOKEN = get\_env\_var("SLACK\_ADMIN\_TOKEN")

# Debug flag. GREATLY increases the verbosity of logging

DEBUG = get\_env\_var("DEBUG", default=False)

# Method to fetch audit logs and ingest to chronicle

get\_and\_ingest\_audit\_logs()

return "OK"