```
import os
import json
import logging
import io
from email.message import EmailMessage
import smtplib
from datetime import datetime
import pytz
import oci

# --- Configuration: All loaded from env vars! ---
SMTP_SERVER = os.environ.get("SMTP_SERVER")
SMTP_PORT = int(os.environ.get("SMTP_PORT", 587))
OCI_SMTP_USER = os.environ.get("OCI_SMTP_USER")
OCI_SMTP_PW = os.environ.get("OCI_SMTP_PW")
EMAIL_FROM = os.environ.get("EMAIL_FROM")
EMAIL_TO = os.environ.get("EMAIL_TO")

# --- In-memory region-to-timezone mapping ---
REGION_TZ_MAP = {
    "ap-sydney-1": "Australia/Sydney",
    "ap-melbourne-1": "Australia/Melbourne",
    "sa-saopaulo-1": "America/Sao_Paulo",
    "sa-vinhedo-1": "America/Sao_Paulo",
    "ca-montreal-1": "America/Toronto",
    "ca-toronto-1": "America/Toronto",
    "sa-santiago-1": "America/Santiago",
    "sa-valparaiso-1": "America/Santiago",
    "sa-bogota-1": "America/Bogota",
    "eu-paris-1": "Europe/Paris",
    "eu-marseille-1": "Europe/Paris",
    "eu-frankfurt-1": "Europe/Berlin",
    "ap-hyderabad-1": "Asia/Kolkata",
    "ap-mumbai-1": "Asia/Kolkata",
    "ap-batam-1": "Asia/Jakarta",
    "il-jerusalem-1": "Asia/Jerusalem",
    "eu-milan-1": "Europe/Rome",
    "eu-turin-1": "Europe/Rome",
    "ap-osaka-1": "Asia/Tokyo",
```

```
    "ap-tokyo-1": "Asia/Tokyo",
    "mx-queretaro-1": "America/Mexico_City",
    "mx-monterrey-1": "America/Monterrey",
    "eu-amsterdam-1": "Europe/Amsterdam",
    "me-riyadh-1": "Asia/Riyadh",
    "me-jeddah-1": "Asia/Riyadh",
    "eu-jovanovac-1": "Europe/Belgrade",
    "ap-singapore-1": "Asia/Singapore",
    "ap-singapore-2": "Asia/Singapore",
    "af-johannesburg-1": "Africa/Johannesburg",
    "ap-seoul-1": "Asia/Seoul",
    "ap-chuncheon-1": "Asia/Seoul",
    "eu-madrid-1": "Europe/Madrid",
    "eu-madrid-3": "Europe/Madrid",
    "eu-stockholm-1": "Europe/Stockholm",
    "eu-zurich-1": "Europe/Zurich",
    "me-abudhabi-1": "Asia/Dubai",
    "me-dubai-1": "Asia/Dubai",
    "uk-london-1": "Europe/London",
    "uk-cardiff-1": "Europe/London",
    "us-ashburn-1": "America/New_York",
    "us-chicago-1": "America/Chicago",
    "us-phoenix-1": "America/Phoenix",
    "us-sanjose-1": "America/Los_Angeles",
    "default": "UTC"
}

def resolve_region():
    """Get region from OCI Resource Principal Signer,
falling back to env variable, then default"""
    try:
        signer =
oci.auth.signers.get_resource_principals_signer()
        region = signer.region
        logging.info(f"Region resolved from resource
principal: {region}")
        return region
    except Exception as e:
        region = os.environ.get('OCI_REGION', 'default')
```

```
        logging.info(f"Resource principal not available,
region from env: {region}")
        return region

region = resolve_region()
target_tz = REGION_TZ_MAP.get(region,
REGION_TZ_MAP["default"])

def convert_utc_to_local(utc_str, target_tz):
    try:
        dt_str = utc_str.rstrip("Z")
        try:
            utc_dt = datetime.strptime(dt_str, "%Y-%m-
%dT%H:%M:%S.%f")
        except ValueError:
            utc_dt = datetime.strptime(dt_str, "%Y-%m-
%dT%H:%M:%S")
        utc_dt = utc_dt.replace(tzinfo=pytz.UTC)
        local_dt =
utc_dt.astimezone(pytz.timezone(target_tz))
        return local_dt.strftime("%Y-%m-%d %I:%M:%S %p %Z")
    except Exception as e:
        logging.error(f"Failed to convert '{utc_str}' to
{target_tz}: {e}")
        return utc_str or "UNKNOWN TIME"

def build_email_body(payload):
    event_time_utc = payload.get('eventTime', 'UNKNOWN
TIME')
    event_time_local = convert_utc_to_local(event_time_utc,
target_tz)
    event_type = payload.get('eventType', 'UNKNOWN EVENT')
    data_section = payload.get('data', {})
    resource_name = data_section.get('resourceName', 'N/A')
    compartment_id = data_section.get('compartmentId',
'N/A')
    add_details = data_section.get('additionalDetails', {})
    request_type = add_details.get('requestType', 'N/A')
    status = add_details.get('status', 'N/A')
```

```
    email_lines = [
        "Oracle Event Notification",
        "-------------------------",
        f"Event Time ({target_tz}): {event_time_local}",
        f"Event Type: {event_type}",
        f"Instance Name: {resource_name}",
        f"Compartment OCID: {compartment_id}",
        f"Request Type: {request_type}",
        f"Status: {status}"
    ]
    return "\n".join(email_lines)

def send_email(subject, body, email_from, email_to):
    msg = EmailMessage()
    msg["Subject"] = subject
    msg["From"] = email_from
    msg["To"] = email_to
    msg.set_content(body)
    try:
        with smtplib.SMTP(SMTP_SERVER, SMTP_PORT) as server:
            server.starttls()
            server.login(OCI_SMTP_USER, OCI_SMTP_PW)
            server.send_message(msg)
        logging.info(f"Email sent to {email_to}")
    except Exception as e:
        logging.error("Failed to send email.",
exc_info=True)

def handler(ctx, data: io.BytesIO = None):
    logging.info("Handler invoked.")
    try:
        payload = json.loads(data.getvalue())
        event_type = payload.get('eventType', '')
        data_section = payload.get('data', {})
        add_details = data_section.get('additionalDetails',
{})
        request_type = add_details.get('requestType', '')
        status = add_details.get('status', '')
        resource_name = data_section.get('resourceName', '')
```

```
        refresh_type = add_details.get('refreshType', '')
        source_type = add_details.get('sourceType', '')
        event_time_utc = payload.get("eventTime", "")

        if event_type ==
'com.oraclecloud.analyticswarehouse.pipeline.datarefresh.est
imate':
            estimatedCompletionTime =
add_details.get("estimatedCompletionTime", "")
            est_local =
convert_utc_to_local(estimatedCompletionTime, target_tz)
            subject = f"Oracle FAIDP event for
{resource_name}: {request_type} estimated to complete at
{est_local}"
        elif event_type ==
'com.oraclecloud.analyticswarehouse.pipeline.datarefresh.com
plete':
            completionTime =
add_details.get("completionTime", "")
            comp_local =
convert_utc_to_local(completionTime, target_tz)
            subject = f"Oracle FAIDP event for
{resource_name}: {request_type} {status} at {comp_local}"
        else:
            evt_local = convert_utc_to_local(event_time_utc,
target_tz)
            subject = f"Oracle FAIDP event for
{resource_name}: {request_type} {status} at {evt_local}"

        email_body = build_email_body(payload)
        send_email(subject, email_body, EMAIL_FROM,
EMAIL_TO)
        return "Event processed and email sent."
    except Exception:
        logging.error("Error processing event or sending
email.", exc_info=True)
        return "Failed to process payload."
    raise
```