import os

import itertools

import subprocess

import math

SERVICE\_NAME = 'uua-client-event'

CPU\_NUM = 3

HEAP\_SIZE = 3 \* GB

RAM\_SIZE = HEAP\_SIZE + 1 \* GB

# We make disk size larger than HEAP so that if we ever need to do a heap dump, it will fit on disk.

DISK\_SIZE = HEAP\_SIZE + 2 \* GB

class Profile(Struct):

package = Default(String, SERVICE\_NAME)

cmdline\_flags = Default(String, '')

log\_level = Default(String, 'INFO')

instances = Default(Integer, 1000)

kafka\_bootstrap\_servers = Default(String, '/s/kafka/client-events:kafka-tls')

kafka\_bootstrap\_servers\_remote\_dest = Default(String, '/s/kafka/bluebird-1:kafka-tls')

source\_topic = Default(String, 'client\_event')

sink\_topics = Default(String, 'unified\_user\_actions,unified\_user\_actions\_engagements')

decider\_overlay = Default(String, '')

resources = Resources(

cpu = CPU\_NUM,

ram = RAM\_SIZE,

disk = DISK\_SIZE

)

install = Packer.install(

name = '{{profile.package}}',

version = Workflows.package\_version()

)

async\_profiler\_install = Packer.install(

name = 'async-profiler',

role = 'csl-perf',

version = 'latest'

)

setup\_jaas\_config = Process(

name = 'setup\_jaas\_config',

cmdline = '''

mkdir -p jaas\_config

echo "KafkaClient {

com.sun.security.auth.module.Krb5LoginModule required

principal=\\"discode@TWITTER.BIZ\\"

useKeyTab=true

storeKey=true

keyTab=\\"/var/lib/tss/keys/fluffy/keytabs/client/discode.keytab\\"

doNotPrompt=true;

};" >> jaas\_config/jaas.conf

'''

)

main = JVMProcess(

name = SERVICE\_NAME,

jvm = Java11(

heap = HEAP\_SIZE,

extra\_jvm\_flags =

'-Djava.net.preferIPv4Stack=true'

' -XX:MaxMetaspaceSize=536870912'

' -XX:+UseNUMA'

' -XX:+AggressiveOpts'

' -XX:+PerfDisableSharedMem' # http://www.evanjones.ca/jvm-mmap-pause.html

' -Dlog\_level={{profile.log\_level}}'

' -Dlog.access.output=access.log'

' -Dlog.service.output={{name}}.log'

' -Djava.security.auth.login.config=jaas\_config/jaas.conf'

),

arguments =

'-jar {{name}}-bin.jar'

' -admin.port=:{{thermos.ports[health]}}'

' -kafka.bootstrap.servers={{profile.kafka\_bootstrap\_servers}}'

' -kafka.bootstrap.servers.remote.dest={{profile.kafka\_bootstrap\_servers\_remote\_dest}}'

' -kafka.group.id={{name}}-{{environment}}'

' -kafka.producer.client.id={{name}}-{{environment}}'

' -kafka.max.pending.requests=10000'

# CE events is about 0.4-0.6kb per message on the consumer side. A fetch size of 6~18 MB get us

# about 10k ~ 20k of messages per batch. This fits the size of our pending requests queue and

# within the limit of the max poll records.

' -kafka.consumer.fetch.max=9.megabytes'

' -kafka.consumer.fetch.min=3.megabytes'

' -kafka.max.poll.records=40000'

' -kafka.commit.interval=20.seconds'

' -kafka.producer.batch.size=4.megabytes'

' -kafka.producer.buffer.mem=64.megabytes'

' -kafka.producer.linger=100.millisecond'

' -kafka.producer.request.timeout=30.seconds'

' -kafka.producer.compression.type=lz4'

' -kafka.worker.threads=4'

' -kafka.source.topic={{profile.source\_topic}}'

' -kafka.sink.topics={{profile.sink\_topics}}'

' -decider.base=decider.yml'

' -decider.overlay={{profile.decider\_overlay}}'

' -cluster={{cluster}}'

' {{profile.cmdline\_flags}}',

resources = resources

)

stats = Stats(

library = 'metrics',

port = 'admin'

)

job\_template = Service(

name = SERVICE\_NAME,

role = 'discode',

instances = '{{profile.instances}}',

contact = 'disco-data-eng@twitter.com',

constraints = {'rack': 'limit:1', 'host': 'limit:1'},

announce = Announcer(

primary\_port = 'health',

portmap = {'aurora': 'health', 'admin': 'health'}

),

task = Task(

resources = resources,

name = SERVICE\_NAME,

processes = [async\_profiler\_install, install, setup\_jaas\_config, main, stats],

constraints = order(async\_profiler\_install, install, setup\_jaas\_config, main)

),

health\_check\_config = HealthCheckConfig(

initial\_interval\_secs = 100,

interval\_secs = 60,

timeout\_secs = 60,

max\_consecutive\_failures = 4

),

update\_config = UpdateConfig(

batch\_size = 1000,

watch\_secs = 90,

max\_per\_shard\_failures = 3,

max\_total\_failures = 0,

rollback\_on\_failure = False

)

)

PRODUCTION = Profile(

# go/uua-decider

decider\_overlay = '/usr/local/config/overlays/discode-default/UnifiedUserActions/prod/{{cluster}}/decider\_overlay.yml'

)

STAGING = Profile(

package = SERVICE\_NAME+'-staging',

cmdline\_flags = '',

kafka\_bootstrap\_servers\_remote\_dest = '/s/kafka/custdevel:kafka-tls',

decider\_overlay = '/usr/local/config/overlays/discode-default/UnifiedUserActions/staging/{{cluster}}/decider\_overlay.yml' # go/uua-decider

)

DEVEL = STAGING(

log\_level = 'INFO',

)

prod\_job = job\_template(

tier = 'preferred',

environment = 'prod',

).bind(profile = PRODUCTION)

staging\_job = job\_template(

environment = 'staging'

).bind(profile = STAGING)

devel\_job = job\_template(

environment = 'devel'

).bind(profile = DEVEL)

jobs = []

for cluster in ['atla', 'pdxa']:

jobs.append(prod\_job(cluster = cluster))

jobs.append(staging\_job(cluster = cluster))

jobs.append(devel\_job(cluster = cluster))