环境部署

环境部署分为以下几个部分

- 存储依赖部署
- 埋点采集环境部署
- 主程序部署

存储依赖部署

启动存储服务

CUGE 的存储依赖有 MySQL, Redis和ETCD。您可以参考以下的docker compose文件写法来部署。需要注意的是，以下配置文件意在 展示CUGE对数据存储实例的运行参数要求，通过此配置文件启动的存储服务，都运行在单机模式，不适合生产环境使用。生产环境请联系运维为其分配有主从节点互备的实例。

注意修改密码以及挂载点的物理路径。

```
version: "3.1"

services:
  db:
    image: mysql:8.0.22
    command: --default-authentication-plugin=mysql_native_password
    restart: always
    container_name: mysql
    ports:
      - "13306:3306"
    environment:
      MYSQL_ROOT_PASSWORD: mysql-root-password
    volumes:
      - /your/path/to/physical/volume:/var/lib/mysql

  redis:
    image: redis:6.0.9
    command: redis-server --appendonly yes --requirepass redis-root-password
    restart: always
    container_name: redis
    ports:
      - "16379:6379"
    volumes:
      - /your/path/to/physical/volume:/data

  etcd:
    image: quay.io/coreos/etcd:latest
    ports:
      - "2379:2379"
      - "2380:2380"
    container_name: etcd
    restart: always
    volumes:
      - /your/path/to/physical/volume:/etcd-data
```

```
    command: /usr/local/bin/etcd --data-dir=/etcd-data --name node1  --initial-
advertise-peer-urls http://实例机器IP:2380 --listen-peer-urls http://0.0.0.0:2380
--advertise-client-urls http://实例机器IP:2379 --listen-client-urls
http://0.0.0.0:2379 --initial-cluster node1=http://实例机器IP:2380

networks:
  default:
    external:
      name: cuge-network
```

对于ETCD，您还需要进入etcdctl V3 API，添加一个账户并打开鉴权。有关更多的信息，请参考 [etcd docs](#)。

埋点采集环境部署

CUGE 埋点采集环境使用了FileBeats，Logstash及Elasticsearch三件套，埋点可视化环境使用了Kibana。您可以参考以下配置文件启动Elasticsearch及Kibana。

```
version: "3.1"

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch:7.9.3
    restart: always
    container_name: es
    environment:
      - discovery.type=single-node
    ports:
      - "9200:9200"
      - "9300:9300"
    volumes:
      - /path/to/elasticsearch/data:/usr/share/elasticsearch/data

  kibana:
    image: docker.elastic.co/kibana/kibana:7.9.3
    restart: always
    container_name: kibana
    depends_on:
      - elasticsearch
    ports:
      - "5601:5601"

networks:
  default:
    external:
      name: cuge-network
```

接下来是启动FileBeats及Logstash。这两个容器，由于需要去watch CUGE产生的日志文件，因此配置较多，需要额外注意。

```
version: "3.1"

services:
  filebeat:
    image: docker.elastic.co/beats/filebeat:7.10.2
    command: filebeat -e -strict.perms=false
```

```
    restart: always
    container_name: filebeat
    depends_on:
      - logstash
    volumes:
      - /path/to/logstash/filebeats.yml:/usr/share/filebeat/filebeat.yml:ro # 把
filebeats.yml文件映射进入容器。
      - /path/to/cuge/:/root/cuge # 把CUGE的日志产生目录从物理机映射进filebeats容器。
      - /var/lib/docker/containers:/var/lib/docker/containers:ro
      - /var/run/docker.sock:/var/run/docker.sock:ro

  logstash:
    image: logstash:7.11.2
    command: logstash -f /root/config/cuge-trackparam-pipeline.conf --
config.reload.automatic
    restart: always
    container_name: logstash
    volumes:
      - /path/to/logstash/config:/root/config # 把logstash的配置文件映射进容器，以便启
动参数指定配置文件。

networks:
  default:
    external:
      name: cuge-network
```

> 上面给出的原版`logstash`内部使用的是CMS GC，且堆设置的很小(768mb)，这会导致当埋点数量大的时候`logstash`频繁GC，严重影响性能。建议有需要的用户进去手动修改一下`logstash`的GC引擎及堆大小。推荐设置GC引擎为G1GC，且堆大小给到1.5GB。

FileBeats的配置可以参考

```
filebeat.config:
  modules:
    path: ${path.config}/modules.d/*.yml
    reload.enabled: false

filebeat.inputs:
- type: log
  paths:
    - /root/cuge/logs/UserGrowingEngine/trackparam.log

output.logstash:
  hosts: ["logstash:5044"]
```

而`cuge-trackparam-pipeline.conf`的配置请务必使用以下配置，除非您非常清楚您在做什么，否则**不要修改!**

```
input {
    beats {
        port => "5044"
    }
}

filter {
    grok {
```

```
        match => [
            "message", "\|%{NUMBER:timestamp}\|###\|%{NUMBER:order}\|###\|(?
<deviceId>[a-zA-Z0-9._-]+)\|###\|%{NUMBER:userId}\|###\|(?<bizLine>[a-zA-Z0-
9._-]+)\|###\|%{NUMBER:appId}\|###\|(?<sdkVersion>v[0-9]+\.[0-9]+\.
[0.9]+)\|###\|%{NUMBER:eventId}\|###\|tp\.(?<param0>[A-Za-z0-9_+=/]{1,}+)\.(?
<param1>[A-Za-z0-9_+=/]{1,}+)\.(?<param2>[A-Za-z0-9_+=/]{1,})\.(?<param3>[A-Za-
z0-9_+=/]{2,})\|###\|(?<extdata>.*)\|",
            "message", "\|%{NUMBER:timestamp}\|###\|%{NUMBER:order}\|###\|(?
<deviceId>[a-zA-Z0-9._-]+)\|###\|%{NUMBER:userId}\|###\|(?<bizLine>[a-zA-Z0-
9._-]+)\|###\|%{NUMBER:appId}\|###\|(?<sdkVersion>v[0-9]+\.[0-9]+\.
[0.9]+)\|###\|%{NUMBER:eventId}\|###\|tp\.(?<param0>[A-Za-z0-9_+=/]{1,}+)\.(?
<param1>[A-Za-z0-9_+=/]{1,}+)\.(?<param2>[A-Za-z0-9_+=/]{2,})\|###\|(?
<extdata>.*)\|",
            "message", "\|%{NUMBER:timestamp}\|###\|%{NUMBER:order}\|###\|(?
<deviceId>[a-zA-Z0-9._-]+)\|###\|%{NUMBER:userId}\|###\|(?<bizLine>[a-zA-Z0-
9._-]+)\|###\|%{NUMBER:appId}\|###\|(?<sdkVersion>v[0-9]+\.[0-9]+\.
[0.9]+)\|###\|%{NUMBER:eventId}\|###\|tp\.(?<param0>[A-Za-z0-9_+=/]{1,}+)\.(?
<param1>[A-Za-z0-9_+=/]{2,})\|###\|(?<extdata>.*)\|"
        ]
    }

    date {
        timezone => "UTC"
        match => ["timestamp", "UNIX_MS"]
        target => "@ev_timestamp"
    }

    ruby {
        init => "require 'base64'"
        code =>
"event.set('extdata_decoded',Base64.decode64(event.get('extdata'))) if
event.include?('extdata') ;
event.set('param3_decoded',Base64.decode64(event.get('param3'))) if
event.include?('param3')"
    }
}

output {
    stdout { codec => rubydebug }
    elasticsearch {
        hosts => [ "elasticsearch:9200" ]
        index => "cuge-%{+YYYY.MM.dd}"
    }
}
```

主程序部署

复制一份`application-example.properties`到`application.properties`。

配置存储服务链接

先填入刚才部署的MySQL, Redis, ETCD的实例地址，及认证用户名密码。

配置Switch平台的认证用户名密码

CUGE 内置了一套可以通过管理页面动态地修改程序内部变量值的插件。如果被未经授权的用户访问，可能产生安全风险。因此，强烈建议您为`switch.endpoint.web.username`及`switch.endpoint.web.password`设置一套难以猜测的用户名密码。或者 如果您不需要Switch服务，您可以通过`switch.endpoint.enable=false`，`switch.endpoint.web=false`关闭Switch服务，

```
# Switch center
switch.endpoint.enable=true
switch.endpoint.web=true
switch.endpoint.web.username=
switch.endpoint.web.password=
```

配置邮箱发信服务

CUGE 在注册新用户、忘记密码、绑定邮箱等情况下，均需要向指定邮箱发送验证码，因此，您需要配置STMP发信服务，以便此功能生效。

```
# Mail Sender Configuration
spring.mail.host=
spring.mail.from=
spring.mail.port=
spring.mail.username=
spring.mail.password=
```

常见的QQ邮箱SMTP发信配置，是这个样子的

```
# Mail Sender Configuration
spring.mail.host=smtp.qq.com
spring.mail.from=QQ号码@qq.com
spring.mail.port=465
spring.mail.username=QQ号码
spring.mail.password=开启STMP服务时随机生成的密码
```

配置Elasticsearch实例地址

CUGE 需要连接到先前您配置的ElasticSearch服务，以便在管理后台展示数据看板。

```
# ElasticSearch
elasticsearch.host=
elasticsearch.port=
```

配置端SDK签名Secret

```
sdk.secure.android.secret=
sdk.secure.web.secret=
sdk.secure.key.storage.path=${user.home}/cuge/keys # 默认业务线密钥存储路径。一般不需要修改
sdk.secure.expire=172800
```

配置Eureka地址

CUGE 通过Eureka实现集群服务发现及对等机器注册。请将以下两个地址配置成**CUGE集群内其他机器可以互通互访的地址**。

```
# Instance name (related to crowd location)
eureka.client.serviceUrl.ugeZone=http://localhost:10010/eureka/
eureka.client.serviceUrl.defaultZone=http://localhost:10010/eureka/
```

（可选）配置人群文件及物料文件存储路径

一般不需要改动。不过如果您的主程序处于容器中，请将这些路径显式设置为能够持久化到物理设备上的挂载路径，否则可能出现容器重启后人群文件丢失的问题。

```
# Crowd files
crowd.bitmaskfile.path=${user.home}/cuge/crowd/bitmask
crowd.material.path=${user.home}/cuge/crowd/material
```

（可选）配置比特位人群是否进行压缩

开启比特位人群压缩能够大幅降低稠密人群的文件大小，但是会延长人群创建的时间。

如果您的业务规模其实不是很大，或者您的人群一般比较稀疏，不需要打开此选项。

```
crowd.bitmask.compact=false
```

配置内部RPC调用鉴权Secret

CUGE 会在需要访问集群内的其他实例时，向其发出RPC调用。为了避免未经授权的用户随便对CUGE集群中的机器发起RPC调用，您需要设置并保管好调用认证Secret。另外需要注意的是，CUGE集群内的所有机器均需要保持RPC Secret一致，否则将无法完成RPC调用。

```
# 内部RPC鉴权
rpc.secret=
```

最后，如果您需要区分环境，对于不同的环境应用不同的设置，您可以新建一个 `application-环境名.properties` 的文件，然后在其中覆盖掉刚才您配置的一些配置项。您不需要复制整个 `application.properties`。

主程序打包

执行

```
 mvn package -Dmaven.test.skip=true -DskipTests
```

生成构建产物 `user-growing-engine-start-0.0.1-SNAPSHOT.jar`。

第一次运行，需要初始化数据库表结构以及创建超级管理员账户，需要带参数运行，如下所示

```
java -jar user-growing-engine-start-0.0.1-SNAPSHOT.jar --database --admin --
email admin@example.com --password your-admin-password
```

携带初始化参数后，程序将自动从OSS拉取建表DDL，初始化数据库表结构，同时创建一个通过参数指定的邮箱及密码登录的超级管理员账号。

> 请不要在生产环境中加初始化参数，这会导致每次启动应用，都走一遍初始化流程，存在破坏表结构，丢失数据的风险！

当数据库结构以及超级管理员用户创建完成后，可以开始打Docker镜像。

镜像配置可以参考以下Dockerfile。

```
FROM openjdk:8-jdk

WORKDIR /app

COPY . /app

RUN rm -rf /etc/localtime

RUN ln -s /usr/share/zoneinfo/Asia/Shanghai /etc/localtime

RUN ls -l /etc/localtime

CMD ["java","-Xmx2g","-XX:+UseG1GC","-XX:+UseCompressedOops","-server","-
XX:InitiatingHeapOccupancyPercent=70","-Duser.timezone=Asia/Shanghai","-
Dproject.name=UserGrowingEngine","-
agentlib:jdwp=transport=dt_socket,server=y,suspend=n,address=8099","-jar","user-
growing-engine-start-0.0.1-SNAPSHOT.jar", "--spring.profiles.active=环境名"]
```

最后，在确保存储服务(MySQL, Redis, Etcd)及日志采集服务(Filebeats, Logstash, Elasticsearch)都正常运行后，可以启动CUGE。

参考启动命令如下。

```
docker run --rm -it -d -p 10010:10010 -p 8099:8099 -v /path/to/cuge:/root/cuge -
v /path/to/cuge/logs:/root/logs --name cuge -h 唯一标识当前实例的名称 --network
cuge-network cuge:deploy
```

查看启动日志是否正常，然后再测试相关功能是否可用。