应用服务器部署

概述

> 应用端包括我们开发的代码应用，以及所使用的第三方服务程序。原则上这些程序都已docker的方式运行在openshift平台上，这样我们不需要额外部署服务器资源，简化了部署。

代码应用

参见DevOps

问题：

针对不同环境，所使用的环境配置文件采用哪种方式部署比较好？

微服务依赖的第三方服务

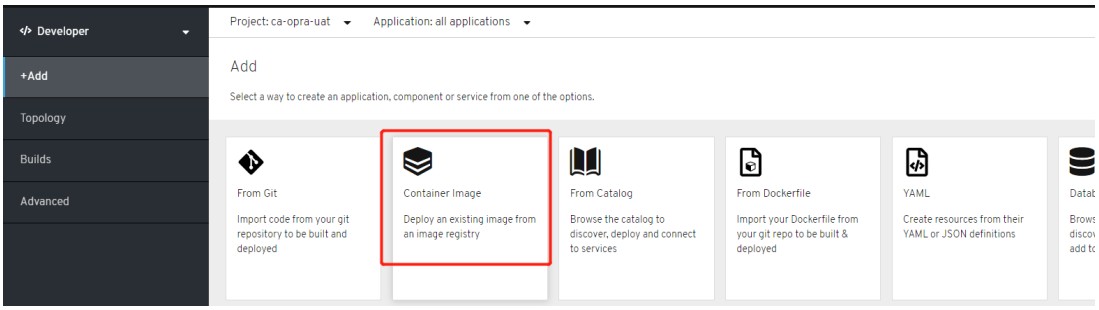
Consul（已废弃）

当前Openshift平台不能联网，镜像从Quay外部镜像仓库获取。镜像仓库地址：https://clair.ocp.acca/repository

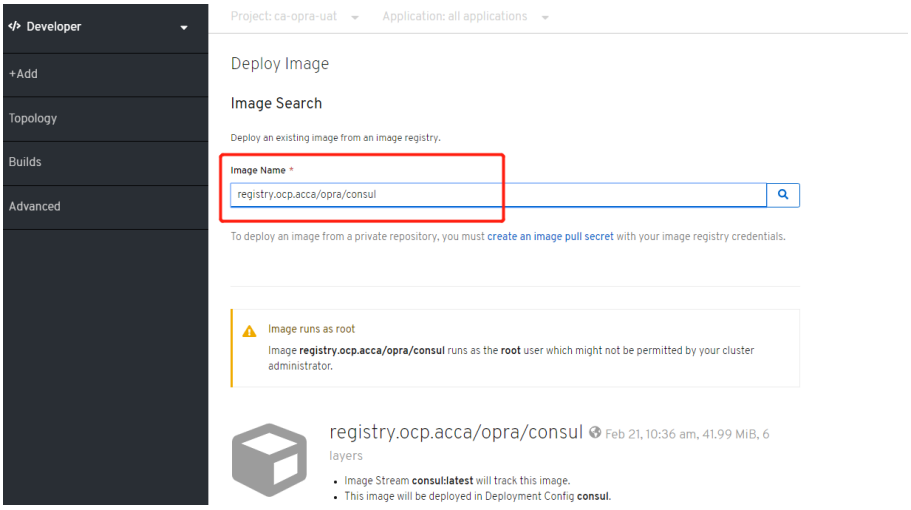
配置host地址

```
10.1.18.236 registry.ocp.acca
10.1.18.236 clair.ocp.acca
```

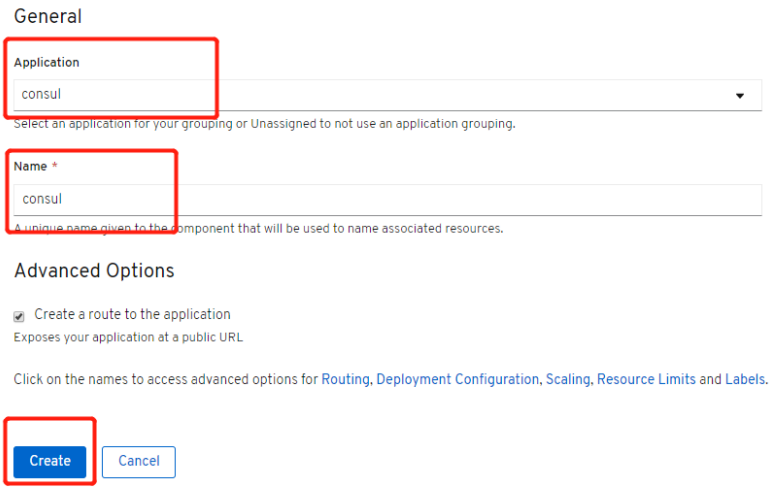
在oc平台选择从镜像发布consul



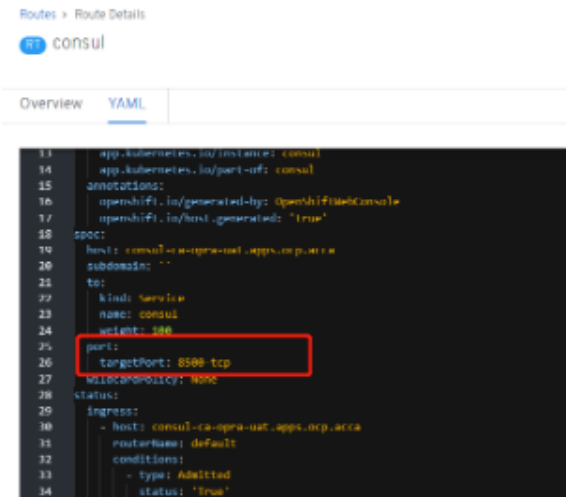
输入从Quay获取的镜像地址



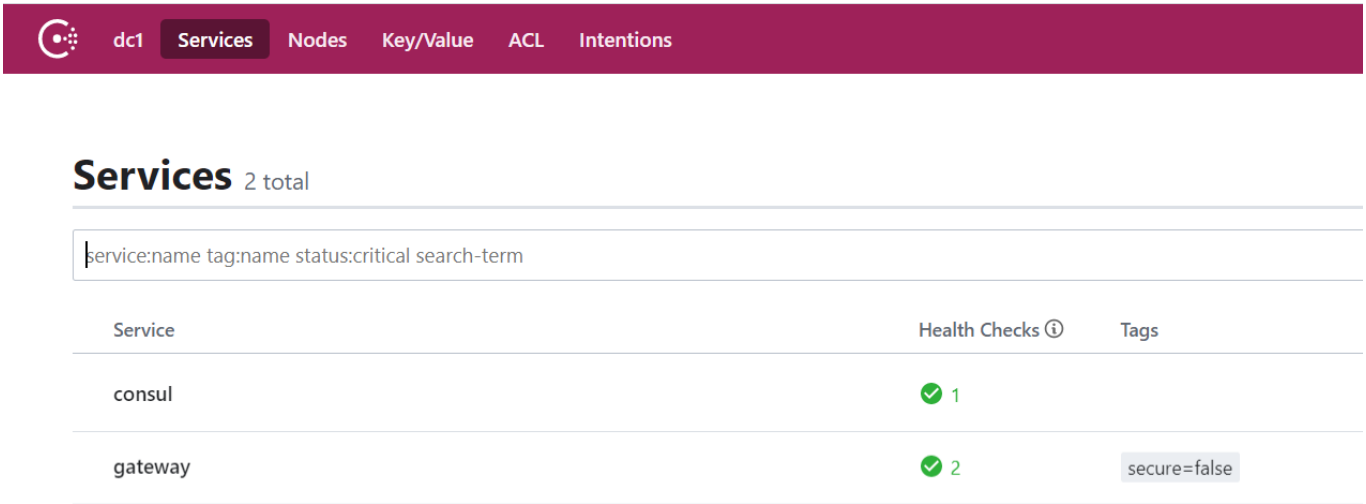
命名应用名字



修改consul默认生成的Route绑定的target端口为8500



通过Route在浏览器可以访问Consul的监控页面，如下



说明发布成功

Nacos安装

服务注册与发现服务器。

集群安装需要mysql，目前开发测试环境安装的是单机版。

单机版

从官网下载最新版本

https://github.com/alibaba/nacos/releases

解压：

```
tar -xvf nacos-server-1.3.0.tar.gz
```

修改日志级别：

```
        <appender name="naming-server"
              class="ch.qos.logback.core.rolling.RollingFileAppender">
        <file>${LOG_HOME}/naming-server.log</file>
        <append>true</append>
        <rollingPolicy class="ch.qos.logback.core.rolling.
SizeAndTimeBasedRollingPolicy">
            <fileNamePattern>${LOG_HOME}/naming-server.log.%d{yyyy-MM-dd}.%
i</fileNamePattern>
            <maxFileSize>100MB</maxFileSize>
            <MaxHistory>7</MaxHistory>
            <totalSizeCap>700MB</totalSizeCap>
            <cleanHistoryOnStart>true</cleanHistoryOnStart>
        </rollingPolicy>
        <encoder>
            <Pattern>%date %level %msg%n%n</Pattern>
            <charset>UTF-8</charset>
        </encoder>
    </appender>
appender
1naming-server
2naming-raft
3config-memory
4config-server
5config-client-request
6config-trace
```

启动：

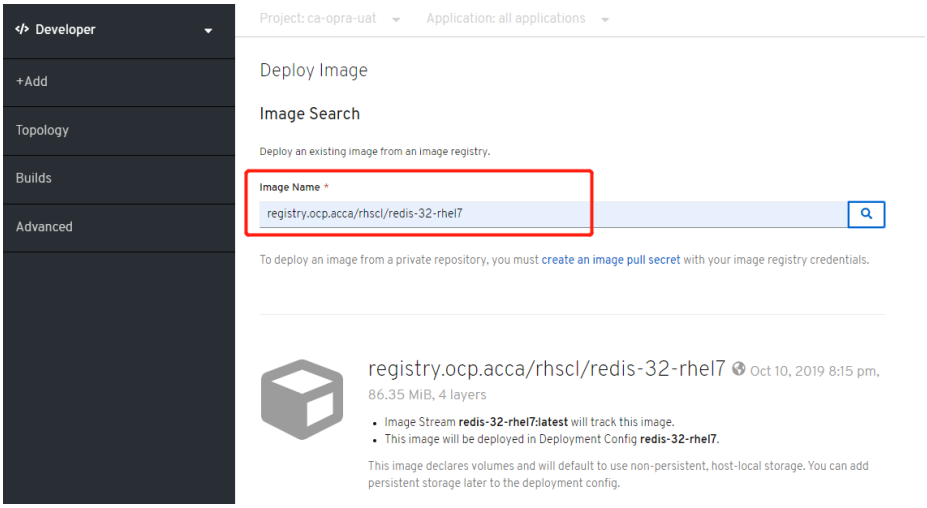
```
cd  ${nacos_home}/bin
./startup.sh -m standalone
```

集群版

待补充

redis-openshift

依然从Quay获取redis的镜像地址，在OC平台选择从镜像发布



后续过程与发布Consul过程相同

redis-虚拟机

1.安装准备

下载 redis-5.0.8.tar.gz　安装包（10.1.19.208:~上有备份）

下载 redis-5.0.8.tar_installed.tar　已编译好的安装包（10.1.19.208:~上有备份）

1.1 检查并设置文件数设置及用户最大进程数

```
sudo vi /etc/security/limits.conf


#
* soft nofile 65536
* hard nofile 131072
* soft nproc 65536
* hard nproc 65536


#


#
sudo vi /etc/security/limits.d/20-nproc.conf


#
*          soft    nproc     65536
root       soft    nproc     unlimited
```

1.2 修改单个进程最大线程数据

```
sudo vi /etc/sysctl.conf


#
vm.max_map_count=655350

#
sudo sysctl -p /etc/sysctl.conf
```

2.安装redis

```
#


#
tar -zxvf redis-5.0.8.tar.gz -C /data/

cd /data/redis-5.0.8/src


#gcc
sudo yum install -y gcc


#
make MALLOC=libc
make install PREFIX=/data/redis-5.0.8

#
cp /data/redis-5.0.8/redis.conf /data/redis-5.0.8/bin/
vi /data/redis-5.0.8/bin/redis.conf


#
bind 127.0.0.1  --> #bind 127.0.0.1
protected-mode yes --> protected-mode no
daemonize no --> daemonize yes

#
----------------------------------------------------
----------------------------------------------------
# redis


#
tar -zxvf redis-5.0.8_installed.tar -C /data/


#10.1.18.2810.1.18.251,10.1.18.251
vi /data/redis-5.0.8/bin/redis.conf


#
replicaof 10.1.18.28 6379
```

3. 启动

```
#
/data/redis-5.0.8/bin/redis-server /data/redis-5.0.8/bin/redis.conf
```

4. 测试是否成功

```
#
/data/redis-5.0.8/bin/redis-cli -h 127.0.0.1 -p 6379
set hello word
get hello
```

测试效果如下：

```
[appadm@i-opra-t-sys-2_src]$ /data/redis-5.0.8/bin/redis-cli -h 127.0.0.1 -p 6379
127.0.0.1:6379> set hello word
OK
127.0.0.1:6379> get hello
"word"
127.0.0.1:6379> exit
```

kafka

1. 安装准备

下载　kafka_2.12-2.5.0.tgz　　安装包（10.1.19.221上有备份）

下载 jdk-8u241-linux-x64.tar.gz 安装包（10.1.17.248上有备份）

2操作系统准备

2.1检查并设置文件数设置

```
sudo vi /etc/security/limits.conf


#
* soft nofile 65536
* hard nofile 131072
* soft nproc 65536
* hard nproc 65536


#


#
sudo vi /etc/security/limits.d/20-nproc.conf


#
*          soft    nproc     65536
root       soft    nproc     unlimited
```

2.2修改单个进程最大线程数据

```
sudo vi /etc/sysctl.conf


#
vm.max_map_count=655350


#
sudo sysctl -p /etc/sysctl.conf
```

3.安装JDK

```
#
tar -zxvf jdk-8u241-linux-x64.tar.gz -C /data/
#
sudo ln -s /data/jdk1.8.0_241/ /usr/jdk
#profile
sudo vi /etc/profile


#
JAVA_HOME=/usr/jdk/
CLASSPATH=$JAVA_HOME/lib/
PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH

#
source /etc/profile

#
java -version
```

4. 安装kafka

```
#
tar -zxvf kafka_2.12-2.5.0.tgz -C /data/
```

5. zookeeper启动属性调整

```
vi /data/kafka_2.12-2.5.0/config/zookeeper.properties

------------------------------------------------------------
#
#
dataDir=/data/zookeeper



-------------------------------------------------------------
#,
#
dataDir=/data/zookeeper
#
tickTime=2000
initLimit=10
syncLimit=5
#broker Id
server.1=10.1.18.27:2888:3888
server.2=10.1.18.247:2888:3888
server.3=10.1.18.254:2888:3888


---------------------------------------------------------------
#
10.1.18.27
mkdir /data/zookeeper/
echo 1 > /data/zookeeper/myid


10.1.18.247
mkdir /data/zookeeper/
echo 2 > /data/zookeeper/myid


10.1.18.254
mkdir /data/zookeeper/
echo 3 > /data/zookeeper/myid




#zookeeper
vi /data/kafka_2.12-2.5.0/bin/zookeeper-server-start.sh


export KAFKA_HEAP_OPTS="-Xmx512M -Xms512M"

export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"
```

6. kafka启动属性调整

```
vi /data/kafka_2.12-2.5.0/config/server.properties

------------------------------------------------------------
#
#
broker.id=1
listeners=PLAINTEXT://10.1.18.27:9092  #ip
log.dirs=/data/kafka-logs
num.partitions=8
log.flush.interval.messages=10000
log.flush.interval.ms=1000
zookeeper.connect=10.1.18.27:2181
#
delete.topic.enable=true



-------------------------------------------------------------
#,
#
broker.id=1 #1,broker.id=1 2,broker.id=2 3,broker.id=3,
listeners=PLAINTEXT://10.1.18.27:9092  #ip
log.dirs=/data/kafka-logs
#369*2365
num.partitions=5
log.flush.interval.messages=10000
log.flush.interval.ms=1000
zookeeper.connect=10.1.18.27:2181,10.1.18.247:2181,10.1.18.254:2181
#,topicserver.propertiesdelete.topic.enable=true
delete.topic.enable=true
#ip
host.name=10.1.18.27

#
default.replication.factor=3

#offsets
offsets.topic.replication.factor=3
transaction.state.log.replication.factor=3
transaction.state.log.min.isr=2



#kafka
vi /data/kafka_2.12-2.5.0/bin/kafka-server-start.sh


export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"


export KAFKA_HEAP_OPTS="-Xmx5G -Xms5G -XX:G1HeapRegionSize=5M -XX:
```

```
MetaspaceSize=96m -XX:MinMetaspaceFreeRatio=50 -XX:
MaxMetaspaceFreeRatio=80"
export JMX_PORT="9999"  jmx
```

7. zookeeper启动

```
mkdir /data/logs
nohup /data/kafka_2.12-2.5.0/bin/zookeeper-server-start.sh /data/kafka_2.
12-2.5.0/config/zookeeper.properties >>/data/logs/nohup_zk.log &
```

8. kafka启动

```
nohup /data/kafka_2.12-2.5.0/bin/kafka-server-start.sh /data/kafka_2.12-
2.5.0/config/server.properties >>/data/logs/nohup_kafka.log &
```

Kafka eagle安装

安装文件放在10.1.18.254机器/data/kafkaeagle目录下

解压并添加环境变量

```
tar -zxvf kafka-eagle-xxx-bin.tar.gz
==============================================================

vi /etc/profile
export KE_HOME=
export PATH=$PATH:$KE_HOME/bin

================================================================


kafka eagle conf
# Multi zookeeper&kafka cluster list -- The client connection address of
the Zookeeper cluster is set here
kafka.eagle.zk.cluster.alias=cluster1,cluster2
cluster1.zk.list=tdn1:2181,tdn2:2181,tdn3:2181
cluster2.zk.list=xdn1:2181,xdn2:2181,xdn3:2181
```

```
# It is important to note that the '/hadoop/kafka-eagle/db' path must
exist.
kafka.eagle.url=jdbc:sqlite:/data/kafka-eagle/db/ke.db


######################################
# kafka metrics, 15 days by default
######################################
kafka.eagle.metrics.charts=true




=====================================================================

cd ${KE_HOME}/bin
chmod +x ke.sh
./ke.sh start


ke.sh restart
ke.sh stop



http://10.1.18.254:8048/ke/account/signin?/ke/
admin
123456


kafkajmxkafka-server-start.sh

if [ "x$KAFKA_HEAP_OPTS" = "x" ]; then
    export KAFKA_HEAP_OPTS="-Xmx1G -Xms1G"
    export JMX_PORT="9999"
fi
```

配置管理服务器

elasticsearch

1安装准备

下载 elasticsearch-6.4.3.tar.gz　安装包（10.1.17.248上有备份）

下载　elasticsearch-analysis-ik-6.4.3.zip　分词插件（10.1.17.248上有备份）

下载 jdk-8u241-linux-x64.tar.gz 安装包（10.1.17.248上有备份）

2操作系统准备

如下所有操作需要非root用户

操作系统注意事项

```
－查看防火墙状态
systemctl status firewalld.service
－永久关闭防火墙
systemctl disable firewalld.service (开启 enable)
－查看selinux是否关闭
getenforce
－显示permissive即为开启状态 Enforcing 为关闭状态
－临时关闭selinux
setenforce 1 --对应Enforcing 开启状态
setenforce 0 --对应permissive 关闭状态
－永久关闭 需要重启服务器
vi /etc/selinux/config/
```

2.1 检查并设置文件数设置

```
sudo vi /etc/security/limits.conf


#
* soft nofile 65536
* hard nofile 131072
* soft nproc 65536
* hard nproc 65536

* soft memlock unlimited
* hard memlock unlimited

#


#
sudo vi /etc/security/limits.d/20-nproc.conf


#
*          soft    nproc     65536
root       soft    nproc     unlimited
```

2.2 修改单个进程最大线程数据

```
sudo vi /etc/sysctl.conf


#
vm.max_map_count=655350
vm.swappiness=0

#
sudo sysctl -p /etc/sysctl.conf
```

3安装Elasticsearch

```
#
tar -zxvf elasticsearch-6.4.3.tar.gz  -C /data/

vi /data/elasticsearch-6.4.3/config/elasticsearch.yml

#
cluster.name: opra-es-uat
node.name: node-1
path.data: /data/es-data/data
path.logs: /data/es-data/logs
bootstrap.memory_lock: true
network.host: 0.0.0.0
http.port: 9200
bootstrap.system_call_filter: false
http.cors.enabled: true
http.cors.allow-origin: "*"


#
discovery.zen.ping.unicast.hosts: ["10.1.21.26", "10.1.21.27",
"10.1.21.28"]
discovery.zen.minimum_master_nodes: 2





#jvm
vi /data/elasticsearch-6.4.3/config/jvm.options


#16G8g
-Xms8g
-Xmx8g
```

4安装分词插件 elasticsearch-analysis-ik

```
#

unzip elasticsearch-analysis-ik-6.4.3.zip -d /data/elasticsearch-6.4.3
/plugins/analysis-ik
```

5安装JDK

```
#
tar -zxvf jdk-8u241-linux-x64.tar.gz -C /data/
#
sudo ln -s /data/jdk1.8.0_241/ /usr/jdk
#profile
sudo vi /etc/profile


#
JAVA_HOME=/usr/jdk/
CLASSPATH=$JAVA_HOME/lib/
PATH=$PATH:$JAVA_HOME/bin
export PATH JAVA_HOME CLASSPATH

#
source /etc/profile

#
java -version
```

6启动elasticsearch

```
/data/elasticsearch-6.4.3/bin/elasticsearch -d
```

修改最大查询条数的限制

es索引的查询，默认最大支持1w条，需要修改。5.*版本之后，不支持在配置文件中修改，需要使用restful的api服务来修改。使用postman运行下面的url。

```
http://10.1.19.224:9200/_settings
urles
Http Method PUT
Request Body
{ "index" : { "max_result_window" : 100000000}}





{
    "acknowledged": true
}
```

Kibana安装

1. 确认elasticsearch是否为集群

1.1 当elasticsearch 为非集群环境

直接跳到步骤2安装kibana即可

1.2 当elasticsearch 为集群环境

```
# 1.elasticsearches
# 2. elasticsearch.yml


#es
cluster.name: opra-es-uattest
#
node.master: false
node.data: false
node.ingest: false
#
node.name: node-4
#esip
network.host: 10.1.18.251
http.port: 9200
#es
discovery.zen.ping.unicast.hosts: ["10.1.17.15", "10.1.17.240",
"10.1.19.224"]


# 3.


/data/elasticsearch-6.4.3/bin/elasticsearch -d
```

2. kibana安装

kibana用来做es的数据展示工具。

可以做日志展示工具。

解压/data/software/kibana-6.4.3-linux-x86_64.tar.gz ，修改config/kibana.yaml文件

```
# Specifies the address to which the Kibana server will bind. IP addresses
and host names are both valid values.
# The default is 'localhost', which usually means remote machines will not
be able to connect.
# To allow connections from remote users, set this parameter to a non-
loopback address.
server.host: "10.1.17.13"     kibana



# The URL of the Elasticsearch instance to use for all your queries.
# esip
# es2esip
elasticsearch.url: "http://10.1.17.13:9200"
```

启动

```
nohup ./kibana > /dev/null &
```

访问地址：

http://10.1.17.13:5601/

kafka虚拟机安装

下载kafka二进制文件:https://www.apache.org/dyn/closer.cgi?path=/kafka/2.4.0/kafka_2.11-2.4.0.tgz

暂时选用kafka_2.11-2.4.0

解压,复制到服务器

进入kafka目录

修改zookeeper配置

```
vi config/zookeeper.properties
#zookeeper
dataDir=/data/zookeeper
```

修改kafka配置

```
vi config/server.properties
#id
broker.id=0
#
log.dirs=/data/kafka-logs
#
num.partitions=4
#zookeeperip
zookeeper.connect=localhost:2181
```

将bin下bash文件设置成可执行

```
chomd +x bin/*
```

启动zookeeper

```
bin/zookeeper-server-start.sh -daemon config/zookeeper.properties
```

启动kafka

```
bin/kafka-server-start.sh -daemon config/server.properties
```

查看kafka进程

```
ps -ef|grep kafka
```

查看端口占用

```
netstat -tunple|grep 9092
```

adminserver

adminserver是一个监控微服务的开源程序，可以查看在线日志，修改日志级别，查看应用jvm等功能。

adminserver采用代码启动的方式，具体对应的git仓库为：

http://git.acca.com.cn:7990/projects/OPRA-GIT/repos/araf-admin-server/browse

部署方式与一般的微服务部署完全相同，启动后访问地址为：

http://adminserver-ca-opra-dev.apps.ocp.acca/#/applications 注意替换不同的环境参数

zipkinserver

zipkinserver是一个监控服务调用的的开源程序，可以查看服务之间的调用情况，依赖情况等。

zipkinserver采用代码启动的方式，具体对应的git仓库为：

http://git.acca.com.cn:7990/projects/OPRA-GIT/repos/araf-zipkin-server/browse

部署方式与一般的微服务部署完全相同，启动后访问地址为：

http://zipkinserver-ca-opra-dev.apps.ocp.acca/ 注意替换不同的环境参数

依赖图的生成

生成依赖图，需要额外的运行一个命令。

1、登录到elasticsearch所在服务器。

```
10.1.17.13/data/software
zipkin-dependencies-2.4.2.jar
zipkin-dependencies.sh
es


zipkin-dependencies.shes
STORAGE_TYPE=elasticsearch ES_HOSTS=10.1.17.13 java -jar /data/software
/zipkin-dependencies-2.4.2.jar


crontab -e

0 17 * * * /data/software/zipkin-dependencies.sh >> /data/software/zipkin-
de.log 2>&1
```

seata-分布式事务

1.PG数据库初始化

```
-- -------------------------------- The script used when storeMode is 'db'
--------------------------------
-- the table to store GlobalSession data
CREATE TABLE IF NOT EXISTS global_table
(
    xid                       VARCHAR(128) NOT NULL,
    transaction_id            BIGINT,
    status                    SMALLINT     NOT NULL,
    application_id            VARCHAR(32),
    transaction_service_group VARCHAR(32),
    transaction_name          VARCHAR(128),
    timeout                   INT,
    begin_time                BIGINT,
    application_data          VARCHAR(2000),
    gmt_create                TIMESTAMP(0),
    gmt_modified              TIMESTAMP(0),
    CONSTRAINT pk_global_table PRIMARY KEY (xid)
);

CREATE INDEX idx_gmt_modified_status ON global_table (gmt_modified,
status);
CREATE INDEX idx_transaction_id ON global_table (transaction_id);

-- the table to store BranchSession data
CREATE TABLE IF NOT EXISTS branch_table
(
    branch_id         BIGINT       NOT NULL,
    xid               VARCHAR(128) NOT NULL,
    transaction_id    BIGINT,
    resource_group_id VARCHAR(32),
    resource_id       VARCHAR(256),
    branch_type       VARCHAR(8),
    status            SMALLINT,
    client_id         VARCHAR(64),
    application_data  VARCHAR(2000),
    gmt_create        TIMESTAMP(6),
    gmt_modified      TIMESTAMP(6),
    CONSTRAINT pk_branch_table PRIMARY KEY (branch_id)
);

CREATE INDEX idx_xid ON branch_table (xid);

-- the table to store lock data
CREATE TABLE IF NOT EXISTS lock_table
(
    row_key        VARCHAR(128) NOT NULL,
    xid            VARCHAR(96),
    transaction_id BIGINT,
    branch_id      BIGINT       NOT NULL,
    resource_id    VARCHAR(256),
    table_name     VARCHAR(32),
    pk             VARCHAR(36),
```

```
    gmt_create     TIMESTAMP(0),
    gmt_modified   TIMESTAMP(0),
    CONSTRAINT pk_lock_table PRIMARY KEY (row_key)
);

CREATE INDEX idx_branch_id ON lock_table (branch_id);
```

2.nacos初始化配置

2.1下载config-center.zip并解压

```
 unzip config-center.zip -d config-center
```

2.2#修改config.txt中数据库相关参数

```
//


store.db.dbType=postgresql
store.db.driverClassName=org.postgresql.Driver
store.db.url=jdbc:postgresql://10.1.19.5:5432/seata
store.db.user=seata
store.db.password=seata
```

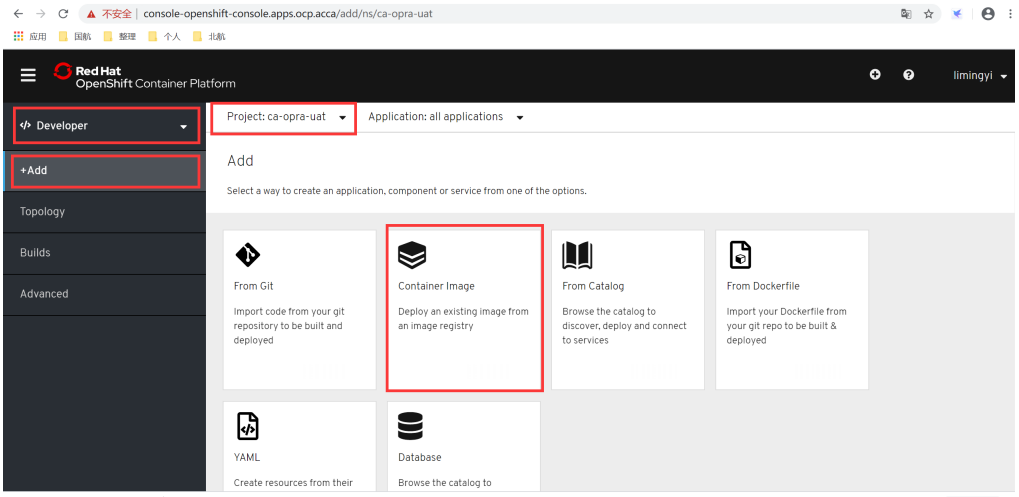
2.3进入nacos目录，执行脚本

```
chmod +x nacos-config.sh
./nacos-config.sh -h 10.1.17.17 -p 8848 -g SEATA_GROUP -t dev


//
10.1.17.17=nacosIp-
8848=nacosPort-
SEATA_GROUP=groupName-
dev=nacosNamespace -uatca
```

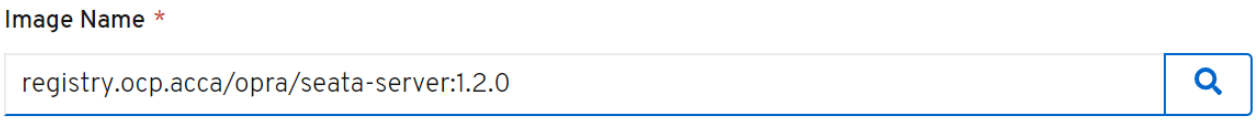
3.openshift部署seata

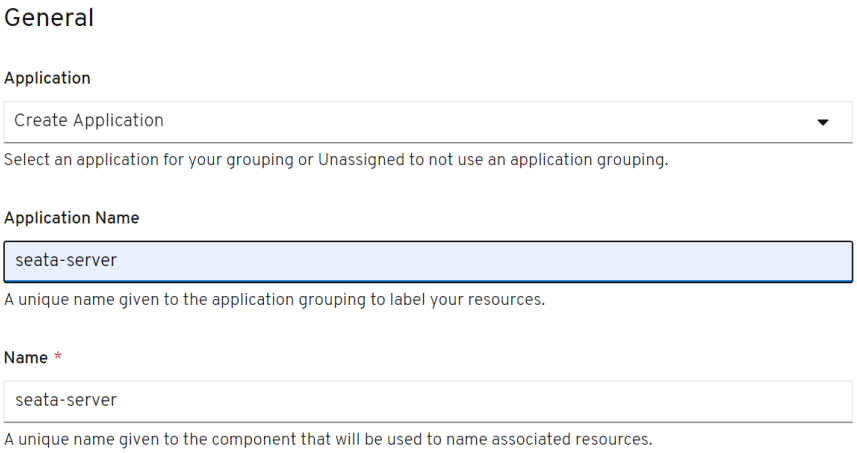
3.1从镜像安装seata

3.1.1登录openshift平台，选择相应的环境，并选择container image



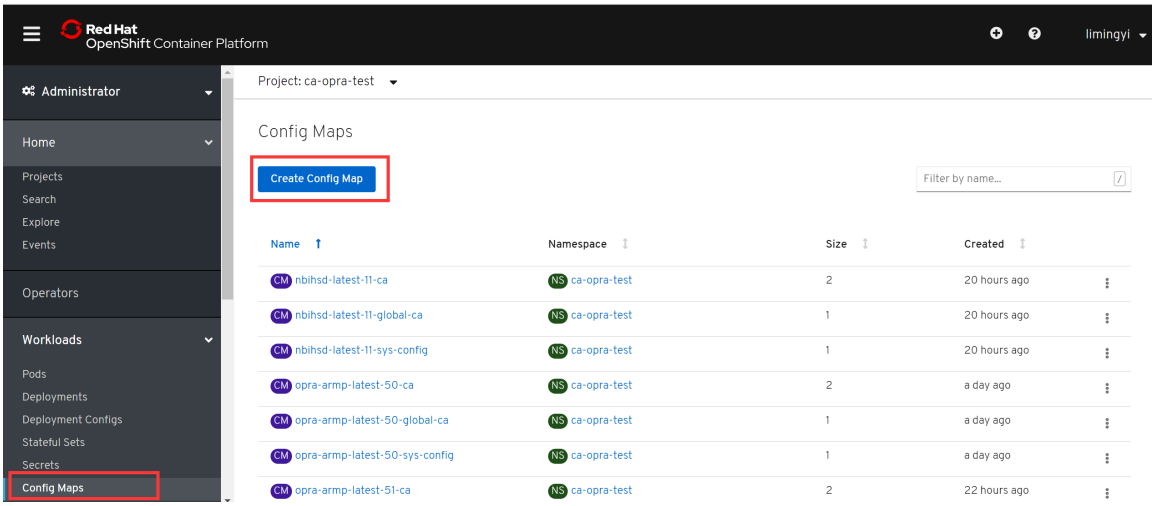
3.1.2录入相关信息，create registry.ocp.acca/opra/seata-server:1.2.0





3.2准备Config Maps

3.2.1进入

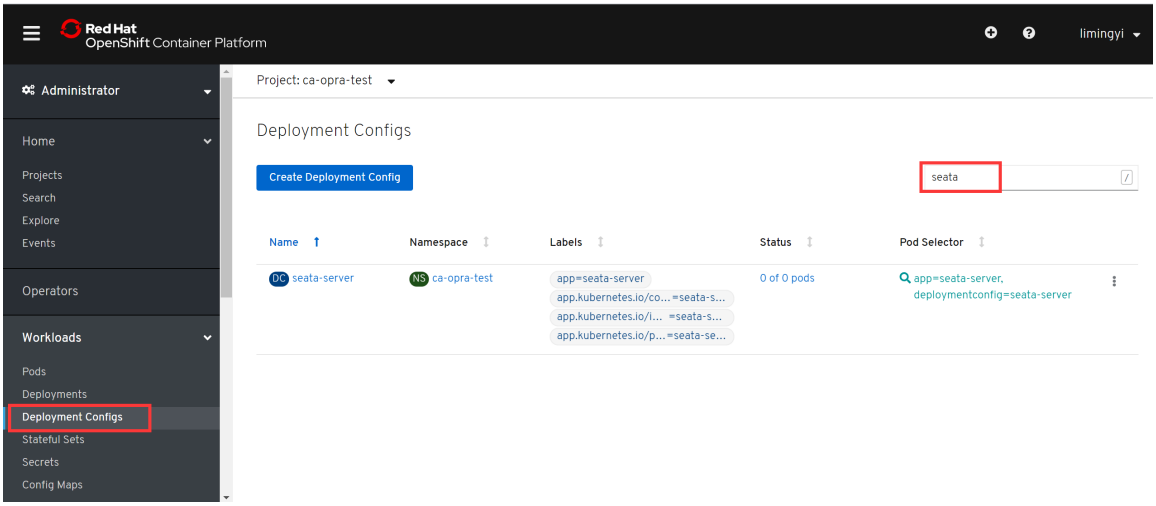

Red Hat
OpenShift Container Platform
limingyi
Administrator
Home
Projects
Search
Explore
Events
Operators
Workloads
Pods
Deployments
Deployment Configs
Stateful Sets
Secrets
Config Maps
Project: ca-opra-test
Config Maps
Create Config Map
Filter by name...
Name
Namespace
Size
Created
nbihsd-latest-11-ca
ca-opra-test
2
20 hours ago
nbihsd-latest-11-global-ca
ca-opra-test
1
20 hours ago
nbihsd-latest-11-sys-config
ca-opra-test
1
20 hours ago
opra-armp-latest-50-ca
ca-opra-test
2
a day ago
opra-armp-latest-50-global-ca
ca-opra-test
1
a day ago
opra-armp-latest-50-sys-config
ca-opra-test
1
a day ago
opra-armp-latest-51-ca
ca-opra-test
2
22 hours ago

3. 2. 2

```
//env
namespace: ca-opra-test    namespace: ca-opra-uat,openshift

namespace = "test"     namespace = "uatca"




kind: ConfigMap
apiVersion: v1
metadata:
  name: seata-server-config
  namespace: ca-opra-test
data:
  registry.conf: |
    registry {
      type = "nacos"
      nacos {
        application = "seata-server"
        serverAddr = "10.1.17.17:8848"
        namespace = "test"
        cluster = "default"
        username = ""
        password = ""
      }
    }
    config {
      type = "nacos"
      nacos {
        serverAddr = "10.1.17.17:8848"
        namespace = "test"
        group = "SEATA_GROUP"
        username = ""
        password = ""
      }
    }
```

3.3更新Deployment Configs

3.3.1进入并搜索seata



3.3.2修改yaml，添加标红部分参数

```
52      labels:
53        app: seata-server
54        deploymentconfig: seata-server
55      annotations:
56        openshift.io/generated-by: OpenShiftWebConsole
57    spec:
58      volumes:
59        - name: seata-config
60          configMap:
61            name: seata-server-config
62            defaultMode: 420
63      containers:
64        - resources:
65            limits:
66              cpu: '2'
67              memory: 4Gi
68            requests:
69              cpu: '2'
70              memory: 1Gi
71          terminationMessagePath: /dev/termination-log
72          name: seata-server
73          env:
74            - name: SEATA_CONFIG_NAME
75              value: 'file:/data/seata-config/registry'
76          ports:
77            - name: http
78              containerPort: 8091
79              protocol: TCP
80          imagePullPolicy: IfNotPresent
81          volumeMounts:
82            - name: seata-config
83              mountPath: /data/seata-config
84          terminationMessagePolicy: File
85          image: >-
86            registry.ocp.acca/opra/seata-server@sha256:7b350ebce39c6f7d1c2c46fdb13697ecdc8d8d3e81ebebac65a0093a831c853e
87      restartPolicy: Always
88      terminationGracePeriodSeconds: 30
89      dnsPolicy: ClusterFirst
90      securityContext: {}
91      schedulerName: default-scheduler
92  status:
93    observedGeneration: 37
94    details:
```

```
    spec:
      volumes:
        - name: seata-config
          configMap:
            name: seata-server-config
            defaultMode: 420
      containers:
        - resources:
            limits:
              cpu: '2'
              memory: 4Gi
            requests:
              cpu: '2'
              memory: 1Gi
          terminationMessagePath: /dev/termination-log
          name: seata-server
          env:
            - name: SEATA_CONFIG_NAME
              value: 'file:/data/seata-config/registry'
          ports:
            - name: http
              containerPort: 8091
              protocol: TCP
          imagePullPolicy: IfNotPresent
          volumeMounts:
            - name: seata-config
              mountPath: /data/seata-config
          terminationMessagePolicy: File
          image: >-
            registry.ocp.acca/opra/seata-server@sha256:
7b350ebce39c6f7d1c2c46fdb13697ecdc8d8d3e81ebebac65a0093a831c853e
      restartPolicy: Always
      terminationGracePeriodSeconds: 30
      dnsPolicy: ClusterFirst
      securityContext: {}
      schedulerName: default-scheduler
```

3.4修改完yaml后，会自动重新部署容器

skywalking安装

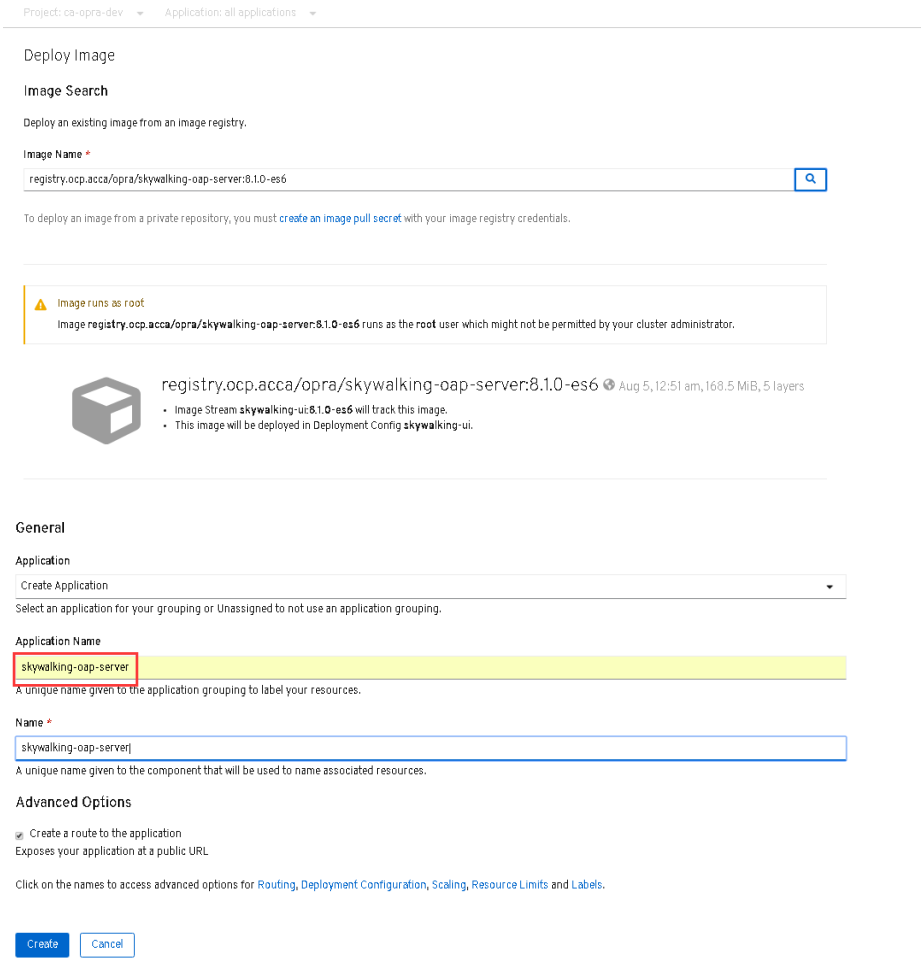
skywalking信息收集服务器安装

部署镜像

直接通过container image部署

image名为 ： registry.ocp.acca/opra/skywalking-oap-server:8.1.0-es6

注意部署的服务名必须为 skywalking-oap-server 如图中红框中所示。



添加config map

修改其中的命名空间，与es服务器地址

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: skywalking-oap-config
  namespace:
data:
  application.yml: |
    # Licensed to the Apache Software Foundation (ASF) under one or more
    # contributor license agreements.  See the NOTICE file distributed with
    # this work for additional information regarding copyright ownership.
    # The ASF licenses this file to You under the Apache License, Version
2.0
    # (the "License"); you may not use this file except in compliance with
    # the License.  You may obtain a copy of the License at
    #
    #     http://www.apache.org/licenses/LICENSE-2.0
    #
    # Unless required by applicable law or agreed to in writing, software
```

```
    # distributed under the License is distributed on an "AS IS" BASIS,
    # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied.
    # See the License for the specific language governing permissions and
    # limitations under the License.


    cluster:
      selector: ${SW_CLUSTER:standalone}
      standalone:
      # Please check your ZooKeeper is 3.5+, However, it is also
compatible with ZooKeeper 3.4.x. Replace the ZooKeeper 3.5+
      # library the oap-libs folder with your ZooKeeper 3.4.x library.


    core:
      selector: ${SW_CORE:default}
      default:
        # Mixed: Receive agent data, Level 1 aggregate, Level 2 aggregate
        # Receiver: Receive agent data, Level 1 aggregate
        # Aggregator: Level 2 aggregate
        role: ${SW_CORE_ROLE:Mixed} # Mixed/Receiver/Aggregator
        restHost: ${SW_CORE_REST_HOST:0.0.0.0}
        restPort: ${SW_CORE_REST_PORT:12800}
        restContextPath: ${SW_CORE_REST_CONTEXT_PATH:/}
        restMinThreads: ${SW_CORE_REST_JETTY_MIN_THREADS:1}
        restMaxThreads: ${SW_CORE_REST_JETTY_MAX_THREADS:200}
        restIdleTimeOut: ${SW_CORE_REST_JETTY_IDLE_TIMEOUT:30000}
        restAcceptorPriorityDelta: ${SW_CORE_REST_JETTY_DELTA:0}
        restAcceptQueueSize: ${SW_CORE_REST_JETTY_QUEUE_SIZE:0}
        gRPCHost: ${SW_CORE_GRPC_HOST:0.0.0.0}
        gRPCPort: ${SW_CORE_GRPC_PORT:11800}
        gRPCSslEnabled: ${SW_CORE_GRPC_SSL_ENABLED:false}
        gRPCSslKeyPath: ${SW_CORE_GRPC_SSL_KEY_PATH:""}
        gRPCSslCertChainPath: ${SW_CORE_GRPC_SSL_CERT_CHAIN_PATH:""}
        gRPCSslTrustedCAPath: ${SW_CORE_GRPC_SSL_TRUSTED_CA_PATH:""}
        downsampling:
          - Hour
          - Day
        # Set a timeout on metrics data. After the timeout has expired,
the metrics data will automatically be deleted.
        enableDataKeeperExecutor: ${SW_CORE_ENABLE_DATA_KEEPER_EXECUTOR:
true} # Turn it off then automatically metrics data delete will be close.
        dataKeeperExecutePeriod: ${SW_CORE_DATA_KEEPER_EXECUTE_PERIOD:5} #
How often the data keeper executor runs periodically, unit is minute
        recordDataTTL: ${SW_CORE_RECORD_DATA_TTL:3} # Unit is day
        metricsDataTTL: ${SW_CORE_METRICS_DATA_TTL:7} # Unit is day
        # Cache metrics data for 1 minute to reduce database queries, and
if the OAP cluster changes within that minute,
        # the metrics may not be accurate within that minute.
        enableDatabaseSession: ${SW_CORE_ENABLE_DATABASE_SESSION:true}
        topNReportPeriod: ${SW_CORE_TOPN_REPORT_PERIOD:10} # top_n record
worker report cycle, unit is minute
```

```
        # Extra model column are the column defined by in the codes, These
columns of model are not required logically in aggregation or further
query,
        # and it will cause more load for memory, network of OAP and
storage.
        # But, being activated, user could see the name in the storage
entities, which make users easier to use 3rd party tool, such as Kibana-
>ES, to query the data by themselves.
        activeExtraModelColumns: ${SW_CORE_ACTIVE_EXTRA_MODEL_COLUMNS:
false}
        # The max length of service + instance names should be less than
200
        serviceNameMaxLength: ${SW_SERVICE_NAME_MAX_LENGTH:70}
        instanceNameMaxLength: ${SW_INSTANCE_NAME_MAX_LENGTH:70}
        # The max length of service + endpoint names should be less than
240
        endpointNameMaxLength: ${SW_ENDPOINT_NAME_MAX_LENGTH:150}
    storage:
      selector: ${SW_STORAGE:elasticsearch}
      elasticsearch:
        nameSpace: ${SW_NAMESPACE:""}
        clusterNodes: ${SW_STORAGE_ES_CLUSTER_NODES:es:9200}
        protocol: ${SW_STORAGE_ES_HTTP_PROTOCOL:"http"}
        trustStorePath: ${SW_STORAGE_ES_SSL_JKS_PATH:""}
        trustStorePass: ${SW_STORAGE_ES_SSL_JKS_PASS:""}
        user: ${SW_ES_USER:""}
        password: ${SW_ES_PASSWORD:""}
        secretsManagementFile: ${SW_ES_SECRETS_MANAGEMENT_FILE:""} #
Secrets management file in the properties format includes the username,
password, which are managed by 3rd party tool.
        dayStep: ${SW_STORAGE_DAY_STEP:1} # Represent the number of days
in the one minute/hour/day index.
        indexShardsNumber: ${SW_STORAGE_ES_INDEX_SHARDS_NUMBER:1} # Shard
number of new indexes
        superDatasetIndexShardsFactor:
${SW_STORAGE_ES_SUPER_DATASET_INDEX_SHARDS_FACTOR:5} # Super data set has
been defined in the codes, such as trace segments. This factor provides
more shards for the super data set, shards number = indexShardsNumber *
superDatasetIndexShardsFactor. Also, this factor effects Zipkin and Jaeger
traces.
        indexReplicasNumber: ${SW_STORAGE_ES_INDEX_REPLICAS_NUMBER:0}
        bulkActions: ${SW_STORAGE_ES_BULK_ACTIONS:1000} # Execute the bulk
every 1000 requests
        flushInterval: ${SW_STORAGE_ES_FLUSH_INTERVAL:10} # flush the bulk
every 10 seconds whatever the number of requests
        concurrentRequests: ${SW_STORAGE_ES_CONCURRENT_REQUESTS:2} # the
number of concurrent requests
        resultWindowMaxSize: ${SW_STORAGE_ES_QUERY_MAX_WINDOW_SIZE:10000}
        metadataQueryMaxSize: ${SW_STORAGE_ES_QUERY_MAX_SIZE:5000}
        segmentQueryMaxSize: ${SW_STORAGE_ES_QUERY_SEGMENT_SIZE:200}
        profileTaskQueryMaxSize: ${SW_STORAGE_ES_QUERY_PROFILE_TASK_SIZE:
200}
        advanced: ${SW_STORAGE_ES_ADVANCED:""}
```

```
    agent-analyzer:
      selector: ${SW_AGENT_ANALYZER:default}
      default:
        sampleRate: ${SW_TRACE_SAMPLE_RATE:10000} # The sample rate
precision is 1/10000. 10000 means 100% sample in default.
        slowDBAccessThreshold: ${SW_SLOW_DB_THRESHOLD:default:200,mongodb:
100} # The slow database access thresholds. Unit ms.

    receiver-sharing-server:
      selector: ${SW_RECEIVER_SHARING_SERVER:default}
      default:
        host: ${SW_RECEIVER_JETTY_HOST:0.0.0.0}
        contextPath: ${SW_RECEIVER_JETTY_CONTEXT_PATH:/}
        authentication: ${SW_AUTHENTICATION:""}
        jettyMinThreads: ${SW_RECEIVER_SHARING_JETTY_MIN_THREADS:1}
        jettyMaxThreads: ${SW_RECEIVER_SHARING_JETTY_MAX_THREADS:200}
        jettyIdleTimeOut: ${SW_RECEIVER_SHARING_JETTY_IDLE_TIMEOUT:30000}
        jettyAcceptorPriorityDelta: ${SW_RECEIVER_SHARING_JETTY_DELTA:0}
        jettyAcceptQueueSize: ${SW_RECEIVER_SHARING_JETTY_QUEUE_SIZE:0}
    receiver-register:
      selector: ${SW_RECEIVER_REGISTER:default}
      default:

    receiver-trace:
      selector: ${SW_RECEIVER_TRACE:default}
      default:

    receiver-jvm:
      selector: ${SW_RECEIVER_JVM:default}
      default:

    receiver-clr:
      selector: ${SW_RECEIVER_CLR:default}
      default:

    receiver-profile:
      selector: ${SW_RECEIVER_PROFILE:default}
      default:

    service-mesh:
      selector: ${SW_SERVICE_MESH:default}
      default:

    istio-telemetry:
      selector: ${SW_ISTIO_TELEMETRY:default}
      default:

    envoy-metric:
      selector: ${SW_ENVOY_METRIC:default}
      default:
        acceptMetricsService: ${SW_ENVOY_METRIC_SERVICE:true}
```

```
    alsHTTPAnalysis: ${SW_ENVOY_METRIC_ALS_HTTP_ANALYSIS:""}

prometheus-fetcher:
  selector: ${SW_PROMETHEUS_FETCHER:default}
  default:
    active: ${SW_PROMETHEUS_FETCHER_ACTIVE:false}

kafka-fetcher:
  selector: ${SW_KAFKA_FETCHER:-}
  default:
    bootstrapServers: ${SW_KAFKA_FETCHER_SERVERS:localhost:9092}
    partitions: ${SW_KAFKA_FETCHER_PARTITIONS:3}
    replicationFactor: ${SW_KAFKA_FETCHER_PARTITIONS_FACTOR:2}
    enableMeterSystem: ${SW_KAFKA_FETCHER_ENABLE_METER_SYSTEM:false}
    isSharding: ${SW_KAFKA_FETCHER_IS_SHARDING:false}
    consumePartitions: ${SW_KAFKA_FETCHER_CONSUME_PARTITIONS:""}

receiver-meter:
  selector: ${SW_RECEIVER_METER:-}
  default:

receiver-oc:
  selector: ${SW_OC_RECEIVER:-}
  default:
    gRPCHost: ${SW_OC_RECEIVER_GRPC_HOST:0.0.0.0}
    gRPCPort: ${SW_OC_RECEIVER_GRPC_PORT:55678}

receiver_zipkin:
  selector: ${SW_RECEIVER_ZIPKIN:-}
  default:
    host: ${SW_RECEIVER_ZIPKIN_HOST:0.0.0.0}
    port: ${SW_RECEIVER_ZIPKIN_PORT:9411}
    contextPath: ${SW_RECEIVER_ZIPKIN_CONTEXT_PATH:/}
    jettyMinThreads: ${SW_RECEIVER_ZIPKIN_JETTY_MIN_THREADS:1}
    jettyMaxThreads: ${SW_RECEIVER_ZIPKIN_JETTY_MAX_THREADS:200}
    jettyIdleTimeOut: ${SW_RECEIVER_ZIPKIN_JETTY_IDLE_TIMEOUT:30000}
    jettyAcceptorPriorityDelta: ${SW_RECEIVER_ZIPKIN_JETTY_DELTA:0}
    jettyAcceptQueueSize: ${SW_RECEIVER_ZIPKIN_QUEUE_SIZE:0}

receiver_jaeger:
  selector: ${SW_RECEIVER_JAEGER:-}
  default:
    gRPCHost: ${SW_RECEIVER_JAEGER_HOST:0.0.0.0}
    gRPCPort: ${SW_RECEIVER_JAEGER_PORT:14250}

query:
  selector: ${SW_QUERY:graphql}
  graphql:
    path: ${SW_QUERY_GRAPHQL_PATH:/graphql}

alarm:
  selector: ${SW_ALARM:default}
  default:
```

```
    telemetry:
      selector: ${SW_TELEMETRY:none}
      none:
      prometheus:
        host: ${SW_TELEMETRY_PROMETHEUS_HOST:0.0.0.0}
        port: ${SW_TELEMETRY_PROMETHEUS_PORT:1234}

    configuration:
      selector: ${SW_CONFIGURATION:none}
      none:
      grpc:
        host: ${SW_DCS_SERVER_HOST:""}
        port: ${SW_DCS_SERVER_PORT:80}
        clusterName: ${SW_DCS_CLUSTER_NAME:SkyWalking}
        period: ${SW_DCS_PERIOD:20}
      apollo:
        apolloMeta: ${SW_CONFIG_APOLLO:http://106.12.25.204:8080}
        apolloCluster: ${SW_CONFIG_APOLLO_CLUSTER:default}
        apolloEnv: ${SW_CONFIG_APOLLO_ENV:""}
        appId: ${SW_CONFIG_APOLLO_APP_ID:skywalking}
        period: ${SW_CONFIG_APOLLO_PERIOD:5}
      zookeeper:
        period: ${SW_CONFIG_ZK_PERIOD:60} # Unit seconds, sync period.
Default fetch every 60 seconds.
        nameSpace: ${SW_CONFIG_ZK_NAMESPACE:/default}
        hostPort: ${SW_CONFIG_ZK_HOST_PORT:localhost:2181}
        # Retry Policy
        baseSleepTimeMs: ${SW_CONFIG_ZK_BASE_SLEEP_TIME_MS:1000} # initial
amount of time to wait between retries
        maxRetries: ${SW_CONFIG_ZK_MAX_RETRIES:3} # max number of times to
retry
      etcd:
        period: ${SW_CONFIG_ETCD_PERIOD:60} # Unit seconds, sync period.
Default fetch every 60 seconds.
        group: ${SW_CONFIG_ETCD_GROUP:skywalking}
        serverAddr: ${SW_CONFIG_ETCD_SERVER_ADDR:localhost:2379}
        clusterName: ${SW_CONFIG_ETCD_CLUSTER_NAME:default}
      consul:
        # Consul host and ports, separated by comma, e.g. 1.2.3.4:
8500,2.3.4.5:8500
        hostAndPorts: ${SW_CONFIG_CONSUL_HOST_AND_PORTS:1.2.3.4:8500}
        # Sync period in seconds. Defaults to 60 seconds.
        period: ${SW_CONFIG_CONSUL_PERIOD:60}
        # Consul aclToken
        aclToken: ${SW_CONFIG_CONSUL_ACL_TOKEN:""}
      k8s-configmap:
        period: ${SW_CONFIG_CONFIGMAP_PERIOD:60}
        namespace: ${SW_CLUSTER_K8S_NAMESPACE:default}
        labelSelector: ${SW_CLUSTER_K8S_LABEL:app=collector,
release=skywalking}
      nacos:
        # Nacos Server Host
```

```
        serverAddr: ${SW_CONFIG_NACOS_SERVER_ADDR:127.0.0.1}
        # Nacos Server Port
        port: ${SW_CONFIG_NACOS_SERVER_PORT:8848}
        # Nacos Configuration Group
        group: ${SW_CONFIG_NACOS_SERVER_GROUP:skywalking}
        # Nacos Configuration namespace
        namespace: ${SW_CONFIG_NACOS_SERVER_NAMESPACE:}
        # Unit seconds, sync period. Default fetch every 60 seconds.
        period: ${SW_CONFIG_NACOS_PERIOD:60}

    exporter:
      selector: ${SW_EXPORTER:-}
      grpc:
        targetHost: ${SW_EXPORTER_GRPC_HOST:127.0.0.1}
        targetPort: ${SW_EXPORTER_GRPC_PORT:9870}

    health-checker:
      selector: ${SW_HEALTH_CHECKER:-}
      default:
        checkIntervalSeconds: ${SW_HEALTH_CHECKER_INTERVAL_SECONDS:5}
```

在dc中添加configmap

```
spec.template.spec
      volumes:
        - name: skywalking-oap-config
          configMap:
            name: skywalking-oap-config
            defaultMode: 420




spec.template.spec.containers
          volumeMounts:
            - name: skywalking-oap-config
              mountPath: /skywalking/config/application.yml
              subPath: application.yml
```

保存，并自动rollout

skywalking-ui部署

采用container image部署

镜像名称为： registry.ocp.acca/opra/skywalking-ui:8.1.0

Deploy Image

Image Search

Deploy an existing image from an image registry.

Image Name *

registry.ocp.acca/opra/skywalking-ui:8.1.0

To deploy an image from a private repository, you must create an image pull secret with your image registry credentials.

Image runs as root

Image registry.ocp.acca/opra/skywalking-ui:8.1.0 runs as the root user which might not be permitted by your cluster administrator.

registry.ocp.acca/opra/skywalking-ui:8.1.0 Sep 23, 6:51 pm, 137.2 MiB, 8 layers

- Image Stream skywalking-ui:8.1.0 will track this image.
- This image will be deployed in Deployment Config skywalking-ui.

General

Application

Create Application

Select an application for your grouping or Unassigned to not use an application grouping.

Application Name

skywalking-ui

A unique name given to the application grouping to label your resources.

Name *

skywalking-ui

A unique name given to the component that will be used to name associated resources.

Advanced Options

Create a route to the application

Exposes your application at a public URL

Click on the names to access advanced options for Routing, Deployment Configuration, Scaling, Resource Limits and Labels.

Create Cancel

访问地址： 查看routes中的skywalking-ui

RT skywalking-ui NS ca-opra-dev http://skywalking-ui-ca-opra-dev.apps.ocp.acca

应用添加agent

在应用的Build config中添加jvm参数， 应用名称改为与spring.application.name配置相同

```
-Duser.timezone=GMT+08 -XX:MaxRAMFraction=2
            -javaagent:/opt/app-root/skywalking/agent/skywalking-agent.jar
            -Dskywalking.agent.service_name=
```