

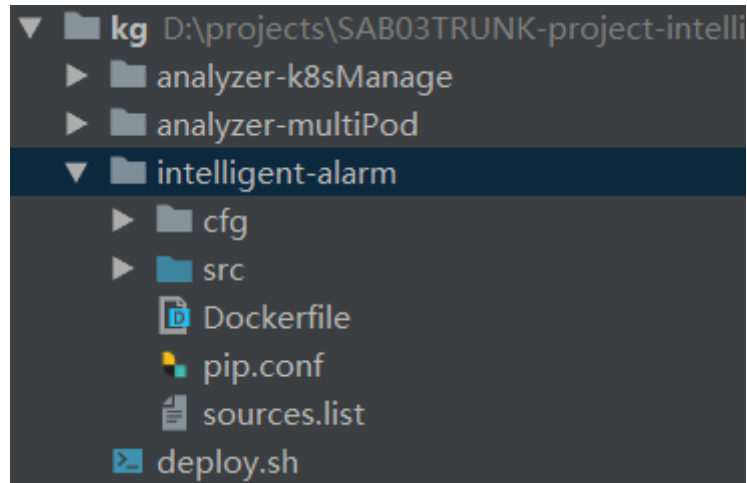
# SA打包流程

以智能告警为例

## Docker镜像创建

Dockerfile 是一个用来构建镜像的文本文件，文本内容包含了一条条构建镜像所需的指令和说明。

1. 在自己的项目路径下编写Dockerfile:



Dockerfile示例：

```
FROM ubuntu:16.04

RUN mkdir /root/.config\
    && mkdir /root/src\
    && mkdir /root/cfg

COPY pip.conf /root/.config/pip/
COPY sources.list /etc/apt/
COPY cfg /root/cfg
COPY src /root/src

RUN apt-get autoclean\
    && apt-get update\
    && apt-get install -y --no-install-recommends\
    build-essential \
    vim\
    python3-dev\
    python3-pip

RUN pip3 install --upgrade pip
RUN pip3 install --upgrade wheel
RUN pip3 --no-cache-dir install "setuptools<50.0.0" --upgrade

ENV PYTHONIOENCODING utf-8
ENV LANG C.UTF-8

RUN apt-get install tzdata -y
```

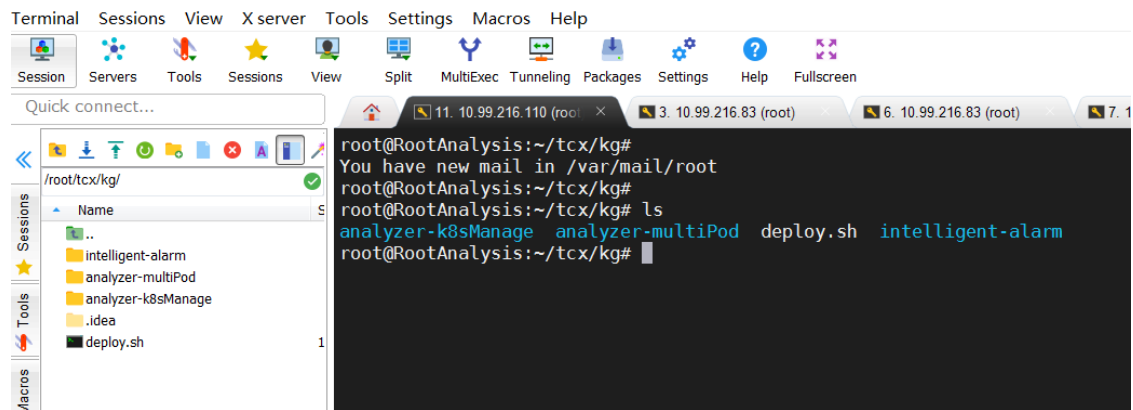
```
RUN cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo  
'Asia/Shanghai'>/etc/timezone
```

```
EXPOSE 6688
```

```
WORKDIR /root/src
```

```
RUN pip3 install -r requirements.txt
```

## 2. 上传最新的项目代码到镜像打包环境 ( 10.99.216.110 )



## 3. 执行镜像构建脚本

```
root@RootAnalysis:~/tcx/kg# sh deploy.sh
```

该脚本中的镜像构建命令如下：

```
docker build --no-cache -t sa/itoe-intelligent-alarm:1.0.0 .
```

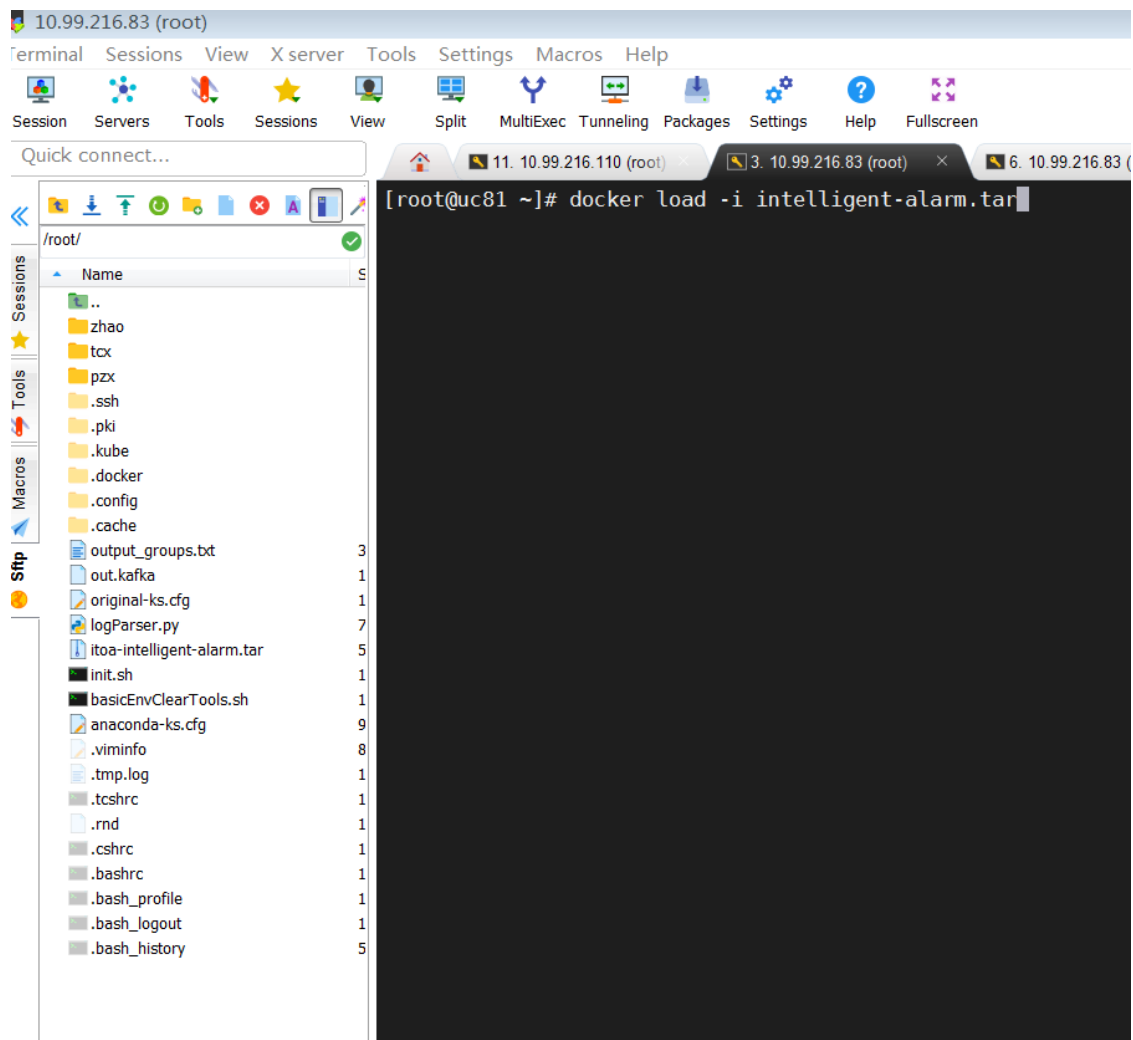
## 4. 构建完毕后保存镜像文件

```
root@RootAnalysis:~/tcx/kg# docker save -o itoe-intelligent-alarm.tar sa/itoe-intelligent-alarm:1.0.0
```

## 5. 上传镜像文件包至SA环境 ( 10.99.216.83 )

```
root@RootAnalysis:~/tcx/kg# scp itoe-intelligent-alarm.tar root@10.99.216.83:/root
```

## 6. 在SA服务器 ( 10.99.216.83 ) 上加载镜像



## Pod部署

Pod代表部署的一个单位：Kubernetes中单个应用的实例，它可能由单个容器或多个容器共享组成的资源。

### 1. 编写pod的yaml配置

Pod的yaml配置文件所在路径：

```
[root@uc81 k8s-resources]# pwd
/opt/matrix/app/install/metadata/SA/analysis/itoa-knowledge-graph/k8s-resources
[root@uc81 k8s-resources]# ls
itoa-intelligent-alarm.yaml itoa-knowledge-graph-manager-service.yaml itoa-knowledge-graph-manager.yaml itoa-knowledge-graph-master-
```

在SA代码中的路径：

```
health/package/itoa-3.0/metadata/analysis/itoa-knowledge-graph/k8s-
resources/itoa-intelligent-alarm.yaml
```

yaml配置文件示例 ( itoa-intelligent-alarm.yaml )，各参数详解可参考<https://www.cnblogs.com/bigberg/p/9203619.html> )：

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: itoa-intelligent-alarm
  namespace: sa
labels:
  app: intelligent-alarm
spec:
  replicas: 1
```

```

selector:
matchLabels:
  app: intelligent-alarm
template:
metadata:
  labels:
    app: intelligent-alarm
spec:
  nodeSelector:
    seeranalyzer-label: seeranalyzer-label
  containers:
  - name: intelligent-alarm
    command: ["sh", "/root/src/start.sh"]
    image: sa/itoe-intelligent-alarm:1.0.0
    imagePullPolicy: IfNotPresent
    resources:
      limits:
        cpu: "5"
      requests:
        cpu: "1"
        # hostPort: 5555
    volumeMounts:
      - name: app-prop-v
        mountPath: /root/cfg/intelligent-alarm.ini
        subPath: re
  volumes:
  - name: log
    emptyDir: {}
  - name: log-diag
    hostPath:
      path: /var/log/matrix-diag/SA
  - name: app-prop-v
    configMap:
      name: web-backend-props #已经创建的configMap的名称
      items:
        - key: itoe-intelligent-alarm.ini
          path: re

```

## 2. 配置文件

其中，configmap存储程序运行所需的外部文件，文件格式没有限制，SA环境上对应的路径：

```

[root@uc01 web-backend-props]# pwd
/opt/matrix/app/install/metadata/SA/scripts/web-backend-props
[root@uc01 web-backend-props]# ls
itoe-action-issue.properties  itoe-data-analyze-trackAnalyze.properties  itoe-int-analysis.properties  itoe-protocol-analysis.properties  itoe-wired-auth.properties
itoe-agent.properties        itoe-data-clean-indexOfEs.properties       itoe-intelligent-alarm.ini    itoe-rule-engine-indexOfEs.properties  itoe-wireless-function.properties
itoe-alops-pa-config.yaml    itoe-data-clean.properties                itoe-log-alert.properties    itoe-rule-engine.properties           itoe-wireless-mybatis.xml
itoe-alops-pd-config.yaml    itoe-data-collection.properties            itoe-log-alert-quartz.properties  itoe-scheduler-center.properties     itoe-wireless-web-server.properties
itoe-alops-task-service-config.json  itoe-data-core.properties                itoe-log-forward.properties  itoe-seer-agent-application.yaml      sa-changes-analysis.properties
itoe-al.properties          itoe-data-report.properties              itoe-logframe.properties      itoe-seer-agent-config.json          sa-data-analysis.properties
itoe-alarm.properties        itoe-data-search-indexOfEs.properties     itoe-logframe-quartz.properties  itoe-seer-agent-redis.conf          sa-user-web.properties
itoe-app-manager.properties  itoe-data-search.properties              itoe-mdpbusiness-application.properties  itoe-smtp.properties              SNA-ITOA-menu.json
itoe-asset.properties        itoe-digitaltwin.properties              itoe-mdpbusiness-es.properties  itoe-ssn-deployerConfigContext.xml  SNA-ITOA-menu-wan-tan.json
itoe-auth-application.properties  itoe-drools.properties                  itoe-mdpbusiness-redis.properties  itoe-subscription.properties        SNA-ITOA-menu-wan-tan.json
itoe-clean-agent.properties    itoe-grafana.ini                        itoe-netconf.properties        itoe-system-alert.properties        update_properties.sh
itoe-config-center.properties  itoe-grpc.properties                   itoe-problem-center.properties  itoe-task-manager.properties        itoe-urule.properties
itoe-data-analyze.properties  itoe-health-analysis.properties          itoe-prometheus.yml            itoe-urule.properties

```

新增配置文件 itoe-intelligent-alarm.ini，内容：

```
[DEFAULT]
timeslot = 60
timeout = 300
group_overlap = 10
low_prior_filter = True
flapping_filter = True
kafka.consumer.servers = itoa-kafka-service1:6667
kafka.consumer.group.id = intelligent-alarm
kafka.consumer.topic = security
url = http://itoa-health-analysis:8080/healthAnalysis/alarmToUcenter/send
```

在SA项目代码中的路径为

```
health/package/itoa-3.0/metadata/scripts/web-backend-props/itoa-intelligent-
alarm.ini
```

对于新增的配置文件，需执行目录下的 `update_properties.sh` 来使之生效，更新k8s的 configmap

```
[root@uc81 web-backend-props]# sh update_properties.sh
```

### 3. 根据yaml配置创建pod

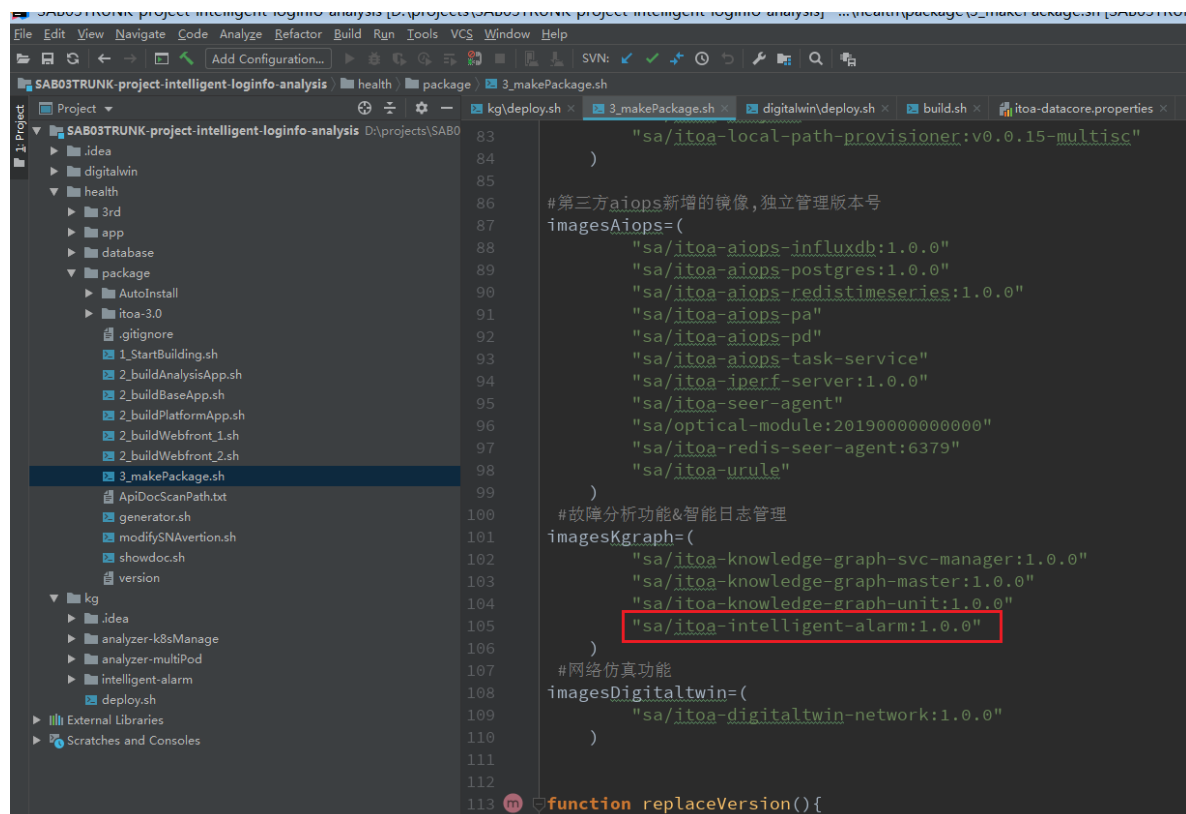
进入yaml文件所在路径，执行如下命令创建pod

```
[root@uc81 k8s-resources]# pwd
/opt/matrix/app/install/metadata/SA/analysis/itoa-knowledge-graph/k8s-resources
[root@uc81 k8s-resources]# kubectl create -f itoa-intelligent-alarm.yaml
```

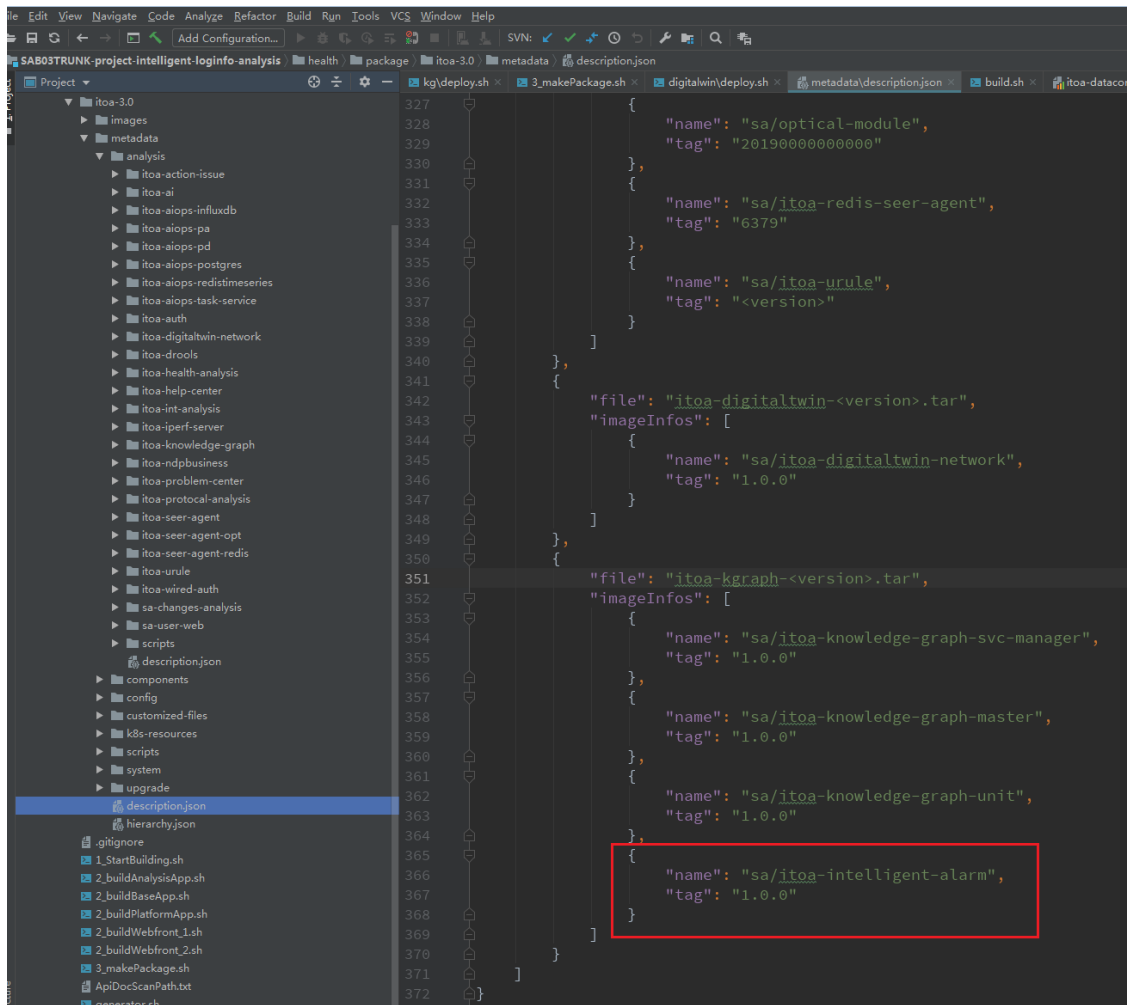
至此可以完成pod的创建，运行项目代码

以上是手动部署pod的流程，想要实现jenkins构建时在代码中自动打包部署，还需要更新相应的构建脚本

#### 1. health/package/3\_makePackage.sh

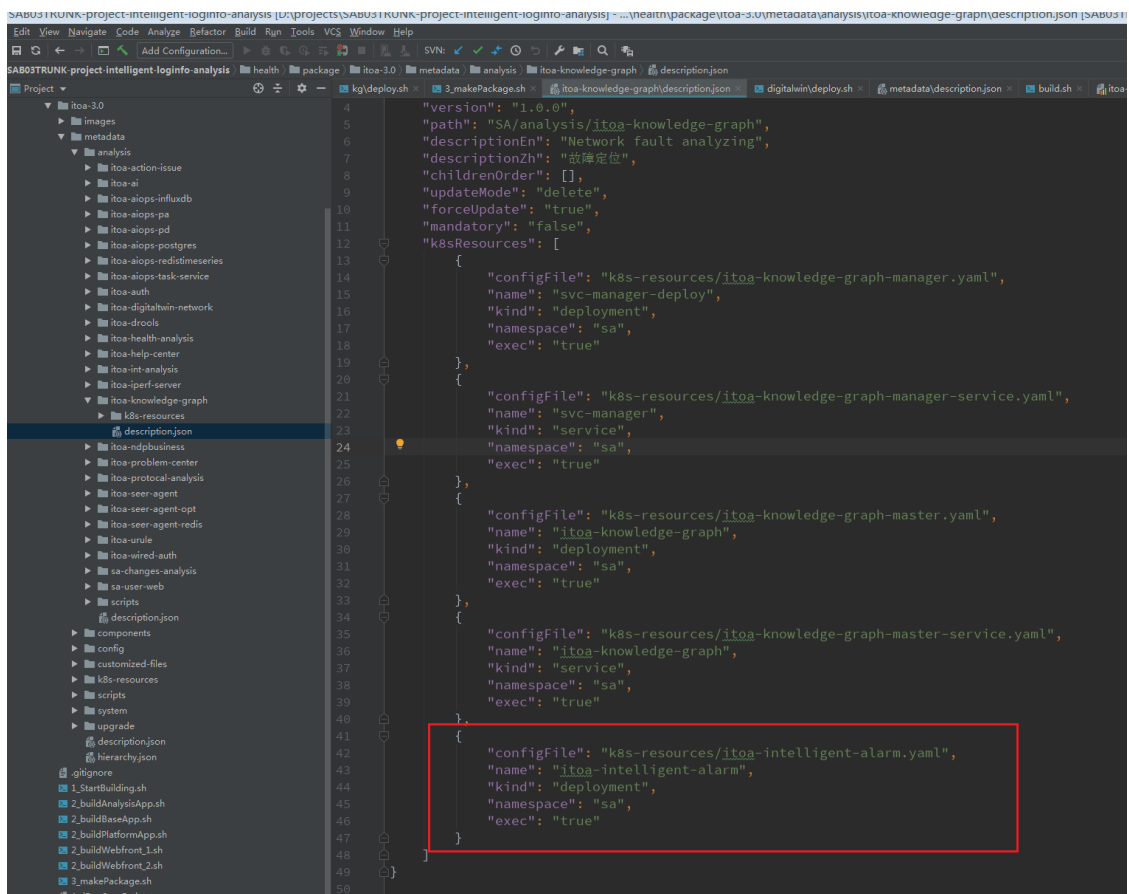


## 2. health/package/itoe-3.0/metadata/description.json



```
327 {
328   "name": "sa/optical-module",
329   "tag": "20190000000000"
330 },
331 {
332   "name": "sa/itoe-redis-seer-agent",
333   "tag": "6379"
334 },
335 {
336   "name": "sa/itoe-urule",
337   "tag": "<version>"
338 }
339 ],
340 {
341   "file": "itoe-digitaltwin-<version>.tar",
342   "imageInfos": [
343     {
344       "name": "sa/itoe-digitaltwin-network",
345       "tag": "1.0.0"
346     }
347   ]
348 },
349 {
350   "file": "itoe-kgraph-<version>.tar",
351   "imageInfos": [
352     {
353       "name": "sa/itoe-knowledge-graph-svc-manager",
354       "tag": "1.0.0"
355     },
356     {
357       "name": "sa/itoe-knowledge-graph-master",
358       "tag": "1.0.0"
359     },
360     {
361       "name": "sa/itoe-knowledge-graph-unit",
362       "tag": "1.0.0"
363     },
364     {
365       "name": "sa/itoe-intelligent-alarm",
366       "tag": "1.0.0"
367     }
368   ]
369 }
370 ]
371 }
372 }
```

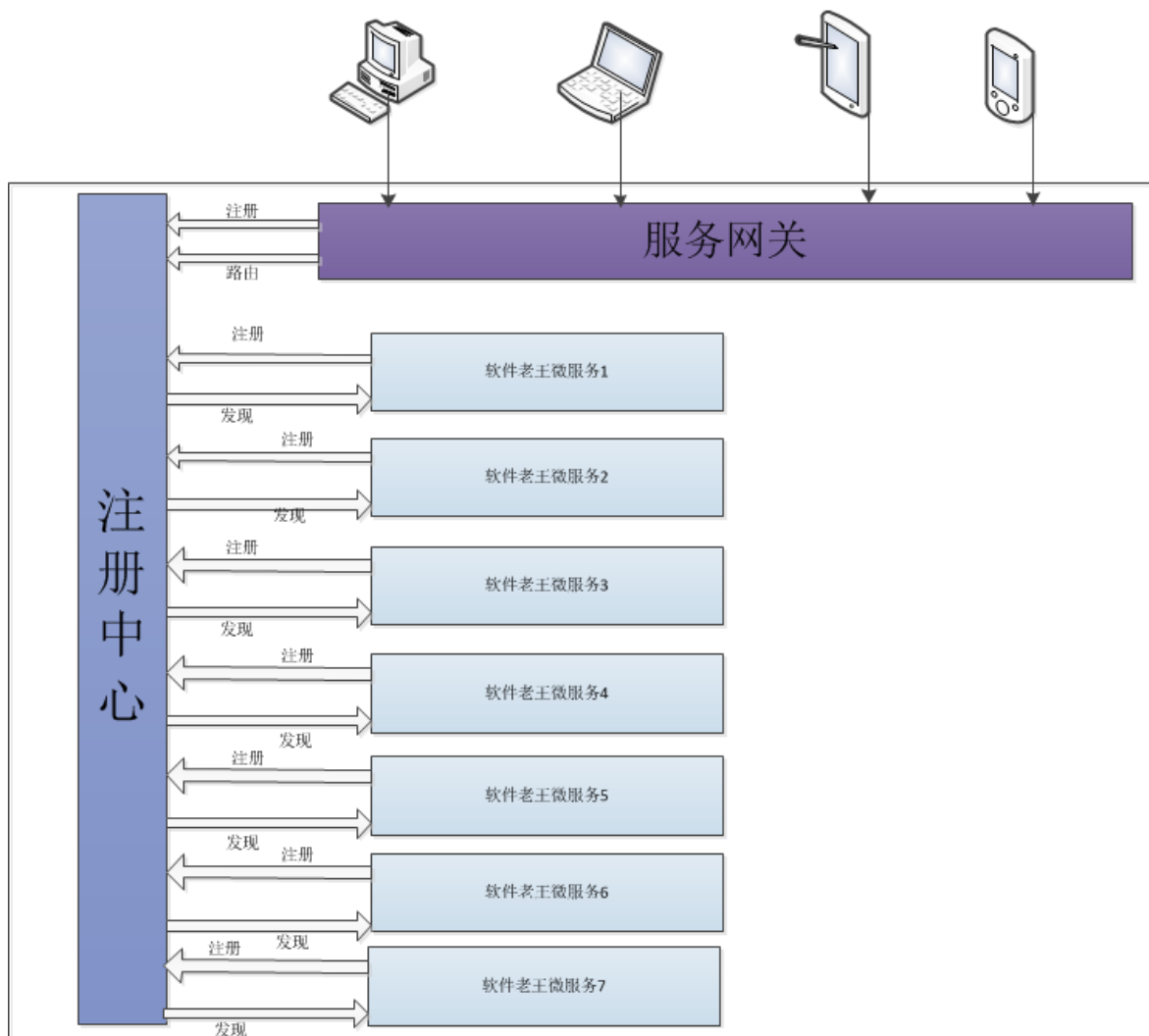
## 3. health/package/itoe-3.0/metadata/analysis/itoe-knowledge-graph/description.json



```
4 "version": "1.0.0",
5 "path": "SA/analysis/itoe-knowledge-graph",
6 "descriptionEn": "Network fault analyzing",
7 "descriptionZh": "故障定位",
8 "childrenOrder": [],
9 "updateMode": "delete",
10 "forceUpdate": "true",
11 "mandatory": "false",
12 "k8sResources": [
13   {
14     "configFile": "k8s-resources/itoe-knowledge-graph-manager.yaml",
15     "name": "svc-manager-deploy",
16     "kind": "deployment",
17     "namespace": "sa",
18     "exec": "true"
19   },
20   {
21     "configFile": "k8s-resources/itoe-knowledge-graph-manager-service.yaml",
22     "name": "svc-manager",
23     "kind": "service",
24     "namespace": "sa",
25     "exec": "true"
26   },
27   {
28     "configFile": "k8s-resources/itoe-knowledge-graph-master.yaml",
29     "name": "itoe-knowledge-graph",
30     "kind": "deployment",
31     "namespace": "sa",
32     "exec": "true"
33   },
34   {
35     "configFile": "k8s-resources/itoe-knowledge-graph-master-service.yaml",
36     "name": "itoe-knowledge-graph",
37     "kind": "service",
38     "namespace": "sa",
39     "exec": "true"
40   },
41   {
42     "configFile": "k8s-resources/itoe-intelligent-alarm.yaml",
43     "name": "itoe-intelligent-alarm",
44     "kind": "deployment",
45     "namespace": "sa",
46     "exec": "true"
47   }
48 ]
49 }
50 }
```

## 拓展：增加网关

（智能告警目前不涉及对外接口，因此以分支站点模块举例）



（一）所有应用或者服务要想对外提供服务（包括网关），必须首先到注册中心进行注册。

（二）所有访问通过服务网关进行访问，然后由服务网关路由到对应服务中心进行交互访问。

以SA上的JAVA代码数据采集SiteAnalysis模块举例：

1. 在eureka注册中心注册自己模块的服务：

SA代码路径 `health/package/itoea-3.0/metadata/scripts/web-backend-props/itoea-site-analysis.properties`

```
spring.application.name=SiteAnalysis

eureka.client.service-url.defaultZone=http://itoea-eureka:8080/eureka
eureka.instance.prefer-ip-address=true
eureka.client.healthcheck.enable=true
#eureka.instance.instanceId=${spring.application.name}:Maiintn
eureka.instance.hostname=itoea-site-analysis
```

2. 在zuul网关配置中增加相应路由:

SA代码路径 `health/package/itoea-3.0/metadata/scripts/web-backend-props/itoea-dacore.properties`

```
#ZUUL SiteAnalysis
zuul.SiteAnalysis.path=/SiteAnalysis/**
zuul.routes.SiteAnalysis.stripPrefix=false
zuul.routes.SiteAnalysis.serviceId=SiteAnalysis
```

至此，便可以在外部调用SA上自己开发的SiteAnalysis模块相关接口

## 添加子页签

在应用分析下面添加子页签：

metadata\scripts\register-ucenter\menu\sdwan\sa\_sdwan.json——sdwan页签配置文件

```
{
  "id": "wan.analysis.health.application.applicationanalysis",
  "url":
"/app/itoea/#/healthAnalysis/appStreamAnalysis/siteAppPerformance/applicationPerf
ormance",
  "nameEn": "Application Group Flow Analysis",
  "nameZh": "应用流分析",
  "priority": 11,
  "appName": "ucenter-analyzer",
  "parentId": "wan.analysis.health.application",
  "productName": "AD-WAN"
},
```

metadata/scripts/register-ucenter/menu/sdwan/single\_sdwan.json——单机版sdwan页签配置文件

```
{
  "id": "singlewan.analysis.health.application.applicationanalysis",
  "url":
"/app/itoea/#/healthAnalysis/appStreamAnalysis/siteAppPerformance/applicationPerf
ormance",
  "nameEn": "Application Group Flow Analysis",
  "nameZh": "应用流分析",
  "priority": 11,
  "appName": "sa-sdwan",
  "parentId": "singlewan.analysis.health.application",
  "productName": "SA-WAN"
}
```

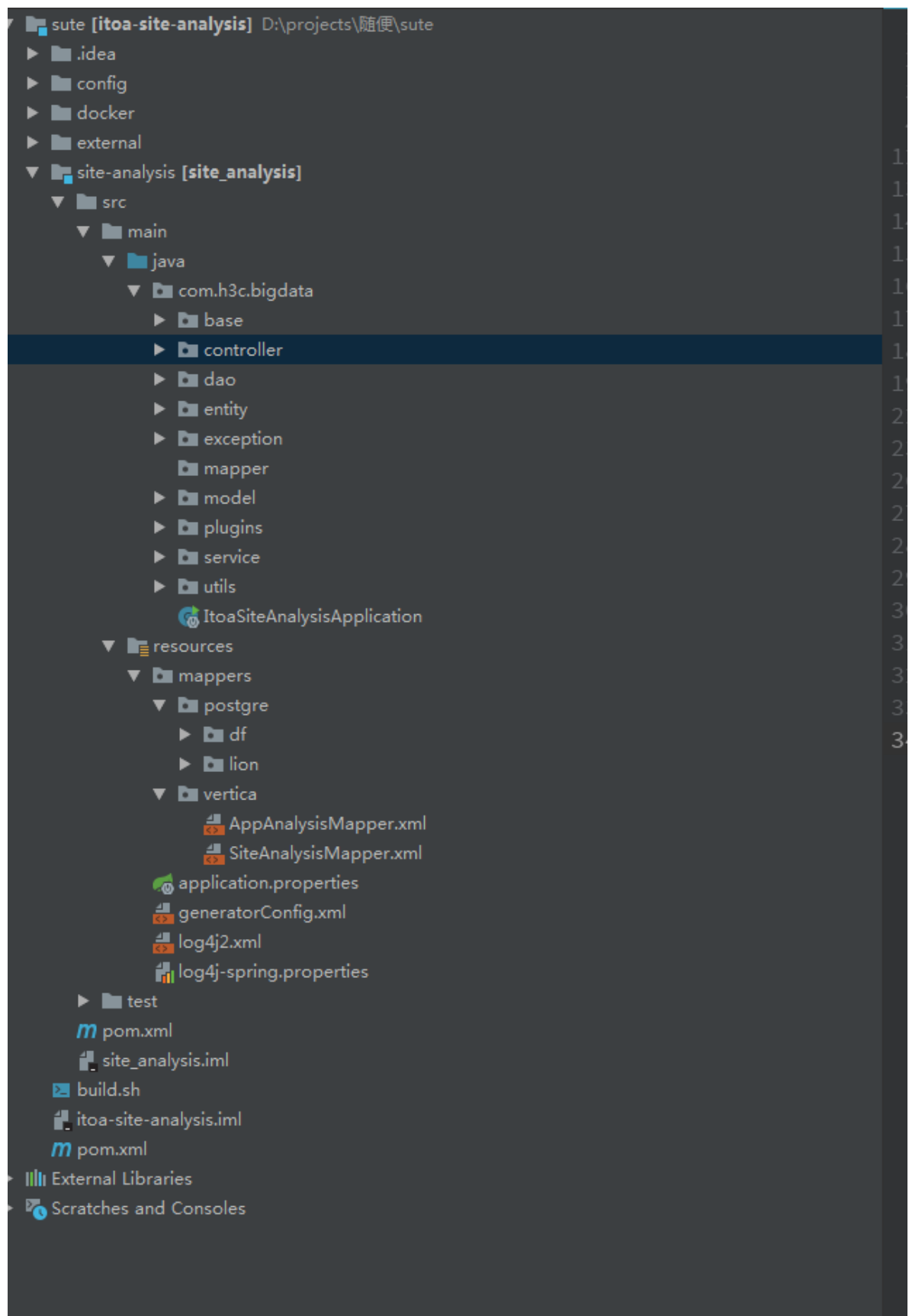
metadata/scripts/register-ucenter/menu/universe/universe\_sdwan.json —— 全域场景页签配置文件



```
{
  "id": "ucenter.analysis.health.application.applicationanalysis",
  "url":
"/app/itoea/#/healthAnalysis/appStreamAnalysis/siteAppPerformance/applicationPerf
ormance",
  "nameEn": "Application Group Flow Analysis",
  "nameZh": "应用流分析",
  "priority": 51,
  "appName": "ucenter-analyzer",
  "parentId": "ucenter.analysis.health.application",
  "productName": "Ucenter",
  "menuType": "universe"
},
```

## Spring目录结构

---



bigdata :

base: 基础公共类

controller:视图层, 对外提供接口, 提供接口路径, 接收前端参数。——1

service: controller接收参数之后，调用service接口，进行运算或者逻辑处理。——2

dao: service需要查数据库是，提供数据库接口。——3

entity / model : 自定义实体类

exception : 自定义异常

plugins : 插件, 例如数据库的启动类, 在spring启动时, 会扫描并自动注入, 完成连接。

utils: 公共工具类

resources : 配置文件, 包括xml, properties等

application.properties : spring项目配置文件

log4j2.\* : 日志配置文件, 包含日志输出类型, 文件位置等

mapper: sql语句。

test : 测试用例