RuoYi-Cloud 微服务项目全链路监控实战（下）

前言：

实验环境：

课程名称：RuoYi-Cloud 微服务项目全链路监控实战（下）

服务安装在 k8s 集群，k8s 环境如下：

角色	IP	主机名	组件	硬件
控制节点	192.168.128.11	k8s-master01	apiserver controller-manager scheduler etcd docker	CPU：4vCPU 硬盘：100G 内存：6GB 开启虚拟化
工作节点	192.168.128.21	k8s-node01	kubelet kube-proxy docker calico coredns	CPU：8vCPU 硬盘：100G 内存：16GB 开启虚拟化

张岩峰老师微信，加我微信，邀请你加入 VIP 交流答疑群：

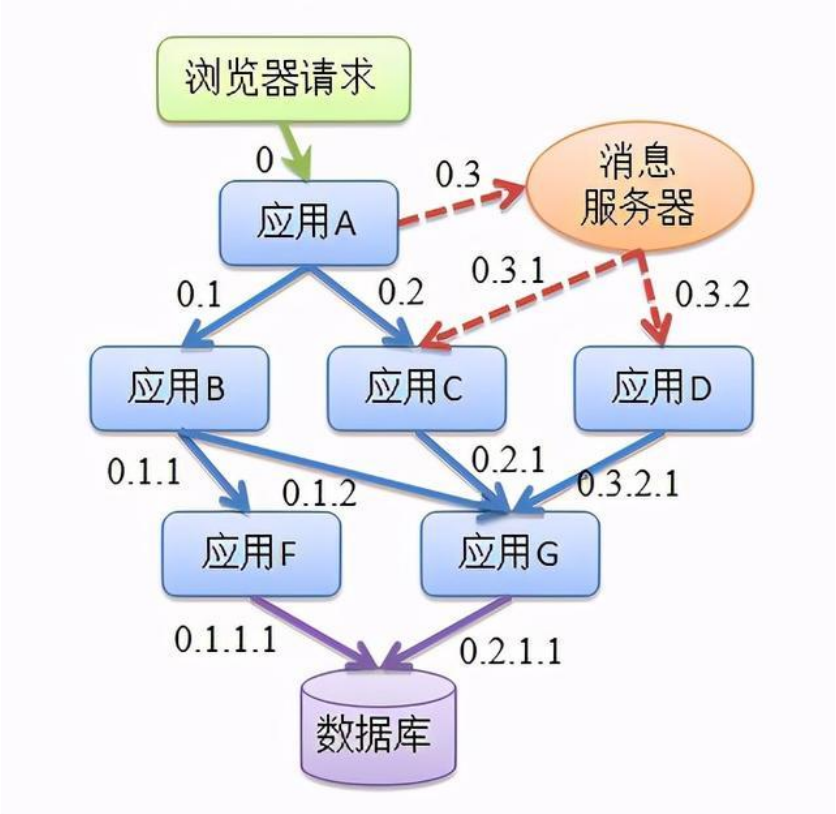
微信号：ZhangYanFeng0429

二维码：

1、链路监控介绍

1.1、什么是全链路监控？

在分布式微服务架构中，系统为了接收并处理一个前端用户请求，需要让多个微服务应用协同工作，其中的每一个微服务应用都可以用不同的编程语言构建，由不同的团队开发，并可以通过多个对等的应用实例实现水平扩展，甚至分布在横跨多个数据中心的数千台服务器上。单个用户请求会引发不同应用之间产生一串顺序性的调用关系，如果要对这些调用关系进行监控，了解每个应用如何调用，这就产生了全链路监控。



1.2、为什么要进行全链路监控？

在微服务架构中，服务会被拆分成多个模块，这些模块可能由不同的开发团队开发、维护，也可能使用不同的编程语言来实现、也有可能分布在多台服务器上，由于服务的拆分，单个用户的请求会经过多个微服务，相互之间形成复杂的调用关系，传统的监控手段已经不能实现如此复杂的链路之间的监控了，因此，就需要一些可以帮助理解系统行为、用于分析性能问题的工具，以便发生故障的时候，能够快速定位和解决问题。

1.3、全链路监控能解决哪些问题？

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

1、请求链路追踪，故障快速定位：可以通过调用链结合业务日志快速定位错误信息。
2、可视化：各个阶段耗时，进行性能分析。
3、依赖优化：各个调用环节的可用性、梳理服务依赖关系以及优化。
4、数据分析，优化链路：可以得到用户的行为路径，汇总分析应用在很多业务场景。

2、初始 Skywalking

github 地址：https://github.com/apache/incubator-skywalking
Skywalking 官网地址：https://skywalking.apache.org/

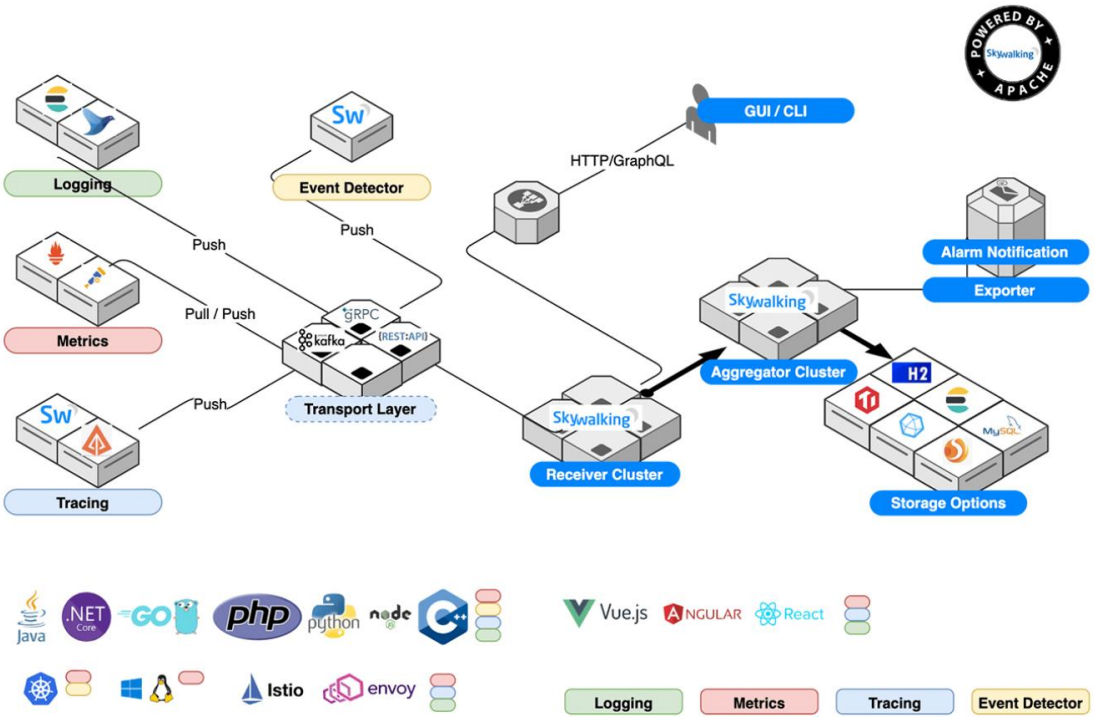
2.1、Skywalking 是什么

skywalking 是本土开源的调用链追踪系统，包括监控、跟踪、诊断功能，目前已加入 Apache 孵化器，专门为微服务、云本地和基于容器（Docker、Kubernetes、Mesos）架构设计。

主要功能如下：

1、服务、服务实例、端点指标数据分析
2、根本原因分析，在运行时评测代码
3、服务拓扑图分析
4、服务、服务实例和端点依赖性分析
5、检测到慢速服务和终结点
6、性能优化
7、分布式跟踪和上下文传播
8、数据库访问度量。检测慢速数据库访问语句（包括 SQL 语句）。
9、报警
10、浏览器性能监视

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。



2.2、Skywalking 核心组件

SkyWalking 采用组件式开发，易于扩展，主要组件作用如下：

1、Skywalking Agent：

链路数据采集 tracing（调用链数据）和 metric（指标）信息并上报，上报通过 HTTP 或者 gRPC 方式发送数据到 Skywalking OAP。

2、Skywalking OAP：

链路数据收集器，对 agent 传过来的 tracing 和 metric 数据进行整合分析通过 Analysis Core 模块处理并落入相关的数据存储中，同时会通过 Query Core 模块进行二次统计和监控告警。

3、Storage：

Skywalking 的存储，支持以 ElasticSearch、Mysql、TiDB、H2 等主流存储作为存储介质进行数据存储,H2 仅作为临时演示单机用。

4、SkyWalking UI：

Web 可视化平台，用来展示落地的数据，目前官方采纳了 RocketBot 作为 SkyWalking 的主 UI。

3、基于 Kubernetes 部署 Skywalking 服务端

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

3.1、准备 elasticsearch 环境

Agent 端会采集数据上报到 Skywalking OAP，如果数据不做持久化存储的话，服务重启，那么采集汇报上来的数据就会丢失。

Skywalking OAP 默认使用的是 H2 内置的内存型数据库。这里我们部署 ES 集群，指定 Skywalking OAP 存储数据的 ES 集群内。

（1）创建 skywalking 命名空间

```
[root@k8s-master01 ~]# kubectl create ns skywalking
namespace/skywalking created
```

（2）创建 elasticsearch handless

```
1、创建资源清单存放位置
[root@k8s-master01 ~]# mkdir /ruoyi/elasticsearch
[root@k8s-master01 ~]# cd /ruoyi/elasticsearch/

2、创建资源清单文件
[root@k8s-master01 elasticsearch]# vi elasticsearch-svc.yaml
kind: Service
apiVersion: v1
metadata:
  name: elasticsearch
  namespace: skywalking
  labels:
    app: elasticsearch
spec:
  selector:
    app: elasticsearch
  clusterIP: None
  ports:
    - name: tcp-9200
      port: 9200
    - name: tcp-9300
      port: 9300

2、更新资源清单文件
[root@k8s-master01 elasticsearch]# kubectl apply -f elasticsearch-svc.yaml
service/elasticsearch created
```

（3）使用 statefulset 资源部署 ES

```
1、创建资源清单文件
[root@k8s-master01 elasticsearch]# vi elasticsearch-statefulset.yaml
```

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: es-cluster
  namespace: skywalking
spec:
  serviceName: elasticsearch
  replicas: 3
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      initContainers:
      - name: fix-permissions
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ["sh", "-c", "chown -R 1000:1000 /usr/share/elasticsearch/data"]
        securityContext:
          privileged: true
        volumeMounts:
        - name: data
          mountPath: /usr/share/elasticsearch/data
      - name: increase-vm-max-map
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ["sysctl", "-w", "vm.max_map_count=262144"]
        securityContext:
          privileged: true
      - name: increase-fd-ulimit
        image: busybox
        imagePullPolicy: IfNotPresent
        command: ["sh", "-c", "ulimit -n 65536"]
        securityContext:
          privileged: true
      containers:
      - name: elasticsearch
        image: elasticsearch:7.17.8
        imagePullPolicy: IfNotPresent
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
          resources:
              limits:
                cpu: 1000m
              requests:
                cpu: 100m
          ports:
          - containerPort: 9200
            name: tcp-9200
            protocol: TCP
          - containerPort: 9300
            name: tcp-9300
            protocol: TCP
          volumeMounts:
          - name: data
            mountPath: /usr/share/elasticsearch/data
          env:
            - name: cluster.name
              value: k8s-logs
            - name: node.name
              valueFrom:
                fieldRef:
                  fieldPath: metadata.name
            - name: discovery.seed_hosts
              value:
"es-cluster-0.elasticsearch.skywalking.svc.cluster.local,es-cluster-1.elasticsearch.skywalki
ng.svc.cluster.local,es-cluster-2.elasticsearch.skywalking.svc.cluster.local"
            - name: cluster.initial_master_nodes
              value: "es-cluster-0,es-cluster-1,es-cluster-2"
            - name: ES_JAVA_OPTS
              value: "-Xms512m -Xmx512m"
    volumeClaimTemplates:
    - metadata:
        name: data
        labels:
          app: elasticsearch
      spec:
        accessModes: [ "ReadWriteOnce" ]
        storageClassName: nfs-storage
        resources:
          requests:
            storage: 20Gi
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

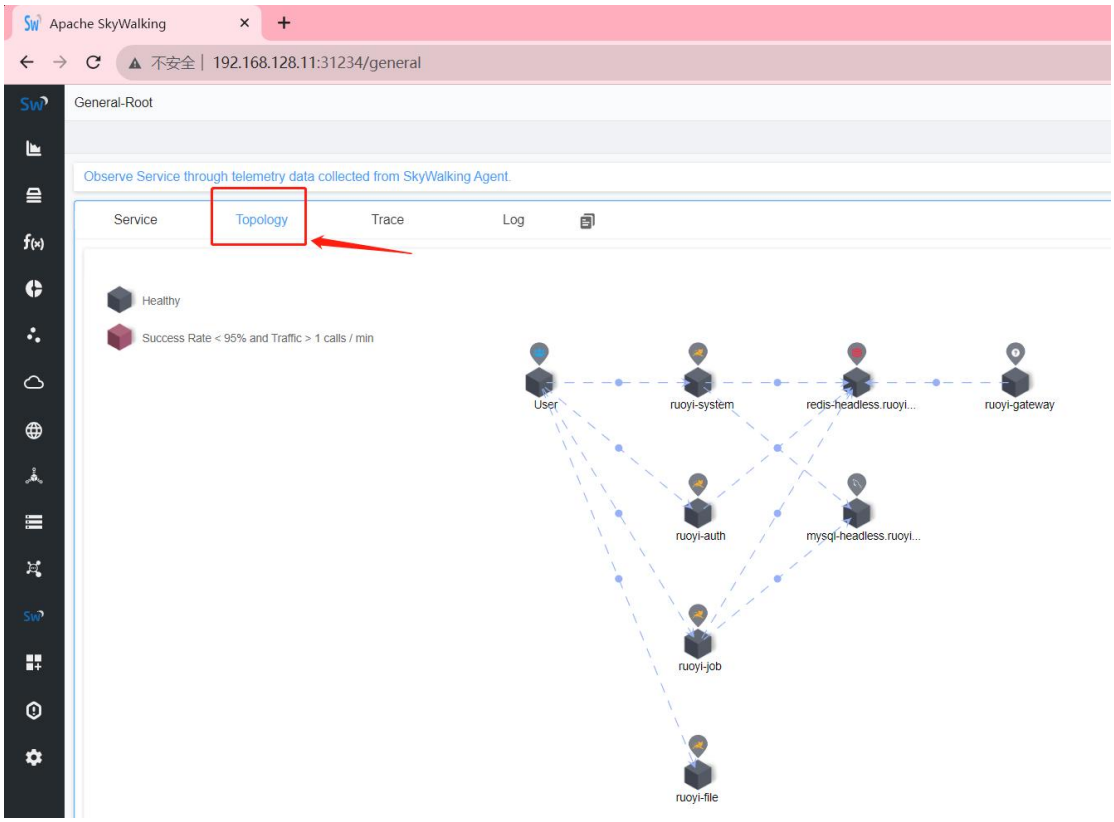
```
2、更新资源清单文件
    [root@k8s-master01 elasticsearch]# kubectl apply -f elasticsearch-statefulset.yaml
    statefulset.apps/es-cluster created

3、查看 pod
    [root@k8s-master01 elasticsearch]# kubectl get pods -n skywalking
    NAME           READY   STATUS    RESTARTS   AGE
    es-cluster-0   1/1     Running   0          2m58s
    es-cluster-1   1/1     Running   0          24s
    es-cluster-2   1/1     Running   0          15s
```

3.2、基于 Kubernetes 部署 Skywalking

这里部署的是 skywalking 9.5 最新版本。Skywalking 在 8.5 版本之后，ui 界面就出现了一些变化，增加了很多新功能，但其实很多新功能我们并用不到，我们仅需要能够监控到服务之间的链路调用即可。如果想要老版本的 UI 可以部署 8.5 版本。

老版本 UI：

新版本 UI：



（1）使用 deployment 控制器部署 skywalking-oap

```
1、创建资源清单存放位置
    [root@k8s-master01 ~]# mkdir /ruoyi/skywalking
    [root@k8s-master01 ~]# cd /ruoyi/skywalking

2、创建资源清单文件
    [root@k8s-master01 skywalking]# vi skywalking-oap-deployment.yaml
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: skywalking-oap
      namespace: skywalking
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: skywalking-oap
      template:
        metadata:
          labels:
            app: skywalking-oap
```

```
      spec:
        containers:
        - name: oap
          image: apache/skywalking-oap-server:9.5.0
          imagePullPolicy: IfNotPresent
          ports:
          - name: grpc
            containerPort: 11800
          - name: rest
            containerPort: 12800
          env:
          - name: SW_STORAGE
            value: "elasticsearch"
          - name: SW_STORAGE_ES_CLUSTER_NODES
            value: "elasticsearch.skywalking.svc.cluster.local:9200"
          readinessProbe:
            tcpSocket:
              port: 12800
            initialDelaySeconds: 30
            periodSeconds: 10
          livenessProbe:
            tcpSocket:
              port: 12800
            initialDelaySeconds: 30
            periodSeconds: 10
          volumeMounts:
          - name: localtime
            mountPath: /etc/localtime
        volumes:
          - name: localtime
            hostPath:
             path: /usr/share/zoneinfo/Asia/Shanghai

3、更新资源清单文件
   [root@k8s-master01 skywalking]# kubectl apply -f
skywalking-oap-deployment.yaml
   deployment.apps/skywalking-oap created
```

（2）创建 skywalking-oap svc

```
1、创建资源清单文件
   [root@k8s-master01 skywalking]# vi skywalking-oap-headless-svc.yaml
   apiVersion: v1
```

```
    kind: Service
    metadata:
      name: skywalking-oap-headless
      namespace: skywalking
    spec:
      ports:
      - port: 12800
        name: rest
      - port: 11800
        name: grpc
      selector:
        app: skywalking-oap

2、更新资源清单文件
    [root@k8s-master01 skywalking]# kubectl apply -f
skywalking-oap-headless-svc.yaml
    service/skywalking-oap-headless created
```

（3）使用 deployment 控制器部署 skywalking-ui

```
1、创建资源清单文件
    [root@k8s-master01 skywalking]# vi skywalking-ui-deployment.yaml
    apiVersion: apps/v1
    kind: Deployment
    metadata:
      name: skywalking-ui
      namespace: skywalking
      labels:
        app: skywalking-ui
    spec:
      replicas: 1
      selector:
        matchLabels:
          app: skywalking-ui
      template:
        metadata:
          labels:
            app: skywalking-ui
        spec:
          containers:
          - name: ui
            image: apache/skywalking-ui:9.5.0
            ports:
```

```
            - name: page
              containerPort: 8080
            env:
            - name: SW_OAP_ADDRESS
              value:
http://skywalking-oap-headless.skywalking.svc.cluster.local:12800
            readinessProbe:
              tcpSocket:
                port: 8080
              initialDelaySeconds: 30
              periodSeconds: 10
            livenessProbe:
              tcpSocket:
                port: 8080
              initialDelaySeconds: 30
              periodSeconds: 10
            volumeMounts:
            - name: localtime
              mountPath: /etc/localtime
          volumes:
            - name: localtime
              hostPath:
                path: /usr/share/zoneinfo/Asia/Shanghai

2、更新资源清单文件
    [root@k8s-master01        skywalking]#        kubectl        apply        -f
skywalking-ui-deployment.yaml
    deployment.apps/skywalking-ui created
```

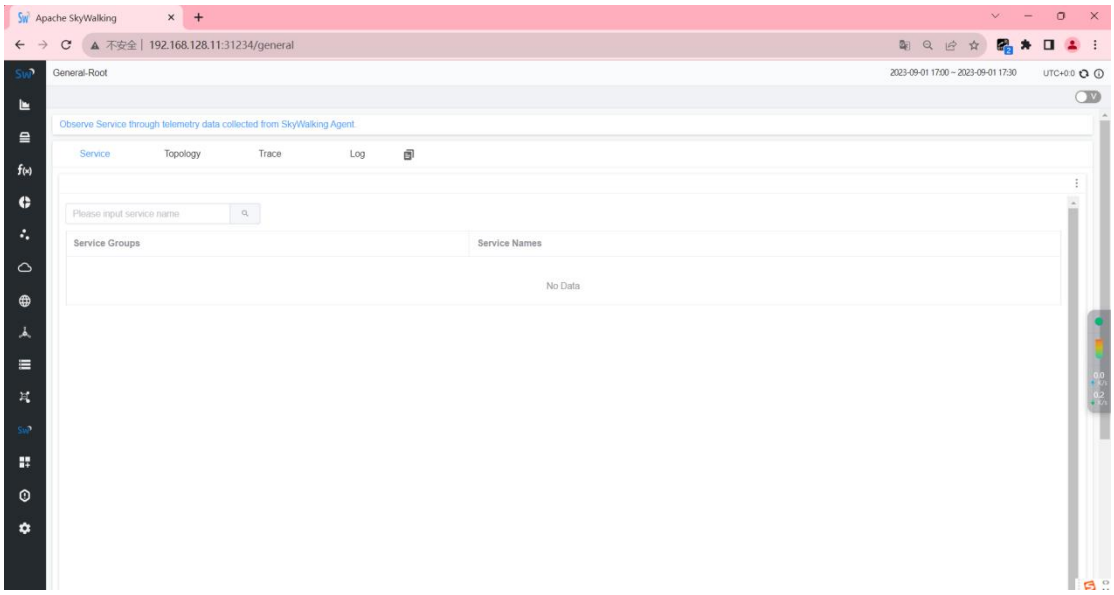
（4）创建 skywalking-ui svc

```
1、创建资源清单文件
    [root@k8s-master01 skywalking]# vi skywalking-ui-external-svc.yaml
    apiVersion: v1
    kind: Service
    metadata:
      name: skywalking-ui-external
      namespace: skywalking
    spec:
      ports:
      - port: 8080
        name: page
        nodePort: 31234
```

```
      type: NodePort
      selector:
        app: skywalking-ui

2、更新资源清单文件
    [root@k8s-master01        skywalking]#        kubectl        apply        -f
skywalking-ui-external-svc.yaml
    service/skywalking-ui-external created
```

（5）浏览器访问

浏览器访问：“http://192.168.128.11:31234”，访问结果如下：



4、集成 Skywalking agent 到微服务组件

想要将 skywalking agent 集成到微服务内，有两种方式：

1、以边车代理方式，使用初始化容器，将 agent 拷贝到业务 containerd 内，再传递参数调用 agent 程序即可。

2、在构建镜像时，直接将 agent 做到镜像内。

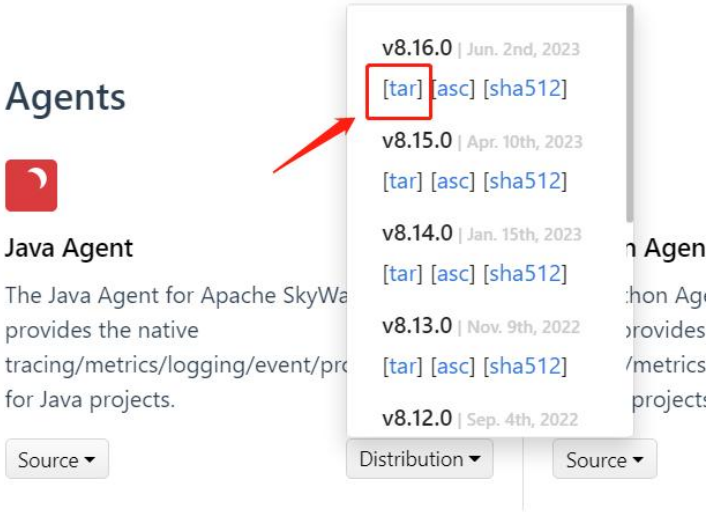
本小节主要讲解第 2 种方法。

4.1、下载 Skywalking agent

下载地址：https://skywalking.apache.org/downloads/

下载 Java Agent，这里下载的是 v8.16.0，如下图：

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。



（1）将下载好的软件包上传到 k8s-master01 节点：

```
[root@k8s-master01 ~]# ll apache-skywalking-java-agent-8.16.0.tgz
-rw-r--r-- 1 root root 31794715 Sep  1 10:24 apache-skywalking-java-agent-8.16.0.tgz
```

（2）解压、拷贝解压后内容到各个/ruoyi/java/ruoyi-xxx/文件夹下

```
1、解压
    [root@k8s-master01 ~]# tar xf apache-skywalking-java-agent-8.16.0.tgz

2、拷贝解压后内容到各个 ruoyi 微服务组件文件夹内
    [root@k8s-master01 ~]# for i in `ls /ruoyi/java/`; do cp -r skywalking-agent /ruoyi/java/${i}/agent ;done

3、检查，确认拷贝成功
    [root@k8s-master01 ~]# for i in `ls /ruoyi/java/`; do ll -d /ruoyi/java/${i}/agent ;done
    drwxr-xr-x 10 root root 221 Sep  1 17:35 /ruoyi/java/ruoyi-auth/agent
    drwxr-xr-x 10 root root 221 Sep  1 17:35 /ruoyi/java/ruoyi-file/agent
    drwxr-xr-x 10 root root 221 Sep  1 17:35 /ruoyi/java/ruoyi-gateway/agent
    drwxr-xr-x 10 root root 221 Sep  1 17:35 /ruoyi/java/ruoyi-job/agent
    drwxr-xr-x 10 root root 221 Sep  1 17:35 /ruoyi/java/ruoyi-monitor/agent
    drwxr-xr-x 10 root root 221 Sep  1 17:35 /ruoyi/java/ruoyi-system/agent
```

4.2、部署带 Skywalking agent 的 ruoyi-gateway 服务

（1）修改 Dockerfile 文件

```
[root@k8s-master01 ~]# cd /ruoyi/java/ruoyi-gateway/
```

```
[root@k8s-master01 ruoyi-gateway]# vi Dockerfile
# 指定基础镜像
FROM openjdk:8-jdk
# 配置环境变量
ENV PARAMS="--server.port=8080 \
--spring.profiles.active=prod \
--spring.cloud.nacos.discovery.server-addr=nacos-headless.ruoyi.svc.cluster.local:8848 \
--spring.cloud.nacos.config.server-addr=nacos-headless.ruoyi.svc.cluster.local:8848 \
--spring.cloud.nacos.config.namespace=prod \
--spring.cloud.nacos.config.file-extension=yml"
ENV AGENTS="-javaagent:agent/skywalking-agent.jar \
-Dskywalking.agent.service_name=ruoyi-gateway \
-Dskywalking.collector.backend_service=skywalking-oap-headless.skywalking.svc.cluster.local:11800"
# 同步时间
RUN /bin/cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo 'Asia/Shanghai' >/etc/timezone
# 创建文件夹
RUN mkdir -p /home/ruoyi
# 指定工作目录
WORKDIR /home/ruoyi
# 拷贝 jar 包
COPY ruoyi-gateway.jar /home/ruoyi/app.jar
# 拷贝 agent
COPY agent /home/ruoyi/agent
# 暴露端口
EXPOSE 8080
# 运行 jar 包
ENTRYPOINT ["/bin/sh","-c","java ${AGENTS} -Dfile.encoding=utf8 -jar app.jar ${PARAMS}"]
```

对新添加的参数说明如下：

```
-javaagent:：指定 skywalking agent jar 包
-Dskywalking.agent.service_name=：指定 agent 名称，在 UI 显示的名称
-Dskywalking.collector.backend_service=：指定 skywalking OAP 地址
```

（2）构建镜像

```
[root@k8s-master01 ruoyi-gateway]# docker build -t 192.168.128.11/ruoyi/ruoyi-gateway:v2 .
```

（3）推送镜像

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    [root@k8s-master01          ruoyi-gateway]#          docker          push
192.168.128.11/ruoyi/ruoyi-gateway:v2
```

（4）更新服务

```
1、替换版本
    [root@k8s-master01      ruoyi-gateway]#      sed      -i      "s#:v1#:v2#g"
ruoyi-gateway-deployment.yaml

2、更新服务
    [root@k8s-master01       ruoyi-gateway]#       kubectl       apply       -f
ruoyi-gateway-deployment.yaml
    deployment.apps/ruoyi-gateway configured
```

4.3、部署带 Skywalking agent 的 ruoyi-system 服务

（1）修改 Dockerfile 文件

```
    [root@k8s-master01 ~]# cd /ruoyi/java/ruoyi-system/
    [root@k8s-master01 ruoyi-system]# vi Dockerfile
    # 指定基础镜像
    FROM openjdk:8-jdk
    # 配置环境变量
    ENV PARAMS="--server.port=8080 \
    --spring.profiles.active=prod \
    --spring.cloud.nacos.discovery.server-addr=nacos-headless.ruoyi.svc.cluster.
local:8848 \
    --spring.cloud.nacos.config.server-addr=nacos-headless.ruoyi.svc.cluster.lo
cal:8848 \
    --spring.cloud.nacos.config.namespace=prod \
    --spring.cloud.nacos.config.file-extension=yml"
    ENV AGENTS="-javaagent:agent/skywalking-agent.jar \
    -Dskywalking.agent.service_name=ruoyi-system \
    -Dskywalking.collector.backend_service=skywalking-oap-headless.skywalking.s
vc.cluster.local:11800"
    # 同步时间
    RUN  /bin/cp  /usr/share/zoneinfo/Asia/Shanghai  /etc/localtime  &&  echo
'Asia/Shanghai' >/etc/timezone
    # 创建文件夹
    RUN mkdir -p /home/ruoyi
    # 指定工作目录
    WORKDIR /home/ruoyi
    # 拷贝 jar 包
    COPY ruoyi-modules-system.jar /home/ruoyi/app.jar
```

```
    # 拷贝 agent
    COPY agent /home/ruoyi/agent
    # 暴露端口
    EXPOSE 8080
    # 运行 jar 包
    ENTRYPOINT ["/bin/sh","-c","java ${AGENTS} -Dfile.encoding=utf8 -jar app.jar ${PARAMS}"]
```

（2）构建镜像

```
    [root@k8s-master01 ruoyi-system]# docker build -t 192.168.128.11/ruoyi/ruoyi-system:v2 .
```

（3）推送镜像

```
    [root@k8s-master01 ruoyi-system]# docker push 192.168.128.11/ruoyi/ruoyi-system:v2
```

（4）更新服务

```
1、替换版本
    [root@k8s-master01 ruoyi-gateway]# sed -i "s#:v1#:v2#g" ruoyi-system-deployment.yaml

2、更新服务
    [root@k8s-master01 ruoyi-system]# kubectl apply -f ruoyi-system-deployment.yaml
    deployment.apps/ruoyi-system configured
```

4.4、部署带 Skywalking agent 的 ruoyi-auth 服务

（1）修改 Dockerfile 文件

```
    [root@k8s-master01 ~]# cd /ruoyi/java/ruoyi-auth/
    [root@k8s-master01 ruoyi-auth]# vi Dockerfile
    # 指定基础镜像
    FROM openjdk:8-jdk
    # 配置环境变量
    ENV PARAMS="--server.port=8080 \
    --spring.profiles.active=prod \
    --spring.cloud.nacos.discovery.server-addr=nacos-headless.ruoyi.svc.cluster.local:8848 \
    --spring.cloud.nacos.config.server-addr=nacos-headless.ruoyi.svc.cluster.local:8848 \
    --spring.cloud.nacos.config.namespace=prod \
    --spring.cloud.nacos.config.file-extension=yml"
```

```
    ENV AGENTS="-javaagent:agent/skywalking-agent.jar \
    -Dskywalking.agent.service_name=ruoyi-auth \
    -Dskywalking.collector.backend_service=skywalking-oap-headless.skywalking.svc.cluster.local:11800"
    # 同步时间
    RUN /bin/cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo 'Asia/Shanghai' >/etc/timezone
    # 创建文件夹
    RUN mkdir -p /home/ruoyi
    # 指定工作目录
    WORKDIR /home/ruoyi
    # 拷贝 jar 包
    COPY ruoyi-auth.jar /home/ruoyi/app.jar
    # 拷贝 agent
    COPY agent /home/ruoyi/agent
    # 暴露端口
    EXPOSE 8080
    # 运行 jar 包
    ENTRYPOINT ["/bin/sh","-c","java ${AGENTS} -Dfile.encoding=utf8 -jar app.jar ${PARAMS}"]
```

（2）构建镜像

```
    [root@k8s-master01 ruoyi-auth]# docker build -t 192.168.128.11/ruoyi/ruoyi-auth:v2 .
```

（3）推送镜像

```
    [root@k8s-master01 ruoyi-auth]# docker push 192.168.128.11/ruoyi/ruoyi-auth:v2
```

（4）更新服务

```
1、替换版本
    [root@k8s-master01 ruoyi-auth]# sed -i "s#:v1#:v2#g" ruoyi-auth-deployment.yaml

2、更新服务
    [root@k8s-master01 ruoyi-auth]# kubectl apply -f ruoyi-auth-deployment.yaml
    deployment.apps/ruoyi-auth configured
```

4.5、部署带 Skywalking agent 的 ruoyi-job 服务

（1）修改 Dockerfile 文件

```
    [root@k8s-master01 ~]# cd /ruoyi/java/ruoyi-job/
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
[root@k8s-master01 ruoyi-job]# vi Dockerfile
# 指定基础镜像
FROM openjdk:8-jdk
# 配置环境变量
ENV PARAMS="--server.port=8080 \
--spring.profiles.active=prod \
--spring.cloud.nacos.discovery.server-addr=nacos-headless.ruoyi.svc.cluster.local:8848 \
--spring.cloud.nacos.config.server-addr=nacos-headless.ruoyi.svc.cluster.local:8848 \
--spring.cloud.nacos.config.namespace=prod \
--spring.cloud.nacos.config.file-extension=yml"
ENV AGENTS="-javaagent:agent/skywalking-agent.jar \
-Dskywalking.agent.service_name=ruoyi-job \
-Dskywalking.collector.backend_service=skywalking-oap-headless.skywalking.svc.cluster.local:11800"
# 同步时间
RUN /bin/cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo 'Asia/Shanghai' >/etc/timezone
# 创建文件夹
RUN mkdir -p /home/ruoyi
# 指定工作目录
WORKDIR /home/ruoyi
# 拷贝 jar 包
COPY ruoyi-modules-job.jar /home/ruoyi/app.jar
# 拷贝 agent
COPY agent /home/ruoyi/agent
# 暴露端口
EXPOSE 8080
# 运行 jar 包
ENTRYPOINT ["/bin/sh","-c","java ${AGENTS} -Dfile.encoding=utf8 -jar app.jar ${PARAMS}"]
```

（2）构建镜像

```
[root@k8s-master01 ruoyi-job]# docker build -t 192.168.128.11/ruoyi/ruoyi-job:v2 .
```

（3）推送镜像

```
[root@k8s-master01 ruoyi-job]# docker push 192.168.128.11/ruoyi/ruoyi-job:v2
```

（4）更新服务

```
1、替换版本
```

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

```
    [root@k8s-master01 ruoyi-job]# sed -i "s#:v1#:v2#g" ruoyi-job-deployment.yaml

2、更新服务
    [root@k8s-master01 ruoyi-job]# kubectl apply -f ruoyi-job-deployment.yaml
    deployment.apps/ruoyi-job configured
```

4.6、部署带 Skywalking agent 的 ruoyi-file 服务

（1）修改 Dockerfile 文件

```
    [root@k8s-master01 ~]# cd /ruoyi/java/ruoyi-file/
    [root@k8s-master01 ruoyi-file]# vi Dockerfile
    # 指定基础镜像
    FROM openjdk:8-jdk
    # 配置环境变量
    ENV PARAMS="--server.port=8080 \
    --spring.profiles.active=prod \
    --spring.cloud.nacos.discovery.server-addr=nacos-headless.ruoyi.svc.cluster.
local:8848 \
    --spring.cloud.nacos.config.server-addr=nacos-headless.ruoyi.svc.cluster.lo
cal:8848 \
    --spring.cloud.nacos.config.namespace=prod \
    --spring.cloud.nacos.config.file-extension=yml"
    ENV AGENTS="-javaagent:agent/skywalking-agent.jar \
    -Dskywalking.agent.service_name=ruoyi-file \
    -Dskywalking.collector.backend_service=skywalking-oap-headless.skywalking.s
vc.cluster.local:11800"
    # 同步时间
    RUN /bin/cp /usr/share/zoneinfo/Asia/Shanghai /etc/localtime && echo
'Asia/Shanghai' >/etc/timezone
    # 创建文件夹
    RUN mkdir -p /home/ruoyi
    # 指定工作目录
    WORKDIR /home/ruoyi
    # 拷贝 jar 包
    COPY ruoyi-modules-file.jar /home/ruoyi/app.jar
    # 拷贝 agent
    COPY agent /home/ruoyi/agent
    # 暴露端口
    EXPOSE 8080
    # 运行 jar 包
    ENTRYPOINT ["/bin/sh","-c","java ${AGENTS} -Dfile.encoding=utf8 -jar app.jar
${PARAMS}"]
```

（2）构建镜像

```
[root@k8s-master01 ruoyi-file]# docker build -t 192.168.128.11/ruoyi/ruoyi-file:v2 .
```

（3）推送镜像

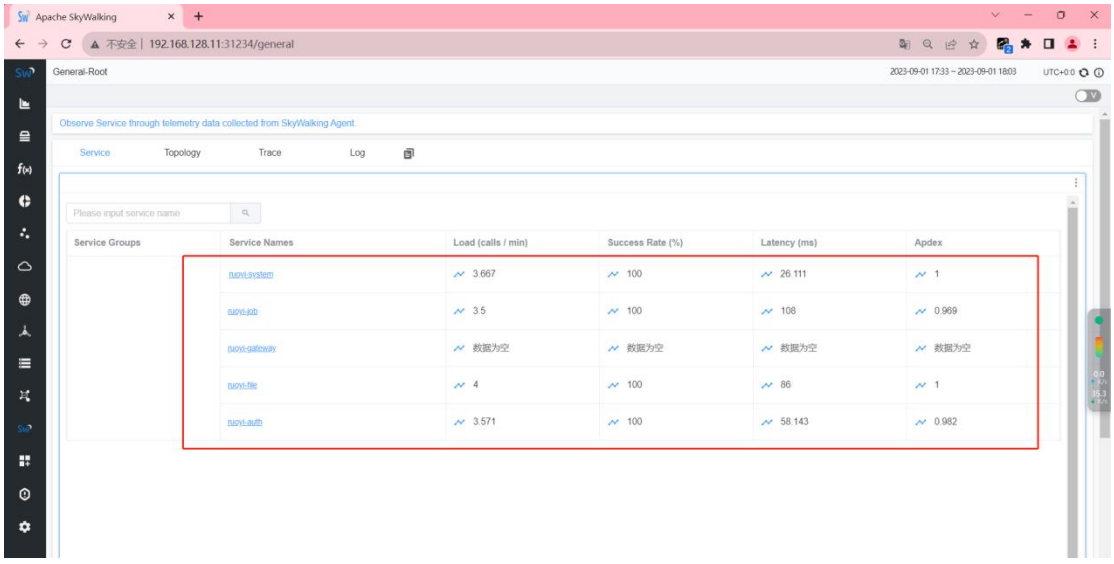
```
[root@k8s-master01 ruoyi-file]# docker push 192.168.128.11/ruoyi/ruoyi-file:v2
```

（4）更新服务

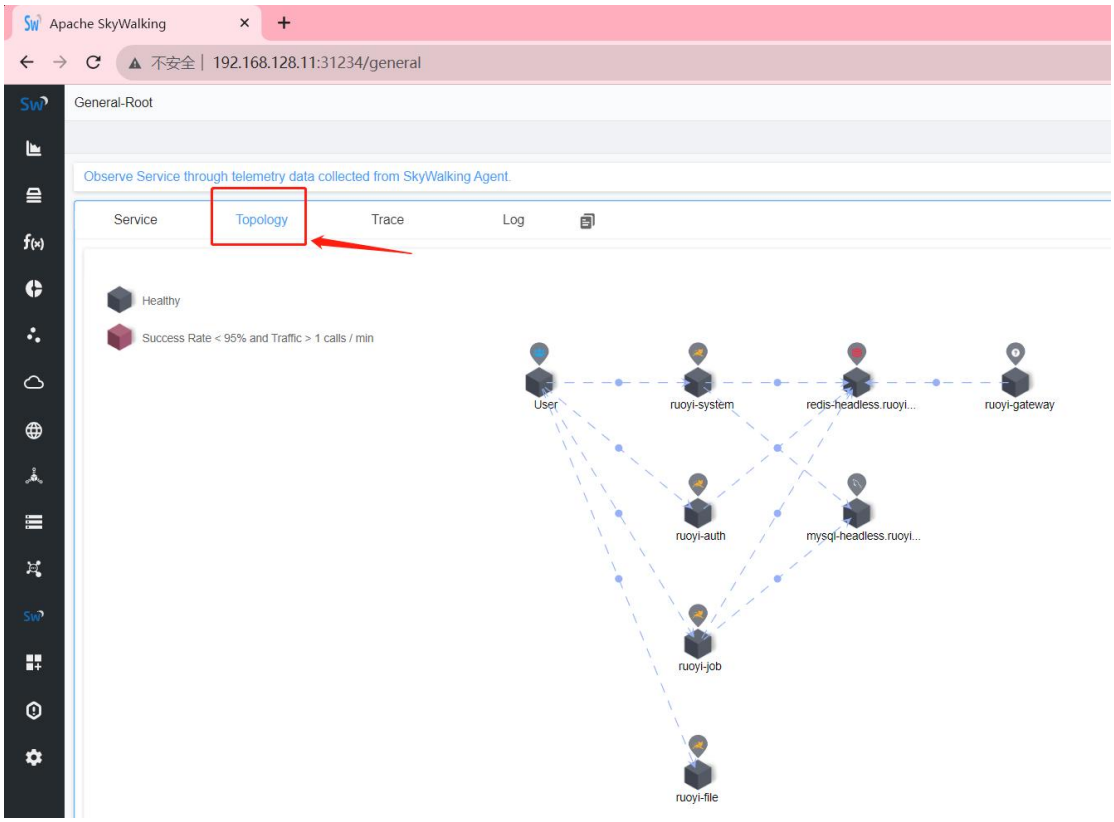
```
1、替换版本
[root@k8s-master01 ruoyi-file]# sed -i "s#:v1#:v2#g" ruoyi-file-deployment.yaml

2、更新服务
[root@k8s-master01 ruoyi-file]# kubectl apply -f ruoyi-file-deployment.yaml
deployment.apps/ruoyi-file configured
```

4.7、Skywalking UI 查看拓扑

（1）如下图，可以看到微服务已经上报数据到了 skywalking oap：



（2）查看拓扑



5、以 SideCar 方式集成 Skywalking agent 到微服务组件

这里演示一下如何以 SideCar 方式集成 Skywalking agent 到微服务组件内。

实现步骤：

1、构建 SiderCar 镜像，将 skywalking agent 软件封装到镜像内。

2、构建 ruoyi-system 镜像，在构建时，设置一个变量，稍等直接将 skywalking agent 启动命令以变量的方式传入即可。

3、修改 deployment 文件，添加初始化容器、临时存储。将临时存储挂载到初始化容器内，将 agent 软件拷贝到临时存储内。再将临时存储挂载到主容器。传入 skywalking agent 启动命令，服务正常启动。

（1）构建 SideCar 镜像

```
1、创建目录，拷贝 agent 目录资源
    [root@k8s-master01 ~]# mkdir /ruoyi/skywalking-agent-sidecar
    [root@k8s-master01 ~]# cd /ruoyi/skywalking-agent-sidecar/
    [root@k8s-master01 skywalking-agent-sidecar]# cp -r /root/skywalking-agent/
agent

2、创建 Dockerfile
    [root@k8s-master01 skywalking-agent-sidecar]# cat Dockerfile
```

```
    FROM busybox:latest
    RUN mkdir /home/ruoyi
    COPY agent /home/ruoyi/agent

3、构建 SideCar 镜像
    [root@k8s-master01    skywalking-agent-sidecar]#    docker    build    -t
192.168.128.11/ruoyi/skywalking-agent-sidecar:v1 .

4、推送镜像到仓库
    [root@k8s-master01       skywalking-agent-sidecar]#       docker       push
192.168.128.11/ruoyi/skywalking-agent-sidecar:v1
```

（2）配置 ruoyi-system 微服务以 SidCar 方式集成 skywalking agent

```
1、修改 dockerfile 文件
    [root@k8s-master01 ~]# cd /ruoyi/java/ruoyi-system/
    [root@k8s-master01 ruoyi-system]# vi Dockerfile
    # 指定基础镜像
    FROM openjdk:8-jdk
    # 配置环境变量
    ENV PARAMS="--server.port=8080 \
    --spring.profiles.active=prod \
    --spring.cloud.nacos.discovery.server-addr=nacos-headless.ruoyi.svc.cluster.
local:8848 \
    --spring.cloud.nacos.config.server-addr=nacos-headless.ruoyi.svc.cluster.lo
cal:8848 \
    --spring.cloud.nacos.config.namespace=prod \
    --spring.cloud.nacos.config.file-extension=yml"
    # 同步时间
    RUN  /bin/cp  /usr/share/zoneinfo/Asia/Shanghai  /etc/localtime  &&  echo
'Asia/Shanghai' >/etc/timezone
    # 创建文件夹
    RUN mkdir -p /home/ruoyi
    # 指定工作目录
    WORKDIR /home/ruoyi
    # 拷贝 jar 包
    COPY ruoyi-modules-system.jar /home/ruoyi/app.jar
    # 暴露端口
    EXPOSE 8080
    # 运行 jar 包
    ENTRYPOINT ["/bin/sh","-c","java ${AGENTS} -Dfile.encoding=utf8 -jar app.jar
${PARAMS}"]
```

2、构建镜像

```
[root@k8s-master01 ruoyi-system]# docker build -t 192.168.128.11/ruoyi/ruoyi-system:v3 .
```

3、推送镜像到仓库

```
[root@k8s-master01 ruoyi-system]# docker push 192.168.128.11/ruoyi/ruoyi-system:v3
```

4、修改 ruoyi-system-deployment.yaml 文件

```
[root@k8s-master01 ruoyi-system]# vi ruoyi-system-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ruoyi-system
  namespace: ruoyi
spec:
  replicas: 1
  selector:
    matchLabels:
      project: ruoyi
      app: ruoyi-system
  template:
    metadata:
      labels:
        project: ruoyi
        app: ruoyi-system
    spec:
      imagePullSecrets:
      - name: registry-pull-secret
      initContainers:
      - name: init-check-harbor
        image: 192.168.128.11/ruoyi/skywalking-agent-sidecar:v1
        imagePullPolicy: IfNotPresent
        command: ['sh', '-c', "cp -r /home/ruoyi/agent/* /cache/"]
        volumeMounts:
        - mountPath: /cache
          name: cache-volume
      containers:
      - name: ruoyi-system
        image: 192.168.128.11/ruoyi/ruoyi-system:v3
        imagePullPolicy: Always
        env:
```

```
        - name: AGENTS
          value:                    '-javaagent:agent/skywalking-agent.jar
-Dskywalking.agent.service_name=ruoyi-system-agent
-Dskywalking.collector.backend_service=skywalking-oap-headless.skywalking.svc.c
luster.local:11800'
          ports:
            - protocol: TCP
              containerPort: 8080
          readinessProbe:
            tcpSocket:
              port: 8080
            initialDelaySeconds: 10
            periodSeconds: 10
          livenessProbe:
            tcpSocket:
              port: 8080
            initialDelaySeconds: 10
            periodSeconds: 10
          volumeMounts:
          - name: localtime
            mountPath: /etc/localtime
          - mountPath: /home/ruoyi/agent
            name: cache-volume
        volumes:
          - name: cache-volume
            emptyDir: {}
          - name: localtime
            hostPath:
              path: /usr/share/zoneinfo/Asia/Shanghai
    strategy:
      type: RollingUpdate
      rollingUpdate:
        maxSurge: 1
        maxUnavailable: 0
```

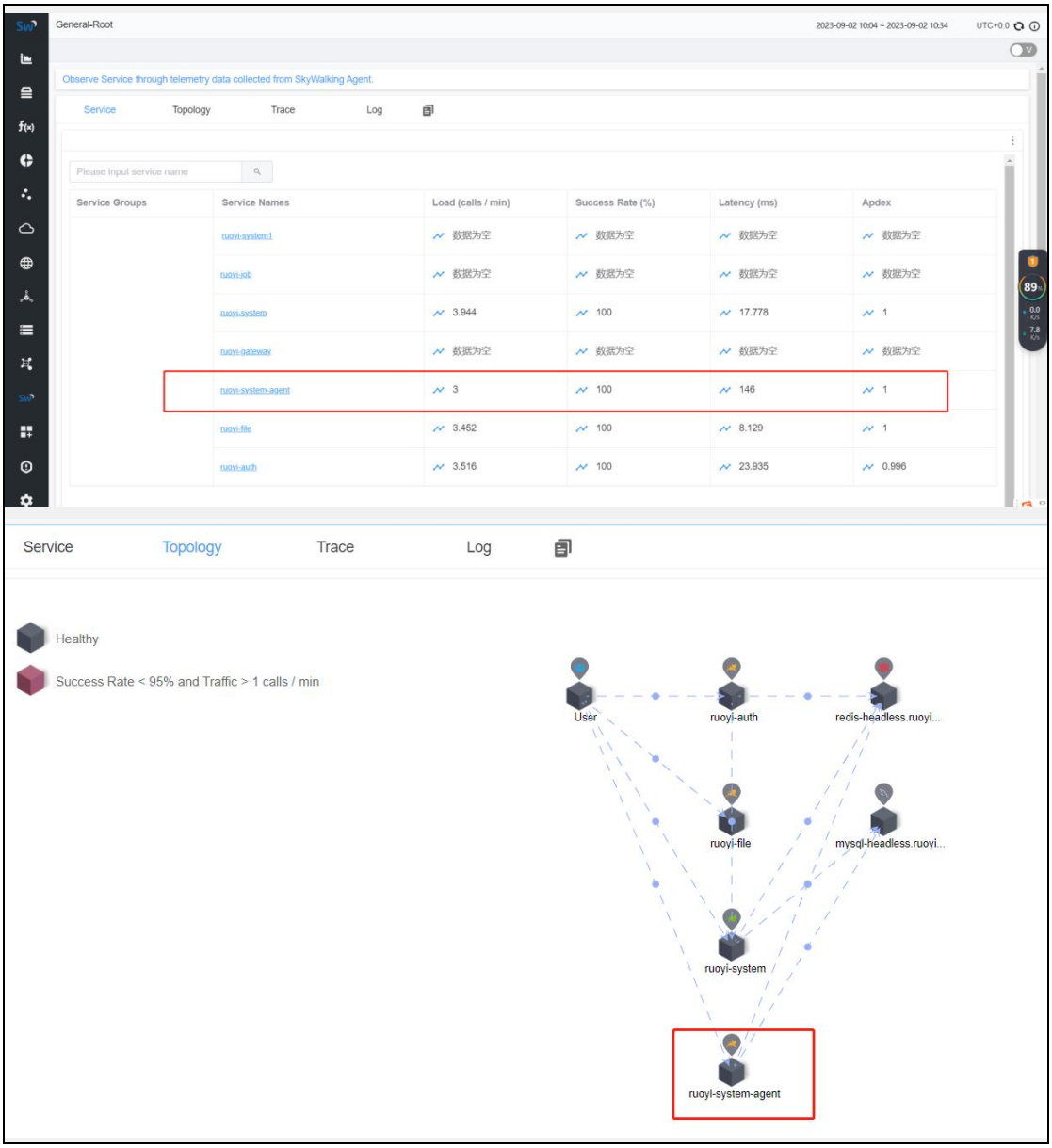
5、重新创建 ruoyi-system 服务

```
[root@k8s-master01       ruoyi-system]#       kubectl       delete       -f
ruoyi-system-deployment.yaml
[root@k8s-master01       ruoyi-system]#       kubectl       apply       -f
ruoyi-system-deployment.yaml
```

（3）Skywalking UI 查看拓扑，如下图，可以看到 ruoyi-system-agent 已经被

版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。

监控到。



版权声明，本文档全部内容及版权归“张岩峰”老师所有，只可用于自己学习使用，禁止私自传阅，违者依法追责。