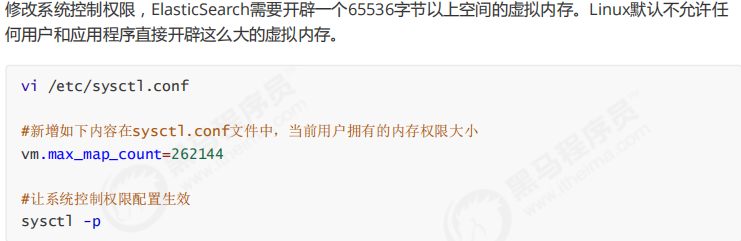
关闭docker里面的es，因为版本太低，影响正常es

重置

只允许一个值，否则不生效

通过skywalking进行监控链路

在测试环境，按生产环境等比缩放

然后用jmeter对链路进行加压测试，观察各个节点的表现。

如果skywalking提示错误了，那么就达到这个节点的最大限度了，从而知道系统的瓶颈和最大承压限度。

因为jmeter只能看到调用接口的反馈情况，而链路中各个节点发生了什么，不知道。

用skywalking可以对每个节点进行跟踪。

服务之间的调用有业务关系，但是不能感知到性能瓶颈在哪里。

探针会把数据上报，后端服务接收到探针的数据（包括微服务网格上报的数据），通过grpc/http，分布式链路追踪的数据，可累加的数据（统计性数据）。然后把这些数据存放到存储介质中。Ui调用查询引擎的接口，就可以访问存储介质中的数据了。

代码上线了就是服务， 节点就是skywalking中的实例的概念，服务对外暴露的接口就是端点

用es作为存储源

操作

1、创建目录

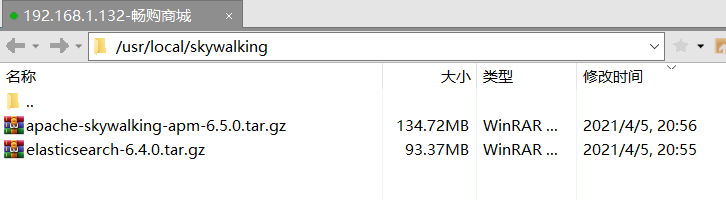
建议将虚拟机内存设置为3G并且将CPU设置成2核，防止资源不足。

2、将资源目录中的elasticsearch和skywalking安装包上传到虚拟机/usr/local/skywalking目录下。

elasticsearch-6.4.0.tar.gz ---elasticsearch 6.4的安装包，Skywalking对es版本号有一定要求，最

好使用6.3.2以上版本，如果是7.x版本需要额外进行配置。

apache-skywalking-apm-6.5.0.tar.gz ---Skywalking最新的安装包



mkdir /usr/local/skywalking3、首先安装elasticsearch，将压缩包解压。

tar -zxvf ./elasticsearch-6.4.0.tar.gz

修改Linux系统的限制配置，将文件创建数修改为65536个。

1. 修改系统中允许应用最多创建多少文件等的限制权限。Linux默认来说，一般限制应用最多

创建的文件是65535个。但是ES至少需要65536的文件创建数的权限。

2. 修改系统中允许用户启动的进程开启多少个线程。默认的Linux限制root用户开启的进程可

以开启任意数量的线程，其他用户开启的进程可以开启1024个线程。必须修改限制数为

4096+。因为ES至少需要4096的线程池预备。

vi /etc/security/limits.conf

#新增如下内容在limits.conf文件中

es soft nofile 65536

es hard nofile 65536

es soft nproc 4096

es hard nproc 4096

修改系统控制权限，ElasticSearch需要开辟一个65536字节以上空间的虚拟内存。Linux默认不允许任

何用户和应用程序直接开辟这么大的虚拟内存。

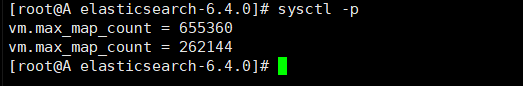
vi /etc/sysctl.conf

#新增如下内容在sysctl.conf文件中，当前用户拥有的内存权限大小

vm.max\_map\_count=262144

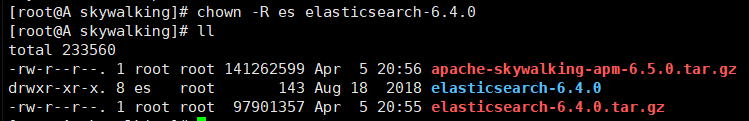
#让系统控制权限配置生效

sysctl -p



建一个用户， 用于ElasticSearch启动。

123456789a



ES在5.x版本之后，强制要求在linux中不能使用root用户启动ES进程。所以必须使用其他用户启

动ES进程才可以。

#创建用户

useradd es

#修改上述用户的密码

passwd es

#修改elasicsearch目录的拥有者

chown -R es elasticsearch-6.4.0

使用es用户启动elasticsearch

#切换用户

su es

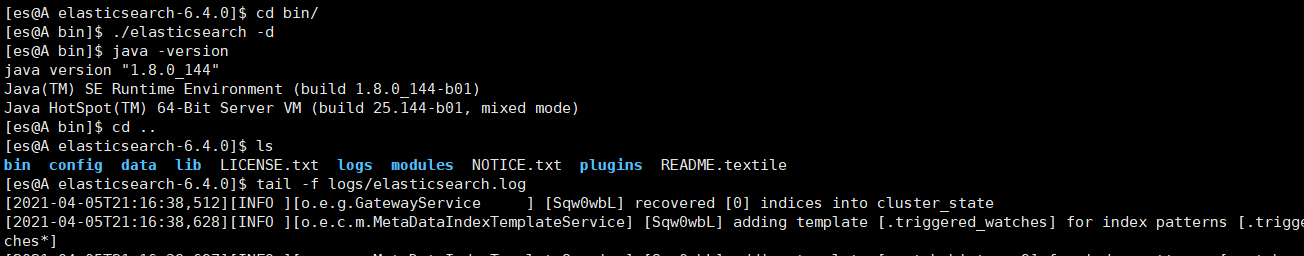
#到ElasticSearch的bin目录下

cd bin/

#后台启动

./elasticsearch -d

默认ElasticSearch是不支持跨域访问的，所以在不修改配置文件的情况下我们只能从虚拟机内部进行访



问测试ElasticSearch是否安装成功，使用curl命令访问9200端口：curl http://localhost:9200

如果显示出如下信息，就证明ElasticSearch安装成功：

{

"name" : "xbruNxf",

"cluster\_name" : "elasticsearch",

"cluster\_uuid" : "JJQfHN9QQVuXpH5fu9H1jg",

"version" : {

"number" : "6.4.0",

"build\_flavor" : "default",

"build\_type" : "tar",

"build\_hash" : "595516e",

"build\_date" : "2018-08-17T23:18:47.308994Z",

"build\_snapshot" : false,

"lucene\_version" : "7.4.0",

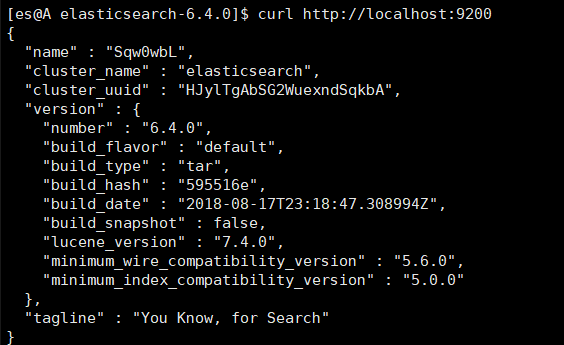
"minimum\_wire\_compatibility\_version" : "5.6.0",

"minimum\_index\_compatibility\_version" : "5.0.0"

},

"tagline" : "You Know, for Search"

}



4、安装Skywalking，分为两个步骤：

安装Backend后端服务 -收集探针上报的数据，需要对数据进行加工，加工后放入es，提供接口

安装UI -查询数据，可视化页面

首先切回到root用户,切换到目录下，解压Skywalking压缩包。

#切换到root用户

su root

#切换到skywalking目录

cd /usr/local/skywalking

#解压压缩包

tar -zxvf apache-skywalking-apm-6.4.0.tar.gz

修改Skywalking存储的数据源配置：

cd apache-skywalking-apm-bin

vi config/application.yml



Agent是探针文件夹

Bin存储了所有可执行文件

Config后端服务的配置

Webapp用来启动ui的安装包以及他的配置文件

我们可以看到默认配置中，使用了H2作为数据源。我们将其全部注释。

# h2:

# driver: ${SW\_STORAGE\_H2\_DRIVER:org.h2.jdbcx.JdbcDataSource}

# url: ${SW\_STORAGE\_H2\_URL:jdbc:h2:mem:skywalking-oap-db}

# user: ${SW\_STORAGE\_H2\_USER:sa}

# metadataQueryMaxSize: ${SW\_STORAGE\_H2\_QUERY\_MAX\_SIZE:5000}

# mysql:

# metadataQueryMaxSize: ${SW\_STORAGE\_H2\_QUERY\_MAX\_SIZE:5000}

将ElasticSearch对应的配置取消注释：

storage:elasticsearch:

nameSpace: ${SW\_NAMESPACE:""}

clusterNodes: ${SW\_STORAGE\_ES\_CLUSTER\_NODES:localhost:9200}

protocol: ${SW\_STORAGE\_ES\_HTTP\_PROTOCOL:"http"}

trustStorePath: ${SW\_SW\_STORAGE\_ES\_SSL\_JKS\_PATH:"../es\_keystore.jks"}

trustStorePass: ${SW\_SW\_STORAGE\_ES\_SSL\_JKS\_PASS:""}

user: ${SW\_ES\_USER:""}

password: ${SW\_ES\_PASSWORD:""}

indexShardsNumber: ${SW\_STORAGE\_ES\_INDEX\_SHARDS\_NUMBER:2}

indexReplicasNumber: ${SW\_STORAGE\_ES\_INDEX\_REPLICAS\_NUMBER:0}

# Those data TTL settings will override the same settings in core module.

recordDataTTL: ${SW\_STORAGE\_ES\_RECORD\_DATA\_TTL:7} # Unit is day

otherMetricsDataTTL: ${SW\_STORAGE\_ES\_OTHER\_METRIC\_DATA\_TTL:45} # Unit is day

monthMetricsDataTTL: ${SW\_STORAGE\_ES\_MONTH\_METRIC\_DATA\_TTL:18} # Unit is

month

# # Batch process setting, refer to

https://www.elastic.co/guide/en/elasticsearch/client/java-api/5.5/java-docs-

bulk-processor.html

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

metadataQueryMaxSize: ${SW\_STORAGE\_ES\_QUERY\_MAX\_SIZE:5000}

segmentQueryMaxSize: ${SW\_STORAGE\_ES\_QUERY\_SEGMENT\_SIZE:200}

默认使用了localhost下的ES,所以我们可以不做任何处理，直接进行使用。启动OAP程序：

bin/oapService.sh

这样安装Backend后端服务就已经完毕了，接下来我们安装UI。先来看一下UI的配置文件：

cat webapp/webapp.yml

#默认启动端口

server:

port: 8080

collector:

path: /graphql

ribbon:

ReadTimeout: 10000

#OAP服务，如果有多个用逗号隔开

listOfServers: 127.0.0.1:12800

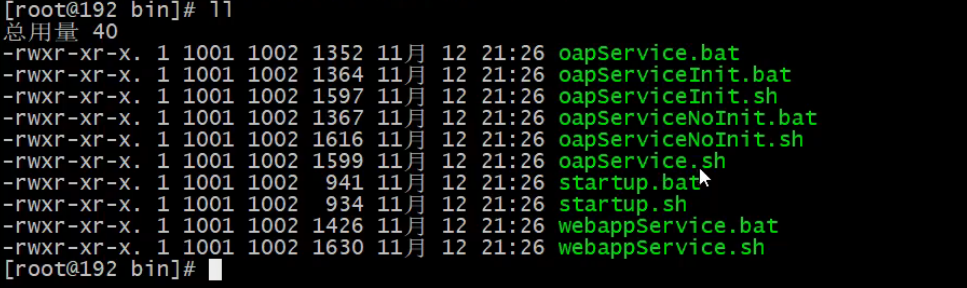
目前的默认配置不用修改就可以使用，启动UI程序：

/bin/webappService.sh

然后我们就可以通过浏览器访问Skywalking的可视化页面了，访问地址:http://虚拟机IP地址:8080,如果

出现下面的图，就代表安装成功了。

/bin/startup.sh可以同时启动backend和ui，后续可以执行该文件进行重启



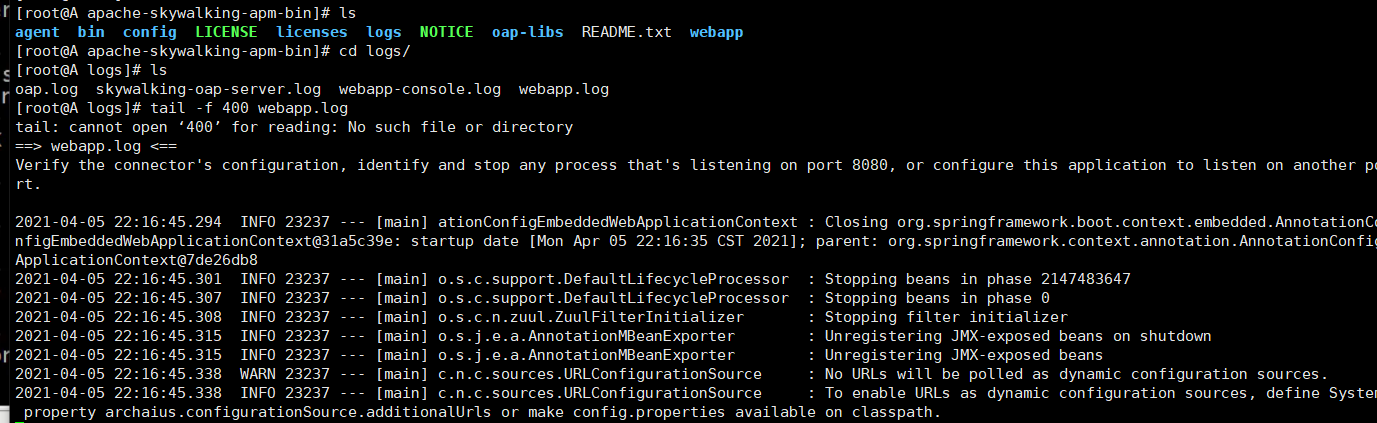
oapservice.sh 用来启动后端服务

webappservice.sh用来启动页面

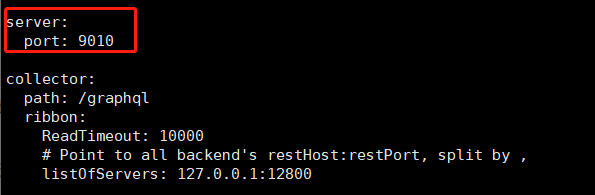
startup.sh把这两部分都启动起来







注意空格不能少，否则不识别



<http://192.168.1.132:9010/>

下面就可以使用探针把数据上报

Agent

Springmvc，mysql，记录这些组件调用的时常

**2.1 agent的使用**

agent探针可以让我们不修改代码的情况下，对java应用上使用到的组件进行动态监控，获取运行数据

发送到OAP上进行统计和存储。agent探针在java中是使用java agent技术实现的，不需要更改任何代

码，java agent会通过虚拟机(VM)接口来在运行期更改代码。

Agent探针支持 JDK 1.6 - 12的版本，Agent探针所有的文件在Skywalking的agent文件夹下。文件目录

如下；

部分插件在使用上会影响整体的性能或者由于版权问题放置于可选插件包中，不会直接加载，如

果需要使用，将可选插件中的jar包拷贝到plugins包下。

+-- agent

+-- activations

apm-toolkit-log4j-1.x-activation.jar

apm-toolkit-log4j-2.x-activation.jar

apm-toolkit-logback-1.x-activation.jar

...

//配置文件

+-- config

agent.config

//组件的所有插件

+-- plugins

apm-dubbo-plugin.jar

apm-feign-default-http-9.x.jar

apm-httpClient-4.x-plugin.jar

.....

//可选插件

+-- optional-plugins

apm-gson-2.x-plugin.jar

.....

+-- bootstrap-plugins

jdk-http-plugin.jar

.....

+-- logs

skywalking-agent.jar由于没有修改agent探针中的应用名，所以默认显示的是Your\_ApplicationName。我们修改下应用名

称，让他显示的更加正确。编辑agent配置文件：

我们在配置中找到这么一行：

这里的配置含义是可以读取到SW\_AGENT\_NAME配置属性，如果该配置没有指定，那么默认名称为

Your\_ApplicationName。这里我们把Your\_ApplicationName替换成skywalking\_tomcat。

然后将tomcat重启:

环境变量有值用环境变量的值，没有就用后面的值

**2.1.1 Linux 下Tomcat7和8中使用**

1.要使用Skywalking监控Tomcat中的应用，需要先准备一个Spring Mvc项目，在资源中已经提供了打

包好的文件

skywalking\_springmvc-1.0-SNAPSHOT.war 。

以下是该项目的接口代码：

将资源文件下的 apache-tomcat-8.5.47.tar.gz 文件上传至虚拟机/usr/local/skywalking目录下，然

后解压:

cd /usr/local/skywalking/apache-skywalking-apm-bin/agent/config

vi agent.config

# The service name in UI

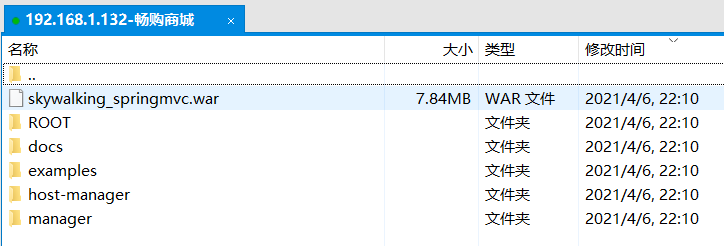
agent.service\_name=${SW\_AGENT\_NAME:Your\_ApplicationName}

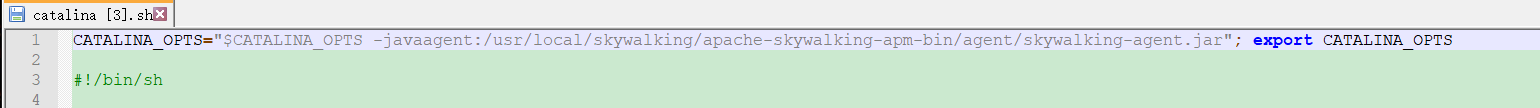
# The service name in UI

agent.service\_name=${SW\_AGENT\_NAME:skywalking\_tomcat}

./shutdown.sh

./startup.sh





注意：

1 linux环境，必须把干扰的es都停掉（docker中的es）

2配置文件要按照说明配置好，注意空格

3要按顺序启动，先不要使用start.sh。否则容易获取不到数据，从而导致12800监听不到，要按顺序启动。oapService.sh启动后查看日志，没问题再启动下一个webappService.sh