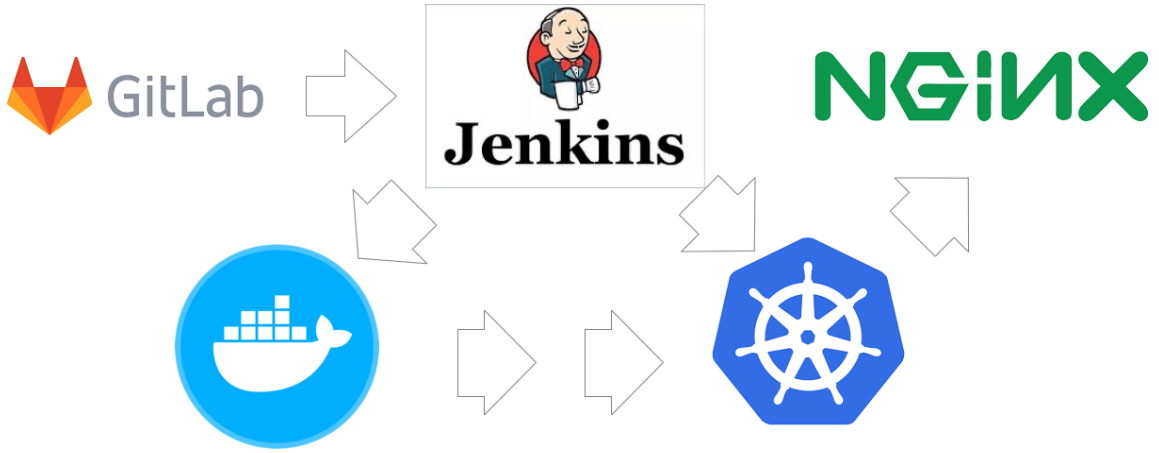
讲师(老司机)

容器虚拟化技术和自动化部署



SonarQube

简介

- 代码质量、安全扫描和分析的平台
- 多维度分析代码：代码量、安全隐患、编写规范隐患、重复度、复杂度、代码增量、单元测试覆盖率等
- 支持近30+种开发语言：包含主流开发语言java/python/c#/javascript/go/c++等
- 能够与开发工具(idea,eclipse)、CI/CD平台(jenkins)、版本控制管理工具(gitlab,github)等完美集成
- 能够帮助开发人员写出更干净、高质量、更安全的代码

官网地址

官网

```
https://www.sonarqube.org
```

docker官网

```
https://hub.docker.com/_/sonarqube
```

sonarscanner官网

```
https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/

github官网
https://github.com/newtmitch/docker-sonar-scanner

docker官网
https://hub.docker.com/r/newtmitch/sonar-scanner/
```

PostgreSQL官网

SonarQube 7.9以后的版本已经放弃了MySQL 。国内下载速度比较理想。

```
https://www.postgresql.org/

下载地址：
https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
```

sonarqube版本

- 社区版(Community)
- 企业版(Enterprise)
- 数据中心版(Data Center)
- 开发版(Developer)

SonarQube也是一个C/S架构的服务。根据其官网所述，包含以下架构：

SonarQube服务端:

- Web服务器
- 搜索引擎-Elasticsearch to back searches from the UI
- 后台计算服务-连接数据库

后台数据库:

- SonarQube实例的配置信息，如安全、插件等
- 项目、视图的质量快照数据

SonarQube Plugin

- 安装在服务端的插件，例如语言包、SCM、认证、治理等等

windows版

sonarqube目录

```
这里简单说下每个目录作用
1.bin 用来启动 SonarQube 服务，这里已经提供好了不同系统启动 | 停止脚本了，目前提供了linux-x86-32、linux-x86-64、macosx-universal-64、windows-x86-32、windows-x86-64

```

```
2.conf 用来存放配置文件，若需要修改配置，修改 sonar.properties 文件即可。

3.data 用来存放数据，SonarQube默认使用 h2 数据库存储，同时支持其他如Orace、Mssql、Postgresql数据库存储。

4.extensions 用来存放插件 jar 包，以后我们需要安装插件就放在这里。

5.lib 用来存放各种所依赖的 jar 包，包括上边各数据库驱动包 (默认已提供一个版本，如果版本不匹配，则在这里手动更新下)。

6.logs 用来存放各日志信息

7.web 用来提供 SonarQube web 网页服务。
```

安装bug1

错误信息

```
jvm 1    | WrapperSimpleApp: Encountered an error running main: java.lang.IllegalStateException: SonarQube requires Java 11+ to run
jvm 1    | java.lang.IllegalStateException: SonarQube requires Java 11+ to run
jvm 1    |      at org.sonar.application.App.checkJavaVersion(App.java:93)
jvm 1    |      at org.sonar.application.App.start(App.java:56)
jvm 1    |      at org.sonar.application.App.main(App.java:98)
jvm 1    |      at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
jvm 1    |      at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
jvm 1    |      at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
jvm 1    |      at java.lang.reflect.Method.invoke(Method.java:498)
jvm 1    |      at org.tanukisoftware.wrapper.WrapperSimpleApp.run(WrapperSimpleApp.java:240)
jvm 1    |      at java.lang.Thread.run(Thread.java:748)
wrapper  | <-- Wrapper Stopped
```

解决方案

sonarqube7.9 ->必须使用JDK11->不能使用其他JDK版本

```
更换本机JAVA_HOME变量为JDK11版本。

小伙伴们可以自行测试JDK14是否支持sonarqube7.9.X
```

安装bug2

错误信息

```
app[][o.e.t.n.Netty4Transport] exception caught on transport layer [[id: 0xe5d5a747, L:/127.0.0.1:18558 - R:/127.0.0.1:9001]], closing connection
io.netty.handler.codec.DecoderException: java.io.StreamCorruptedException: invalid internal transport message format, got (48,54,54,50)
```

解决方案

```
1.查看ELK的日志
%sonarqube_home%/logs/es.log
发现9001端口被windows系统占用。

2.打开cmd窗口。输入命令:
netstat -ano

3.发现PID=4的进程占用9001端口
打开任务管理器，找到PID=4的进程。结束进程，发现无法结束进程。
只能进入sonar.properties里修改ES端口。


4.更改ELK启动端口为9003
打开文件:
%sonarqube_home%/conf/sonar.properties

更改文件272行左右。去掉注释，更新端口为9003
sonar.search.port=9003
```

中文插件

下载sonar-l10n-zh-plugin-1.29.jar

```
https://github.com/SonarQubeCommunity/sonar-l10n-zh
```

SonarLint

sonarlint官网

```
https://www.sonarlint.org/
```

idea插件官网

```
https://plugins.jetbrains.com/

sonarlint下载地址：
https://plugins.jetbrains.com/plugin/7973-sonarlint
```

安装sonarlint

```
离线安装
```

####

Sonar可以从以下七个维度检测代码质量，而作为开发人员至少需要在本地开发环境处理前6种代码质量问题

1. **不遵循代码标准：** sonar可以通过PMD,CheckStyle,Findbugs等等代码规则检测工具规范代码编写
2. **潜在的缺陷：** sonar可以通过PMD,CheckStyle,Findbugs等等代码规则检测工具检测出潜在的缺陷
3. **糟糕的复杂度分布：** 文件、类、方法等，如果复杂度过高将难以改变，这会使得开发人员难以理解它们 且如果没有自动化的单元测试，对于程序中的任何组件的改变都将可能导致需要全面的回归测试
4. **重复：** 显然程序中包含大量复制粘贴的代码是质量低下的，sonar可以展示源码中重复严重的地方
5. **注释不足或者过多：** 没有注释将使代码可读性变差，特别是当不可避免地出现人员变动时，程序的可读性将大幅下降 而过多的注释又会使得开发人员将精力过多地花费在阅读注释上，亦违背初衷
6. **缺乏单元测试：** sonar可以很方便地统计并展示单元测试覆盖率
7. **糟糕的设计：** 通过sonar可以找出循环，展示包与包、类与类之间相互依赖关系，可以检测自定义的架构规则 通过sonar可以管理第三方的jar包， 检测耦合。

配置sonarlint

SonarLint General Settings :针对IDEA所有打开项目之后的SonarLint通用配置.
SonarLint Project Settings :针对当前这一个项目配置生效.

```
需要sonarqube生成token或者是用户名、密码方式配置

0de45442607056fe6f694f67642e92c94d611dc1
```

使用sonarlint

```
1.本地检测单个文件
2.本地检测项目
```

sonarscanner

可以脱离开发环境进行测试。sonarscanner可以帮助开发人员将代码上传远程sonarqube服务器上。

SonarQube在服务器端不支持32位系统。Sonarqube-scanner支持32位系统。最新版本已经无法下载32位操作系统的版本

官方文档

```
https://docs.sonarqube.org/latest/analysis/scan/sonarscanner/
```

环境变量

```
配置环境变量 path中添加D:\sonarqube\sonar-scanner-4.2.0.1873-windows\bin
在cmd检查安装是否成功

sonar-scanner -v
```

配置sonar-scanner

```
在sonar-scanner.properties中配置对应的sonar.host.url 、sonar.sourceEncoding
D:\sonarqube\sonar-scanner-4.2.0.1873-windows\conf\sonar-scanner.properties


文件内容调整如下：
sonar.host.url=http://localhost:9000

#----- Default source code encoding
sonar.sourceEncoding=UTF-8
```

sonar-project.properties

```
#项目的key  唯一不重复即可
sonar.projectKey=sonarScannerTest
#项目的名字
sonar.projectName=sonarScannerTest
#项目的版本
sonar.version=1.0
#需要分析的源码的目录，多个目录用英文逗号隔开
sonar.sources=src
#sonarQube扫描的对象是.class而不是.java文件
sonar.java.binaries=target/classes
#解析的开发语言，如果需要解析其他语言，需要下载对应的插件
sonar.language=java
#编码格式
sonar.sourceEncoding=UTF-8
```

扫描项目

```
首先执行mvn打包命名
mvn clean package

上传代码
sonar-scanner
```

集成maven

官方文档

```
https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-maven/
```

maven插件方式

```
            <plugin>
                <groupId>org.sonarsource.scanner.maven</groupId>
                <artifactId>sonar-maven-plugin</artifactId>
                <version>3.7.0.1746</version>
            </plugin>
```

java编译插件,windows系统已默认安装->sonar-java-plugin-5.13.1.18282.jar

mvn sonar:sonar -Dsonar.java.binaries=target/sonar -Dsonar.host.url=http://localhost:9000 -Dsonar.login=0de45442607056fe6f694f67642e92c94d611dc1

settings.xml文件方式

```
<settings>
<pluginGroups>
<pluginGroup>org.sonarsource.scanner.maven</pluginGroup>
</pluginGroups>
<profiles>
       <profile>
          <id>sonar</id>
          <activation>
             <activeByDefault>true</activeByDefault>
          </activation>
          <properties>
             <sonar.language>java</sonar.language>
             <sonar.login>admin</sonar.login>
             <sonar.password>admin</sonar.password>
             <sonar.host.url>http://localhost:9000</sonar.host.url>
             <!--代码分析包括哪些文件需要分析，英文逗号分隔-->
             <sonar.exclusions>
```

```
                **/*.java,**/*.xml
            </sonar.exclusions>
        </properties>
      </profile>
    </profiles>
  <activeProfiles>
    <!-- 这步配置，sonar的profile配置才能生效 -->
    <activeProfile>sonar</activeProfile>
  </activeProfiles>
</settings>
```

运行项目

```
mvn clean verify sonar:sonar
```

docker版

sonarqube官网

```
https://hub.docker.com/_/sonarqube
```

postgreSQL官网

```
https://hub.docker.com/_/postgres
```

基础镜像

```
docker pull sonarqube:7.9.4-community
docker pull sonarqube:8.4.2-community

最新版本
docker pull postgres:13-alpine

根据官方要求：
sonarqube:7.9.4-community版本postgreSQL数据的版本为：9.3-9.6。最高支持10版本。
docker pull postgres:9.6.19-alpine

sonarqube:8.4.2-community版本postgreSQL数据的版本为：9-11。最高支持12版本。
docker pull postgres:12.4-alpine
```

ELK初始化设置

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

添加参数: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

重启生效:让系统控制权限配置生效
sysctl -p
```

自定义镜像

查看sonarqube版本

查看sonarqube版本的目的是为了自定义镜像。

```

docker run -itd --name=sonarqube -p 9000:9000 sonarqube:7.9.4-community
docker exec -it sonarqube /bin/bash
cat /etc/issue
返回信息如下:
Debian GNU/Linux 10 \n \l

结论：sonarqube7.9.x是debian系统
exit
docker stop sonarqube
docker rm sonarqube



docker run -itd --name=sonarqube -p 9000:9000 sonarqube:8.4.2-community
docker exec -it sonarqube /bin/bash
cat /etc/issue
返回信息如下:
Welcome to Alpine Linux 3.11
Kernel \r on an \m (\l)


结论：sonarqube8.4.x是Alpine系统
exit
docker stop sonarqube
docker rm sonarqube
```

dockerfile

sonarqube7.x是debian系统。

```
FROM sonarqube:7.9.4-community
# 作者信息
MAINTAINER sonarqube from date UTC by Asia/Shanghai "www.lagou.com"
ENV TZ Asia/Shanghai
#将中文jar包复制到对应目录
ADD sonar-l10n-zh-plugin-1.29.jar /opt/sonarqube/extensions/plugins/
```

制作镜像

```
docker build --rm -t lagou/sonarqube:7.9.4-community .
```

试安装

试安装的目的是为了获得容器内挂载卷的内容

```
docker run -itd --name=sonarqube -p 9000:9000 lagou/sonarqube:7.9.4-community
docker cp sonarqube:/opt/sonarqube/ /data/


chmod -R 777 /data/sonarqube/*
```

docker-compose

ELK初始化设置

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

添加参数: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

重启生效:让系统控制权限配置生效
sysctl -p
```

挂载卷

```
创建容器映射路径
mkdir -p /data/postgresql/data
mkdir -p  /data/sonarqube/{extensions,logs,data,conf}

启动容器映射路径权限问题
chmod -R 777 /data/postgresql/*
chmod -R 777 /data/sonarqube/*
```

docker-compose.yml

```yaml
version: "3.6"
services:
  db:
    image: postgres:9.6.19-alpine
    ports:
    - 5432:5432
    restart: always
    environment:
      - POSTGRES_DB=sonar
      - POSTGRES_USER=sonar
      - POSTGRES_PASSWORD=sonar
      - TZ=Asia/Shanghai
    volumes:
    - /data/postgresql/data:/var/lib/postgresql/data
  sonarqube:
    image: lagou/sonarqube:7.9.4-community
    ports:
    - 9000:9000
    environment:
      - SONARQUBE_JDBC_URL=jdbc:postgresql://db:5432/sonar
    restart: always
    depends_on:
      - db
    volumes:
    - /data/sonarqube/conf:/opt/sonarqube/conf
    - /data/sonarqube/data:/opt/sonarqube/data
    - /data/sonarqube/logs:/opt/sonarqube/logs
    - /data/sonarqube/extensions:/opt/sonarqube/extensions
```

运行容器

```
docker-compose up -d
docker-compose logs -f

docker-compose ps

```