

阅读目录

- [前言](#)
- [系统架构](#)
 - [Jenkins系统架构](#)
 - [EasyCi系统架构](#)

前言

本人是一家互联网公司的java开发，由于公司初期公司未招运维人员，恰好我对linux比较熟悉，便在公司服务器搭建了一套Jenkins、Gitlab、Maven私服、Docker私服、Sonarqube、ELK、FastDFS等一套持续集成的运维环境。

后来发现，运维这块以后也归我管了……平时做系统开发，还要兼职运维，一些前端或者后端的发布都要找我来创建Jenkins项目、添加gitlab hook、添加dockerfile文件等等。

所以就想自己写一套简单的持续化集成发布的系统。便有了接下来的EasyCi。

EasyCi系统开发的目的是免去远程发布的免密登录、拉取gitlab代码的认证、手动添加gitlab hook、查看gitlab中该项目的git地址等等多余的操作。这些操作均有后台自动完成，系统提供运行环境一键安装脚本、自动化安装部署本系统、开箱即用，只需要几个参数即可实现项目的远程构建，暂时只支持vue和java项目的构建。

EasyCi系统采用B/S架构，后端采用springboot框架、前端采用Vue的element ui、数据库采用mysql、运行工具为shell脚本、采用websocket进行实时日志传输。

由于系统由本人独立开发，对前端开发不是很擅长，页面比较简单，只为实现基本功能，后续会对功能和页面进行优化。

系统架构

Jenkins系统架构

工具链

Jenkins：集成各种工具，持续集成、持续交付

Gitlab：代码托管库，通过gitlab hook触发Jenkins持续集成交付

Maven私服：私有jar包仓库

Docker私服：私有docker镜像仓库，用于远程构建

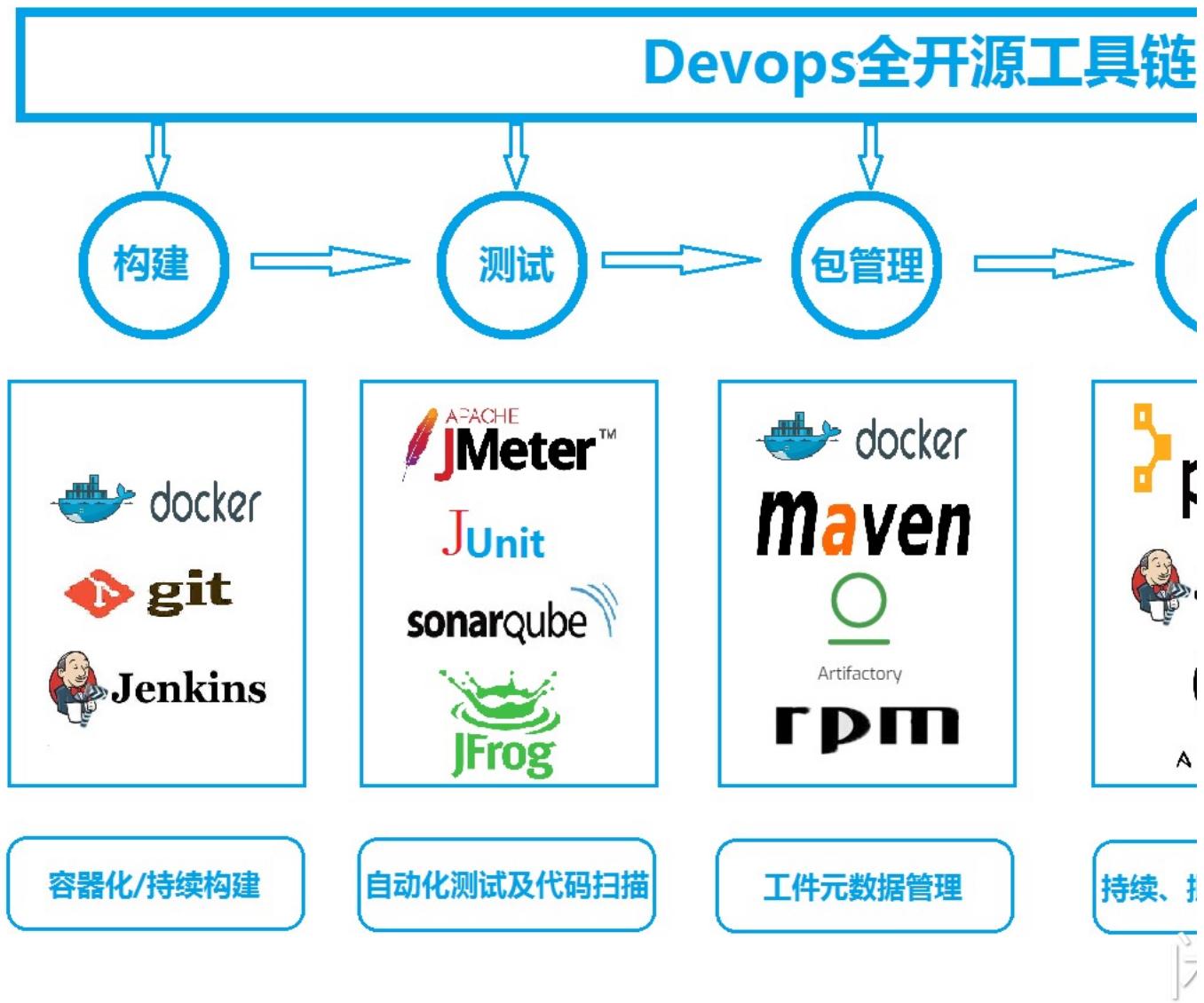
Sonarqube：代码审查工具

ELK：日志分析系统、日志系统

流程

- 1.Jenkins集成gitlab、maven、docker、sonarqube，安装各种插件
- 2.开发人员提交代码到gitlab，触发gitlab hook中添加的jenkins项目url进行jenkins构建
- 3.jenkins自动拉取该项目的gitlab代码，进行打包、再打包成docker镜像，docker镜像提交到docker私服
- 4.连接需要发布的服务器，拉去docker私服中该项目的镜像，docker run运行容器
- 5.Jenkins构建项目运行完成后添加gitlab tag并提交gitlab作为以后回滚的依据
- 6.Jenkins构建失败则自动拉取最新的gitlab tag重新构建，执行回滚，回滚再次构建失败便停止
- 7.构建成功或者回滚成功或者回滚失败都会通过linux的sendmail发送邮件提醒，并附加构建日志
- 8.构建的代码会通过sonarqube插件上传sonarqube进行代码审查

系统图



GitLab Enterprise Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Sign in	Register
 Username or email <input type="text"/>	
 Password <input type="password"/>	
<input type="checkbox"/> Remember me	Forgot your password?
Sign in	

The dashboard shows a grid of build status icons (S, W, Name) and a list of recent builds:

S	W	Name	上次成功
●	●		1天 0 小时 - #5
●	●		3天 16 小时 - #6
●	●		3天 18 小时 - #7
●	●		3天 22 小时 - #3
●	●		3天 22 小时 - #16
●	●		23 小时 - #1
●	●		1天 0 小时 - #2
●	●		1天 0 小时 - #1
●	●		23 小时 - #4
●	●		23 小时 - #1
●	●		23 小时 - #2
●	●		3天 17 小时 - #2

图标: 尘虫大

构建通知 : [REDACTED] - Build # 1 - Successful! ★

[REDACTED]
发着了 [REDACTED]

build.log (22 KB)

(本邮件是程序自动下发的, 请勿回复!)

构建结果 - Successful

构建信息

- 项目名称 : [REDACTED]
- 构建编号 : 第1次构建
- 构建人 : [REDACTED] master
- 触发原因: Started by GitLab push by jiangwd
- 构建日志: [http://192.168.8.30/job/\[REDACTED\]/1/console](http://192.168.8.30/job/[REDACTED]/1/console)
- 构建 Url: [http://192.168.8.30/job/\[REDACTED\]/1/](http://192.168.8.30/job/[REDACTED]/1/)
- 工作目录: [http://192.168.8.30/job/\[REDACTED\]/ws](http://192.168.8.30/job/[REDACTED]/ws)
- 项目 Url: [http://192.168.8.30/job/\[REDACTED\]/api/](http://192.168.8.30/job/[REDACTED]/api/)
- 代码审查: [http://192.168.8.100:9002/review/\[REDACTED\]](http://192.168.8.100:9002/review/[REDACTED])

Changes Since Last Successful Build:

- 历史变更记录 : [http://192.168.8.30/job/\[REDACTED\]/changes](http://192.168.8.30/job/[REDACTED]/changes)

Failed Test Results

No tests ran.

构建日志 (最后 100行):

```
[...truncated 15.50 KB...]
INFO: Sensor SonarCSS Metrics [cssfamily] (done) | time=95ms
INFO: Sensor SonarCSS Rules [cssfamily]
INFO: Sensor SonarCSS Rules [cssfamily] (done) | time=1256ms
INFO: Sensor JaCoCo XML Report Importer [jacoco]
INFO: Sensor JaCoCo XML Report Importer [jacoco] (done) | time=3ms
INFO: Sensor SonarJS [javascript]
INFO: 68 source files to be analyzed
ERROR: Unable to parse file: file:///home/jenkins/.jenkins/workspace/[REDACTED]
ERROR: Parse error at line 43 column 10:
33:     debug
```

闲鱼@jxd806089486

Maven project [redacted]

需要如下参数用于构建项目:

Tag	rc_5 rc_42 rc_41
-----	------------------------

上次构建成功的Tag

如需发布新版本则点击“开始构建”。如需回退版本，请选择需要回退的Tag

开始构建

闲鱼@jxd806089486

Jenkins > [redacted] >

Maven project [redacted]

返回面板 状态 修改记录 工作空间 Build with Parameters 删除 Maven project 配置 模块 Email Template Testing SonarQube 重命名

SonarQube 工作区 最新修改

SonarQube Quality Gate

OK server-side processing: Success

相关链接

- [Last build\(#5\).1天 0 小时之前](#)
- [Last stable build\(#5\).1天 0 小时之前](#)
- [Last successful build\(#5\).1天 0 小时之前](#)
- [Last failed build\(#3\).1天 1 小时之前](#)
- [Last unsuccessful build\(#3\).1天 1 小时之前](#)
- [Last completed build\(#5\).1天 0 小时之前](#)

闲鱼@jxd806089486

Build History

#	构建时间	状态
#5	2019-9-23 上午10:21	Started by GitLab push by [redacted]
#4	2019-9-23 上午9:55	
#3	2019-9-23 上午9:50	
#2	2019-9-23 上午9:48	Started by GitLab push by [redacted]
#1	2019-9-20 下午5:33	Started by GitLab push by jianyed

SonarQube 项目 问题 代码规则 质量配置 质量阈 配置

最近一次分析出

总览 问题 指标 代码 活动 配置

质量阈 正常

因为代码太少，有些新代码中的质量阈条件会被忽略。

可靠性 指标

7 D 起始于 4 天前	活动 0 A 新增 Bugs
---	--

安全性 指标

15 E 漏洞	9 B 安全热点	0 A 新增漏洞	0 B 新安全热点
---	---	--	--

可维护性 指标

4天 A 债务	241 B 异味	0 A 新债务	0 B 新增代码异味
---	---	---	---

覆盖率 指标

0.0% C 覆盖率	0.0% B 覆盖 1 覆盖的新代码行数
--	--

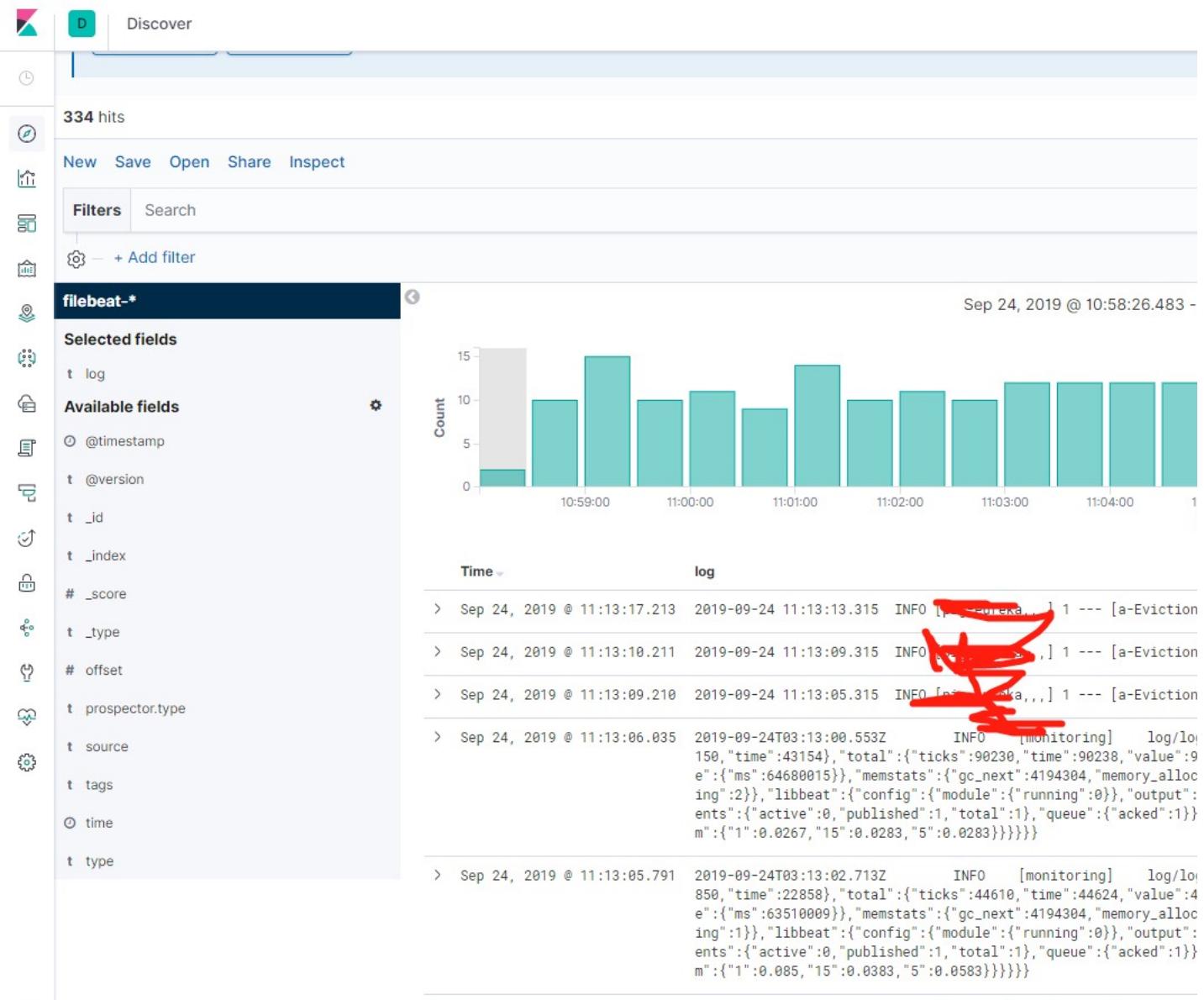
闲鱼@jxd

Nexus Repository Manager OSS 3.6.0-02

Components / java

Group ↑

- org.springframework.boot



EasyCI系统架构

工具链

EasyCI : easyci系统后台，调用shell脚本、通过接口将结果返回前端

EasyCI-UI : easyci系统前台，vue。调用后台接口展示数据

Gitlab : 代码托管工具，通过gitlab hook触发easyci持续集成交付

shell脚本 : linux脚本

Docker私服 : 私有docker镜像仓库，用于远程构建

MySQL : easyci系统数据库

流程

1.根据安装文档安装系统，启动系统

2.访问系统首先需要验证gitlab : gitlab的url、用户名和密码。用于选择项目构建、服务器拉取代码验证、自动添加hook，免去手动操作。

3.添加远程发布服务器：服务器ip、端口、用户名、密码。后台自动完成服务器间的免密登录，用于项目远程发布、查看服务器容器以及容器的各种操作。

4.部署：选择项目url、输入docker容器端口映射关系、选择项目类型、输入收件人邮箱、选择部署服务器，只需要输入两项内容即可完成部署。

5.点击部署，会弹出窗口显示系统部署日志。

6.部署会自动添加easyci的hook接口url到该部署项目的hook中，用于自动触发构建。

7.部署完成自动发送邮件给收件人邮箱，显示构建结果、构建时间、构建项目、构建日志。

8.后续开发人员提交代码到该项目的gitlab，会触发gitlab的hook调用easyCi接口，查询之前部署的信息进行自动部署，实现持续集成。

9.页面展示easyCi本机正在运行的容器和添加的远程服务器正在运行的容器，支持数据自动刷新与关闭

10.可以选择添加的远程服务器，查看该服务器的容器列表

11.容器操作：可点击启动、停止、重启、销毁在页面对服务器中运行的容器进行操作，即docker start|stop|restart|rm 容器名称

12.容器实时日志：点击日志，可以查看该容器的实时日志，即docker logs -f 容器名称

安装教程

1.系统准备

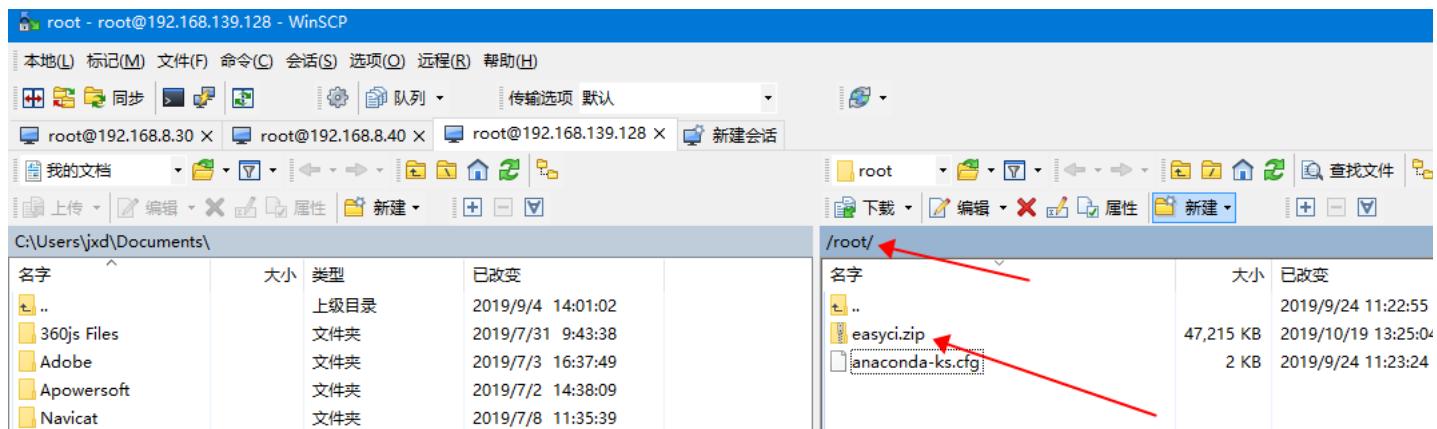
- 本系统仅支持Centos7系统
- 配置2H4G
- 固定好ip

1.在公司服务器中分出一个虚拟机作为easyCi的运行环境，要求如上图所示。

注意：该虚拟机最好不要运行其他服务，因为easyCi会先在本地运行容器测试部署结果，再发送到远程服务器运行容器，要注意端口占用。

2.将easyCi.zip通过winscp或者xftp等。

这里我放到/root目录下。



3.通过xshell或者其他工具连接服务器。

```
Connecting to 192.168.139.128:22...
Connection established.
To escape to local shell, press 'Ctrl+Alt+]'.
WARNING! The remote SSH server rejected X11 forwarding request.
Last login: Sat Oct 19 12:26:51 2019 from 192.168.139.1
[root@localhost ~]#
```

4.进入/root目录，将easyCi.zip解压

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls -l
总用量 47220
-rw----- 1 root root 1244 9月 24 11:23 anaconda-ks.cfg
-rw-r--r-- 1 root root 48347652 10月 19 13:25 easyCi.zip
[root@localhost ~]# unzip easyCi.zip
-bash: unzip: 未找到命令
[root@localhost ~]# yum install -y unzip
```

如果解压unzip没有命令，可以执行 `yum install -y unzip` 安装

再次解压。解压后生成easyCi文件夹。

```
[root@localhost ~]# pwd
/root
[root@localhost ~]# ls -l
总用量 47220
drwxr-xr-x 4 root root 89 10月 18 14:33 easyCi
-rw-r--r-- 1 root root 48347652 10月 19 13:25 easyCi.zip
```

5.进入easyCi文件夹

```
[root@localhost easyci]# pwd
/root/easyci
[root@localhost easyci]# ls -l
总用量 16
-rw-r--r-- 1 root root 5533 10月 19 12:23 install.sh 一键安装环境脚本
drwxr-xr-x 2 root root 200 10月 17 09:46 sh sh文件夹里是系统shell脚本
-rw-r--r-- 1 root root 166 10月 18 15:15 startEasyCi.sh 系统运行脚本
-rw-r--r-- 1 root root 245 10月 18 15:15 stopEasyCi.sh 系统停止脚本
drwxr-xr-x 3 root root 84 10月 18 14:45 work
```

6.编辑install.sh，修改第35行中的localhost修改为本机ip

- vi install

按住“*i*”进入编辑，修改localhost为本机ip

```
echo 'docker daemon配置文件修改'
echo { '"registry-mirrors":'["https://registry.docker-cn.com"],'"insecure-registries":'["localhost:5000"] } > /etc/docker/daemon.json
systemctl daemon-reload
echo 'docker服务重启'
systemctl restart docker
echo '验证docker私服'
```



修改完成后，按“Esc”退出编辑状态，按“：“输入wq回车，保存。

```
docker ps
echo '5.安装docker-compose'
#curl -L https://github.com/docker/compose/releases/download/1.25.4/docker-compose-$(uname -s)-$(uname -m)
#sudo chmod +x /usr/local/bin/docker-compose
#echo '验证docker-compose'
#docker-compose -v

echo '6.安装git'
yum install -y git
echo '验证git'
git --version
:wq
```



该install脚本为easyci系统部署所需要所有工具的安装，可按需调整。

7.运行install脚本，进行系统安装

- bash install.sh

系统安装会持续20分钟左右，具体看网速。

出现下图，表示系统所需工具初步安装完成。

```
=====系统安装完成=====
=====访问路径为http://本机ip:81=====
=====本系统日志路径为/home/easyci/nohup.out=====
=====项目部署日志路径为/home/log/=====
=====项目工作路径为/home/workspace/=====
=====前端静态文件存放路径为/usr/share/nginx/html=====
=====nginx配置文件路径为/etc/nginx/nginx.conf=====
=====数据库连接为127.0.0.1::3306=====
=====本机占用端口为3306、9875、5000、22、25、323、81=====
=====由于本系统需要在本地打包测试容器运行情况=====
=====容器运行良好再发布到其他服务器=====
=====请不要使用上述端口=====
+-----+
+-----+
+-----+系统配置文件路径为/home/easyci+++++
+-----+系统开启/home/easyci/startEasyCi.sh+++++
+-----+系统关闭/home/easyci/stopEasyCi.sh+++++
+-----+
+-----+
+-----+ ! ! ! ! ! ! ! ! 系统配置文件不可删除 ! ! ! ! ! ! !
+-----+ ! ! ! ! ! ! ! ! 系统配置文件不可删除 ! ! ! ! ! ! !
+-----+ ! ! ! ! ! ! ! ! 系统配置文件不可删除 ! ! ! ! ! ! !
+-----+
```

8.重载/etc/profile配置文件。

- source /etc/profile

测试java、maven、node等命令是否正常安装。

- java -version
- mvn -v
- node -v
- npm -v

```
[root@localhost easyci]# source /etc/profile
[root@localhost easyci]# java -version
java version "1.8.0_131"
Java(TM) SE Runtime Environment (build 1.8.0_131-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.131-b11, mixed mode)
[root@localhost easyci]# mvn -v
Apache Maven 3.6.1 (d66c90b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-05T03:00:29+08:00)
Maven home: /home/maven
Java version: 1.8.0_131, vendor: Oracle Corporation, runtime: /home/jdk/jre
Default locale: zh_CN, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-957.el7.x86_64", arch: "amd64", family: "unix"
[root@localhost easyci]# node -v
v9.8.0
[root@localhost easyci]# npm -v
5.6.0
```

ok,工具按住完成。

9.永久关闭selinux

- vi /etc/selinux/config

将SELINUX=enforcing改为SELINUX=disabled

设置后需要**重启**才能生效。

10.开放mysql的root用户远程访问。

- mysql -hlocalhost -uroot -p123456
- grant all privileges on *.* to root@ "%" identified by ".";
- flush privileges;
- exit

```
[root@localhost easyci]# mysql -hlocalhost -uroot -p123456
Warning: Using a password on the command line interface can be insecu
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.6.46 MySQL Community Server (GPL)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights res

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help:' or '\h' for help. Type '\c' to clear the current input st

mysql> grant all privileges on *.* to root@ "%" identified by ".";
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> exit
Bye
```

11.启动系统

```
[root@localhost easyci]# pwd
/home/easyCi
[root@localhost easyci]# ls -l
总用量 50628
-rw-r--r-- 1 root root 51832508 10月 19 14:01 ci-0.0.1-SNAPSHOT.jar jar是java服务
drwxr-xr-x 2 root root 200 10月 19 14:01 sh      sh文件夹存放的shell脚本
-rwxr-xr-x 1 root root 166 10月 19 14:01 startEasyCi.sh
-rwxr-xr-x 1 root root 245 10月 19 14:01 stopEasyCi.sh
[root@localhost easyci]#                                         启动和停止系统脚本
[root@localhost easyci]#
```

进入easyCi系统家目录/home/easyCi

执行./startEasyCi.sh启动系统

```
[root@localhost easyci]# ./startEasyCi.sh
启动web
启动后台
启动数据库
nohup: 重定向标准错误到标准输出
[root@localhost easyci]# netstat -tunlp | grep -E "9875|81"
tcp        0      0 0.0.0.0:81          0.0.0.0:*          LISTEN      63087/nginx: master
tcp6       0      0 ::1:9875           ::*:*              LISTEN      63090/java
```

前端81端口

后端9875端口

- netstat -tunlp | grep -E "9875|81"

查看前端81端口和后端9875端口是否开启成功。

使用教程

1.访问系统 浏览器打开 ip:81

系统启动首先需要验证gitlab。

gitlaburl : 输入gitlab的地址，不要加http :

用户名密码：建议用户名和密码输入管理员root的用户名和密码。

注：这里的密码不要包含@ / 等特殊符号，否则免认证拉取git代码会失败

此处输入点击登录，后台会调用gitlab api拿到access-token，用于在服务器中拉取免密拉取代码、部署时选择构建的项目。

2.添加远程部署服务器

输入信息点击添加，系统会生成本服务器的公钥，添加到该服务器的认证文件中实现免密登陆。

添加完成之后会展示该服务器的docker容器列表，可以进行启动、停止、重启、销毁、查看实时日志操作。

添加服务器完成，会显示本机的容器列表和添加服务器的容器列表。

请选择服务器	<input type="button" value="新增"/>	<input type="button" value="查看"/>	<input type="button" value="重置"/>	<input checked="" type="checkbox" value="开启自动刷新"/> 开启自动刷新			
构建项目	<input type="button" value="端口5000:5000"/>	项目语言	<input type="button" value="▼"/>	请输入收件人邮箱	部署方式	<input type="button" value="▼"/>	<input type="button" value="部署"/>

本机容器列表

容器id	镜像名称	状态	创建时间	端口映射	容器名称	IP
89fcbea867c2	registry:latest		27 minutes ago	0.0.0.0:5000->5000/tcp	qdockerhub	本机

服务器容器列表

容器id	镜像名称	状态	创建时间	端口映射	容器名称	IP
8614029e6404	zhangyuming/elasticproxy		3 weeks ago	0.0.0.0:8899->8899/tcp	elasticproxy	192.168.1.11:8899
8a7e8cef453e	dockerelk_kibana		3 weeks ago	0.0.0.0:5601->5601/tcp	dockerelk_kibana_1	192.168.1.11:5601
bdf57283e519	dockerelk_logstash		3 weeks ago	0.0.0.0:5000->5000/tcp, 0.0.0.0:9600->9600/tcp, 5044/tcp	dockerelk_logstash_1	192.168.1.11:5000
2c13a2c341ca	dockerelk_elasticsearch		3 weeks ago	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	dockerelk_elasticsearch_1	192.168.1.11:9200
31749410acdf	prima/filebeat		3 weeks ago		elkfilebeat_filebeat_1	192.168.1.11:5044

如果有多个远程服务器，可以选择某个服务器，查看该服务器的容器列表，也可以开启自动刷新，实时刷新容器数据。

The screenshot shows a top navigation bar with tabs for '新增' (Add), '查看' (View), '重置' (Reset), and '开启自动刷新' (Enable automatic refresh). Below this is a search bar with fields for '端口5000:5000' (Port 5000:5000), '项目语言' (Project Language) set to 'Java', and '请输入收件人邮箱' (Enter recipient email). There are dropdowns for '部署方式' (Deployment method) and a '部署' (Deploy) button. A red arrow points to the '查看' (View) tab.

容器id	镜像名称	状态	创建时间	端口映射	容器名称
89fcbea867c2	registry:latest		30 minutes ago	0.0.0.0:5000->5000/tcp	qdockerhub

服务器容器列表

容器id	镜像名称	状态	创建时间	端口映射	容器名称
8614029e6404	zhangyuming/elasticproxy		3 weeks ago	0.0.0.0:8899->8899/tcp	elasticproxy
8a7e8cef453e	dockerelk_kibana		3 weeks ago	0.0.0.0:5601->5601/tcp	dockerelk_ki
bdf57283e519	dockerelk_logstash		3 weeks ago	0.0.0.0:5000->5000/tcp, 0.0.0.0:9600->9600/tcp, 5044/tcp	dockerelk_lc_1
2c13a2c341ca	dockerelk_elasticsearch		3 weeks ago	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	dockerelk_e_rch_1
31749410acdf	prima/filebeat		3 weeks ago		elkfilebeat_fi
c50de1a6f0d4	prima/filebeat		3 weeks ago		elkfilebeat_fi
141849ceea2	mobz/elasticsearch-head:5		3 weeks ago	0.0.0.0:9000->9100/tcp	es_admin
af895f4ddc62	docker.elastic.co/elasticsearch/elasticsearch:6.3.2		3 weeks ago	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	es

“启动 | 停止 | 重启 | 销毁” 等操作同步 docker start | stop | restart | rm 容器名称。

“日志” 可以查看该服务器的实时日志，即 docker logs -f -tail 200 容器名称。查看该容器近200行日志，并实时刷新。

af895f4ddc62	docker.elastic.co/elasticsearch/elasticsearch:6.3.2		3 weeks ago	0.0.0.0:9200->9200/tcp, 0.0.0.0:9300->9300/tcp	es
67b0ac405616	registry:latest		4 weeks ago	0.0.0.0:5000->5000/tcp	qdockerhub
a3b	实时日志				
b65	<pre>2019-10-19 01:21:00,098 INFO (ServletContextImpl.java:364)- Initializing Spring DispatcherServlet 'dispatcherServlet' 2019-10-19 01:21:00,099 INFO (FrameworkServlet.java:521)- Initializing Servlet 'dispatcherServlet' 2019-10-19 01:21:00,124 INFO (FrameworkServlet.java:543)- Completed initialization in 25 ms 2019-10-19 01:21:00,187 DEBUG (JWTFilter.java:89)- Authentication required: sending 401 Authentication challenge response. 2019-10-19 01:21:00,249 DEBUG (JWTFilter.java:89)- Authentication required: sending 401 Authentication challenge response. 2019-10-19 01:50:10,870 INFO (CacheTask.java:29)- delete expired user 2019-10-19 02:37:31,982 DEBUG (BaseJdbcLogger.java:159)- ==> Preparing: SELECT count(0) FROM m_news 2019-10-19 02:37:31,983 DEBUG (BaseJdbcLogger.java:159)- ==> Parameters: 2019-10-19 02:37:31,986 DEBUG (BaseJdbcLogger.java:159)- <== Total: 1 2019-10-19 02:37:31,989 DEBUG (BaseJdbcLogger.java:159)- ==> Preparing: select * from m_news ORDER BY year desc,month desc,day desc LI 2019-10-19 02:37:31,990 DEBUG (BaseJdbcLogger.java:159)- ==> Parameters: 3(Integer) 2019-10-19 02:37:31,992 DEBUG (BaseJdbcLogger.java:159)- <== Total: 1 2019-10-19 02:50:10,871 INFO (CacheTask.java:29)- delete expired user 2019-10-19 03:50:10,870 INFO (CacheTask.java:29)- delete expired user 2019-10-19 04:50:10,871 INFO (CacheTask.java:29)- delete expired user 2019-10-19 05:50:10,872 INFO (CacheTask.java:29)- delete expired user 2019-10-19 06:32:44,325 DEBUG (BaseJdbcLogger.java:159)- ==> Preparing: SELECT count(0) FROM m_news 2019-10-19 06:32:44,327 DEBUG (BaseJdbcLogger.java:159)- ==> Parameters: 2019-10-19 06:32:44,330 DEBUG (BaseJdbcLogger.java:159)- <== Total: 1 2019-10-19 06:32:44,331 DEBUG (BaseJdbcLogger.java:159)- ==> Preparing: select * from m_news ORDER BY year desc,month desc,day desc LI 2019-10-19 06:32:44,331 DEBUG (BaseJdbcLogger.java:159)- ==> Parameters: 3(Integer) 2019-10-19 06:32:44,333 DEBUG (BaseJdbcLogger.java:159)- <== Total: 1 2019-10-19 06:50:10,871 INFO (CacheTask.java:29)- delete expired user 2019-10-19 07:50:10,869 INFO (CacheTask.java:29)- delete expired user 2019-10-19 08:50:10,872 INFO (CacheTask.java:29)- delete expired user 2019-10-19 09:50:10,871 INFO (CacheTask.java:29)- delete expired user 2019-10-19 10:50:10,871 INFO (CacheTask.java:29)- delete expired user 2019-10-19 11:26:40,847 DEBUG (JWTFilter.java:89)- Authentication required: sending 401 Authentication challenge response. 2019-10-19 11:50:10,873 INFO (CacheTask.java:29)- delete expired user 2019-10-19 12:50:10,870 INFO (CacheTask.java:29)- delete expired user 2019-10-19 13:30:52,772 DEBUG (JWTFilter.java:89)- Authentication required: sending 401 Authentication challenge response. 2019-10-19 13:50:10,870 INFO (CacheTask.java:29)- delete expired user</pre>				
c64					
4b1					
609					
e42					
b6e					
a8f					
5d3					
3b2					
c73					

3.项目构建

点击选择需要部署的gitlab醒目

The screenshot shows the GitLab CI interface for a project named 'easyci'. It includes fields for selecting a server, port (5000), language (Java), recipient email, deployment method, and a deployment table.

服务器名称	服务器	操作
admin@verhuu	本地	

Below the table, a list of pipelines is shown:

- http://192.168.8.10:5000/easyci/ping-ui.git
- http://192.168.8.10:5000/easyci/ping-auth.git
- http://192.168.8.10:5000/easyci/ping-auth.git
- http://192.168.8.10:5000/easyci/ping-auth.git

端口输入端口映射对。

由于是采用docker容器运行。需要设置映射端口

例如6000:5000

前面的6000为宿主机对外访问开放的端口，后面的5000为构建项目的端口，注意“：“为英文状态下

项目语言选择vue或者java。

收件人邮箱：如果有多个以“，”分割

部署方式：选择要部署的服务器。

注意：本系统是在本地先运行测试容器是否正常运行。运行正常则本地会删除容器释放端口，再发布到远程服务器运行。

所以注意本地端口占用。系统初始端口如下：

22、25、81、5000、3306、9875、68、323

即端口映射前面的端口不能为上述端口。

```
[root@localhost easyci]# netstat -tunlp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address          Foreign Address        State      PID/Program name
tcp     0      0 127.0.0.1:25            0.0.0.0:*              LISTEN    1330/master
tcp     0      0 0.0.0.0:81             0.0.0.0:*              LISTEN    1954/nginx: master
tcp     0      0 0.0.0.0:22             0.0.0.0:*              LISTEN    989/sshd
tcp6    0      0 ::1:25                ::*:*                  LISTEN    1330/master
tcp6    0      0 ::5000                ::*:*                  LISTEN    1371/docker-proxy
tcp6    0      0 ::3306                ::*:*                  LISTEN    1724/mysql
tcp6    0      0 ::9875                ::*:*                  LISTEN    1957/java
tcp6    0      0 ::22                 ::*:*                  LISTEN    989/sshd
udp     0      0 0.0.0.0:68             0.0.0.0:*              LISTEN    792/dhclient
udp     0      0 127.0.0.1:323           0.0.0.0:*              LISTEN    739/chronyd
udp6   0      0 ::1:323               ::*:*                  LISTEN    739/chronyd
```

如果本地或者远程端口被占用，会停止构建，弹窗提醒。

这里easyci本地运行的docker私服占用了5000端口，再发布5000端口的宿主映射会提示“端口已被占用”



本机容器列表

容器id	镜像名称	状态	创建时间	端口映射	容器名称
89fcbea867c2	registry:latest		39 minutes ago	0.0.0.0:5000->5000/tcp	qdockerhub

换其他端口，开始部署，显示部署日志

部署日志

```
docker容器名称为: pig-ui
docker容器端口为5000
映射宿主机端口为5001
构建项目日志文件为: /home/logs/pig-ui-project.log
构建项目工作目录为: /home/workspace/pig-ui
收件人邮箱为: jiangxiao@mingbytc.com
部署服务器为: 192.168.8.10
```

服务器容器列表

容器id	镜像名称
c50de1a6f0d4	prima/filebeat
141849ceeeae2	mobz/elasticsearch-head:5
af895f4ddc62	docker.elastic.co/elasticsearch
67b0ac405616	registry:latest

部署日志

```
docker容器名称为: pig-ui
docker容器端口为5000
映射宿主机端口为5001
构建项目日志文件为: /home/logs/pig-ui-project.log
构建项目工作目录为: /home/workspace/pig-ui
收件人邮箱为: jiangxiao@mingbytc.com
部署服务器为: 192.168.8.10

=====删除旧的构建=====
=====删除旧容器=====
=====删除无用镜像=====

Untagged: 192.168.8.40:5000/pig-ui@sha256:a7e0db5ec3514731b5ab72377a60070a18d2e2e5d7a11ea7904db96f98
Deleted: sha256:07f1c0e180dd47ab95aa4a8fe46008f1ba44be1fbe798d10f1176cbe2d2525df
Deleted: sha256:4fb1ca3a477aecf8f30a7c1f0f5664868846c34aa37ee65230a6e0d850a6ae0d
Deleted: sha256:ec4a171b621aa57c1d0e6a351af75454111731ca6eee9c451d70b2f2d7dae5cd
Deleted: sha256:ae4bf519ef5909ce18021dd0ea87e5858f4be305e63c67ea513c043291508a3
Deleted: sha256:bbcfec2ee2df65d9be2e7287a53ac7f29a03da5ba5845cf3e1d3c9f352155f3ed
=====删除多余tag, 保留最新三个=====
```

容器部署成功会有邮件提醒。

pig-ui	5001:5000	vue	192.168.8.10	部署
--------	-----------	-----	--------------	----

本机容器列表

容器id	镜像名称
31e1a0aeb96e	registry:latest

服务器容器列表

容器id	镜像名称
c50de1a6f0d4	prima/filebeat
141849ceeeae2	mobz/elasticsearch-head:5
af895f4ddc62	docker.elastic.co/elasticsearch
67b0ac405616	registry:latest

部署日志

```

873004793757: Pull complete
dd8e4ce2ad6b: Pull complete
927685055a4e: Pull complete
Digest: sha256:6e032be86bacbb6e2faea0cf7be5f7d5d7808c652d4a5685f36498e9c5e210e0
Status: Downloaded newer image for 192.168.8.40:5000/pig-ui:latest
拉取镜像成功
=====删除旧容器=====
=====删除无用镜像=====
Untagged: 192.168.8.40:5000/pig-ui:latest
Untagged: 192.168.8.40:5000/pig-ui@sha256:6e032be86bacbb6e2faea0cf7be5f7d5d7808c652d4a5685f36498e9c5e:
Deleted: sha256:cd595c6bd243e6b0e09c119e94051457a7e376a91edf9e4ad826b988a327efb6
Deleted: sha256:9ec5d42a1ee6562bec2c1e721f8d0628432f1e6033fb04a8ed6f1f880ce88dc2
Deleted: sha256:1d89b4d1861cf1e1b8ec5a7ae5eb544b1e34b950e696bc28de199d1d6008ad5f
Deleted: sha256:92a2fd5d4ab25f3f58fd05274ac91b2cfdf31aaea31464911e035fec0c5acde
Deleted: sha256:ec8273278e40e8ec1bac734baf584fa354723bca65781ce549968c83218da9e2
afd99effb7c9d4d0d5a3a7ac36ab745868bdf6d75f1d36c7347c9f6c681564ec
退出远程服务器
邮件发送成功
容器远程部署成功

```

部署日志

