解决方案实践

中软国际教育科研云与人才培养解决方案实践

文档版本 1.1

发布日期 2024-05-08

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

安全声明

漏洞处理流程

华为公司对产品漏洞管理的规定以“漏洞处理流程”为准，该流程的详细内容请参见如下网址：
https://www.huawei.com/cn/psirt/vul-response-process
如企业客户须获取漏洞信息，请参见如下网址：
https://securitybulletin.huawei.com/enterprise/cn/security-advisory

目 录

1 方案概述

应用场景

- **场景介绍**

  企业助力高校培养契合企业需求的人才，反向推动产业发展，基于华为在ICT产业和技术的先进优势，以云+网+应用模式，联合多家合作伙伴基于云计算、大数据、人工智能等创新技术，为院校培养ICT教育人才提供实践云平台和相关服务，提供ICT行业各个技术方向的优质教学资源和高效实践环境及服务，打造产业协同、产教融合、产学合作的ICT人才新模式

- **场景趋势**

  人才培养领域，学校为培养校企合作成为趋势。党的十九大指出：实现高等教育内涵式发展，深化产教融合、产学研结合、校企合作是高等教育，特别是应用型高等教育发展的必经之路。高校应积极响应国家发展战略转型，立足产业发展对云计算，大数据，人工智能，数字内容等领域人才培养的需要，探索人才培养范式，推动学科发展，瞄准世界科技前沿，为我国新一代新兴领域人才发展提供战略支撑。高校应坚持以经济社会发展需要为导向，深化职业教育、高等教育改革，加快建设实体经济、科技创新、现代金融、人力资源协同发展的产业体系，加速产教融合协同人才培养速度，紧密对接经济带、城市群、产业链布局，全面深化综合改革，优化高校专业结构，助力产业转型升级

- **解决方案：智云枢平台**

  智云枢平台是中软国际教育科技集团持续打造的核心产品，以能力驱动为教学理念，岗位为导向“岗课赛证”一体化人才培养教学平台，以岗位目标为培养方向，帮助学生拓宽认知，明确方向。平台模拟真实工作场景，锻炼学生项目实操能力，使教学过程/成果可视化，它由混合式教学、在线实验和项目实训三大子系统组成，公共模块为各个子系统提供基础功能。

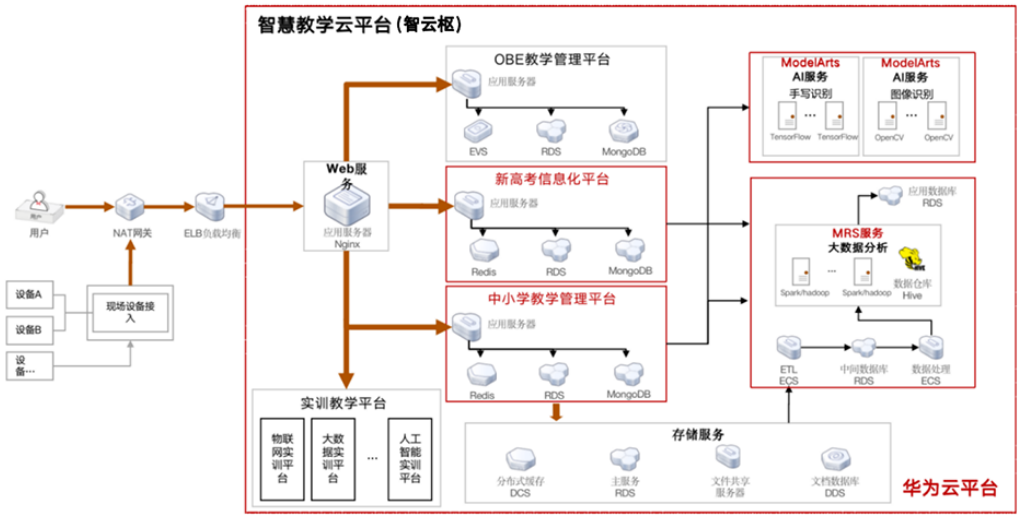
  a. 在混合式教学中，教师可以灵活设计课前-课中-课后的各个教学环节，学生按照教学设计参与学习活动。支持Web浏览器和微信小程序多端互动教学，提升学习体验。针对不同类型的学生或班级，可以通过分层教学模式，设置不同的教学内容和作业，满足个性化教学需求。系统通过数据看板，还提供了详尽的学习行为和结果数据统计分析，帮助教师和学生及时了解当前学情。

  b. 在线实验系统支持多种交互实验方式，例如WebIDE形式、命令行形式、Jupyter Notebook形式，以及完整桌面操作系统形式。可以有效地支持系统管理、编程开发、大数据、人工智能等多种类型的实验。部分实验实现了结果的自动验证和评分，降低了教师的负担。

c. 项目实训系统支持项目阶段的划分，并针对每个阶段分别设置考核内容、考核方式和评分标准。系统提供了完整的项目管理功能，能够为团队和个人制定项目计划和任务分工，并全程跟踪任务执行进度。每个项目组将获得一个代码仓库，并且对成员提交的代码进行扫描并评估其质量。系统还支持申请云上虚拟机环境，以便项目的快速部署、运行、测试及演示。

方案架构

图 1-1 方案部署架构



1. 用户可以多终端介入系统，支持：微信小程序、微信公众号、移动APP、校园门户、IOC门户。
2. 使用ECS云服务器搭建集群，使用ELB负载均衡满足用户高并发的访问需求。
3. OBE教学管理平台（智云衡）、新高考信息化平台、中小学教学管理平台等都是采用插拔机制和智云枢大平台进行整合。
4. 智云枢底层存储服务采用分布式缓存、文件共享服务、RDS、文档数据库DDS华为云高阶服务，以保证系统数据安全、可靠和高效。
5. 在线实验和在线实训功能，不同专业的实验案例和实训项目案例会议使用到华为云相应的高阶服务，如：人工智能专业学生在实验和实训时，基于ModeArts平台在线开发，系统会调用AI高阶服务：手写识别、图像识别、活体识别等等。
6. 课程知识图谱、学生能力画像(多级钻取)、自适应测验、自适应个性学习、自动学习路径等功能，依据MapReduce服务进行海量数据抓取和分析，使用已经训练的学生模型匹配，然后给出最后的推荐学习建议和学习路径，以及学生结业后的360度全方位画像。

技术特点

1. 平台支持国内多种直播平台，系统也内嵌WEBRTC直播，满足公有云和私有云直播；
2. 平台提供课堂直播和课堂重现，方便学生课后复习；
3. 平台实验支持逻辑编程题实验、数据库实验和web前端实验，满足基础课程教学；

4. 产品同时支持华为公有云和私有云环境，通过对云平台的基础层构建以及平台层的虚拟化构建，为学生提供稳定可靠的云实验和云实训环境；
5. 针对云实验环境，系统支持桌面或字符界面虚拟机、Docker终端、命令行终端、Jupyter Notebook等实验环境，并支持多虚拟机之间进行切换；
6. 平台实训体现“做中学、学中做”的应用理念，学生可以按照平台提供任务步骤Step By Step来执行，同时系统提供良好的视频资源进行支撑，为学生多方位技术支持与锻炼；
7. 云实训支持学生的代码版本管理、代码质量检查、代码贡献率统计，满足高校计算机类实训；
8. 系统云实训严格按照CMMI5级标准对项目的全流程进行监控管理,保障学生按照企业软件开发流程和规范进行项目实践、提交和验收。

产品优势

智云枢是为服务IT类的人才培养过程而设计的，产品主要特点包括以下内容：

- **行业前沿课程体系**

  课程资源涵盖公开课、岗位课、直播课、实验案例、项目案例等。其中公共课资源涵盖有：前端开发、Java、人工智能、数据库、云计算、大数据、物联网、PHP、C_嵌入式、UXD、项目管理、测试、鲲鹏训练营、鲲鹏云-HCIA、GaussDB-HCIP、等课程方向，可以满足多方向学生的学习要求。

- **真实的项目案例**

  智云枢具有丰富的真实项目案例，均是由中软国际卓越研究院根据企业真实项目案例开发转化而来，如银行用户画像系统、基于python的电商产品评论数据情感分析、地铁一卡通大数据可视化系统等等。

- **全闭环教学过程管理**

  智云枢能够辅助高校教师实现信息化教学，功能涵盖学生管理、班级管理、教学资源管理、在线备课、在线直播、在线学习、课后作业、在线实验、在线考试、在线实训、考评结果、成绩统计、数据分析等，从而实现教学过程全面管理，提升教师工作效率。

- **支持多类实验/实训环境**

  智云枢可以创建虚拟机模板包括：Openstack虚拟机模板、Docker字符虚拟机模板、Docker图形界面模板、AI实训环境模板、WebIDE编辑代码模板。平台能够支撑大数据专业、人工智能专业、软件工程专业及其它计算机相关专业的实验环境要求。

- **安装部署方便灵活**

  智云枢可根据高校需求，支持本地化私有云部署，同时也支持华为云公有云部署。本地化部署需要高校提供硬件服务器环境，公有云部署需要高校提供华为云云资源环境。

- **系统维护灵活、扩张性**

  项目设计遵循以下原则，为后期的项目维护灵活性作保证以及便于服务拓展。

2 资源和成本规划

表 2-1 智云枢平台所需资源清单

云资源	规格	数量	每年费用（元）
平台web服务器ECS	平台web服务器ECS	4	13,020.00
平台直播服务器ECS	1. 鲲鹏通用计算增强型：kc1.4xlarge.2 16vCPUs | 32GB 2. CPU：Huawei Kunpeng 920 3. 基准带宽：6Gbit/s 4. 最大带宽：12Gbit/s 5. 网内收发包能力：140万	1	13,759.00
静态资源服务器ECS	1. X86通用计算增强型：c6.2xlarge.4 16vCPUs | 32GB 2. CPU：Intel 2.6GHz以上 3. 基准带宽：6Gbit/s 4. 最大带宽：12Gbit/s 5. 网内收发包能力：140万	1	7,260.00
RDS for MySQL主备	1. 鲲鹏通用计算增强型：kc1.2xlarge.2 8vCPUs | 16GB 2. CPU：Huawei Kunpeng 920 2.6GHz 3. 基准带宽：3Gbit/s 4. 最大带宽：7Gbit/s 5. 网内收发包能力：80万	2	25,080.00
Redis	1. 鲲鹏通用增强型：kc1.2xlarge.4 8vCPUs | 32GB 2. 最大链接：8000	1	3,841.00
GitLab服务器ECS	主备|8G|ARM|DRAM	1	7,050.00

云资源	规格	数量	每年费用（元）
Sonar服务器ECS	1. 鲲鹏通用计算增强型：kc1.2xlarge.2 8vCPUs | 16GB 2. CPU：Huawei Kunpeng 920 3. 基准带宽：6Gbit/s 4. 最大带宽：12Gbit/s 5. 网内收发包能力：140万	1	7,050.00
NAT网关	1. 鲲鹏通用计算增强型：kc1.2xlarge.2 8vCPUs | 16GB 2. CPU：Huawei Kunpeng 920 3. 基准带宽：6Gbit/s 4. 最大带宽：12Gbit/s 5. 网内收发包能力：140万	1	5,865.00
公网带宽	中型、NAT网关支持最大连接数50,000	1	81,592.80
公网带宽	1. 流量包120TB 2. 可用性：不低于99.95%可用性保障 3. 安全：免费开启DDoS基础防护 4. 监控：免费提供分钟级粒度的流量监控，监控带宽流量波动、出入网带宽速率等指标详情	8	2,910
弹性公网EIP	1. 按带宽5M 2. 可用性：不低于99.95%可用性保障 3. 安全：免费开启DDoS基础防护 4. 监控：免费提供分钟级粒度的流量监控，监控带宽流量波动、出入网带宽速率等指标详情	9	1,401.60
云硬盘I	1. 全动态BGP 2. 可用性：不低于99.95%可用性保障 3. 安全：免费开启DDoS基础防护 4. 监控：免费提供分钟级粒度的流量监控，监控带宽流量波动、出入网带宽速率等指标详情	7	840.00
云硬盘II	1. 通用型SSD 100G 2. IOPS上限1,100 3. IOPS突发上限5,000 4. 支持磁盘共享 5. 支持磁盘加密 6. 支持SCSI指令透传 7. 支持自动备份	1	12,000.00

云资源	规格	数量	每年费用（元）
OBS存储	1. 超高IO 1000G 2. IOPS上限33000 3. 支持磁盘共享 4. 支持磁盘加密 5. 支持SCSI指令透传 6. 支持自动备份	1	41,058.00
域名，虚拟私有云VPC	1. 50TB 2. 免费赠送读请求1.2亿次/月 3. 免费赠送写请求3000万次/月	1	100.00
安全证书	1. 顶级域名 2. 可以创建多个子网 3. 子网间支持建立对等链接	1	5,6760
Web应用防火墙 WAF	1. GeoTrust 通配符域名 2. OVSSL 专业版	1	54,800.00

3 实施步骤

3.1 准备工作

软件版本

1. CentOS Linux release 7.6 (Core)
2. jdk-8u291-linux-x64.tar.gz
3. MySQL 5.7 2核4G
4. Redis 2核4G
5. Nacos14.2
6. Seata14.2 4核8G
7. Elk7.6（日志收集处理） (视频) 2核8G
8. Rabbitmq3.8 2核4G
9. Openoffice（转码服务）（后台）
10. Minio（分布式文件存储） 2核4G
11. Gitlab(14+以上)（实训使用） 4核8G以上
12. jenkins（实训使用）
13. sonar（实训使用） 4核8G 4台 +k8s
14. K8s（实训,实验使用）

操作系统设置

Centos7 修改主机名（永久）

```
# hostnamectl --static set-hostname esxi-centos7-college
```

3.2 关闭 SELINUX

```
vi  /etc/sysconfig/selinux
```

调整

```
SELINUX=disabled（修改完需重启服务器）
getenforce （查看selinux状态）
如果内网部署可直接关闭防火墙，无需添加防火墙例外
systemctl stop firewalld.service #停止firewall
systemctl disable firewalld.service #禁止firewall开机启动
```

确定时区

date

如果时间和当前时间不对执行

```
timedatectl set-timezone Asia/Shanghai
```

3.3 JDK1.8 部署

安装

1. 复制文件 jdk-8u291-linux-x64.tar.gz 到 /root。
2. 解压
```
# tar -zxvf jdk-8u291-linux-x64.tar.gz -C /usr/local/
```
3. 改变文件拥有者
```
# chown root:root -R /usr/local/jdk1.8.0_291/
```

配置

1. 设置环境变量 编辑/etc/profile在文档最后追加以下内容：
```
# vi /etc/profile
## set java environment
JAVA_HOME=/usr/local/jdk1.8.0_291
JRE_HOME=/usr/local/jdk1.8.0_291/jre
CLASS_PATH=.:$JAVA_HOME/lib/dt.jar:$JAVA_HOME/lib/tools.jar:$JRE_HOME/lib
PATH=$PATH:$JAVA_HOME/bin:$JRE_HOME/bin
export JAVA_HOME JRE_HOME CLASS_PATH PATH
```
2. 使配置文件生效：
```
# source /etc/profile
```

验证

```
# java -version
```

控制输出如下内容说明jdk安装并配置成功！

```
java version "1.8.0_191"
Java(TM) SE Runtime Environment (build 1.8.0_191-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)
```

3.4 Nginx 部署

添加源

默认情况Centos7中无Nginx的源，最近发现Nginx官网提供了Centos的源地址。因此可以如下执行命令添加源：

```
# sudo rpm -Uvh http://nginx.org/packages/centos/7/noarch/RPMS/nginx-release-
centos-7-0.el7.ngx.noarch.rpm
```

安装 Nginx

通过yum search nginx看看是否已经添加源成功。如果成功则执行下列命令安装Nginx。

```
# yum install -y nginx
```

1. 启动Nginx
```
# systemctl start nginx.service
```

2. 设置开机自动运行
```
# systemctl enable nginx.service
```

添加防火墙例外

1. 使用root用户执行命令 开启80 端口
```
firewall-cmd --zone=public --add-port=80/tcp --permanent
```

2. 重新加载
```
firewall-cmd --reload
```

文件上传大小配置

```
vi /etc/nginx/nginx.conf
```

添加

```
client_max_body_size 4086M;
```

使最新的防火墙设置规则生效。

3.5 安装 MQ

1. 安装依赖项

   Mq要求版本3.7以上，建议版本rabbitmq-server3.8安装基本依赖

```
yum install epel-release
yum install unixODBC unixODBC-devel wxBase wxGTK SDL wxGTK-gl
```

2. 安装erlang(通过安装包安装)
```
rpm -ivh esl-erlang_24.1-1_centos_7_amd64.rpm
```

3. 添加 mq 到你的系统 repository 列表中，执行
```
curl -s https://packagecloud.io/install/repositories/rabbitmq/rabbitmq-server/script.rpm.sh | sudo bash
```

4. yum安装mq
```
yum install rabbitmq-server-3.8.26-1.el7.noarch
```

   开启web控制台

```
rabbitmq-plugins enable rabbitmq_management
```

   安装所需插件

```
rabbitmq-plugins enable rabbitmq_web_stomp
rabbitmq-plugins enable rabbitmq_web_stomp_examples
rabbitmq-plugins enable rabbitmq_delayed_message_exchange
(需要将rabbitmq_delayed_message_exchange放到/usr/lib/rabbitmq/lib/rabbitmq_server-3.8.26/plugins下)
```

   重启服务

```
systemctl restart rabbitmq-server.service
查看当前用户命令：rabbitmqctl list_users
创建用户：rabbitmqctl add_user admin 123456
将用户给于管理员权限：rabbitmqctl set_user_tags admin administrator
```

   进入web页面设置

图 3-1 设置 1

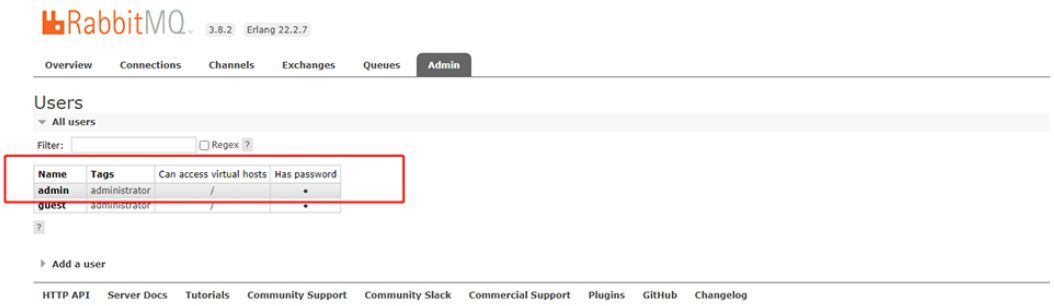
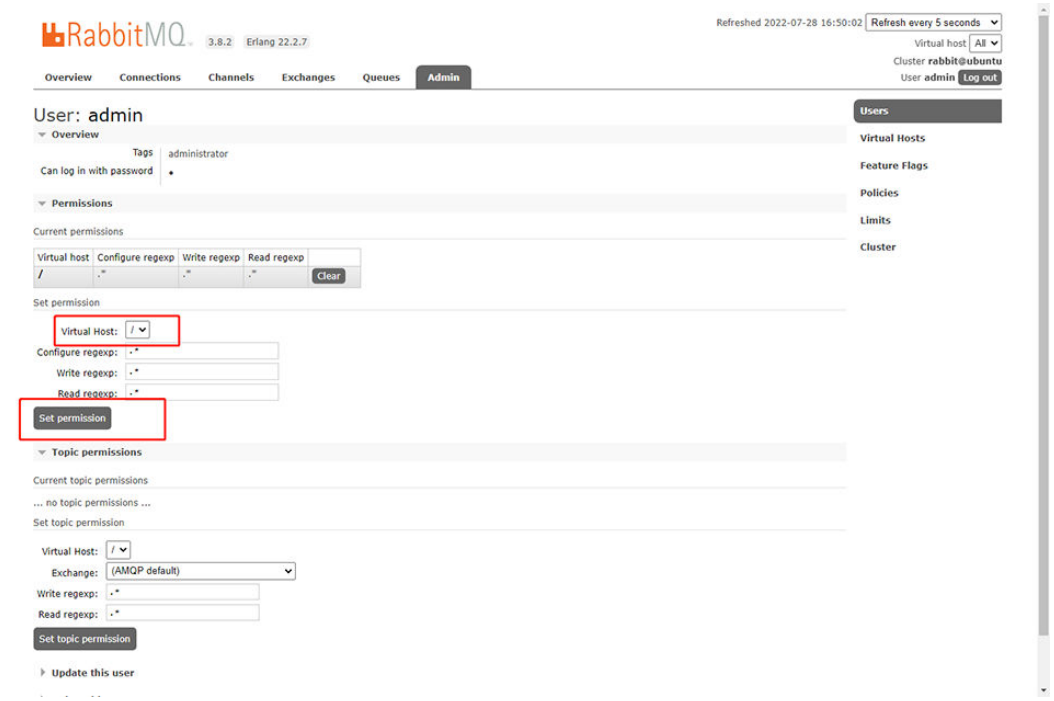


图 3-2 设置 2



3.6 安装 mysql5.7

1. 下载 MySQL Yum Repository

   执行

   ```
   wget -i -c http://dev.mysql.com/get/mysql57-community-release-el7-10.noarch.rpm
   ```

2. 添加 MySQL Yum Repository 到你的系统 repository 列表中

   ```
   yum localinstall mysql57-community-release-el7-10.noarch.rpm
   ```

   验证下是否添加成功

   ```
   yum repolist enabled | grep "mysql.*-community.*"
   ```

   选择要启用 MySQL 版本

   查看 MySQL 版本，执行

   ```
   yum repolist all | grep mysql
   ```

3. 通过 Yum 来安装 MySQL

```
yum -y install mysql-community-server
```

linux安装MySQL时报错：

图 3-3 安装报错

```
Is this ok [y/N]: y


Public key for mysql-community-client-5.7.38-1.el7.x86_64.rpm is not installed


 Failing package is: mysql-community-client-5.7.38-1.el7.x86_64
 GPG Keys are configured as: file:///etc/pki/rpm-gpg/RPM-GPG-KEY-mysql

[root@localhost opt]#
```

原因：MySQL GPG 密钥已过期导致

解决办法：执行以下命令，解决

```
rpm --import https://repo.mysql.com/RPM-GPG-KEY-mysql-2022
```

启动 MySQL Server

```
systemctl start mysqld.service
```

查看 MySQL Server 状态

```
systemctl status mysqld.service
```

关闭 MySQL Server

```
systemctl stop mysqld.service
```

关闭 MySQL Server

```
systemctl enable mysqld.service
```

经过 grep "password" /var/log/mysqld.log 命令，返回结果最后冒号后面的字符串就是root的默认密码。

图 3-4 默认密码

```
[root@VM_0_15_centos ~]# grep "temporary password" /var/log/mysqld.log
2019-06-28T05:10:08.531788Z 1 [Note] A temporary password is generated for root@localhost: hi%Dgp9ye(_s
```

使用此密码登录后，Mysql 会要求第一件作的事就是改root密码，并且是要求强密码。能够经过set password=password('密码')来更改。数据库

测试是否安装成功

```
Mysql  -u  root  -p
```

修改当前登录用户密码：

```
mysql>SET PASSWORD = PASSWORD('Zretc137!');
```

允许mysql远程访问

mysql默认是不允许远程访问的.

```
mysql -u root -p xxxx
mysql>use mysql
mysql>grant all privileges on *.* to 'root'@'%' identified by 'Zretc137!';
mysql>flush privileges;
```

mysql增加配置

```
vi /etc/my.cnf
[mysqld]
event_scheduler = on
```

```
wait_timeout=2880000
interactive_timeout =2880000
max_allowed_packet=10M
ft_min_word_len=1
lower_case_table_names=1
sql_mode =STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
max_connections=500
```

3.7 安装 minio

1. 根据提供的minio包，直接运行即可

```
chmod +x minio
MINIO_ACCESS_KEY=myminioadmin MINIO_SECRET_KEY=myminioadmin nohup ./minio server --address '0.0.0.0:9000' --console-address '0.0.0.0:9001' /opt/data > /opt/minio.log 2>&1 &
```

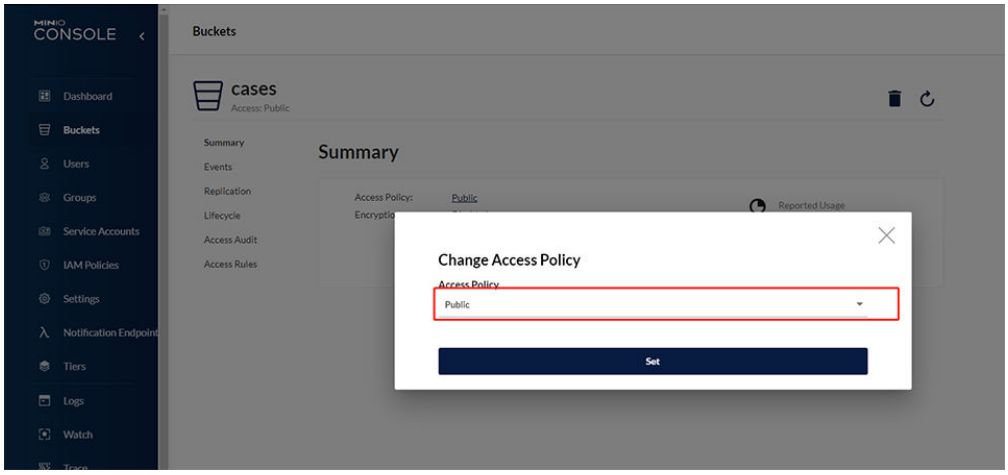
说明

需要绑定所提供的数据文件夹

2. 测试minio是否正确安装

访问ip http://127.0.0.1:9000,账户密码为：myminioadmin myminioadmin，能正常访问及安装完成

图 3-5 测试 minio



Minio所有桶权限都修改为public

3.8 安装 redis

1. yum安装redis

```
yum install redis
```

2. 修改redis密码

```
vi  /etc/redis.conf
requirepass root123
bind 0.0.0.0
```

安装完成检查：

```
netstat -anp |grep 6379
```

出现对接信息及安装成功

3.9 安装 job

1. 用压缩工具打开jar包修改jar包配置文件

   修改数据库地址

   ```
   xxl-job-admin-2.3.0\BOOT-INF\application.properties
   ```

2. 运行job jar包

   ```
   nohup java -jar  xxl-job-admin-2.3.0.jar &
   ```

   成功验证：

   访问浏览器：127.0.0.1:8912/xxl-job-admin/

   密码：admin 123456

3.10 安装 Seata1.4.2+Nacos1.4.2

创建数据库nacos导入数据库文件nacos-mysql.sql

创建数据库seata_db 导入数据库文件db_store.sql

1. 通过安装包安装nacos和seata

   ```
   tar -zxvf nacos-server-1.4.2.tar.gz
   unzip seata.zip
   ```

2. 修改nacos配置文件

   修改配置文件

   ```
   cd /opt/nacos/conf
   vim application.properties
   spring.datasource.platform=mysql
   db.num=1
   db.url.0=jdbc:mysql://127.0.0.1:3306/nacos?
   characterEncoding=utf8&connectTimeout=1000&socketTimeout=3000&autoReconnect=true&useUnico
   de=true&useSSL=false&serverTimezone=UTC
   db.user.0=root
   db.password.0=Zretc137!
   server.tomcat.accesslog.enabled=false(减少日志文件)(103行)
   ```

3. 加大nacos占用内存

   ```
   vi startup.sh
   JAVA_OPT="${JAVA_OPT} -Xms2048m -Xmx2048m -Xmn1024m"
   ```

4. 启动nacos

   ```
   sh startup.sh -m standalone
   ```

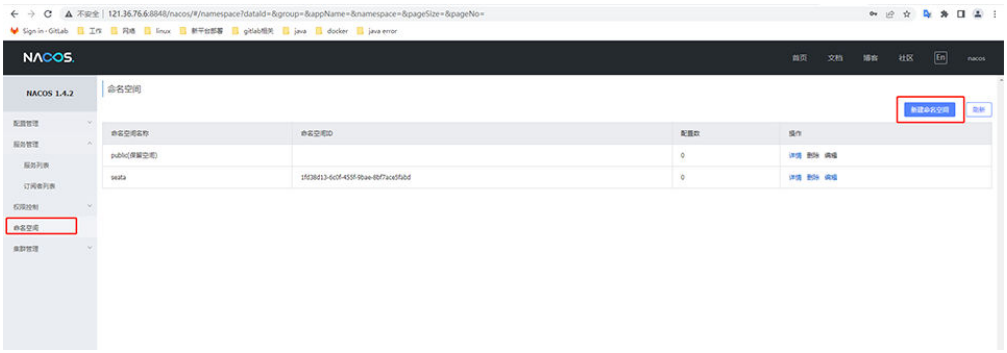
5. 登录nacos

   默认账户密码：nacos/nacos

   地址：http://host:8848/nacos

   增加seata命名空间

图 3-6 新建命名空间



6. 修改seata配置文件

```
Vi /opt/seata-server-1.4.2/conf
```

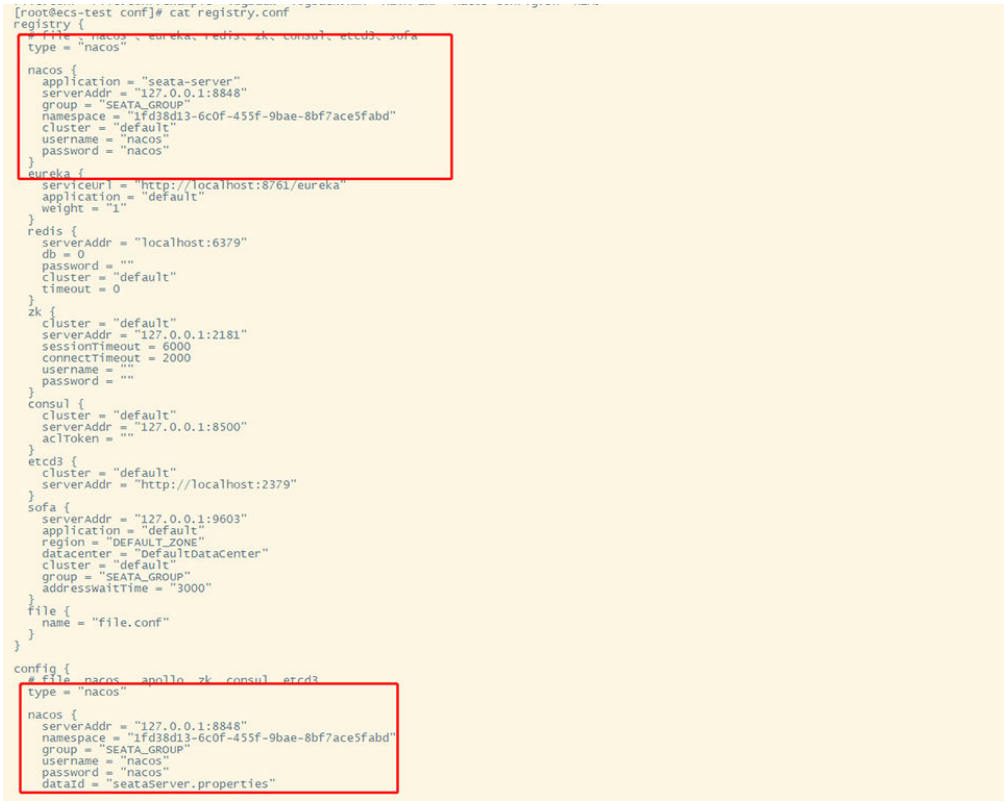
图 3-7 修改配置文件

```
## transaction log store, only used in seata-server
store {
  ## store mode: file、db、redis
  mode = "db"
  ## rsa decryption public key
  publicKey = ""
  ## file store property
  file {
    ## store location dir
    dir = "sessionStore"
    # branch session size , if exceeded first try compress lockkey, still exceeded throws exceptions
    maxBranchSessionSize = 16384
    # globe session size , if exceeded throws exceptions
    maxGlobalSessionSize = 512
    # file buffer size , if exceeded allocate new buffer
    fileWriteBufferCacheSize = 16384
    # when recover batch read size
    sessionReloadReadSize = 100
    # async, sync
    flushDiskMode = async
  }

  ## database store property
  db {
    ## the implement of javax.sql.DataSource, such as DruidDataSource(druid)/BasicDataSource(dbcp)/HikariDataSource(hikari) etc.
    datasource = "druid"
    ## mysql/oracle/postgresql/h2/oceanbase etc.
    dbType = "mysql"
    driverClassName = "com.mysql.jdbc.Driver"
    ## if using mysql to store the data, recommend add rewriteBatchedStatements=true in jdbc connection param
    url = "jdbc:mysql://127.0.0.1:3306/seata_db?rewriteBatchedStatements=true"
    user = "root"
    password = "Zretc137!"
    minConn = 5
    maxConn = 100
    globalTable = "global_table"
    branchTable = "branch_table"
    lockTable = "lock_table"
    queryLimit = 100
    maxWait = 5000
  }

  ## redis store property
  redis {
    ## redis mode: single、sentinel
    mode = "single"
    ## single mode property
    single {
      host = "192.168.101.210"
```

创建数据库，导入sql db_store.sql

```
Vi registry.conf
```

图 3-8 创建数据库



7. 将seata 配置文件导入nacos
   修改config.txt文件

图 3-9 修改 config.txt 文件 1

```
transport.type=TCP
transport.server=NIO
transport.heartbeat=true
transport.enableClientBatchSendRequest=false
transport.threadFactory.bossThreadPrefix=NettyBoss
transport.threadFactory.workerThreadPrefix=NettyServerNIOWorker
transport.threadFactory.serverExecutorThreadPrefix=NettyServerBizHandler
transport.threadFactory.shareBossWorker=false
transport.threadFactory.clientSelectorThreadPrefix=NettyClientSelector
transport.threadFactory.clientSelectorThreadSize=1
transport.threadFactory.clientWorkerThreadPrefix=NettyClientWorkerThread
transport.threadFactory.bossThreadSize=1
transport.threadFactory.workerThreadSize=default
transport.shutdown.wait=3
service.vgroupMapping.my_test_tx_group=default
service.default.grouplist=192.168.101.211:8091
service.enableDegrade=false
service.disableGlobalTransaction=false
client.rm.asyncCommitBufferLimit=10000
client.rm.lock.retryInterval=10
client.rm.lock.retryTimes=30
client.rm.lock.retryPolicyBranchRollbackOnConflict=true
client.rm.reportRetryCount=5
client.rm.tableMetaCheckEnable=false
client.rm.tableMetaCheckerInterval=60000
client.rm.sqlParserType=druid
client.rm.reportSuccessEnable=false
client.rm.sagaBranchRegisterEnable=false
client.tm.commitRetryCount=5
client.tm.rollbackRetryCount=5
client.tm.defaultGlobalTransactionTimeout=60000
client.tm.degradeCheck=false
client.tm.degradeCheckAllowTimes=10
client.tm.degradeCheckPeriod=2000
store.mode=db
store.publicKey=
store.file.dir=file_store/data
store.file.maxBranchSessionSize=16384
store.file.maxGlobalSessionSize=512
store.file.fileWriteBufferCacheSize=16384
store.file.flushDiskMode=async
store.file.sessionReloadReadSize=100
store.db.datasource=druid
store.db.dbType=mysql
store.db.driverClassName=com.mysql.jdbc.Driver
store.db.url=jdbc:mysql://127.0.0.1:3306/seata_db?useUnicode=true&rewriteBatchedStatements=true
store.db.user=root
store.db.password=Zretc137!
store.db.minConn=5
store.db.maxConn=30
store.db.globalTable=global_table
store.db.branchTable=branch_table
store.db.queryLimit=100
store.db.lockTable=lock_table
store.db.maxWait=5000
store.redis.mode=single
store.redis.single.host=127.0.0.1
store.redis.single.port=6379
store.redis.sentinel.masterName=
store.redis.sentinel.sentinelHosts=
store.redis.maxConn=10
store.redis.minConn=1
store.redis.maxTotal=100
store.redis.database=0
store.redis.password=
store.redis.queryLimit=100
server.recovery.committingRetryPeriod=1000
server.recovery.asynCommittingRetryPeriod=1000
"config.txt" 89L, 3140C
```

```
sh nacos-config.sh -h 127.0.0.1 -p 8848 -g SEATA_GROUP -t b952ebce-5404-428f-8b07-6243bffdd5a9
-u nacos -w nacos
```

图 3-10 修改 config.txt 文件 2

```
Set store.db.globalTable=global_table successfully
Set store.db.branchTable=branch_table successfully
Set store.db.queryLimit=100 successfully
Set store.db.lockTable=lock_table successfully
Set store.db.maxWait=5000 successfully
Set store.redis.mode=single successfully
Set store.redis.single.host=127.0.0.1 successfully
Set store.redis.single.port=6379 successfully
nacos-config.sh: line 88: [: too many arguments
Set store.redis.sentinel.masterName= failure
nacos-config.sh: line 88: [: too many arguments
Set store.redis.sentinel.sentinelHosts= failure
Set store.redis.maxConn=10 successfully
Set store.redis.minConn=1 successfully
Set store.redis.maxTotal=100 successfully
Set store.redis.database=0 successfully
nacos-config.sh: line 88: [: too many arguments
Set store.redis.password= failure
Set store.redis.queryLimit=100 successfully
Set server.recovery.committingRetryPeriod=1000 successfully
Set server.recovery.asynCommittingRetryPeriod=1000 successfully
Set server.recovery.rollbackingRetryPeriod=1000 successfully
Set server.recovery.timeoutRetryPeriod=1000 successfully
Set server.maxCommitRetryTimeout=-1 successfully
Set server.maxRollbackRetryTimeout=-1 successfully
Set server.rollbackRetryTimeoutUnlockEnable=false successfully
Set client.undo.dataValidation=true successfully
Set client.undo.logSerialization=jackson successfully
Set client.undo.onlyCareUpdateColumns=true successfully
Set server.undo.logSaveDays=7 successfully
Set server.undo.logDeletePeriod=86400000 successfully
Set client.undo.logTable=undo_log successfully
Set client.undo.compress.enable=true successfully
Set client.undo.compress.type=zip successfully
Set client.undo.compress.threshold=64k successfully
Set log.exceptionRate=100 successfully
Set transport.serialization=seata successfully
Set transport.compressor=none successfully
Set metrics.enabled=false successfully
Set metrics.registryType=compact successfully
Set metrics.exporterList=prometheus successfully
Set metrics.exporterPrometheusPort=9898 successfully
=========================================================================
 Complete initialization parameters,  total-count:89 ,  failure-count:4
=========================================================================
 init nacos config fail.
```

启动seata

```
nohup ./seata-server.sh >log.out 2>1 &
```

注意事项：nacos额外增加配置base-config-test.yaml(导入后，需要修改配置文件与自己本地相同)

成功验证：

访问浏览器：http://127.0.0.1:8848/nacos/

3.11 安装 OpenOffice

1. 根据OpenOffice包安装

   解压，得到zh-CN目录。

   进入zh-CN文件夹下的RPMS目录下，执行yum localinstall *.rpm安装必要的包

   进入RPMS目下的desktop-integration，执行命令：

   ```
   rpm -ivh openoffice4.1.4-redhat-menus-4.1.4-9788.noarch.rpm
   ```

2. 安装openoffice依赖启动项

   ```
   yum install libXext.x86_64
   yum groupinstall "X Window System"
   ```

   说明

   - openoffice需要上传语言包(chinese.tar.gz为字体包)
   - 能正常启动后需要kill掉，需要通过运行的jar去自启动。
   - Openoffice需要和resource-convert-3.5.0-SNAPSHOT.jar在同一台服务器。

   成功验证：

   netstat -anp |grep 8100

   有正常端口监测代表正常安装

3.12 yum 安装 npm 和 nodejs

下载

首先在官网查看当前最新的版本 **https://nodejs.org/dist/**

```
# cd /opt
wget https://nodejs.org/dist/v15.0.0/node-v12.18.1-linux-x64.tar.gz
```

安装

1. 下载完成后解压

   ```
   # tar -zxvf node-v15.0.0-linux-x64.tar.gz
   ```

2. 重命名

   ```
   # mv node-v15.0.0-linux-x64/ node-v15.0.0
   ```

3. 配置环境变量

   ```
   # vi /etc/profile
   ```

   在最后边添加

```
#set for nodejs
export NODE_HOME=/opt/node-v15.0.0
export PATH=$NODE_HOME/bin:$PATH
```

4. 保存退出（:wq）执行命令是更改生效

```
# source /etc/profile
```

检查

使用命令查看版本，出现相应版本号则表示成功

```
# node -v
```

3.13 node 项目安装软件列表

安装最新本pm2

```
npm install pm2@latest -g
```

安装最新本fis3

```
npm install -g fis3
```

3.14 安装 neo4j

将安装包复制至服务器

```
tar -zxvf neo4j-community-3.5.31-unix.tar.gz
neo4j根目录/conf/neo4j.conf
"dbms.default_listen_address=0.0.0.0"取消注释
dbms.connector.bolt.listen_address=:7687取消注释
dbms.connector.http.listen_address=:7474 取消注释
```

启动服务

neo4j根目录bin/neo4j start

浏览器中打开http://<ip>:7474/browser/，就可以访问了（如果打开后整个页面都是空白的，是浏览器不兼容，换个浏览器）

初始用户名和密码都是neo4j，登录后会让改下密码

3.15 安装 kp-abstract

```
tar kp-abstract-dev.tar.gz
yum install -y centos-release-scl
yum install -y rh-python38 which
scl enable rh-python38 bash
```

查看python位置

```
which python{,2,3}
```

超链接

ll /opt/rh/rh-python38/root/usr/bin/python*

图 3-11 超链接

```
# which python{,2,3}
/opt/rh/rh-python38/root/usr/bin/python
/usr/bin/python2
/opt/rh/rh-python38/root/usr/bin/python3

# ll /opt/rh/rh-python38/root/usr/bin/python*
lrwxrwxrwx. 1 root root     9 Jun 24 10:31 /opt/rh/rh-python38/root/usr/bin/python -> ./python3
lrwxrwxrwx. 1 root root     9 Jun 24 10:31 /opt/rh/rh-python38/root/usr/bin/python3 -> python3.8
-rwxr-xr-x. 1 root root 15280 May 28 09:39 /opt/rh/rh-python38/root/usr/bin/python3.8

# python -V
Python 3.8.0

# python3 -V
Python 3.8.0
```

查看python版本

```
python -V
Python 3.8.0
```

进入项目文件夹

```
cd  kp-abstract
pip  install poetry
poetry install
poetry shell
nohup poetry run python -m uvicorn app.main:app --host 0.0.0.0 --port 8918 &
```

3.16 安装 elk

1. 解压node包

   ```
   tar xvJf node-v12.18.1-linux-x64.tar.xz
   mv node-v12.18.1-linux-x64 node-v12.18.1
   ```

2. 将node写入配置文件

   ```
   vi /etc/profile
   #set for nodejs
   export NODE_HOME=/opt/node-v12.18.1
   export PATH=$NODE_HOME/bin:$PATH
   source /etc/profile
   ```

3. 安装elk启动依赖

   ```
   npm install -g grunt-cli
   ```

4. 创建elk用户

   elasticsearch不能通过root用户启动，创建es用户

   ```
   adduser es
   passwd es
   ```

5. 给es用户赋予权限

   ```
   chown -R es:root /elk/elasticsearch-7.16.2/*
   chmod -R 755 /elk/elasticsearch-7.16.2/*
   ```

6. 修改对应配置文件

   修改config elasticsearch.yml

network.host = 自己

elasticsearch用户拥有的可创建文件描述的权限太低，至少需要65536,

处理办法: #切换到root用户修改

7. 修改配置文件

   vim /etc/security/limits.conf # 在最后面追加下面内容

   ```
   es hard nofile 65536
   es soft nofile 65536
   #***  是启动elk的用户
   ```

8. 启动提示vma过少

   max_map_count文件包含限制一个进程可以拥有的VMA(虚拟内存区域)的数量

   处理办法: #切换到root用户修改

   vim /etc/sysctl.conf # 在最后面追加下面内容

   ```
   vm.max_map_count=655360
   ```

   执行

   ```
   sysctl -p
   ```

9. 启动Elasticsearch

   ```
   bin/elasticsearch -d
   ```

10. 启动logstash

    修改配置文件logstash.conf

    ```
    nohup ./logstash -f ../config/logstash.conf &
    ```

11. 启动Elasticsearch-head-master

    ```
    nohup grunt server &
    ```

说明

需要将sys后台服务日志挂载至本服务器

3.17 Gitlab 14.2.1 部署

下载

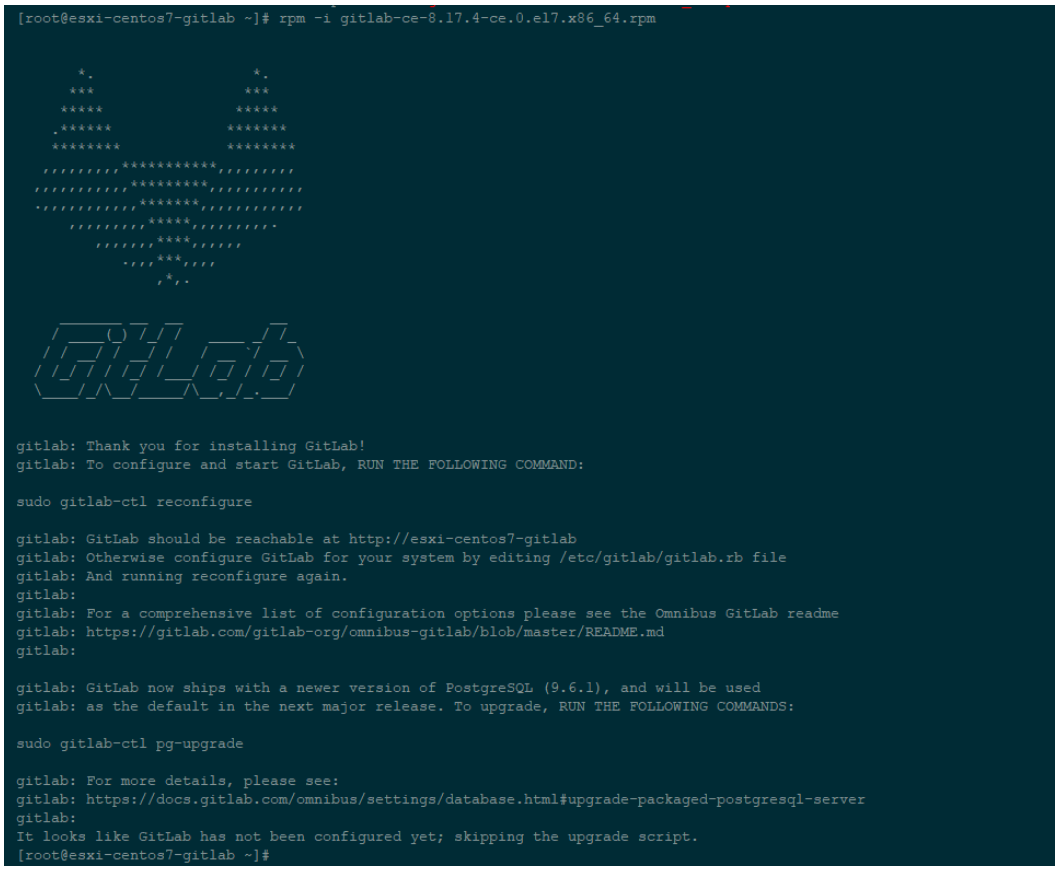
安装并配置必要的依赖项

```
sudo yum install -y curl policycoreutils-python openssh-server
sudo systemctl enable sshd
sudo systemctl start sshd
sudo firewall-cmd --permanent --add-service=http
sudo systemctl reload firewalld
curl -s https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash
yum makecache
sudo yum install gitlab-ce-14.10.4-ce.0.el7.x86_64
```

安装

安装过程需要些时间，如果出现下图，则说明安装成功。

图 3-12 安装 1



修改gitlab配置文件指定服务器ip和自定义端口：

```
# vi /etc/gitlab/gitlab.rb
```

图 3-13 安装 2

重置并启动GitLab

执行：

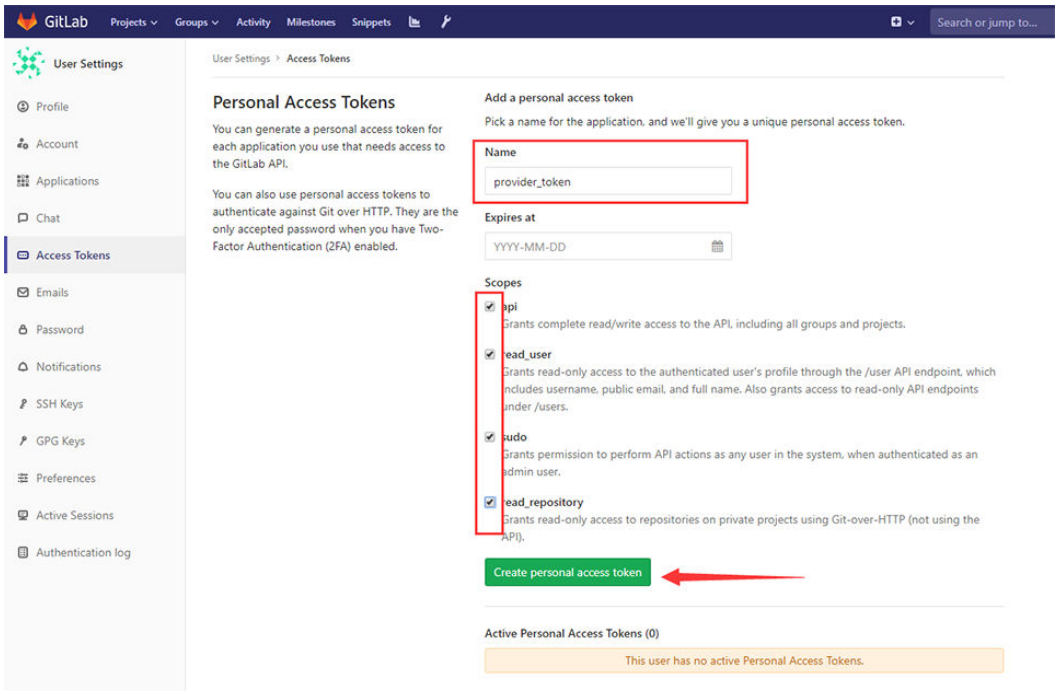
```
# gitlab-ctl reconfigure
# gitlab-ctl restart
```

提示 "ok: run:"表示启动成功

创建 Access Token

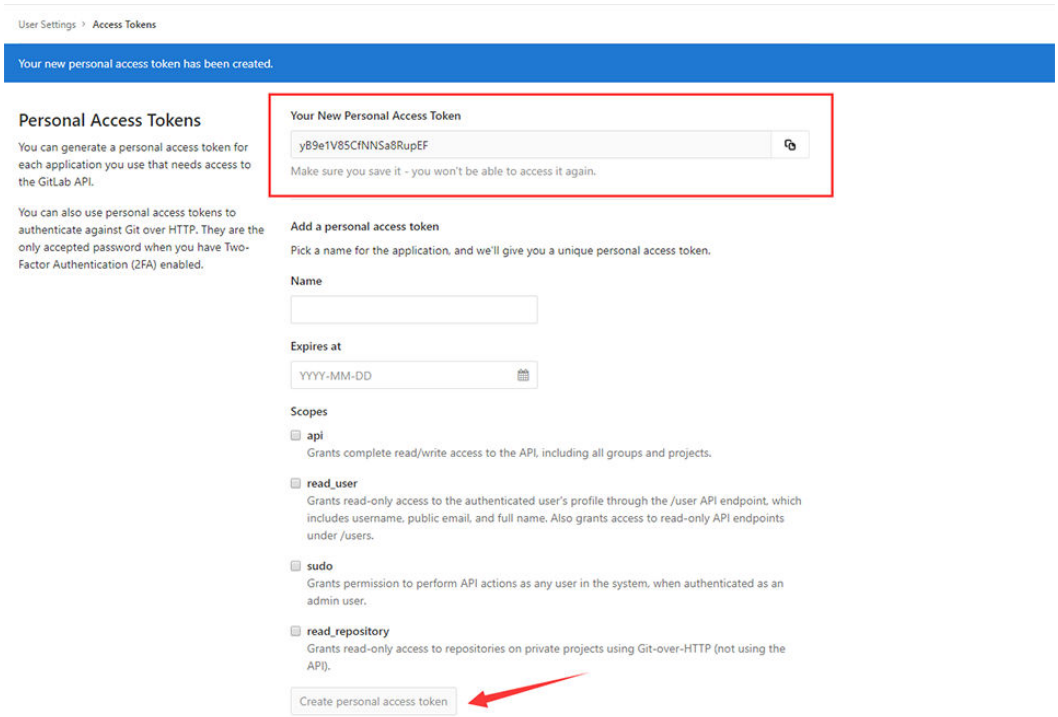
浏览器打开页面：http://主机ip/profile/personal_access_tokens

图 3-14 创建 1



Name随便填写。过期时间不填默认永久有效，勾选所有权限，单击Create personal access token按钮生成token。

图 3-15 创建 2



单击Create personal access token按钮完成按钮置灰后，页面上方位置显示了新创建了access token。请复制保存，部署后端项目portal-web配置gitlab相关配置时使用。需要注意的是该页面只显示一次，如果没有及时保存可重新创建即可。

完全卸载删除 gitlab

1. 停止gitlab
   ```
   gitlab-ctl stop
   ```
2. 卸载gitlab（注意这里写的是gitlab-ce）
   ```
   rpm -e gitlab-ce
   ```
3. 查看gitlab进程
   ```
   ps aux | grep gitlab
   ```
4. 终止第一个进程（就是带有好多.....的进程）
   ```
   kill -9 18777
   ```
   终止后，在ps aux | grep gitlab确认一遍，还有没有gitlab的进程
5. 删除所有包含gitlab文件
   ```
   find / -name gitlab | xargs rm -rf
   ```

3.18 SonarQube 部署

查看 MySQL 引擎

结合 SonarQube，MySQL 数据库最好使用 InnoDB 引擎，可提高性能。 看你的 mysql 现在已提供什么存储引擎:

```
mysql> show engines;
```

图 3-16 查看引擎 1

```
Last login: Fri Feb  2 02:58:27 2018 from 192.168.1.101
[heyuqiang@es-db-01 ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.6.38 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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 statement.

mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.00 sec)

mysql>
```

看你的 mysql 当前默认的存储引擎:

```
mysql> show variables like '%storage_engine%';
```

图 3-17 查看引擎 2

```
Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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 statement.

mysql> show engines;
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                        | Transactions | XA   | Savepoints |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
| FEDERATED          | NO      | Federated MySQL storage engine                                 | NULL         | NULL | NULL       |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                          | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                          | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                             | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables      | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                         | NO           | NO   | NO         |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys     | YES          | YES  | YES        |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                             | NO           | NO   | NO         |
+--------------------+---------+----------------------------------------------------------------+--------------+------+------------+
9 rows in set (0.00 sec)

mysql> show variables like '%storage_engine%';
+-------------------------------+--------+
| Variable_name                 | Value  |
+-------------------------------+--------+
| default_storage_engine        | InnoDB |
| default_tmp_storage_engine    | InnoDB |
| storage_engine                | InnoDB |
+-------------------------------+--------+
3 rows in set (0.00 sec)

mysql>
```

如果查询的结果和上图显示的效果不一致，需要做以下配置：

修改 MySQL 存储引擎为 InnoDB

[mysqld] 下面加入 default-storage-engine=INNODB

```
# vi /etc/my.cnf
[mysqld]
default-storage-engine = INNODB
```

重启 mysql 服务器

```
# systemctl restart mysql.service
```

再次登录 MySQL 查看默认引擎设置是否生效

```
mysql> show variables like '%storage_engine%';
```

图 3-18 查看引擎 3

```
Last login: Fri Feb  2 02:59:09 2018 from 192.168.1.101
[heyuqiang@es-db-01 ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.6.38 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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 statement.

mysql> show variables like '%storage_engine%';
+-------------------------------+--------+
| Variable_name                 | Value  |
+-------------------------------+--------+
| default_storage_engine        | InnoDB |
| default_tmp_storage_engine    | InnoDB |
| storage_engine                | InnoDB |
+-------------------------------+--------+
3 rows in set (0.00 sec)

mysql>
```

配置 innodb_buffer_pool_size

innodb_buffer_pool_size 参数值设置得尽可能大一点 这个参数主要作用是缓存 innodb 表的索引，数据，插入数据时的缓冲 默认值:128M，专用 mysql 服务器设置的大小:操作系统内存的 70%-80%最佳。 设置方法:my.cnf 文件[mysqld] 下面加入 innodb_buffer_pool_size 参数

```
# vi /etc/my.cnf
[mysqld]
innodb_buffer_pool_size = 256M
```

说明

这里设置为 256M，因为不是专用的 MySQL 数据库服务器，还有很多其他的服务需要占用系统内存

配置 query_cache_size

设置 MySQL 的查询缓存 query_cache_size, 最少设置 15M

```
# vi /etc/my.cnf
[mysqld]
query_cache_type=1
query_cache_size=32M
```

重启 mysql 服务器

```
# systemctl restart mysql.service
```

验证缓存设置是否生效:

```
mysql> show variables like '%query_cache%';
```

图 3-19 验证缓存

```
Last login: Fri Feb  2 03:28:15 2018 from 192.168.1.101
[heyuqiang@es-db-01 ~]$ mysql -u root -p
Enter password:
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password: YES)
[heyuqiang@es-db-01 ~]$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 9
Server version: 5.6.38 MySQL Community Server (GPL)

Copyright (c) 2000, 2017, Oracle and/or its affiliates. All rights reserved.

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 statement.

mysql> show variables like '%query_cache%';
+------------------------------+----------+
| Variable_name                | Value    |
+------------------------------+----------+
| have_query_cache             | YES      |
| query_cache_limit            | 1048576  |
| query_cache_min_res_unit     | 4096     |
| query_cache_size             | 33554432 |
| query_cache_type             | ON       |
| query_cache_wlock_invalidate | OFF      |
+------------------------------+----------+
6 rows in set (0.00 sec)

mysql>
```

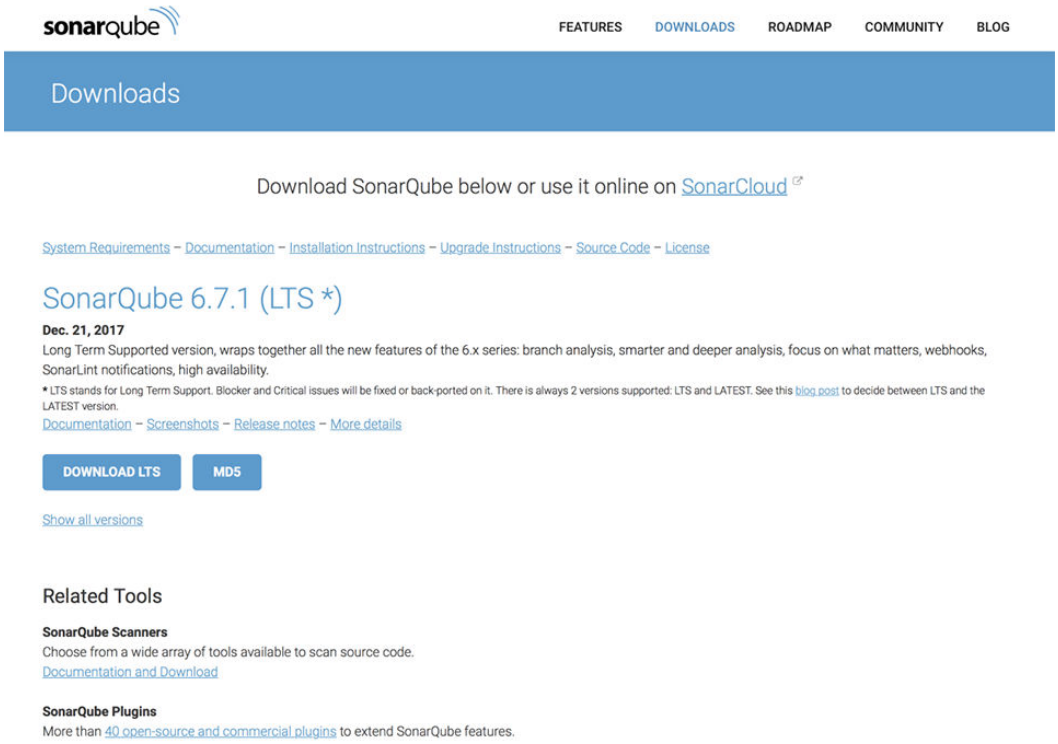
创建 sonarqube 数据库(UTF-8 编码)

```
mysql> CREATE SCHEMA `sonar` DEFAULT CHARACTER SET utf8;
```

安装 SonarQube Web Server

下载最新 LTS 版的 SonarQube 安装包(当前版本为 sonarqube-6.7.1.zip): 下载地址: **http://www.sonarqube.org/downloads/**

图 3-20 下载地址



下载 sonarqube

1. 切换到普通用户（ heyuqiang ）

```
$ cd /heyuqiang
$ wget https://binaries.sonarsource.com/Distribution/sonarqube/sonarqube-6.7.7.zip
```

2. 解压安装

```
$ unzip sonarqube-6.7.7.zip
$ mv sonarqube-6.7.7 sonarqube
```

配置 sonarqube

```
$ cd /home/heyuqiang/sonarqube/conf
$ vi sonar.properties
sonar.jdbc.username=root
sonar.jdbc.password=heyuqiang
#----- MySQL 5.6 or greater
# Only InnoDB storage engine is supported (not myISAM).
```

```
# Only the bundled driver is supported. It can not be changed.
sonar.jdbc.url=jdbc:mysql://192.168.1.181:3306/sonarqube?
useUnicode=true&characterEncoding=utf8&rewrite
BatchedStatements=true&useConfigs=maxPerformance&useSSL=false
sonar.web.host=0.0.0.0
sonar.web.context=/sonar
sonar.web.port=9000
```

保存以上配置(注意，要看看默认的 9000 端口是否已被占用)

添加防火墙例外

1. 使用root用户执行命令 开启9000 端口。
   ```
   firewall-cmd --zone=public --add-port=9000/tcp --permanent
   ```

2. 重新加载
   ```
   firewall-cmd --reload
   ```

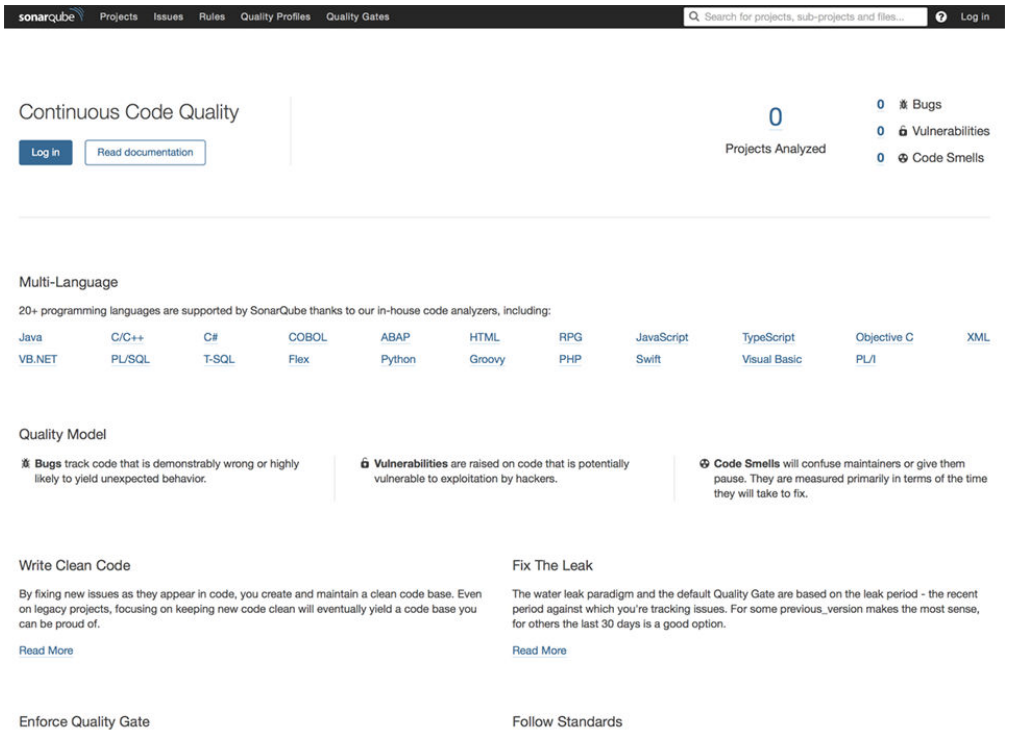
启动和测试

1. 切换到普通用户（ heyuqiang ）启动SonarQube Web Server
   ```
   $ /home/heyuqiang/sonarqube/bin/linux-x86-64/sonar.sh start
   ```
   初次启动会自动建表和做相应的初始化

2. 查看日志
   ```
   $ tail -f /home/heyuqiang/sonarqube/logs/sonar.log
   ```

3. 浏览器访问：http://192.168.1.180:9000/sonarqube/

图 3-21 启动服务

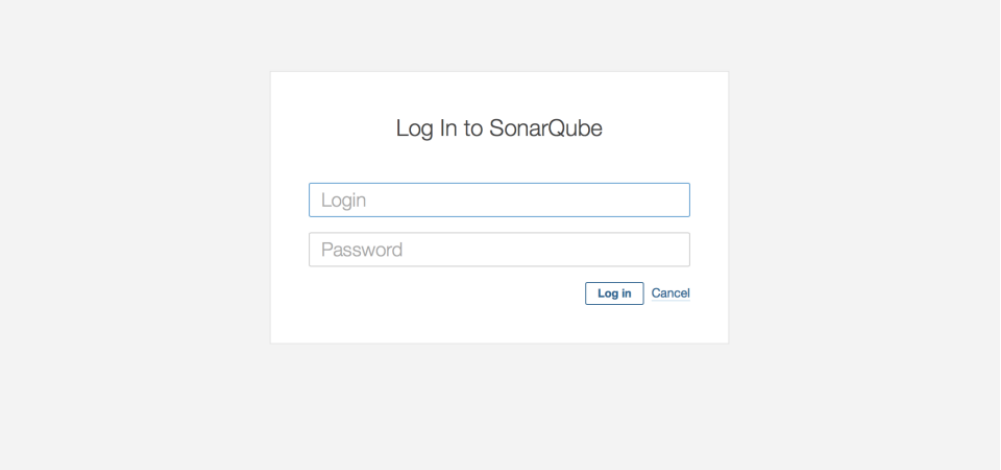


看到上图页面说明sonarqube服务启动了。

初次登录 sonarqube 设置

1. Log in
   sonarqube默认用户名：admin 密码：admin

   图 3-22 登录



2. 首次登录成功后提示生成令牌，输入用户单击Gennerater

3. >heyuqiang: 6700712a 2c8ef60a36dfc67e77a6a665491aa9b9

   图 3-23 设置 1

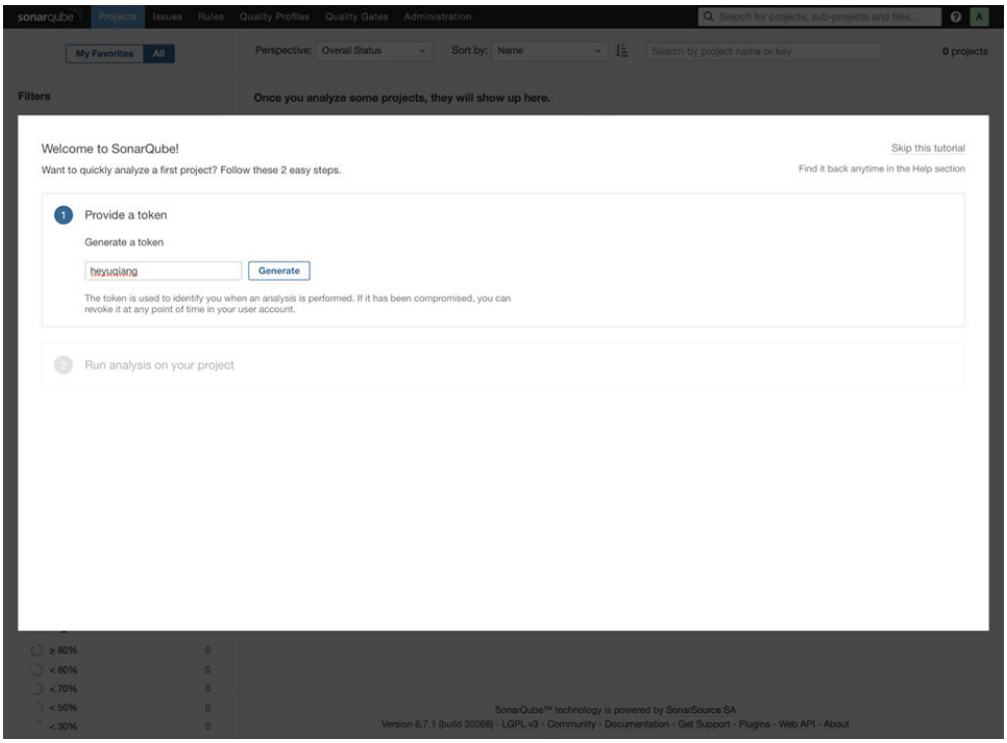
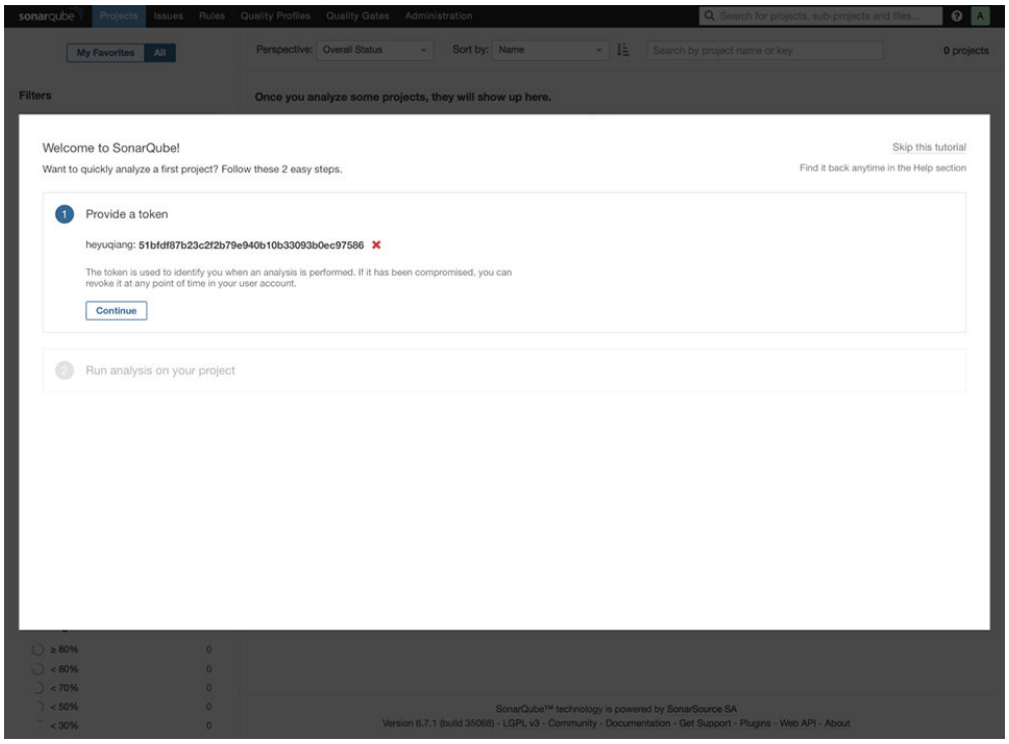
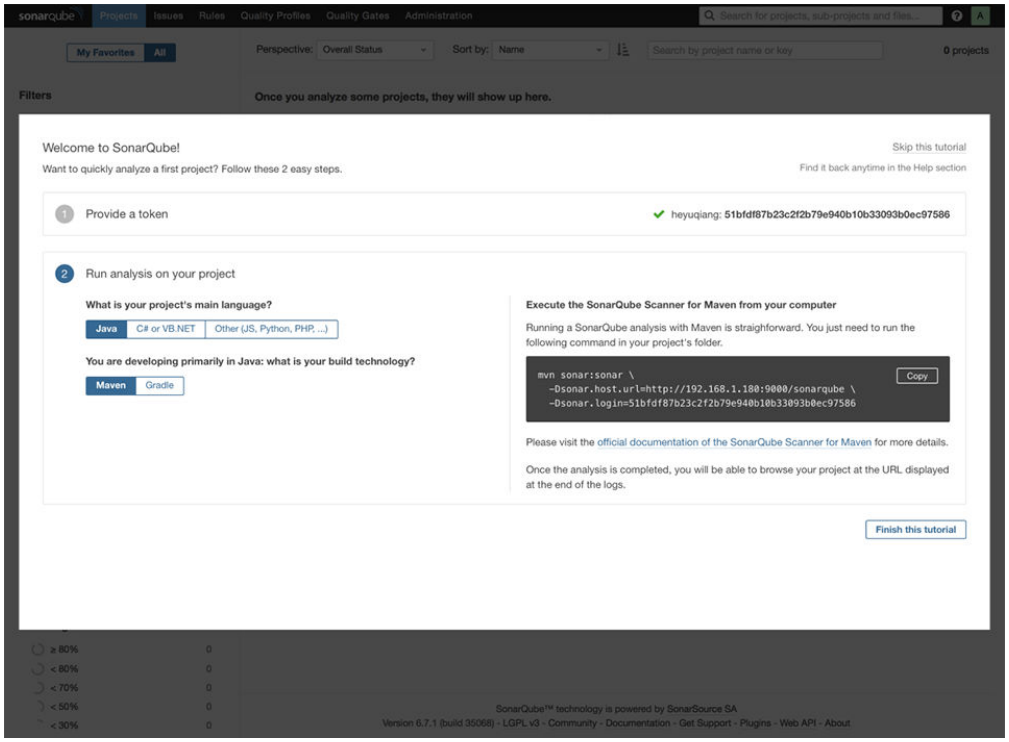


图 3-24 设置 2



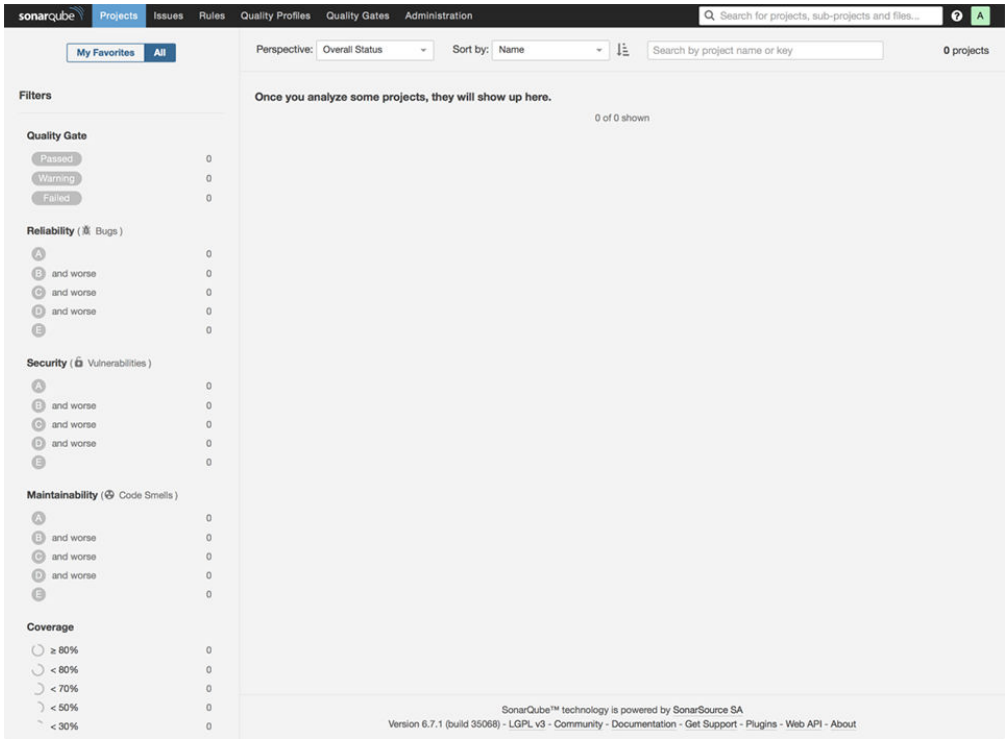
4. 选择开发语言和工具

图 3-25 开发语言与工具选择 1




```
mvn sonar:sonar \
-Dsonar.host.url=http://192.168.1.180:9000/sonarqube \
-Dsonar.login=51bfdf87b23c2f2b79e940b10b33093b0ec97586
```

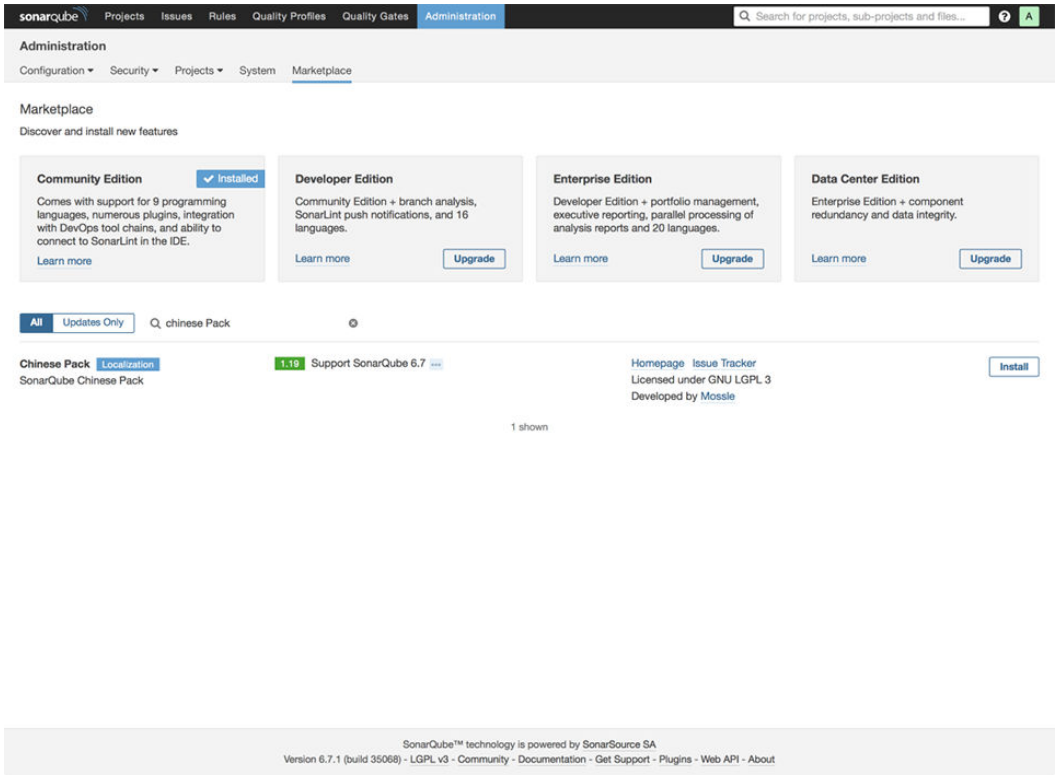
图 3-26 开发语言与工具选择 2



完成初次设置。

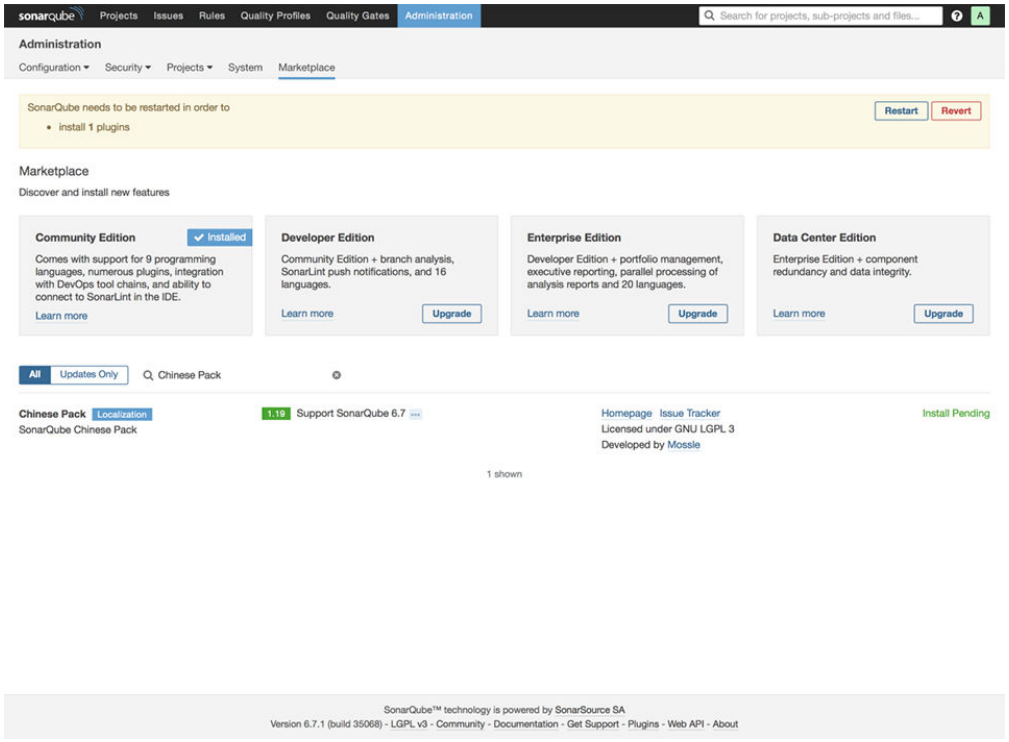
安装 chinese Pack 汉化包

图 3-27 汉化包安装 1



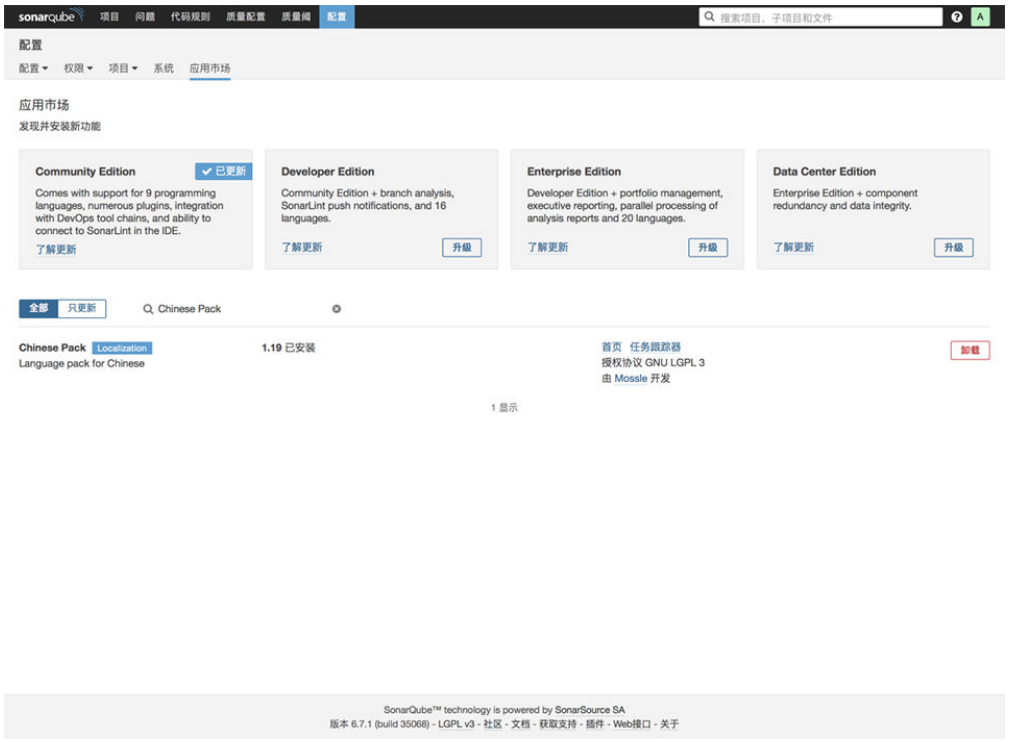
1. 单击Install 安装汉化包。

图 3-28 汉化包安装 2



2. 安装完成之后单击Restart重启sonarqube 汉化完成。

图 3-29 汉化包安装 3



设置 sonarqube 开机启动

使用root用户编辑/etc/rc.local文件，执行命令：

```
# vi /etc/rc.local
```

加入：

```
## sonarqube
su - heyuqiang -c '/home/heyuqiang/sonarqube/bin/linux-x86-64/sonar.sh start'
```

温馨提示：

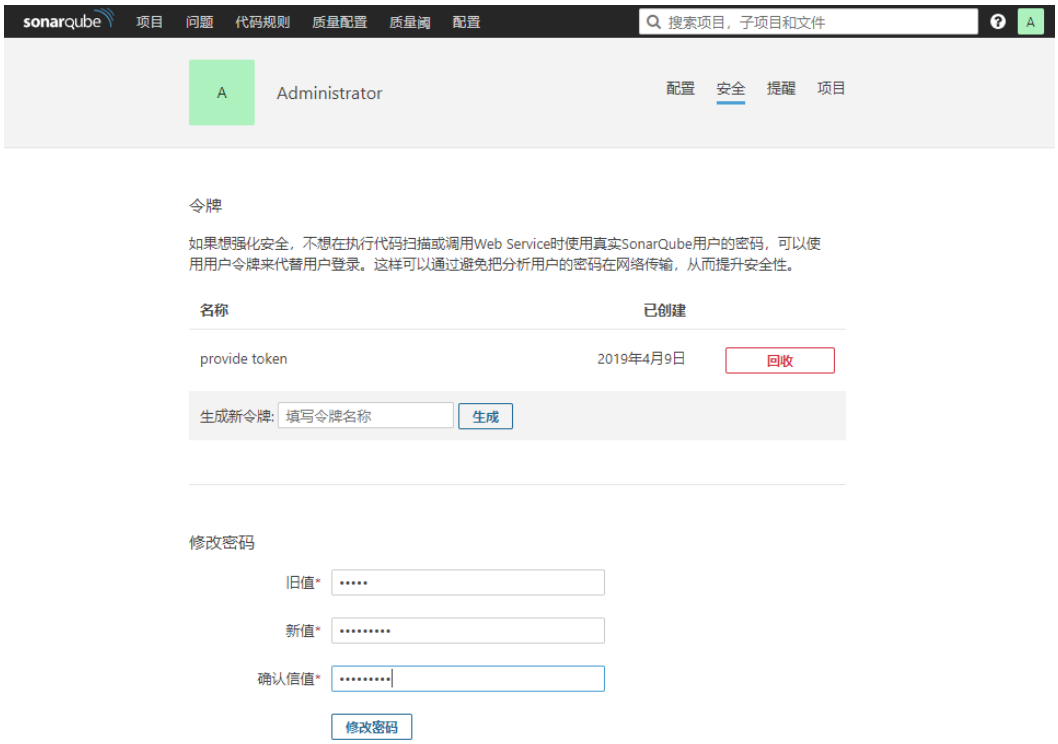
第一次使用开机启动时需要为/etc/rc.local 文件赋权限

```
chmod +x /etc/rc.local
chmod +x /etc/rc.d/rc.local
```

修改管理员 admin 账号密码

浏览器访问：http://10.0.0.85:9000/sonar/account/security/

图 3-30 管理员账号密码修改



3.19 Jenkins 依赖环境准备

安装 maven

1. 复制文件apache-maven-3.8.1-bin.tar.gz 到 /usr/local

2. 解压
```
cd /usr/local
tar -zxvf apache-maven-3.8.1-bin.tar.gz
```

3. 备份
```
cp apache-maven-3.8.1-bin.tar.gz /root/software/
```

配置 maven

1. 设置环境变量 编辑/etc/profile在文档最后追加以下内容：
```
vi /etc/profile
## maven env
export MAVEN_HOME=/usr/local/apache-maven-3.8.1
export PATH=$PATH:$MAVEN_HOME/bin
```

2. 使配置文件生效：
```
source /etc/profile
```

验证 maven

```
mvn -v
```

正确配置jdk会输出以下内容：

```
[root@localhost apache-maven-3.8.1]# mvn -v
Apache Maven 3.8.1 (ff8f5e7444045639af65f6095c62910b5713f426; 2017-04-04T03:39:06+08:00)
Maven home: /usr/local/apache-maven-3.8.1
Java version: 1.8.0_141, vendor: Oracle Corporation
Java home: /usr/local/java/jdk1.8.0_291/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-514.21.1.el7.x86_64", arch: "amd64", family: "unix"
```

大功告成maven安装成功！

部署 Tomcat8 运行 Jenkins.war

```
cd /root
mkdir software
mkdir servers
```

目录用途说明：software 用于存放所有使用到的安装包文件；servers 用于部署所有tomcat等各种服务器运行程序

准备安装

1. 复制apache-tomcat-8.5.16.tar.gz 到 /root/servers 目录下。

2. 解压
```
cd /root/servers
tar -zxvf apache-tomcat-8.5.16.tar.gz
```

3. 备份安装包到/root/software 以便以后使用。
```
cp apache-tomcat-8.5.16.tar.gz /root/software
```

4. 重命名apache-tomcat-8.5.16 为 jenkins-tomcat8
```
mv apache-tomcat-8.5.16 jenkins-tomcat8
```

修改端口与配置

编辑conf/server.xml修改端口共3处：

```
vi /root/servers/jenkins-tomcat8/conf/server.xml
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" />
```

修改后

```
<Connector port="8080" protocol="HTTP/1.1"
connectionTimeout="20000"
redirectPort="8443" URIEncoding="UTF-8" />
```

URIEncoding="UTF-8" 解决tomcat 中文乱码问题。

内存调优

编辑bin/catalina.sh 设置JAVA_OPTS 提高JVM栈内存Increase JVM heap memory

```
vi /root/servers/jenkins-tomcat8/bin/catalina.sh
JAVA_OPTS="-Djava.awt.headless=true -Dfile.encoding=UTF-8
-server -Xms1024m -Xmx2048m
-XX:NewSize=512m -XX:MaxNewSize=512m -XX:+DisableExplicitGC"
```

添加防火墙例外

1. 使用root用户执行命令 开启8080 端口。
   ```
   firewall-cmd --zone=public --add-port=8080/tcp --permanent
   ```

2. 重新加载
   ```
   firewall-cmd --reload
   ```

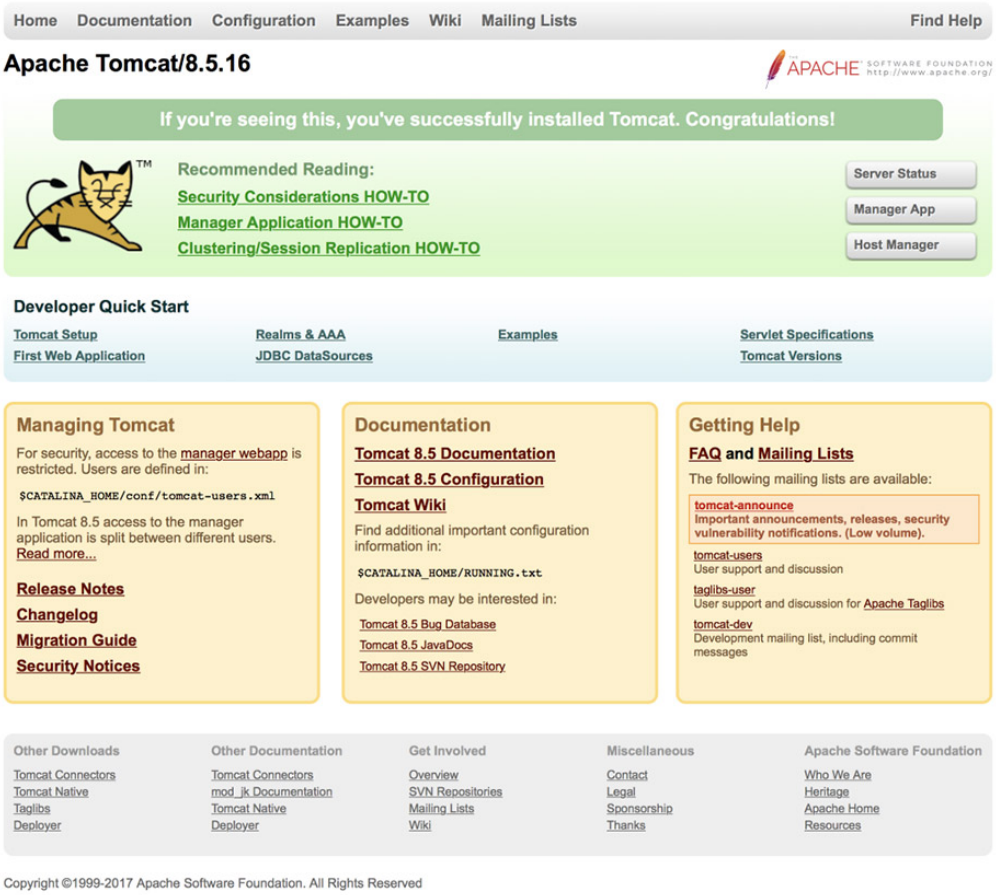
启动并验证服务器运行是否正常

1. 启动tomcat服务
   ```
   /opt/jenkins-tomcat8/bin/startup.sh
   ```

2. 查看日志
   ```
   tail -f /opt/jenkins-tomcat8/logs/catalina.out
   ```

3. 浏览器访问地址：http://server_ip:8080

图 3-31 启动



看到这个页面说明tomcat服务器部署成功了！

设置 Jenkins-tomcat 开机启动

使用root用户编辑/etc/rc.local文件，执行命令：

```
vi /etc/rc.local
```

加入：

```
## jenkins-tomcat8
su - root -c '/opt/jenkins-tomcat8/bin/startup.sh'
```

温馨提示：

第一次使用开机启动时需要为/etc/rc.local 文件赋权限

```
chmod +x /etc/rc.local
chmod +x /etc/rc.d/rc.local
```

3.20 Jenkins 安装与配置

部署 jenkins.war

1. 删除tomcat-tomcat8/webapps/ 里面所有项目文件
```
rm -rf /root/servers/jenkins-tomcat8/webapps/*
```

2. 复制jenkins.war到/root/servers/jenkins-tomcat8/webapps/ 目录下

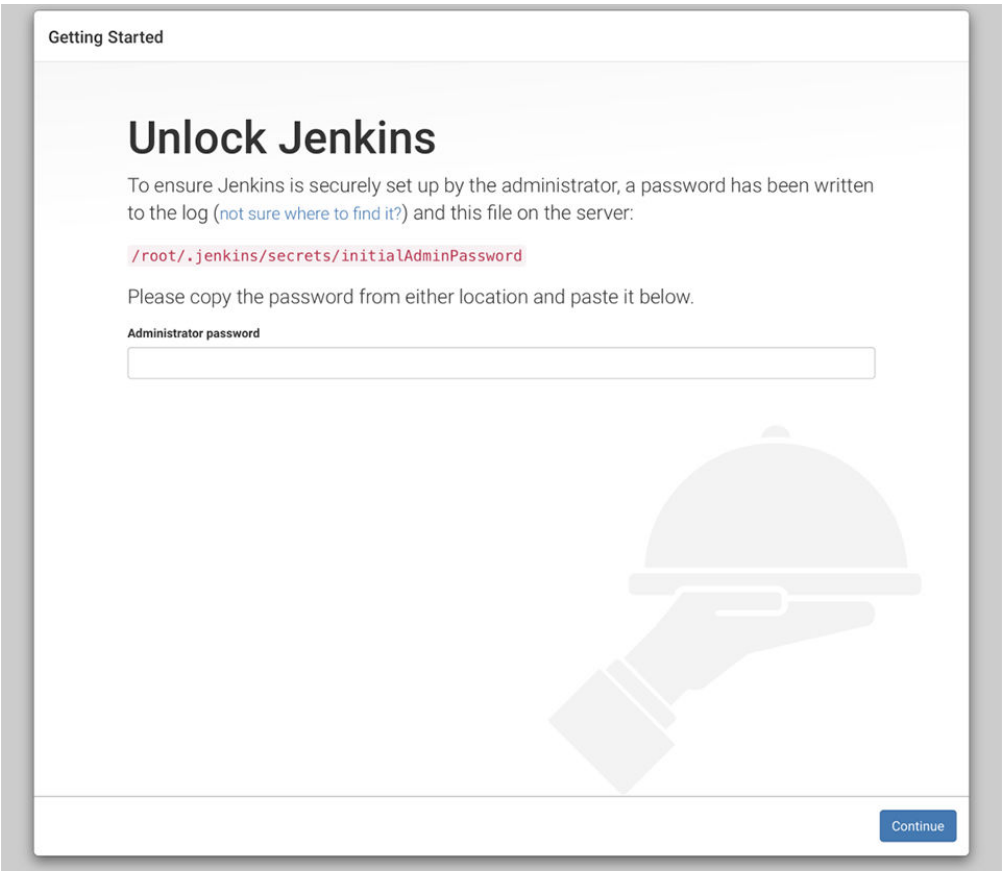
3. 停止并重新启动tomcat服务
```
/root/servers/jenkins-tomcat8/bin/shutdown.sh
/root/servers/jenkins-tomcat8/bin/startup.sh
```

启动 jenkins 安装程序

1. 浏览器访问地址：http://server_ip:8080/jenkins运行jenkins安装程序。

   首次访问jenkins需要提供超级管理员密码，超级管理员密码串在jenkins首次启动后会自动生成，存放在/root/.jenkins/secrets/initialAdminPassword 文件中。

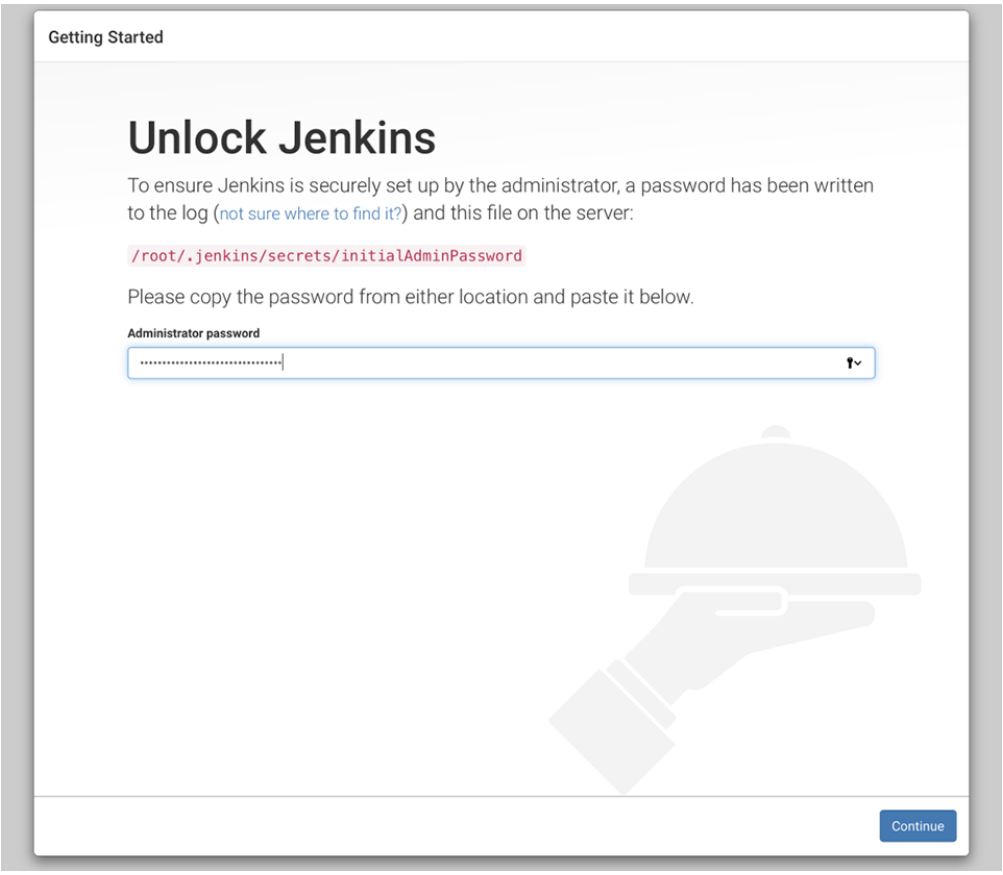
   图 3-32 安装 1



2. 获取超级管理密码执行命令：
```
cat /root/.jenkins/secrets/initialAdminPassword
[root@localhost ~]# cat /root/.jenkins/secrets/initialAdminPassword
```

3. 复制密码串并粘贴到Administrator password输入框内单击Continue

图 3-33 安装 2



插件安装

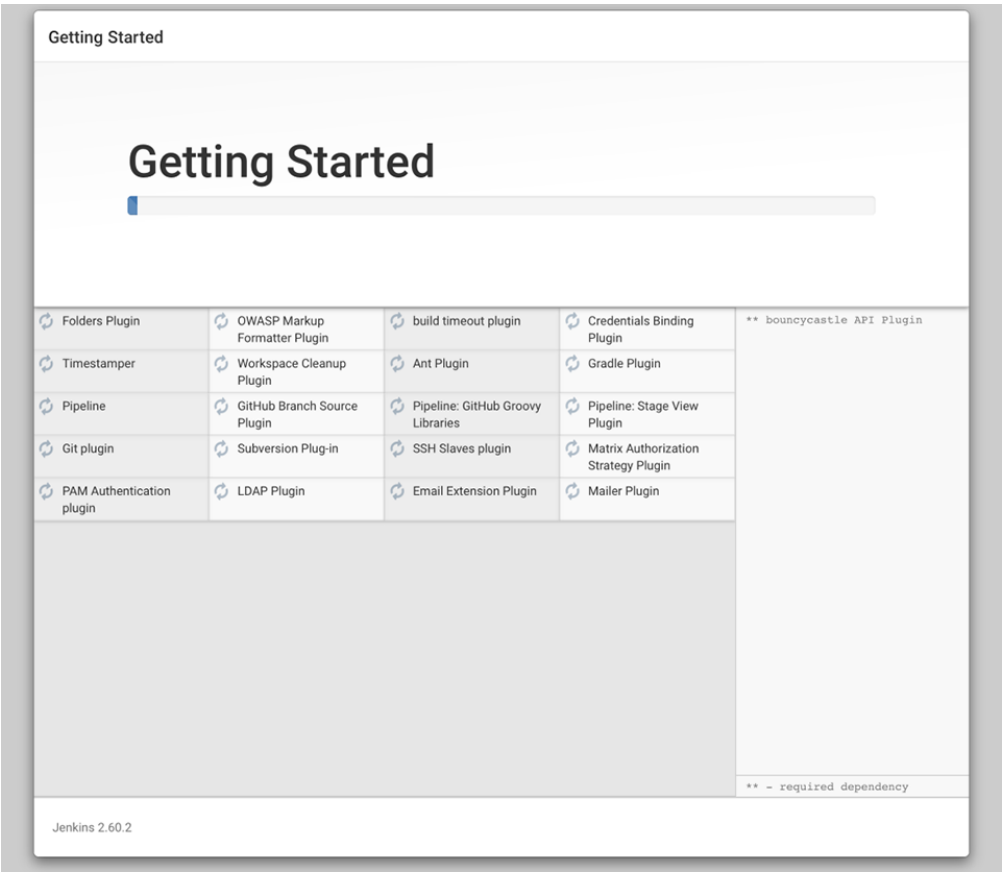
图 3-34 插件安装 1

1. 选择安装模式

   鼠标单击左侧蓝色框安装社区推荐插件包Install suggested plugins Install plugins the Jenkins community finds most useful 安装插件，这一步安装插件受网络影响安装时间可能比较长，建议普通用户选择推荐安装；

   右侧灰色背景框是插件可选框，可以理解为插件自定义选择安装，建议有一些经验的用户使用；本演示教程已经典安装模式为例。

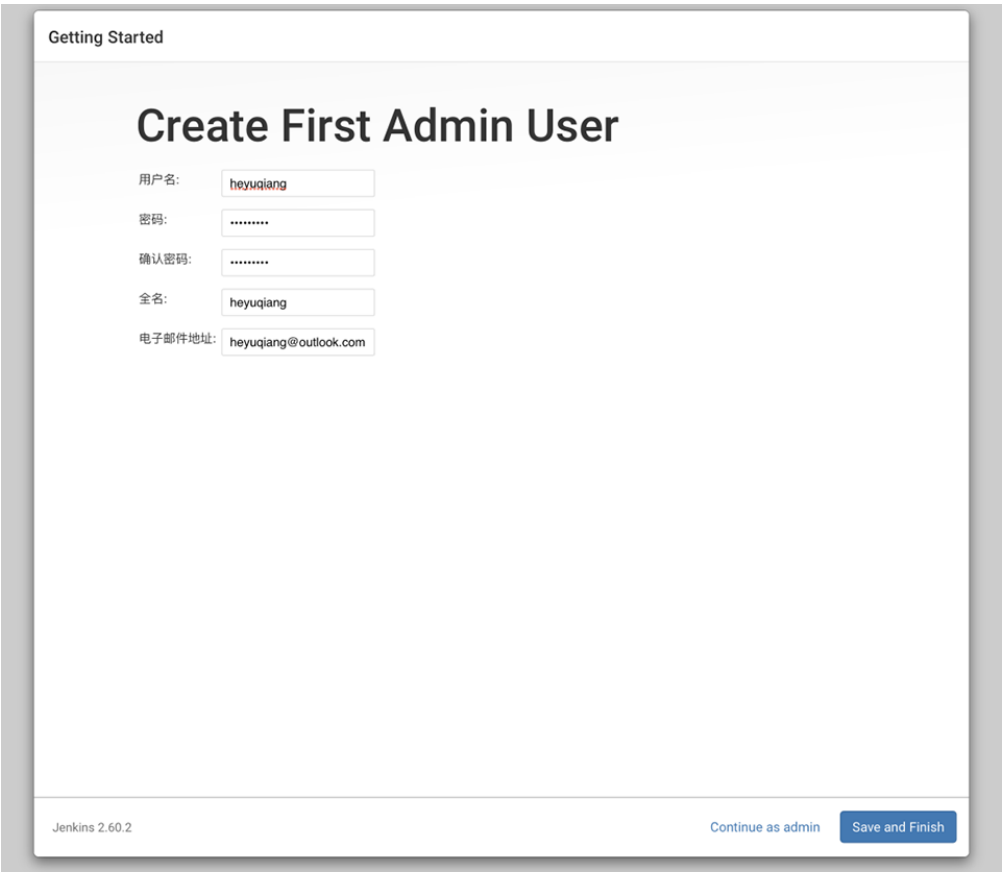
图 3-35 插件安装 2



开始安装插件包，在右侧可以看到事实的安装日志。绿色对勾表示安装成功

2. 创建Admin用户
   这里先简单设置一个用户，后续的配置中会创建一个超级管理员账户。

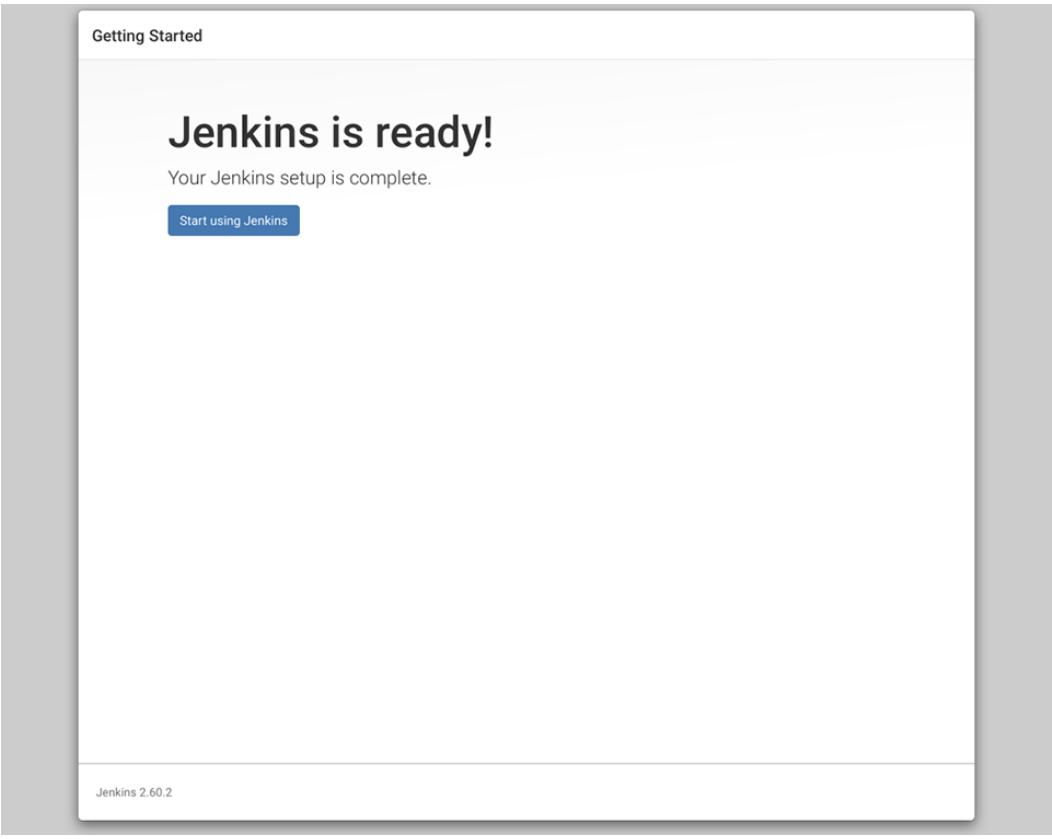
图 3-36 插件安装 3



单击Save and Finish 完成用户创建。

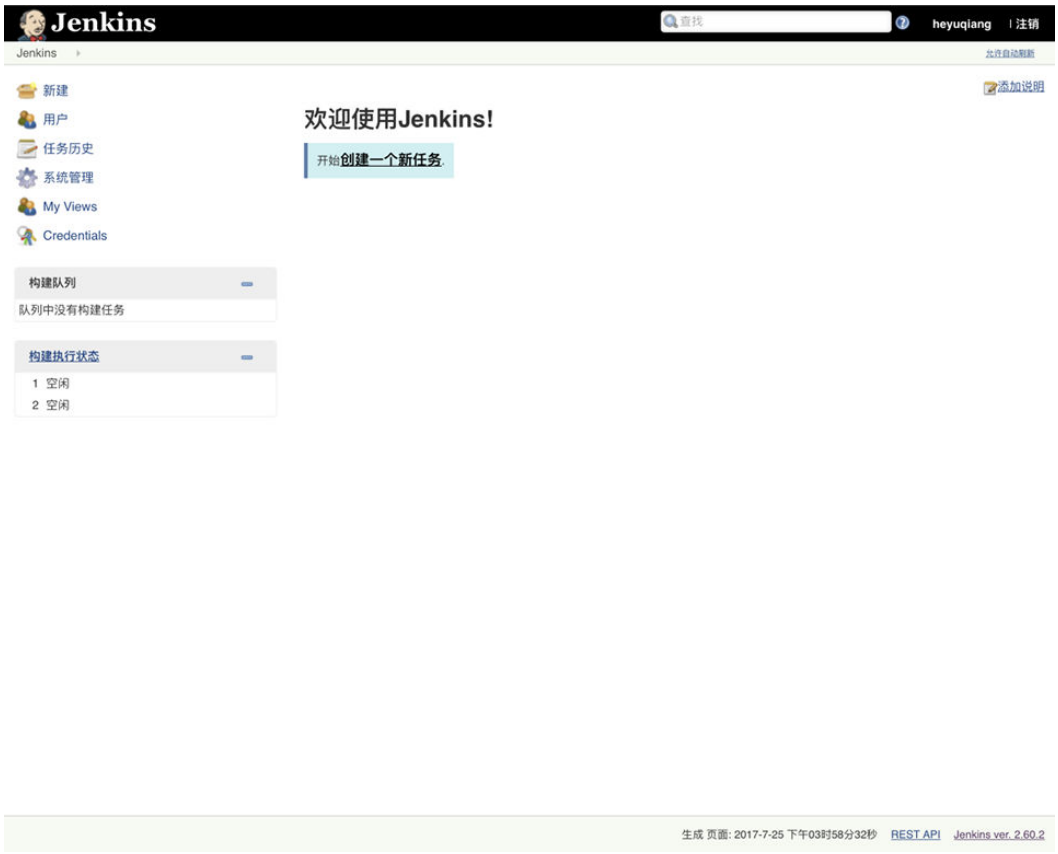
安装完成

图 3-37 安装完成



至此，Jenkins的初步安装就已经完成啦！接下来还要做进一的步配置和优化。

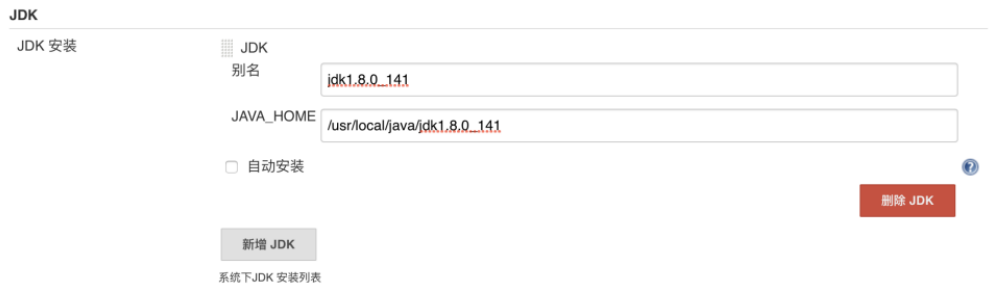
图 3-38 开始使用



全局工具配置

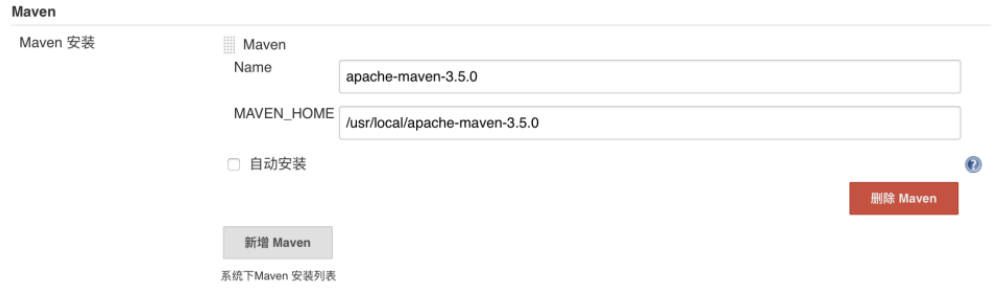
1. 配置JDK
   别名：jdk1.8.0_291
   JAVA_HOME ： /usr/local/jdk1.8.0_291

   图 3-39 配置 JDK



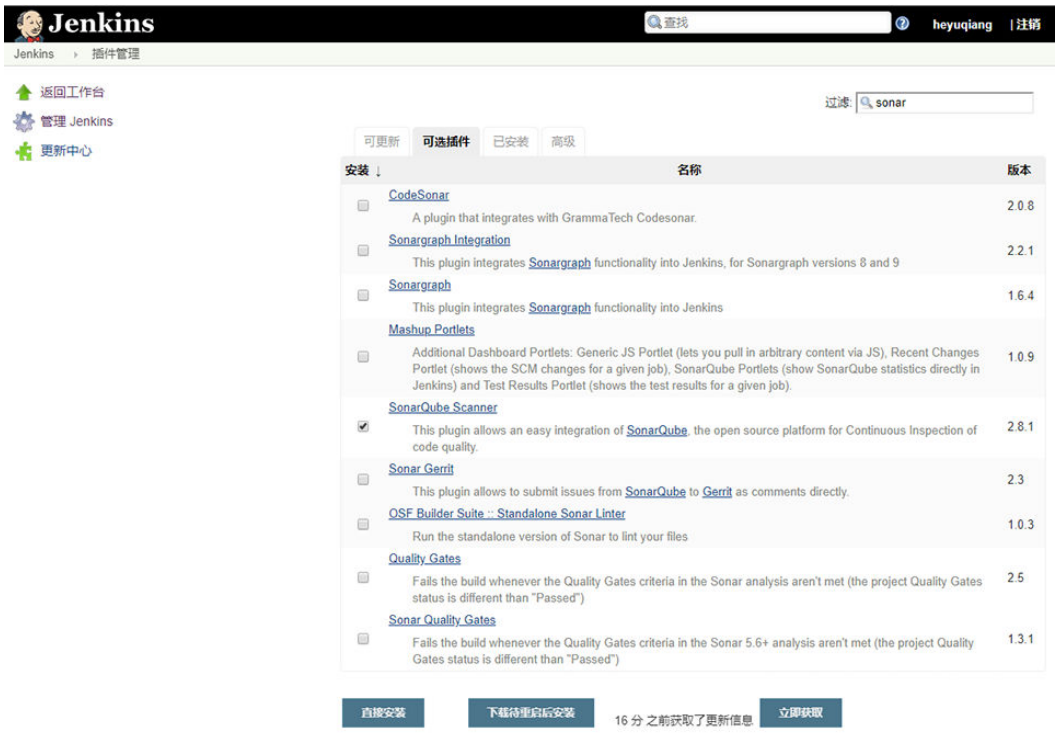
2. 配置MAVEN
   Name apache-maven-3.8.1
   MAVEN_HOME /usr/local/apache-maven-3.8.1

图 3-40 配置 MAVEN



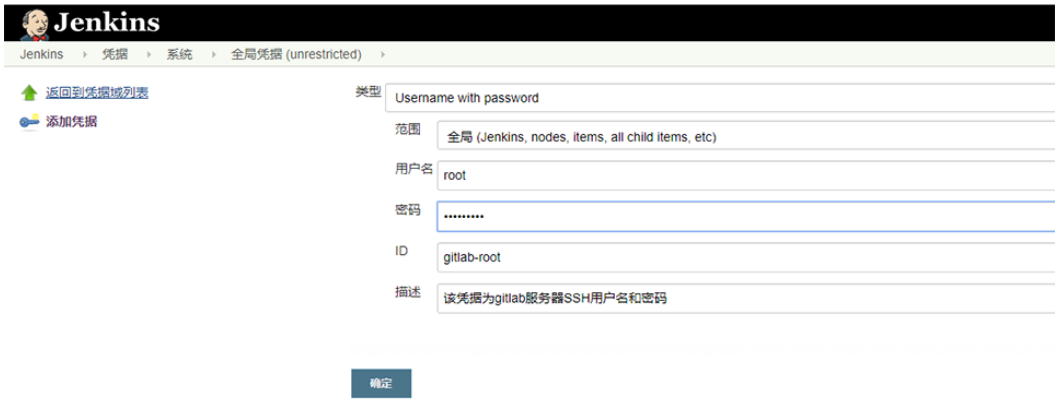
SonarQube Scanner 插件安装

图 3-41 插件安装



创建 gitlab-root 凭据

图 3-42 创建凭据



该凭据为gitlab的用户名和密码，ID也可以修改但是要在portal-web同步配置。

Configure Global Security

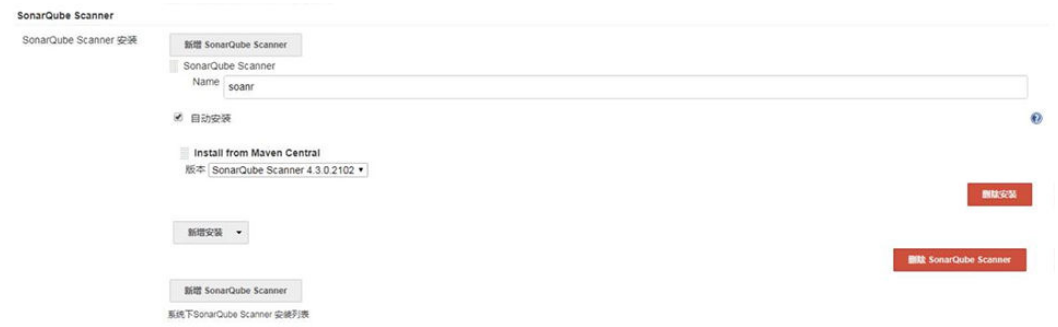
跨站请求伪造保护去掉对勾：

□ 防止跨站点请求伪造

配置 sonar scanner

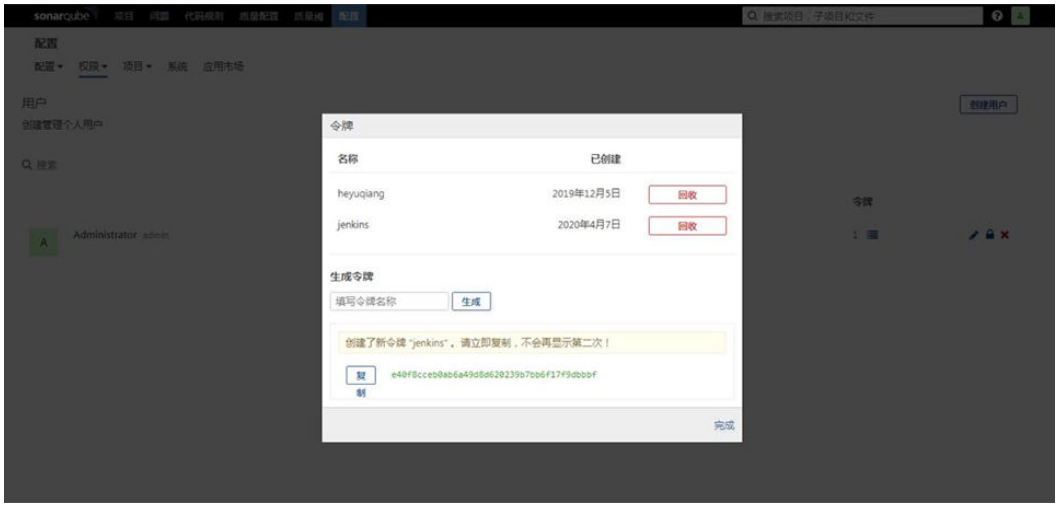
配置自动安装Sonar Scanner

图 3-43 自动安装



Sonerqube server 生成token

图 3-44 生成 token



Jenkins 添加Sonarqube token

图 3-45 凭据

配置Sonarqube Servers

图 3-46 配置 Sonarqube Servers

注意事项

1. minio打包文件上传记得开启utf-8解码
2. nginx设置的上传文件大小调整（client_max_body_size 500m）
3. 修改数据后需要清楚指定数据项缓存（邮箱开启sys_dictionary sys库）（调用sys服务的接口）

4. 请redis接口/api/user/role/releaseRoleAuthCache/3
   /api/user/role/releaseRoleAuthCache/4

验证是否好用

1. 检查网关项目所有服务接口是否能正常访问
2. 检查nacos的服务是否都正常切属于同组
3. 转换服务，上传文档，视频，正常转换
4. websocrt 课堂教学可以正常进行

nginx配置文件：

Minio服务器配置文件+前后端服务器配置文件

后台jar包启动方式：

```
java -jar sys-service-3.4.0-SNAPSHOT.jar --spring.config.location[0]=./application.yml --
spring.config.location[1]=./bootstrap.yml --spring.profiles.active=pro
```

3.21 安装 NFS 服务端和客户端

操作系统

CentOS Linux release 7.2.1511 (Core)

NFS 服务端安装

```
yum install nfs-utils -y
```

- **创建共享**

  a. 在/etc/exports配置文件中添加以下内容，保存退出；
  ```
  /sfs-date/mnt-pro/lab 172.16.79.15/255.255.254.0(rw,sync,no_root_squash)
  ```
  或者
  ```
  /home 168.170.249.1/24(rw,sync,no_root_squash)
  ```

  b. 创建共享目录并给予写的权限
  ```
  mkdir /home/ITP/share
  chmod o+w /home/ITP/share
  ```

- **启动NFS服务**
  ```
  systemctl restart rpcbind
  systemctl restart nfs-server（nfs）
  systemctl enable rpcbind
  systemctl enable nfs-server(nfs)
  ```

NFS 客户端安装

```
yum install nfs-utils -y
```

1. 创建挂载目录
   ```
   mkdir -p /mnt/lab
   ```

2. 挂载NFS共享
```
mount -t nfs 59.68.143.78:/home/ITP/share /home/ITP/share
```

3. 查看客户端挂载信息
```
df -h
```

4 修订记录

表 4-1 修订记录

发布日期	修订记录
2024-05-08	规范词、敏感词专项处理，章节优化
2024-01-04	第一次正式发布。