分布式部署

技术文档

版本：（1.0）

日期：2019.06.17

负责公司：高专资讯科技

**文档信息**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 项目名称**：**  **互太Mes系统** | 项目负责人**：**  **Tony（黄记新）** | 负责人电话**：**  **13926205227** | 文档起草作者**：**  **张志成** | 文档起草日期**：**  **2019.06.17** |
| 复审人**：** | 复审日期**：** |  |  |  |

**版本记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 版本编号 | 版本日期 | 修改人 | 备注 |
| **V1.0** | **2019.06.17** | **张志成** |  |
|  |  |  |  |

1. **简介**

## 背景

## 定义，缩略语

### 专业名词解释

|  |  |
| --- | --- |
| 术语 | 解释 |
| …… | …… |

【通过简短的篇幅说明本项目的来由和约束等，让读者可以在短时间内把握本项目的大致情况。】

### 用例名词解释

|  |  |
| --- | --- |
| 术语 | 解释 |
| BD | 本地（数据库） |
| …… | …… |

## 约束

【列出会影响系统的需求、设计、实现或测试方式的主要限制或约束。可在“非功能性需求”一节中进一步说明。】

## 参考资料以及相关业务联系人

【列出文档所参考的上游文档、资料等。】

|  |  |  |
| --- | --- | --- |
| **资料名称** | **版本/日期** | **说明** |
| 【例：合同】 |  |  |
| 【例：投标书】 |  |  |
| 【例：技术方案】 |  |  |

# 目标、涉众分析和范围

## 目标

【用简单的几句话描述本系统的目标，目标体现了系统的最终效果。】

## 涉众分析

【通过组织架构图、涉众分析表等方式，列出本系统各涉众的关注点。】

* 工厂使用人员，系统开发人员与运维人员
* 涉众分析表

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **涉众** | **代表人物** | **待解决问题/对系统的期望** |
|  |  |  |  |
|  |  |  |  |

## 范围

【说明系统的总体范围、与其他系统的关系、系统的地域使用范围等。说明本项目包括的服务内容，如软硬件采购、培训、实施等。需特别说明本系统不包括什么内容。】

### 系统总体范围

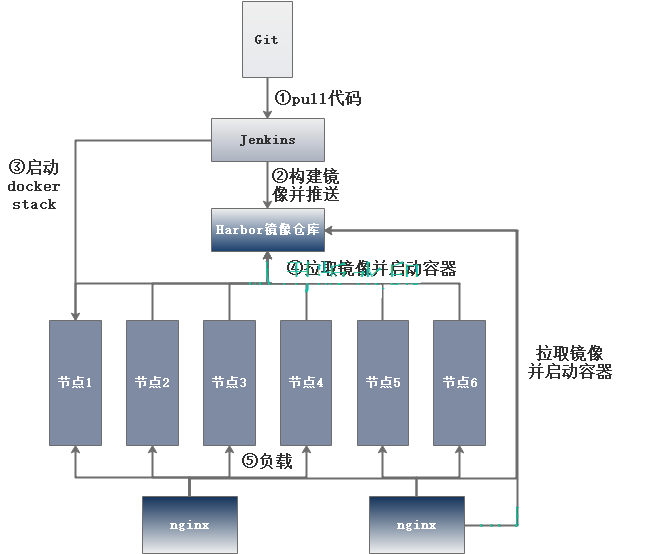
### 关联外部系统

|  |  |  |
| --- | --- | --- |
| **序号** | **系统名称** | **主要提供服务** |
| 1 |  |  |

### 系统的地域使用范围

### 其他需注意事项

# 结构



# Swarm初始化

## 节点1

docker swarm init --advertise-addr <NODE1\_IP>

## 节点2-6

docker swarm join --token <token> --advertise-addr <NODE[2-6]\_IP> <NODE1\_IP>:2377

## 创建网络

docker network create --subnet 10.254.3.0/24 --gateway 10.254.3.1 -d overlay --attachable swarm-overlay

# 搭建Rancher/server

## 启动

docker run -d --restart=unless-stopped \

-p 18080:8080 \

rancher/server:stable

如果需要部署高可用，需要使用外部数据库，再使用nginx负载

docker run -d --restart=unless-stopped \

-p 18080:8080 rancher/server \

--db-host myhost.example.com \

--db-port 3306 \

--db-user username \

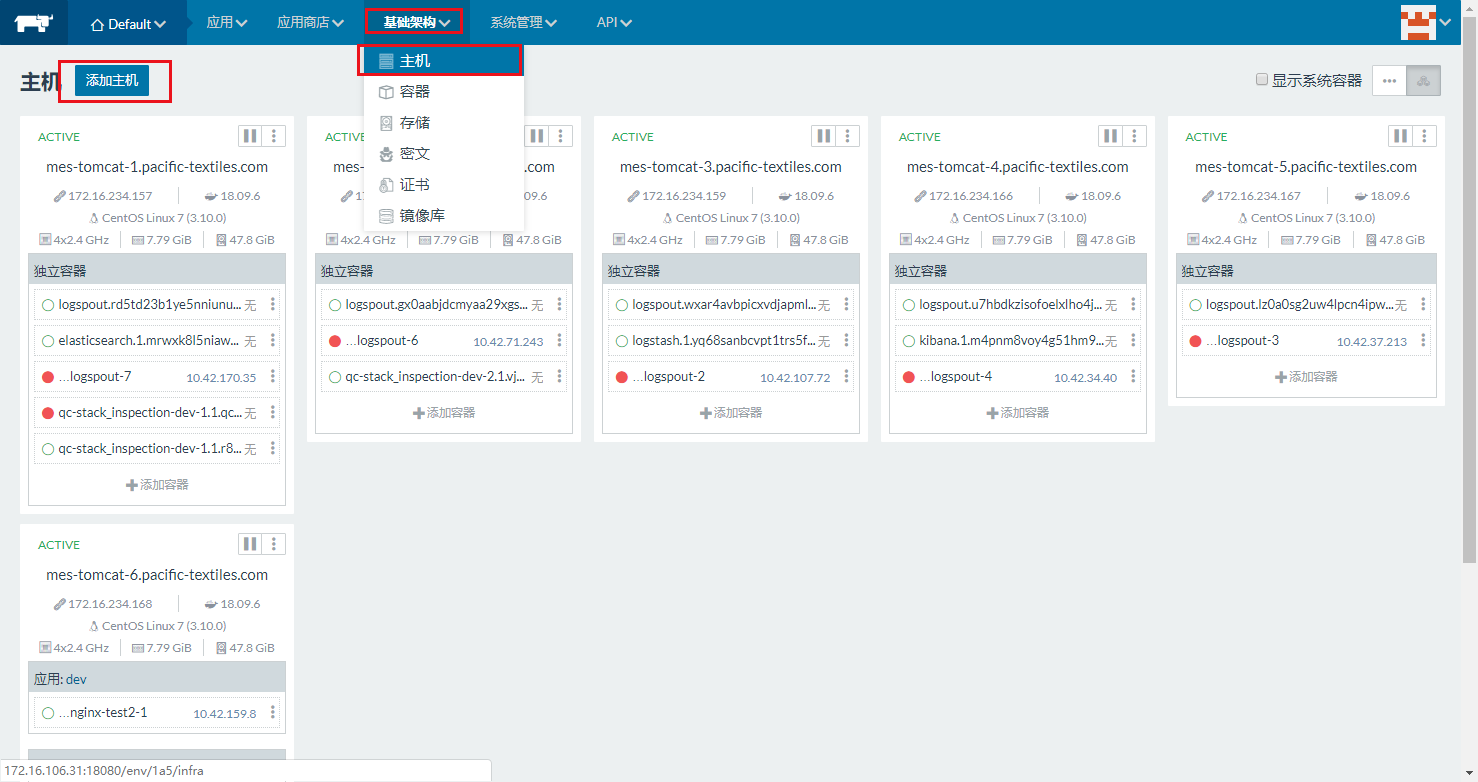
--db-pass password \

--db-name cattle

访问<http://IP:18080>页面

## 添加主机

（1）点击基础架构 -> 主机 -> 添加主机



（2）在步骤4中输入要添加主机的IP，并点击步骤5后的复制按钮

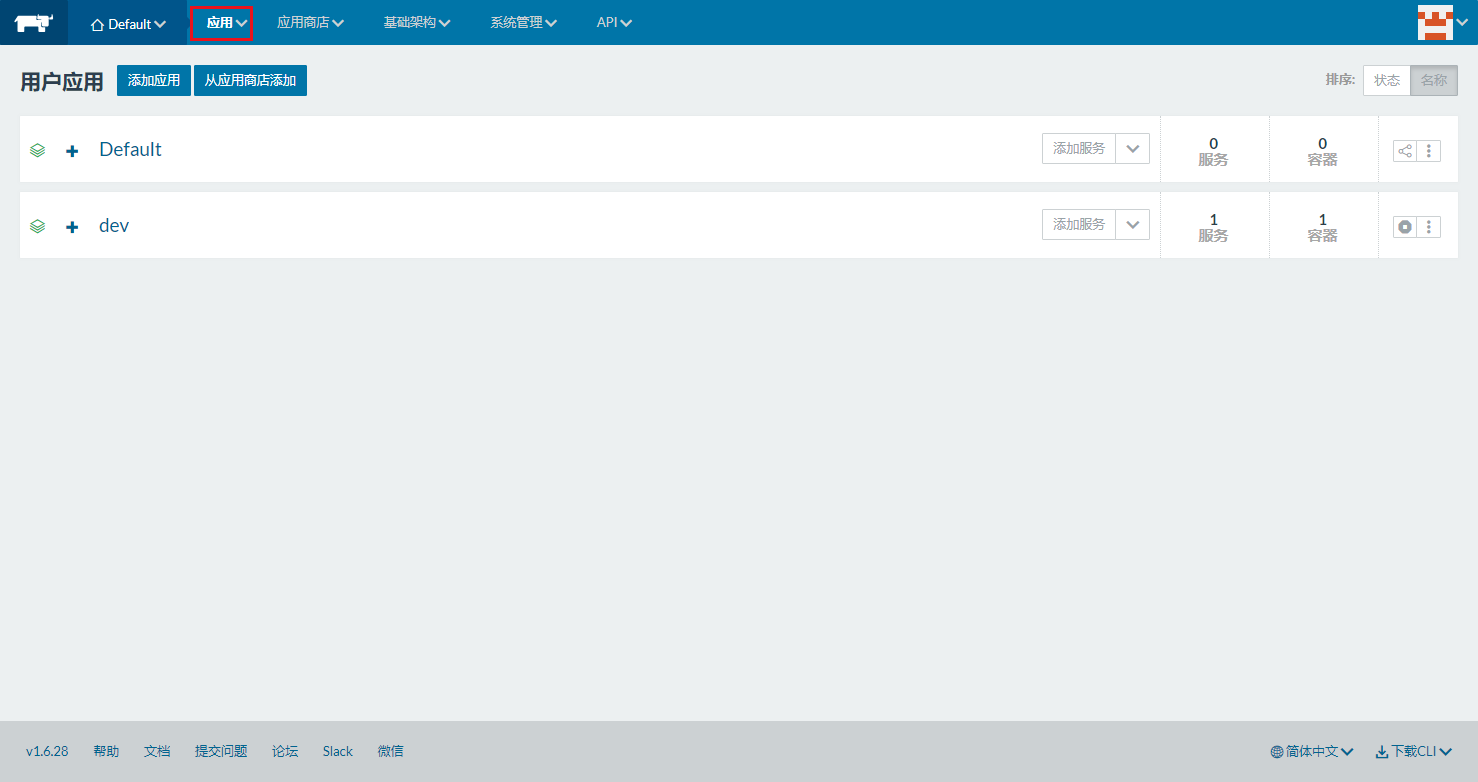


（3）在需要添加的，并且IP与上一部对应的主机运行已复制的命令

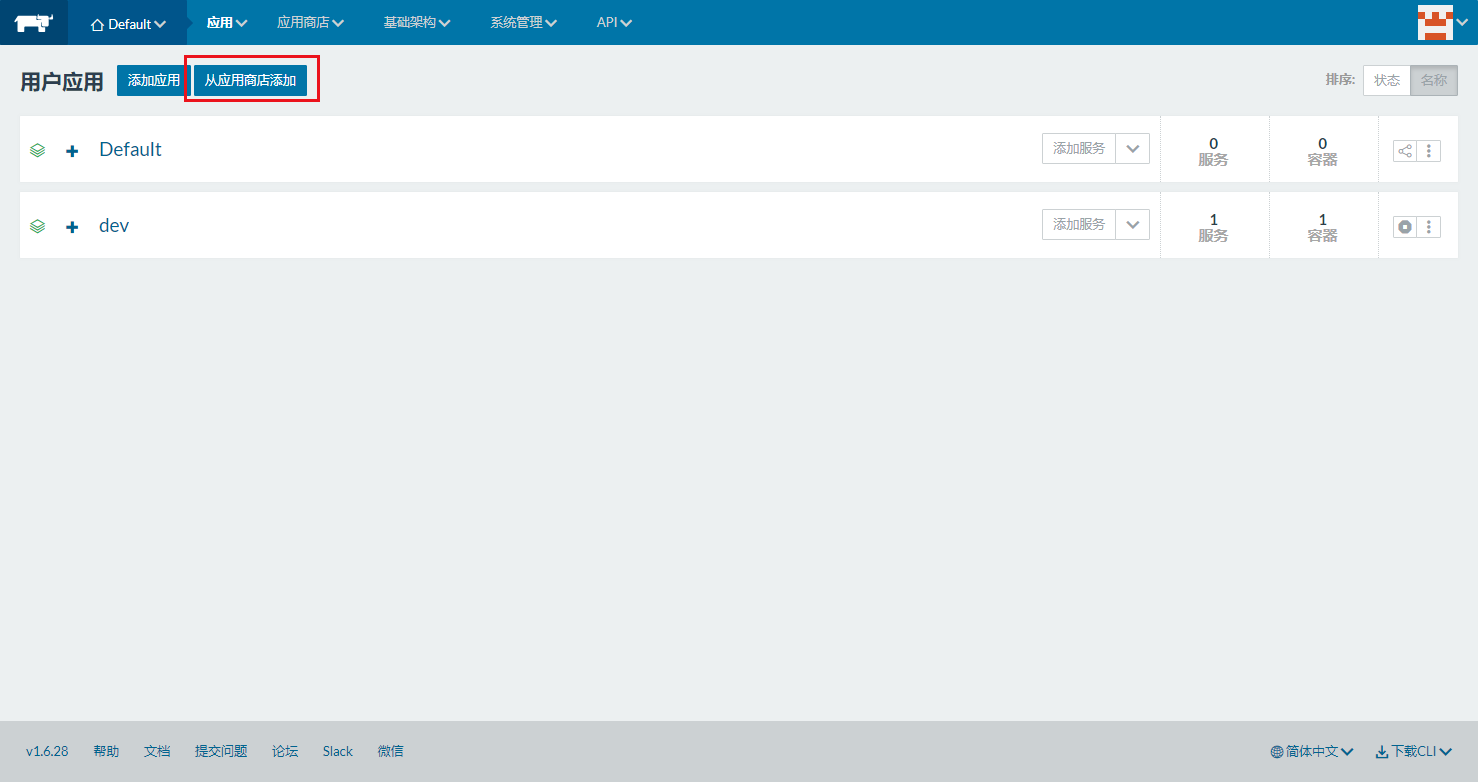
# 搭建Portainer

确认所有主机都添加完毕。

1. 点击应用按钮

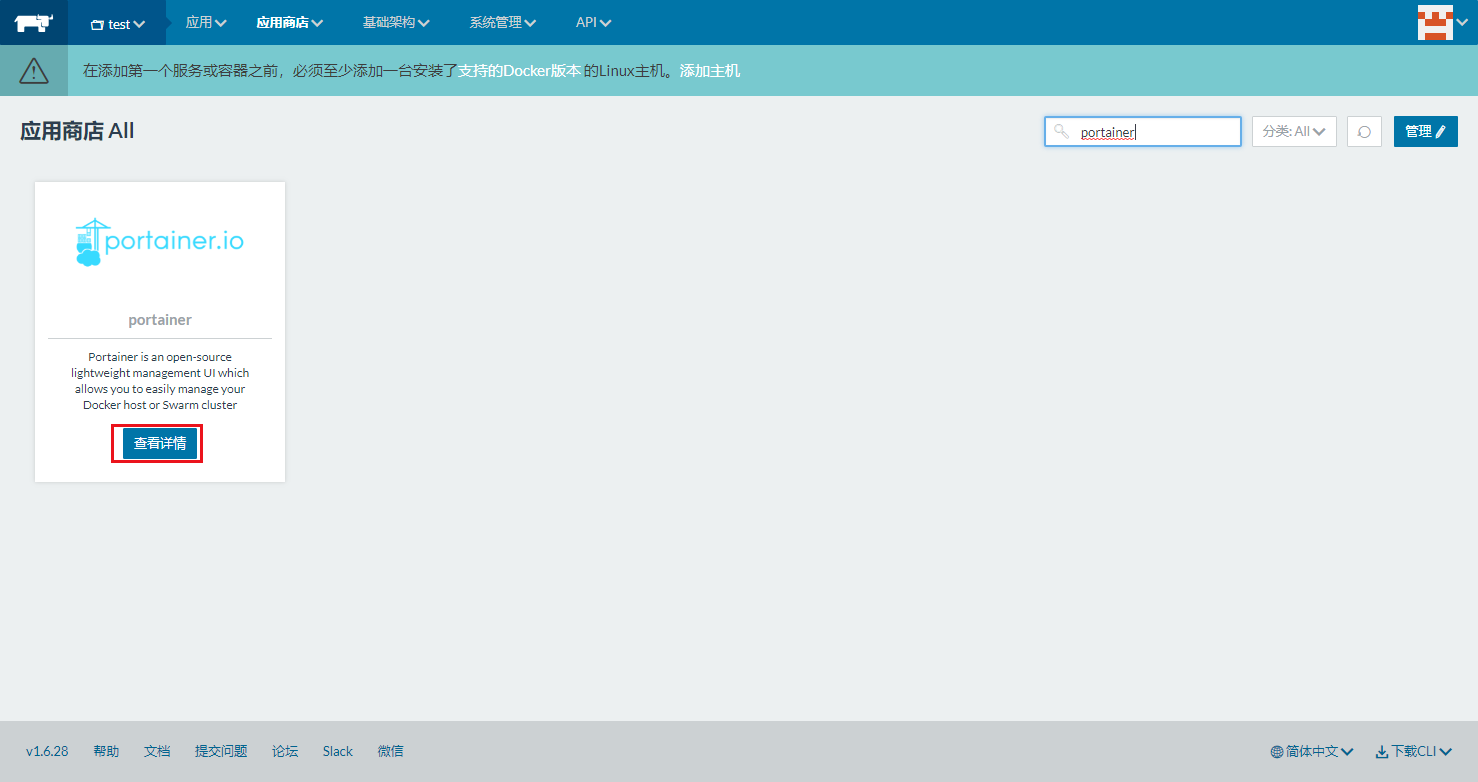


1. 点击从应用商店添加 或 浏览应用商店

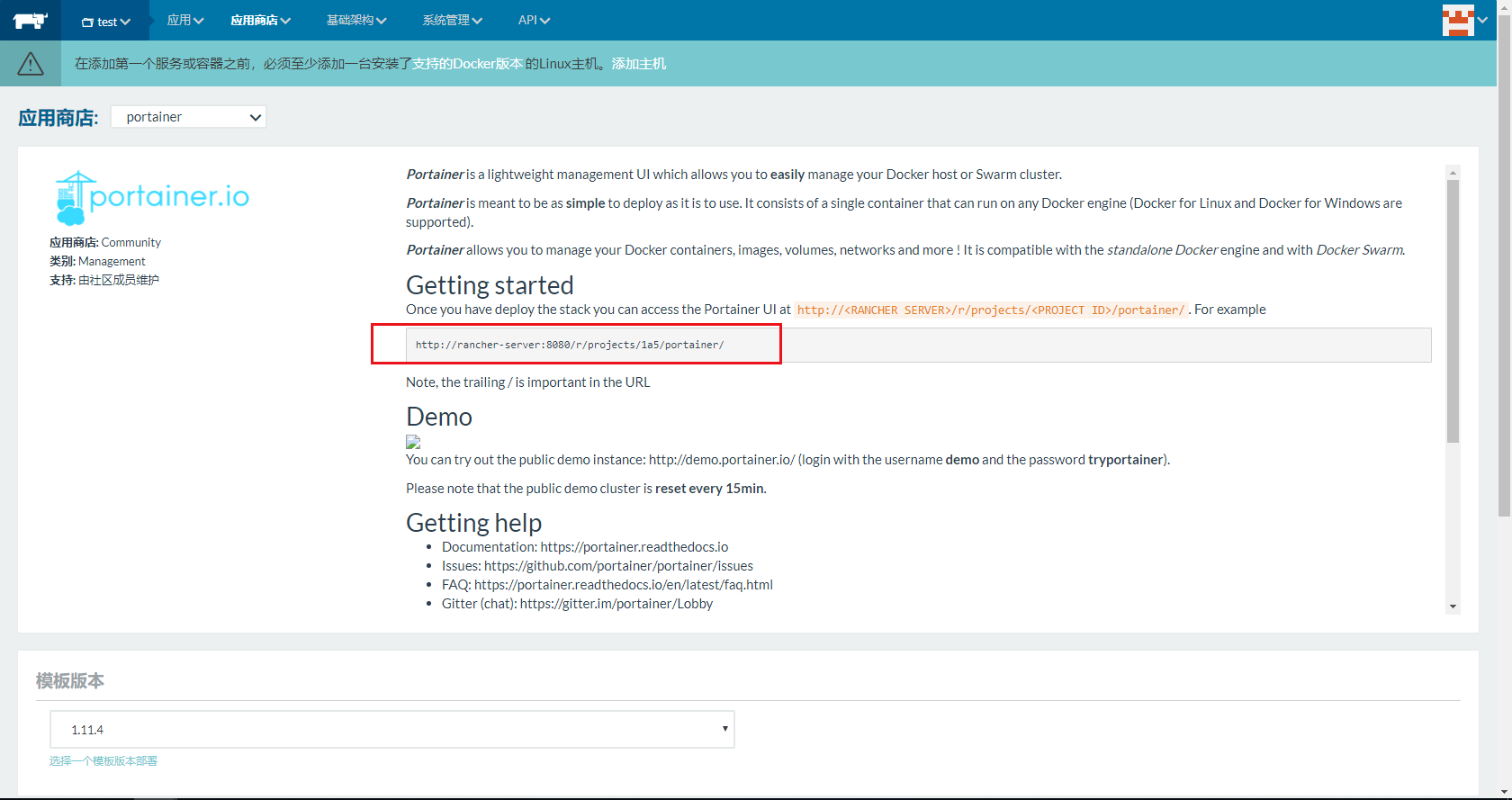




1. 搜索portainer，点击查看详情

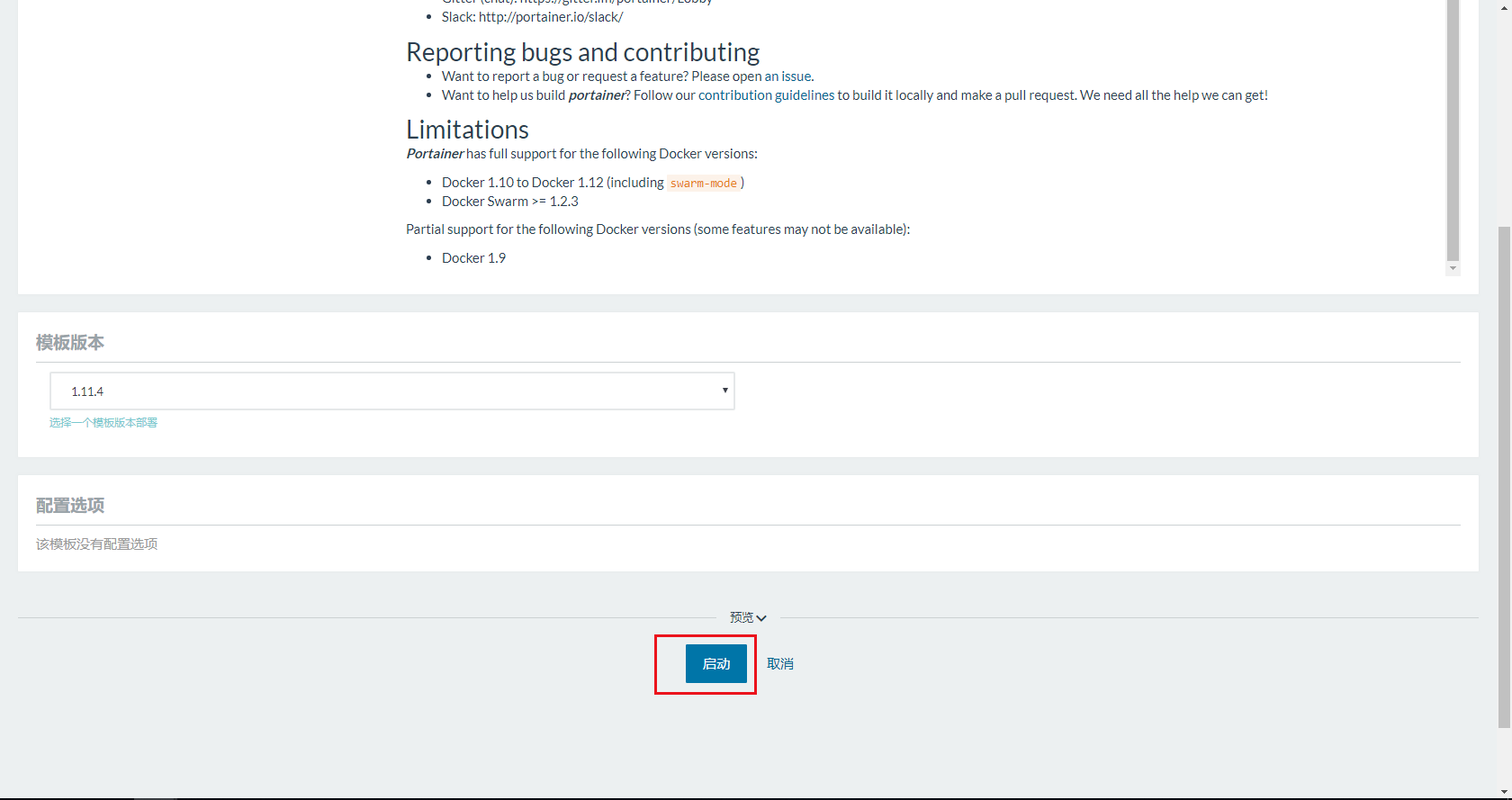


1. 记录此URL



rancher-server 和 8080端口 替换为自己的rancher的IP和端口

1. 启动



# 搭建ELK

## 创建数据卷

在节点1创建数据卷

docker volume create esdata

## 设置虚拟内存区域的数量

每个节点运行：

sysctl -w vm.max\_map\_count=262144

## 创建logstash.conf配置文件

配置文件内容：

input {

udp {

port => 5000

codec => json

}

}

filter {

if [docker][image] =~ /logstash/ {

drop { }

}

}

output {

elasticsearch { hosts => ["elasticsearch:9200"] }

stdout { codec => rubydebug }

}

将配置文件拷贝到所有节点同一目录下

## 编写compose文件

version: '3.3'

services:

elasticsearch:

image: docker.elastic.co/elasticsearch/elasticsearch:7.1.0

ports:

- "19200:9200"

- "19300:9300"

environment:

- ES\_JAVA\_OPTS: "-Xmx256m -Xms256m"

- cluster.name: "docker-cluster"

- network.host: "0.0.0.0"

- discovery.zen.minimum\_master\_nodes: "1"

- discovery.type: "single-node"

networks:

- swarm-overlay

volumes:

- esdata:/usr/share/elasticsearch/data

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

placement:

constraints:

- node.hostname == 节点1的机器名

logstash:

image: docker.elastic.co/logstash/logstash:7.1.0

ports:

- "15000:5000"

- "19600:9600"

environment:

- LS\_JAVA\_OPTS: "-Xmx256m -Xms256m"

- http.host: "0.0.0.0"

- path.config: "/usr/share/logstash/pipeline"

networks:

- swarm-overlay

volumes:

- 文件目录/logstash.conf:/usr/share/logstash/pipeline/logstash.conf

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

logspout:

image: bekt/logspout-logstash

environment:

ROUTE\_URIS: 'logstash://logstash:5000'

volumes:

- /var/run/docker.sock:/var/run/docker.sock

networks:

- swarm-overlay

deploy:

mode: global

restart\_policy:

condition: on-failure

delay: 30s

kibana:

image: docker.elastic.co/kibana/kibana:7.1.0

ports:

- "15601:5601"

environment:

- server.name: "kibana"

- server.host: "0"

- elasticsearch.url: "http://elasticsearch:9200"

networks:

- swarm-overlay

deploy:

mode: replicated

replicas: 1

restart\_policy:

condition: on-failure

networks:

swarm-overlay:

external:

name: swarm-overlay

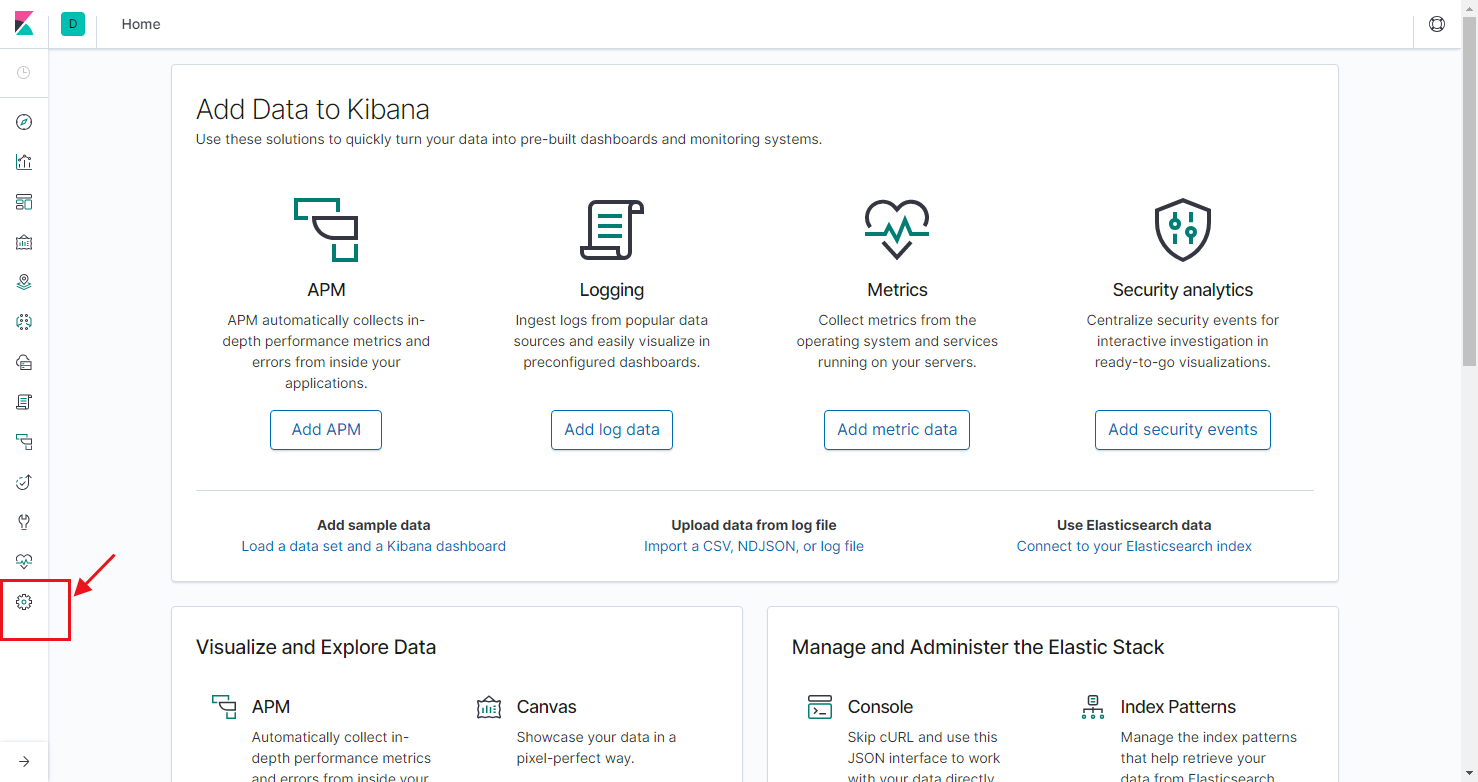
## 启动

docker stack deploy -c <fileName.yml> elk

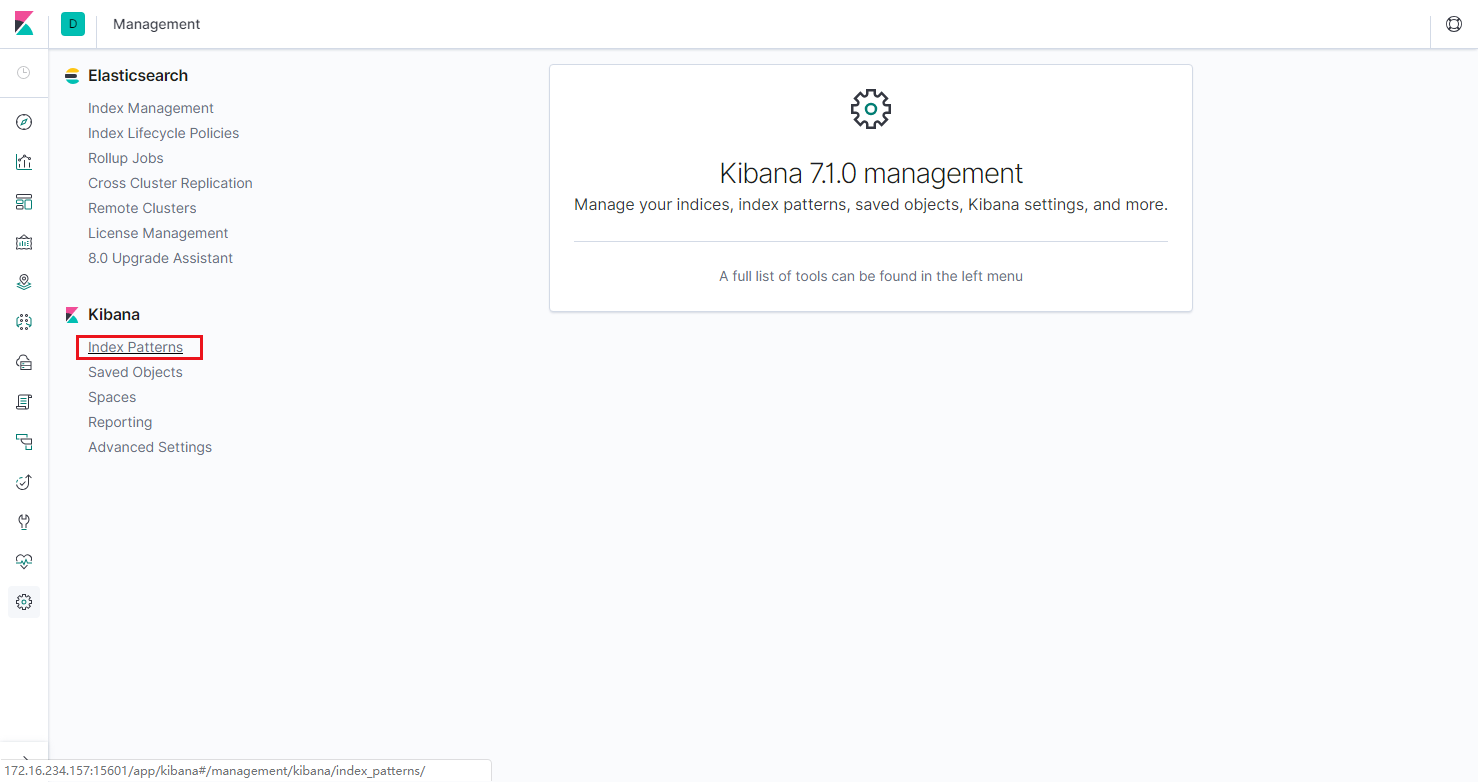
## 验证

访问<http://任一节点IP:15601>

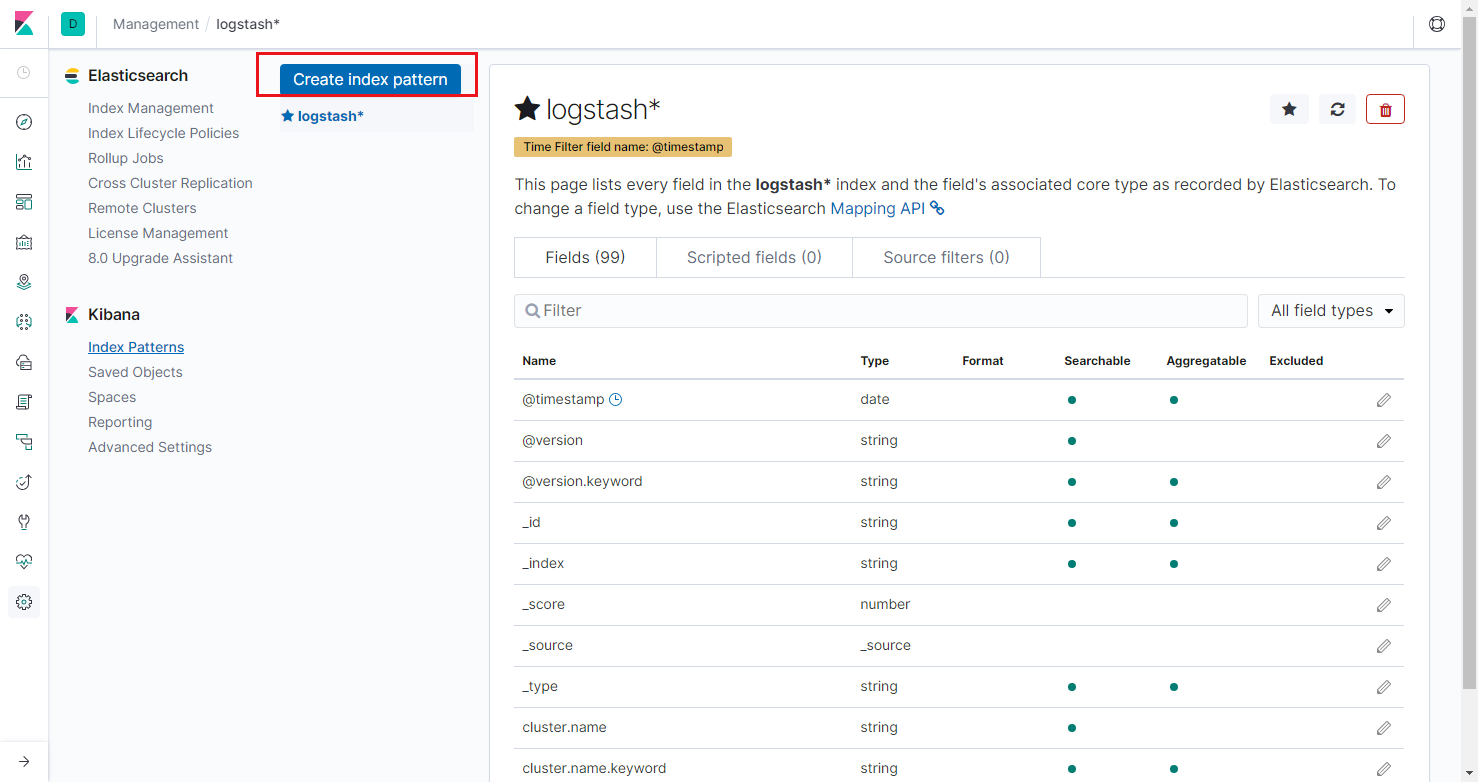
点击左下角设置按钮



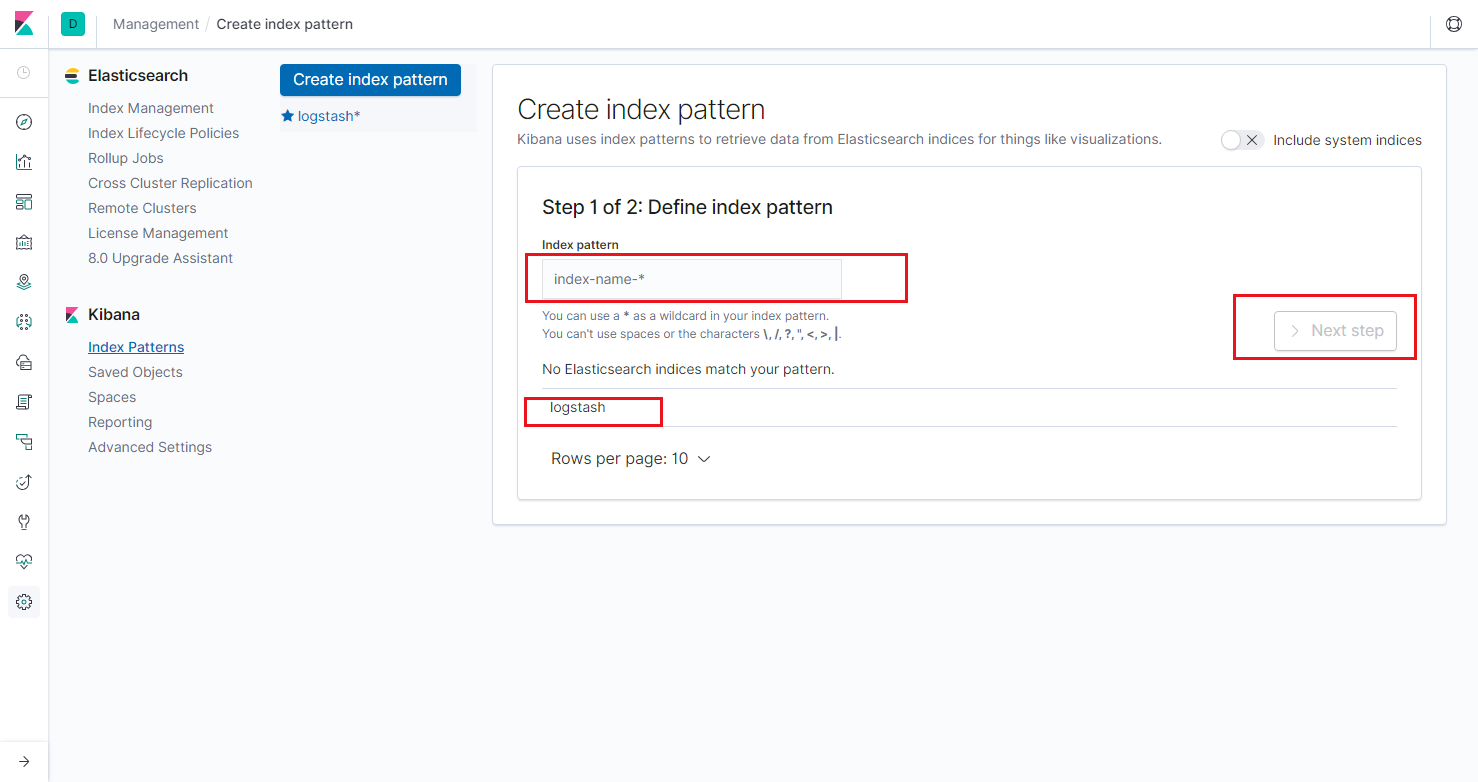
选择Index Patterns



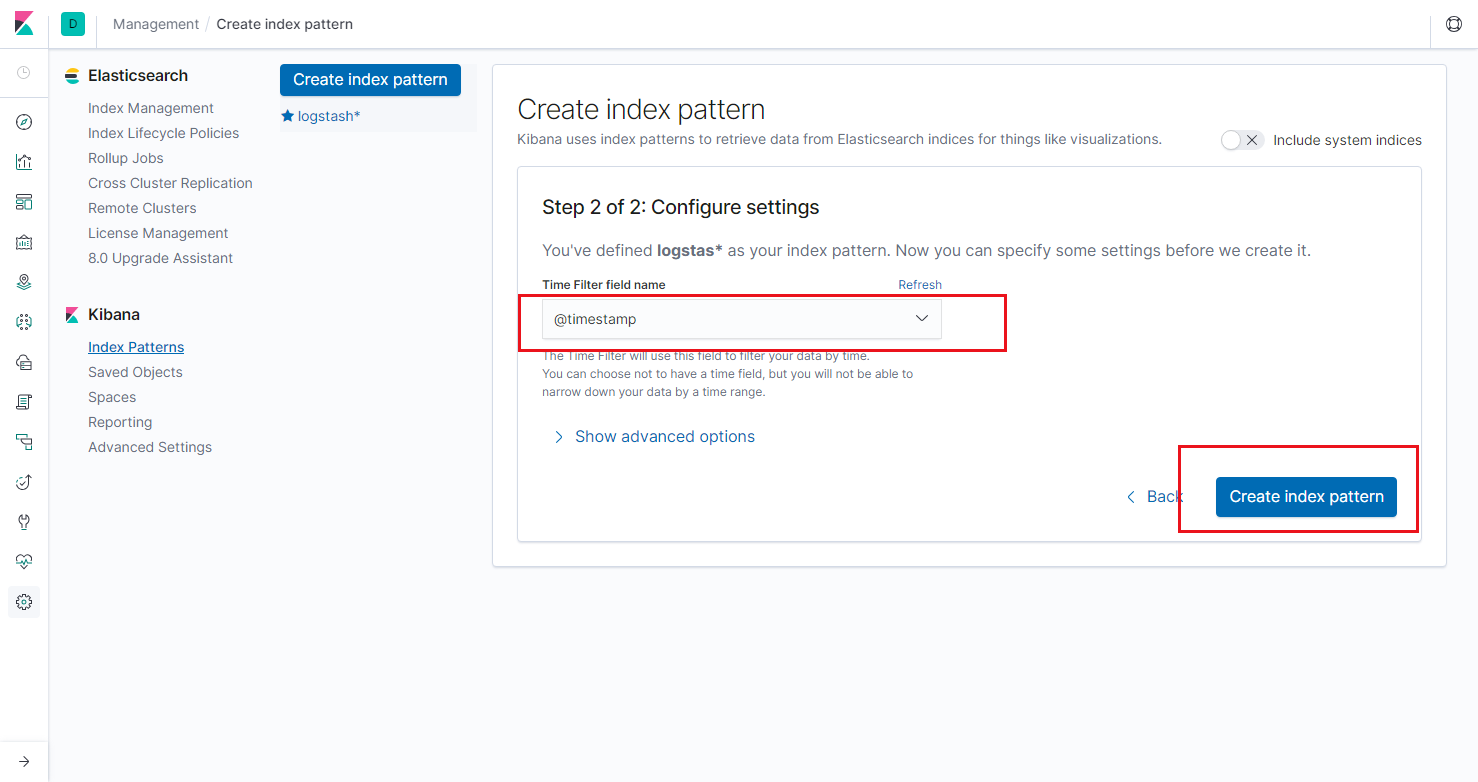
创建索引



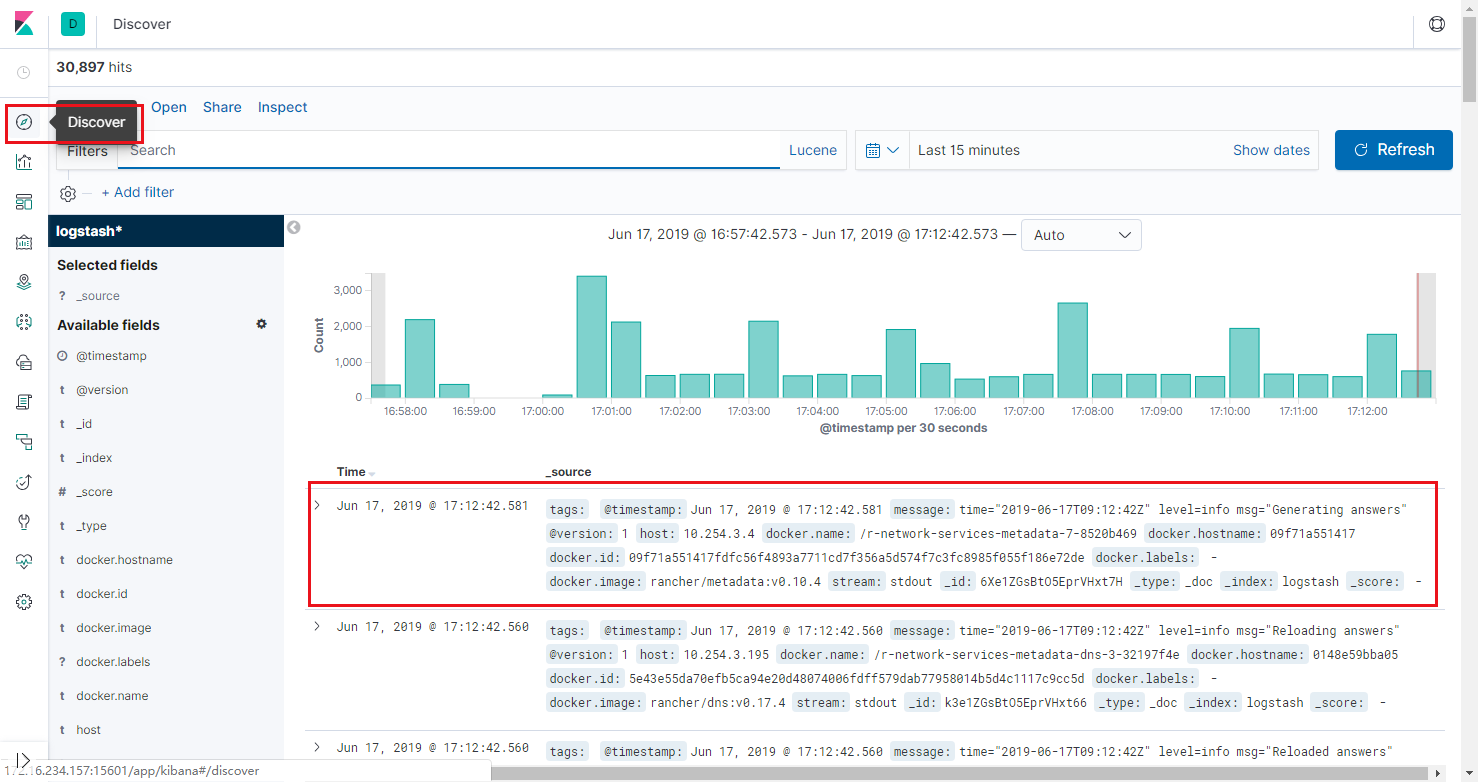
输入logstash\* 并点击下一步



选择@timestamp，并点击创建索引



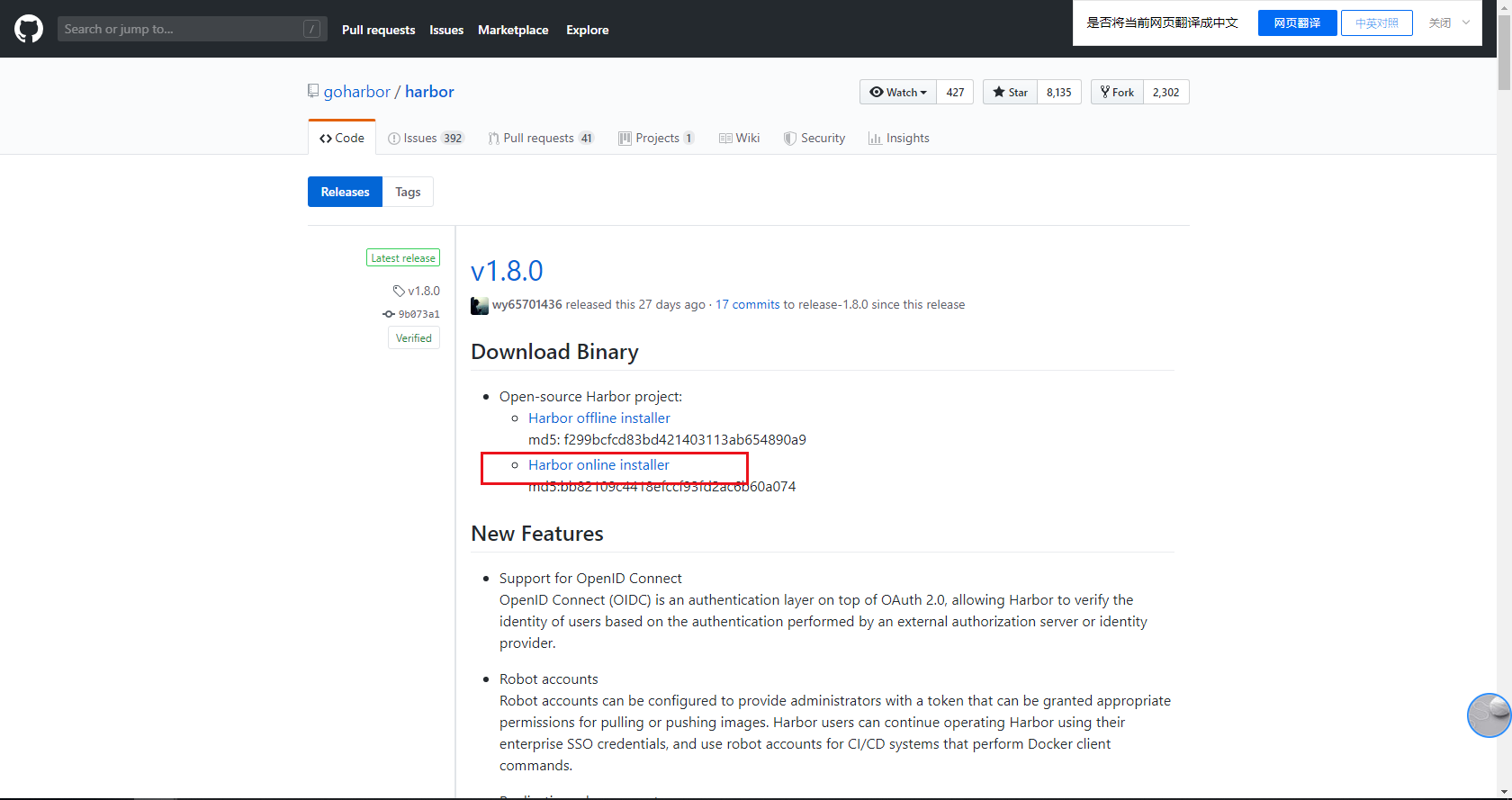
点击左侧Discover，查看/搜索日志



# 搭建Harbor

## 下载

下载地址：https://github.com/goharbor/harbor/releases



解压tar -xf harbor-offline-installer-<version>.tgz

## 配置

vi harbor.yml

hostname: <IP>

http:

port: 80

harbor\_admin\_password: <password> #初始登录密码

## 安装

运行命令

./prepare

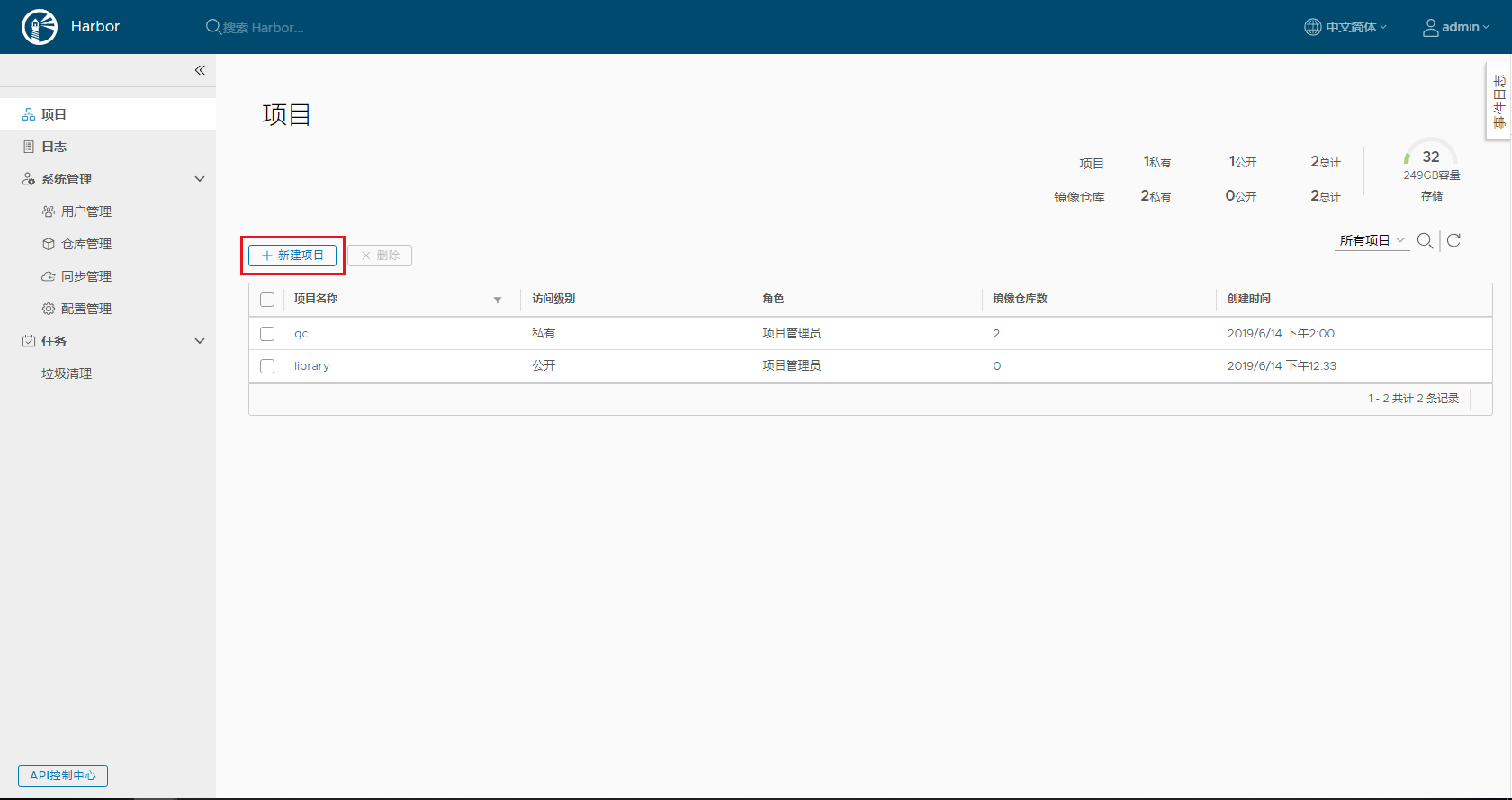
./install.sh

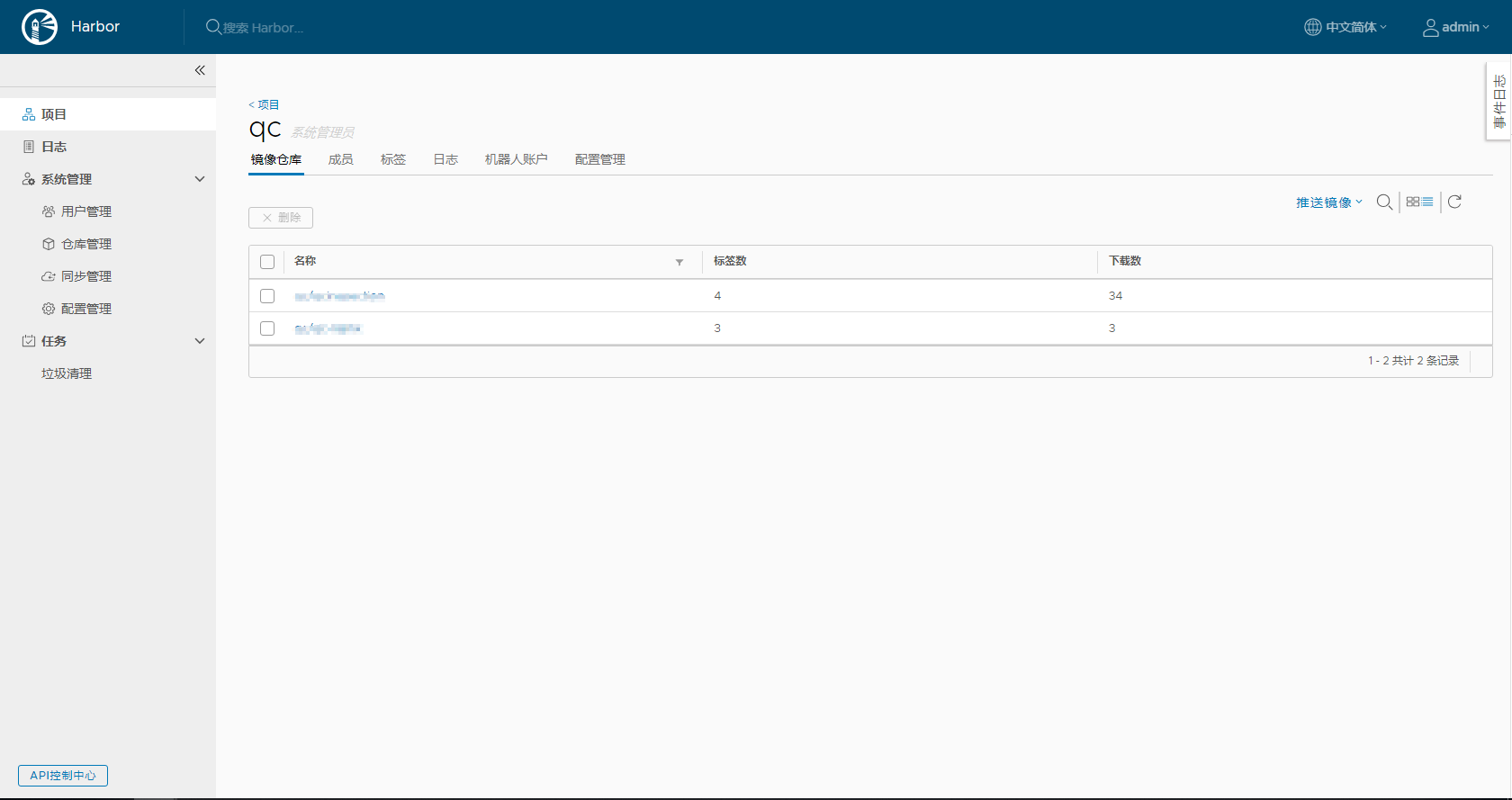
## 创建项目

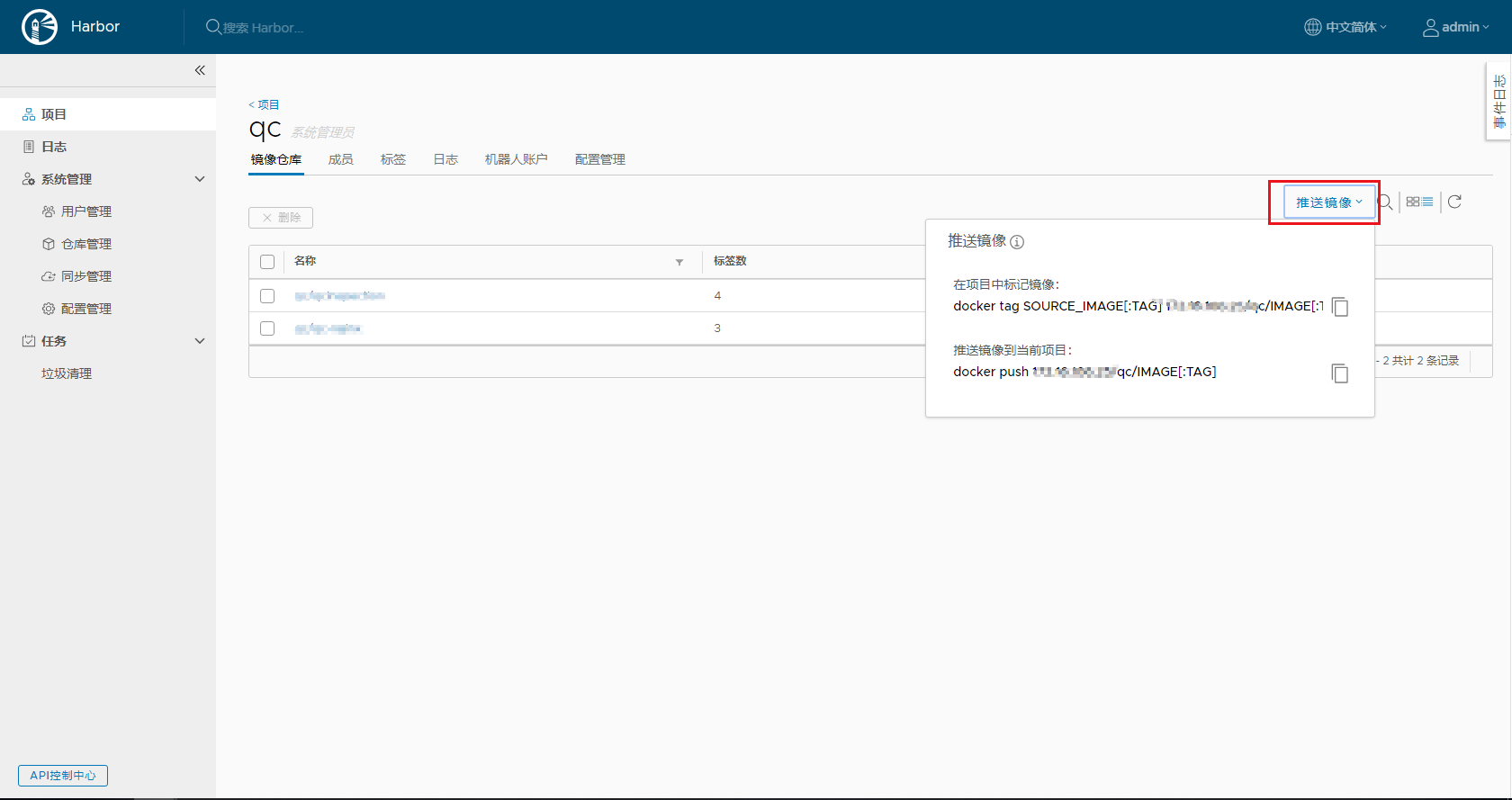
登录<http://IP>



新建项目







## 推送/拉取镜像

在需要推送或拉去镜像的机器配置

（1）vi /etc/hosts

添加内容

<HARBOR\_HOST\_IP> register.myharbor.com

（2）vi /etc/docker/daemon.json

添加内容

{

"insecure-registries": ["register.myharbor.com"]

}

（3）重启docker

systemctl restart docker

（4）修改镜像名

docker tag SOURCE\_IMAGE[:TAG] register.myharbor.com/项目名/IMAGE[:TAG]

（5）推送

docker push register.myharbor.com/项目名/IMAGE[:TAG]

# 配置jenkins

## 安装插件

Extended Choice Parameter Plug-In

## 创建构建任务

**（1）执行shell**

#构建镜像，以构建次数作为镜像Tag，注意最后的点

docker build -t register.myharbor.com/项目名/镜像名:0.0.$BUILD\_NUMBER.dev -f src/main/docker/Dockerfile --no-cache .

#推送镜像

docker push register.ptl-harbor.com/qc/qcinspection:0.0.$BUILD\_NUMBER.dev

#记录构建，向文件末尾追加构建数，

sed -i "s/$/&$BUILD\_NUMBER,/g" /目录/build-history.txt

#记录构建，向文件末尾追加构建时间

sed -i "s/$/&`date +%Y-%m-%d-%H:%M:%S`,/g" /目录/build-history-des.txt

#删除镜像

docker rmi -f register.myharbor.com/qc/qcinspection:0.0.$BUILD\_NUMBER.dev

**（2）Send files or execute commands over SSH**

#设置环境变量

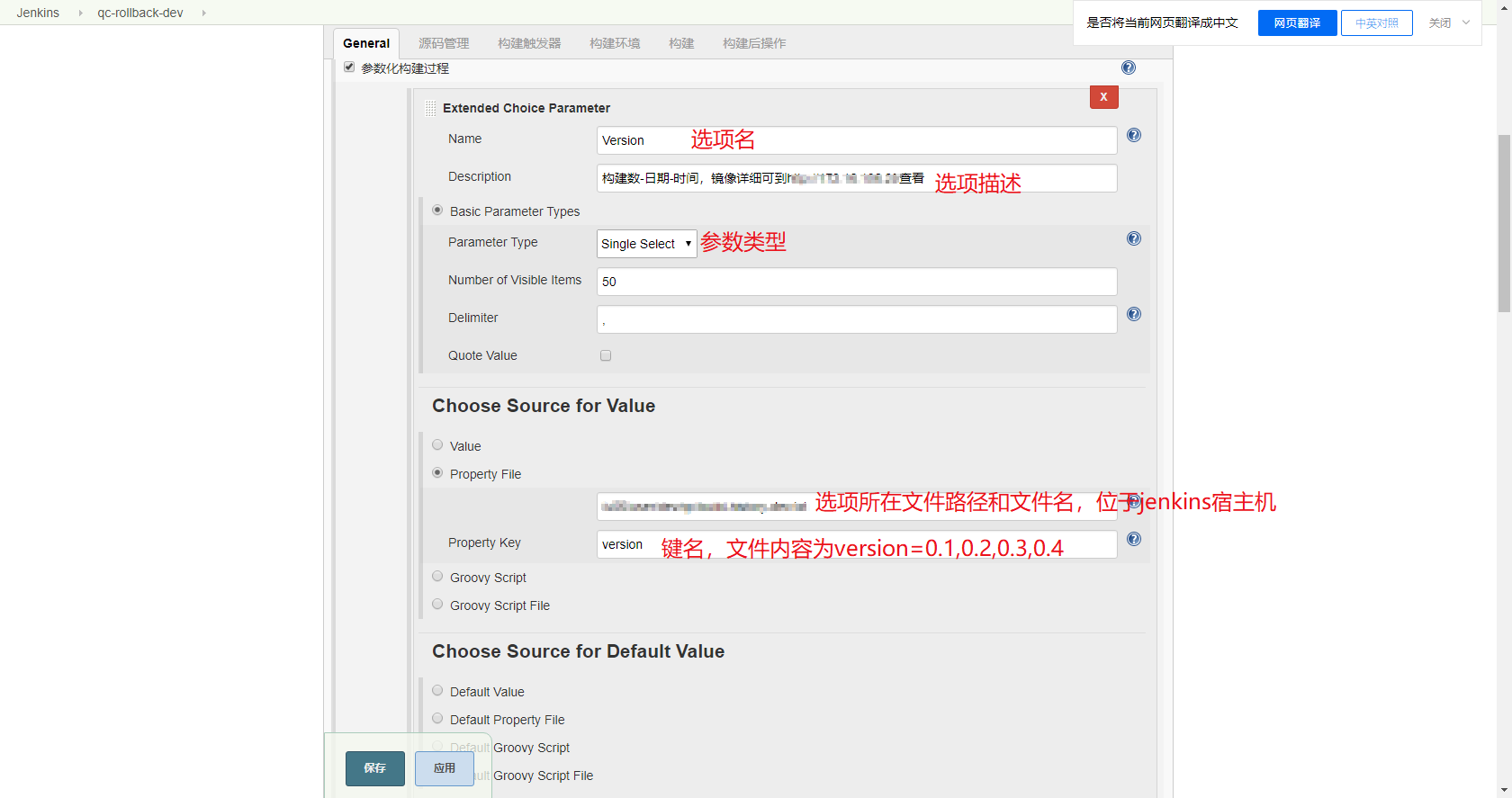
export BUILD\_NUMBER=${BUILD\_NUMBER}

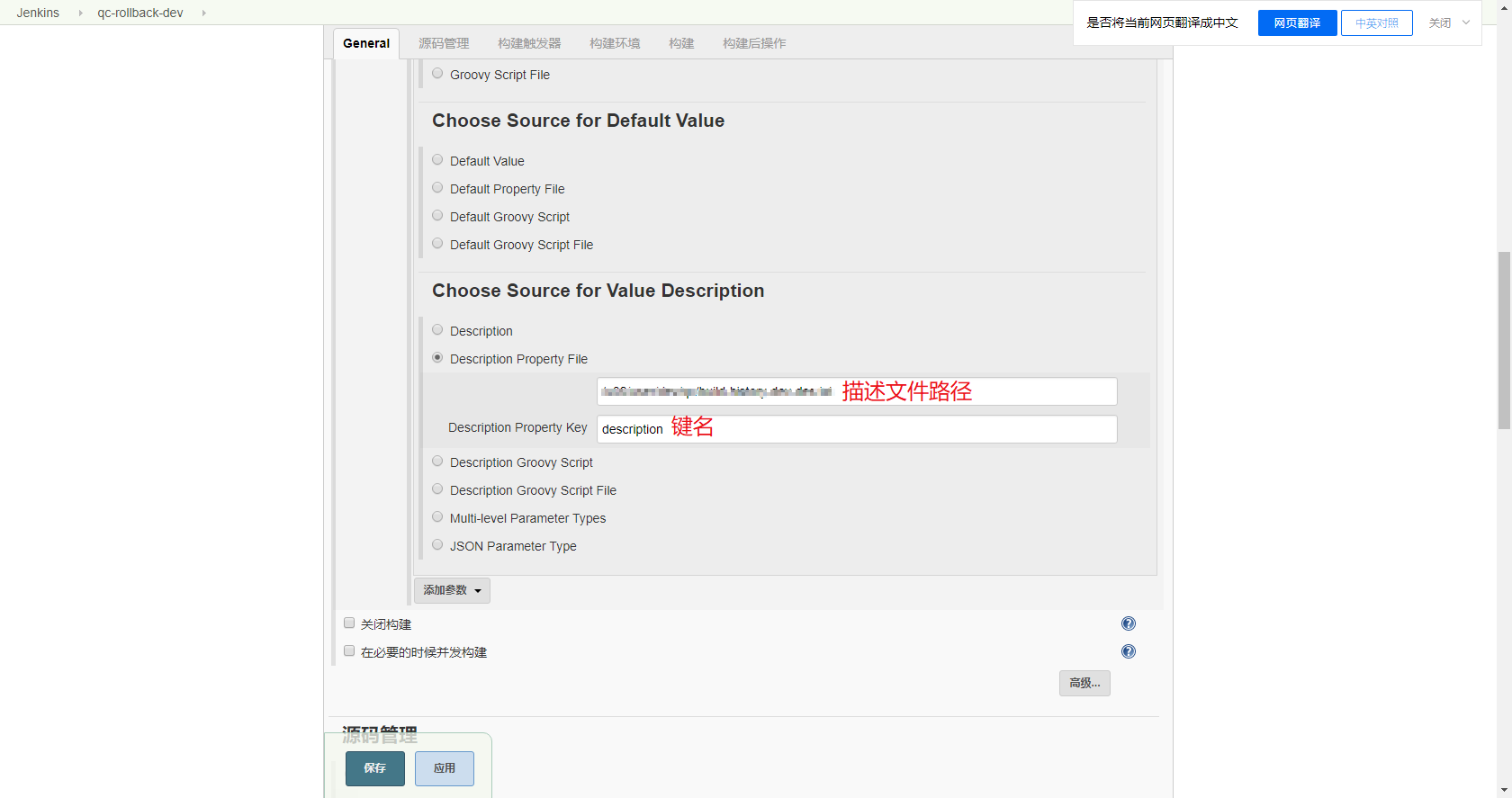
#启动stack

docker stack deploy -c /目录/文件名.yml --with-registry-auth mystack

## 创建回滚任务

1. **参数化构建**





1. **Send files or execute commands over SSH**

#设置环境变量

export BUILD\_NUMBER=${Version}

#启动/更新stack

docker stack deploy -c /路径/文件名.yml --with-registry-auth mystack

# 问题和解决方法

## docker 登录问题

**问题：**

Error response from daemon: Get https://IP/v1/users/: dial tcp IP:443: getsockopt: connection refused

或

Error response from daemon: Get https://IP/v2/:http:server gave HTTP response to HTTPS client

**原因：**

使用http协议，而不是https

**解决方法：**

修改/etc/docker/daemon.json

添加如下内容：

{

"insecure-registries": ["register.ptl-harbor.com"]

}

# 待改进

**（1）**Extended Choice Parameter Plug-In插件可以使用javascript、groovy脚本，通过脚本使用harbor的API可以直接获取镜像情况，而不需要在构建时，将构建情况存入文件。