# 一、ansible安装配置

ansible是新出现的自动化运维工具，基于Python开发，集合了众多运维工具（puppet、cfengine、chef、func、fabric）的优点，实现了批量系统配置、批量程序部署、批量运行命令等功能。

ansible是基于模块工作的，本身没有批量部署的能力。真正具有批量部署的是ansible所运行的模块，ansible只是提供一种框架。主要包括：

(1)、连接插件connection plugins：负责和被监控端实现通信；

(2)、host inventory：指定操作的主机，是一个配置文件里面定义监控的主机；

(3)、各种模块核心模块、command模块、自定义模块；

(4)、借助于插件完成记录日志邮件等功能；

(5)、playbook：剧本执行多个任务时，非必需可以让节点一次性运行多个任务。

## 1.安装ansible

yum install epel-release

yum install ansible

配置文件

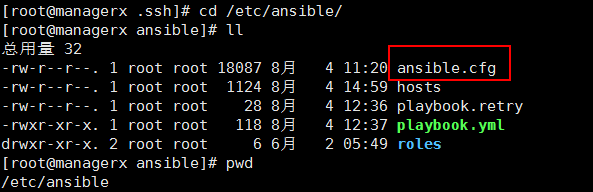
*/etc/ansible*

*hosts 配置服务器组*



## 2.配置ansible

配置文件：/etc/ansible/ansible.cfg



1> 禁用每次执行ansbile命令检查ssh key host

host\_key\_checking = False

2> 开启日志记录

log\_path = /var/log/ansible.log

3> ansible连接加速配置

[accelerate]

#accelerate\_port = 5099

accelerate\_port = 10000

#accelerate\_timeout = 30

#accelerate\_connect\_timeout = 5.0

# If set to yes, accelerate\_multi\_key will allow multiple

# private keys to be uploaded to it, though each user must

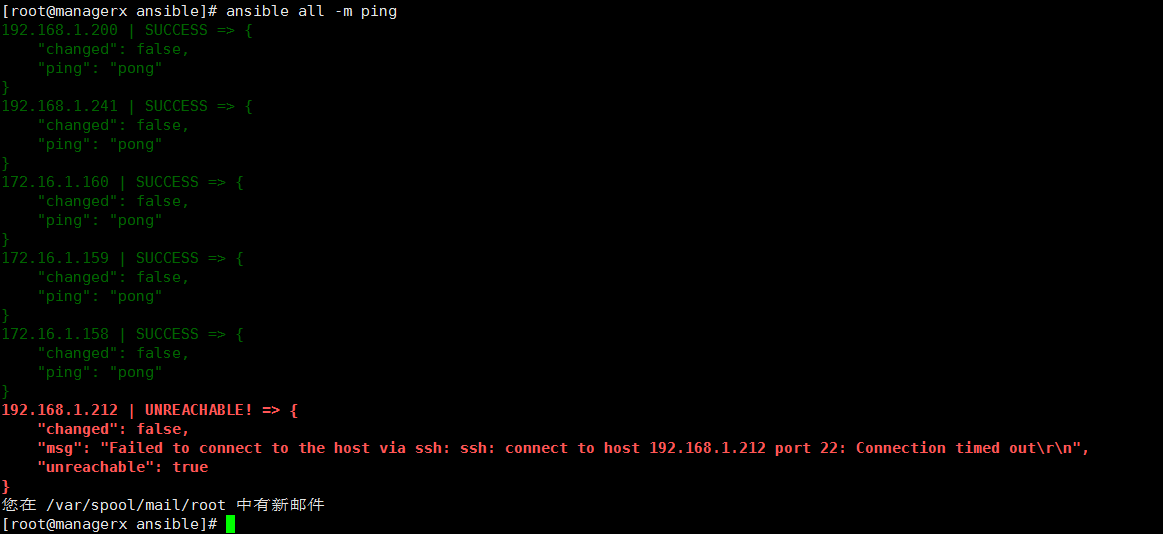
# have access to the system via SSH to add a new key. The default

# is "no".

accelerate\_multi\_key = yes

## 3.测试ansible

在管理端执行命令：ansible all -m ping



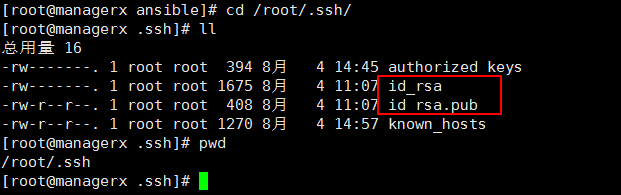
绿色表示成功的，红色表示失败（网络不通）

## 4.生成ssh密钥

管理端服务器执行：ssh-keygen -t rsa

一路回车，生产密钥文件

文件路径：~/.ssh/



复制公共密钥文件内容到被管理端

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@ip1

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@ip2

ssh-copy-id -i ~/.ssh/id\_rsa.pub root@ip3

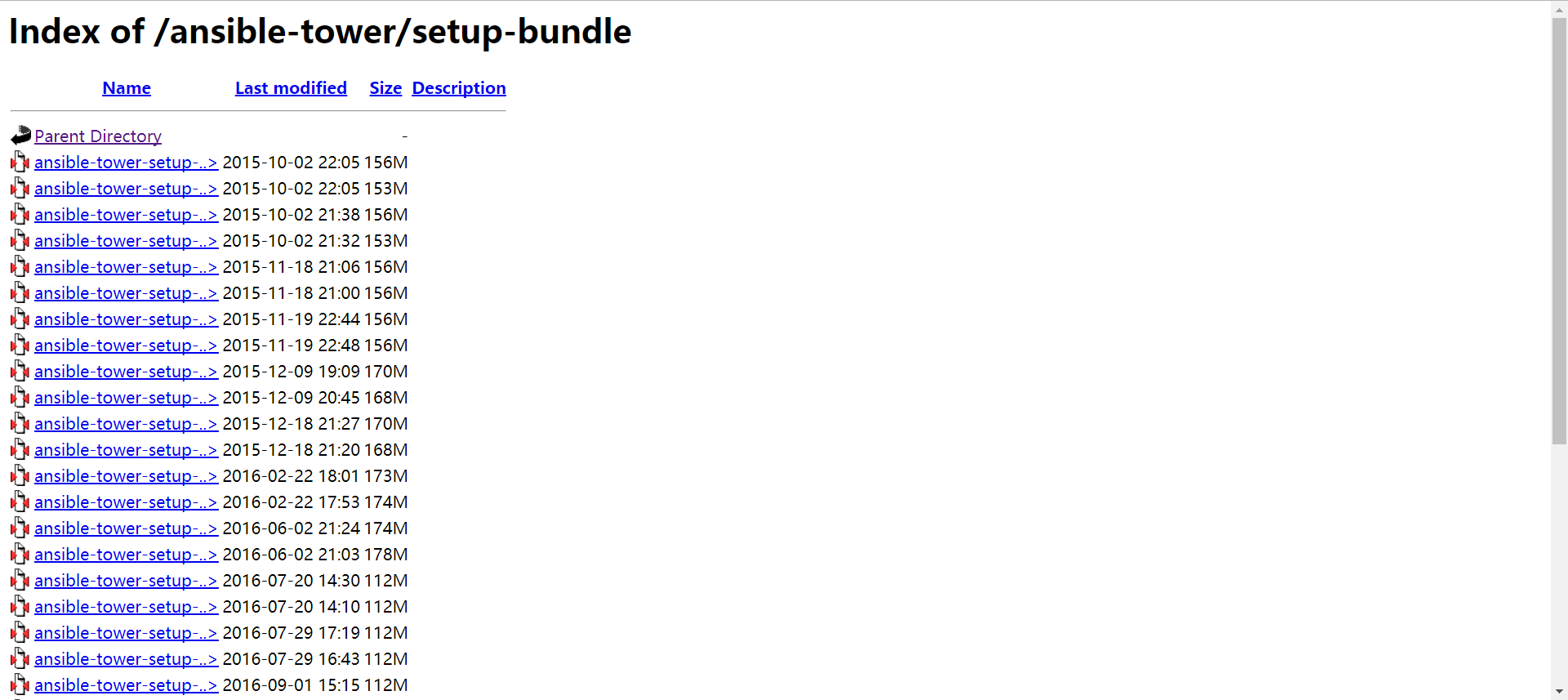
*注：ssh-copy-id命令会自动将id\_rsa.pub文件的内容****追加****到远程主机root用户下.ssh/authorized\_keys文件中,如果没有这个文件手动新建一个*

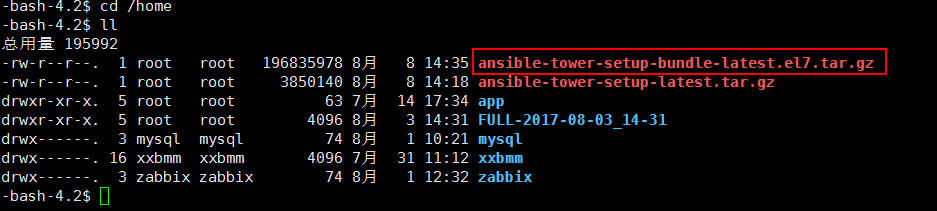
# 二、ansible tower图形化工具安装配置

ansible tower是基于ansible的图形化的操作界面，相对于ansible命令行模式来说更新简洁，操作方便。

## 1.下载ansible tower安装包

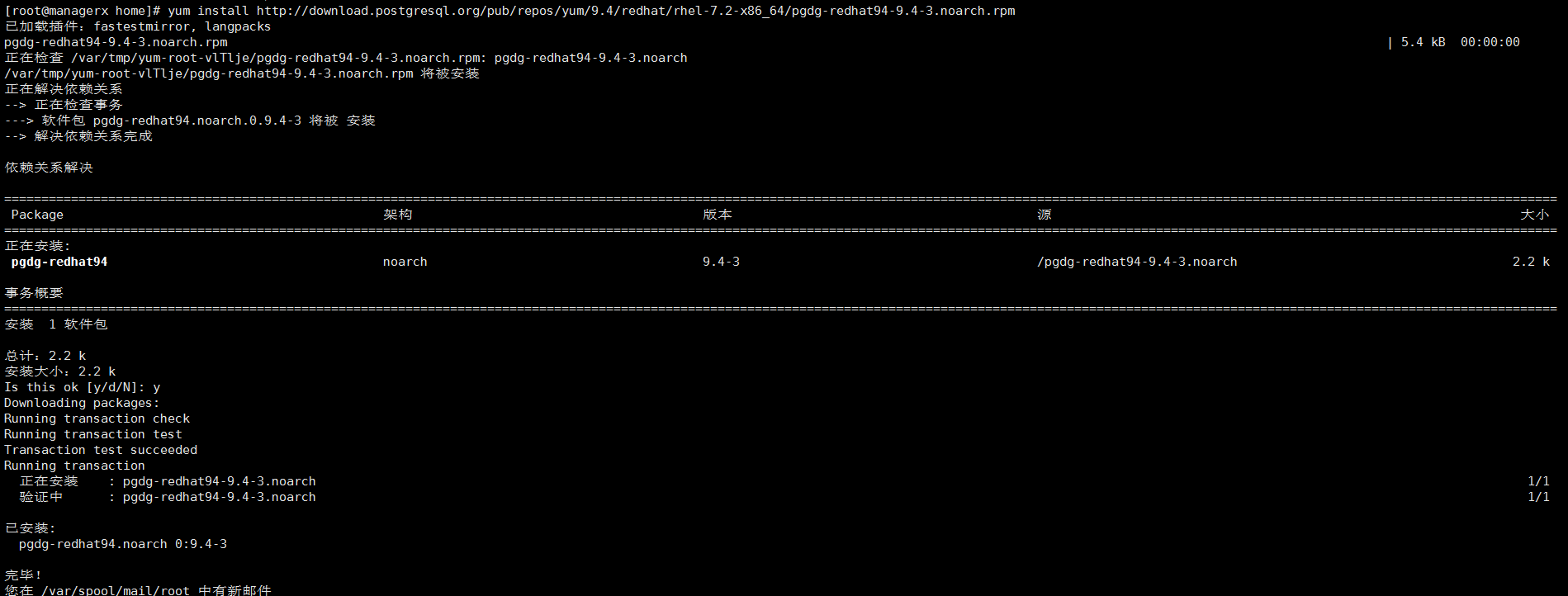
<http://releases.ansible.com/ansible-tower/setup-bundle/>



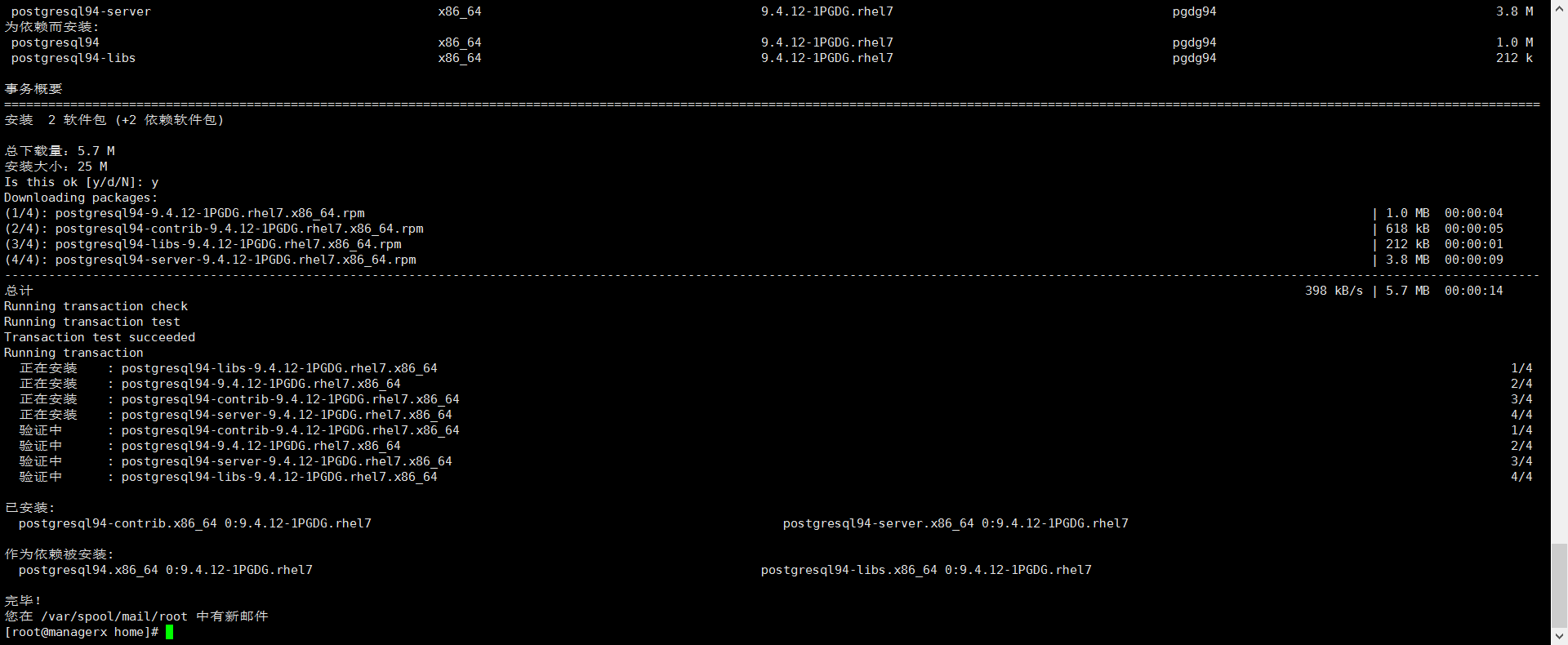


## 2.安装postgresql数据库

yum install http://download.postgresql.org/pub/repos/yum/9.4/redhat/rhel-7.2-x86\_64/pgdg-redhat94-9.4-3.noarch.rpm



yum install postgresql94-server postgresql94-contrib



设置开机启动

 systemctl enable postgresql-9.4.service



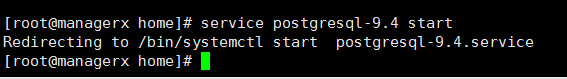
初始化数据库

 /usr/pgsql-9.4/bin/postgresql94-setup initdb



启动postgresql

service postgresql-9.4 start



创建用户

su - postgres

-bash-4.2$ psql

    postgres=# CREATE ROLE awx CREATEDB PASSWORD 'admin' LOGIN;

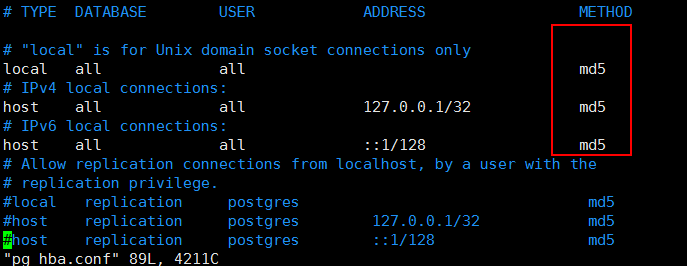
    postgres=# \q

-bash-4.2$ exit



sed -i 's#peer#md5#g' /var/lib/pgsql/9.4/data/pg\_hba.conf

sed -i 's#ident#md5#g' /var/lib/pgsql/9.4/data/pg\_hba.conf



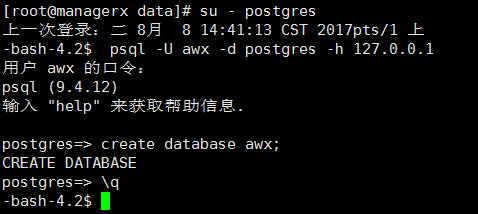
service postgresql-9.4 restart

测试awx用户连接，输入密码连接，并创建数据库

-bash-4.2$ psql -U awx -d postgres -h 127.0.0.1

postgres=# create database awx;

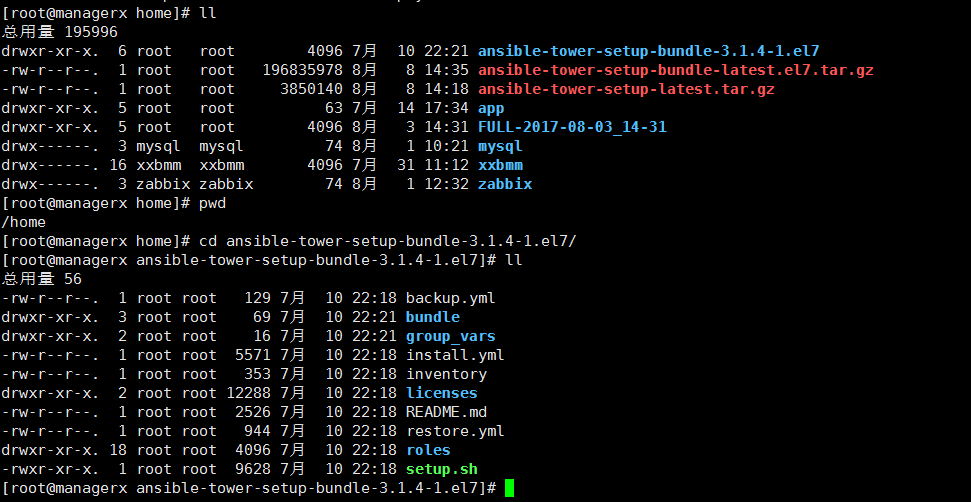
postgres=# \q



## 3.安装ansible tower

解压tar zxvf ansible-tower-setup-bundle-latest.el7.tar.gz

cd ansible-tower-setup-bundle-latest

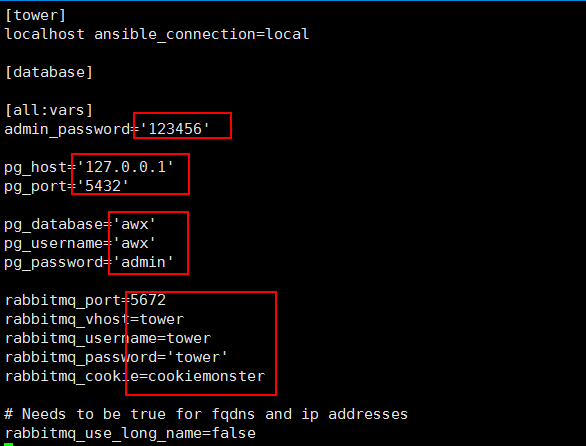


设置主机信息vim inventory

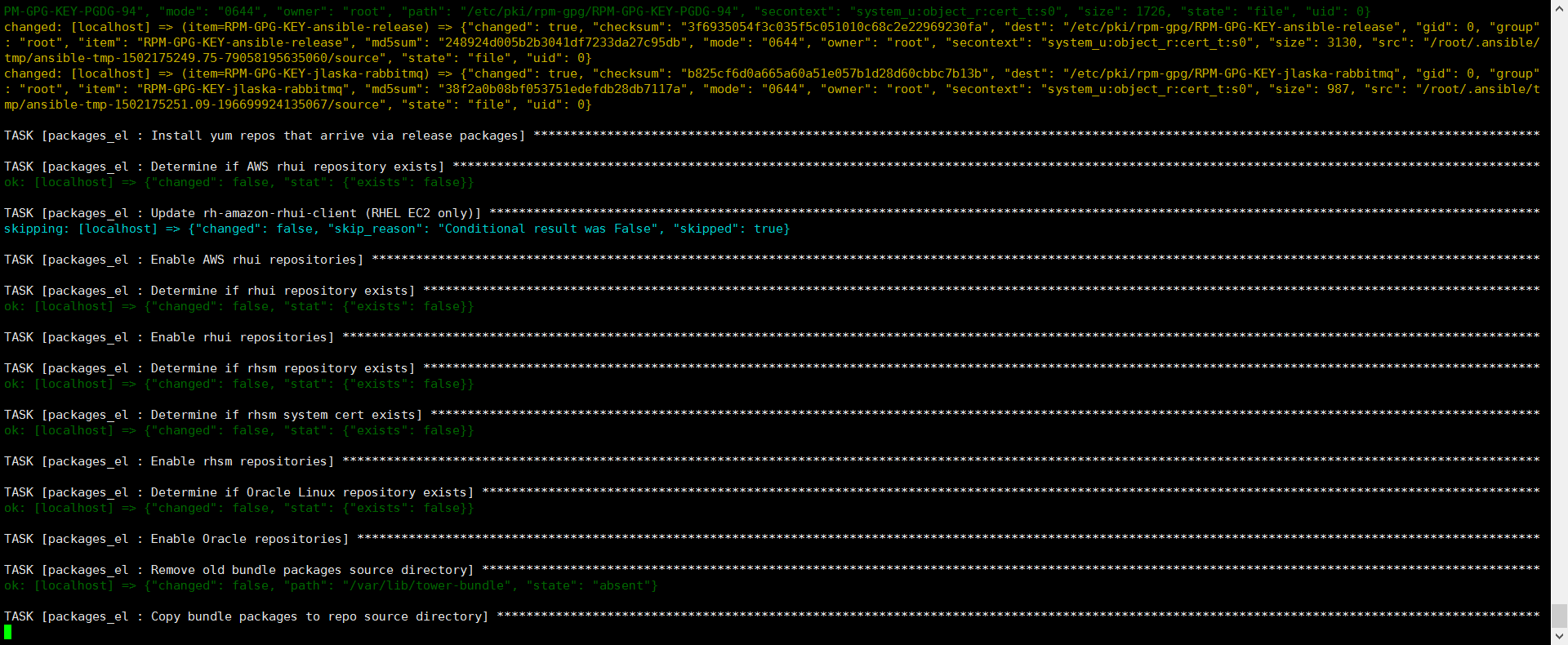
sed -i "s#admin\_password=''#admin\_password='123456'#g" inventory

sed -i "s#pg\_host=''#pg\_host='127.0.0.1'#g" inventory

sed -i "s#pg\_port=''#pg\_port='5432'#g" inventory

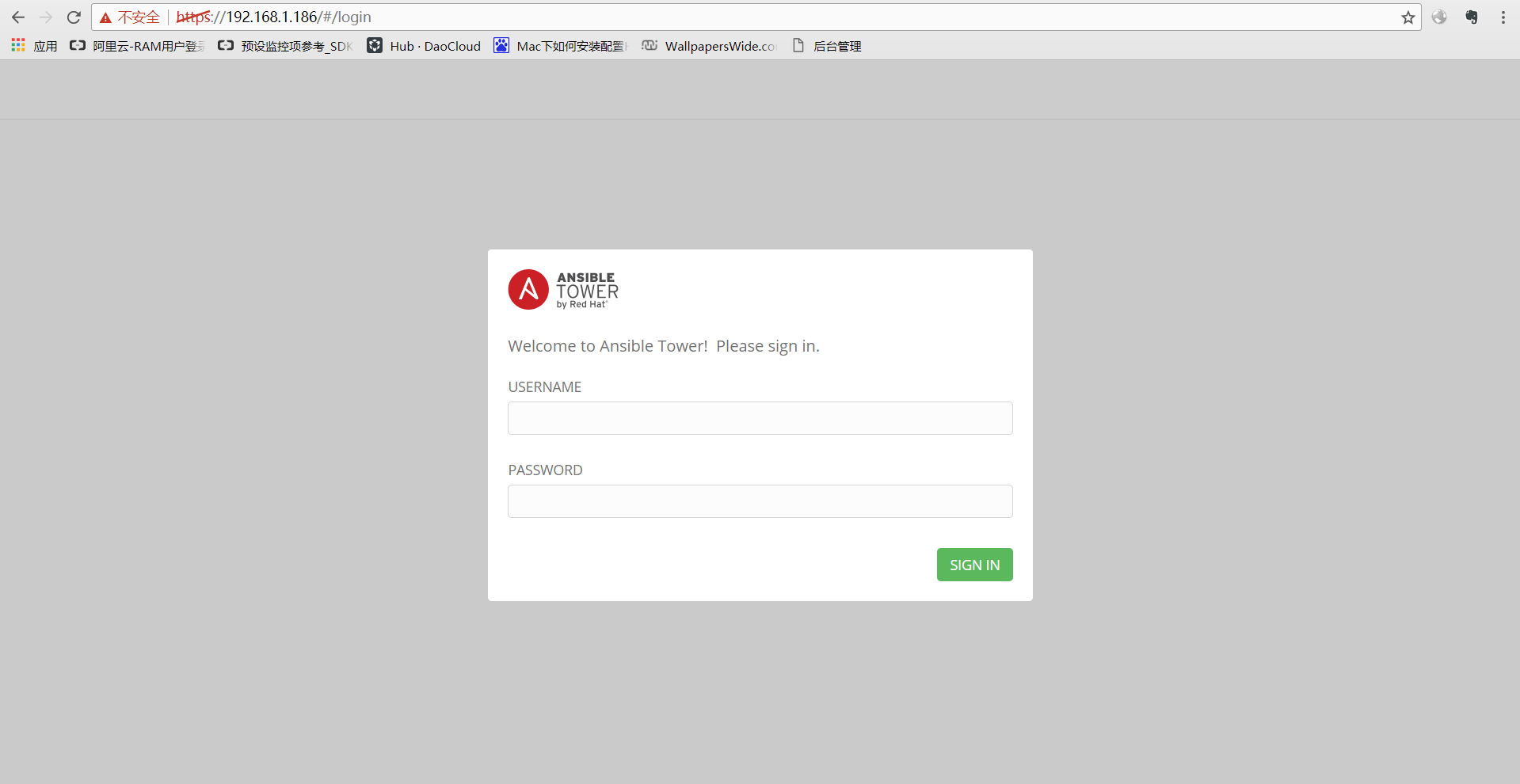


执行./setup.sh进行安装tower



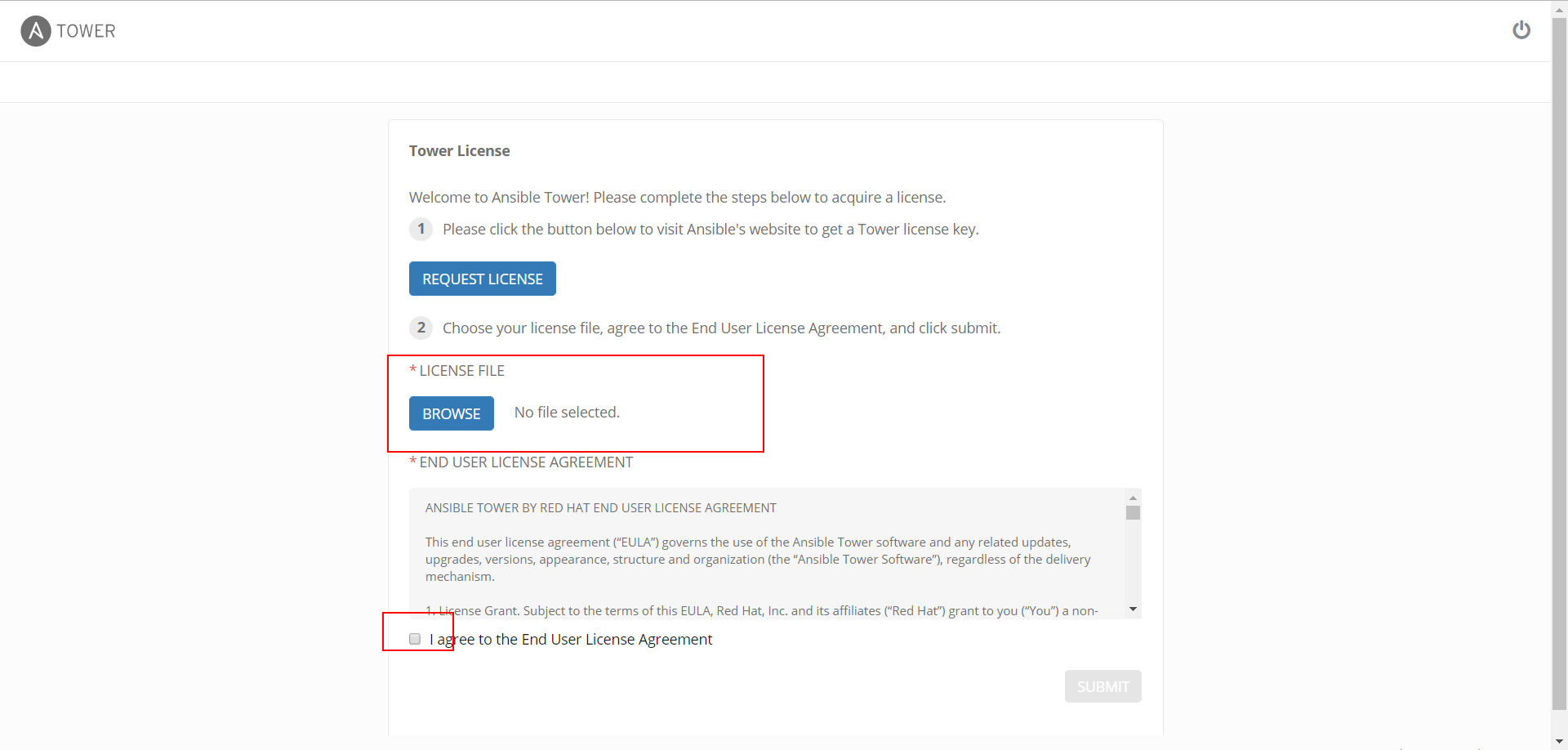
安装完成

## 4.登录ansible tower

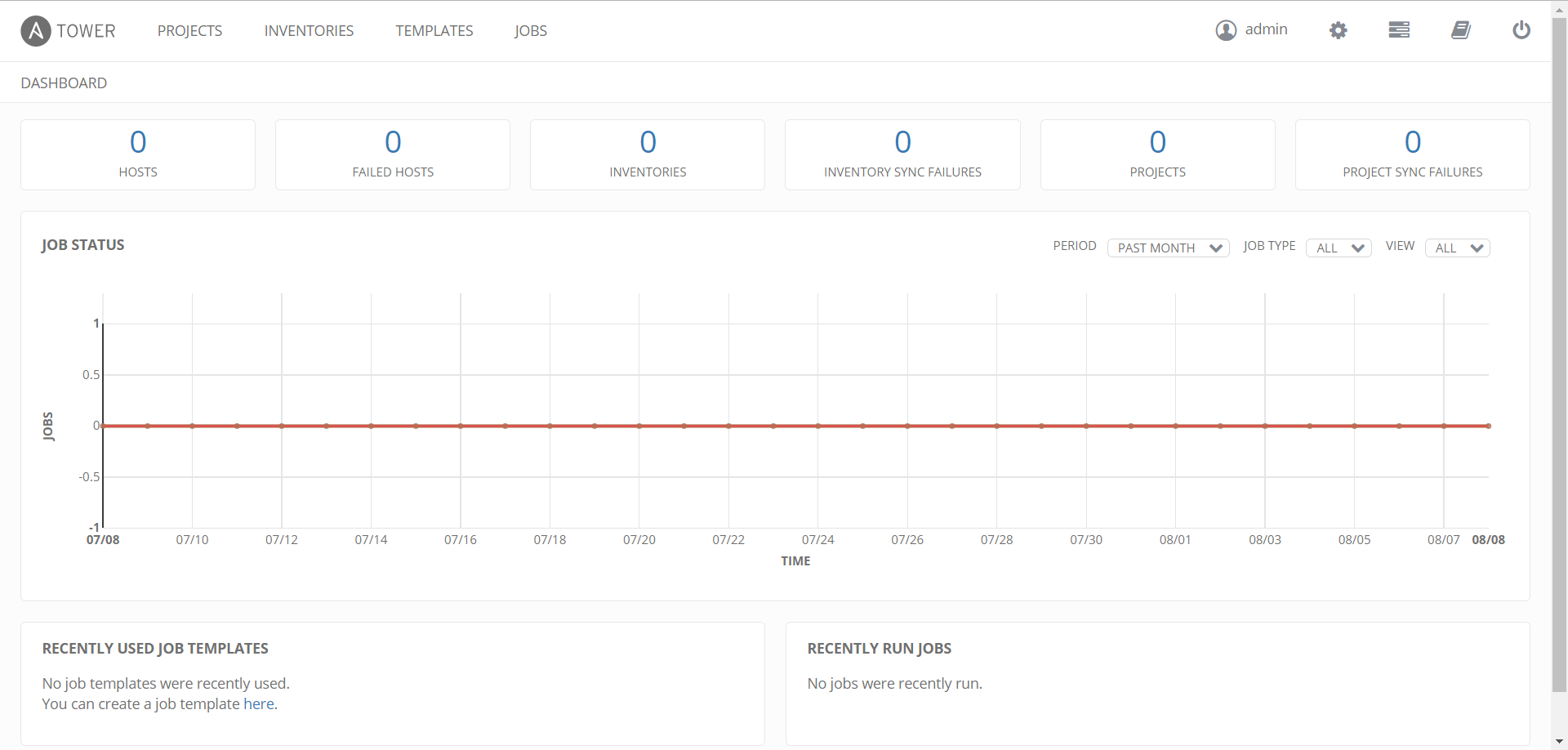


用户密码：admin 123456

登录成功后导入授权文件



程序主界面

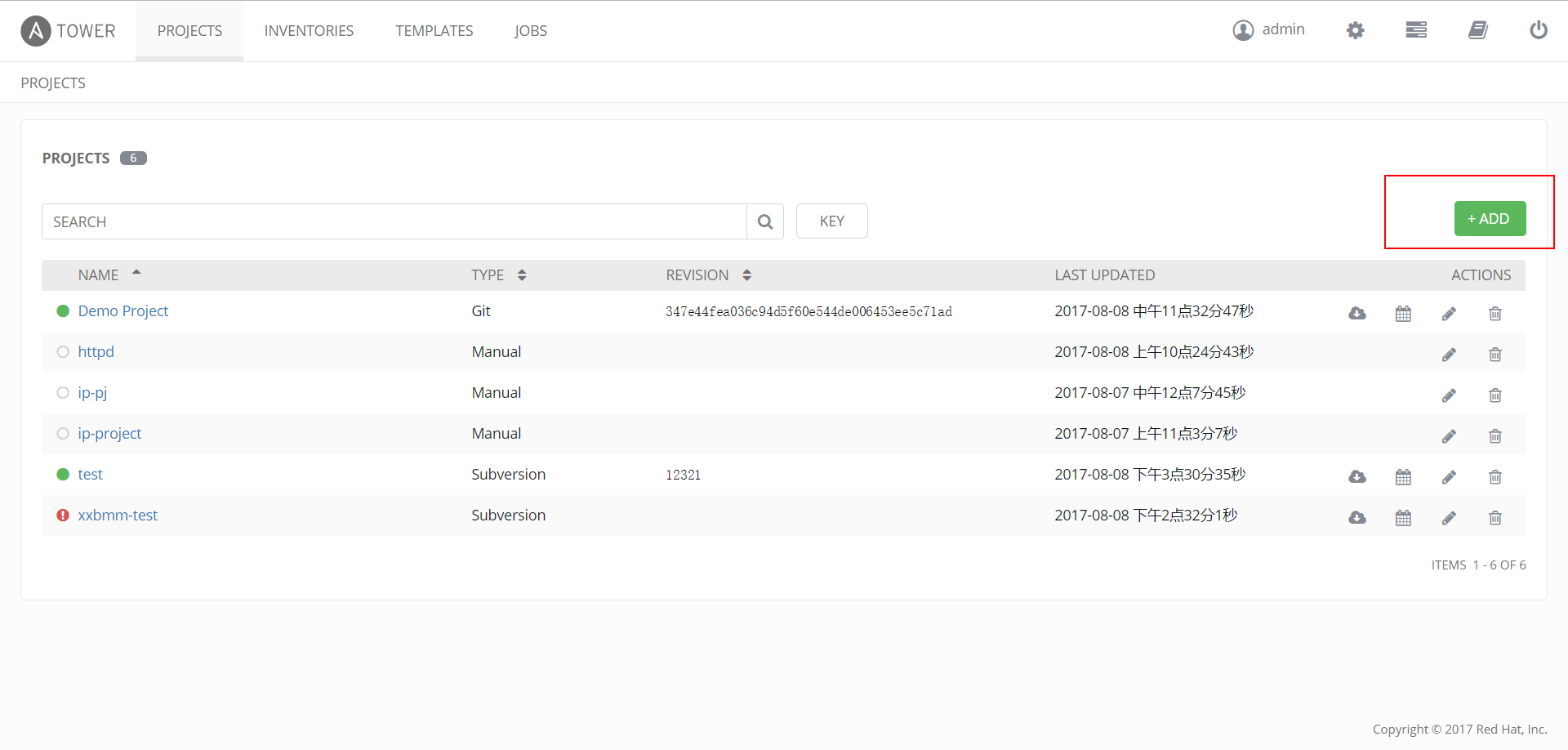


# 三、ansible tower实际操作

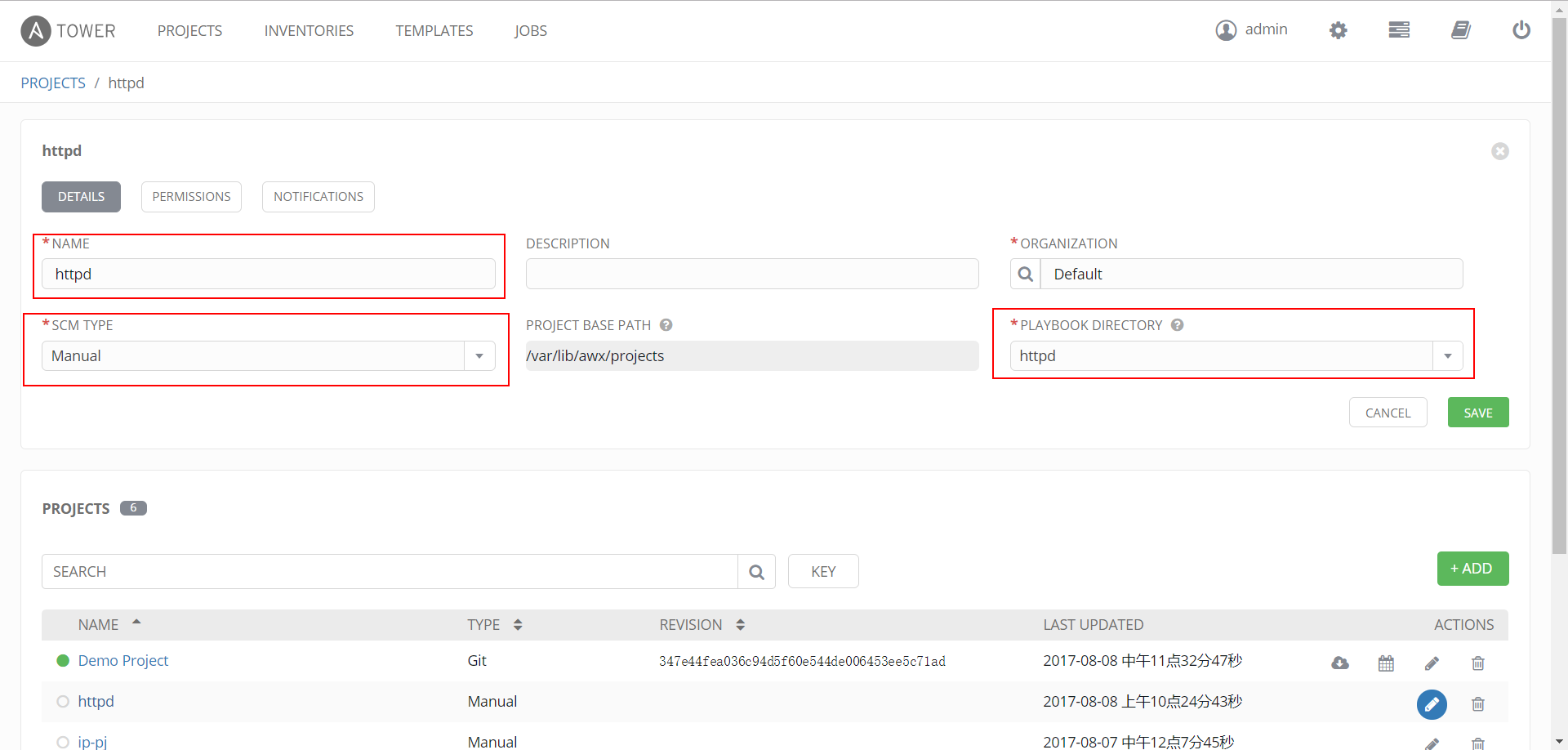
## 1.新建PROJECTS

1）、新建手工运行模式的projects

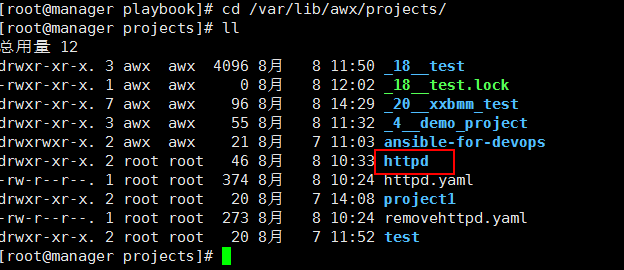
点击页面ADD



编辑填写好带“\*”的内容



注意：PLAYBOOK DIRECTORY需要在目录/var/lib/awx/projects/下新建才能在下来表看到



点击保存完成新建

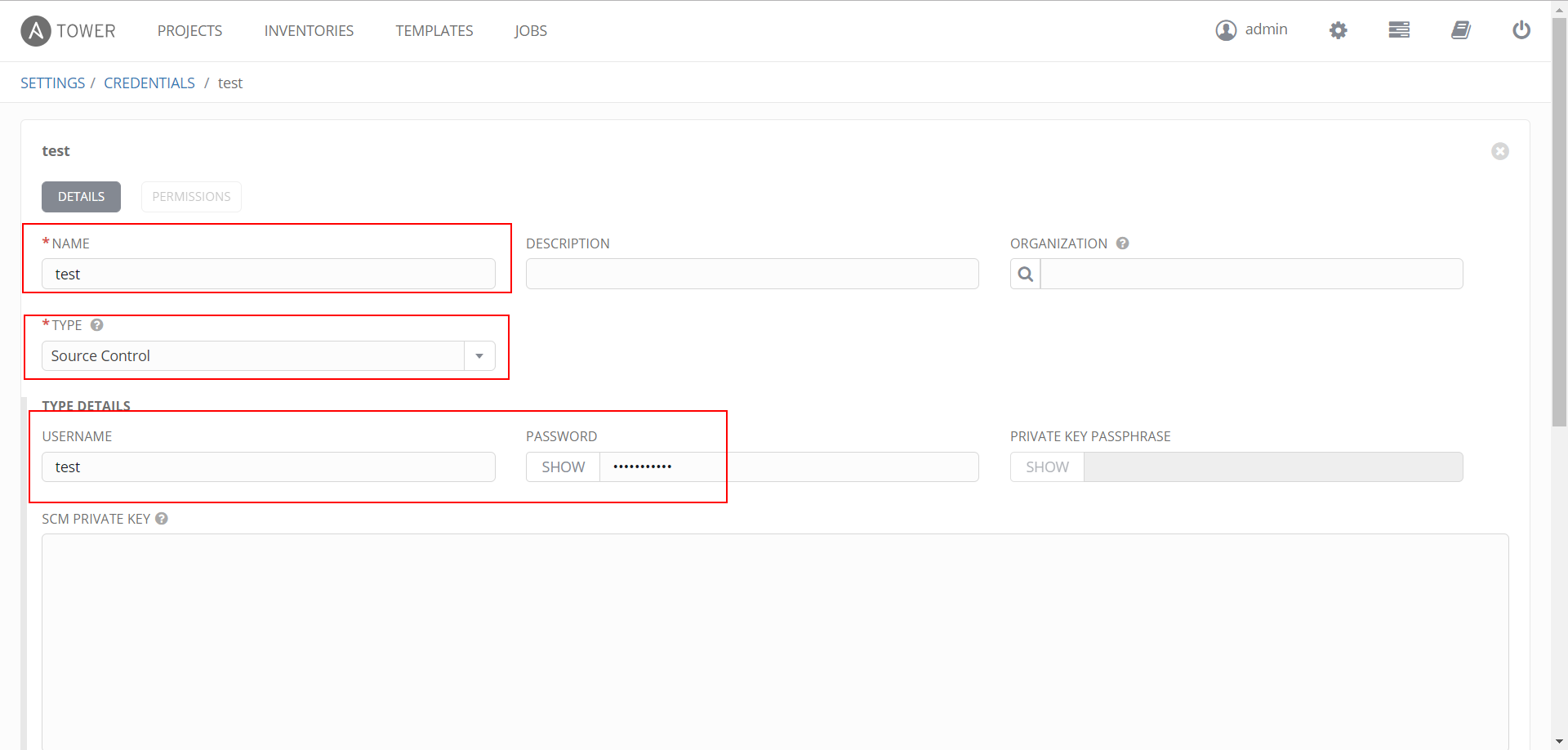
2）、新建自动运行projects

和上步手工运行不同的是，该工程所有配置文件都放在SVN服务端，如果svn上面文件有变化，会自动下载到本地目录

（1）.建立svn用户

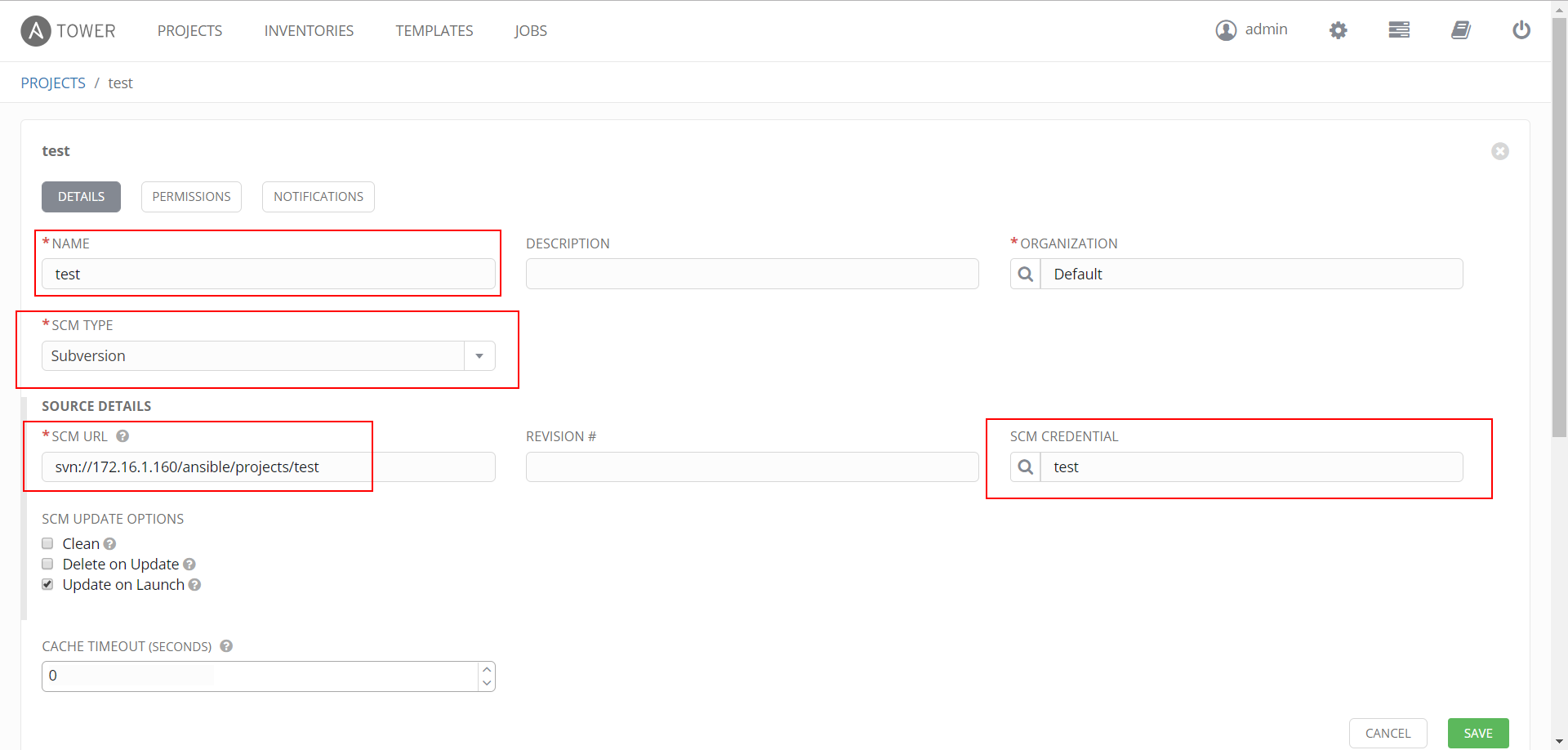
[SETTINGS](https://192.168.1.200/#/setup)  -CREDENTIALS

点击add按照如下配置填写



保存

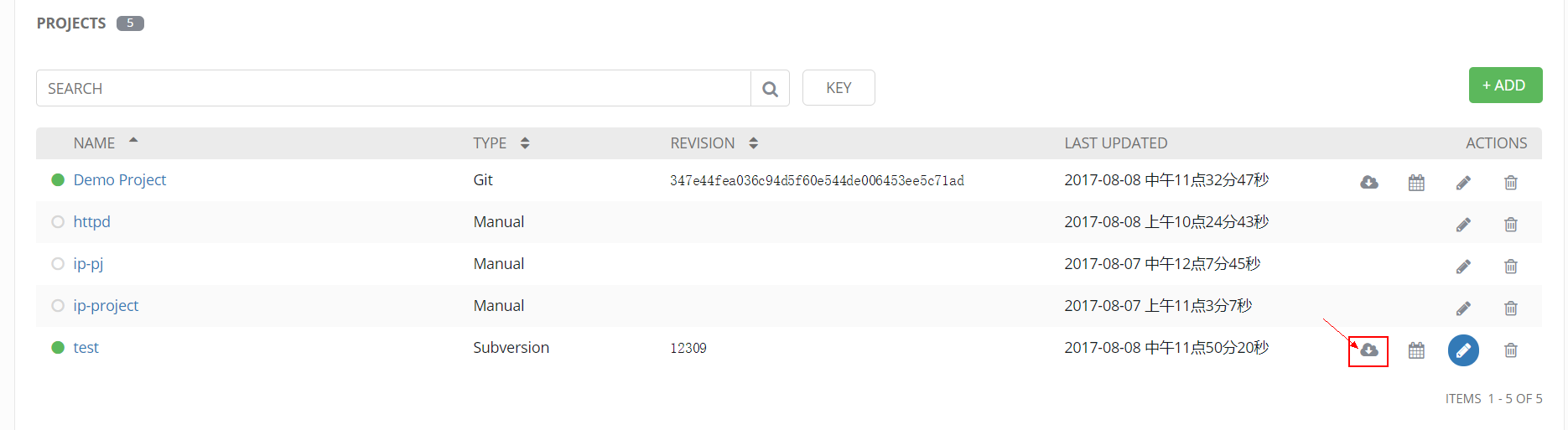
（2）.建立project



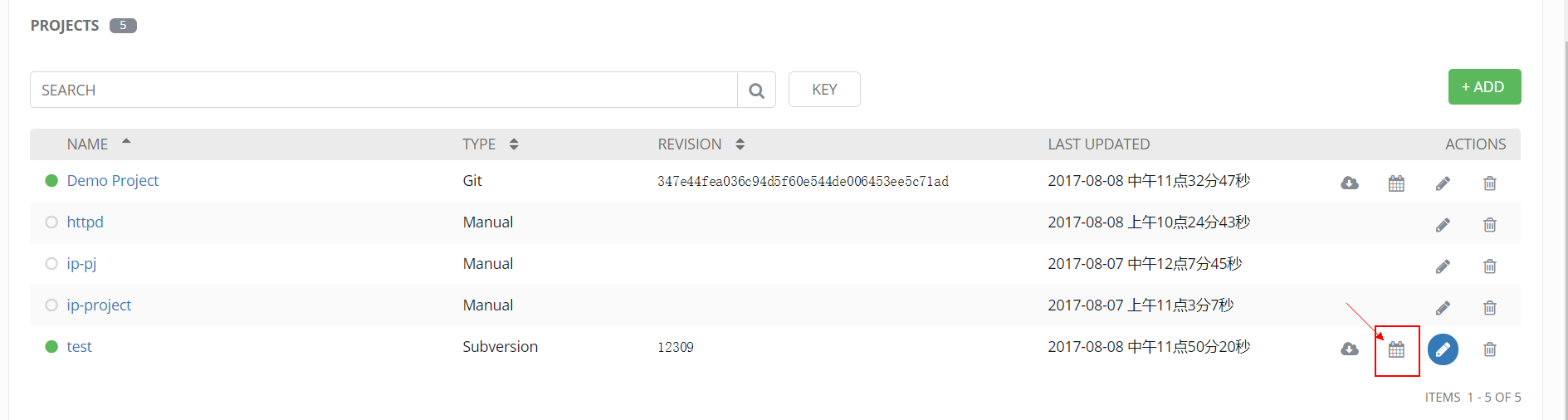
填入相关信息

点击save

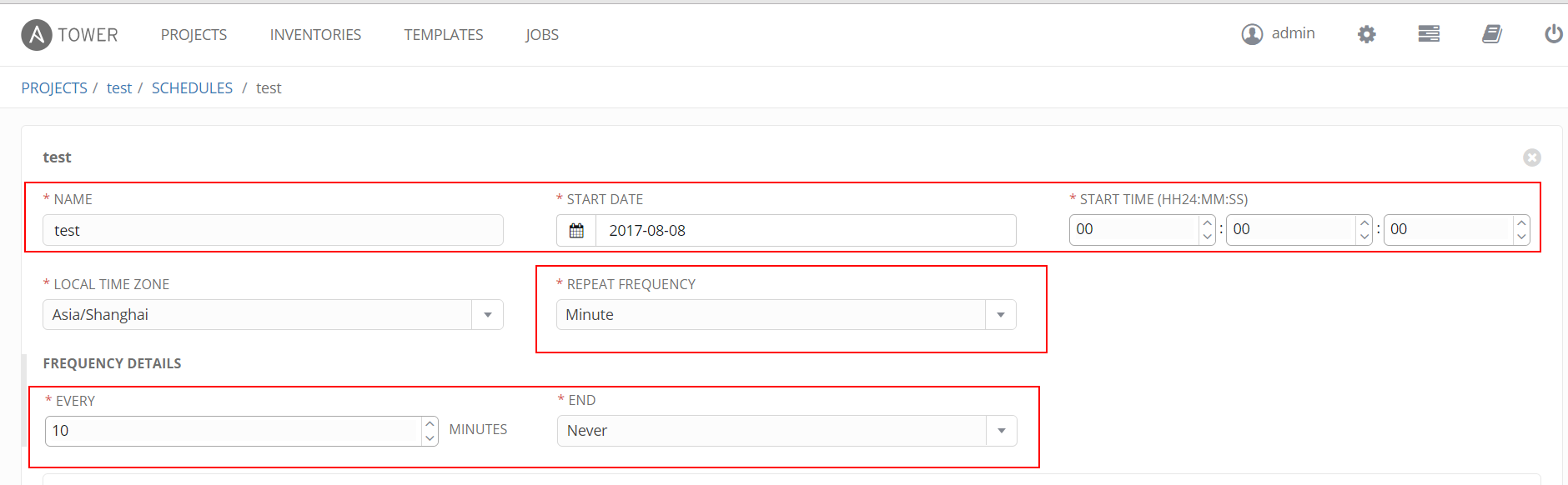
执行project



（3）.在project中建立任务计划



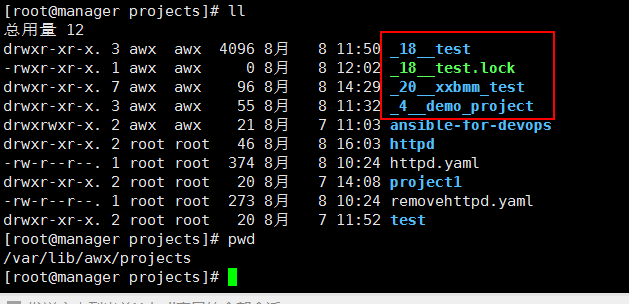
点击add



根据实际情况配置

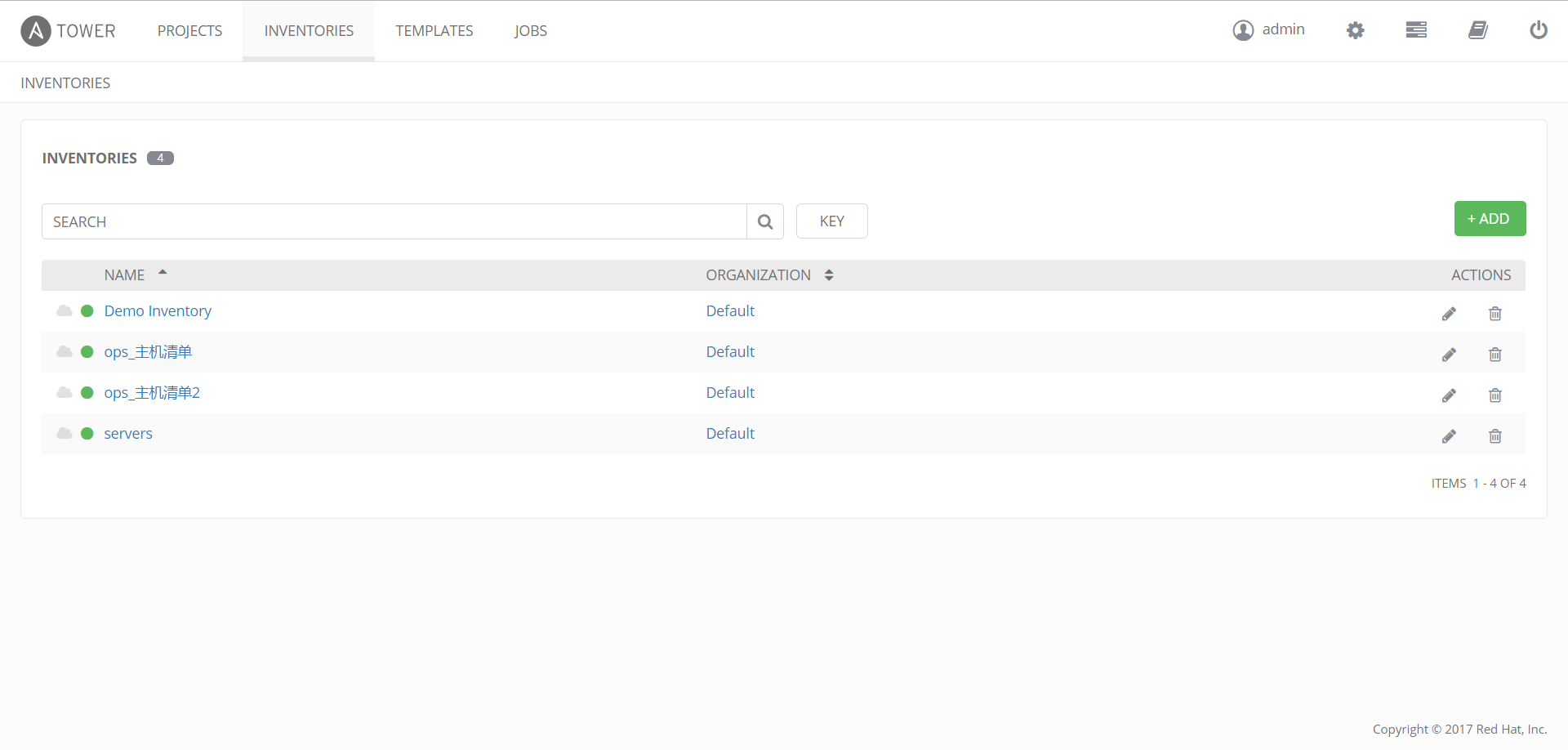
如上表示从当天起，每十分钟扫描一次svn，永久运行，如果有更新会自动下载到本地

本地目录为自动生成：



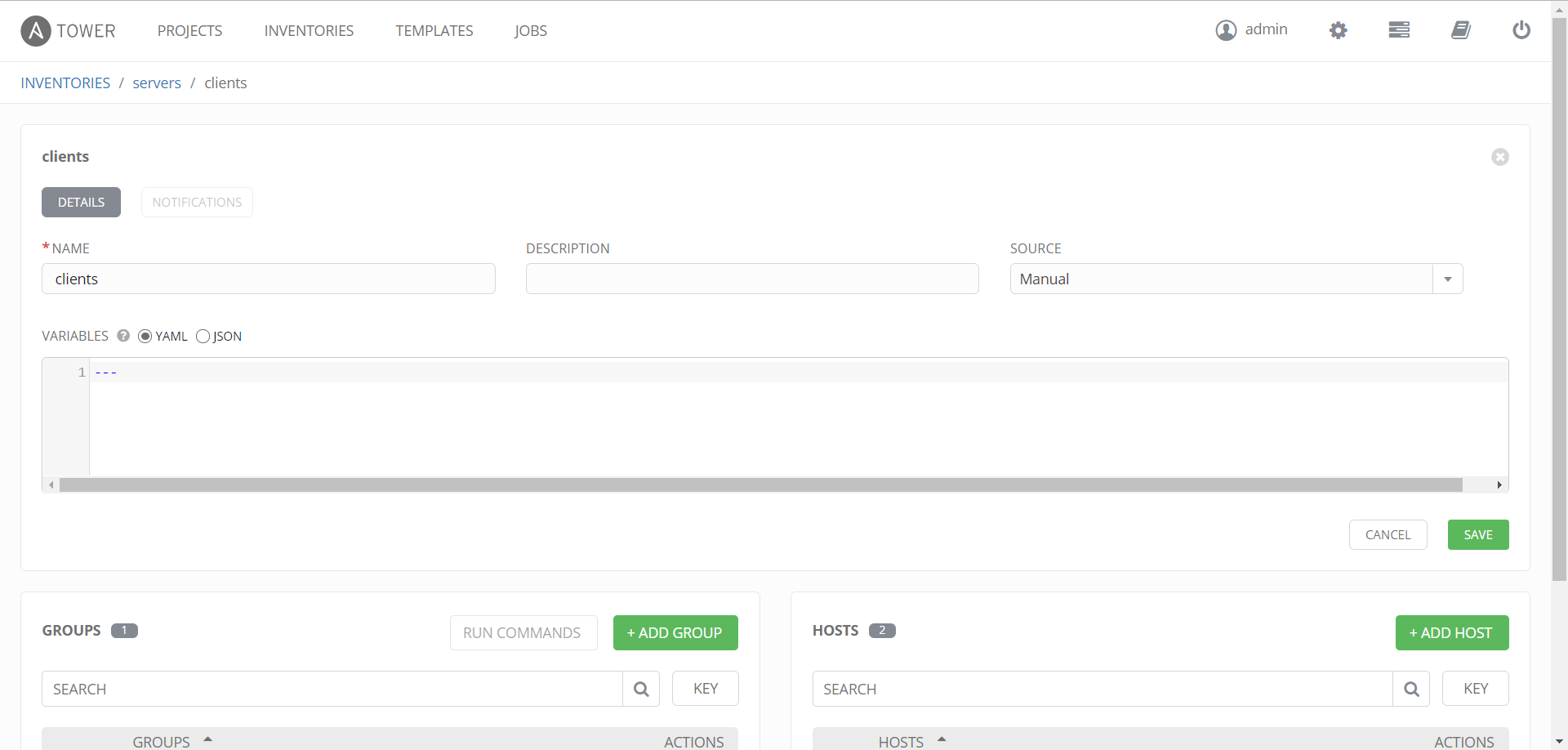
## 2.新建INVENTORIES

INVENTORIES保存主机及组等信息，类似ansible的hosts文件

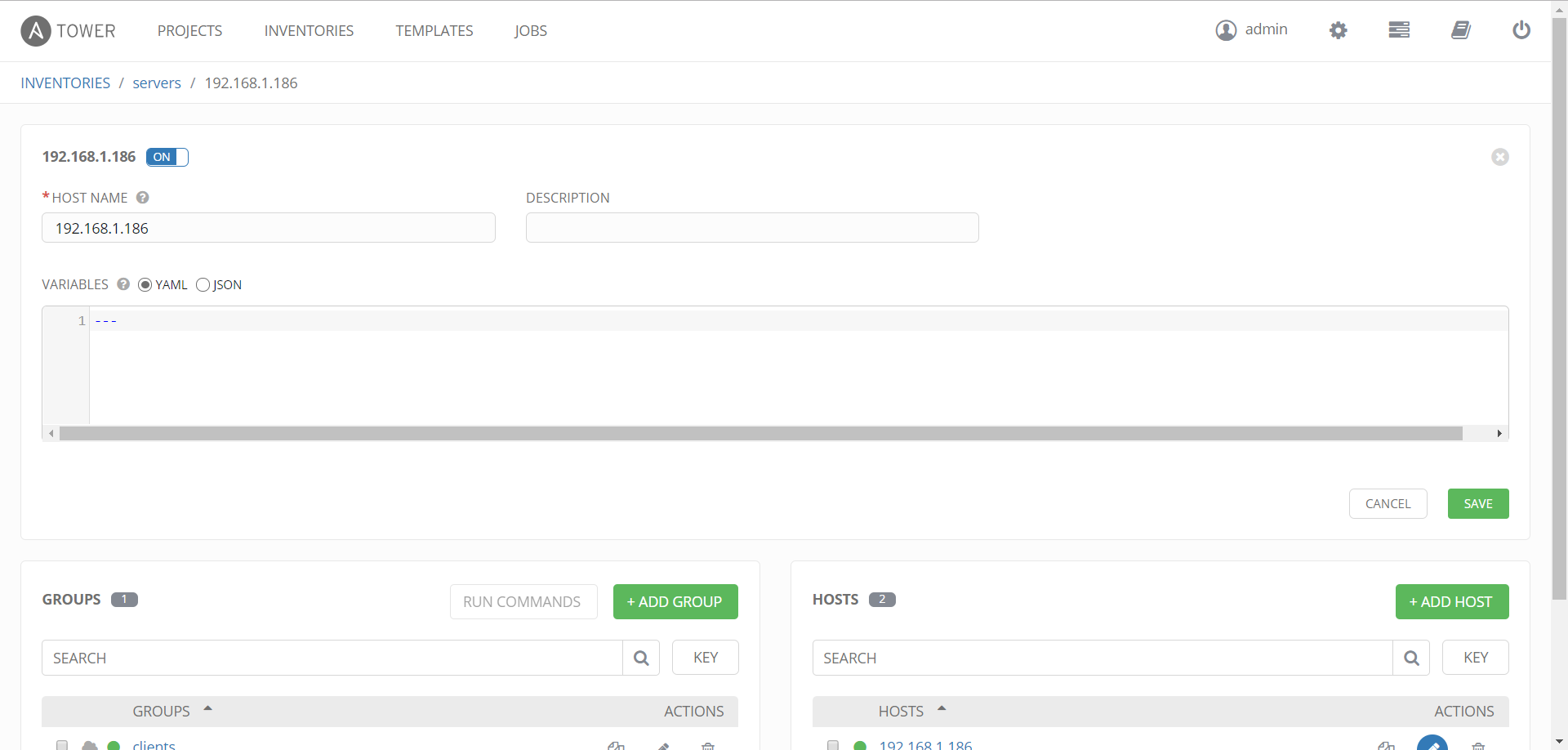


点击ADD新增

1）、新建group

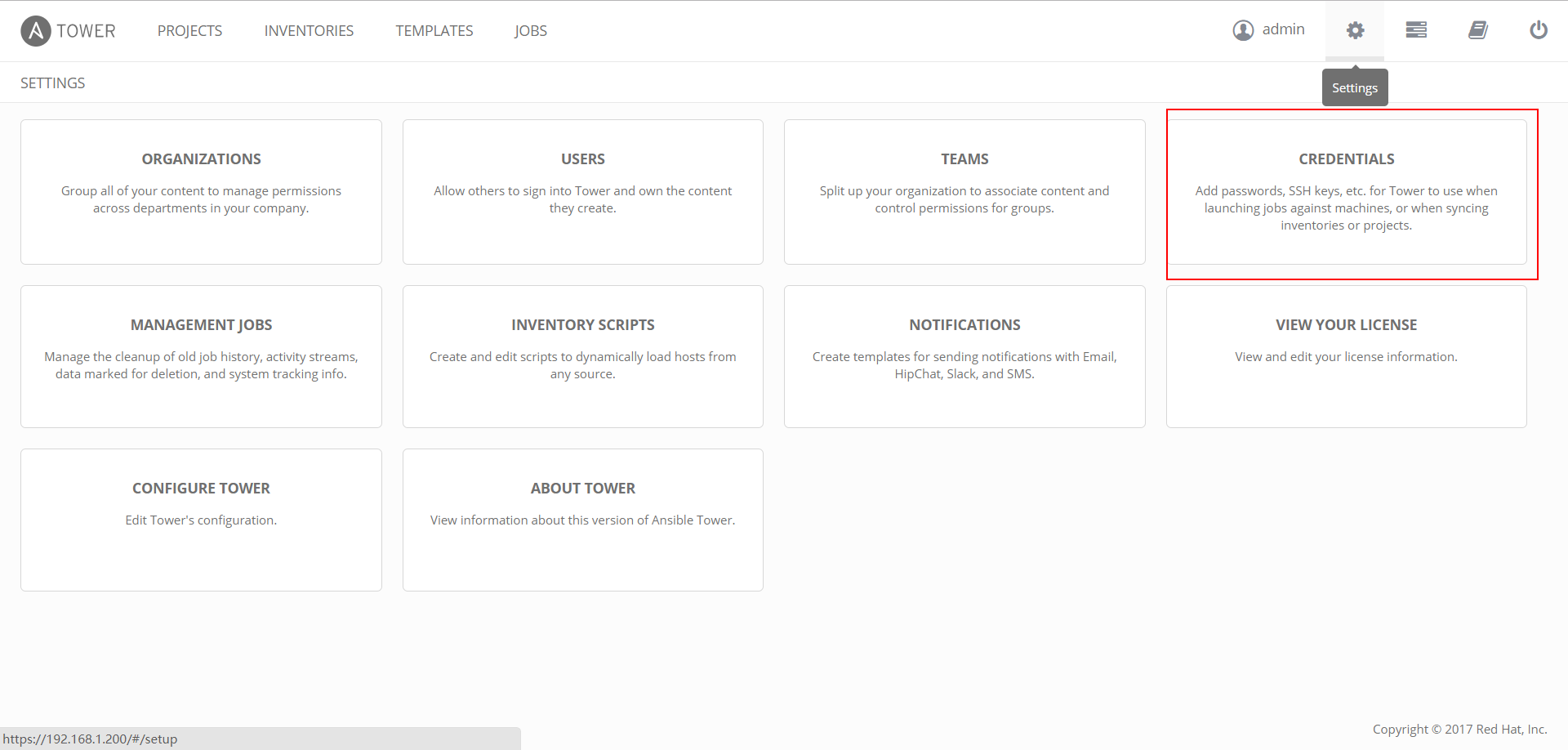


2）、在组里添加主机（也可以分别单独各自建立，最后把主机加入到组即可）



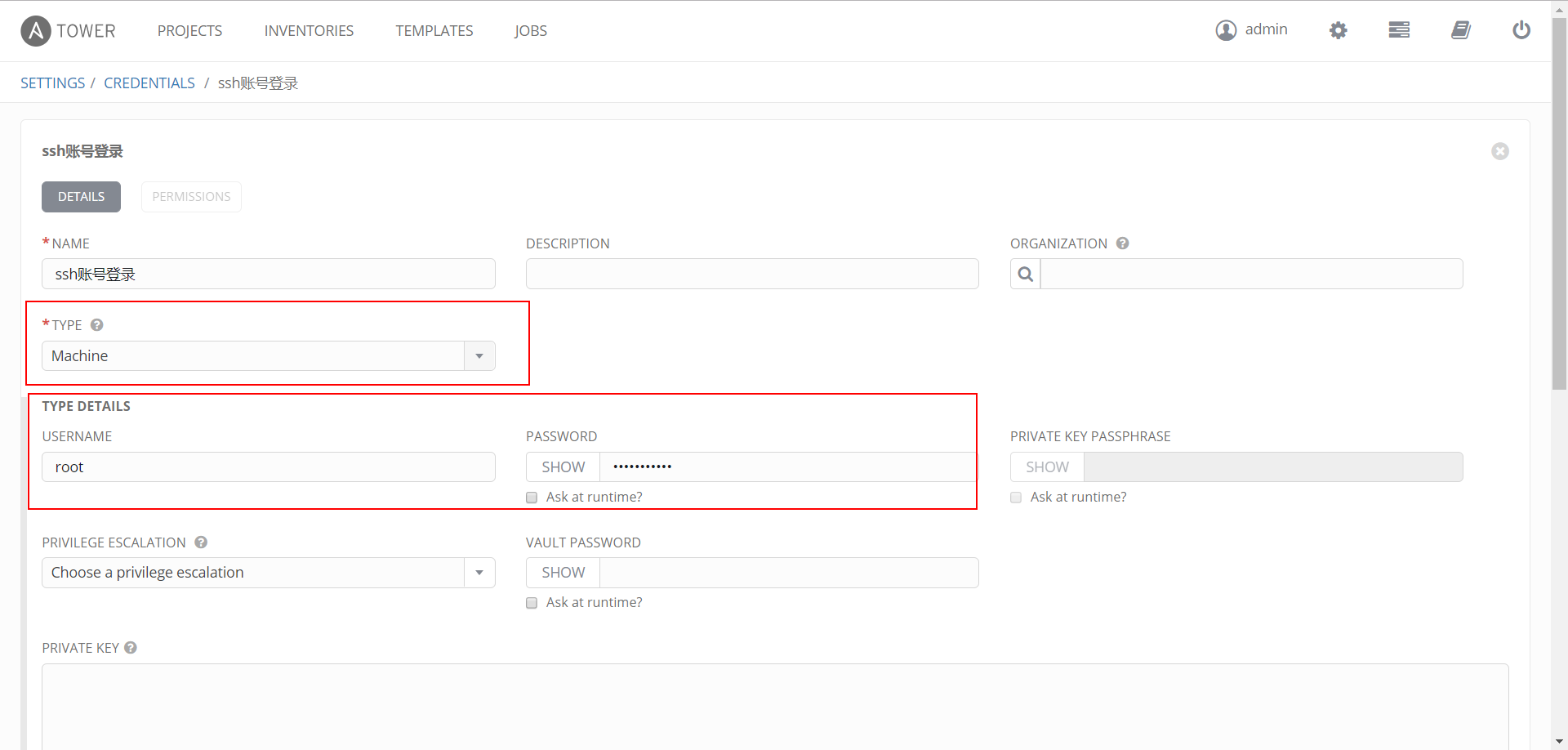
## 3.新建CREDENTIAL

点击settings进入系统设置页面，选择CREDENTIALS

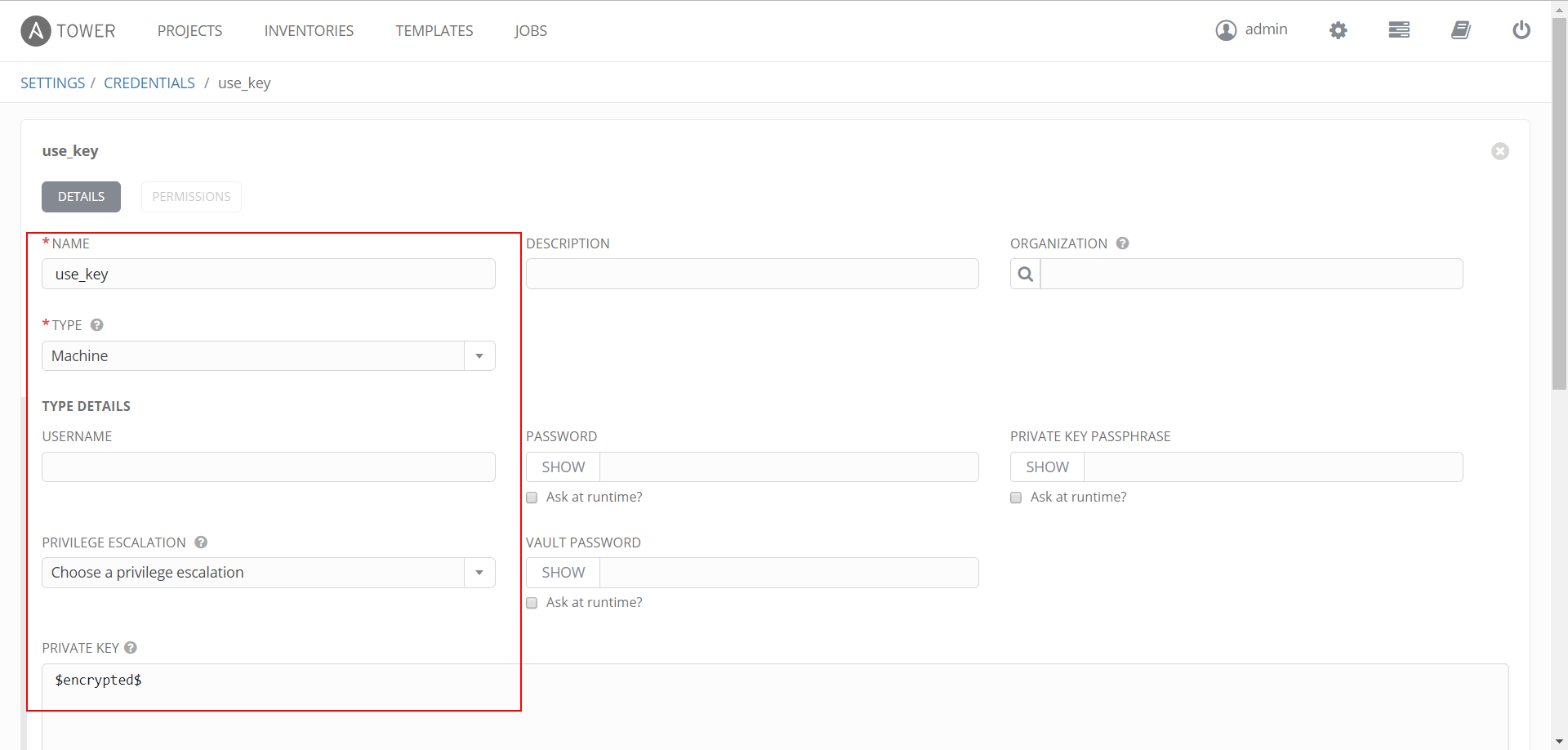


新建登录凭证

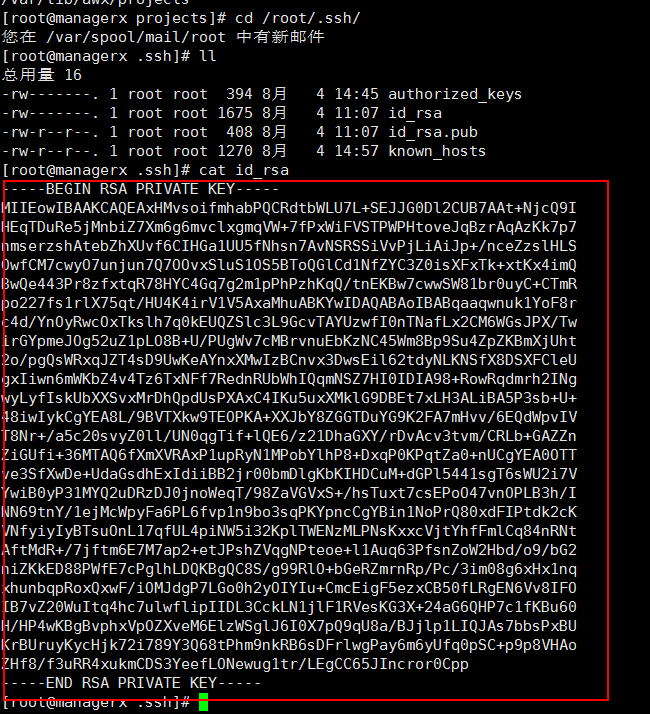
1）、可以直接使用账号密码



2）、可以使用sshkey



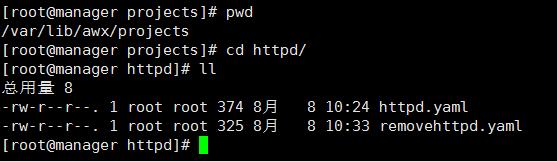
将管理端服务器密钥文件内容粘贴到private key里面即可

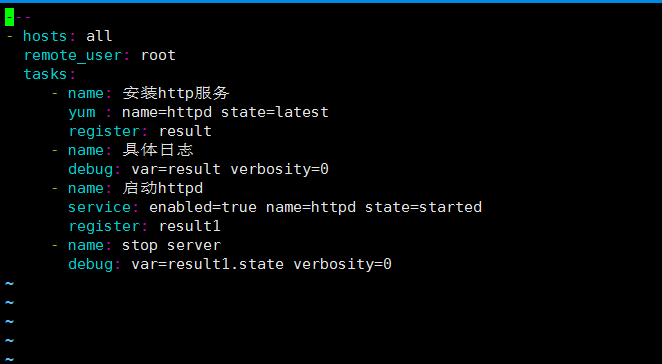


## 4.新建YAML编排文件

示例，安装httpd

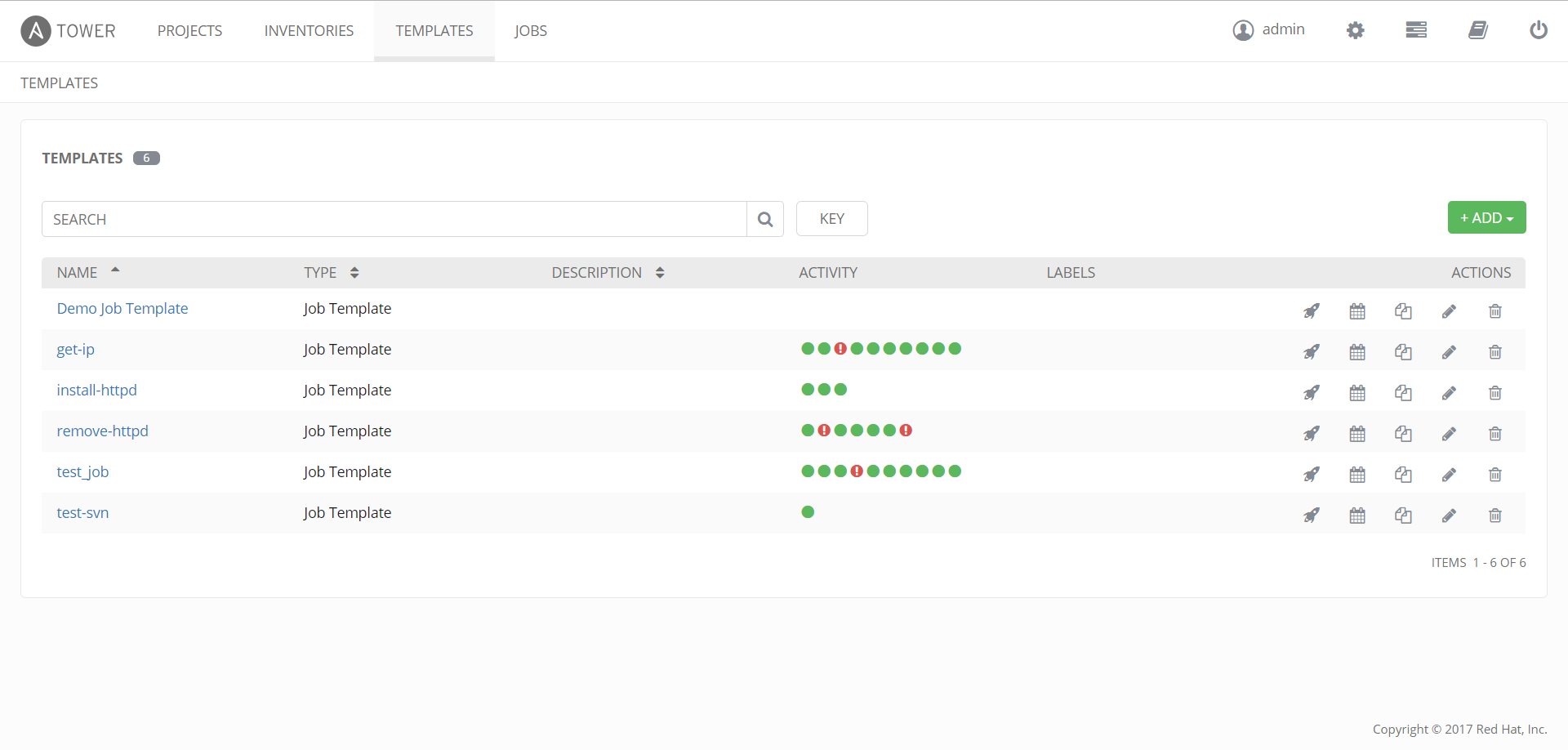
文件路径：参考PROJECTS配置



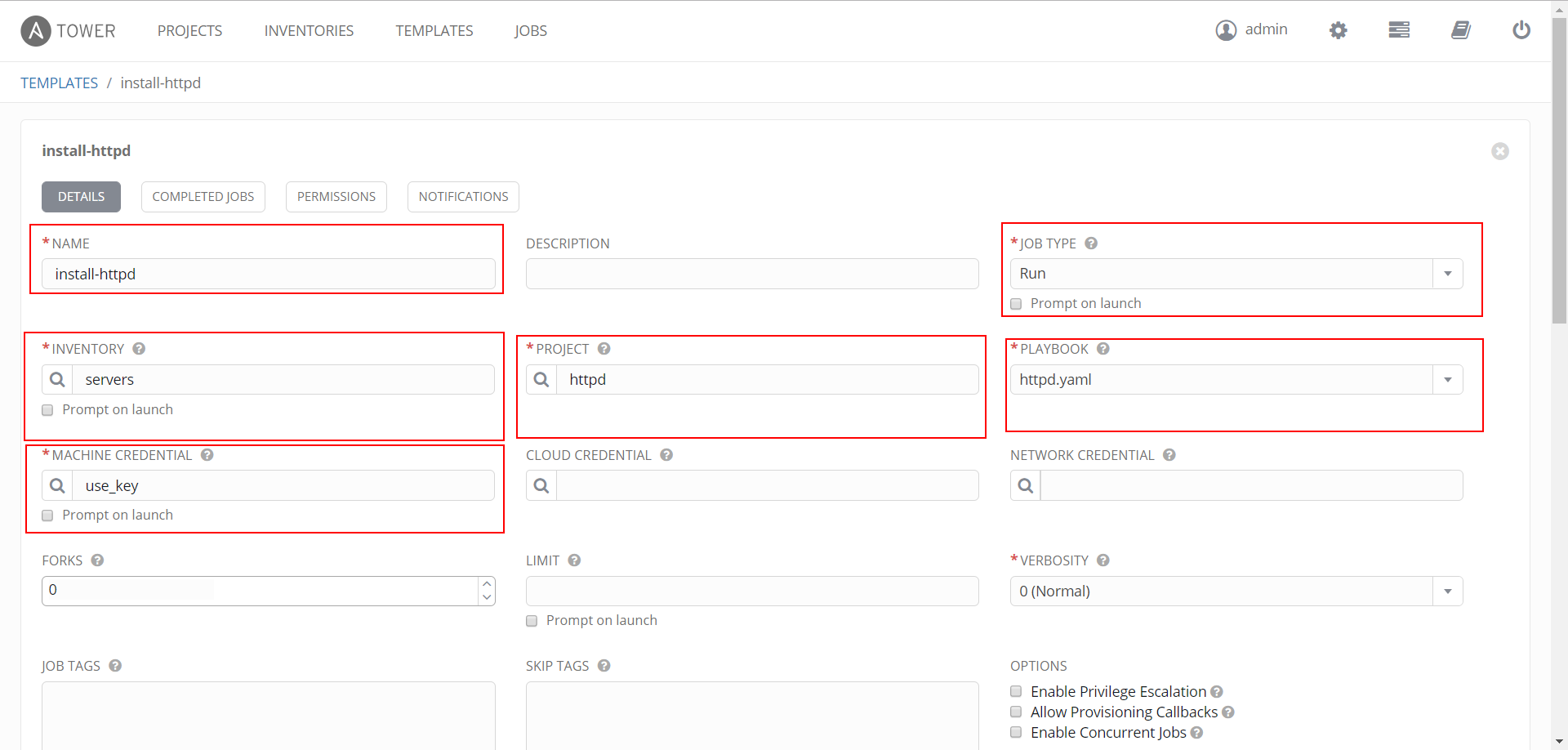


yaml语言详解建附录。

## 5.新建TEMPLATES



TEMPLATSE模板包含任务运行需要的各种资源



按照图示填入基本信息，如上是在服务器上面安装httpd服务的例子

其中：

**INVENTORIES**:第二步定义的的计算机组

**PROJECT**:第一步建立的工程

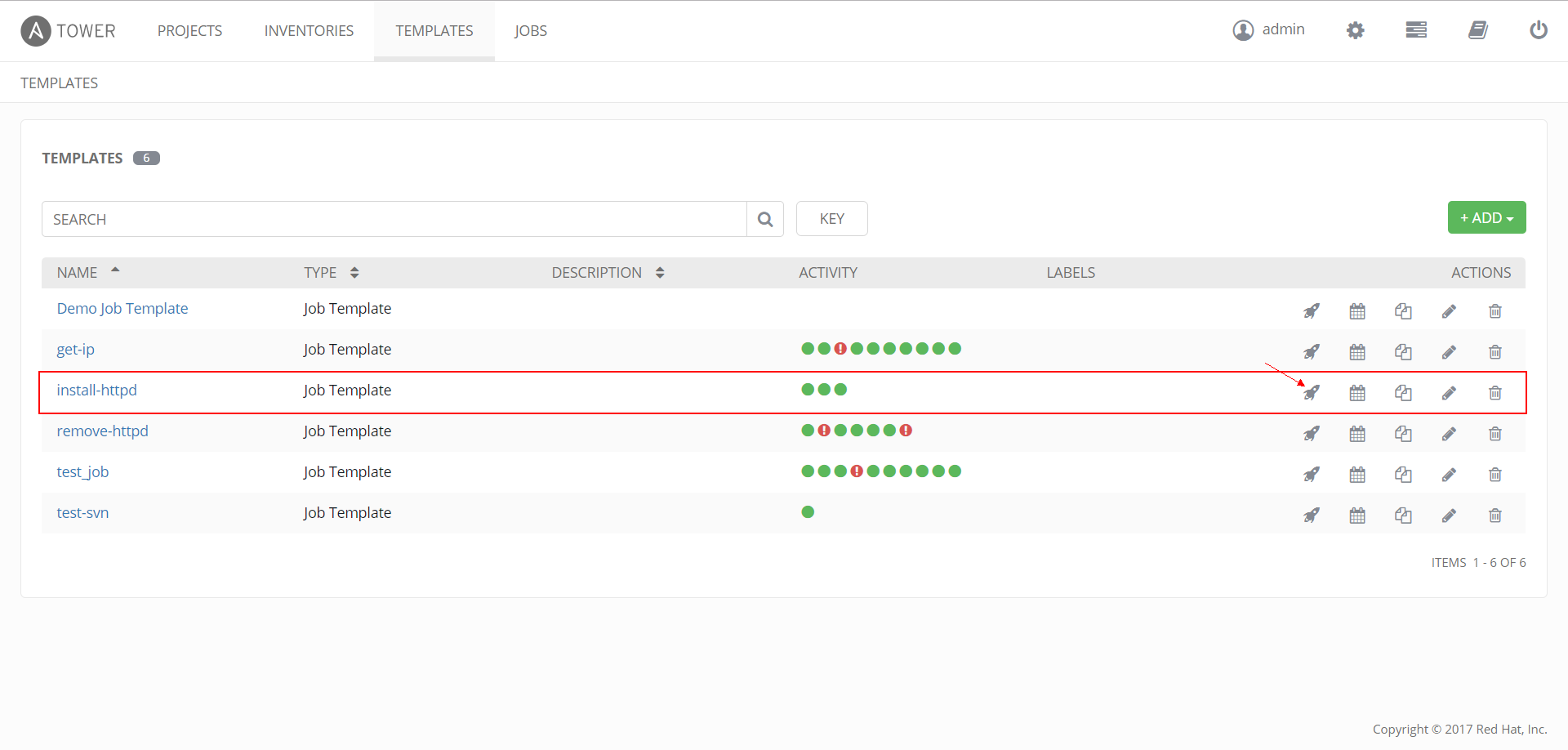
**PLAYBOOK**:编写好的yaml文件（内容就是安装httpd相关配置脚本）见第四步

**MACHINE** **CREDENTIAL**:第三步中建立的登录服务器组的凭证（分为直接使用用户密码或者使用ssh密钥）

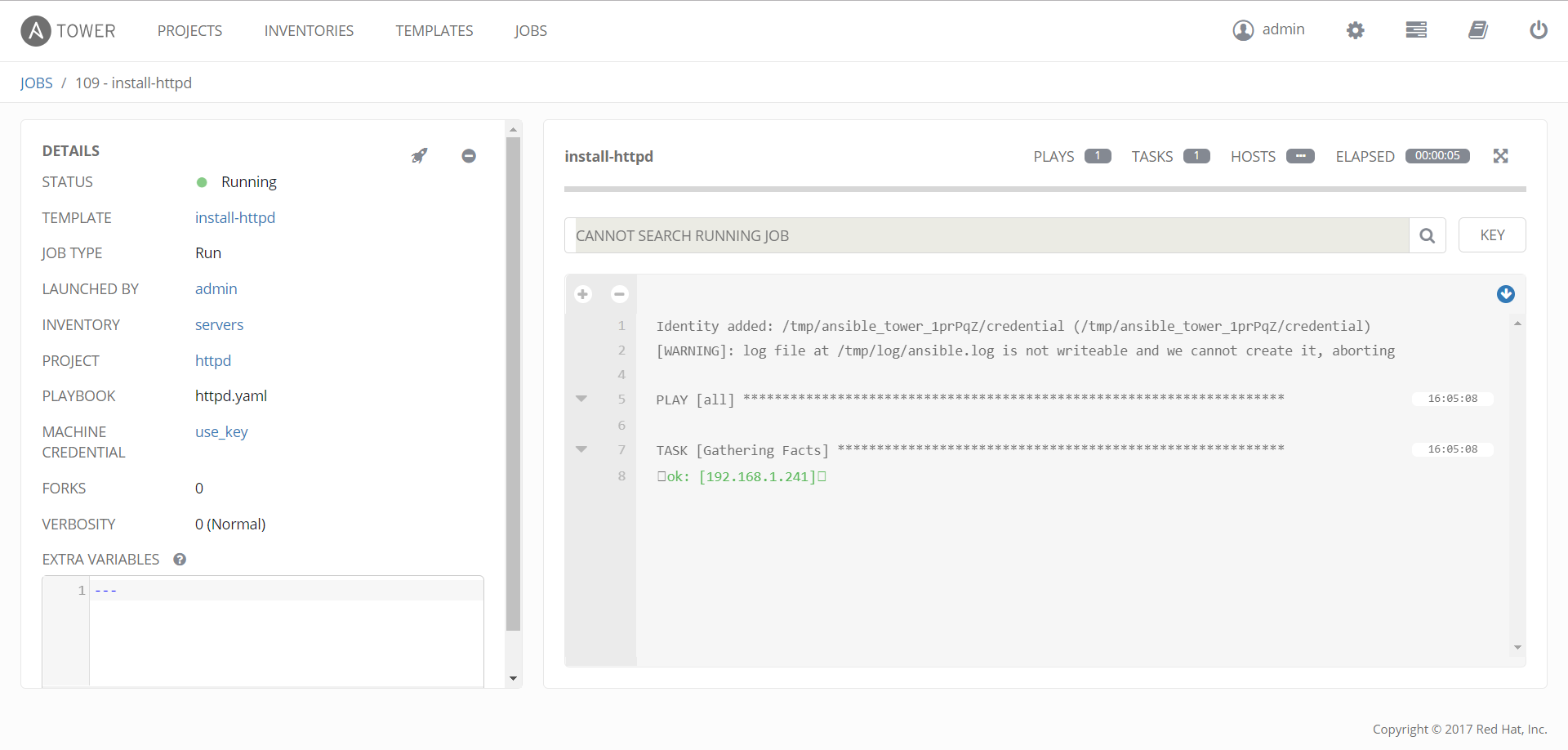
其它项目可以默认

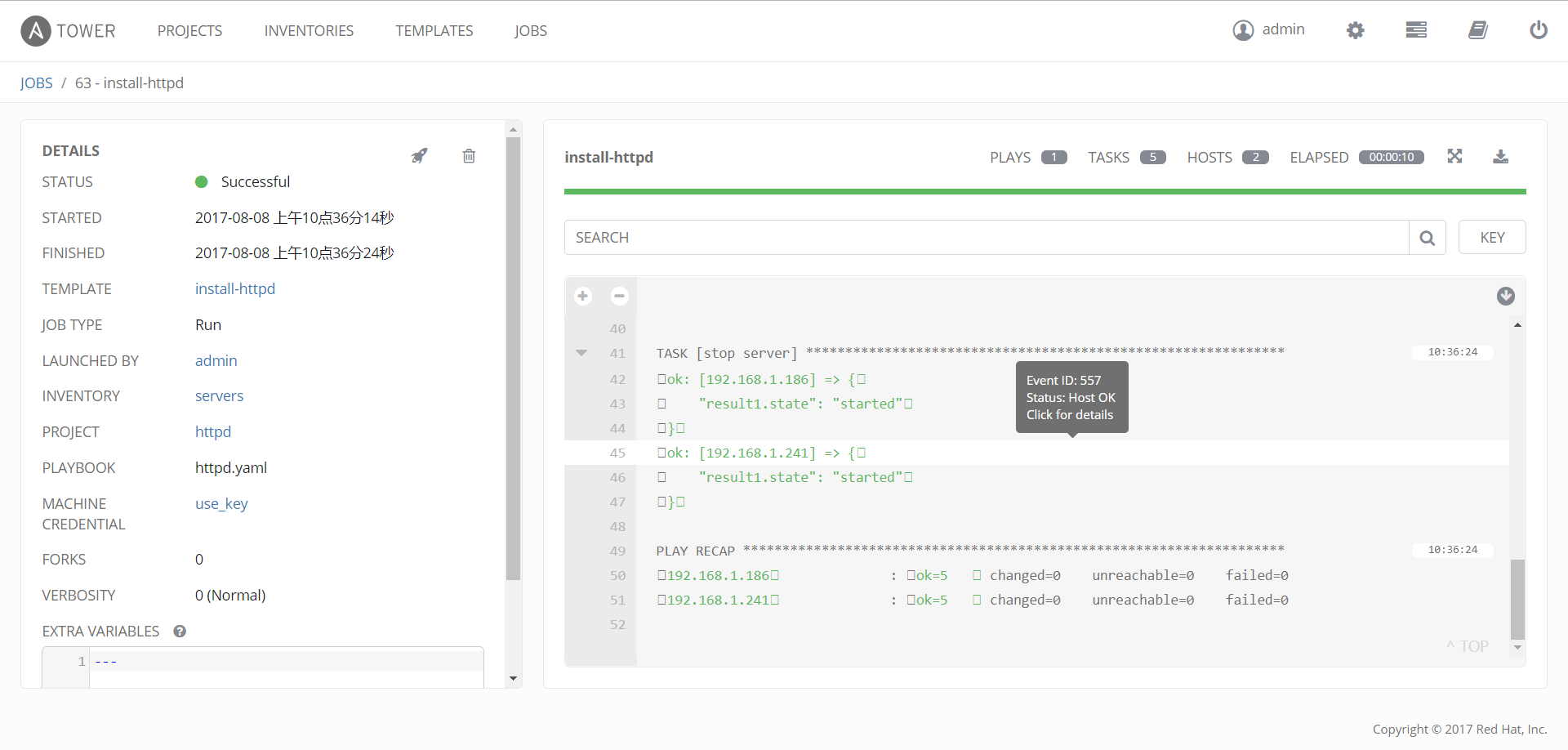
## 6.运行任务JOB

新建号模板后，可点击如图火箭按钮，即刻运行任务



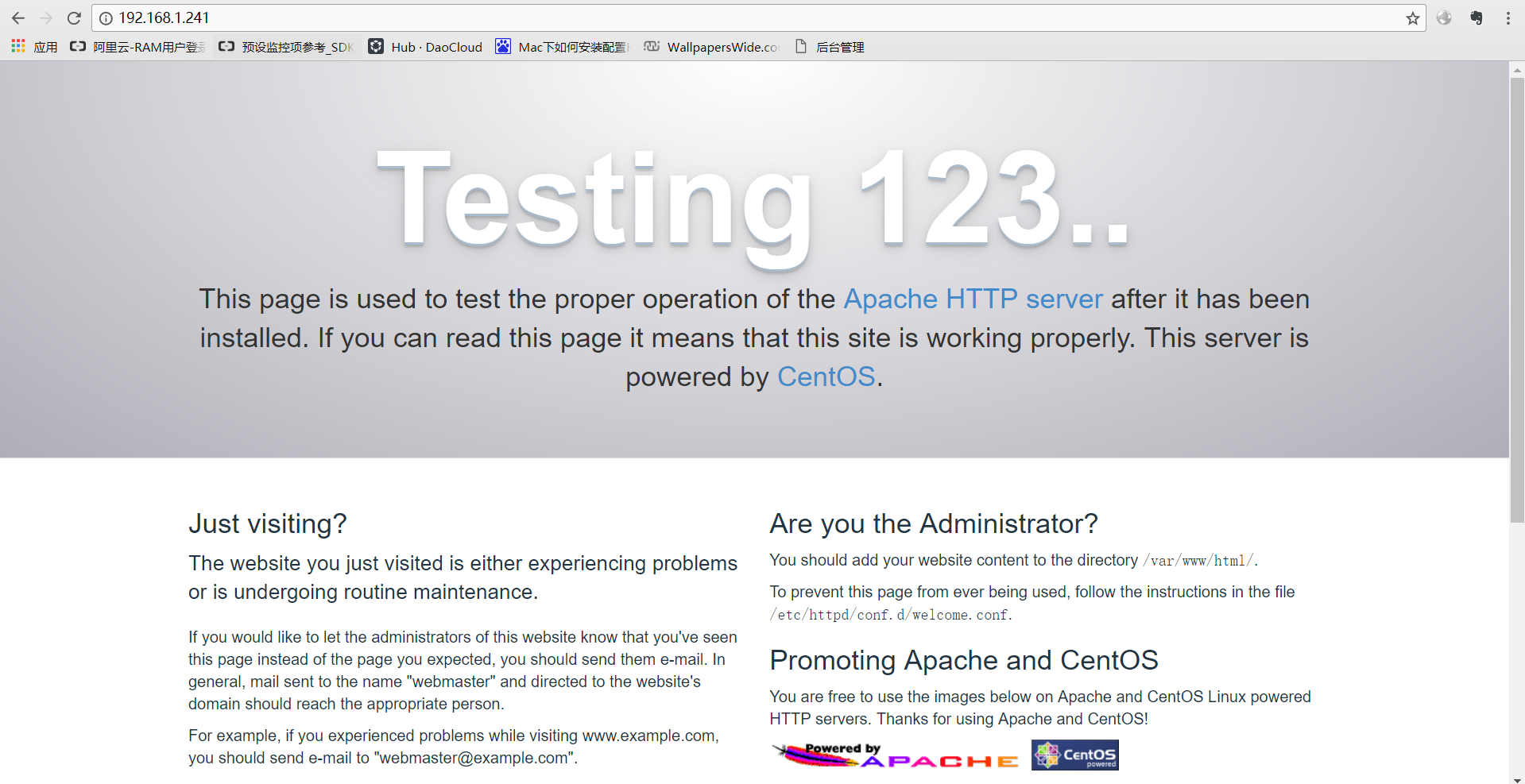
点击JOBS标签查看详细运行过程





可以看到同时在多台机器上面安装部署了httpd服务

可打开测试页面验证



# 附录

## A、ansible基本命令

### 1、/etc/ansible/hosts 文件

## Ansible 定义主机、组规则的配置文件

[复制代码](http://www.cnblogs.com/wangxiaoqiangs/p/5685239.html)

shell > vim /etc/ansible/hosts

www.abc.com # 定义域名

192.168.1.100 # 定义 IP

192.168.1.150:37268 # 指定端口号

[WebServer] # 定义分组

192.168.1.10

192.168.1.20

192.168.1.30

[DBServer] # 定义多个分组

192.168.1.50

192.168.1.60

Monitor ansible\_ssh\_port=12378 ansible\_ssh\_host=192.168.1.200 # 定义别名

# ansible\_ssh\_host 连接目标主机的地址

# ansible\_ssh\_port 连接目标主机的端口，默认 22 时无需指定

# ansible\_ssh\_user 连接目标主机默认用户

# ansible\_ssh\_pass 连接目标主机默认用户密码

# ansible\_ssh\_connection 目标主机连接类型，可以是 local 、ssh 或 paramiko

# ansible\_ssh\_private\_key\_file 连接目标主机的 ssh 私钥

# ansible\_\*\_interpreter 指定采用非 Python 的其他脚本语言，如 Ruby 、Perl 或其他类似 ansible\_python\_interpreter 解释器

[webservers] # 主机名支持正则描述

www[01:50].example.com

[dbservers]

db-[a:f].example.com

[复制代码](http://www.cnblogs.com/wangxiaoqiangs/p/5685239.html)

### 2、Ansible 常用模块学习

shell > ansible-doc -l # 列出 Ansible 支持的模块

shell > ansible-doc ping # 查看该模块帮助信息

>> 远程命令模块（ command / script / shell ）

command 作为 Ansible 的默认模块，可以运行远程权限范围所有的 shell 命令，不支持管道符。

例：

shell > ansible Client -m command -a "free -m" # 查看 Client 分组主机内存使用情况

script 的功能是在远程主机执行主控端存储的 shell 脚本文件，相当于 scp + shell 组合。

例：

shell > ansible Client -m script -a "/home/test.sh 12 34" # 远程执行本地脚本

shell 的功能是执行远程主机上的 shell 脚本文件，支持管道符。

例：

shell > ansible Client -m shell -a "/home/test.sh" # 执行远程脚本

>> copy 模块（实现主控端向目标主机拷贝文件，类似于 scp 功能）

例：

shell > ansible Client -m copy -a "src=/home/test.sh desc=/tmp/ owner=root group=root mode=0755" # 向 Client 组中主机拷贝 test.sh 到 /tmp 下，属主、组为 root ，权限为 0755

>> stat 模块（获取远程文件状态信息，atime/ctime/mtime/md5/uid/gid 等信息）

例：

shell > ansible Client -m stat -a "path=/etc/syctl.conf"

>> get\_url 模块（实现在远程主机下载指定 URL 到本地，支持 sha256sum 文件校验）

例：

shell > ansible Client -m get\_utl -a "url=http://www.baidu.com dest=/tmp/index.html mode=0440 force=yes"

>> yum 模块（软件包管理）

例：

shell > ansible Client -m yum -a "name=curl state=latest"

>> cron 模块（远程主机 crontab 配置）

例：

shell > ansible Client -m cron -a "name='check dirs' hour='5,2' job='ls -alh > /dev/null'"

效果：

#Ansible: check dirs

\* 5,2 \* \* \* ls -alh > /dev/null

>> mount 模块（远程主机分区挂载）

例：

shell > ansible Client -m mount -a "name=/mnt/data src=/dev/sd0 fstype=ext4 opts=ro state=present"

>> service 模块（远程主机系统服务管理）

例：

shell > ansible Client -m service -a "name=nginx state=stoped"

shell > ansible Client -m service -a "name=nginx state=restarted"

shell > ansible Client -m service -a "name=nginx state=reloaded"

>> user 服务模块（远程主机用户管理）

例：

shell > ansible Client -m user -a "name=wang comment='user wang'"

shell > ansible Client -m user -a "name=wang state=absent remove=yes" # 添加删除用户

五、Ansible-playbook

# 使用 Ansible-playbook 可以完成一组复杂的动作，例如部署环境、搭建服务、修改配置等。

简单示例：

[复制代码](http://www.cnblogs.com/wangxiaoqiangs/p/5685239.html)

shell > vim /etc/ansible/playbook.yml # 将远程主机IP地址写入文件中保存

---

- hosts: Client

remote\_user: root

tasks:

- name: Save IP To info.txt

shell: "ifconfig eth0 | awk -F '[ :]'+ '/inet addr/{print $4}' > ~/info.txt"

# hosts 指定执行操作主机

# remote\_user 指定执行用户

# tasks 指明有哪些动作

# name 动作描述

# shell 模块，后面为具体指令

[复制代码](http://www.cnblogs.com/wangxiaoqiangs/p/5685239.html)

## B、YAML语言基础

假设有三台主机为：n1,n2,n3

n1作为mater主机，其他三台作为受控主机

172.16.253.30主机为centos6系统，其他主机均为centos7系统

ssh-keygen -t rsa

#跳过生成私钥和公钥中间的填写内容

ssh-copy-id 172.16.253.184

ssh-copy-id 172.16.254.89

ssh-copy-id 172.16.253.47

ssh-copy-id 172.16.253.30

vim /etc/ansible/ansible.cfg

#替换下面指令为：

host\_key\_checking = False #本指令的含义为：每次在执行命令前都要检查该主机是否已经基于key做过认证

ansible srvs -m ping

#在每台主机上创建组group和user，并设置密码

ansible srvs -m group -a 'name="tomcat" state="present"'

ansible srvs -m user -a 'name="tomcat" group="tomcat" password="abcd" state="present"'

#以achudk用户的身份执行以root用户的权限执行ping指令

su - achudk

ansible srvs -m ping --become-user=root

基本使用

* 测试执行 ” -C ” 选项

ansible HOSTS -m MODULES -a 'DIRECTIVE' -C

#或者

ansible-playbook PLAYBOOK\_NAME -C

* 查询所有模块的命令

ansible-doc -l

command模块与shell模块的区别

shell功能全面但执行效率低

command不支持：逻辑运算符、条件判断符号、重定向命令或管道

ansible srvs -m command -a 'echo "$HOSTNAME"'

ansible srvs -m shell -a 'echo "$HOSTNAME"'

使用command模块会报错

ansible srvs -m command -a 'cat /etc/issue > /tmp/issue.tmp'

使用shell模块则不会

ansible srvs -m shell -a 'cat /etc/issue > /tmp/issue.tmp'

ansible srvs -m command -a 'cat /tmp/issue.tmp'

文件传输模块：copy、fetch、template

* copy模块：复制本机文件，传输至远程主机，并修改属主、属组和权限
* ==注意：可以在参数中增加 backup=yes ，可以实现备份的功能==

实现copy功能

ansible srvs -m copy -a 'src=/etc/yum.repos.d/CentOS-Base.repo dest=/etc/yum.repos.d/CentOS-Base.repo owner=root group=root mode=0644'

ansible srvs -m command -a 'yum clean all'

ansible srvs -m command -a 'yum repolist'

实现备份功能

ansible srvs -m copy -a 'src=/root/test.txt dest=/root/ backup=yes'

* fetch模块：复制远程文件，传输至本机（仅支持传输文件，如果传输目录可以先归档或压缩为一个文件后传输）

在n2主机的/root目录下创建文件f1，在n3主机的/root目录下创建文件f2，假设n2主机的/root目录下没有f2这个文件，假设n3主机的/root目录下没有f1这个文件，如果我们想从n2中传输f1这个文件至n1的/root目录下

* ==注意：使用fetch模块传输至本地主机的文件，全部会保存在目录，文件实际保存路径为：==
* /”dest参数指定的目录”/”源文件所在主机的IP地址”/”执行命令的用户名”/”传输的文件”

#在n1主机执行命令

ansible srvs -m fetch -a 'src=/root/f1 dest=/root/ backup=yes'

#f1文件的路径

ls 172.16.254.89/root/f1

cron模块

* 增加计划任务

设置每天的3:15，执行磁盘空间查看命令，并将命令结果保存至指定位置

ansible srvs -m cron -a 'name="disk check loger" minute=15 hour=3 job="df -hT >> /tpm/df.log"'

#验证结果

ansible srvs -m command -a 'crontab -l'

ansible srvs -m command -a 'cat /tmp/df.log'

* 移除指定计划任务

ansible srvs -m cron -a 'name="disk check loger" state=absent'

#验证

ansible srvs -m command -a 'crontab -l'

yum模块

安装epel源

ansible srvs -m yum -a 'name=epel-release state=latest'

安装应用

ansible srvs -m yum -a 'name=nginx state=latest'

ansible srvs -m yum -a 'name=ntpdate state=latest'

ansible srvs -m command -a 'ntpdate 172.16.0.1'

管理各种服务：service模块

ansible srvs -m service -a 'name=nginx state=started'

service模块中的state支持的选择有：started, stopped, restarted, reloaded

ansible srvs -m service -a 'name=nginx state=restarted'

setup 模块

自动从远程主机上获取可用与playbook中的变量

* 获取所有可用变量

ansible srvs -m setup

* 对变量进行过滤，仅获取符合匹配要求的变量

查询与系统版本相关的变量

“ansible\_distribution”: “CentOS”

“ansible\_distribution\_major\_version”: “7”

#方法1

ansible srvs -m setup -a 'filter=\*distribut\*'

#方法2

ansible srvs -m setup | grep -i distribut

debug 模块

可以通过debug模块调用变量显示想要的信息

tasks:

- name: show hosts information

tags: info

debug: msg="{{ ansible\_distribution }}-{{ ansible\_distribution\_major\_vers

ion }}-{{ ansible\_distribution\_release }}"

验证结果

ansible-playbook /root/ansible/httpd-manager-j2.yml --tags=info

#结果为

ok: [172.16.254.89] => {

"msg": "CentOS-7-Core"

}

script 模块

准备[**测试**](http://lib.csdn.net/base/softwaretest)脚本

vim /root/test.sh

#!/bin/bash

#

touch /tmp/test.log

echo `date` >> /tmp/test.log

ls -l /tmp/test.log

cat /tmp/test.log

在所有主机执行脚本

ansible srvs -m script -a '/root/test.sh'

ansible-vault 命令

* 加密 .yml 文件

#准备一个.yml文件

ansible-vault encrypt test.yml

#输入密码并确认

此时无法使用普通方法查看test.yml文件

* 查看加密的.yml文件

ansible-vault view test.yml

#输入密码

* 解密.yml文件

ansible-vault decrypt test.yml

#输入密码

此时可以使用普通方法查看.yml文件内容

ansible-galaxy 命令

* 文件下载至/etc/ansible/roles目录下
* ==注意：下载其他人共享的roles，在执行前应审核一遍文件内容，防止危险操作==
* 安装并使用其他人共享的roles文件

ansible-galaxy install devops.nginx

ls /etc/ansible/roles/devops.nginx

* 想删除下载的roles文件，将install替换为delete即可

ansible-console 命令

ansible的交互式命令工具

ansible-console 的命令提示符

root@all (4)[f:5]$ cd srvs

root@srvs (4)[f:5]$ forks 4

root@srvs (4)[f:4]$ list

172.16.253.184

172.16.254.89

172.16.253.47

172.16.253.30

其中：

root ：当前执行用户

@ ：分隔符

all ：分组的组名

(4) ：当前分组内的主机数量

[f:4] ：并发线程数

ansible-console 的使用

ansible-console工具可自动补全选项，使用格式为：

模块名 选项1=value 选项2=value

service name=nginx state=started

yum name=tree state=latest

command tree -L 1 /

**ansible-playbook命令及YAML语法**

YAML语法

YAML ：YAML ain’t a Markup Language ，以数据为中心、而不是以标记语言为重点的一种语言

* 首行以 “—” 开始，且需顶格书写
* 次行开始写playbook的正文，建议写明playbook的功能等描述信息
* 注释符为 ” # “
* 格式非常严格，严格控制缩进统一，不能和空格或Tab混用
* 缩进级别必须一致，缩进级别相同代表同级别，级别的判断是通过缩进结合换行实现的
* YAML文件内容和Linux系统大小写的判断方式保持一致，是区分大小写的，k/v的值均对大小写敏感
* k/v的值可同行写，也可换行写，同行书写使用分隔符 ” : ” ，换行写分隔符 ” - “
* 一个完整的代码块所必需包含的元素：name、task
* 一个name只能包括一个task

test.yml 示例

vim /root/test.yml

#文件内容

---

- hosts: srvs

remote\_user: root

tasks:

- name: adduser test

user: name=test shell=/bin/bash home=/home/test

playbook 核心元素

* hosts ：指定运行任务的目标主机
* tasks ：任务列表
* variables ：变量
* templates ：模板
* handlers ：动作执行探测器，探测到指定条件，触发执行动作
* roles ：按照固定目录结构组成的playbook

playbook中tags、notify、handlers的使用

* ==playbook实例 ：httpd-manager.yml==

---

- hosts: srvs

remote\_user: root

tasks:

- name: install httpd

tags: install

yum: name=httpd state=latest

- name: start httpd.service

tags: start

service: name=httpd state=started

- name: start httpd.service

tags: restart

service: name=httpd state=restarted

- name: stop httpd.service

tags: stop

service: name=httpd state=stopped

- name: config httpd

tags: config

copy: src=/root/ansible/httpd.conf dest=/etc/httpd/conf/ backup=yes

when: ansible\_distribution\_major\_version == "7"

notify: restart httpd

handlers:

- name: restart httpd

service: name=httpd state=restarted

* 实现功能：
* 单独调用其中一个功能

ansible-playbook httpd-server-deploy.yml --tags=start

* 同时调用多个功能

ansible-playbook httpd-server-deploy.yml --tags="stop,config"

* 针对某一单个成员执行某一功能

ansible-playbook httpd-server-deploy.yml --tags="start,stop" -l 172.16.253.30

playbook中变量的使用

定义变量

在/etc/ansible/hosts文件中定义变量，可针对某台主机定义变量，也可以对整个分组定义变量

组变量和主机变量的生效优先级为：如果针对主机定义了变量（无论是否定义了组变量），主机变量生效；如果未定义主机变量，仅定义了组变量，组变量生效。

vim /etc/ansible/hosts

#组变量

[srvs:vars]

webport=80

#主机变量

[srvs]

172.16.253.184 webport=10080 hostname=node1

172.16.254.89 webport=10180 hostname=node2

172.16.253.47 webport=10280 hostname=node3

172.16.253.30 webport=10380 hostname=node4

调用变量

调用变量的两种方式的生效优先级：在命令行下直接定义的变量优先级高于在hosts文件中定义的变量（主机变量或分组变量）

1. 可以在命令行模式直接使用”ansible-playbook test.yml -e “webport=2000”

2. 调用在hosts文件中定义好的变量

引用变量注意：使用双大括号 “{{ VAR-NAME }}” ，且括号前后要有空格

* 修改主机名

ansible srvs -m hostname -a 'name=“name-{{ hostname }}-port-{{ webport }}"'

ansible srvs -m command -a 'hostname'

#执行结果

172.16.253.30 | SUCCESS | rc=0 >>

name-node4-port-10380

172.16.253.184 | SUCCESS | rc=0 >>

name-node1-port-10080

172.16.253.47 | SUCCESS | rc=0 >>

name-node3-port-10280

172.16.254.89 | SUCCESS | rc=0 >>

name-node2-port-10180

ansible-playbook 之强大的 template

* jianja2语法格式：

支持算术运算：+ ， - ， \* ， / ， \*\*

支持比较操作符：== ， >= , != , <= , > , <

支持逻辑运算符： and ， or ， not

支持流表达式：for ， if ， while

使用template在配置文件中调用变量

* 设置template配置文件

#设置jianja2文件

cp /root/ansible/httpd.conf /root/ansible/httpd.conf.j2

vim /root/ansible/httpd.conf.j2

#修改Listen一项的值为 "{{ webport }}"

Liten {{ webport }}

#重新设置yml文件

cp /root/ansible/httpd-manager.yml /root/ansible/httpd-manager-j2.yml

vim /root/ansible/httpd-manager-j2.yml

#修改config配置块中的copy指令为template指令，其他部分不变，示例如下：

- name: config httpd

tags: config

template: src=/root/ansible/httpd.conf.j2 dest=/etc/httpd/conf/httpd.conf

when: ansible\_distribution\_major\_version == "7"

notify: restart httpd

* 执行playbook

ansible-playbook /root/ansible/httpd-manager-j2.yml --tags=config

#执行结果：对分组中不同的主机采用不同的端口

ansible srvs -m shell -a 'ss -tnlp | grep httpd'

* template还支持算术运算

将所有端口加1

vim /root/ansible/httpd.conf.j2

#修改以下内容

Listen {{ webport+1 }}

通过template在配置文件中使用for循环和if语句

* 定义基于不同域名的多个虚拟主机

#查找主机名变量

ansible srvs -m setup | grep hostname

#修改httpd.conf.j2文件

Listen {{ webport }}

<VirtualHost \*.{{ webport }}>

ServerName {{ ansible\_hostname }}.achudk.com

DocumentRoot /var/www/html

<Directory "/var/www">

Require all granted

</Directory>

</VirtualHost>

#验证结果

ansible-playbook /root/ansible/httpd-manager-j2.yml --tags=config

curl curl 172.16.254.89:10180

* 使用for循环生成多个基于不同域名且不同端口的虚拟主机，并使用if判断条件

#修改httpd.conf.j2文件

vim httpd.conf.j2

{% for vport in vports %}

<VirtualHost \*.{{ vport }}>

ServerName {{ ansible\_hostname }}.achudk.com

DocumentRoot /var/www/html

<Directory "/var/www">

Require all granted

</Directory>

{% if errorlog is not defined %}

ErrorLog "{{ errorlog | default('/web/httpd/log/vh1/error\_log') }}"

{% endif %}

</VirtualHost>

{% endfor %}

#修改httpd-manager-j2.yml文件,在起始处增加vars指令，余下内容不变，如下：

vim httpd-manager-j2.yml

#正文如下

---

- hosts: srvs

remote\_user: root

vars:

- vport:

- 80

- 443

- 8080

- errorlog:

- "/web/httpd/log/vh1/error\_log"

#验证结果

ansible-playbook /root/ansible/httpd-manager-j2.yml --tags=config

* when条件判断

#当系统为CentOS7时，才启用该配置片段

- name: config httpd

tags: config

template: src=/root/ansible/httpd.conf.j2 dest=/etc/httpd/conf/httpd.conf

when: ansible\_distribution == "CentOS"

when: ansible\_distribution\_major\_version == "7"

notify: restart httpd

* ==when指令的多条件联合判断==

多个条件之间使用 “and” 连接

tasks:

- name: show hosts information

tags: info

debug: msg={{ ansible\_distribution }},msg={{ ansible\_distribution\_major\_v

ersion }}

when: ansible\_distribution == "CentOS" and ansible\_distribution\_major\_ver

sion == "6"

==最终实例==

将所学内容综合起来使用，形成具有管理功能的yml文件

---

- hosts: srvs

remote\_user: root

vars:

- vports:

- 80

- 443

- 8080

- errorlog:

- "/web/httpd/log/vh1/error\_log"

tasks:

- name: show hosts information

tags: info

debug: msg="{{ ansible\_distribution }}-{{ ansible\_distribution\_major\_version }}-{{ ansible\_distribution\_release }}"

- name: install httpd

tags: install

yum: name=httpd state=latest

- name: start httpd.service

tags: start

service: name=httpd state=started

- name: start httpd.service

tags: restart

service: name=httpd state=restarted

- name: stop httpd.service

tags: stop

service: name=httpd state=stopped

- name: config httpd

tags: config

template: src=/root/ansible/httpd.conf.j2 dest=/etc/httpd/conf/httpd.conf

when: ansible\_distribution == "CentOS" and ansible\_distribution\_major\_ver

sion == "7"

notify: restart httpd

handlers:

- name: restart httpd

service: name=httpd state=restarted