# 注意事项

### 1. 使用root用户

- 固定IP
- 安装相关yum源
- 安装nginx、keepalived相关rpm包
- 关闭防火墙
- 系统参数调整
- 修改用户权限

### 2. 使用cloud用户

- 除以上使用**root**用户的情况，其余全用**cloud**用户。
- 所有服务安装在**/home/cloud/apps**目录下。

# VMware安装centos7镜像

### 1. 参考文档

https://blog.csdn.net/babyxue/article/details/80970526

### 2. 注意事项

- 磁盘配额
- SOFT SELECTION => Server with GUI指选择图形界面，根据需要是否选择。
- 用户名/密码: **root/123456**

### 3. 固定IP

```
[root@sit ~]# vi /etc/sysconfig/network-scripts/ifcfg-ens33
```

```
TYPE="Ethernet"
PROXY_METHOD="none"
BROWSER_ONLY="no"
# dhcp改为static
BOOTPROTO="static"
DEFROUTE="yes"
IPV4_FAILURE_FATAL="no"
IPV6INIT="yes"
IPV6_AUTOCONF="yes"
IPV6_DEFROUTE="yes"
IPV6_FAILURE_FATAL="no"
IPV6_ADDR_GEN_MODE="stable-privacy"
NAME="ens33"
UUID="faf68ce4-4ed9-4991-9475-7874b8fde25f"
DEVICE="ens33"
ONBOOT="yes"
# 填写需要的静态ip,xxx与宿主机相同,yyy指定数字
IPADDR="192.168.xxx.yyy"
# 宿主机网关
GATEWAY="192.168.xxx.1"
# 宿主机子网掩码
```

```
NETMASK="255.255.255.0"
DNS1=114.114.114.114
DNS2=8.8.8.8
```

```
[root@sit ~]# service network restart
```

## 4. 安装相关yum源

```
# ifconfig
[root@sit ~]# yum install -y net-tools
# telnet
[root@sit ~]# yum install -y telnet-server
[root@sit ~]# yum install -y telnet.*
# wget
[root@sit ~]# yum install -y wget
# gcc
[root@sit ~]# yum install -y gcc
# g++
[root@sit ~]# yum install -y gcc-c++ libstdc++-devel
# vim
[root@sit ~]# yum install -y vim*
# lsof
[root@sit ~]# yum install -y lsof
# perl
[root@sit ~]# yum install -y perl
# openssl
[root@sit ~]# yum install -y openssl
# unzip
[root@sit ~]# yum install -y unzip zip
# libnl
[root@sit ~]# yum install -y libnl libnl-devel
# libnfnetlink-devel
[root@sit ~]# yum install -y libnfnetlink-devel
# git
[root@sit ~]# yum install -y git
```

## 5. yum常用命令

```
# 列举所有包
[root@sit ~]# yum list
# 搜索包
[root@sit ~]# yum search
# 安装包，-y免于确认是否安装
[root@sit ~]# yum -y install 包名
# 升级包，一定要指定包，不然linux全局更新
[root@sit ~]# yum -y update 包名
# 卸载包，尽量不卸载
[root@sit ~]# yum -y remove 包名
```

## 6. 关闭防火墙

```
# 检查防火墙的状态
[root@sit ~]# firewall-cmd --state
# 停止firewall
[root@sit ~]# systemctl stop firewalld.service
# 禁止firewall开机启动
[root@sit ~]# systemctl disable firewalld.service
```

## 7. 虚拟机与宿主机时间同步

- 点击虚拟机的上面菜单栏 VM选择Install VMware Tools，这就安装了虚拟机工具。
- 点击VM选择settings，再选择options页签，选择VMware Tools，勾选上右上方的Synchronize guest
  time with host 这样在xshell中执行date，即可看到时间已同步。

## 8. 设置开机自启动

```
# 查看rc.local文件权限
[root@sit4 ~]# ll /etc/rc.d/rc.local /etc/rc.local
# 如果没有x权限
[root@sit4 ~]# chmod 744 /etc/rc.d/rc.local
```

```
# 查看是否开启rc.local服务
[root@sit4 ~]# systemctl list-unit-files|grep rc.local
static 代表已开启；disable 代表未开启
```

```
# 编辑rc.local文件
[root@sit4 ~]# vim /etc/rc.d/rc.local
```

```
#!/bin/bash
# THIS FILE IS ADDED FOR COMPATIBILITY PURPOSES
#
# It is highly advisable to create own systemd services or udev rules
# to run scripts during boot instead of using this file.
#
# In contrast to previous versions due to parallel execution during boot
# this script will NOT be run after all other services.
#
# Please note that you must run 'chmod +x /etc/rc.d/rc.local' to ensure
# that this script will be executed during boot.

touch /var/lock/subsys/local
su - cloud -c "cd /home/cloud/apps/nginxs/nginx/sbin && ./nginx"
su - cloud -c "cd /home/cloud/apps/emqx-centos7-4.2.14-x86_64/bin && ./emqx
start"
```

```
# 启动rc.local服务
[root@sit ~]# systemctl enable rc-local.service
```

```
# 手动启动rc.local服务
[root@sit ~]# systemctl status rc-local.service
```

```
# 自动启动rc.local服务
[root@sit ~]# systemctl start rc-local.service
```

### 8.1 注意

- /etc/rc.d/rc.local文件中需要加上"#!/bin/bash"
- 每次修改/etc/rc.d/rc.local最好重启一次虚拟机服务器

### 8.2 参考

Linux中/etc/rc.d/rc.local中配置的启动项未生效原因总结

# 系统参数调整

## 1. TCP通讯参数调整

```
[root@sit ~]# vi /etc/sysctl.conf
```

```
#增加或修改以下内容:
net.ipv4.tcp_tw_reuse=1
net.ipv4.tcp_tw_recycle=0
net.ipv4.tcp_syncookies=1
net.ipv4.tcp_fin_timeout=30
net.ipv4.tcp_keepalive_time=1200
net.ipv4.tcp_max_syn_backlog=8192
net.ipv4.tcp_max_tw_buckets=20000
net.ipv4.ip_local_port_range=10240 65535
net.ipv4.tcp_retries2=5
net.ipv4.tcp_syn_retries=3
#ES相关服务器新增
vm.max_map_count=655360
fs.file-max=655360
#关闭IPV6
net.ipv6.conf.all.disable_ipv6=1
net.ipv6.conf.default.disable_ipv6=1
net.ipv6.conf.lo.disable_ipv6=1
```

```
# 修改完后执行以下命令使之生效
[root@sit ~]# /sbin/sysctl -p
```

## 2. 修改文件句柄数及用户可用最大线程数

```
[root@sit ~]# vi /etc/security/limits.conf
```

```
#新增以下内容(注意必须有空格)
*               soft    nofile          65535
*               hard    nofile          65535
*               soft    nproc          65535
*               hard    nproc          65535
*               soft    memlock          unlimited
*               hard    memlock          unlimited
```

## 3. 修改用户可用最大线程数

```
[root@sit ~]# vi /etc/security/limits.d/20-nproc.conf
```

```
#新增以下内容(若已配置则需修改，需修改成102400):
*          soft    nproc     102400
```

### 4. 测试参数生效

root用户修改文件句柄数后，执行命令**ulimit -a**文件句柄数依然是1024？如果不生效则依次执行以下命令？

```
[root@sit ~]# echo "UsePAM yes"  >> /etc/ssh/sshd_config && echo "#%PAM-1.0" >> /etc/pam.d/sshd && echo "auth include  password-auth" >> /etc/pam.d/sshd && echo "account  include  password-auth" >> /etc/pam.d/sshd && echo "password  include password-auth" >> /etc/pam.d/sshd && echo "session include  password-auth" >> /etc/pam.d/sshd
```

```
[root@sit ~]# systemctl restart sshd
```

```
[root@sit ~]# exit
```

退出重新连接后文件句柄数发生变化，变为65535。

## 用户操作

### 1. 新增用户组

```
[root@sit ~]# groupadd cloud
```

### 2. 新增用户

```
[root@sit ~]# useradd -g cloud cloud
```

```
# ubuntu系统执行如下操作

# -m 相当于会创建对应的用户家目录
[root@sit ~]# useradd username -m
# 指定shell，否则会非常不便于终端操作(username登录出现$...)
[root@sit ~]# usermod -s /bin/bash username
```

### 3. 修改密码

```
# 连续两次输入修改后的密码即可
[root@sit ~]# passwd cloud
```

### 4. 合并语句

```
# 新增用户组、新增用户、修改密码
[root@sit ~]# groupadd cloud && useradd -g cloud cloud && echo "cloud" | passwd cloud --stdin > /dev/null 2>&1
```

### 5. 切换cloud用户

```
# 针对所用机器
[cloud@sit ~]$ mkdir apps && mkdir logs
```

### 6. linux用户登录后显示bash-4.2$

```
[root@sit ~]# cp /etc/skel/.bashrc  /home/someUser
[root@sit ~]# cp /etc/skel/.bash_profile  /home/someUser
[root@sit ~]# cp /etc/skel/.bash_logout  /home/someUser
```

### 7. 修改用户权限

```
[root@sit ~]# vim /etc/sudoers
```

```
找到这行root ALL=(ALL) ALL, 在他下面添加xxx ALL=(ALL) ALL(这里的xxx是你的用户名)
```

```
ps:这里说下你可以sudoers添加下面四行中任意一条
youuser            ALL=(ALL)                ALL
%youuser           ALL=(ALL)                ALL
youuser            ALL=(ALL)                NOPASSWD: ALL
%youuser           ALL=(ALL)                NOPASSWD: ALL

第一行: 允许用户youuser执行sudo命令(需要输入密码)
第二行: 允许用户组youuser里面的用户执行sudo命令(需要输入密码)
第三行: 允许用户youuser执行sudo命令,并且在执行的时候不输入密码
第四行: 允许用户组youuser里面的用户执行sudo命令,并且在执行的时候不输入密码
```

```
e.g:
%jenkins           ALL=(ALL)                NOPASSWD: ALL
%cloud           ALL=(ALL)                NOPASSWD: ALL
```

## 磁盘挂载

### 1. 列出所有可用块设备的信息

```
[cloud@sit ~]$ lsblk
```

```
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0  100G  0 disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   99G  0 part
  ├─centos-root 253:0    0   50G  0 lvm  /
  ├─centos-swap 253:1    0   10G  0 lvm  [SWAP]
  └─centos-home 253:2    0   39G  0 lvm  /home
sdb               8:16   0  512G  0 disk
sr0              11:0    1  4.3G  0 rom  /run/media/root/CentOS 7 x86_64
```

### 2. 分区

```
[cloud@sit ~]$ fdisk /dev/sdb
```

```
n // 添加新分区
p // 选择主分区
w // 写入分区并保存退出
(全部默认选择)
```

### 3. 同步分区

```
[cloud@sit ~]$ partprobe /dev/sdb1
```

### 4. 添加ext4文件系统

```
[cloud@sit ~]$ mkfs.ext4 /dev/sdb1
```

### 5. 查询文件系统类型

```
[cloud@sit ~]$ blkid /dev/sdb1
```

```
/dev/sdb1: UUID="79c0bfe4-b1a9-4c48-a2a4-991b61fec621" TYPE="ext4"
```

### 6. 挂载分区

```
[cloud@sit ~]$ mkdir /app
[cloud@sit ~]$ blkid /dev/sdb1
[cloud@sit ~]$ vim /etc/fstab
```

```
UUID="79c0bfe4-b1a9-4c48-a2a4-991b61fec621" /app ext4  defaults 0 0
......

:r!blkid /dev/sdb1 // 添加UUID(命令blkid)
```

```
[cloud@sit ~]$ mount -a
```

### 7. 查看

```
[cloud@sit ~]$ lsblk
```

```
NAME            MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
sda               8:0    0  100G  0 disk
├─sda1            8:1    0    1G  0 part /boot
└─sda2            8:2    0   99G  0 part
  ├─centos-root 253:0    0   50G  0 lvm  /
  ├─centos-swap 253:1    0   10G  0 lvm  [SWAP]
  └─centos-home 253:2    0   39G  0 lvm  /home
sdb               8:16   0  512G  0 disk
└─sdb1            8:17   0  512G  0 part /app
sr0              11:0    1  4.3G  0 rom  /run/media/root/CentOS 7 x86_64
```

## ssh和sftp免密登录

### 1. local_user生成秘钥对

```
# 如果本地不存在秘钥对，则创建
[cloud@sit ~]$ ssh-keygen -t rsa
```

```
一路回车生成
私钥: /home/local_user/.ssh/id_dsa
公钥: /home/local_user/.ssh/id_dsa.pub
```

### 2. remote_user获取公钥

#### 2.1 创建.ssh文件夹

```
[cloud@sit ~]$ cd /home/remote_user/.ssh
```

```
# 不存在则创建
[cloud@sit ~]$ mkdir -p /home/remote_user/.ssh
# .ssh目录必须是755或者700权限
[cloud@sit .ssh]$ chmod -R 755 .ssh
```

#### 2.2 创建authorized_keys文件

```
# 不存在则创建
[cloud@sit .ssh]$ touch authorized_keys
# authorized_keys权限必须是644
[cloud@sit .ssh]$ chmod -R 644 authorized_keys
```

最后复制id_rsa.pub文本内容至authorized_keys文件下。

## 安装JDK

所有服务器均需安装。

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf jdk1.8.0_181.tar.gz && rm jdk1.8.0_181.tar.gz &&
chmod -R 755 jdk1.8.0_181/
```

### 2. 配置环境变量

```
[cloud@sit apps]$ vi ~/.bash_profile
```

```
# 文件末尾添加
JAVA_HOME=/home/cloud/apps/jdk1.8.0_181
CLASSPATH=$JAVA_HOME/lib/
# 指定顺序，避免优先使用其他版本的JDK
PATH=$JAVA_HOME/bin:$PATH
export PATH JAVA_HOME CLASSPATH
```

```
# 刷新配置文件
[cloud@sit apps]$ source ~/.bash_profile
```

### 3. 验证

```
[cloud@sit apps]$ 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)
```

## 安装MYSQL

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf mysql-5.7.29.tar.gz && rm mysql-5.7.29.tar.gz &&
chmod -R 755 mysql-5.7.29/
```

### 2. 创建日志文件夹

```
[cloud@sit ~]$ mkdir -p /home/cloud/logs/mysql-5.7.29
```

### 3. 创建.cnf文件

```
[cloud@sit mysql-5.7.29]$ mkdir data
[cloud@sit mysql-5.7.29]$ vi mysql.cnf
```

```
# 添加以下内容
[client]
port=3306
socket=/home/cloud/apps/mysql-5.7.29/mysql.sock
[mysqld]
port=3306
# 务必在同一行，否则启动报错
sql_mode=STRICT_TRANS_TABLES,ERROR_FOR_DIVISION_BY_ZERO,NO_AUTO_CREATE_USER,NO_E
NGINE_SUBSTITUTION
init_connect='SET NAMES utf8'
basedir=/home/cloud/apps/mysql-5.7.29
datadir=/home/cloud/apps/mysql-5.7.29/data
pid-file=/home/cloud/apps/mysql-5.7.29/mysql.pid
socket=/home/cloud/apps/mysql-5.7.29/mysql.sock
log_error=/home/cloud/logs/mysql-5.7.29/error.log
collation_server=utf8_general_ci
character_set_server=utf8
log-bin=/home/cloud/logs/mysql-5.7.29/mysql-bin
server-id=100
lower_case_table_names=1
explicit_defaults_for_timestamp=ON

binlog_error_action=IGNORE_ERROR
innodb_flush_log_at_trx_commit=2
innodb_buffer_pool_size=9024M
```

```
innodb_log_files_in_group=4
innodb_log_file_size=1024M
innodb_page_cleaners=2
expire_logs_days=3
default-time_zone = '+8:00'
event_scheduler = 1
max_connections =3000
```

`sql_mode=......`务必在同一行，否则**启动报错。**

## 4. 安装

```
[cloud@sit bin]$ ./mysqld --defaults-file=/home/cloud/apps/mysql-5.7.29/mysql.cnf --initialize --user=cloud
```

此步骤会出现随机密码，记录此密码。

```
[cloud@sit ~]$ less /home/cloud/logs/mysql-5.7.29/error.log
```

```
2021-06-23T02:47:49.555673Z 1 [Note] A temporary password is generated for root@localhost: aaK=qrm:+43-
```

## 5. 启动

```
[cloud@sit bin]$ ./mysqld --defaults-file=/home/cloud/apps/mysql-5.7.29/mysql.cnf --user=cloud &
```

## 6. 建立软连接

```
[cloud@sit ~]$ ln -s /home/cloud/apps/mysql-5.7.29/mysql.sock /tmp/mysql.sock
```

## 7. MYSQL操作

```
# 登录,密码使用上述记录的临时密码。
[cloud@sit bin]$ ./mysql -u root -p
```

```
# 设置密码
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '123456' PASSWORD EXPIRE NEVER;
mysql> flush privileges;
```

```
# 赋权远程客户端链接
mysql> use mysql;
mysql> update user set host='%' where user ='root';
mysql> flush privileges;
```

```
# 密码永不过期
mysql> use mysql;
mysql> ALTER USER 'root'@'%' IDENTIFIED BY '123456' PASSWORD EXPIRE NEVER;
mysql> flush privileges;
```

```
# 修改最大链接
mysql> use mysql;
mysql> set global max_connect_errors=1000;
mysql> flush privileges;
```

```
# 退出
mysql> quit;
```

```
# 用户名/密码
用户名: root
密码: 123456
```

## MYSQL备份

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf mysql_backup.tar.gz && rm mysql_backup.tar.gz &&
chmod -R 755 mysql_backup/
```

### 2. 修改配置文件

- start.sh
- backup.sh

### 3. 定时备份

```
[cloud@sit ~]$ crontab -e
```

```
# 每分钟执行(测试使用)
*/1 * * * * /home/cloud/apps/mysql_backup/start.sh
```

```
# 每天凌晨2点执行一次
0 2 * * * /home/cloud/apps/mysql_backup/start.sh
```

## 安装REDIS

目前安装**单机版一主二从哨兵模式。**

### 1. 参考文档

https://blog.csdn.net/qq_40953197/article/details/108639539

### 2. 基本操作

```
[cloud@sit apps]$ tar -zxvf smart-redis.tar.gz && rm smart-redis.tar.gz && chmod
-R 755 smart-redis/
```

### 3. 修改配置文件

redis1、redis2、redis3同时修改。

- redis.conf

- sentinel.conf

### 4. 启动

redis1、redis2、redis3执行以下操作。

```
[cloud@sit redis-5.0.0]$ ./redis-server ./redis.conf
[cloud@sit redis-5.0.0]$ ./redis-sentinel ./sentinel.conf
```

### 5. 验证

```
[cloud@sit ~]$ ps -ef|grep redis
```

```
cloud      8836      1  0 08:00 ?        00:00:01 ./redis-server 127.0.0.1:6379
cloud      8842      1  0 08:00 ?        00:00:01 ./redis-sentinel
127.0.0.1:6380 [sentinel]
cloud      8863      1  0 08:01 ?        00:00:01 ./redis-server 127.0.0.1:6381
cloud      8869      1  0 08:01 ?        00:00:01 ./redis-sentinel
127.0.0.1:6382 [sentinel]
cloud      8877      1  0 08:01 ?        00:00:01 ./redis-server 127.0.0.1:6383
cloud      8883      1  0 08:01 ?        00:00:01 ./redis-sentinel
127.0.0.1:6384 [sentinel]
```

### 6. 常用命令

```
# 启动
[cloud@sit redis-5.0.0]$ ./redis-server ./redis.conf
# 关闭
[cloud@sit redis-5.0.0]$ ./redis-cli -h 127.0.0.1 -p 6379 shutdown
```

## 安装NGINX

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf nginxs.tar.gz && rm nginxs.tar.gz && chmod -R 755
nginxs/
```

### 2. 确认gcc/g++环境

```
[cloud@sit ~]$ whereis gcc
[cloud@sit ~]$ whereis g++
```

如果缺失gcc/g++环境，使用**root**用户登录并执行以下命令。

```
[root@sit ~]# cd /home/cloud/apps/nginxs/gcc
[root@sit gcc]# rpm -Uvh *.rpm --nodeps --force
[root@sit ~]# cd /home/cloud/apps/nginxs/gcc-c++/
[root@sit gcc-c++]# rpm -Uvh *.rpm --nodeps --force
```

### 3. 检查pcre

```
[cloud@sit ~]$ rpm -qa pcre
```

如果未安装pcre，则执行以下命令。

```
[cloud@sit ~]$ cd /home/cloud/apps/nginxs/pcre-8.32
[cloud@sit pcre-8.32]$ ./configure
[cloud@sit pcre-8.32]$ make
[cloud@sit pcre-8.32]$ make install
```

### 4. 检查zlib

```
[cloud@sit ~]$ rpm -qa zlib
```

如果未安装zlib，则执行以下命令。

```
[cloud@sit ~]$ cd /home/cloud/apps/nginxs/zlib-1.2.11
[cloud@sit zlib-1.2.11]$ ./configure
[cloud@sit zlib-1.2.11]$ make
[cloud@sit zlib-1.2.11]$ make install
```

### 5. 检查openssl

```
[cloud@sit ~]$ rpm -qa openssl
```

如果未安装openssl，则执行以下命令。

```
[cloud@sit ~]$ cd /home/cloud/apps/nginxs/openssl-1.0.2t
[cloud@sit openssl-1.0.2t]$ ./configure
[cloud@sit openssl-1.0.2t]$ make
[cloud@sit openssl-1.0.2t]$ make install
```

### 6. 安装nginx

```
[cloud@sit ~]$ cd /home/cloud/apps/nginxs/nginx-1.16.1
[cloud@sit nginx-1.16.1]$ ./configure --prefix=/home/cloud/apps/nginxs/nginx --
sbin-path=/home/cloud/apps/nginxs/nginx/sbin/nginx --conf-
path=/home/cloud/apps/nginxs/nginx/conf/nginx.conf --pid-
path=/home/cloud/apps/nginxs/nginx/nginx.pid --with-http_ssl_module --with-
pcre=/home/cloud/apps/nginxs/pcre-8.32 --with-zlib=/home/cloud/apps/nginxs/zlib-
1.2.11 --with-openssl=/home/cloud/apps/nginxs/openssl-1.0.2t --add-
module=/home/cloud/apps/nginxs/ngx-fancyindex-0.5.1 --with-stream
[cloud@sit nginx-1.16.1]$ make
[cloud@sit nginx-1.16.1]$ make install
```

### 7. 安装注意事项

- ./configure --prefix...后操作报错

```
Operating system: x86_64-whatever-linux2 You need Perl 5.
```

```
# 更新yum源
[root@sit ~]# yum install perl
```

- yum install perl操作后报错

```
# 检查报错
[root@sit ~]# journalctl -xe
```

```
Aug 18 08:17:02 sit.com sshd[30308]: /usr/sbin/sshd: /lib64/libcrypto.so.10:
version `OPENSSL_1.0.2' not found (required by /usr/sbin/sshd)
```

```
# 更新yum源
[root@sit ~]# yum -y install openssl
```

### 8. 修改配置文件

- nginx.conf

### 9. 启动

```
[cloud@sit sbin]$ ./nginx
```

### 10. 验证

```
[cloud@sit ~]$ cd /home/cloud/apps/nginxs/nginx/sbin/
[cloud@sit sbin]$ ./nginx -v
```

浏览器访问http://ip:1080/。

### 11. 常用命令

```
#启动
[cloud@sit sbin]$ ./nginx
#停止
[cloud@sit sbin]$ ./nginx -s stop
#热部署
[cloud@sit sbin]$ ./nginx -s reload
#检查配置文件
[cloud@sit sbin]$ ./nginx -t
#查看版本
[cloud@sit sbin]$ ./nginx -v
```

## 安装KEEPALIVED

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf smart-keepalived.tar.gz && rm smart-
keepalived.tar.gz && chmod -R 755 smart-keepalived/
```

### 2. 安装

```
[cloud@sit keepalived-2.2.4]$ ./configure --prefix=/home/cloud/apps/smart-
keepalived/keepalived
[cloud@sit keepalived-2.2.4]$ make
# 不加sudo会报错
[cloud@sit keepalived-2.2.4]$ sudo make install
```

### 3. 注意事项

- OpenSSL is not properly installed on your system

```
# root用户执行
[root@sit openssl]$ rpm -Uvh *.rpm --force --nodeps
```

### 4. 修改配置文件

上传配置文件heartbeat.sh和keepalived.conf.master(keepalived.conf.backup)。
keepalived.conf.master(keepalived.conf.backup)重命名去掉尾缀。

- keepalived.conf (/home/cloud/apps/smart-keepalived/keepalived/etc/keepalived/keepalived.conf)

### 5. 启动

```
# 修改文件权限必须为644(/home/cloud/apps/smart-
keepalived/keepalived/etc/keepalived)
[cloud@sit keepalived]# chmod 644 keepalived.conf

# 修改文件权限为755
[cloud@sit keepalived]# chmod 755 heartbeat.sh

# 启动(绝对路径,需要sudo)
[cloud@sit sbin]$ sudo /home/cloud/apps/smart-
keepalived/keepalived/sbin/keepalived -f /home/cloud/apps/smart-
keepalived/keepalived/etc/keepalived/keepalived.conf
```

```
# 目前keepalived脚本执行失败，暂时用crontab -e代替(每隔5s执行一次)

*/1 * * * * sleep 5 && /home/cloud/apps/smart-
keepalived/keepalived/etc/keepalived/heartbeat.sh
```

### 6. 测试结果

```
[cloud@sit ~]$ ip addr
```

### 7. 注意事项

- 主机和备机Keepalived.conf配置中virtual_router_id、authentication、virtual_ipaddress都要一样。
- virtual_ipaddress配置的虚拟ip需和实际物理ip需在同一个网段，如实际物理ip是192.168.21.11，虚拟ip可配置为192.168.21.100。
- 主机和备机区别在于state、interface节点配置不一样，主机(state MASTER)、备机(state BACKUP)、interface(网卡名称本人两台网卡都一致)、其余都一样。

## 安装nacos

### 1. 创建database

```
create database nacos_config
```

然后导入**nacos-mysql.sql**。

### 2. 基本操作

```
[cloud@sit apps]$ tar -zxvf nacos-1.3.1.tar.gz && rm nacos-1.3.1.tar.gz && chmod
-R 755 nacos-1.3.1/
```

### 3. 修改配置文件

- application.properties
- cluster.conf

### 4. 启动

```
[cloud@sit bin]$ sh startup.sh
```

### 5. 验证

```
[cloud@sit ~]$ lsof -i:8848
```

然后，nginx配置集群。

### 6. 登录

单机: http://ip1:8848/nacos/#/login,http://ip2:8848/nacos/#/login,http://ip3:8848/nacos/#/login

用户名/密码: nacos/nacos

### 7. 注意

自定义服务如果添加nacos作为**服务发现**或者**配置中心时**脚本最好添加如下参数。

```
# nacos配置信息缓存目录
NACOS_OPTS="-DJM.SNAPSHOT.PATH=/home/cloud/logs/nacos-1.3.1 -
Dcom.alibaba.nacos.naming.cache.dir=/home/cloud/logs/nacos-1.3.1"
```

### 8. 系统参数

```
https://www.jianshu.com/p/7b065cd688bc
```

## 安装smart-cloud-gateway

### 1. 基本操作

```
[cloud@sit smart-cloud]$ tar -zxvf smart-cloud-gateway.tar.gz && rm smart-cloud-
gateway.tar.gz && chmod -R 755 smart-cloud-gateway/
```

### 2. 启动

```
[cloud@sit smart-cloud-gateway]$ sh deploy.sh start
```

### 3. 修改配置文件

- bootstrap.yml
- nacos.config

- sentinel.config

### 4. 验证

```
[cloud@sit smart-cloud-gateway]$ lsof -i:9999
```

## 安装smart-cloud-demo

### 1. 基本操作

```
[cloud@sit smart-cloud]$ tar -zxvf smart-cloud-demo.tar.gz && rm smart-cloud-demo.tar.gz && chmod -R 755 smart-cloud-demo/
```

### 2. 启动

```
[cloud@sit smart-cloud-demo]$ sh deploy.sh start
```

### 3. 修改配置文件

- application.yml

### 4. 验证

```
[cloud@sit ~]$ lsof -i:7001
```

## 安装sentinel-dashboard[废弃]

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf sentinel-dashboard-1.7.2.tar.gz && rm sentinel-dashboard-1.7.2.tar.gz && chmod -R 755 sentinel-dashboard-1.7.2/
```

### 2. 启动

```
[cloud@sit sentinel-dashboard-1.7.2]$ sh deploy.sh start
```

```
# 启动异常
...deploy.sh: line 55: /home/cloud/logs/sentinel-dashboard-1.7.2/out.log: No such file or directory
```

```
[cloud@sit ~]$ mkdir -p /home/cloud/logs/sentinel-dashboard-1.7.2
```

### 3. 验证

```
[cloud@sit ~]$ lsof -i:8718
[cloud@sit ~]$ lsof -i:8719
```

### 4. 登录

http://192.168.2.121:8718/

用户名/密码: sentinel/sentinel

## 安装sentinel-dashboard-nacos

### 1. 参考

```
https://blog.visionki.com/index.php/archives/101/
```

### 2. 基本操作

```
[cloud@sit apps]$ tar -zxvf sentinel-dashboard-nacos-1.8.0.tar.gz && rm
sentinel-dashboard-nacos-1.8.0.tar.gz && chmod -R 755 sentinel-dashboard-nacos-
1.8.0/
```

### 3. 修改配置文件

- application.properties
- nacos.config
- sentinel.config

### 4. 启动

```
[cloud@sit sentinel-dashboard-1.7.2]$ sh deploy.sh start
```

```
...deploy.sh: line 61: /home/cloud/logs/sentinel-dashboard-nacos-1.8.0/out.log:
No such file or directory
```

```
[cloud@sit ~]$ mkdir -p /home/cloud/logs/sentinel-dashboard-nacos-1.8.0
```

### 5. 验证

```
[cloud@sit ~]$ lsof -i:8718
[cloud@sit ~]$ lsof -i:8719
```

### 6. 登录

http://192.168.2.121:8718/

用户名/密码: sentinel/sentinel

## 安装smart-monitor

区分prometheus安装服务器和非prometheus安装服务器。所有非prometheus安装服务器找到相应压缩包运行node_exporter。

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf smart-monitor.tar.gz && rm smart-monitor.tar.gz &&
chmod -R 755 smart-monitor/
```

### 2. 安装node_exporter

#### 2.1 启动

```
[cloud@sit node_exporter-0.18.1]$ sh start.sh
```

#### 2.2 验证

```
[cloud@sit ~]$ lsof -i:9100
```

## 3. 安装promethues

#### 3.1 修改配置文件

- prometheus.yml
- 上传gateway文件夹
- 上传node_exporter文件夹

#### 3.2 启动

```
[cloud@sit prometheus-2.6.0]$ sh start.sh
```

#### 3.3 验证

```
[cloud@sit ~]$ lsof -i:9090
```

## 4. 安装kafka_exporter

#### 4.1 基本操作

```
[cloud@sit smart-monitor]$ tar -zxvf kafka_exporter-1.2.0.tar.gz && rm
kafka_exporter-1.2.0.tar.gz && chmod -R 755 kafka_exporter-1.2.0/
```

#### 4.2 启动

```
[cloud@sit kafka_exporter-1.2.0]$ nohup ./kafka_exporter --
kafka.server=123.56.9.196:9092 >out.log 2>&1 &
```

#### 4.3 验证

```
[cloud@sit ~]$ lsof -i:9308
```

## 5. 安装redis_exporter

#### 5.1 基本操作

```
[cloud@sit smart-monitor]$ tar -zxvf redis_exporter-1.8.0.tar.gz && rm
redis_exporter-1.8.0.tar.gz && chmod -R 755 redis_exporter-1.8.0/
```

#### 5.2 启动

```
[cloud@sit kafka_exporter-1.2.0]$ sh start.sh
```

#### 5.3 验证

```
[cloud@sit ~]$ lsof -i:9121
```

### 6. 安装prometheus-webhook-dingtalk-0.3.0

#### 6.1 基本操作

```
[cloud@sit smart-monitor]$ tar -zxvf prometheus-webhook-dingtalk-0.3.0.tar.gz && rm prometheus-webhook-dingtalk-0.3.0.tar.gz && chmod -R 755 prometheus-webhook-dingtalk-0.3.0/
```

#### 6.2 启动

```
nohup ./prometheus-webhook-dingtalk --ding.profile="ops_dingding=https://oapi.dingtalk.com/robot/send?access_token=efb5c0c28638be82f7da06f8ff1ee5d7f434bfb41bf05904c63cc53df06230c7" 2>&1 >out.log &
```

## 安装grafana

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf grafana-7.1.5.tar.gz && rm grafana-7.1.5.tar.gz && chmod -R 775 grafana-7.1.5/
```

### 2. 启动

```
[cloud@sit bin]$ sh start.sh
```

### 3. 验证

```
[cloud@sit ~]$ lsof -i:3000
```

浏览器访问http://IP:3000/。

用户名/密码: admin/admin

### 4. 添加面板

以admin/admin用户登录后，配置prometheus数据源，导入JSON模板。

## 安装FRP

### 1. 服务端[LINUX版]

#### 1.1 基本操作

```
[cloud@sit apps]$ tar -zxvf frp_0.30.0_linux_amd64.tar.gz && rm frp_0.30.0_linux_amd64.tar.gz && chmod -R 755 frp_0.30.0_linux_amd64/
```

#### 1.2 修改配置文件

- frps.ini(服务端)
- frpc.ini(客户端)

#### 1.3 启动

```
# 服务端
[cloud@sit frp_0.30.0_linux_amd64]$ nohup ./frps -c frps.ini >./out.log 2>&1 &
```

```
# 客户端
[cloud@sit frp_0.30.0_linux_amd64]$ nohup ./frpc -c frpc.ini >./out.log 2>&1 &
```

### 2 客户端[WINDOW版]

#### 2.1 修改配置文件

- frpc.ini

#### 2.2 启动

```
D:frp_0.30.0_windows_amd64> ./frpc.exe -c frpc.ini
```

## 安装Elasticsearch

### 1. 设置NFS共享目录

略

### 2. 创建共享目录

略

### 3. 基本操作

```
[cloud@sit apps]$ tar -zxvf elasticsearch-6.7.1.tar.gz && rm elasticsearch-6.7.1.tar.gz && chmod -R 755 elasticsearch-6.7.1/
```

### 4. 修改配置文件

- elasticsearch.yml

### 5. 启动

```
[cloud@sit bin]$ sh start.sh
```

### 6. 验证

```
[cloud@sit ~]$ lsof -i:9200
[cloud@sit ~]$ lsof -i:9300
```

es-head连接http://ip:9200/。

### 7. 建立快照

略

## 安装Elasticsearch7.14.0(单机版)

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf elasticsearch-7.14.0.tar.gz && rm -rf elasticsearch-7.14.0.tar.gz && chmod -R 755 elasticsearch-7.14.0/
```

## 2. 修改配置文件

- elasticsearch.yml
- jvm.options

## 3. 设置JDK版本

```
Future versions of Elasticsearch will require Java 11; your Java version from......
```

```
[cloud@sit bin]$ vim elasticsearch-env
```

```
......
ES_CLASSPATH="$ES_HOME/lib/*"

# 设置JDK版本
ES_JAVA_HOME="/app/elasticsearch-7.14.0/jdk"

# now set the path to java
......
```

## 4. 设置密码

```
# 配置文件中设置如下配置内容(如果是第一次配置，配置完之后重启ES)
[cloud@sit config]$ vim elasticsearch.yml
```

```
xpack.security.enabled: true
xpack.license.self_generated.type: basic
xpack.security.transport.ssl.enabled: true
http.cors.allow-methods: OPTIONS, HEAD, GET, POST, PUT, DELETE
# 不添加无法使用head连接es，连接时在http:ip:port/?auth_user=elastic&auth_password=密码
http.cors.allow-headers: Content-Type,Accept,Authorization,x-requested-with
```

```
# 后台启动(kill -9 PID)
[cloud@sit bin]$ ./elasticsearch -d
```

```
# 重启ES之后，执行如下命令
[cloud@sit bin]$ ./elasticsearch-setup-passwords interactive
```

```
Enter password for [elastic]:
Reenter password for [elastic]:
Enter password for [kibana]:
Reenter password for [kibana]:
Enter password for [logstash_system]:
Reenter password for [logstash_system]:
Enter password for [beats_system]:
Reenter password for [beats_system]:
Changed password for user [kibana]
Changed password for user [logstash_system]
Changed password for user [beats_system]
Changed password for user [elastic]
```

集群设置密码参考: es的集群加密码

### 5. 登录

es-head登录。

## 安装MAVEN

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf apache-maven-3.6.0.tar.gz && rm apache-maven-3.6.0.tar.gz && chmod -R 755 apache-maven-3.6.0/
```

### 2. 修改配置文件

- settings.xml

```
修改<localRepository>路径
e.g:
/home/cloud/apps/apache-maven-3.6.0/repository
```

修改完配置文件后可以手动上传本地仓库包至/home/cloud/apps/apache-maven-3.6.0/repository目录下。

### 3. 配置环境变量

```
[cloud@sit apps]$ vi ~/.bash_profile
```

```
# maven变量
export M2_HOME=/home/cloud/apps/apache-maven-3.6.0
export PATH=$PATH:$M2_HOME/bin
```

```
[cloud@sit apps]$ source ~/.bash_profile
```

```
[cloud@sit apps]$ mvn -v
```

```
Apache Maven 3.6.0 (97c98ec64a1fdfee7767ce5ffb20918da4f719f3; 2018-10-
25T02:41:47+08:00)
Maven home: /home/cloud/apps/apache-maven-3.6.0
Java version: 1.8.0_181, vendor: Oracle Corporation, runtime:
/home/cloud/apps/jdk1.8.0_181/jre
Default locale: en_US, platform encoding: UTF-8
OS name: "linux", version: "3.10.0-1127.el7.x86_64", arch: "amd64", family:
"unix"
```

## 安装nodeJS

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf node-v12.18.2-linux-x64.tar.gz && rm node-v12.18.2-
linux-x64.tar.gz && chmod -R 755 node-v12.18.2-linux-x64/
```

### 2. 配置环境变量

```
[cloud@sit apps]$ vi ~/.bash_profile
```

```
# nodejs
export PATH=$PATH:/home/cloud/apps/node-v12.18.2-linux-x64/bin
```

```
[cloud@sit apps]$ source ~/.bash_profile
```

```
[cloud@sit apps]$ node -v
```

```
v12.18.2
```

## 安装Git

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf smart-git.tar.gz && rm smart-git.tar.gz && chmod -R
755 smart-git/
```

### 2. 注意事项

```
# 编译git时报错：zlib.h: No such file or directory
[cloud@sit apps]$ yum install zlib-devel
# 编译git时报错：BEGIN failed--compilation aborted at Makefile.PL line 3.
[cloud@sit apps]$ yum -y install perl-devel
```

### 3. 编译安装

```
[cloud@sit git-1.8.3.1]$ ./configure --prefix=/home/cloud/apps/smart-git/git
[cloud@sit git-1.8.3.1]$ make
[cloud@sit git-1.8.3.1]$ make install
```

### 4. 配置环境变量

```
[cloud@sit apps]$ vi ~/.bash_profile
```

```
# 文件末尾添加
export PATH=$PATH:/home/cloud/apps/smart-git/git/bin
```

```
[cloud@sit apps]$ source ~/.bash_profile
```

### 5. 验证

```
[cloud@sit ~]$ git --version
```

```
git version 1.8.3.1
```

### 6. 配置git

```
[cloud@sit ~]$ sudo yum install -y git
[cloud@sit ~]$ git config --list
[cloud@sit ~]$ git config --global --unset http.proxy
[cloud@sit ~]$ git config --global --unset https.proxy
[cloud@sit5 test]$ sudo git config --system --unset credential.helper
[cloud@sit ~]$ git config --global user.name "songning123456"
[cloud@sit ~]$ git config --global user.email 1457065857@qq.com
```

### 7. 生成公私钥(全部默认回车)

```
[cloud@sit ~]$ ssh-keygen -t rsa
[cloud@sit ~]$ cd ~/.ssh
id_rsa: 私钥
id_rsa.pub: 公钥
拷贝id_rsa.pub至github/gitlab => Setttings => SSH and GPG keys => New SSH key

# Git SSH配置无误，但无法连接github远程仓库“Host Key Verification Failed”
[cloud@sit ~]$ ssh-keyscan -H gitee.com >> ~/.ssh/known_hosts
[cloud@sit ~]$ ssh-keyscan -H github.com >> ~/.ssh/known_hosts
```

## 安装Tomcat

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf apache-tomcat-8.5.78.tar.gz && rm apache-tomcat-8.5.78.tar.gz && chmod -R 755 apache-tomcat-8.5.78/
```

### 2. 修改配置文件

- server.xml

```
# 修改端口号，解决sk-nodeJS项目npm run build时8888端口启动卡死jenkins运行问题
<Connector port="8888" protocol="HTTP/1.1"
              connectionTimeout="20000"
              redirectPort="8443" />
......
```

## 安装jenkins

务必在安装jenkins的机器上安装好**JDK、MAVEN、Git、nodeJS、Tomcat。**

### 1. 复制war包，修改主目录

将jenkins.war上传复制到Tomcat => webapps 文件夹下。

```
[cloud@sit apps]$ vi ~/.bash_profile
```

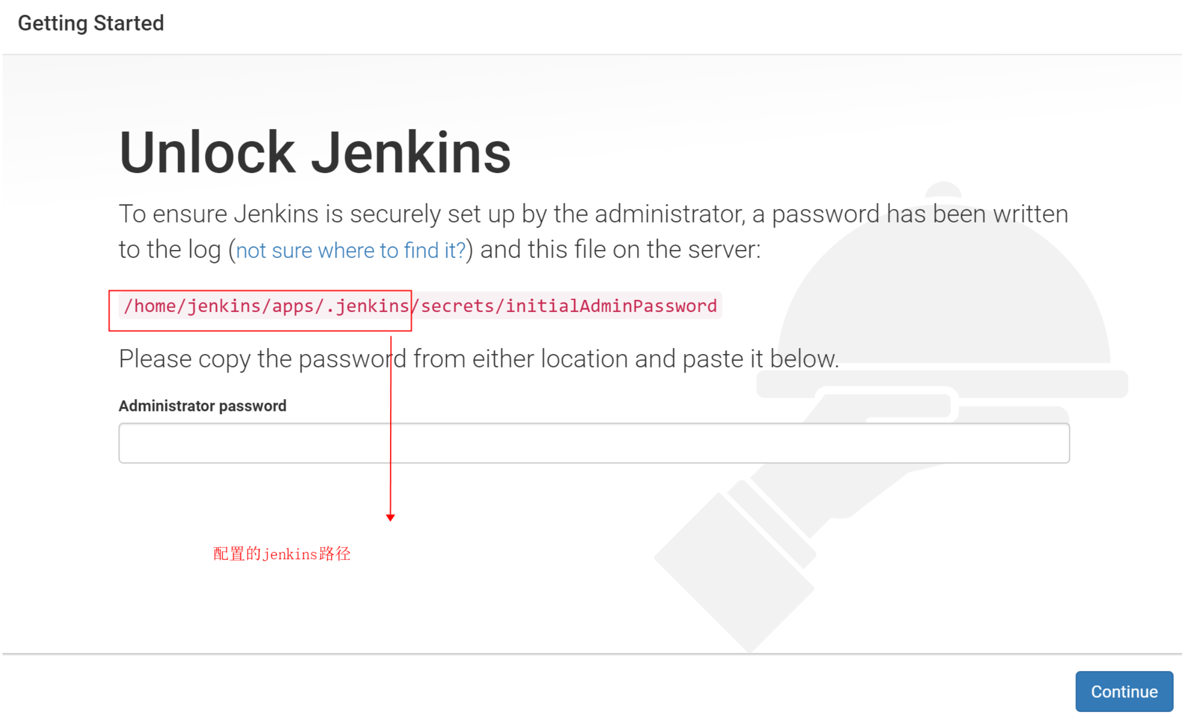
```
export JENKINS_HOME=/home/cloud/apps/.jenkins
```

```
[cloud@sit apps]$ source ~/.bash_profile
```

### 2. 启动Tomcat

```
[cloud@sit bin]$ sh startup.sh
```

```
# 访问
http://ip:8888/jenkins
常用命令
http://localhost:8888/jenkins/restart
http://localhost:8888/jenkins/reload
```


Getting Started

Unlock Jenkins

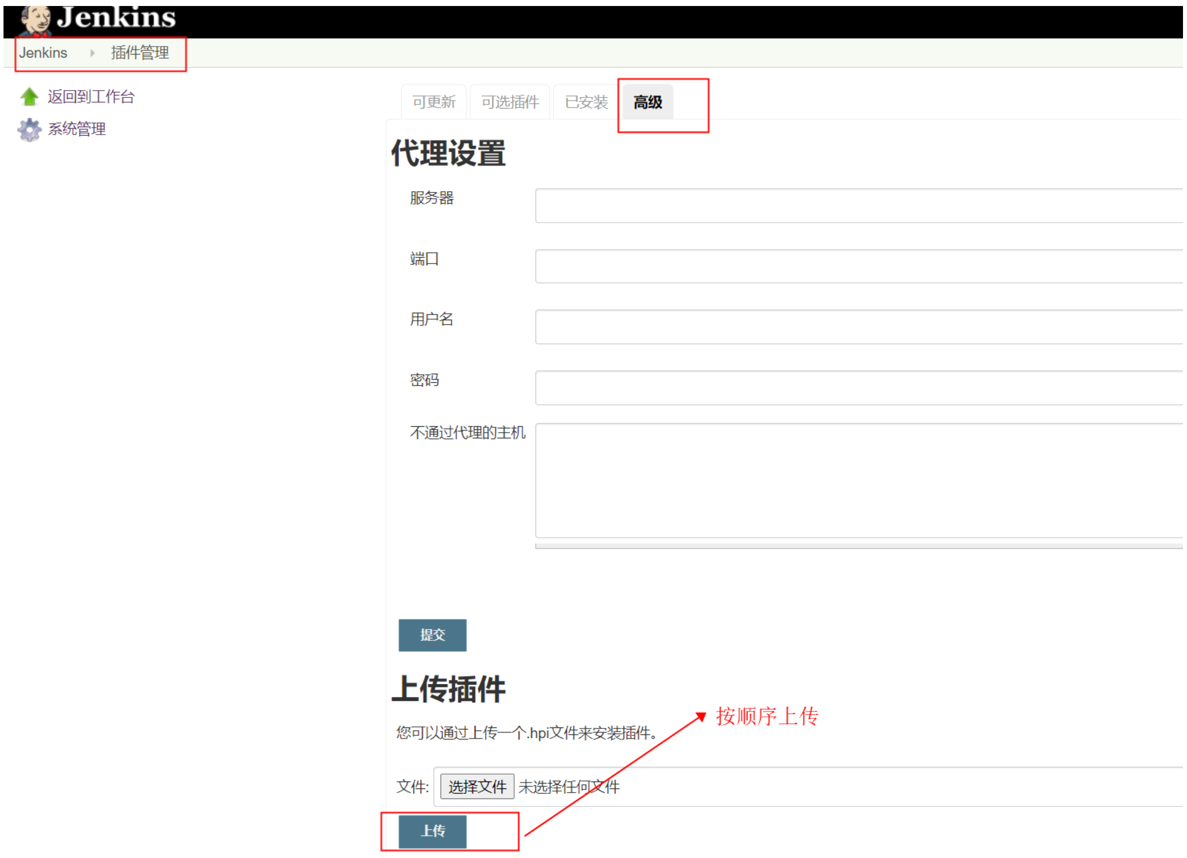
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

```
/home/jenkins/apps/.jenkins/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

### 3. 安装jenkins插件


Jenkins › 插件管理

返回到工作台
系统管理

可更新 | 可选插件 | 已安装 | 高级

代理设置

| | |
|---|---|
| 服务器 | |
| 端口 | |
| 用户名 | |
| 密码 | |
| 不通过代理的主机 | |

提交

上传插件

您可以通过上传一个.hpi文件来安装插件。

文件: 选择文件 未选择任何文件

上传

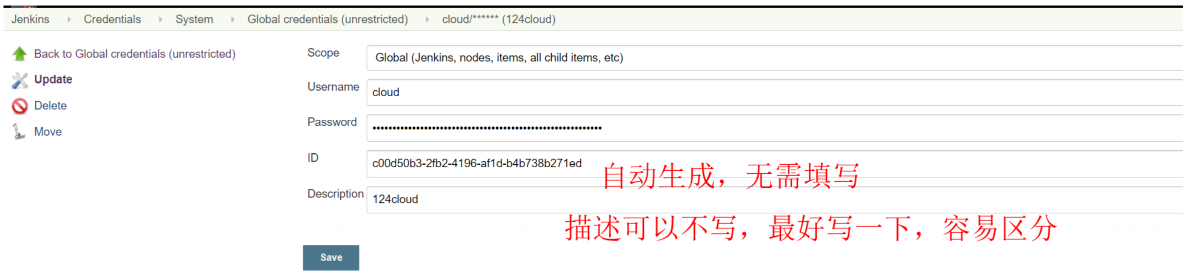
按顺序上传

```
1. Git plugin(支持Git的插件)
2. Maven Integration plugin(构建Maven项目所需要的插件，安装后在创建新项目时可以选择构建
Maven项目)
3. publish over SSH(SSH连接服务器，让项目可以发布在其他服务器上)
```

### 3.1 添加Credentials

#### 3.1.1 访问应用服务器

需要访问的应用服务器的用户名、密码Credentials


Jenkins › Credentials › System › Global credentials (unrestricted) › cloud/****** (124cloud)

Back to Global credentials (unrestricted)
Update
Delete
Move

| | |
|---|---|
| Scope | Global (Jenkins, nodes, items, all child items, etc) |
| Username | cloud |
| Password | •••••••••••••••••••••••••••••••••••••••••••••••••••••••••• |
| ID | c00d50b3-2fb2-4196-af1d-b4b738b271ed |
| Description | 124cloud |

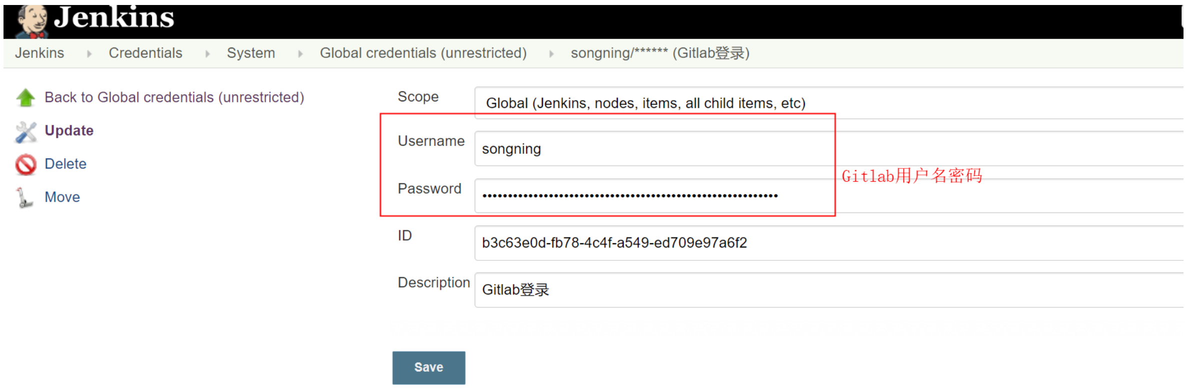
Save

自动生成，无需填写

描述可以不写，最好写一下，容易区分

#### 3.1.2 访问Gitlab-用户名/密码类型

如果是从GitLab拉取代码，还要添加GitLab的用户名、密码Credentials

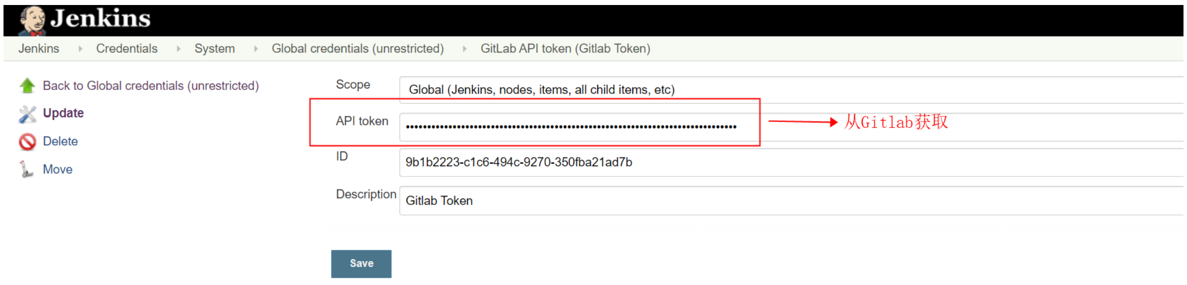

Jenkins › Credentials › System › Global credentials (unrestricted) › songning/****** (Gitlab登录)

Back to Global credentials (unrestricted)
Update
Delete
Move

| | |
|---|---|
| Scope | Global (Jenkins, nodes, items, all child items, etc) |
| Username | songning |
| Password | •••••••••••••••••••••••••••••••••••••••••••••••••• |
| ID | b3c63e0d-fb78-4c4f-a549-ed709e97a6f2 |
| Description | Gitlab登录 |

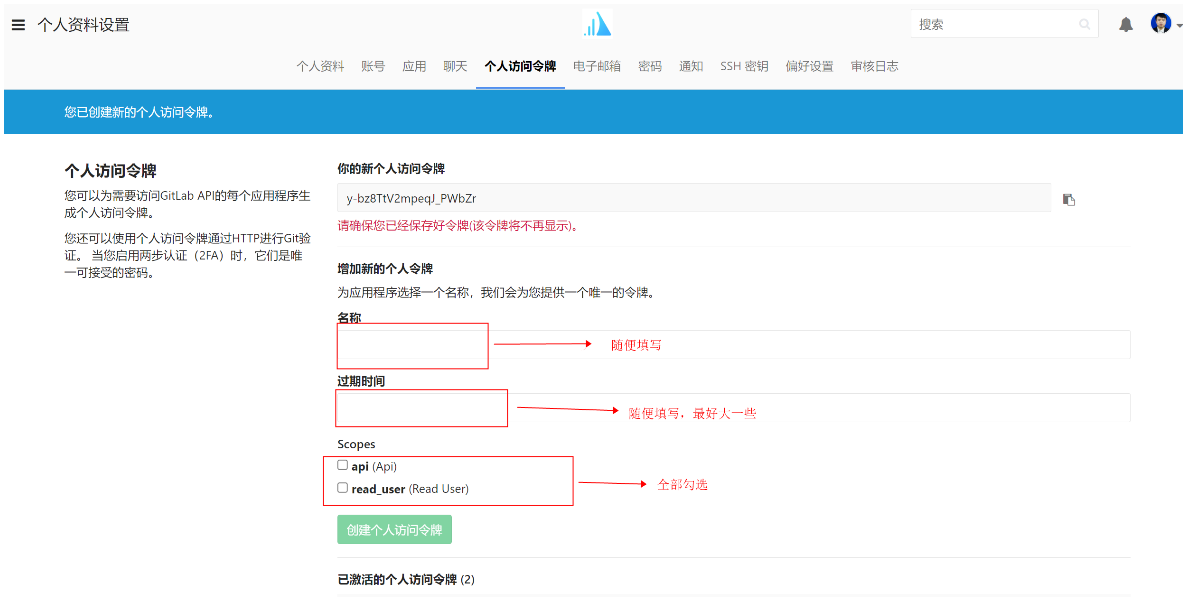
### 3.1.3 访问Gitlab-APIToken类型

如果是从GitLab拉取代码，还要添加GitLab API Token Credentials。（Github和Gitee不需要）


Jenkins › Credentials › System › Global credentials (unrestricted) › GitLab API token (Gitlab Token)

Back to Global credentials (unrestricted)
Update
Delete
Move

| | |
|---|---|
| Scope | Global (Jenkins, nodes, items, all child items, etc) |
| API token | •••••••••••••••••••••••••••••••••••••••••••••••••••• |
| ID | 9b1b2223-c1c6-494c-9270-350fba21ad7b |
| Description | Gitlab Token |



个人资料设置
搜索
个人资料 账号 应用 聊天 个人访问令牌 电子邮箱 密码 通知 SSH 密钥 偏好设置 审核日志

您已创建新的个人访问令牌。

**个人访问令牌**

您可以为需要访问GitLab API的每个应用程序生成个人访问令牌。

您还可以使用个人访问令牌通过HTTP进行Git验证。当您启用两步认证（2FA）时，它们是唯一可接受的密码。

**你的新个人访问令牌**

y-bz8TtV2mpeqJ_PWbZr

请确保您已经保存好令牌(该令牌将不再显示)。

**增加新的个人令牌**

为应用程序选择一个名称，我们会为您提供一个唯一的令牌。

名称

过期时间

Scopes

☐ **api** (Api)
☐ **read_user** (Read User)

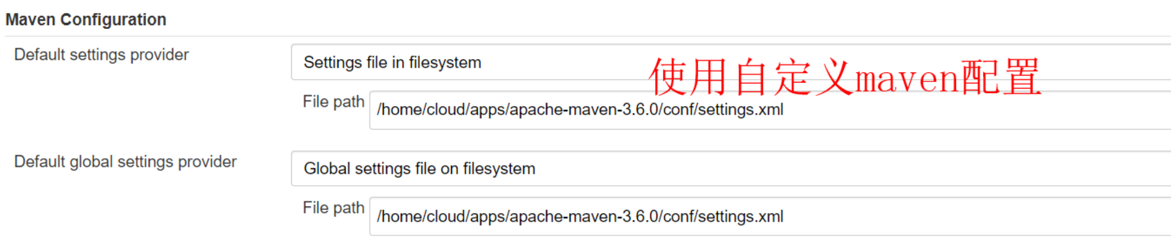
随便填写
随便填写，最好大一些
全部勾选
创建个人访问令牌

**已激活的个人访问令牌** (2)

## 3.2 系统管理-全局工具配置

### 3.2.1 Maven Configuration

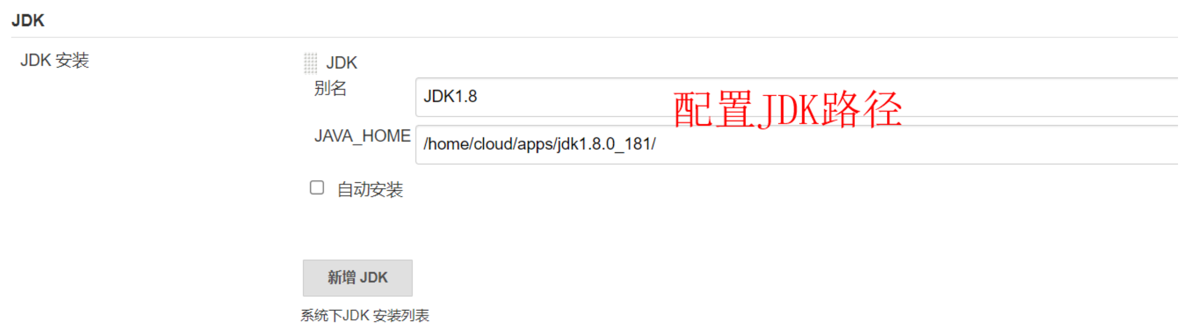
```
/home/cloud/apps/apache-maven-3.6.0/conf/settings.xml
```


**Maven Configuration**

| | |
|---|---|
| Default settings provider | Settings file in filesystem |
| File path | /home/cloud/apps/apache-maven-3.6.0/conf/settings.xml |
| Default global settings provider | Global settings file on filesystem |
| File path | /home/cloud/apps/apache-maven-3.6.0/conf/settings.xml |

### 3.2.2 JDK

```
JDK1.8
/home/cloud/apps/jdk1.8.0_181
```


JDK

JDK 安装

JDK

别名: JDK1.8

JAVA_HOME: /home/cloud/apps/jdk1.8.0_181/

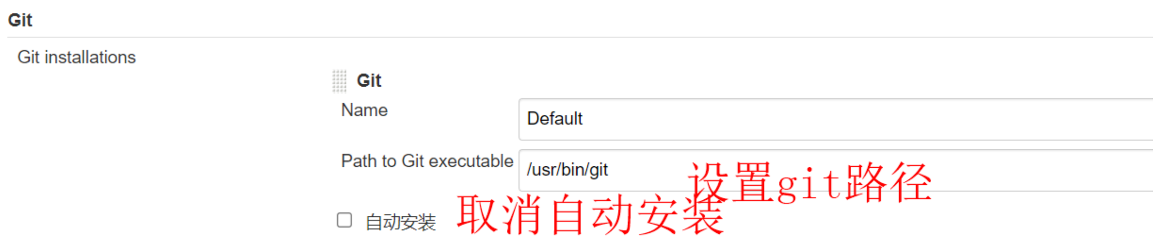
☐ 自动安装

新增 JDK

系统下JDK 安装列表

### 3.2.3 Git

```
Default
/usr/bin/git
```


Git

Git installations

Git

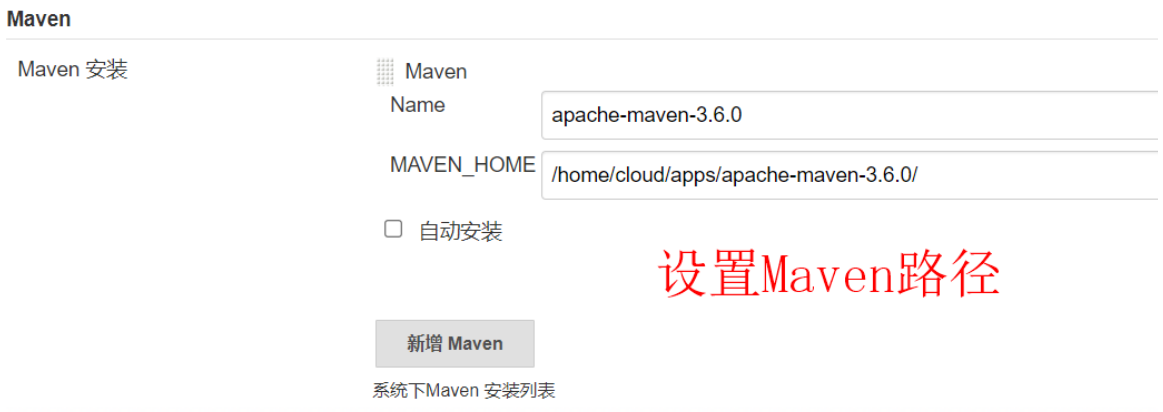
Name: Default

Path to Git executable: /usr/bin/git

☐ 自动安装

### 3.2.4 Maven

```
apache-maven-3.6.0
/home/cloud/apps/apache-maven-3.6.0
```


Maven

Maven 安装

Maven

Name: apache-maven-3.6.0

MAVEN_HOME: /home/cloud/apps/apache-maven-3.6.0/

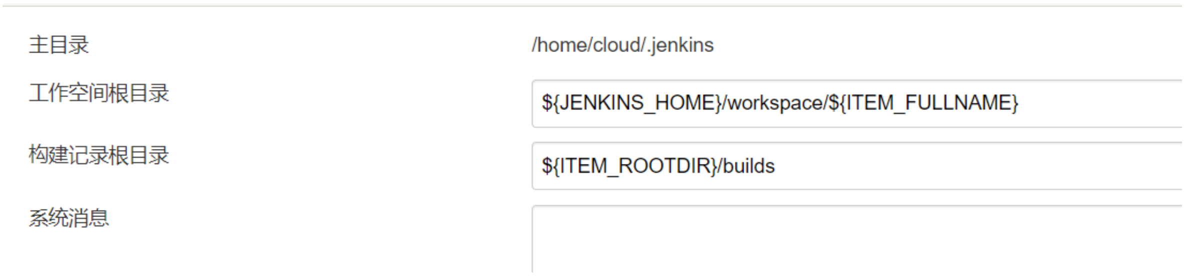
☐ 自动安装

新增 Maven

系统下Maven 安装列表

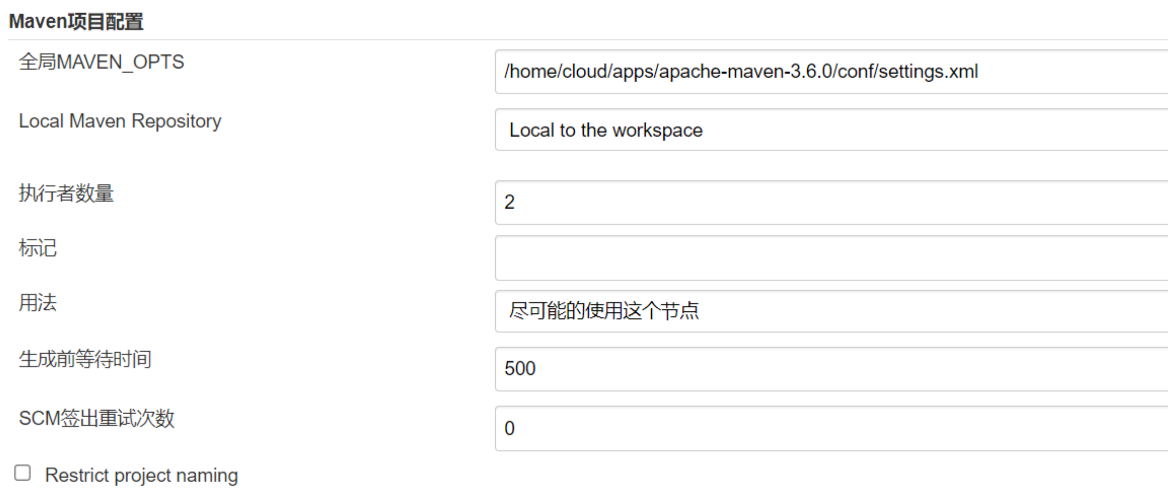
## 3.3 系统管理-系统设置

### 3.3.1 主目录


| | |
|---|---|
| 主目录 | /home/cloud/.jenkins |
| 工作空间根目录 | ${JENKINS_HOME}/workspace/${ITEM_FULLNAME} |
| 构建记录根目录 | ${ITEM_ROOTDIR}/builds |
| 系统消息 | |

### 3.3.2 Maven项目配置

```
/home/cloud/apps/apache-maven-3.6.0/settings.xml
```


**Maven项目配置**

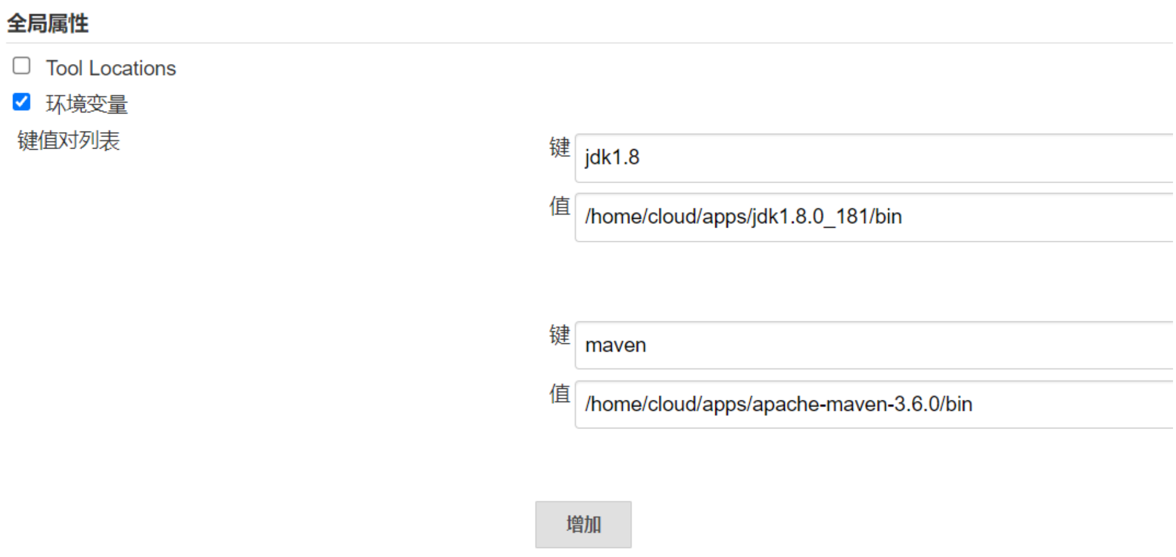
| | |
|---|---|
| 全局MAVEN_OPTS | /home/cloud/apps/apache-maven-3.6.0/conf/settings.xml |
| Local Maven Repository | Local to the workspace |
| 执行者数量 | 2 |
| 标记 | |
| 用法 | 尽可能的使用这个节点 |
| 生成前等待时间 | 500 |
| SCM签出重试次数 | 0 |

☐ Restrict project naming

### 3.3.3 全局属性

```
JDK1.8
/home/cloud/apps/jdk1.8.0_181/bin
```

```
apache-maven-3.6.0
/home/cloud/apps/apache-maven-3.6.0/bin
```


**全局属性**

☐ Tool Locations

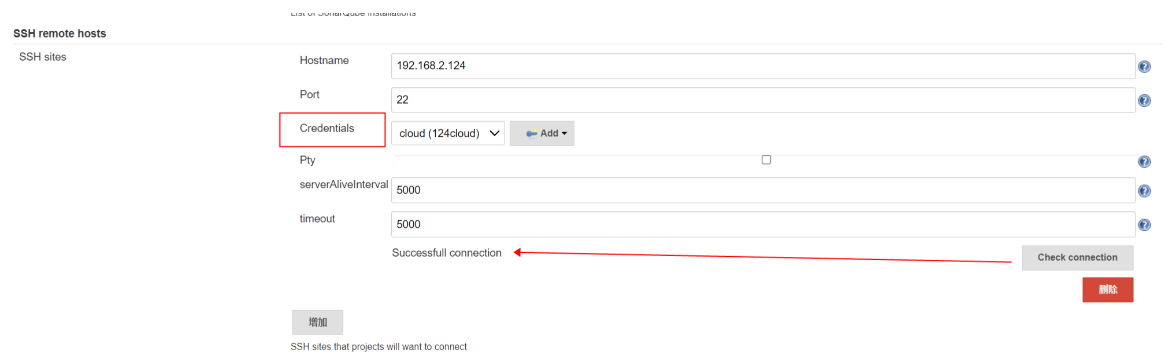
☑ 环境变量

键值对列表

| | |
|---|---|
| 键 | jdk1.8 |
| 值 | /home/cloud/apps/jdk1.8.0_181/bin |
| 键 | maven |
| 值 | /home/cloud/apps/apache-maven-3.6.0/bin |

### 3.3.4 SSH remote hosts

Credentials选择填写的用户名/密码类型的Credentials。


List of SonarQube installations

**SSH remote hosts**

SSH sites

Hostname: 192.168.2.124

Port: 22

Credentials: cloud (124cloud) Add

Pty: ☐

serverAliveInterval: 5000

timeout: 5000

Successfull connection

Check connection

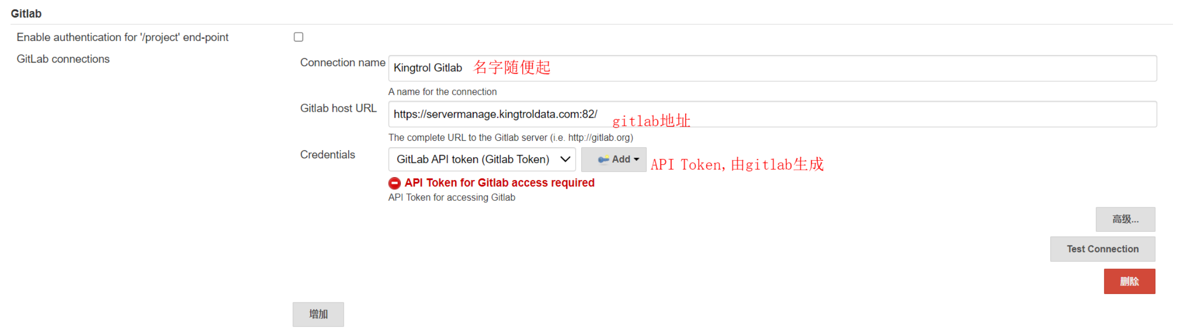
删除

增加

SSH sites that projects will want to connect

## 3.3.5 Gitlab

Test Connection出现的ERROR忽略。


**Gitlab**

Enable authentication for '/project' end-point ☐

GitLab connections

Connection name: Kingtrol Gitlab 名字随便起

A name for the connection

Gitlab host URL: https://servermanage.kingtroldata.com:82/ gitlab地址

The complete URL to the Gitlab server (i.e. http://gitlab.org)

Credentials: GitLab API token (Gitlab Token) Add API Token，由gitlab生成

API Token for Gitlab access required

API Token for accessing Gitlab

高级...

Test Connection

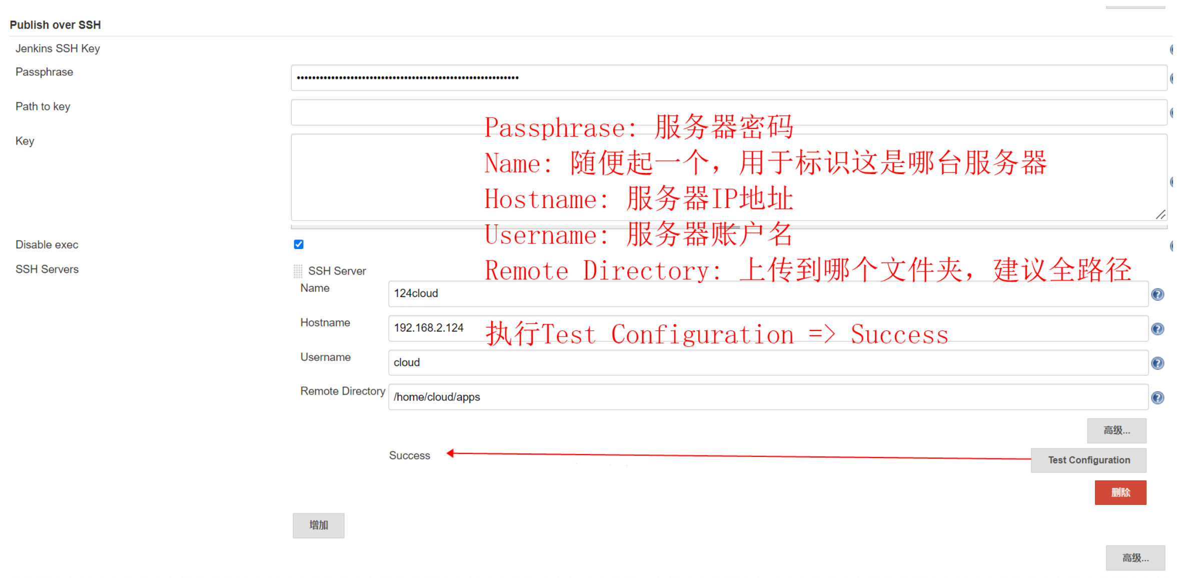
删除

增加

```
如何生成Gitlab API Token，参考：
https://www.jianshu.com/p/26ab9a70bd31
https://zhuanlan.zhihu.com/p/367287537
```

## 3.3.6 Publish over SSH

![](E:\Project\Repo_Cloud\000-md-images\jenkins\Git plugin.png


**Publish over SSH**

Jenkins SSH Key

Passphrase: ••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••

Path to key

Key

Passphrase：服务器密码

Name：随便起一个，用于标识这是哪台服务器

Hostname：服务器IP地址

Username：服务器账户名

Remote Directory：上传到哪个文件夹，建议全路径

执行Test Configuration => Success

Disable exec ☑

SSH Servers

SSH Server

Name: 124cloud

Hostname: 192.168.2.124

Username: cloud

Remote Directory: /home/cloud/apps

高级...

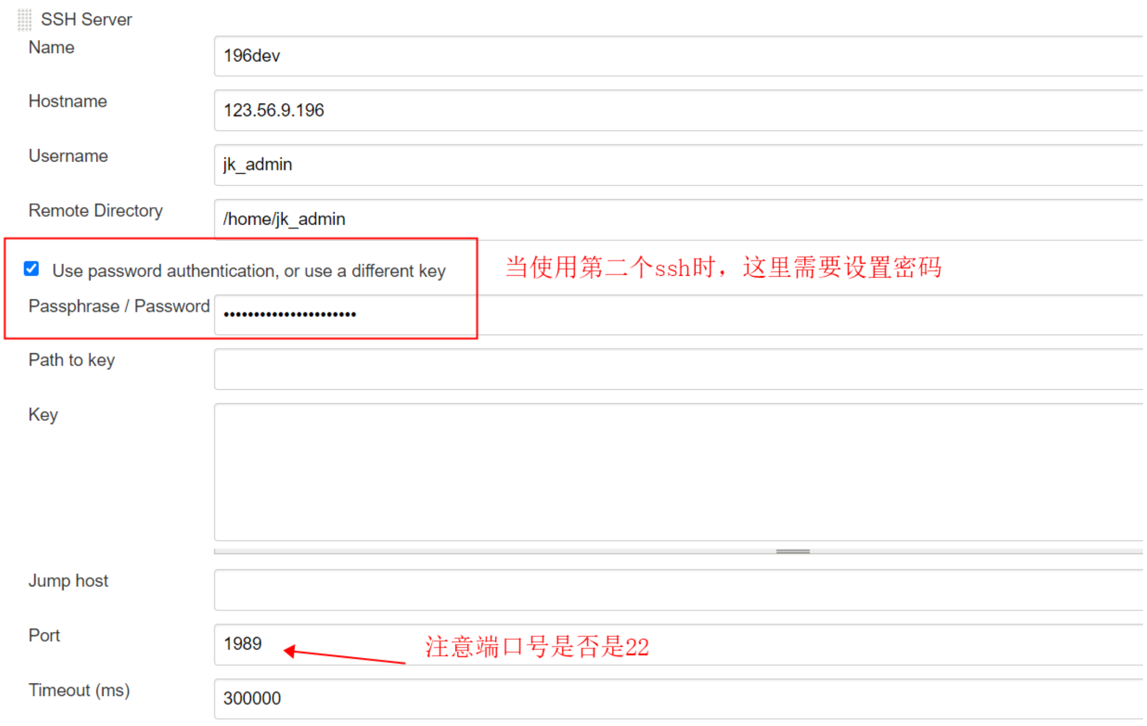
Success

Test Configuration

删除

增加

高级...


SSH Server

Name: 196dev

Hostname: 123.56.9.196

Username: jk_admin

Remote Directory: /home/jk_admin

☑ Use password authentication, or use a different key

Passphrase / Password: ••••••••••••••••••••••

Path to key:

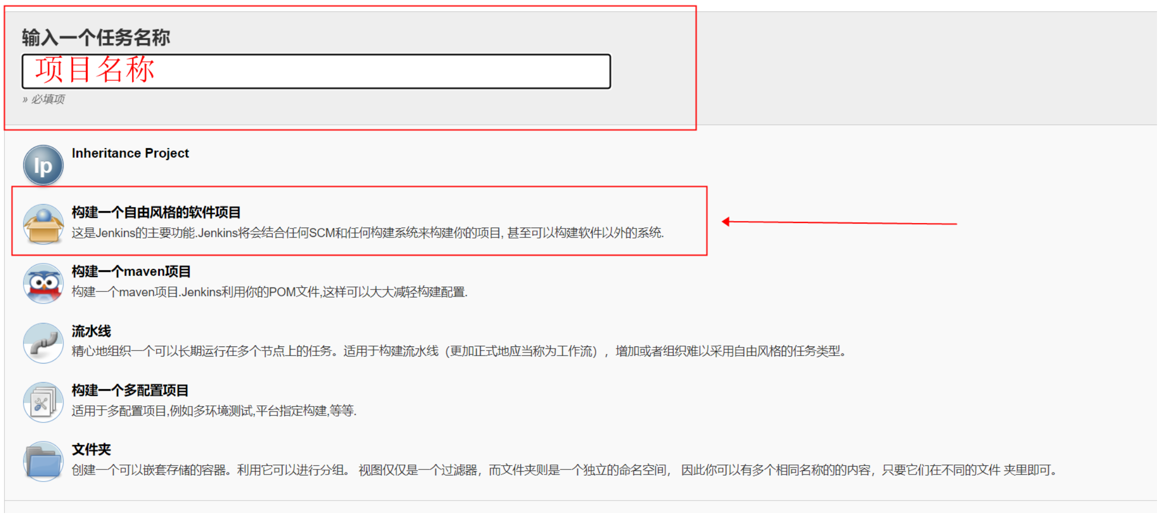
Key:

Jump host:

Port: 1989

Timeout (ms): 300000

## 4. 构建SpringBoot项目


输入一个任务名称

» 必填项

**Inheritance Project**

**构建一个自由风格的软件项目**
这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.

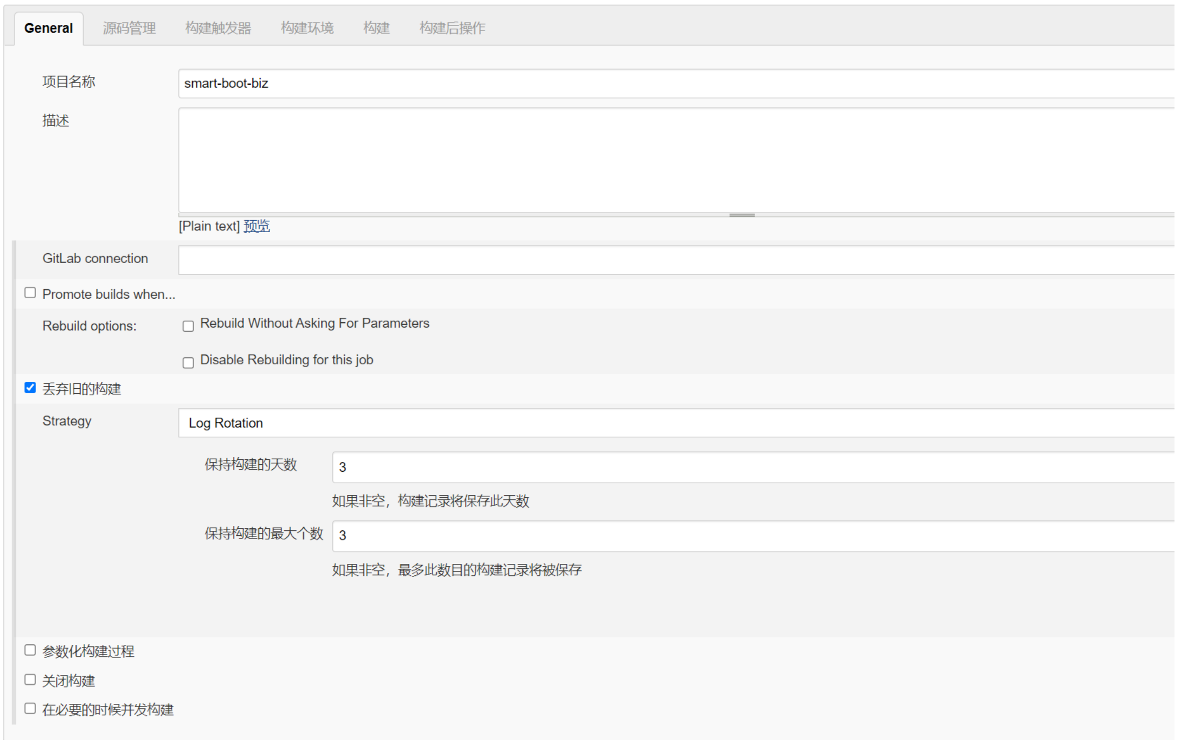
**构建一个maven项目**
构建一个maven项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置.

**流水线**
精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。

**构建一个多配置项目**
适用于多配置项目,例如多环境测试,平台指定构建,等等.

**文件夹**
创建一个可以嵌套存储的容器。利用它可以进行分组。 视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间， 因此你可以有多个相同名称的的内容，只要它们在不同的文件 夹里即可。

### 4.1 General



项目名称: smart-boot-biz

描述:

[Plain text] 预览

GitLab connection:

☐ Promote builds when...

Rebuild options: ☐ Rebuild Without Asking For Parameters ☐ Disable Rebuilding for this job

☑ 丢弃旧的构建

Strategy: Log Rotation

保持构建的天数: 3

如果非空，构建记录将保存此天数

保持构建的最大个数: 3

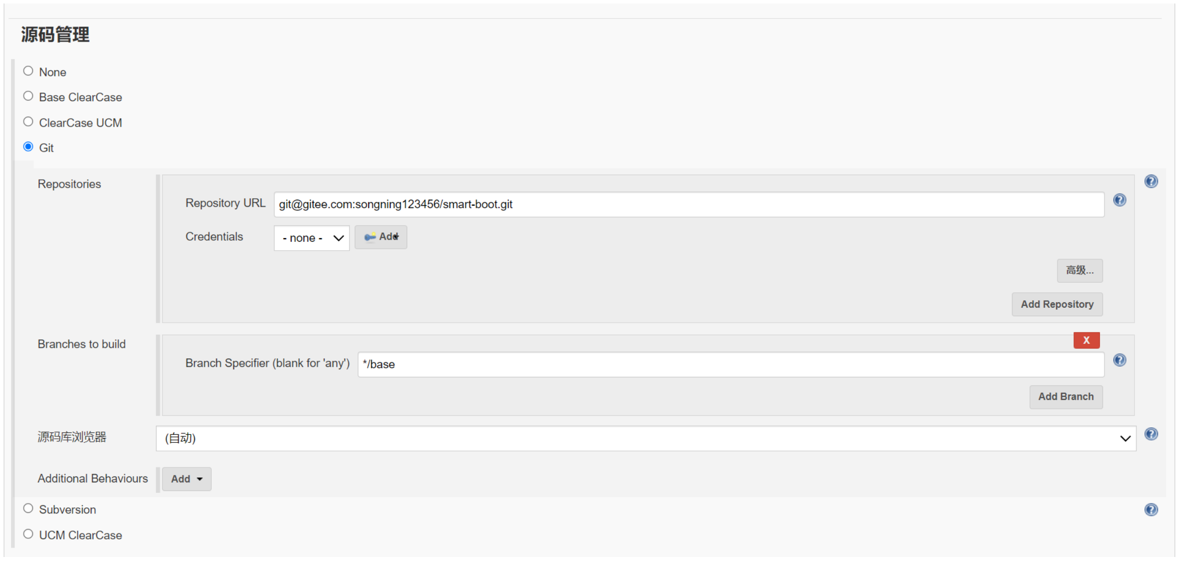
如果非空，最多此数目的构建记录将被保存

☐ 参数化构建过程

☐ 关闭构建

☐ 在必要的时候并发构建

## 4.2 源码管理


源码管理

○ None
○ Base ClearCase
○ ClearCase UCM
● Git

Repositories

Repository URL: git@gitee.com:songning123456/smart-boot.git

Credentials: - none -

Branches to build

Branch Specifier (blank for 'any'): */base

源码库浏览器: (自动)

Additional Behaviours

○ Subversion
○ UCM ClearCase

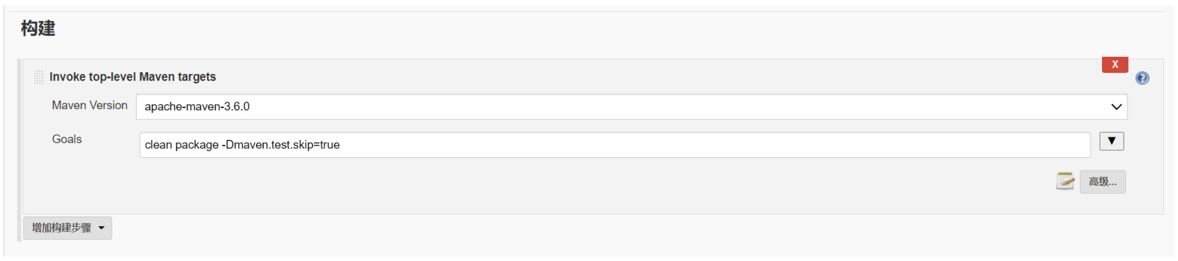
## 4.3 构建触发器

无

## 4.4 构建环境

无

## 4.5 构建

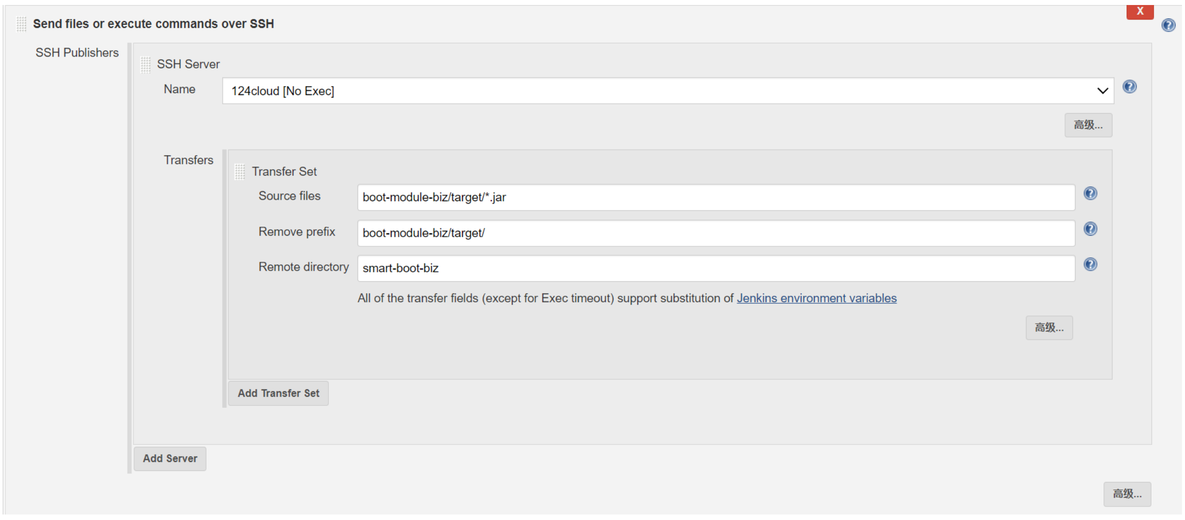

构建

Invoke top-level Maven targets

Maven Version: apache-maven-3.6.0

Goals: clean package -Dmaven.test.skip=true

```
clean package -Dmaven.test.skip=true
```


Send files or execute commands over SSH

SSH Publishers

SSH Server

Name: 124cloud [No Exec]

Transfers

Transfer Set

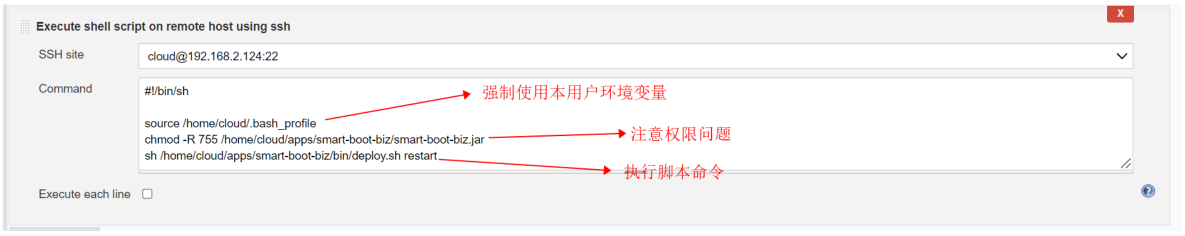
Source files: boot-module-biz/target/*.jar

Remove prefix: boot-module-biz/target/

Remote directory: smart-boot-biz

All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables

```
Source files: 地址的目录是相对于 ~/.jenkins/workspace/${项目名称} 的目录。
Remove prefix: 在Source files输入框中填入的地址，会默认在服务器下创建相同的文件夹，所以需要将我们不需要的文件夹在这里剔除掉。
Remote directory: 发送jar包到目标服务器的路径(默认添加ssh服务器前缀)。
```


Execute shell script on remote host using ssh

SSH site: cloud@192.168.2.124:22

Command:

```
#!/bin/sh

source /home/cloud/.bash_profile
chmod -R 755 /home/cloud/apps/smart-boot-biz/smart-boot-biz.jar
sh /home/cloud/apps/smart-boot-biz/bin/deploy.sh restart
```

e.g: smart-boot项目(指定deploy.sh脚本)

```
#!/bin/sh

source /home/cloud/.bash_profile
cd /home/cloud/apps/smart-boot-biz
chmod -R 755 smart-boot-biz.jar
./bin/deploy.sh restart
```

e.g: sk-scstt项目(指定deploy.sh脚本)

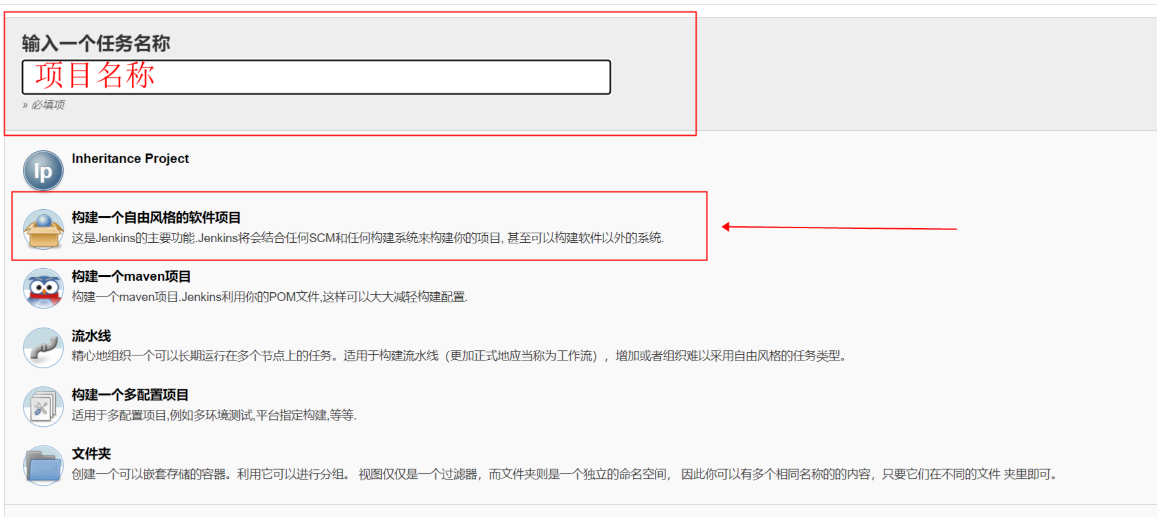
```
#!/bin/sh

# jar包名称
BOOT_JAR=skboot-scstt.jar
# 项目部署jar包所在目录
BOOT_JAR_DIR=/app/skboot/scstt/skboot-base-scstt
# 应用服务器存储jenkins传送jar包所在目录
TMP_BOOT_JAR_DIR=/home/jk_admin/scstt

# 强制使用jk_admin环境变量
source /home/jk_admin/.bash_profile
# 更改权限
chmod -R 755 $TMP_BOOT_JAR_DIR/$BOOT_JAR
# 以时间戳结尾重命名原先文件
sudo mv $BOOT_JAR_DIR/$BOOT_JAR $BOOT_JAR_DIR/$BOOT_JAR$(date +"%Y%m%d%H%M%S")
# 移动文件
sudo mv $TMP_BOOT_JAR_DIR/$BOOT_JAR $BOOT_JAR_DIR
# 执行重启脚本
```

```
sudo sh $BOOT_JAR_DIR/bin/deploy.sh restart
```

### 4.6 构建后操作

无

## 5. 构建nodeJS项目



输入一个任务名称

项目名称

» 必填项

Inheritance Project

构建一个自由风格的软件项目
这是Jenkins的主要功能.Jenkins将会结合任何SCM和任何构建系统来构建你的项目, 甚至可以构建软件以外的系统.

构建一个maven项目
构建一个maven项目.Jenkins利用你的POM文件,这样可以大大减轻构建配置.

流水线
精心地组织一个可以长期运行在多个节点上的任务。适用于构建流水线（更加正式地应当称为工作流），增加或者组织难以采用自由风格的任务类型。

构建一个多配置项目
适用于多配置项目,例如多环境测试,平台指定构建,等等.

文件夹
创建一个可以嵌套存储的容器。利用它可以进行分组。 视图仅仅是一个过滤器，而文件夹则是一个独立的命名空间， 因此你可以有多个相同名称的的内容，只要它们在不同的文件 夹里即可。

### 5.1 General

同SpringBoot项目

### 5.2 源码管理

同SpringBoot项目

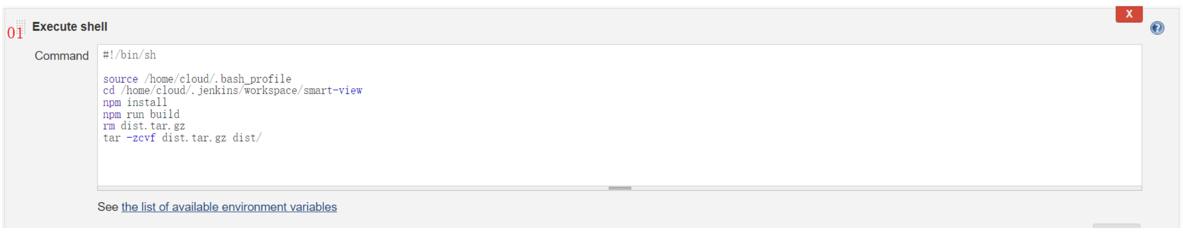
### 5.3 构建触发器

无

### 5.4 构建环境

无

### 5.5 构建



01 Execute shell

Command

```
#!/bin/sh

source /home/cloud/.bash_profile
cd /home/cloud/.jenkins/workspace/smart-view
npm install
npm run build
rm dist.tar.gz
tar -zcvf dist.tar.gz dist/
```

See the list of available environment variables

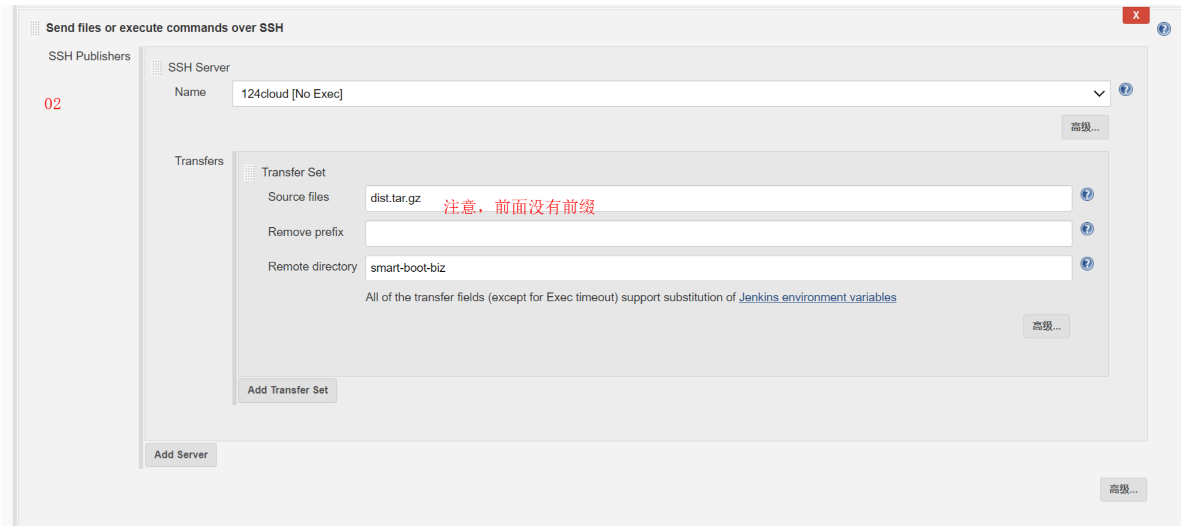
e.g smart-view项目

```
#!/bin/sh

source /home/cloud/.bash_profile
cd /home/cloud/.jenkins/workspace/smart-view
npm install
npm run build
rm dist.tar.gz
tar -zcvf dist.tar.gz dist/
```

e.g sk-view项目

```
#!/bin/sh

# 强制使用jenkins环境变量
source /home/cloud/.bash_profile
# 切到jenkins项目目录下
cd /home/cloud/apps/.jenkins/workspace/sk-view-scstt-dev
# 删除原先的生成文件
rm -rf sk-view
# 删除原先的*.tar.gz压缩包
rm sk-view.tar.gz
# 扩展项目内存
increase-memory-limit
# npm包安装
npm install
# npm构建
npm run build
# 生成新的*.tar.gz压缩包
tar -zcvf sk-view.tar.gz sk-view/
```

```
# sk-nodeJS项目第一次构建 注意事项:
1. npm install -g increase-memory-limit (扩大全局内存)
2. npm config set registry http://103.85.171.27:8082/repository/group-npm/ (设置sk源)
3. git clone base分支
# 之后每次target分支都从base分支拷贝
4. cp base分支 target分支
# 分支的node_modules如果是从windows上传，记得重构sass
5. npm rebuild node-sass
```


Send files or execute commands over SSH

SSH Publishers

02

SSH Server

Name: 124cloud [No Exec]

Transfers

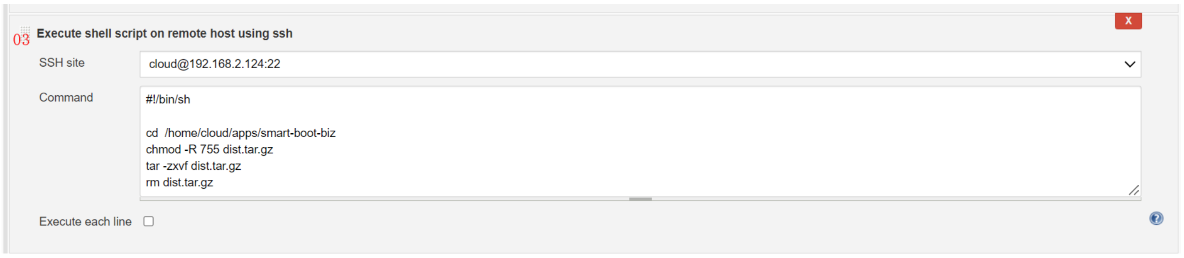
Transfer Set

Source files: dist.tar.gz 注意，前面没有前缀

Remove prefix:

Remote directory: smart-boot-biz

All of the transfer fields (except for Exec timeout) support substitution of Jenkins environment variables


03 Execute shell script on remote host using ssh

SSH site: cloud@192.168.2.124:22

Command:

```
#!/bin/sh

cd  /home/cloud/apps/smart-boot-biz
chmod -R 755 dist.tar.gz
tar -zxvf dist.tar.gz
rm dist.tar.gz
```

Execute each line ☐

e.g smart-view项目

```
#!/bin/sh

cd  /home/cloud/apps/smart-boot-biz
chmod -R 755 dist.tar.gz
tar -zxvf dist.tar.gz
rm dist.tar.gz
```

e.g sk-view项目

```
#!/bin/sh

# VUE项目名称
VIEW_NAME=sk-view
# nginx配置的指定目录
VIEW_DIR=/app/skboot/scstt/skboot-view-scstt
# 应用服务器存储jenkins传送*.tar.gz压缩包所在目录
TMP_VIEW_DIR=/home/jk_admin/scstt

# 更改权限
chmod -R 755 $TMP_VIEW_DIR/$VIEW_NAME.tar.gz
# 以时间戳结尾重命名原先文件
sudo mv $VIEW_DIR/$VIEW_NAME $VIEW_DIR/$VIEW_NAME$(date +"%Y%m%d%H%M%S")
# 解压到nginx配置的指定目录
sudo tar -zxvf $TMP_VIEW_DIR/$VIEW_NAME.tar.gz -C $VIEW_DIR
# 等待3s
sleep 3
# 删除应用服务器存储jenkins传送的*.tar.gz压缩包
sudo rm $TMP_VIEW_DIR/$VIEW_NAME.tar.gz
```

#### 5.6 构建后操作

无

## 6. 注意事项

### 6.1 SSH远程登录应用服务器失败

execute shell操作有可能有SSH远程登录失败问题，root用户登录到到应用服务器修改/etc/ssh/sshd_config文件。

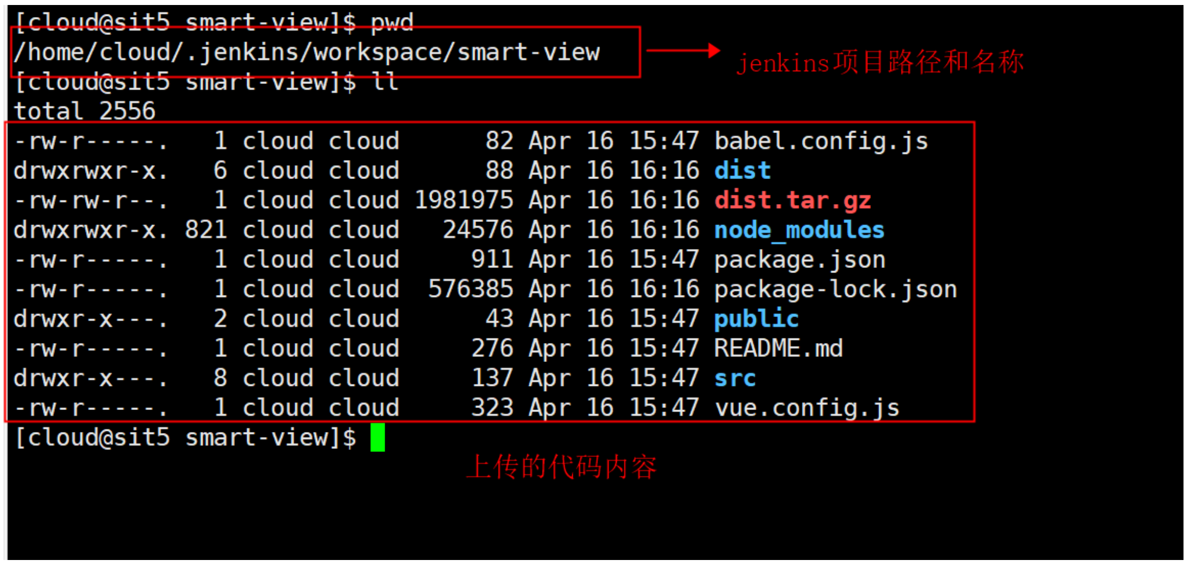
```
[root@sit ssh]# vi /etc/ssh/sshd_config
```

```
# (由No改为Yes)
PasswordAuthentication yes
```

```
# 启动sshd服务
[root@sit ssh]# service sshd restart (或者 systemctl restart sshd.service)
```

### 6.2 jenkins服务器拉取代码失败

可以直接从本地(window)上传代码到jenkins服务器(linux)，或者到.../.jenkins/workspace工作目录下执行git clone然后执行cp操作。



```
[cloud@sit5 smart-view]$ pwd
/home/cloud/.jenkins/workspace/smart-view
[cloud@sit5 smart-view]$ ll
total 2556
-rw-r-----.   1 cloud cloud      82 Apr 16 15:47 babel.config.js
drwxrwxr-x.   6 cloud cloud      88 Apr 16 16:16 dist
-rw-rw-r--.   1 cloud cloud 1981975 Apr 16 16:16 dist.tar.gz
drwxrwxr-x. 821 cloud cloud   24576 Apr 16 16:16 node_modules
-rw-r-----.   1 cloud cloud     911 Apr 16 15:47 package.json
-rw-r-----.   1 cloud cloud  576385 Apr 16 16:16 package-lock.json
drwxr-x---.   2 cloud cloud      43 Apr 16 15:47 public
-rw-r-----.   1 cloud cloud     276 Apr 16 15:47 README.md
drwxr-x---.   8 cloud cloud     137 Apr 16 15:47 src
-rw-r-----.   1 cloud cloud     323 Apr 16 15:47 vue.config.js
[cloud@sit5 smart-view]$
```

### 6.3 git Host key verification failed异常

[Jenkins源码管理git报错：Host key verification failed]

```
# jenkins服务器执行如下命令并yes回车
[jenkins@sit1 .ssh]$ git ls-remote -h
git@servermanage.kingtroldata.com:xiechao/skboot-base-cx.git
```

输入yes回车。这时，ip就已经添加到~/.ssh/known_hosts。

### 6.4 新增应用服务器

每添加一台应用服务器，jenkins需要配置如下:

- 应用服务器Credentials (用户名/密码 类型)
- 系统管理 => 系统设置 => SSH remote hosts
- 系统管理 => 系统设置 => Publish over SSH

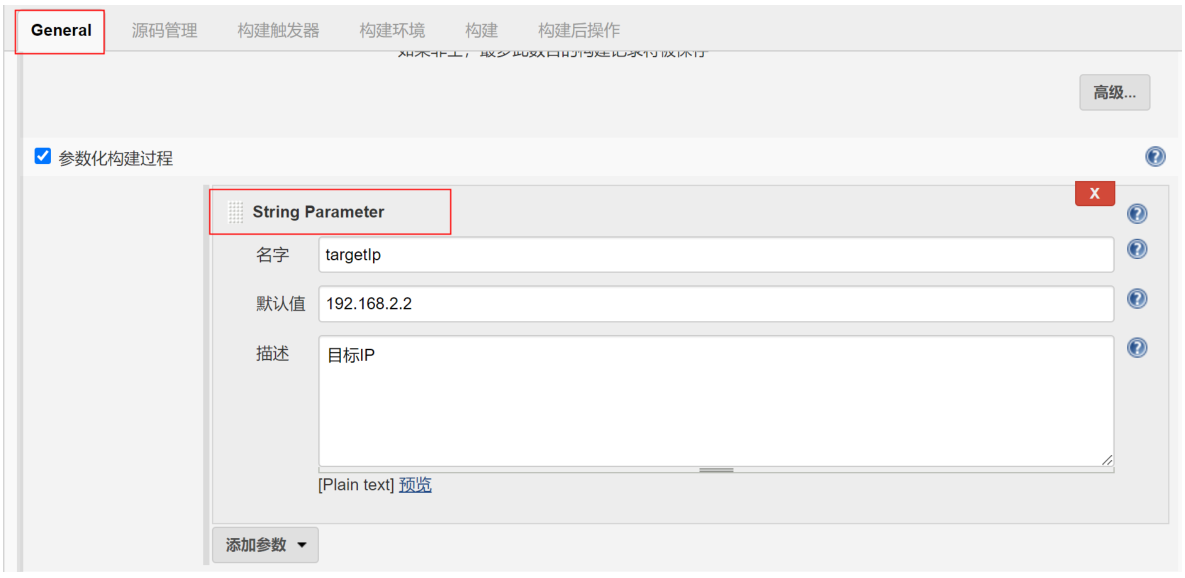
### 6.5 com.jcraft.jsch.JSchException: Auth fail

```
[root@sit ssh]# vim /etc/ssh/sshd_config
```

```
GSSAPIAuthentication no
UseDNS no
```

```
[root@sit ssh]# systemctl restart sshd.service
```

### 6.6 项目配置变量参数

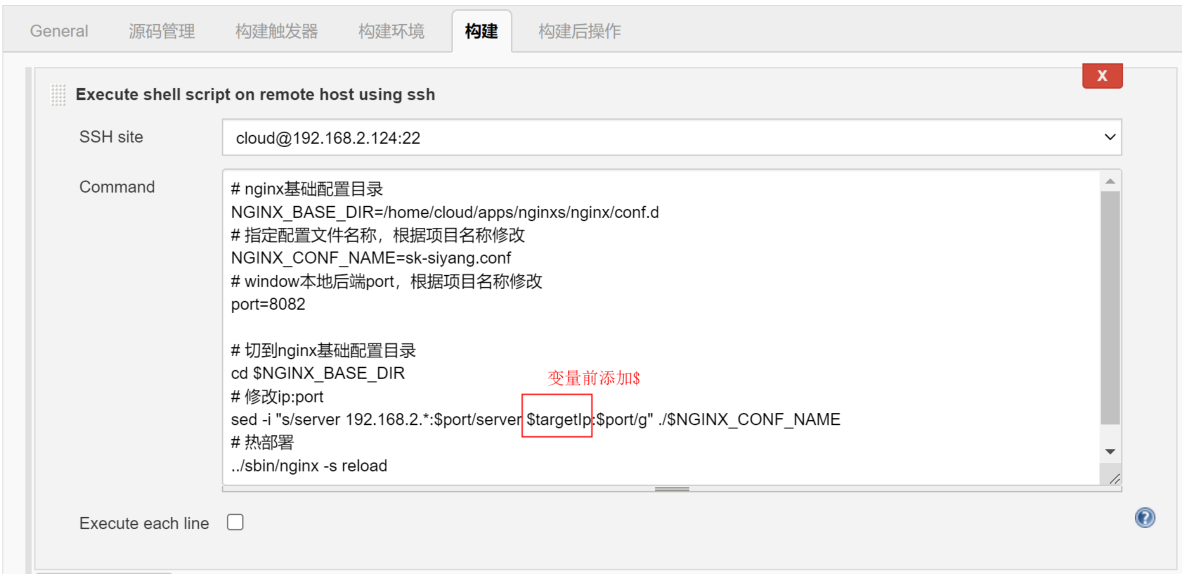


☑ 参数化构建过程

String Parameter

| | |
|---|---|
| 名字 | targetIp |
| 默认值 | 192.168.2.2 |
| 描述 | 目标IP |

[Plain text] 预览



Execute shell script on remote host using ssh

SSH site: cloud@192.168.2.124:22

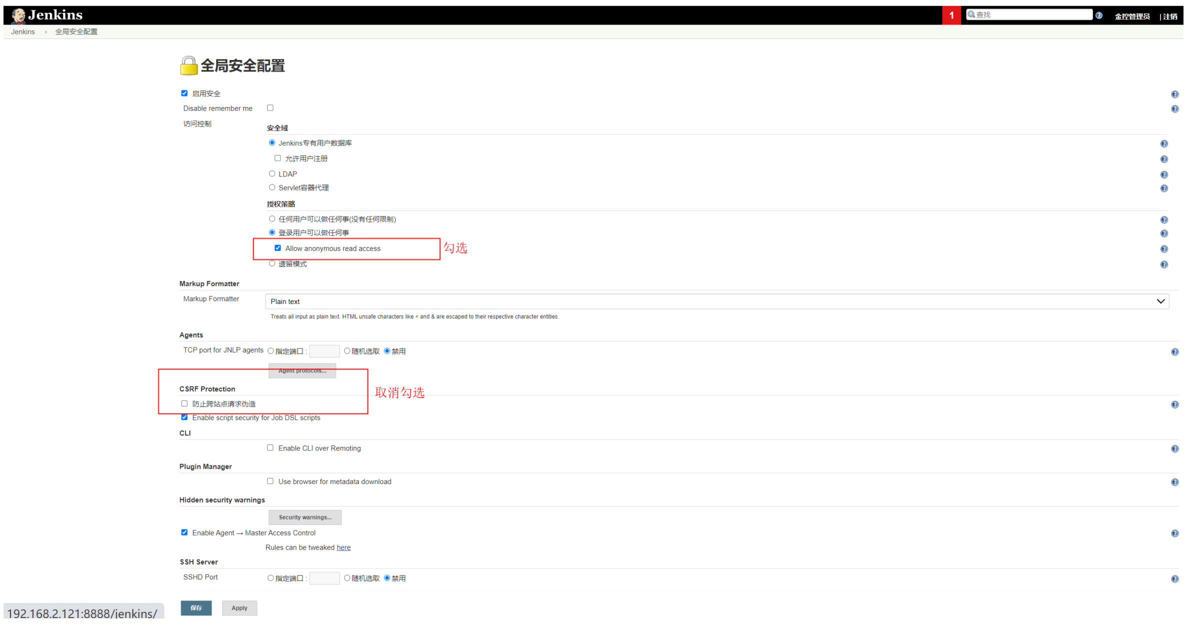
Command:

```
# nginx基础配置目录
NGINX_BASE_DIR=/home/cloud/apps/nginxs/nginx/conf.d
# 指定配置文件名称，根据项目名称修改
NGINX_CONF_NAME=sk-siyang.conf
# window本地后端port，根据项目名称修改
port=8082

# 切到nginx基础配置目录
cd $NGINX_BASE_DIR
# 修改ip:port
sed -i "s/server 192.168.2.*:$port/server $targetIp:$port/g" ./$NGINX_CONF_NAME
# 热部署
../sbin/nginx -s reload
```

Execute each line ☐

## 7. 通过Jenkins API 去build一个job，及相应安全策略处理

参考: [https://blog.csdn.net/carcarrot/article/details/118497920](https://blog.csdn.net/carcarrot/article/details/118497920)



全局安全配置

☑ 启用安全

Disable remember me ☐

访问控制

安全域
- ● Jenkins专有用户数据库
  - ☐ 允许用户注册
- ○ LDAP
- ○ Servlet容器代理

授权策略
- ○ 任何用户可以做任何事(没有任何限制)
- ● 登录用户可以做任何事
  - ☑ Allow anonymous read access
- ○ 遗留模式

Markup Formatter

Markup Formatter: Plain text

Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.

Agents

TCP port for JNLP agents ○指定端口： ○随机选取 ●禁用

CSRF Protection

☐ 防止跨站点请求伪造

☑ Enable script security for Job DSL scripts

CLI

☐ Enable CLI over Remoting

Plugin Manager

☐ Use browser for metadata download

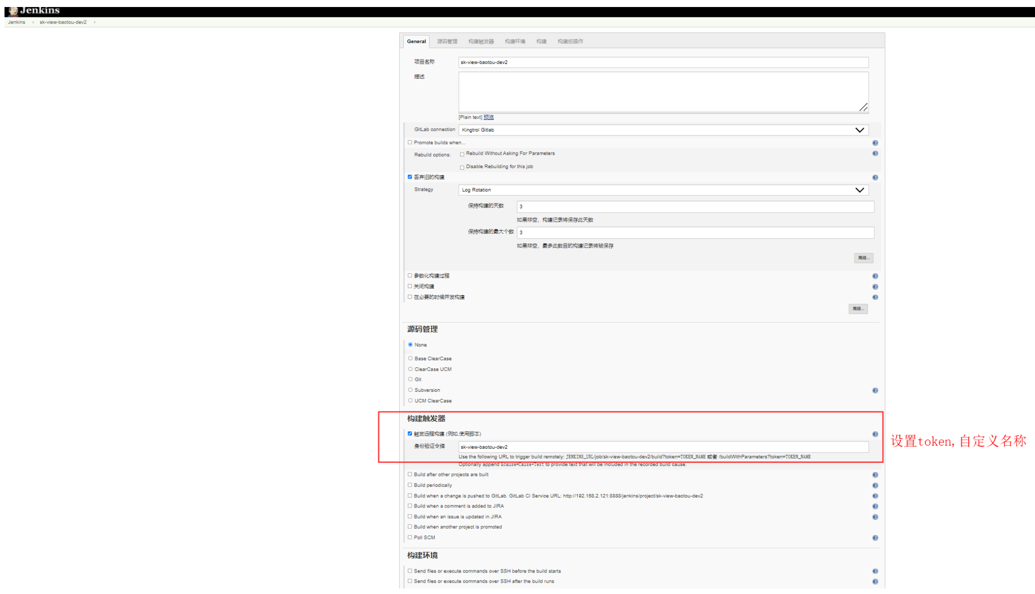
Hidden security warnings

☑ Enable Agent → Master Access Control

Rules can be tweaked here

SSH Server

SSHD Port ○指定端口： ○随机选取 ●禁用



项目名称 sk-view-baotou-dev2

描述

[Plain text] 预览

GitLab connection: Kingtrol Gitlab

☐ Promote builds when...

Rebuild options: ☐ Rebuild Without Asking For Parameters
☐ Disable Rebuilding for this job

☑ 丢弃旧的构建

Strategy: Log Rotation

保持构建的天数 3
如果非空，构建记录将保存此天数

保持构建的最大个数 3
如果非空，最多此数目的构建记录将被保存

☐ 参数化构建过程
☐ 关闭构建
☐ 在必要的时候并发构建

源码管理

● None
○ Base ClearCase
○ ClearCase UCM
○ Git
○ Subversion
○ UCM ClearCase

构建触发器

☑ 触发远程构建 (例如,使用脚本)

身份验证令牌 sk-view-baotou-dev2

Use the following URL to trigger build remotely: JENKINS_URL/job/sk-view-baotou-dev2/build?token=TOKEN_NAME 或者 /buildWithParameters?token=TOKEN_NAME
Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

设置token, 自定义名称

☐ Build after other projects are built
☐ Build periodically
☐ Build when a change is pushed to GitLab. GitLab CI Service URL: http://192.168.2.121:8888/jenkins/project/sk-view-baotou-dev2
☐ Build when a comment is added to JIRA
☐ Build when an issue is updated in JIRA
☐ Build when another project is promoted
☐ Poll SCM

构建环境

☐ Send files or execute commands over SSH before the build starts
☐ Send files or execute commands over SSH after the build runs

curl -X post http://192.168.2.121:8888/jenkins/job/sk-view-baotou-dev2/build?token=${身份验证令牌}

## 安装zookeeper(单机)

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf zookeeper-3.5.9.tar.gz && rm zookeeper-3.5.9.tar.gz
&& chmod -R 755 zookeeper-3.5.9/
```

### 2. 修改配置文件

- zoo.cfg

### 3. 启动

```
[cloud@sit bin]$ sh zkServer.sh start
```

### 4. 验证

```
[cloud@sit bin]$ sh zkServer.sh status
```

## 安装zookeeper集群(单机多节点)

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf zookeeper-cluster.tar.gz && rm zookeeper-
cluster.tar.gz && chmod -R 755 zookeeper-cluster/
```

### 2. 修改配置文件

- zoo.cfg

### 3. 启动

```
[cloud@sit bin]$ sh zkServer.sh start
```

```
# Ubuntu系统报错
zkServer.sh: 78: /.../apache-zookeeper-3.5.6/bin/zkEnv.sh: [[: not found
-p: not found
java is /usr/lib/jvm/jdk1.8.0_191/bin/java
Error: JAVA_HOME is not set and java could not be found in PATH.
```

```
Ubuntu的默认shell为dash，把dash改成bash就可以了。
把sh zkServer.sh start换成bash zkServer.sh start或修改系统默认的shell为bash。
```

```
[cloud@sit bin]$ bash zkServer.sh start
```

```
cd /bin

# 查看默认的shell，可以看到为dash
root@server-4:/bin# ls -l /bin/sh
lrwxrwxrwx 1 root root 4 Oct 17 17:57 /bin/sh -> dash

# 修改为bash
root@server-4:/bin# ln -sf bash /bin/sh

# 再次检查，则为bash
root@server-4:/bin# ls -l /bin/sh
lrwxrwxrwx 1 root root 4 Dec  9 10:22 /bin/sh -> bash
```

### 4. 验证

```
[cloud@sit apps]$ sh zkServer.sh status
```

### 5. 启动失败

```
[cloud@sit bin]$ sh zkServer.sh status
ZooKeeper JMX enabled by default
Using config: /home/cloud/apps/zookeeper-cluster/server_03/zookeeper-
3.5.9/bin/../conf/zoo.cfg
Client port found: 22181. Client address: localhost. Client SSL: false.
Error contacting service. It is probably not running.
```

```
# 在zoo.cfg中不能使用0.0.0.0
server.01=内网IP:2888:3888
server.02=内网IP:12888:13888
server.03=内网IP:22888:23888
```

## 安装zkui

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf zkui.tar.gz && rm zkui.tar.gz && chmod -R 755
zkui.tar.gz/
```

### 2. 启动

```
[cloud@sit zkui]$ sh deploy.sh start
```

### 3. 修改配置文件

- config.cfg

### 4. 验证

```
登陆页面: ip:6090
用户名/密码: admin/manager
```

## 安装kafka(单机)

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf kafka_2.12-2.8.0.tar.gz && rm kafka_2.12-
2.8.0.tar.gz && chmod -R 755 kafka_2.12-2.8.0/
```

### 2. 修改配置文件

- server.properties

### 3. 启动

```
[cloud@sit bin]$ sh kafka-server-start.sh -daemon ../config/server.properties
```

### 4. 验证

```
[cloud@sit ~]$ lsof -i:9092
```

### 5. 参考文档

[Kafka单机多节点部署]()

## 安装kafka集群(单机多节点)

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf kafka-cluster.tar.gz && rm kafka-cluster.tar.gz &&
chmod -R 755 kafka-cluster/
```

### 2. 修改配置文件

- server.properties

```
# 云服务机部署kafka(特别注意)
# listeners：启动kafka服务监听的ip和端口，可以监听内网ip和0.0.0.0(不能为外网ip)，默认为
java.net.InetAddress.getCanonicalHostName()获取的ip
listeners=PLAINTEXT://内网IP:9092
# 生产者和消费者连接的地址，kafka会把该地址注册到zookeeper中，所以只能为除0.0.0.0之外的合法
ip或域名，默认和listeners的配置一致。
advertised.listeners=PLAINTEXT://外网IP:9092
```

### 3. 启动

```
[cloud@sit bin]$ sh kafka-server-start.sh -daemon ../config/server.properties
```

### 4. 停止

```
[cloud@sit bin]$ sh kafka-server-stop.sh
```

### 5. 连接异常

```
> telnet 192.168.2.123 9092 (失败)
> telnet 127.0.0.1 9092 (成功)
```

```
使用 listeners=PLAINTEXT://192.168.2.123:9092
而非 listeners=PLAINTEXT://127.0.0.1:9092
```

### 6. 参考

Kafka单机多节点部署

## 安装jmeter

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf apache-jmeter-5.4.1.tar.gz && rm apache-jmeter-5.4.1.tar.gz && chmod -R 755 apache-jmeter-5.4.1/
```

### 2. 配置环境变量

```
[cloud@sit ~]$ vi ~/.bash_profile
```

```
# 优先配置JDK环境变量
JAVA_HOME=/home/cloud/apps/jdk1.8.0_181
CLASSPATH=$JAVA_HOME/lib/
PATH=$JAVA_HOME/bin:$PATH
export PATH JAVA_HOME CLASSPATH

export PATH=/home/cloud/apps/apache-jmeter-5.4.1/bin:$PATH
export JMETER_HOME=/home/cloud/apps/apache-jmeter-5.4.1
export
CLASSPATH=$JMETER_HOME/lib/ext/ApacheJMeter_core.jar:$JMETER_HOME/lib/jorphan.jar:$CLASSPATH
export PATH=$JMETER_HOME/bin:$PATH
```

```
[cloud@sit ~]$ source ~/.bash_profile
```

### 3. 测试安装成功

```
[cloud@sit ~]$ jmeter -v
```

### 4. 启动

```
[cloud@sit apache-jmeter-5.4.1]$ mkdir report-log
```

```
[cloud@sit report-log]$ jmeter -n -t pginsert.jmx -l pginsert01.jtl -e -o
./pginsert01/
```

```
[cloud@sit report-log]$ nohup jmeter -n -t pginsert.jmx -l pginsert01.jtl -e -o
./pginsert01/ >out.log 2>&1 &
```

```
# Jmeter无界面运行脚本
jmeter -n -t [jmx file] -l [result file] -e -o [Path to web report folder]
例如：jmeter -n -t pengfei.jmx -l result/report.jtl -e -o report
-n 指定 JMeter 将在 cli 模式下运行
-t 包含测试计划的 jmx 文件名称
-l 记录测试结果的 jtl 文件名称
-j 记录 Jmeter 运行日志的文件名称
-g 输出报告文件（ .csv 文件）
-e 生成 html 格式的测试报表
-o 生成测试报表的文件夹文件夹不存在或为空
```

## 5. 生成HTML性能测试报告

生成HTML性能测试报告

## 6. 集群测试

```
master: 172.172.16.127;
slave: 172.172.16.23, 172.172.16.24;

1. 修改slave-172.172.16.23文件: jmeter.properties
```

server.rmi.ssl.disable=true
```


2. 修改slave-172.172.16.23文件: jmeter-server
```

RMI_HOST_DEF=-Djava.rmi.server.hostname=172.172.16.123
```


3. 启动slave-172.172.16.23
```

./jmeter-server
```


4. 所有slave(172.172.16.23, 172.172.16.24...)如上操作

5. 修改master-172.172.16.127文件: jmeter.properties
```

server.rmi.ssl.disable=true
remote_hosts=172.172.16.123:1099,172.172.16.124:1099
```


6. master-172.172.16.127启动
```

// -r: 远程启动slave
```
```

```
jmeter -n -t PA01-400-5.jmx -r -l PA01-400-5-V1.jtl -e -o ./PA01-400-5-V1/
```
```

## 安装EMQX

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf emqx-centos7-4.2.14-x86_64.tar.gz && rm emqx-
centos7-4.2.14-x86_64.tar.gz && chmod -R 755 emqx-centos7-4.2.14-x86_64
```

### 2. 启动

```
[cloud@sit bin]$ ./emqx start
```

```
EMQ X Broker 4.2.14 is started successfully!
```

### 3. 其他命令

```
# 后台启动EMQ X Broker
[cloud@sit bin]$ emqx start

# 关闭EMQ X Broker
[cloud@sit bin]$ emqx stop

# 重启EMQ X Broker
[cloud@sit bin]$ emqx restart

# 使用控制台启动EMQ X Broker
[cloud@sit bin]$ emqx console

# 使用控制台启动EMQ X Broker，与emqx console 不同，emqx foreground不支持输入Erlang命令
[cloud@sit bin]$ emqx foreground

# Ping EMQ X Broker
[cloud@sit bin]$ emqx ping
```

### 4. web访问

```
http://localhost:18083/
```

```
username: admin
password: public
```

Settings => EN/中文 => Apply

### 5. 参考

EMQ X 简介与MQTT集成Java开发

EMQ 安装及简单使用

## 安装Mongodb
```

## 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf mongodb-3.0.15.tar.gz && rm mongodb-3.0.15.tar.gz &&
chmod -R 755 mongodb-3.0.15.tar.gz
```

## 2. 创建目录和文件

```
[cloud@sit mongodb-3.0.15]$ mkdir data && mkdir log && mkdir config
```

```
[cloud@sit mongodb-3.0.15]$ touch log/mongo.log
```

```
[cloud@sit mongodb-3.0.15]$ vim config/mongo.cg
```

```
dbpath=/app/mongodb-3.0.15/data
logpath=/app/mongodb-3.0.15/log/mongo.log
logappend=true
journal=true
quiet=true
port=27017
fork=true # 后台运行
bind_ip=0.0.0.0 # 允许所有IP连接
auth=false # 是否授权连接
```

## 3. 测试是否缺少依赖包

```
[cloud@sit mongodb-3.0.15]$ ldd bin/mongod
```

```
linux-vdso.so.1 (0x00007ffff52e6000)
libpthread.so.0 => /lib/x86_64-linux-gnu/libpthread.so.0 (0x00007f3ecd855000)
libssl.so.1.0.0 => not found
libcrypto.so.1.0.0 => not found
librt.so.1 => /lib/x86_64-linux-gnu/librt.so.1 (0x00007f3ecd850000)
libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f3ecd84b000)
libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f3ecd762000)
libgcc_s.so.1 => /lib/x86_64-linux-gnu/libgcc_s.so.1 (0x00007f3ecd742000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f3ecd51a000)
/lib64/ld-linux-x86-64.so.2 (0x00007f3ecd86a000)
```

如果有缺失，从centos服务器或者其他服务器拷贝到/lib/x86_64-linux-gnu目录下。

```
[cloud@sit mongodb-3.0.15]$ mv /home/sskj/libssl.so.1.0.0 /lib/x86_64-linux-
gnu/libssl.so.1.0.0
[cloud@sit mongodb-3.0.15]$ mv /home/sskj/libcrypto.so.1.0.0 /lib/x86_64-linux-
gnu/libcrypto.so.1.0.0
```

## 4. 启动

```
[cloud@sit mongodb-3.0.15]$ ./bin/mongod --journal -f config/mongo.cfg
```

```
./bin/mongod: /lib/x86_64-linux-gnu/libcrypto.so.1.0.0: no version information
available (required by ./bin/mongod)
./bin/mongod: /lib/x86_64-linux-gnu/libssl.so.1.0.0: no version information
available (required by ./bin/mongod)
warning: bind_ip of 0.0.0.0 is unnecessary; listens on all ips by default
about to fork child process, waiting until server is ready for connections.
forked process: 4637
child process started successfully, parent exiting
```

### 5. 关闭

```
# 同kill -9 PID
[cloud@sit mongodb-3.0.15]$ ./bin/mongod --journal --shutdown -f
config/mongo.cfg
```

## 安装RabbitMQ

### 1. 安装Erlang环境

```
[cloud@sit ~]$ sudo apt-get install build-essential
[cloud@sit ~]$ sudo apt-get install libncurses5-dev
[cloud@sit ~]$ sudo apt-get install libssl-dev
[cloud@sit ~]$ sudo apt-get install m4
[cloud@sit ~]$ sudo apt-get install unixodbc unixodbc-dev
[cloud@sit ~]$ sudo apt-get install freeglut3-dev libwxgtk2.8-dev
[cloud@sit ~]$ sudo apt-get install xsltproc
[cloud@sit ~]$ sudo apt-get install fop
[cloud@sit ~]$ sudo apt-get install tk8.5
```

```
[cloud@sit ~]$ sudo apt-get install erlang
```

```
# 控制台输入erl，查看erlang安装版本情况
[cloud@sit ~]$ erl
```

```
Erlang/OTP 24 [erts-12.2.1] [source] [64-bit] [smp:16:16] [ds:16:16:10] [async-
threads:1] [jit]

Eshell V12.2.1  (abort with ^G)
1>
```

### 2. 基本操作

```
[cloud@sit apps]$ tar -zxvf rabbitmq_server-3.10.7.tar.gz && rm rabbitmq_server-
3.10.7.tar.gz && chmod -R 755 rabbitmq_server-3.10.7.tar.gz
```

### 3. 修改配置文件

```
[cloud@sit ~]$ cd /app/rabbitmq_server-3.10.7/etc/rabbitmq
[cloud@sit rabbitmq]$ vim rabbitmq.config
```

```
[{rabbit, [{loopback_users, []}]}].
```

:wq!保存并退出。

### 4. 安装插件

```
[cloud@sit sbin]$ ./rabbitmq-plugins enable rabbitmq_management
```

### 5. 创建用户并授权

```
# 创建用户
[cloud@sit sbin]$ ./rabbitmqctl add_user user_admin passwd_admin
# 授权administrator角色
[cloud@sit sbin]$ ./rabbitmqctl set_user_tags user_admin administrator
```

### 6. 启动

```
# -detached后台运行
[cloud@sit sbin]$ ./rabbitmq-server -detached
```

```
# 查看运行状态
[cloud@sit sbin]$ ./rabbitmq-server status
```

### 7. 登录

[http://ip:15672/](http://ip:15672/)

```
username: user_admin
password: passwd_admin
```

### 8. 参考文档

[http://www.codebaoku.com/it-linux/it-linux-112105.html](http://www.codebaoku.com/it-linux/it-linux-112105.html)

## 安装kibana(V6.6.0或者V7.14.0)

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf kibana-6.6.0.tar.gz && rm kibana-6.6.0.tar.gz &&
chmod -R 755 kibana-6.6.0/
```

### 2. 修改配置文件

- kibana.yml

```
server.port: 5601
server.host: "0.0.0.0"
server.name: "kibana"
elasticsearch.hosts: ["http://10.1.1.197:9200"]
elasticsearch.ssl.verificationMode: none
# elasticsearch.ssl.certificateAuthorities:
["/data/kibana/config/newfile.crt.pem"]
# elasticsearch.preserveHost: true
# kibana.index: ".kibana"
# i18n.locale: "en"
# elasticsearch.username: "elastic"
# elasticsearch.password: "lianshi2020"
```

### 3. 启动

```
[cloud@sit bin]$ sh start.sh
```

```
# 或者
[cloud@sit bin]$ nohup ./kibana >out.log 2>&1 &
```

### 4. 验证

浏览器访问http://ip:5601。

## 安装influxdb

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf influxdb-1.8.0-1.tar.gz && rm influxdb-1.8.0-
1.tar.gz && chmod -R 755 influxdb-1.8.0-1/
```

### 2. 修改配置文件

```
[cloud@sit apps]$ mkdir -p /app/influxdb-1.8.0-1/dbdata/meta
[cloud@sit apps]$ mkdir -p /app/influxdb-1.8.0-1/dbdata/data
[cloud@sit apps]$ mkdir -p /app/influxdb-1.8.0-1/dbdata/wal
```

```
[cloud@sit influxdb]$ vim influxdb.conf
```

```
[meta]
  dir = "/app/influxdb-1.8.0-1/dbdata/meta"  # meta数据存放目录
[data]
  dir = "/app/influxdb-1.8.0-1/dbdata/data"  # 最终数据（TSM文件）存储目录
  wal-dir = "/app/influxdb-1.8.0-1/dbdata/wal"  # 预写日志存储目录
```

### 3. 启动

```
[cloud@sit ~]$ nohup /app/influxdb-1.8.0-1/usr/bin/influxd -config
/app/influxdb-1.8.0-1/etc/influxdb/influxdb.conf >out.log 2>&1 &
```

### 4. 验证

```
[cloud@sit bin]$ ./influx -port 8086
```

```
Connected to http://localhost:8086 version 1.8.0
InfluxDB shell version: 1.8.0
>
```

### 5. 创建一个用户，作为管理用户

```
> create user "cloud" with password 'cloud' with all privileges
> show users
user admin
---- -----
cloud true
```

### 6. 参考

[InfluxDB安装及使用]

[influxdb可视化工具及使用]

## 基于centos7离线安装telnet

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf telnet_centos7.tar.gz && rm telnet_centos7.tar.gz &&
chmod -R 755 telnet_centos7/
```

### 2. 安装rpm包

```
# root操作
[cloud@sit telnet_centos7]$ rpm -ivh xinetd-2.3.15-14.el7.x86_64.rpm
[cloud@sit telnet_centos7]$ rpm -ivh telnet-0.17-65.el7_8.x86_64.rpm
[cloud@sit telnet_centos7]$ rpm -ivh telnet-server-0.17-65.el7_8.x86_64.rpm
```

### 3. 查看是否安装成功

```
# root操作
[cloud@sit telnet_centos7]$ rpm -qa | grep telnet
[cloud@sit telnet_centos7]$ rpm -qa | grep xinetd
```

### 4. 启动telnet依赖的xinetd服务

```
# root操作
[cloud@sit telnet_centos7]$ service xinetd restart
```

### 5. 查看xinetd是否启动

```
# root操作
[cloud@sit telnet_centos7]$ ps -ef|grep xinetd
```

### 6. 测试

```
[cloud@sit telnet_centos7]$ telnet ip/域名 端口
```

### 7. 参考

[基于centos7离线安装telnet]

## 安装中文字体

使用**root**权限

### 1. 安装字体命令

```
[root@sit ~]# yum -y install fontconfig
```

### 2. 查看linux已安装中文字体

```
[root@sit ~]# fc-list :lang=zh
# 可以看出，linux默认字体是没有中文字体的，需要手动安装。
```

### 3. 安装字体

```
# 创建目录
[root@sit ~]# mkdir -p /usr/share/fonts/my_fonts
# 上传字体
[root@sit my_fonts]# rz (simsun.ttc)
# 安装字体索引指令
[root@sit my_fonts]# yum install mkfontscale
# 生成字体索引
[root@sit my_fonts]# mkfontscale
# 重新生成字体cache
[root@sit my_fonts]# fc-cache -frv
# 查看fc cache内的内容
[root@sit my_fonts]# fc-cat -rv
# 查看(宋体常规)字体是否安装成功
[root@sit my_fonts]# fc-list :lang=zh
```

```
# 重启应用服务
```

## 安装protobuf

### 1. 基本操作

```
[cloud@sit apps]$ tar -zxvf protoc-3.6.1.tar.gz && rm protoc-3.6.1.tar.gz &&
chmod -R 755 protoc-3.6.1/
```

### 2. 生成protobuf文件

```
# $SOURCE_DIR: .proto所在的源目录;
# --java_out: 生成java代码;
# $TARGET_DIR: 生成代码的目标目录;
# xxx.proto: 要针对哪个proto文件生成接口代码;
e.g: protoc -I=$SOURCE_DIR --java_out=$TARGET_DIR $SOURCE_DIR/xxx.proto
```

```
[cloud@sit protoc-3.6.1]$ mkdir source && mkdir target && cd bin/
[cloud@sit bin]$ ./protoc -I=../source/ --java_out=../target/
../source/MsgInfo.proto
```