

# CICD的安装配置

主要工作：  
Gerrit、Jenkins的安装配置，Jobs的导入。

部署方案：

硬件	虚拟机2台(Centos7.2；Node1： 4core， 16G； Node2： 4core， 8G)
安装软件	Gerrit, Jenkins
部署方案	Gerrit： 代码审查平台， 备份至gitlab， 用户管理： ldap； Jenkins： Master， slave1(rpm jobs)， slave2(ios-build)。
实现方案	Node1： Gerrit, Jenkins:master， Jenkins:slave1(docker， rpm)； Node2： slave2(ios-build)。

方案实现原理：  
Gerrit作为独立的代码审查平台， Jenkins作为代码打包(rpm)， 以及发布的工具， build为ios文件， 利用roller进行部署。 Gerrit与gitlab对接， 将本地库备份至gitlab， 利用ldap实现用户管理。  
Jenkins master作为调度器， 包含两类job， 一类打包rpm job， 一类ios-build job。 其中rpm job运行于node1环境中， ios-build job运行于独立的节点上。 在初次安装时， 需要将gitlab库中的库更新至本地， 在随后打包时， 自动更新。 Jenkins的job需要手动构建。

## Gerrit的安装

Gerrit 版本	gerrit-2.12.3	离线安装， 有安装包
依赖包	DB(mariadb.x86_64， mariadb-libs.x86_64， mariadb-server.x86_64)	本地源
依赖包	mysql-connector-java-5.1.21.jar	离线包免安装， 有包
依赖包	>=JDK1.8， git， git-review	本地源
插件	replication.jar， commit-message-length-validator.jar， delete-project.jar， download-commands.jar， reviewnotes.jar， singleusergroup.jar	离线包免安装， 有离线包

### 安装步骤

```
#配置银联的Centos源以及本地源
wget http://172.17.249.122/xhlin/mirrors/repository/archive.tar.gz
tar xzvf archive.tar.gz
cd mirrors.git/
cp config.sh configrun.sh
chmod +x configrun.sh
./configrun.sh -y
yum clean
yum update

#本地源
```

建立本地源的方法可详见[Link](#)。

```
vi /etc/yum.repos.d/local.repo
[local]
name=local
baseurl=file:///root/user/sources
gpgcheck=0
enable=1
```

#/root/user/sources/为软件所在目录，其中包含配置文件repodata，若无此目录，运行yum install createrepo, createrepo /root/user/sources生成。

```
yum makecache
```

```
yum install java
```

```
yum install -y git git-review
```

#安装数据库

```
yum install -y mariadb.x86_64 mariadb-libs.x86_64 mariadb-server.x86_64
```

```
systemctl start mariadb
```

```
systemctl enable mariadb
```

#建立数据库，以及数据库用户

```
mysql -u root
```

```
create database gerritdb;
```

```
create user gerrit IDENTIFIED BY 'OStem@00';
```

```
grant all privileges on gerritdb.* to 'gerrit'@'localhost' identified by 'OStem@00';
```

```
grant all privileges on *.* to 'root'@'%' identified by 'OStem@00';
```

```
grant all privileges on *.* to 'root'@'localhost' identified by 'OStem@00';
```

```
FLUSH PRIVILEGES;
```

```
alter database gerritdb character set utf8;
```

```
exit
```

#安装gerrit

```
mkdir -p /etc/gerrit/gerrit_site
```

```
java -jar gerrit-2.12.3.war init -d /etc/gerrit/gerrit_site/
```

配置参数：

Location of Git repositories [git]:

\*\*\* SQL Database

\*\*\*

Database server type [mysql]: #写mysql

Server hostname [localhost]: # 默认localhost

Server port [(mysql default)]: # 默认

Database name [gerritdb]: # 写gerritdb

Database username [gerrit]: # 写gerrit

两次输入数据库密码

Type [LUCENE/?]: # 默认启用索引

The index must be rebuilt before starting Gerrit:

```
java -jar gerrit.war reindex -d site_path
```

\*\*\* User Authentication

\*\*\*

Authentication method [HTTP/?]: LDAP

{

LDAP server [ldap://localhost]: ldap://172.17.249.41:389

LDAP username : cn=chwei@sysnew,ou=MailUsers,dc=sysnew,dc=com

c's password : 123456

confirm password : 123456

Account BaseDN [DC=168,DC=54,DC=253]:ou=MailUsers,dc=sysnew,dc=com

Group BaseDN [ou=Cloud,dc=ctsi,dc=com,dc=cn]:

Enable signed push support [y/N]? n

}此部分可以暂不配置

\*\*\* Email Delivery

\*\*\*

SMTP server hostname [smtp.163.com]: 172.17.249.182 # 这里写smtp地址

SMTP server port [25]: #需要smtp server端口

SMTP encryption [NONE/?]:

SMTP username [jenkins\_ctsi@163.com]:wenzhang1@sysnew.com # 邮箱名

Change jenkins\_ctsi@163.com's password [y/N]? n # 输入两次授权码

Run as [root]: #运行用户，这里是选择root，如果有gerrit专用户，如gerrit，那么写gerrit.

```

Java runtime [/usr/lib/jvm/java-1.8.0-openjdk-1.8.0.102-1.b14.el7_2.x86_64/jre/]: # 默认
Upgrade gerrit_site/bin/gerrit.war [Y/n]? n # n
*** SSH Daemon
***
Listen on address [*]: # 默认
Listen on port [29418]: # 默认
*** HTTP Daemon
***
Behind reverse proxy [y/N]? n #n, 由于使用了ldap认证, 不需要这个, 这个是在http认证用的。
Use SSL (https://) [y/N]? n # n
Listen on address [*]: # 默认
Listen on port [8081]: # 默认是8080, 本次修改为8081
Canonical URL [http://172.7.140.140:8081/]: # 填写为http://192.9.100.189:8081/

Installing plugins.
Install plugin commit-message-length-validator version v2.12.3 [Y/n]? y
Install plugin download-commands version v2.12.3 [Y/n]? y
Install plugin replication version v2.12.3 [Y/n]? y
Install plugin reviewnotes version v2.12.3 [Y/n]? y
Install plugin singleusergroup version v2.12.3 [Y/n]? y
Initializing plugins.
No plugins found with init steps.
Initialized /etc/gerrit/gerrit_site

```

生成配置文件/etc/gerrit/gerrit\_site/etc/gerrit.config, 以及secure.config:

```

[gerrit]
  basePath = git
  canonicalWebUrl = http://172.17.140.140:8081/
[database]
  type = mysql
  hostname = localhost
  database = gerritdb
  username = gerrit
[index]
  type = LUCENE
[auth]
  type = LDAP
[ldap]
  server = ldap://172.17.249.41:389
  username = chwei@sysnew
  accountBase = ou=MailUsers,dc=sysnew,dc=com
  accountPattern = (&(objectClass=person)(sAMAccountName=${username}))
  accountSshUserName = sAMAccountName
  accountEmailAddress = mail
  accountFullName= ${username}
[receive]
  enableSignedPush = false
[sendemail]
  smtpServer = 172.17.249.182
  smtpUser = wenzhang1@sysnew.com
[container]
  user = root
  javaHome = /usr/lib/jvm/java-1.8.0-openjdk-1.8.0.102-1.b14.el7_2.x86_64/jre/
  war = /root/liyang/package_ci/gerrit-2.12.3.war
[sshd]
  listenAddress = *:29418
[httpd]
  listenUrl = http://*:8081/
[cache]
  directory = cache

```

secure.config主要是ldap账号密码:

```

[database]
  password = OStem@00

```

```
[ldap]
    password = ldap username的密码
[auth]
    registerEmailPrivateKey = RecDFuTC5PTn1UPGTJyv0hlysvjUbajkXSs=
```

依据上述两个文件的参数配置生成的gerrit.config以及secure.config。

```
#离线环境，缺包以及插件
cp mysql-connector-java-5.1.21.jar /etc/gerrit/gerrit_site/lib/
cp commit-message-length-validator.jar /etc/gerrit/gerrit_site/plugin/
cp delete-project.jar /etc/gerrit/gerrit_site/plugin/
cp download-commands.jar /etc/gerrit/gerrit_site/plugin/
cp reviewnotes.jar /etc/gerrit/gerrit_site/plugin/
cp singleusergroup.jar /etc/gerrit/gerrit_site/plugin/

#设置gerrit开机启动
cp /etc/gerrit/gerrit_site/bin/gerrit.sh /etc/init.d/gerrit
vim /etc/init.d/gerrit
    GERRIT_SITE=/etc/gerrit/gerrit_site
update-rc.d gerrit defaults 21
service gerrit restart
```

至此gerrit安装完毕，请阅读注意事项。

## Gerrit的项目迁移和同步

主要工作：同步gerrit以及gitlab。

依赖插件：replication.jar

```
#添加ssh
#第一个登陆的用户为管理员，此处为wenzhang1
#在本地为此用户生成key
ssh-keygen -t rsa -C "wenzhang1@sysnew.com"
#将id_rsa.pub的内容拷贝至wenzhang1用户的ssh public keys中
```

代码迁移：

```
#clone gitlab代码库中的代码
git config --global user.name "wenzhang1"
git config --global user.email "wenzhang1@sysnew.com"
git clone git@172.17.249.122:easystack-openstack/nova.git
cd nova

#编辑.gitreview
vim .gitreview
[gerrit]
host=172.17.140.140 # gerrit 服务器地址
port=29418 # gerrit 端口
project=nova.git #项目名称

# 添加.gitreview到版本库
git add .gitreview
git commit .gitreview -m 'Admin add .gitreview file.'

#在gerrit上创建项目
ssh -p 29418 wenzhang1@172.17.140.140 gerrit create-project nova

#添加gerrit server，推送gitlab的tags 到gerrit.
git remote add gerrit ssh://wenzhang1@172.17.140.140:29418/nova
git push --tags gerrit refs/remotes/origin/*:refs/heads/*
```

同步设置：

完成gerrit中项目的初始化设置，当开发者向gerrit中提交变更，并被接收并入gerrit本地库，此时需要将变更同步至gitlab库，配置同步，/etc/gerrit/gerrit\_site/etc/replication.config:

```
[remote "nova"]
projects = nova
url=git@172.17.249.122:easyStack-openstack/nova.git
push = +refs/heads/*:refs/heads/*
push = +refs/tags/*:refs/tags/*
push = +refs/changes/*:refs/changes/*
thread = 3
```

将本地root用户的id\_rsa.pub添加至gitlab。并且编辑~/.ssh/config文件：

```
vi ~/.ssh/config
Host 172.17.140.140
IdentityFile ~/.ssh/id_rsa
#更改config权限
chmod 600 ~/.ssh/config
#设置host-key
echo "StrictHostKeyChecking no" >> /etc/ssh/ssh_config #默认不添加；
或：
ssh-keyscan -t rsa 172.17.249.122 > ~/.ssh/known_hosts
ssh-keygen -H -f ~/.ssh/known_hosts
```

gerrit项目的group设置，批量创建group分配权限的脚本：

```
# setup-gerrit-access.sh
#!/bin/bash
set -x
#ssh -p 29418 qianli1@172.17.140.140 gerrit create-project $1
ssh -p 29418 wenzhang1@172.17.140.140 gerrit create-group $1
ssh -p 29418 wenzhang1@172.17.140.140 gerrit create-group $1-core
git init $1 ; cd $1
git remote add origin ssh://wenzhang1@172.17.140.140:29418/$1
git pull origin refs/meta/config
cat << EOF > project.config
[access "refs/heads/*"]
    abandon = group $1-core
    label-Code-Review = -2..+2 group $1-core
    submit = group $1-core
[access "refs/heads/*"]
    read = group $1-core
    read = group $1
EOF
cat << EOF > groups
`ssh -p 29418 wenzhang1@172.17.140.140 gerrit ls-groups -v | awk '{print $2 "\t" $1}' | grep -e '$'\t'$1`
EOF
git add --all
git commit -a -m 'Updated permissions'
git push origin HEAD:refs/meta/config
cd ..
```

运行./setup-gerrit-access.sh nova，为nova项目创建group并分配权限。效果如下图：

## Project nova

[Edit](#)

Rights Inherit From: All-Projects

Reference: refs/heads/*	
<b>Read</b> <div>ALLOW ▼ nova-core</div> <div>ALLOW ▼ nova</div>	<input type="checkbox"/> Exclusive
<b>Abandon</b> <div>ALLOW ▼ nova-core</div>	<input type="checkbox"/> Exclusive
<b>Label Code-Review</b> <div>-2 ▼ +2 ▼ nova-core</div>	<input type="checkbox"/> Exclusive
<b>Submit</b> <div>ALLOW ▼ nova-core</div>	<input type="checkbox"/> Exclusive

## 删除项目或组

删除项目可以通过界面操作，如下图：

project nova

[Clone](#) | [Clone with commit-msg hook](#) | [anonymous http](#) | [http](#) | [ssh](#) | [git clone ssh://wenzhang1@172.17.140.140:29418/nova](#)

**Description**

**Project Options**

State: Active ▼

Submit Type: Merge if Necessary ▼

Allow content merges: INHERIT (true) ▼

Create a new change for every commit not in the target branch: INHERIT (false) ▼

Require Change-Id in commit message: INHERIT (true) ▼

Maximum Git object size limit:

**Contributor Agreements**

Require Signed-off-by in commit message: INHERIT (false) ▼

[Save Changes](#)

**Project Commands**

Commands: [Delete...](#) [Run GC](#) [Create Change](#) [Edit Config](#)

Are you really sure you want to delete the project: "nova"?

☒ Delete project even if open changes exist?

☐ Preserve GIT Repository?

[Delete](#)

若要保留本地库，勾选第二个。

删除group需要通过操作数据库进行删除：

```
#如删除nova组
use gerritdb;
delete from account_group_names where name='nova';
delete from account_groups where name='nova';
```

重启gerrit。

## Gerrit的注意事项

1. 第一个登陆的用户为管理员，管理员若要在本地通过ssh登录gerrit，将本身的key添加至Gerrit；
2. 在利用脚本进行批量创建group时，需要删除本地已存在的同名项目，否则会报出author doesnot match错误。
3. 数据库编码需配置成utf8格式，否则在gerrit界面中无法显示fullname，运行：  
alter database gerritdb character set utf8;
4. 每个用户通过ldap认证，但是会在数据库中记录此用户名，以及密码。删除用户在本地数据库的记录  
删除表account\_external\_ids中的用户记录，

#查询用户id:

```
select account_id from accounts where preferred_email="wenzhang1@sysnew.com";
```

```
delete from account where account_id=2;
```

```
delete from account_external_ids where account_id=2; #删除了account_id为2的用户，
```

```
delete from account_group_members where account_id="2" ;
```

```
delete from account_group_members_audit where account_id="2";
```

5. easystack-novnc项目，无法推送至gerrit，错误：HEAD引用一个不存在的tree。

解决方法：git clone --bare git@url/novnc.git;

将此novnc.git拷贝至gerrit本地库/etc/gerrit/gerrit\_site/git/目录下，重启gerrit。

配置replicaation，在web界面手动创建group，并分配权限，点击save shanges。

# Jenkins的安装和配置

## master节点的安装配置(Node1)

Jenkins版本	jenkins-1.625.3	本地源安装，rpm安装包
依赖包	gcc gcc-c++ autoconf glibc glibc-devel curl curl-devel ncurses ncurses-devel	银联源
依赖包	>=JDK1.8	本地源
插件	git, cvs, ldap, etc。	离线包免安装，有离线包

## 安装步骤

```
#master节点的安装
yum install gcc gcc-c++ autoconf glibc glibc-devel curl curl-devel ncurses ncurses-devel
yum install rpm-build
yum install wget
yum install jenkins.noarch
yum install java
systemctl start jenkins
systemctl enable jenkins
#关闭防火墙
systemctl stop firewalld.service
#清除iptables
iptables -F
iptables -X
#查看8080端口是否运行
netstat -anlp | grep 8080
systemctl status httpd
systemctl status jenkins
```

#安装插件

#备份原有的插件

```
mv /var/lib/jenkins/plugins/ /var/lib/jenkins/plugins_bak
```

#安装插件

```
cp -r ${pri-dir}/plugins/ /var/lib/jenkins/ #pri-dir待安装插件的所在目录
```

```
chown -R jenkins:jenkins /var/lib/jenkins/plugins
```

#加载jobs

#拷贝jobs

```
cp -r ${pri-dir}/jobs/* /var/lib/jenkins/jobs/
```

```
chown -R jenkins:jenkins /var/lib/jenkins/jobs

systemctl restart jenkins
#界面操作
点击系统管理->读取设置，加载job。

#拷贝项目打包脚本和项目代码
#项目的打包脚本存放于/var/lib/jenkins/userContent/easystack-build,
#项目打包的rpm配置文件存放于/var/lib/jenkins/userContent/easystack-package,

#项目的源码以及rpm在目录/var/lib/jenkins/userContent/openstack/
#以及/var/lib/jenkins/userContent/openstack/RPMS/下。

#拷贝三个文件夹easystack-build, easystack-package, openstack
cp -r ${pri-dir}/liberty_rpms/* /var/lib/jenkins/userContent/

chown -R jenkins:jenkins easystack-build
chown -R jenkins:jenkins estack-packages
chown -R jenkins:jenkins openstack

#安装必要的rpm依赖包
for i in `cat rpm-requirements.txt`;do echo "begin to install $i...";yum install -y $i;done;

#项目源码
#需要clone源码至openstack文件夹下，由于job的运行过程中需要git fetch,
#因此需要将本地的key上传至代码平台上

#基于gitlab，因此需要将key上传至gitlab中
#root用户下：
ssh-keygen -t rsa -C "${user}@sysnew.com"
#upload id_rsa.pub

#更改jenkins的配置文件。利用root用户运行jenkins
vi /etc/sysconfig/jenkins
    JENKINS_HOME="/var/lib/jenkins"    #line 2
    JENKINS_USER="root"                #line 10
#

cd /var/lib/jenkins/userContent/openstack
git clone ${repo-url}

#用户设置
开启用户注册功能，点击 -> 系统管理 -> Configure Global Security -> 勾选启用安全，
保存后，会自动跳转到登录页面，点击右上角注册按钮
用户注册
输入管理员信息，完成注册
为了安全，设置 Jenkins 不对普通用户开放登录权限，只有管理员可以设置、构建任务，
普通用户可以查看任务状态。点击系统管理 ->Configure Global Security -> 去掉开放用户注册勾。

#配置smtp服务器
在jenkins面板“系统设置下”。SMTP: 172.17.249.182 port:25默认端口，详细配置如下图：
```



### 邮件通知

SMTP服务器	172.17.249.182
用户默认邮件后缀	@sysnew.com
<input checked="" type="checkbox"/> 使用SMTP认证	
用户名	wenzhang1@sysnew.com
密码	.....
使用SSL协议	<input type="checkbox"/>
SMTP端口	25
Reply-To Address	wenzhang1@sysnew.com
字符集	UTF-8
<input type="checkbox"/> 通过发送测试邮件测试配置	

### SCM Polling


Max # of concurrent polling	
-----------------------------	--

### slave节点的安装配置(Node2)

```
#安装必要的依赖
yum install java openssh-server ant createrepo

#添加用户，matser利用此用户管理slave节点。也可以直接利用root用户
useradd -m jenkins -d /home/jenkins
passwd jenkins

#jenkins管理界面添加slave节点
系统管理->管理节点->新建节点->选择Dumb，填好名字，确定->如下设置。
```

Name	<input type="text" value="iso-server"/>		
描述	<input type="text"/>		
# of executors	<input type="text" value="1"/>		
远程工作目录	<input type="text" value="/home/jenkins"/>		
标签	<input type="text" value="iso-server"/>		
用法	<input type="text" value="尽可能的使用这个节点"/>		
启动方法	<input type="text" value="Launch slave agents on Unix machines via SSH"/>		
	Host	<input type="text" value="172.17.140.141"/>	
	Credentials	<input type="text" value="root/*****"/>	<div> Add ▼</div>
Availability	<input type="text" value="Keep this slave on-line as much as possible"/>		

#### Node Properties

- ☐ Environment variables
- ☐ Prepare jobs environment
- ☐ Tool Locations

其中credentials是master连接slave节点的凭证。可以通过add添加多种验证方式。

点击add, 添加认证方法, 有添加密钥和用户名密码的两种方式。

##### 1. 添加密钥的方式

```
ssh-keygen -t rsa
```

```
cd .ssh
```

```
cat id_rsa.pub >> authorized_keys
```


```
chmod 700 authorized_keys
```


```
scp id_rsa root@172.17.140.140:/var/lib/jenkins/slavekeys/
```

即将id\_rsa(相当于privatekey) 拷贝到jenkins master机器上, 首先要在master上新建/var/lib/jenkins/slavekeys

然后chown -R jenkins:jenkins /var/lib/jenkins/slavekeys

##### 2. 用户名密码的方式

 **Jenkins Credentials Provider: Jenkins**

 **Add Credentials**

Domain

Global credentials (unrestricted)

Kind

Username with password

Scope

Global (Jenkins, nodes, items, all child items, etc)

Username

jenkins

Password

.....

ID

Description

Add

Cancel

Kind选择Username with password，添加slave节点上用户名和密码。

保存，启动节点检查设置即可。

```
#slave节点上安装一些依赖包
yum install gem
yum install PyYAML
yum install python-jinja2.noarch
yum install docker
systemctl status docker
systemctl start docker
yum install daemon
yum install python-daemon.noarch
systemctl stop firewalld.service
yum install puppet
yum install lrzip

{
mkdir /iso-builder/
rsync -a ~/roller/estack-main /iso-builder/
cd /iso-builder/
cd estack-main/
rsync -a /local_mirror ./
}#此段/iso-builder目录，视job的配置文件而定
```

## jenkins注意事项

出现网络连接问题。172.18.0.1，此是docker的网络。关闭防火墙。  
出现puppet问题，安装puppet，unknown function parseyaml，selinux必须关闭。

make-iso会需要的npm包：npm: iso-builder/estack-main/build/repos/nailgun/nailgun/npm-shrinkwrap.json，列出了所需的所有的npm包。  
两种方法解决：

1. 安装一个本地源
2. 安装npm是为了产生nailgun.tar包，可以在联网环境中预先制作nailgun.tar包，更改脚本，注释下载继安装npm包的脚本

编辑config.mk，脚本中设计到对项目代码的clone更细，需要用gitlab的项目url替代，主要位于：#Repos and Versions注释下。

```

Subject: [PATCH] Make nailgun static files offline

Change-Id: I8028b439e856ac3bc6b13328077c665e157801c7
---
 packages/rpm/module.mk          | 4 ++--
 packages/rpm/nailgun-static.tar | Bin 0 -> 1320960 bytes
 2 files changed, 2 insertions(+), 2 deletions(-)
 create mode 100644 packages/rpm/nailgun-static.tar

diff --git a/packages/rpm/module.mk b/packages/rpm/module.mk
index 53dee78..c0fb06a 100644
--- a/packages/rpm/module.mk
+++ b/packages/rpm/module.mk
@@ -25,9 +25,9 @@ $(BUILD_DIR)/packages/rpm/$1.done: $(BUILD_DIR)/packages/rpm/sources/$1/$2
 $(BUILD_DIR)/packages/rpm/sources/$1/$2: $(call find-files,$3)
     mkdir -p $(BUILD_DIR)/packages/rpm/sources/$1
 ifeq ($1,nailgun)
-     cd $3 && npm install && grunt build --static-dir=compressed_static
+     cd $3
+     rm -rf $3/static
+     mv $3/compressed_static $3/static
+     tar -xf $(TOP_DIR)/packages/rpm/nailgun-static.tar

```

更改后的module.mk脚本如图:

```

# (eval (call prepare_file_source,package_name,file_name,source_path))
define prepare_file_source
$(BUILD_DIR)/packages/rpm/$1.done: $(BUILD_DIR)/packages/rpm/sources/$1/$2
$(BUILD_DIR)/packages/rpm/sources/$1/$2: $(call find-files,$3)
    mkdir -p $(BUILD_DIR)/packages/rpm/sources/$1
    cp $3 $(BUILD_DIR)/packages/rpm/sources/$1/$2
endef

# Usage:
# (eval (call prepare_python_source,package_name,file_name,source_path))
define prepare_python_source
$(BUILD_DIR)/packages/rpm/$1.done: $(BUILD_DIR)/packages/rpm/sources/$1/$2
$(BUILD_DIR)/packages/rpm/sources/$1/$2: $(call find-files,$3)
    mkdir -p $(BUILD_DIR)/packages/rpm/sources/$1
ifeq ($1,nailgun)
    cd $3
    rm -rf $3/static
    cd $3 && tar -xf $(TOP_DIR)/packages/rpm/nailgun-static.tar
endif
    cd $3 && python setup.py sdist -d $(BUILD_DIR)/packages/rpm/sources/$1
endef

# Usage:
# (eval (call prepare_tgz_source,package_name,file_name,source_path))

```

需要将现有的nailgun.tar包防置在/iso-builder/estack-main/packages/rpm下。

jenkins的任务配置和说明详见文档: [CI-jenkins的任务配置和说明.pdf](#)。

实际的job功能与文档略有不同:

其中: Build\_and\_copy\_all\_rpms\_es\_liberty总的job包含两个子job:

Build\_and\_copy\_os\_rpms\_es\_liberty, Build\_and\_copy\_roller\_rpms\_es\_liberty。

执行Build\_and\_copy\_all\_rpms\_es\_liberty,在执行make\_iso job完成iso的制作。

Build\_and\_copy\_os\_rpms\_es\_liberty:包含整个iso制作的参数,包括每个项目的版本,rpm包的存放位置,等等。

也可以手动执行每个项目的job,在运行opy\_RPMS\_to\_ISO\_server,将rpm包复制至slave节点,运行make\_iso。