Jenkins + Npm 编译 js 代码并部署

Name：曲中岭

Email:zlingqu@126.com

Q Q :441869115

第一章 部署准备

1.1 目的

像 java 项目一样，研发人员只需要提交前端 js 源代码到 svn，即可使用 jenkins 编译打包并部署，而不需要手工编译。

1.2 规划

OS ：Ubuntu 14.04.4 x64
IP ：172.16.7.47
JDK ：1.8.0_171
Jenkins ：2.56
nodejs ：8.12.0
npm ：6.4.1

1.3 tomcat 安装

用来运行 jenkines，步骤略

第二章 node 和 npm 安装

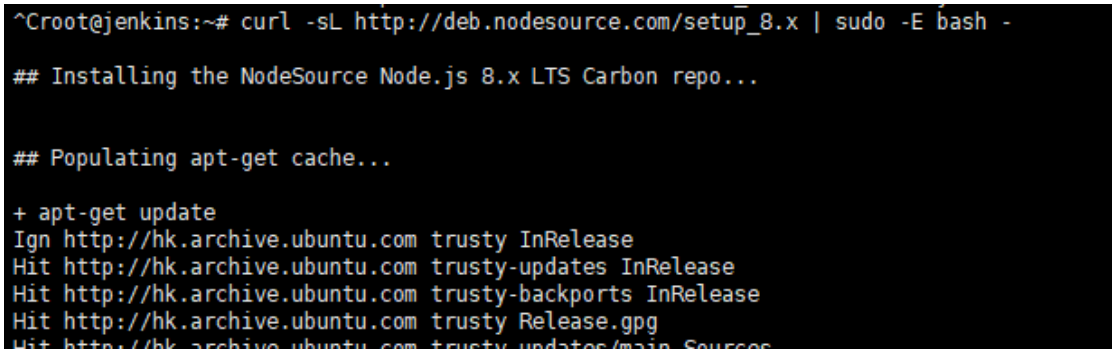
node 和 npm，类似于 python 中 python 和 pip 的关系，前者表示语言环境，后者表示包管理工具。

安装方式有很多，我这里只选择其中一种，使用源安装的方式。

2.1 添加源

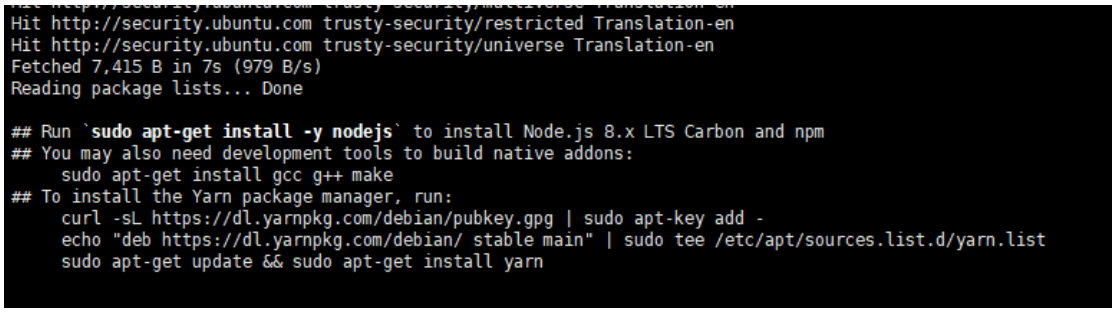
使用如下命令添加 8 版本的源:

```
curl -sL https://deb.nodesource.com/setup_8.x | sudo -E bash –
```



其中 setup_8.x 的 8 表示大版本，当前 10 是最新的，当然也可以换成 9、10 之类的，不过最新版 10 不支持 ubuntu14 了。

安装完成提示如下:



2.2 安装

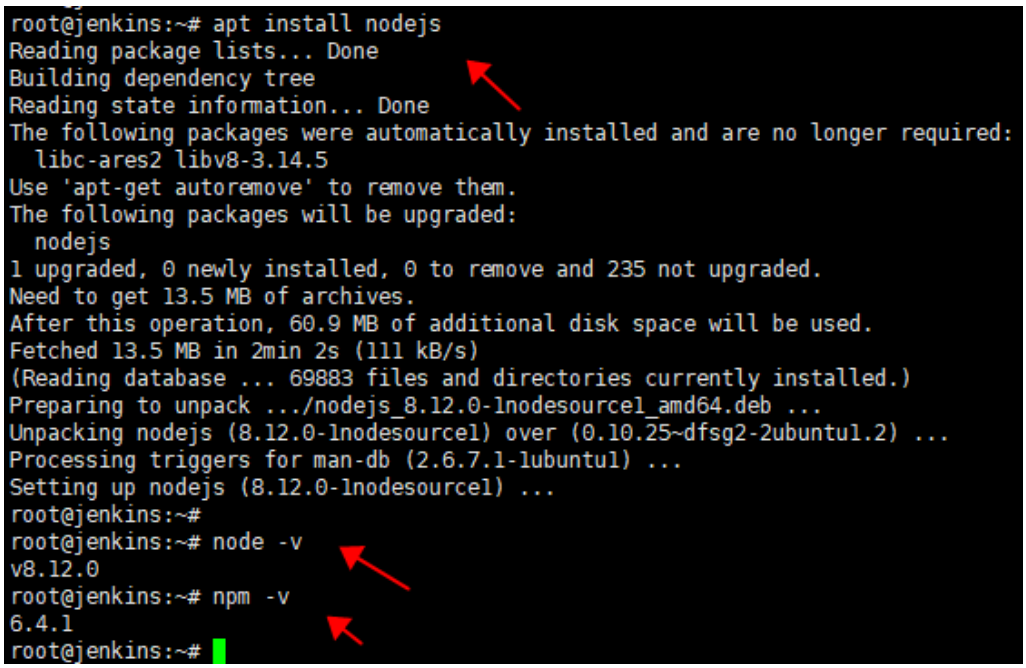
使用 apt 直接安装 nodejs

```
apt install nodejs
```

安装后，使用以下命令查看

```
node –v
npm –v
```

如下图所示:



如果使用 CentOS，安装方式如下:

curl --silent --location https://rpm.nodesource.com/setup_10.x | bash -

#10 表示大版本，可换乘 8、9 等

yum install -y nodejs

2.3 替换源（可选）

默认使用官方的源进行下载依赖等，可能比较慢。

查看当前的源:

```
npm get registry
```

切换成淘宝源:

```
npm config set registry http://registry.npm.taobao.org/
```

切换回官方源:

```
npm config set registry https://registry.npmjs.org/
```

2.4 其他

如果 npm 不是最新版，使用如下命令升级

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

```
    install
root@testserver01:~# npm install npm@latest -g
/usr/bin/npm -> /usr/lib/node_modules/npm/bin/npm-cli.js
/usr/bin/npx -> /usr/lib/node_modules/npm/bin/npx-cli.js
+ npm@6.4.1
updated 1 package in 22.125s
root@testserver01:~#
root@testserver01:~#
root@testserver01:~#
root@testserver01:~# npm -v
6.4.1
root@testserver01:~#
```

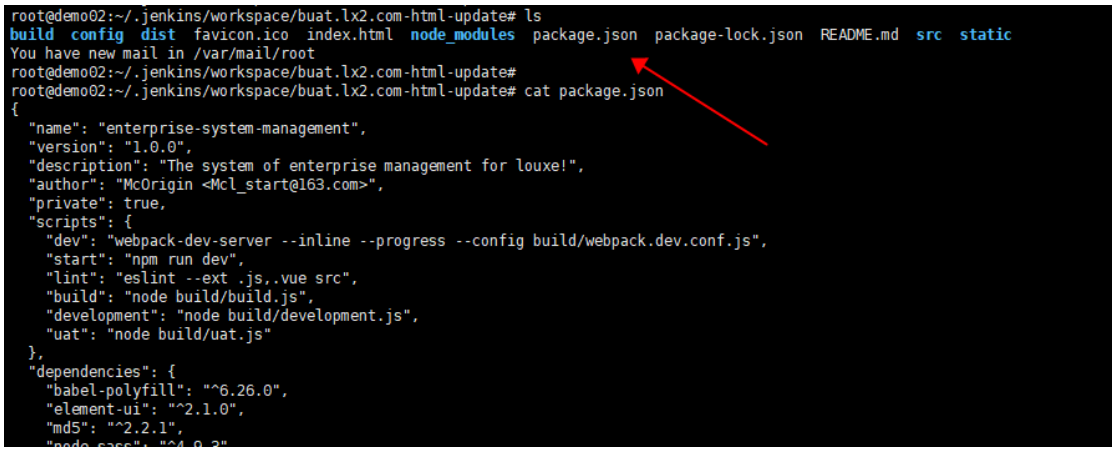
我这里初次安装，默认已经是最新版。

更多 npm 命令参考: https://blog.csdn.net/u014291497/article/details/75193865

第三章 安装依赖模块

3.1 依赖包介绍

如下图，是项目源码目录，文件 package.json 中写了所需要的依赖包，每个项目所需要的依赖是不同的。



3.2 安装依赖包

依赖包的安装必须在项目当前目录下操作。默认安装到 node_modules 目录，且该目录不能使用软连接等，而 jenkins 可能清空当前目录，如果每次都下载，会很费时。

解决方法是手动下载后备份到上层目录，jenkins 中构建时再拷贝进来。

```
cd /root/.jenkins/workspace/buat.lx2.com-html-update
npm install
cp -ar node_modules ../node_modules_for_www.lx2.com_html
```

使用 npm install 是批量安装所需要的所有依赖，有些包可能不能被安装，比如 node-sass，这时要单独安装

```
npm install node-sass
```

第四章 jenkins 配置

如上图，包含两部分。

4.1 构建部分

配置如下：

```
rm -rf node_modules
cp -ar ../node_modules_for_emng.lx2.com_html node_modules
npm run uat
```

先移除 node_modules
再拷贝已经备份好的 node_modules 到当前目录
再使用

```
npm run uat
```

进行构建，其中 uat 表示测试环境，可能还有 pro(生产)、dev(开发)等，具体看项目结构。
可在 build 目录中看到相关的 js 文件。

```
root@demo02:~/.jenkins/workspace/buat.lx2.com-html-update# ll build/
total 72
drwxr-xr-x 2 root root 4096 Oct 12 15:28 ./
drwxr-xr-x 9 root root 4096 Oct 25 15:37 ../
-rw-r--r-- 1 root root 1198 Oct 12 15:28 build.js
-rw-r--r-- 1 root root 1290 Oct 12 15:28 check-versions.js
-rw-r--r-- 1 root root 1165 Oct 12 15:28 development.js
-rw-r--r-- 1 root root 6849 Oct 12 15:28 logo.png
-rw-r--r-- 1 root root 1153 Oct 12 15:28 uat.js
-rw-r--r-- 1 root root 2580 Oct 12 15:28 utils.js
-rw-r--r-- 1 root root  546 Oct 12 15:28 vue-loader.conf.js
```

这种配置，如果需要新的依赖，还是需要手动执行 3.1、3.2 步的。
比如下面：

```
Error: Cannot find module 'eslint-friendly-formatter'
    at Function.Module._resolveFilename (internal/modules/cjs/loader.js:571:15)
    at Function.Module._load (internal/modules/cjs/loader.js:497:25)
    at Module.require (internal/modules/cjs/loader.js:626:17)
    at require (internal/modules/cjs/helpers.js:20:18)
    at createLintingRule (/root/.jenkins/workspace/buat.lx2.com-html-update/build/webpack.base.c
```

这时可使用

```
npm install eslint-friendly-formatter
```

安装指定依赖。

构建后会生成 dist 目录，这就是构建的结果，也是下一步部署需要操作的文件部分。

```
root@demo02:~/.jenkins/workspace/buat.lx2.com-html-update# ll dist/
total 24
drwxr-xr-x 3 root root 4096 Oct 25 15:58 ./
drwxr-xr-x 9 root root 4096 Oct 25 16:04 ../
-rw-r--r-- 1 root root 4286 Oct 25 15:58 favicon.ico
-rw-r--r-- 1 root root  931 Oct 25 15:58 index.html
drwxr-xr-x 8 root root 4096 Oct 25 15:58 static/
```

4.1 部署部分

使用 ansible 脚本部署

```
cd /root/playbook_pre_by_quzl
ansible-playbook -i hosts buat.lx2.com-html.yml -e src_dir=$WORKSPACE/dist
```

第五章 执行

5.1 执行效果

在命令行界面执行

```
npm run uat
```

的效果如下:

```
                static/js/21.04699d4c699986117fda.js.gz    6.93 kB          [emitted]
                static/js/29.7cb962ceef67860e0f25.js.gz    4.88 kB          [emitted]
                static/js/45.80d472cf1b13754793dc.js.gz    3.83 kB          [emitted]
                static/js/57.5493bf9bddd2d8d78985.js.gz    4.25 kB          [emitted]
                static/js/33.728ebe4e29f5fcdbaa60.js.gz    5.14 kB          [emitted]
                static/js/59.13414201267ff8d02a6a.js.gz    5.16 kB          [emitted]
                static/js/49.a18a31896df4464f43c0.js.gz      15 kB          [emitted]
               static/js/app.f07e3ab792149899c8b1.js.gz    10.6 kB          [emitted]
static/css/app.2029ef7b697b57b115a1e583e0cfb59.css.gz      128 kB          [emitted]
          static/js/vendor.458b88db3bf2e84274b8.js.gz      188 kB          [emitted]

  Build complete.

  Tip: built files are meant to be served over an HTTP server.
  Opening index.html over file:// won't work.
```

在 jenkins 中，先编译打包，再部署，效果如下:

```
□[1m□[32mstatic/js/vendor.458b88db3bf2e84274b8.js.gz□[39m□[22m    188 kB       □[1m□[39m□[22m  □[1m□[32m[emitted]□[39m□[22m

  Build complete.

  Tip: built files are meant to be served over an HTTP server.
  Opening index.html over file:// won't work.

[buat.lx2.com-html-update] $ /bin/sh -xe /data/tomcat-jenkins-8080/temp/jenkins1374154089461054691.sh
+ cd /root/playbook_pre_by_quzl
+ ansible-playbook buat.lx2.com-html.yml -e src_dir=/root/.jenkins/workspace/buat.lx2.com-html-update/dist

PLAY [uat-nginx] ***************************************************************

TASK [1 sync file to dest /data/louxe/ent_portal_html] *************************
changed: [172.16.7.46]

PLAY RECAP *********************************************************************
172.16.7.46                : ok=1    changed=1    unreachable=0    failed=0

Finished: SUCCESS
```

5.2 ansible 脚本内容

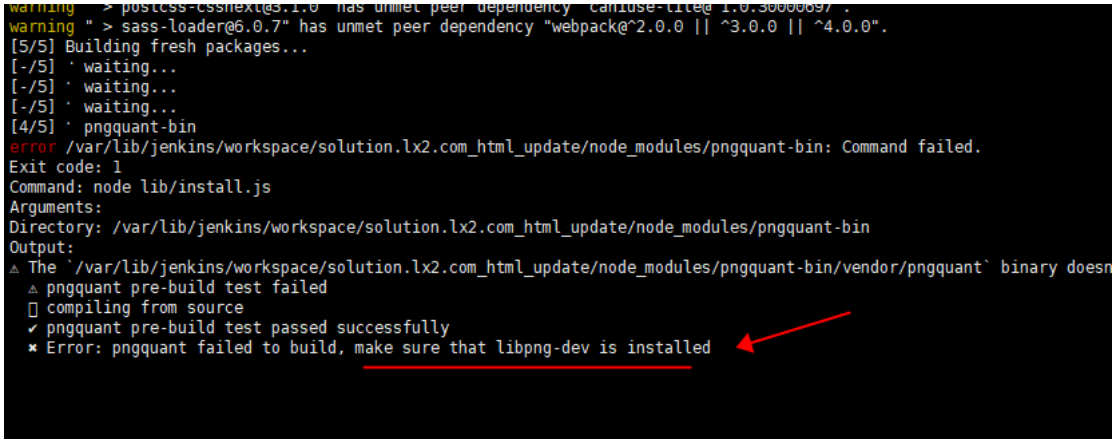
root@demo02:~/playbook_pre_by_quzl# cat buat.lx2.com-html.yml

```
---
- hosts: uat-nginx
  gather_facts: false
  remote_user: root
  vars:
    - dest_dir:   /data/louxe/ent_portal_html/dist
  tasks:
   - name: 1 sync file to dest {{dest_dir}}
```

```
synchronize:
  src={{src_dir}}/
  dest={{dest_dir}}
  delete=yes
  recursive=yes
  rsync_opts="--exclude=.svn"
```

更多 ansible 内容，不是本章节的内容，这里不再介绍

报错：make sure that libpng-dev is installed
如下图：



解决方法：

```
yum install libpng-devel
apt install libpng-dev
```

第六章 yarn 代替 npm

yarn 和 npm 一样，也是 node 的包管理工具，yarn 是后来推出的，更加好用。
yarn 的优势：
1）引入 yarn.lock 文件来管理依赖版本问题，保证每次安装都是一致的。
2）缓存加并行下载保证了安装速度。使用 yarn=npm install 安装依赖时，会自动缓存到服务器一份，下次安装可直接使用，而不用去互联网上下载。
二者的区别：
https://www.jianshu.com/p/254794d5e741

yarn 的安装：
https://yarn.bootcss.com/docs/install/#centos-stable

一个使用 yarn 的例子(楼小二解决方案项目)，该项目不同于一般的纯静态 js 前端，而是使用一个服务来运行 nodejs:
jenkiens 配置：

```
yarn
yarn run build
yarn run tar

cd /root/playbook-pro-by-quzl  #发布
ansible-playbook -i hosts solution.lx2.com-html.yml -e "host_name=$env_choice pname=$pname src_dir=$WORKSPACE/deloy/dist"
```

第七章 补充

ansible 脚本内容:

[root@iZuf61fwacpba8u3ty82udZ playbook-pro-by-quzl]# cat solution.lx2.com-html.yml

```
---
- hosts: "{{host_name}}"
   gather_facts: false
   remote_user: root
   vars:
#通用脚本，变量通过-e 传递进来，例如 ansible-playbook -i hosts update-html.yml -e "pname=$pname src_dir=$WORKSPACE"
       - dest_dir:    /data/www/{{pname}}
       - back_dir: /data/backup
   tasks:
     - name: 1 backup html
        shell: cp -r /data/www {{back_dir}}/www.`date +"%Y-%m-%d-%H-%M"`
        ignore_errors: yes
     - name: 2 stop npm server
        shell: cd {{ dest_dir }} && npm run stop
        ignore_errors: yes
     - name: 3 sync file to dest {{dest_dir}}
        synchronize:
           src={{src_dir}}/
           dest={{dest_dir}}
           delete=yes
           recursive=yes
           rsync_opts="--exclude=.svn"
     - name: 4 start npm server
        shell: cd {{ dest_dir }} && yarn install --prod && npx nodeinstall --install-alinode ^3 && npm run start
```