Vue 项目打包部署总结

mp.weixin.qq.com/s/xwqPuZ5KiEFqC8Wfey8WQQ

↓推荐关注↓

大前端技术之路
分享Web前端，Node.js、React Native等大前端技术栈精选

18篇原创内容

公众号

使用Vue做前后端分离项目时，通常前端是单独部署，用户访问的也是前端项目地址，因此前端开发人员很有必要熟悉一下项目部署的流程与各类问题的解决办法了。

Vue项目打包部署本身不复杂，不过一些前端同学可能对服务器接触不多，部署过程中还是会遇到这样那样的问题。本文介绍一下使用nginx服务器代理前端项目的方法以及项目部署的相关问题，内容概览：

一、准备工作——服务器和nginx使用

1. 准备一台服务器

我的是ubuntu系统，linux系统的操作都差不多。没有服务器怎么破？
如果你只是想体验一下，可以尝试各大厂的云服务器免费试用套餐，比如华为云免费试用，本文相关操作即是在华为云上完成的。
不过如果想时常练练手，我觉得可以购买一台云服务器，比如上面的华为云或者阿里云都还挺可靠。

2. nginx安装和启动

轻装简行，这部分不作过多赘述（毕竟网上相关教程一大堆），正常情况下仅需下面两个指令：

```
# 安装，安装完成后使用nginx -v检查，如果输出nginx的版本信息表明安装成功sudo apt-get install nginx# 启动sudo service nginx start
```

启动后，正常情况下，直接访问 http://服务器ip 或 http://域名 （本文测试用的服务器没有配置域名，所以用ip，就本文而言，域名和ip没有太大区别）应该就能看到nginx服务器的默认页面了——如果访问不到，有可能是你的云服务器默认的http服务端口（80端口）没有对外开放，在服务器安全组配置一下即可。

3、 了解nginx: 修改nginx配置，让nginx服务器代理我们创建的文件

查看nginx的配置，linux系统下的配置文件通常会存放在/etc目录下，nginx的配置文件就在/etc/nginx文件夹，打开文件/etc/nginx/sites-available/default（nginx可以有多个配置文件，通常我们配置nginx也是修改这个文件）：

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```


可以看到默认情况下，nginx代理的根目录是/var/www/html，输入 http://服务器ip会访问这个文件夹下的文件，会根据index的配置值来找默认访问的文件，比如index.html、index.htm之类。

我们可以更改root的值来修改nginx服务代理的文件夹：

1）、创建文件夹/www，并创建index.html，写入"Hello world"字符串

```
mkdir /wwwecho 'Hello world' > /www/index.html
```

2）、修改root值为 /www

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /www;
        index index.html index.htm index.nginx-debian.html;

        server_name _;
        location / {
                try_files $uri $uri/ =404;
        }
}
```


3）、sudo nginx -t 检查nginx配置是否正确

```
root@ecs-s6-large-2-linux-20200108135338:~# sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ecs-s6-large-2-linux-20200108135338:~#
root@ecs-s6-large-2-linux-20200108135338:~#
```

successful说明成功

4）、加载nginx配置：sudo nginx -s reload
再次访问页面，发现页面内容已经变成了我们创建的index.html:

二、Vue项目打包同步文件到远程服务器

1、 打包

默认情况下，使用vue-cli创建的项目，package.json里的script应该已经配置了build指令，直接执行yarn build 或者 npm run build即可。

2、 同步到远程服务器

我们使用nginx部署Vue项目，实质上就是将Vue项目打包后的内容同步到nginx指向的文件夹。之前的步骤已经介绍了怎样配置nginx指向我们创建的文件夹，剩下的问题就是怎么把打包好的文件同步到服务器上指定的文件夹里，比如同步到之前步骤中创建的/www。
同步文件可以在git-bash或者powershell使用scp指令，如果是linux环境开发，还可以使用rsync指令:

```
scp -r dist/* root@117.78.4.26:/www或rsync -avr --delete-after dist/*
root@117.78.4.26:/www
```

注意这里以及后续步骤是root使用用户远程同步，应该根据你的具体情况替换root和ip(ip换为你自己的服务器IP)。
为了方便，可以在package.json脚本中加一个push命令，以使用yarn为例（如果你使用npm，则push命令中yarn改成npm run即可）：

```
"scripts": {    "build": "vue-cli-service build",    "push": "yarn build && scp -r
dist/* root@117.78.4.26:/www"  },
```

这样就可以直接执行yarn push 或者npm run push直接发布了。不过还有一个小问题，就是命令执行的时候要求输入远程服务器的root密码（这里使用root来连接远程的，你可以用别的用户，毕竟root用户权限太高了）。
为了避免每次执行都要输入root密码，我们可以将本机的ssh同步到远程服务器的authorized_keys文件中。

3、 同步ssh key

1. 生成ssh key：使用git bash或者powershell执行ssh-keygen可以生成ssh key。会询问生成的key存放地址，直接回车就行，如果已经存在，则会询问是否覆盖：

```
Administrator@XS-20180324OTTK MINGW64 /d/vue-publish-test (master)
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/Administrator/.ssh/id_rsa):
/c/Users/Administrator/.ssh/id_rsa already exists.
Overwrite (y/n)? n
```


2. 同步ssh key到远程服务器，使用ssh-copy-id指令同步

```
ssh-copy-id -i ~/.ssh/id_rsa.pub root@117.78.4.26
```

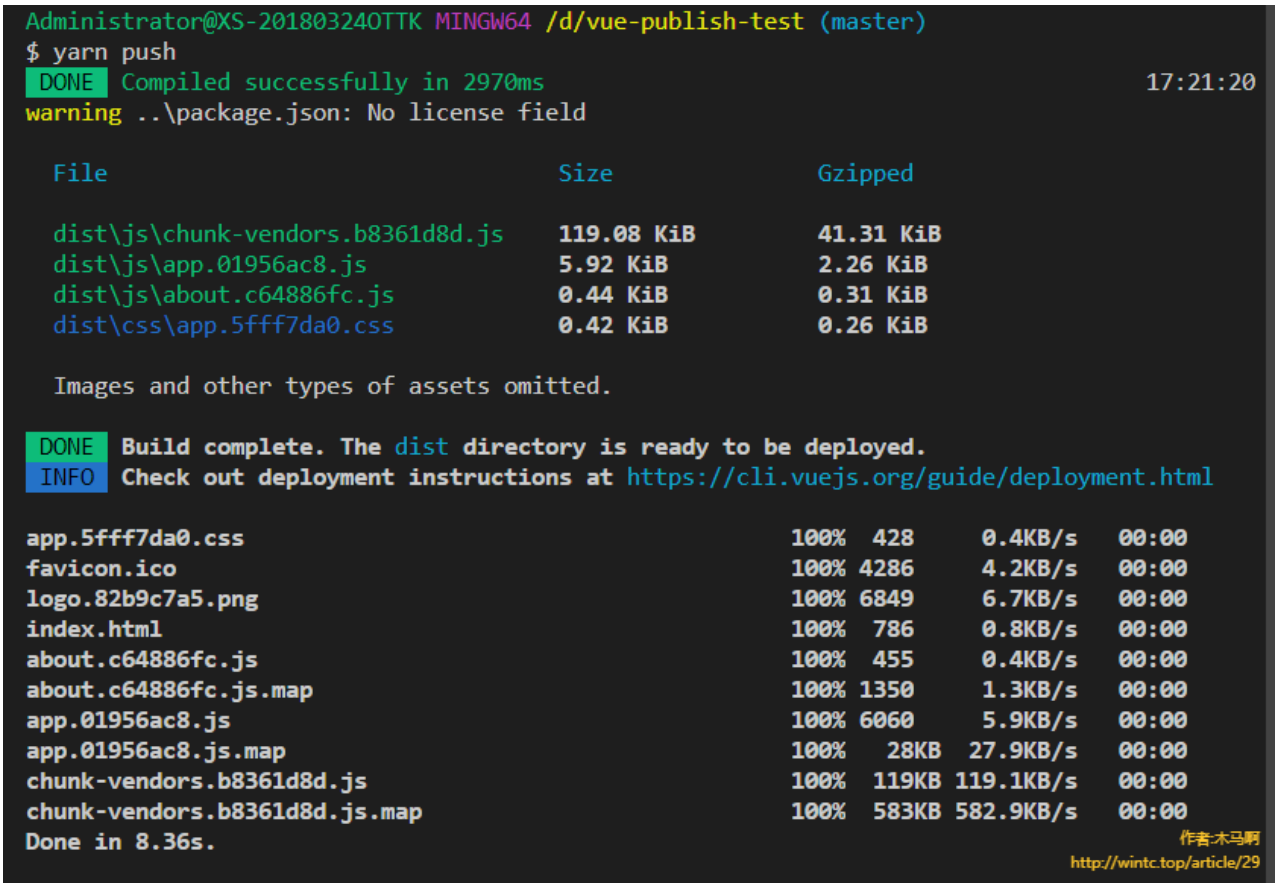
```
$ ssh-copy-id -i ~/.ssh/id_rsa.pub root@117.78.4.26
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
 out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
root@117.78.4.26's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@117.78.4.26'"
and check to make sure that only the key(s) you wanted were added.
```


输入密码后，之后再次同步就不需要输入密码了。其实ssh_key是同步到了服务器（此处是root用户家目录）~/.ssh/authorized_keys文件里：

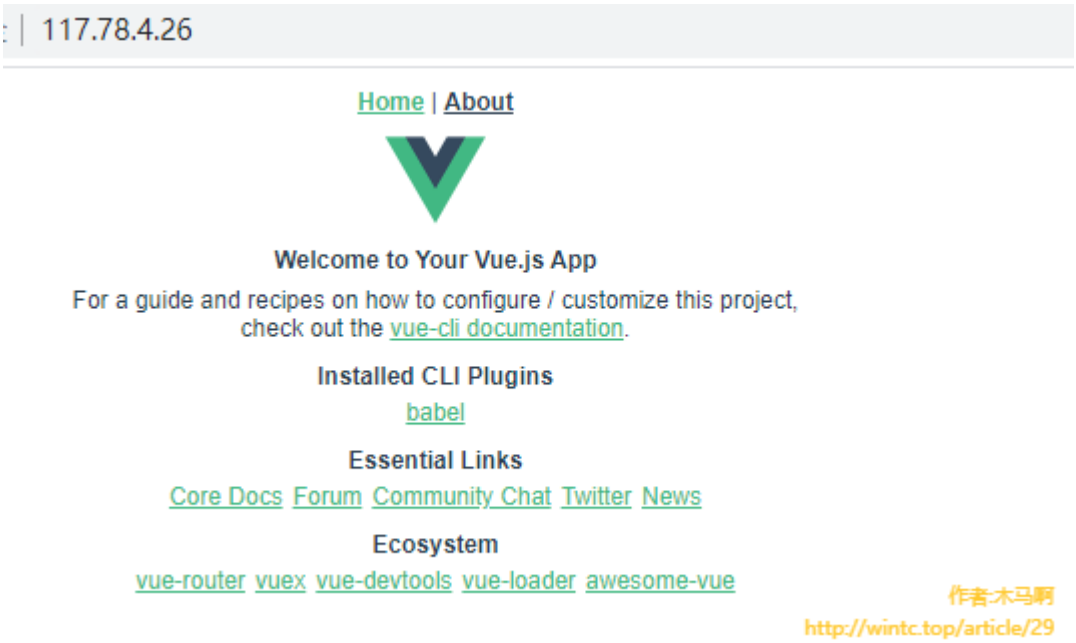
```
root@ecs-s6-large-2-linux-20200108135338:~#
root@ecs-s6-large-2-linux-20200108135338:~# cat ~/.ssh/authorized_keys 同步ssh-key前

root@ecs-s6-large-2-linux-20200108135338:~# cat ~/.ssh/authorized_keys 同步后

ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQC3m8QpCrSNPEjHfHn2CaoS1K9jBGSWjFAOtxKdWtYIfsCnpjuK91R
jtHrUy5zEkb8gcpbY9GcyTNbEjOc9OqF6ufroe3g4g96BPkP5cy/Rc1sX6mF0fgOJibFm54yFMGVPj3InqMzJtTVU+r
Q5+TRP1clzF2/+Ol2WwyGNpWzEQficCsv4JtAOIQaH3z4+av+LsdlIU4T44DOwn3/fHJcjRIlzSkzsCDe8DGnmSOBkM
RuAdbTpRxOsXT+xuxZchwMMdPCvYXqbx66+hHLAVRS2qlEAKkSTdUTINt5C4teVdCiTELYn897LBS+4Dwx4hCRqRJeU
dcrvE3ZGAlJxGc1kIlja07cMK7LpWUHAgRBdaIB++R9E7X9BmEJE70E32tYUyIQWojWcsdjuNzjvE/OjOhipjn+t5K6
W7E9/CqXbECgBFVz1ko28T/v/5X09o7R7Mx8WPC8H8Kgqt1JkXeqLe/vxmfmBayYiKy4NjjjBPOOMOUFlEGDOghkJgJ
g7dXeU0KSeWn53MkiMUAauzIuxkPTBvcjTQ1E2cxdtg73zHOObyucJZSIzSrEBABwsYSX5yX3rsB+9rEnjZ9qFHkyIZ
uZwOrdhSlR8f1AfPXQN8B5zgtRnNWOSbsQCZELbJ6pEmGpUWh3KiWCTZtih8g9WX+xQXIh8/il9KnyNxZF/rQ== lus
hg-tcxg@qq.com
root@ecs-s6-large-2-linux-20200108135338:~#
root@ecs-s6-large-2-linux-20200108135338:~#
root@ecs-s6-large-2-linux-20200108135338:~#
root@ecs-s6-large-2-linux-20200108135338:~#
```


当然你也可以手动复制本地~/.ssh/id_rsa.pub（注意是pub结尾的公钥）文件内容追加到服务器~/.ssh/authorized_keys的后面（从命名可以看出该文件可以存储多个ssh key）

注意：这里全程使用的是root用户，所以没有文件操作权限问题。如果你的文件夹创建用户不是远程登录用户，或许会存在同步文件失败的问题，此时需要远程服务器修改文件夹的读写权限（命令 chmod）。
创建了一个测试项目（点击本链接可以在gihub查看）试一下，打包、文件上传一句指令搞定啦：



访问一下，果然看到了我们熟悉的界面：



至此，常规情况下发布Vue项目就介绍完了，接下来介绍非域名根路径下发布以及history路由模式发布方法。

三、非域名根路径发布

有时候同一台服务器同一端口下可能会根据目录划分出多个不同的项目，比如我们希望项目部署到http://a.com/test下，这样访问http://a.com/test访问到的是项目的首页，而非test前缀的地址会访问到其它项目。此时需要修改nginx配置以及Vue打包配置。

1、 nginx配置

只需要添加一条location规则，分配访问路径和指定访问文件夹。我们可以把/test指向之前创建的/www文件夹，这里因为文件夹名称和访问路径不一致，需要用到alias这个配置：

```
location /test {
        alias /www;
}


location / {
        # First attempt to serve request as file, then
        # as directory, then fall back to displaying a 404.
        try_files $uri $uri/ =404;
}
```


如果文件夹名称与访问路径一致都为test，那这里可以用root来配置：

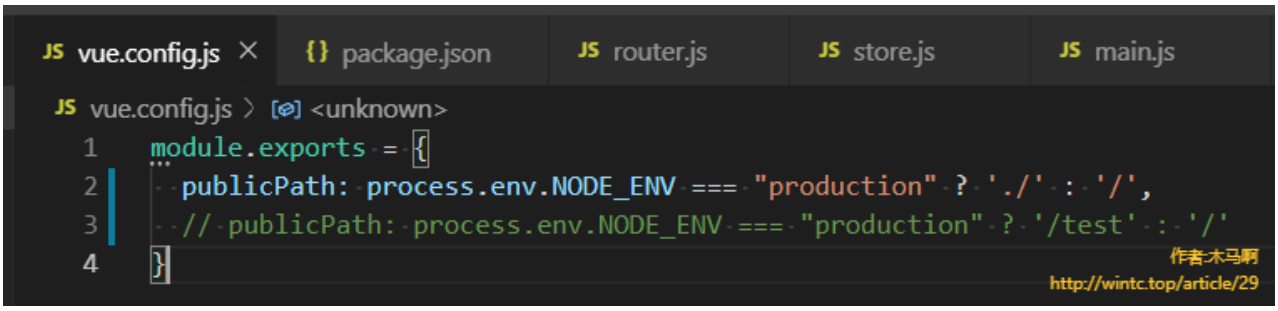
```
    location /test {
            root /;
    }


    location / {
            # First attempt to serve request as file, then
            # as directory, then fall back to displaying a 404.
            try_files $uri $uri/ =404;
    }
```

test文件夹的父级目录完整路径

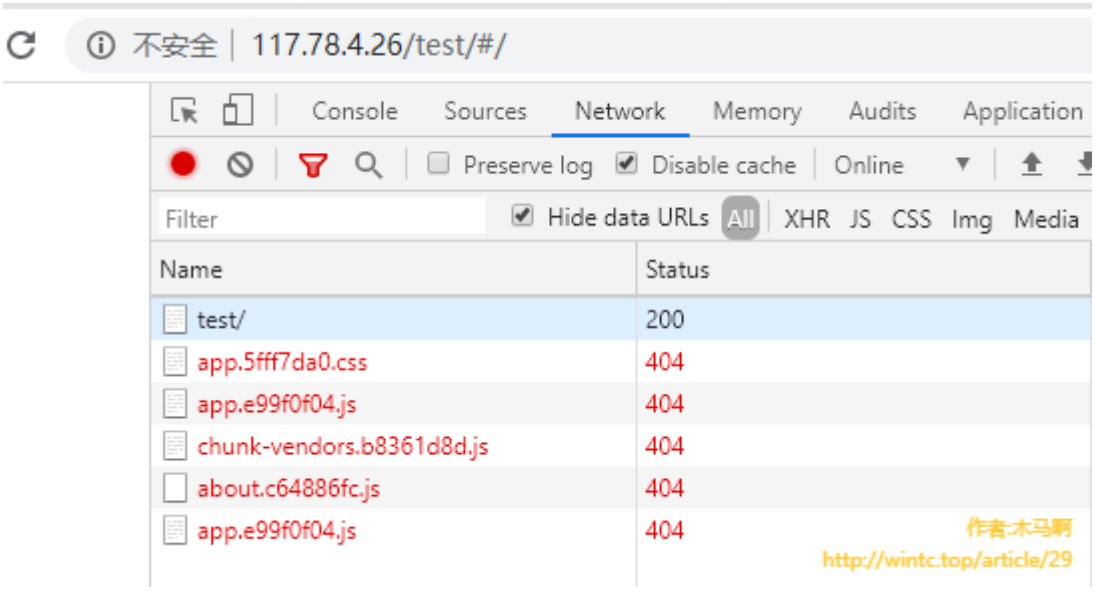
这里要将/test配置放到/之前，意味着在路由进入的时候，会优先匹配/test。如果根路径/下的项目有子路由/test，那http://xxxx/test只会访问到/www里的项目，而不会访问该子路由。

2、项目配置

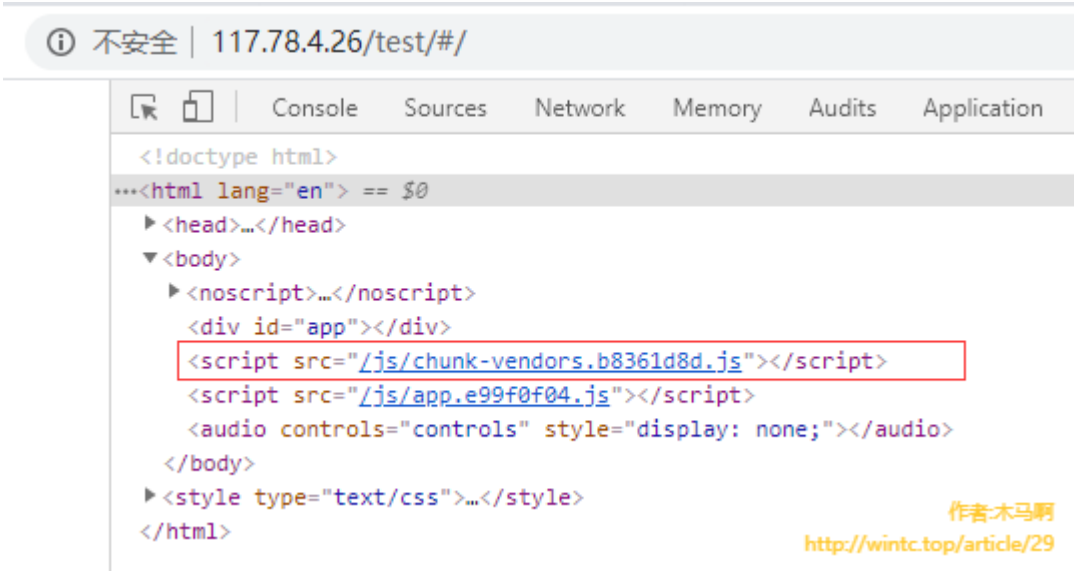
为了解决打包后资源路径不对的问题，需要在vue.config.js中配置publicPath，这里有两种配置方式，分别将publicPath配置为./和/test：



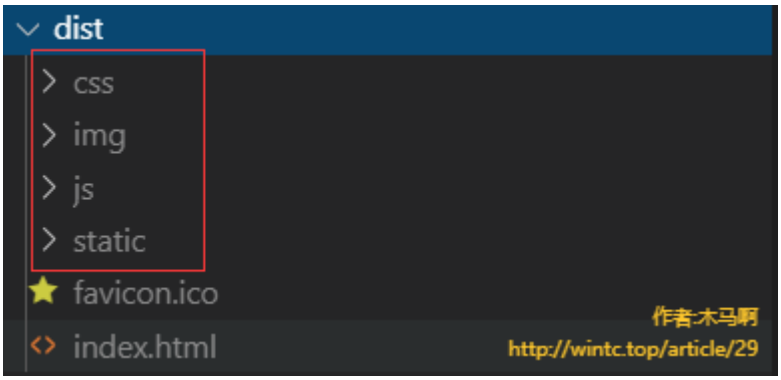
更新nginx配置，发布后即可正常访问啦。这里的两种配置方式是有区别的，接下来会看一下它们的区别。
如果不进行项目配置，直接发布访问会出现JS、CSS等资源找不到导致页面空白的问题：



该问题原因是资源引用路径不对，页面审查元素可以看到，页面引用的js都是从根路径下引用的：

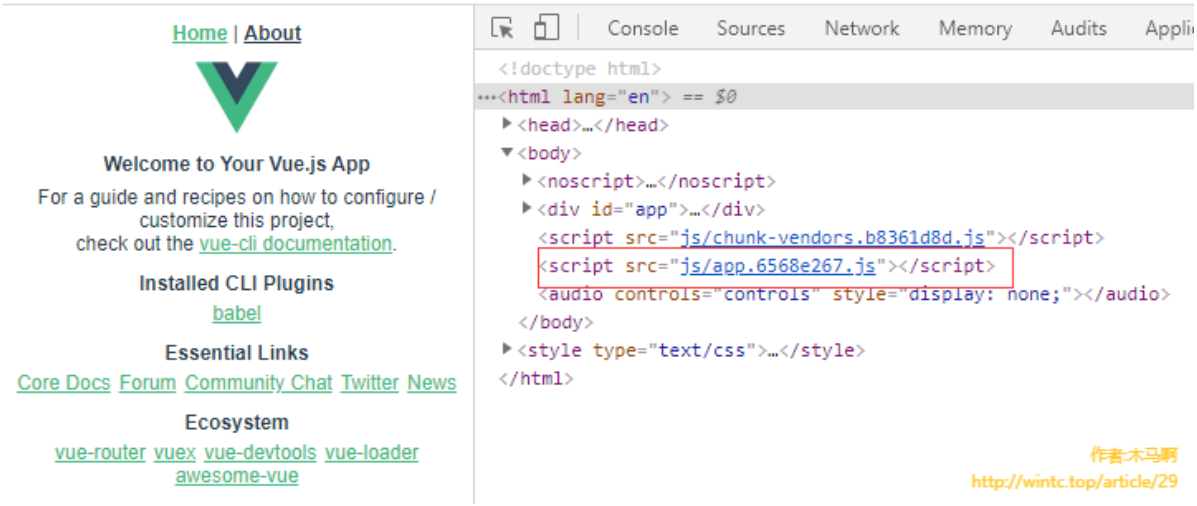


查看打包后的文件结构，可以看到js/css/img/static等资源文件是与index.html处于同级别的：

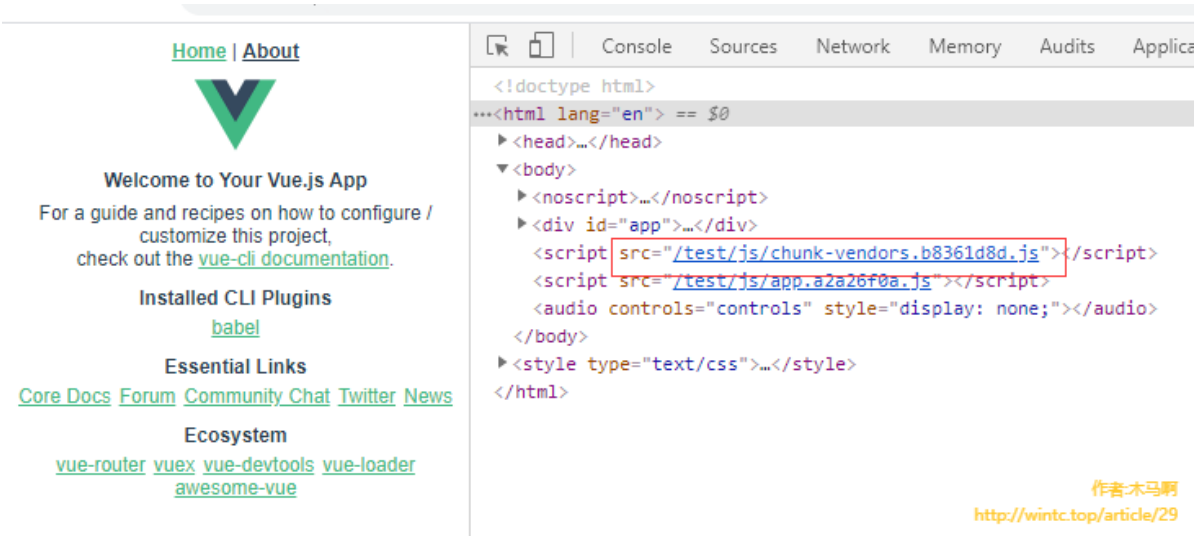


对于两种配置方式，看看都是怎么生效的：

1. publicPath配置为./， 打包后资源引用路径为相对路径：



2. publicPath配置为/test，打包后资源相对路径为从域名根目录开始的绝对路径：



两种配置都可以正确地找到JS、CSS等资源。不过还有个问题，那就是static中的静态资源依旧会找不到。

3、绝对路径引用的静态资源找不到的问题

因为在打包过程中，public下的静态资源都不会被webpack处理，我们需要通过绝对路径来引用它们。当项目部署到非域名根路径上时，这点非常头疼，你需要在每个引用的URL前

面加上process.env.BASE_URL（该值即对应上文配置的publicPath），以使得资源能被正常访问到。我们可以在main.js把这个变量值绑定到Vue.prototype，这样每个Vue组件都可以使用它：

```
Vue.prototype.$pb = process.env.BASE_URL
```

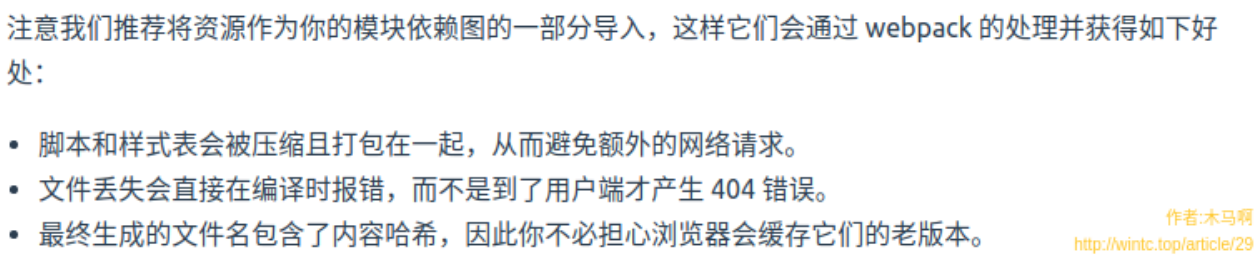
在模板中使用：

```
<img :src="`${$pb}static/logo.png`">
```

然而，更加头疼并且没有良好解决方案的问题是在组件style部分使用public文件夹下的静态资源：

- 如果需要使用图片等作为背景图片等，尽量使用内联方式使用吧，像在模板中使用一样。
- 如果需要引入样式文件，则在index.html中使用插值方式引入吧。

关于静态资源的问题，vue-cli的推荐是尽量将资源作为你的模块依赖图的一部分导入（即放到assets中，使用相对路径引用），避免该问题的同时也带来其它好处：



四、history模式部署

默认情况下，Vue项目使用的是hash路由模式，就是URL中会包含一个#号的这种形式。#号以及之后的内容是路由地址的hash部分。
正常情况下，当浏览器地址栏地址改变，浏览器会重新加载页面，而如果是hash部分修改的话，则不会，这就是前端路由的原理，允许根据不同的路由页面局部更新而不刷新整个页面。
H5新增了history的pushState接口，也允许前端操作改变路由地址但是不触发页面刷新，history模式即利用这一接口来实现。因此使用history模式可以去掉路由中的#号。

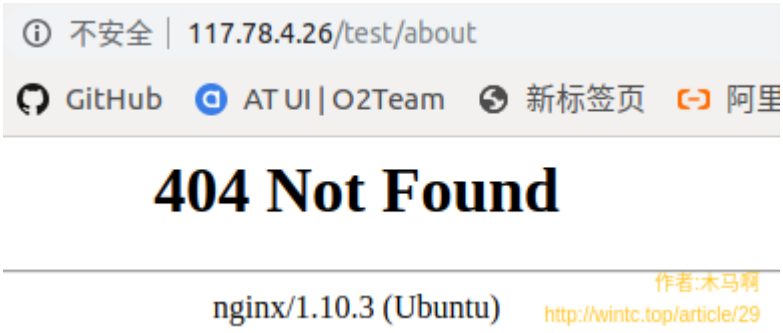
1、项目配置

在vue-router路由选项中配置mode选项和base选项，mode配置为'history'；如果部署到非域名根目录，还需要配置base选项为前文配置的publicPath值（注意：此情况下，publicPath必须使用绝对路径/test的配置形式，而不能用相对路径./）

```
export default new Router({
  mode: 'history',
  base: process.env.BASE_URL,

  routes: [ …
})
```


2、 nginx配置

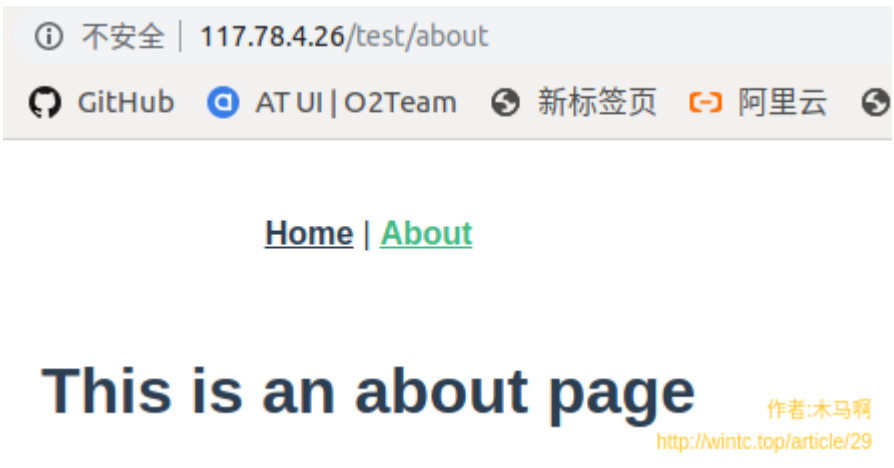
对于history模式，假设项目部署到域名下的/test目录，访问http://xxx/test/about的时候，服务器会去找/test指向的目录下的about子目录或文件，很显然因为是单页面应用，并不会存在a这个目录或者文件，就会导致404错误：



我们要配置nginx让这种情况下，服务器能够返回单页应用的index.html，然后剩下的路由解析的事情就交给前端来完成即可。

```
location /test {
        alias /www;
        try_files $uri $uri/ /test/index.html;
}
```


这句配置的意思就是，拿到一个地址，先根据地址尝试找对应文件，找不到再试探地址对应的文件夹，再找不到就返回/test/index.html。再次打开刚才的about地址，刷新页面也不会404啦：



3、history模式部署到非域名根路径下

非域名根目录下部署，首先肯定要配置publicPath。需要注意的点前面其实已经提过了，就是这种情况下不能使用相对路径./或者空串配置publicPath。
为什么呢？
原因是它会导致router-link等的表现错乱，使用测试项目分别使用两种配置打包发布，审查元素就能看出区别。在页面上有两个router-link，Home和About：

```
<div id="app">
  <div id="nav">
    <router-link to="/">Home</router-link> |
    <router-link to="/about">About</router-link>
  </div>
  <router-view/>
</div>
```


两种配置打包后的结果如下。

1. publicPath配置为./或者空串：

```
<div id="app">
  <div id="nav">
    <a href="/" class="router-link-active">Home</a>
    " |
      "
    <a href="/about" class>About</a> == $0
  </div>
  <!---->
</div>
```


2. publicPath配置为/test：

```
<div id="app">
  <div id="nav">
    <a href="/test/" class="router-link-exact-active router-link-active">Home</a>
    " |
      "
    <a href="/test/about" class>About</a> == $0
  </div>
  <div data-v-376fa61e class="home">…</div>
</div>
```


publicPath配置为相对路径的router-link打包后地址变成了相对根域名下地址，很明显是错误的，所以非域名根路径部署应该将publicPath配置为完整的前缀路径。

五、结语

关于Vue项目发布的相关问题就先总结这么多，几乎在每一步都踩过坑才有所体会，有问题欢迎各位同学一起探讨。

> 来源：https://wintc.top/article/29

- EOF -

推荐阅读 点击标题可跳转
1、整理的一些 Vue3 知识点

2、7 个使 vue 开发如虎添翼的 VS Code 扩展

3、活用 async/await ，让 Vue 变得更好用的装饰器！

觉得本文对你有帮助？请分享给更多人

推荐关注「前端大全」，提升前端技能

前端大全
点击获取精选前端开发资源。「前端大全」日常分享 Web 前端相关的技术文章、实用案例、工具资源、精选课程、热点资讯。

201篇原创内容

公众号

点赞和在看就是最大的支持❤️