

Node.js 服务器环境搭建

背景

开发部署一款微信推送小工具

流程

购买服务器

- [云服务器 ECS_云主机_弹性计算_BGP多线 - 阿里云](#)
- [云服务器_云主机_云计算服务器_弹性云服务器 - 腾讯云](#)

购买域名

- [域名注册_虚拟主机_云服务器_企业邮箱-万网-阿里云旗下品牌](#)
- [域名注册_域名购买、申请-腾讯云](#)

服务器环境搭建

1. 选择系统 CentOS 7.2 64位
2. 本地登录远程服务器

```
ssh root@118.24.157.214
```

3. 安装环境软件

```
# 1. 安装 git
yum install -y git

# 查看是否安装成功
git --version

# 2. 安装 docker
curl -fsSL get.docker.com -o get-docker.sh
sudo sh get-docker.sh --mirror Aliyun

# 变更 docker 国内源
sudo tee /etc/docker/daemon.json <<-'EOF'
{
    "registry-mirrors": ["https://r6878v77.mirror.aliyuncs.com"]
}
```

EOF

启动 docker

```
systemctl start docker
```

查看是否安装成功

```
docker --version
```

3. 安装 docker-compose

```
sudo curl -L https://get.daocloud.io/docker/compose/releases/download/1.27.3/  
docker-compose-`uname -s`-`uname -m` > /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

查看是否安装成功

```
docker-compose --version
```

4. 安装 nvm

```
curl -o- https://cdn.ic47.com/nvm_0.36.0.sh | bash
```

检查是否安装成功

```
source ~/.bashrc
```

```
nvm --version
```

5. 安装 node.js

```
nvm install 12.19.0
```

检查是否安装成功

```
node -v
```

```
npm -v
```

6. 变更全局 npm 国内源

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

```
npm config set ELECTRON_MIRROR https://npm.taobao.org/mirrors/electron/
```

```
npm i node-sass --sass_binary_site=https://npm.taobao.org/mirrors/node-sass/
```

验证是否成功

```
npm config get registry
```

参考文档

- git: [Git](#)
- docker: [Install Docker Engine on CentOS | Docker Documentation](#)
- docker-compose: [Overview of Docker Compose | Docker Documentation](#)
- nvm: [GitHub - nvm-sh/nvm: Node Version Manager - POSIX-compliant bash script to manage multiple active node.js versions](#)
- node.js: [Node.js](#)
- 阿里云镜像加速器: [Login](#)

4. 优化服务器登录

```
# 1. 本地生成 ssh key
ssh-keygen -t rsa -C "dingjb@ik47.com"

# 2. 保存至本地目录 ~/.ssh/id_rsa_cd

# 3. 登录远程服务器
ssh root@118.24.157.214

# 4. 修改 ssh 登录配置, 并保存
sudo vi /etc/ssh/sshd_config
# 打开以下两行注释
PubkeyAuthentication true
PermitRootLogin yes

# 5. 将本地的 id_rsa_cd.pub 内容复制到服务器的 .ssh/authorized_keys 文件中

# 6. 修改本地的 ~/.ssh/config 文件, 在末尾追加以下内容
Host.tencentcd
    HostName 118.24.157.214
    User root
    IdentityFile ~/.ssh/id_rsa_cd

# 7. 退出远程服务器
exit

# 8. 重新连接, 发现不用输入密码也可以直接登录了
# 9. 优化每次输入 ssh.tencentcd 太麻烦, 编辑 ~/.bashrc 创建别名
```

```
vi ~/.bashrc
# 末尾追加，以后就可以通过输入 cs2 来连接远程服务器
alias cs2="ssh.tencentcd"

# 在命令行中运行，是刚刚的修改配置生效
source ~/.bashrc

# 检查是否生效
cs2
```

参考文档

- ssh config: [使用 SSH config 文件](#)
- bash config: [CentOS中环境变量和配置文件 - Ryan.Miao - 博客园](#)

5. 同步文件到服务器

scp 同步

```
scp -r ./ ./.tencentcd:/root/data
```

rsync

```
rsync -av -e ssh --exclude={'node_modules', '.DS_Store'} ./ ./.tencentcd:/root/data
```

差异点：

rsync 可以排除指定目录

参考文档

- scp & rsync: [技术|如何在使用 scp 命令时递归地排除文件](#)

6. 服务端环境启动

```
# 进入 docker-compose.yml 所在的目录，启动服务
docker-compose up -d

# 检查是否启动成功
```

```
docker ps
```

访问 118.24.157.214, 发现展示出来 html 页面, 即为成功



7. 域名映射到服务器, 目的: 让用户访问域名的时候能指向对应的服务器

1. 进入 dnspod 或者 阿里云dns 设置页面
2. 添加一天A记录, 指向服务器IP,
3. 如图



如果我们访问 <http://demo.2k71.com>, 能看到和通过 ip 访问一样的页面。

8. 反向代理设置

目的: 用户访问域名后, nginx 会把请求转发到 node.js 启动的服务端口中, 因此我们要模拟反向代理

```
# 更改本地文件的 nginx 配置
添加我们的域名配置

server {
    listen 80;
    listen 443 ssl;
    ssl_certificate /etc/nginx/conf.d/demo.2k71.com.crt;
    ssl_certificate_key /etc/nginx/conf.d/demo.2k71.com.key;
    ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers
EECDH+CHACHA20:EECDH+AES128:RSA+AES128:EECDH+AES256:RSA+AES256:EECDH+3DES:RSA+3
DES:!MD5;
    ssl_prefer_server_ciphers on;
    ssl_session_timeout 10m;
    ssl_session_cache builtin:1000 shared:SSL:10m;
    ssl_buffer_size 1400;
    add_header Strict-Transport-Security max-age=15768000;
```

```
ssl_stapling on;
ssl_stapling_verify on;

server_name demo.2k71.com;
access_log off;
index index.html index.htm index.php;
location / {
    error_log /root/error.log;
    proxy_pass http://127.0.0.1:7001;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-Proto http;
    proxy_set_header X-NginX-Proxy true;

    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;
    proxy_redirect off;
}
}
```