微信小程序开发

配置小程序服务端

一个比较复杂的示例

```
takePhoto(e) {
  this.setData({ ocrtext : '正在识别···' });
  wx.createCameraContext().takePhoto({
    quality: 'high',
    success: (res) => {
      wx.uploadFile({
        url: 'https://awy.d5h5.com/wxocr',
        filePath: res.tempImagePath,
        name: 'image',
        success : (res) => {
          if (res.statusCode == 200) {
            let ocr = JSON.parse(res.data);
            let text = '';
            if (ocr.errcode == 0) {
              for (let i=0; i<ocr.items.length; i++) {
                text += ocr.items[i].text + '<br>';
              }
            } else {
              text = ocr.errmsg;
            }
            this.setData({ ocrtext: text });
          }
        }
      });
    }
  });
}
```

- 左侧这段代码在拍照时触发。
- 用户通过小程序拍照后，会通过wx.uploadFile接口上传文件到开发者后台API。
- 在开发者后台API会处理上传的图片并调用小程序服务端API进行OCR处理，识别图片文字并返回结果。
- 小程序端接收到结果后提取其中的text字段显示处理结果。

小程序后端技术清单

- 对于以上示例，现在来梳理需要用到的技术：
  - HTTP服务基础（已解决）
  - 云服务器基本操作（部署服务，系统更新，软件安装等）（已解决）
  - 二级域名和DNS解析
  - 反向代理（如果需要同一台设备运行多个服务）
  - HTTP客户端请求
  - Web框架基础（可以不使用，需要自己实现上传处理过程）

域名和端口

- 默认HTTP运行在80端口，HTTPS运行在443端口。

- 运行HTTP和HTTPS服务，使用其他端口也是可以的。对于访问服务器来说，可以直接使用IP地址:端口号的形式访问。

域名和端口

- 问题在于，小程序和公众号在对接开发者服务器时，限制必须要使用域名，并且不能使用端口号。

- 这意味着，一台服务器要运行多个服务，但是对外要仅仅提供80/443的端口访问，就需要在服务端做一些处理。解决方案就是上文中提到的‘反向代理’。

HTTP协议和反向代理

- 要说明反向代理是什么，先要说明HTTP协议的请求头信息。
- HTTP是基于TCP协议的，所以无论云服务器上运行多少个服务，请求一定会连接到同一个IP地址。
- 但是，在HTTP这一层面，可以通过请求数据做判断处理。

HTTP协议和反向代理

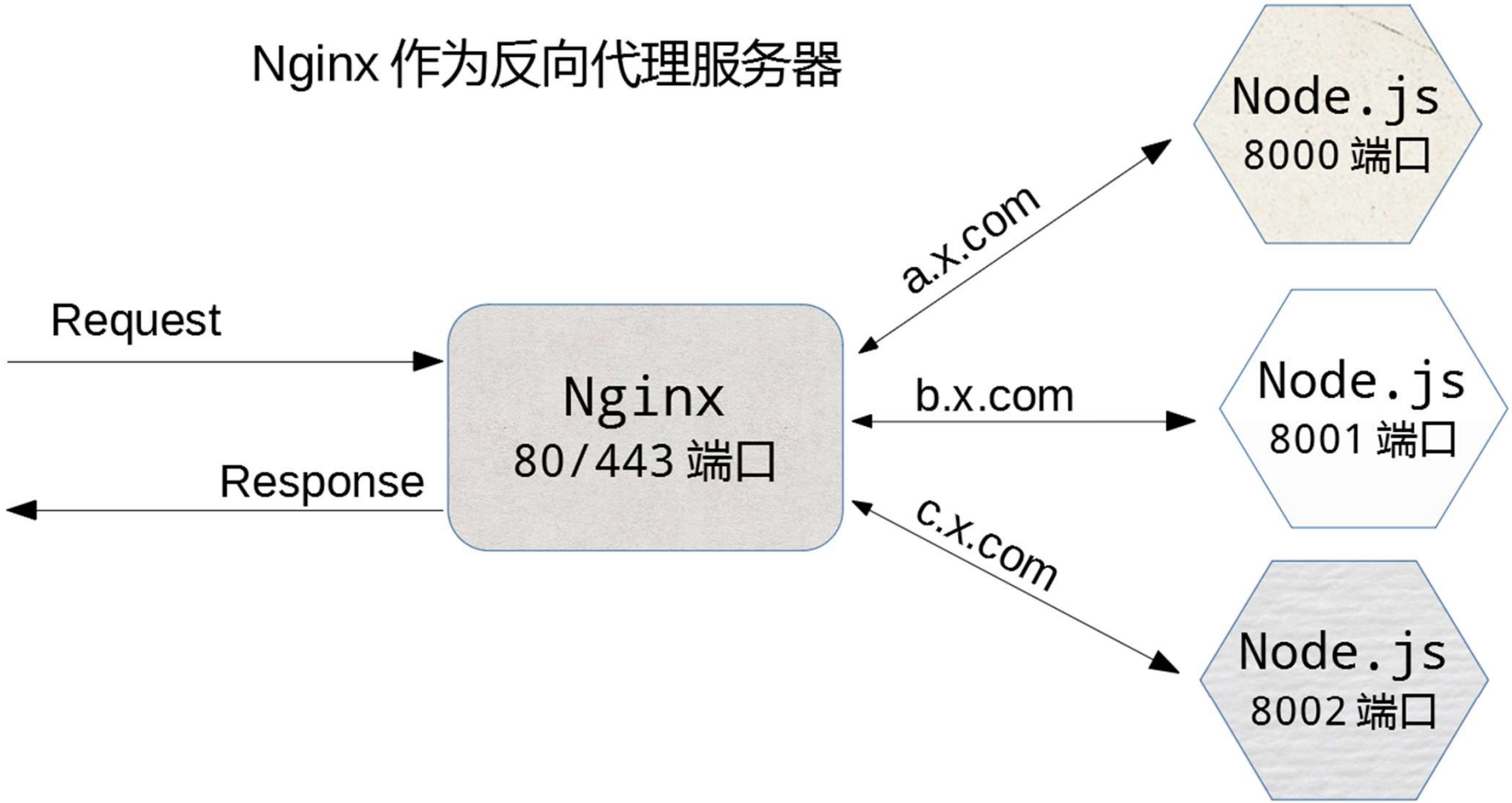
- 在浏览器中通过调试控制台可以看到请求头的具体数据。
- 其中有一项是Host。比如www.a.com和www.b.com都解析到同一个IP地址。

- 但是通过不同的域名访问，其Host值不同。所以，HTTP服务程序可以通过判断Host值的不同把请求转发给不同的服务。

HTTP协议和反向代理

- 如果在云服务器上运行3个服务，分别监听8000，8001，8002端口。而监听80端口的服务则通过Host值把请求分别转发到对应的3个服务上，这种模式称为反向代理。

- 要运行反向代理服务，要求HTTP服务程序既要作为服务端，也要作为客户端。同时，要有多个域名解析到服务器IP地址。


Nginx 作为反向代理服务器
Request
Response
Nginx
80/443 端口
a.x.com
b.x.com
c.x.com
Node.js
8000 端口
Node.js
8001 端口
Node.js
8002 端口

操作流程

- 明确了以上过程，接下来就是逐步完成相关操作:
  - 分配二级域名
  - 安装Nginx作为反向代理服务
  - 配置Nginx代理不同的服务
  - 运行Node服务并测试

二级域名和DNS解析

- 在多人共用一台服务器的情况下，已注册域名并备案后，每个人都可以有一个自己的域名，在注册域名之下，称为二级域名。
- 云服务器管理员可以在登录管理控制台，在域名解析页面给每个人分配二级域名并解析到服务器公网IP。

- 这样一来，大家共用一台服务器，但是每个人拥有不同的域名。这个关键的一步使得后续的操作成为可能。

Nginx简介

- Nginx是一个高性能的Web服务程序。除了HTTP、HTTPS、HTTP/2、FastCGI协议之外，还支持邮件常用协议等。

- Nginx源代码用C编写，在Linux上，Nginx使用了系统底层的epoll异步IO接口处理请求，所以在处理IO密集型任务时，性能极高。

安装Nginx

- 在Debian/Ubuntu上，安装Nginx：

```
sudo apt install nginx
```

- 在CentOS上安装Nginx：

```
sudo yum install nginx
```

- 如果你是`root`用户则不需要使用`sudo`，只需要具备`root`权限的管理员安装`nginx`并配置即可。

快速了解Nginx配置文件

- 通过命令安装的nginx，默认其配置文件在/etc/nginx目录中。

- 因发行版的不同，具体的配置文件可能在/etc/nginx/nginx.conf，也可能在/etc/nginx/conf/nginx.conf。
- nginx.conf是nginx启动时读取的配置文件。

快速了解Nginx配置文件

- 在nginx.conf中，配置是分块的，其中有http {…}配置块。配置的是对HTTP服务起作用的项。
- 反向代理的配置也在这里面，在http配置块中，每个服务都是一个server配置块。
- 默认Nginx已经有一个server块，并且会有相关注释给出示例说明。

添加反向代理server配置块

- 在http配置块中，添加以下server块就添加了到8000端口的反向代理。

```
server {
    listen  80; #监听80端口
    server_name  www.a.com; #要代理的域名
    location {
        #代理的本地地址
        proxy_pass         http://localhost:8000;
        #设置请求头数据
        proxy_set_header  Host       $host;
        proxy_set_header  X-Real-IP  $remote_addr;
    }
}
```

添加多个块

- 给每个用户添加一个server配置块。配置不同的域名和代理端口。
- 注意，listen 是统一端口，对于HTTP来说是80，对于HTTPS来说是443。
- 而proxy_pass 配置的本地地址端口则不同。

重启Nginx服务

- 通过`systemctl`重启服务:

```
sudo systemctl restart nginx
```

- 通过启动脚本重启服务:

```
sudo /etc/init.d/nginx restart
```

运行Node.js测试服务

- 比如，2个用户代理端口分别为8000和8001。Nginx已经配置好域名分别为x.a.com和y.a.com。

- 则A用户运行服务监听8000端口，B用户运行服务监听8001端口。
- 通过浏览器访问http://x.a.com则可以访问到A的服务。
- 通过浏览器访问http://y.a.com则可以访问到B的服务。