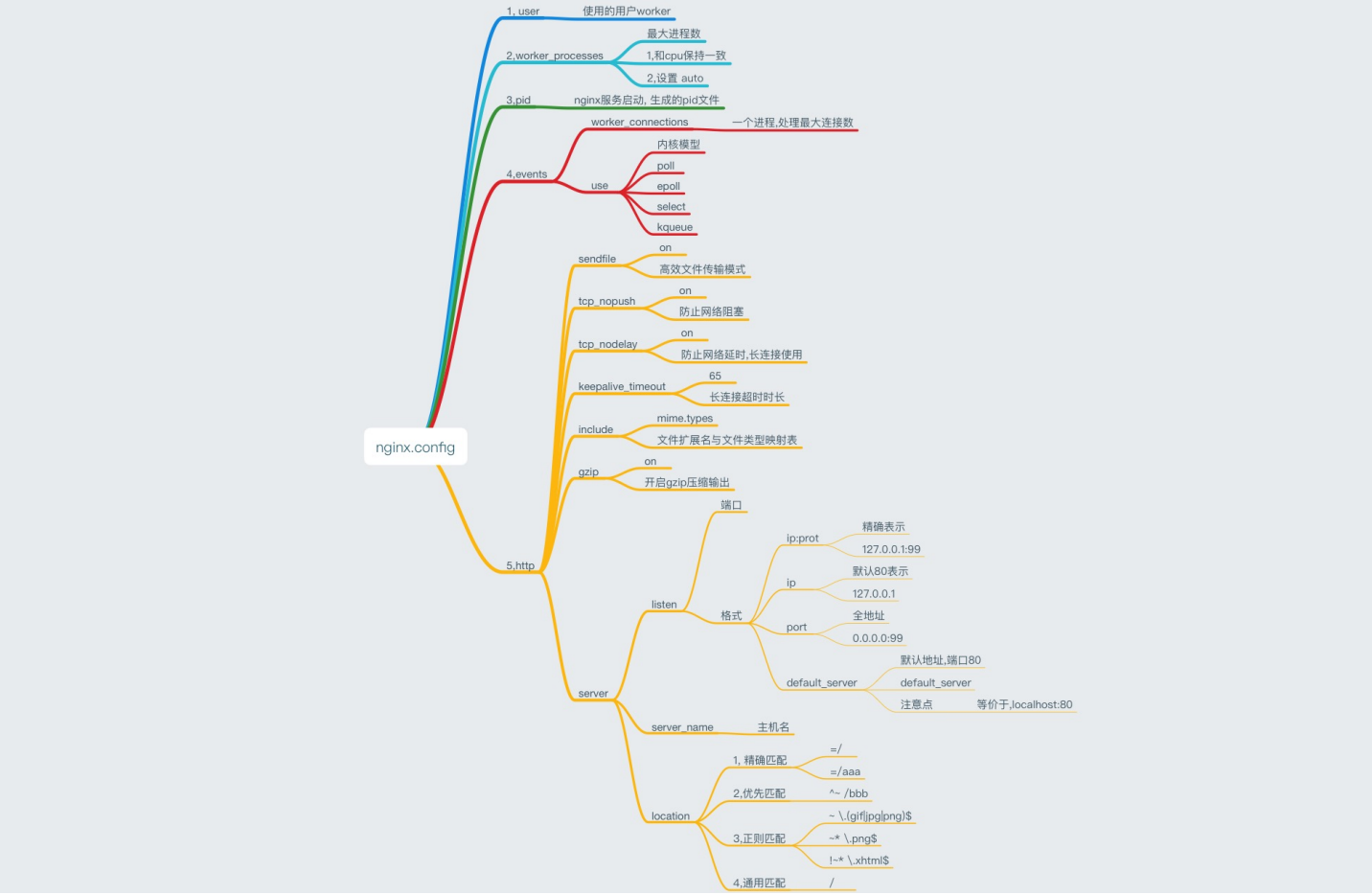
参考文档--------文档 --nginx



1, 部署基础知识

- 目的: 知道开发中项目的生命周期
- 传统项目: 调研 -> 设计 -> 开发 -> 测试 -> 运营
- 新型项目: 计划->设计->验证->开发->发布->修改->... 小步快跑

2,Nginx-介绍,安装,启动

- 目的: 可以按照文档安装Nginx, 并启动
- 特点:
  - 1, 支持高并发
  - 2, 内存消耗少
  - 3, 配置简单,稳定
  - 4, 模块程度高
  - 5, 免费开源, 支持多系统
- 安装:
  - 1, 安装

```
apt-get install -y build-essential libssl-dev libtool libpcre3 libpcre3-dev make openssl zlib1g-dev
```

```
apt-get install nginx -y
```

  - 2, 检查

```
sudo netstat -tnulp | grep nginx
```

- 启动,停止:

```
启动: systemctl start nginx
```

```
停止: systemctl stop nginx
```

3,Nginx-目录配置

- 目的: 知道Nginx中的常见的配置目录
  - 日志目录: /var/log/nginx/
  - 执行文件: /usr/sbin/nginx
  - 启动文件: /etc/init
  - web静态文件目录: /var/www/html/
  - 配置文件: /etc/nginx

4,Nginx.conf-全局配置

- 目的: 知道Nginx中的常见的全局配置段
- 配置:
  - user: 工作用户
  - worker_processes: 最大的工作进程, 一般和CPU的核数一样
  - pid: 进程文件
  - events: 事件处理模块,
    - worker_connections: 最大连接数
    - use: 使用的内核模型
  - http: 负责处理网络请求的

5,nginx-http配置段

- 目的: 知道http中常见的配置段
  - 常见的配置:
  - access_log: 日志访问目录
  - error_log: 错误日志记录
  - include: 包含的其他的配置文件
  - server: 服务
    - listen: 监听的端口,ip
    - server_name: 配置访问域名
    - location: 资源地址
  - upstream: 配置负载均衡

6,server-listen配置

- 目的: 能够使用listen设置常见的监听方式

```
server {
        #1,监听端口,默认ip 0.0.0.0
        #listen 6001;
        #location / {
        #   return 400;
        #}

        #2,监听ip,默认端口80
        #listen 172.16.12.134;
        #location / {
        #   return 401;
        #}

        #3,监听ip+端口
        listen 172.16.12.134:6002;
        location / {
           return 402;
       }
}
```

7,vscode-sftp配置(参考文档)

- 目的: 可以参考文档配置vscode链接ubuntu下面的/etc/nginx
- 文档

8,root根路由,alias子路由,index索引访问静态文件

- 目的: 能够在location中配置root alias, index访问静态文件

- 配置:

```
server {
   #1,root ,index配合使用
   listen 6001;
   location / {
       root /etc/nginx/my_html;
       index index1.html1 index2.html;
  }

   #2,alias ,index配合使用
   # location /aaa {
   #     alias /etc/nginx/my_html;
   #     index index2.html;
   # }

}
```

- 注意点:
  - 1, 如果匹配的路径是/, 需要使用root关联静态文件
  - 2, 如果匹配的是/other, 需要使用alias
  - 3, index 后面可以跟多个静态文件, 如果前面加载不到就可以加载后面的

9,server_name配置--配置域名访问

- 目的: 可以通过配置域名的形式访问，本地访问的话需要配置hosts
- 配置域名访问:

```
server {
   listen 6001;
   server_name www.aaa.com;
   location / {
       return 400;
  }
}
```

10,location配置常见动作

- 目的: 能够知道location中路径配置的规则已经优先级问题

1, 精确匹配

```
server {
   #1,精确匹配, =表示精确匹配
   listen 6001;
   location = / {
       return 400;
  }
}
```

2, 正则匹配

```
server {
   #1,精确匹配, =表示精确匹配
   listen 6001;
  ...

   #2,正则匹配, ~ 正则区分大小写匹配
   location ~ / {
       return 401;
  }

}
```

3, 通用匹配

```
server {
   #1,精确匹配, =表示精确匹配
   listen 6001;
  ..

   #2,正则匹配, ~ 正则不忽略大小写匹配
  ...

   #3,通用规则
   location / {
       return 402;
  }

}
```

- 匹配优先级: 精确 > 正则 > 通用

11, 正则匹配

- 目的: 知道nginx中,常见的正则匹配规则
- 常见的匹配规则:

```
server {
   #1, ~ , 区分大小写
   listen 6001;
   location ~ \.png$ {
       return 400;
  }
   #2,~* 不区分大小写
   location ~* \.JPG$ {
       return 401;
  }
   #3, 不区分大小写,同时匹配多个后缀
   location ~* \.(png|jpg|jpeg|mp4|avi)$ {
       return 402;
  }
   #4,^~ 匹配开头的
   location ^~ /abc {
       return 403;
  }
}
```

12,location临时跳转

- 目的: 可以在nginx中配置页面访问的重定向
- 常见配置:

```
server {
   #1,重定向 302
   listen 6001;
   location / {
       # return 302 http://www.baidu.com; # 外部重定向
       return 302 /login; #内部重定向
  }
   location /login {
       return 400;
  }
}
```

13,localtion访问控制

- 目的: 可以通过配置,允许指定的主机地址访问
- 常见配置:

```
server {
   #1,访问控制
   listen 6001;
   location / {
       stub_status on; #展示连接的页面
       allow 172.16.12.134/16; #允许通过哪个主机地址访问
       deny all; #禁止所有的地址访问
  }
}
```

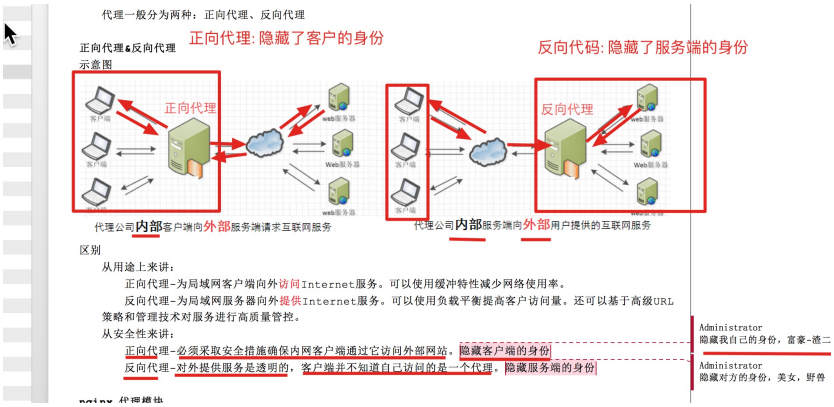
14,location目录列表

- 目的: 能够通过文档配置, 索引的格式
- 配置:

```
server {
   #1,显示文件目录
   listen 6001;
   location /aaa {
       alias /etc/nginx;
       autoindex on; # 激活/关闭自动索引
       autoindex_localtime on; #定索引时文件大小
       autoindex_exact_size off; #开启以本地时间来显示文件时间的功能
  }
}
```

15,反向代理

- 目的: 可以理解正向代理, 反向代理的概念;
  - 正向, 反向的图解:



14,负载均衡, 常见调度算法, 配置反向代理

- 目的: 可以配置负载均衡, 已经知道常见的调度算法
- 配置:

```
#0,负载均衡配置
upstream meiduo {
   #1, 默认,轮询
   # server 172.16.12.134:6002;
   # server 172.16.12.134:6003;

   #2, 加权轮询
   # server 172.16.12.134:6002 weight=1;
   # server 172.16.12.134:6003 weight=3;

   #3, ip_hash, 第一次访问的哪个服务器,以后就走哪个服务器
   ip_hash;
   server 172.16.12.134:6002;
```

```
    server 172.16.12.134:6003;
}


#1, 代理服务器
server {
    listen 6001;
    location / {
        # proxy_pass http://172.16.12.134:6002;
        proxy_pass http://meiduo;
   }
}

#2, 真实业务服务器
server {
    listen 6002;
    location / {
        root /etc/nginx/my_html;
        index index1.html index2.html;
   }
}

#3, 真实业务服务器
server {
    listen 6003;
    location / {
        root /etc/nginx/my_html;
        index index2.html index1.html ;
   }
}
```

15,日志解析

- 目的: 知道日志配置的格式
- 操作流程:
  - 1, nginx.conf配置

```
log_format proxy_format '$remote_addr - $remote_user [$time_local] '
'"$request" $status $body_bytes_sent '
'"$http_referer" "$http_user_agent"';

```

  - 2, 代理服务器

```
server {
    listen 172.16.12.134:6001;
    location / {
        proxy_pass http://172.16.12.134:6002;

        #1,可以转发请求头中的信息
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

   }
}
```

  - 3, 真实服务器

```
server {
    listen 172.16.12.134:6002;
    #1,记录日志信息
```

```
    access_log /var/log/nginx/meiduo/access.log proxy_format;
    real_ip_header X-Forwarded-For;
    set_real_ip_from 172.16.12.134/16;
    real_ip_recursive on;
    location / {
        return 400;
    }
}
```

- 4,创建日志文件(var/log/nginx/meiduo)