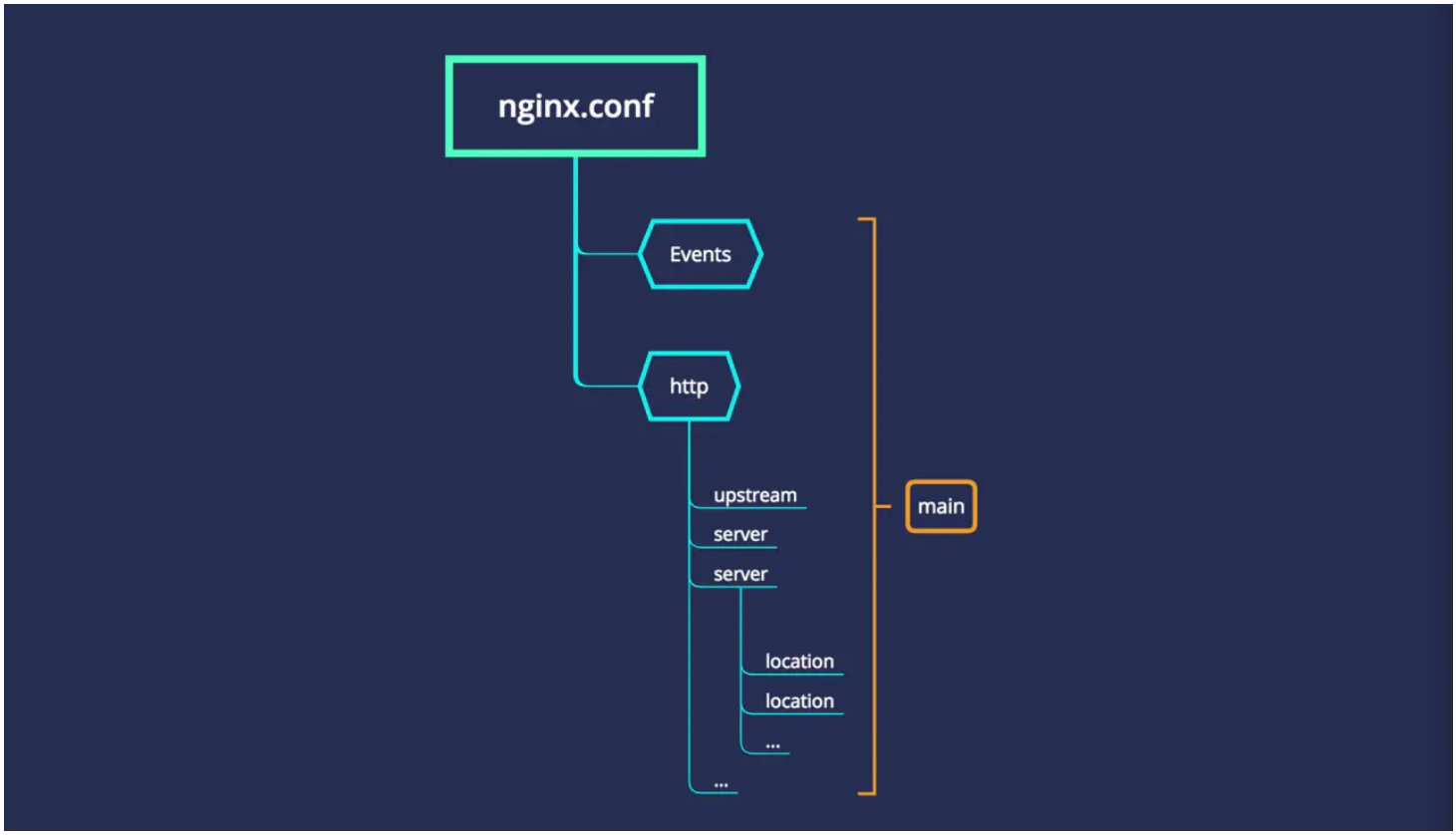
# Nginx命令

- `start nginx`：开机
- `nginx -c [配置文件路径]`：为nginx指定一个配置文件
- `nginx -t`：不运行，只检查配置文件语法的正确性
- `nginx -v`：显示nginx版本号
- `nginx -s stop`：快速关闭(不保存任何信息，直接关闭)
- `nginx -s quit`：正常关闭(保存相关信息，关闭web服务后再关闭nginx)
- `nginx -s reload`：重新加载配置文件
- `nginx -s reopen`：重新打开日志文件

# Nginx配置文件结构




- `main`： Nginx全局配置
  - `events`：配置影响nginx服务器或与用户的网络连接，nginx服务器事件模型相关配置
  - `http`：httpGlobal配置，可以嵌套多个server来配置代理，缓存，日志等
    - `upstream`：配置后端服务器的具体地址，是负载均衡和代理转发的关键配置
    - `server`： 配置虚拟主机相关参数，一个http中可以有多个server，一个server中可以包含多个location
      - `location`：具体路由的配置

样例配置：

```
# 每个配置项都要以; 结尾

## Global配置
#定义Nginx运行的用户和用户组
user www www;

#nginx进程数，通常设置成和cpu的数量相等
worker_processes 4;

#全局错误日志
error_log  logs/error.log;

#进程pid文件
pid logs/nginx.pid;

## events配置
events {
    # 选择事件模型，一般Linux系统就用epoll模型
    use epoll;

    #单个进程最大连接数（ 总最大连接数 = worker_connections * worker_processes ）
    worker_connections 1024;

    # 设置keepalive（http1.1默认开启长连接机制）
    connection: keepalive;
    #keepalive 超时时间，单位是秒
    keepalive_timeout 60;
}

http
{
    #文件扩展名与文件类型映射表
    include /etc/nginx/conf/mime.types;
    #默认文件类型
    default_type  application/octet-stream;

    # 负载均衡配置负载服务器
    upstream backserver {
        server 192.168.0.1;
        server 192.168.0.2;
    }

    server
    {
        location ^~ /rest/someService/data
        {
            # 代理转发
            proxy_pass        backserver;
            proxy_set_header  X-Real-IP  $remote_addr;
        }
```

```
        location ^~ /assets
        {
            # 静态伺服
            alias /var/share/;
            # 也可以用root指令
        }
     }

    # 导入其他配置
    include /d.conf/*
}
```

# Nginx内置变量

| 变量名 | 功能 |
| --- | --- |
| $host | 请求信息中的Host |
| $request_method | 客户端请求类型，如GET、POST等 |
| $remote_addr | 客户端的IP |
| $args | 请求中的query参数 |
| $content_length | 请求头的Content_length字段 |
| $http_user_agent | 客户端agent信息 |
| $http_cookie | 客户端cookie信息 |
| $remote_port | 客户端的端口 |
| $server_protocol | 请求使用的协议, 如HTTP/1.0, HTTP/1.1等 |
| $server_name | 服务器名 |
| $server_port | 服务器端口号 |

# 常见配置场景

## 定制请求头

语法： add_header < name > < value >
默认值： none
作用域： http, server, location
当HTTP应答状态码为 200、204、301、302 或 304 的时候，增加指定的HTTP头标。

## 设置缓存

语法：expires [time|epoch|max|off]
默认值： expires off
作用域： http, server, location
使用本指令可以控制HTTP应答中的“Expires”和“Cache-Control”的头标，（起到控制页面缓存的作用）。
取值：

- epoch：指定Expires请求头为1 January, 1970, 00:00:01 GMT，即不缓存
- max：指定Expires请求头为31 December 2037 23:59:59 GMT，即Cache-Control的值为10年。
- < time >：指定的一个过期时间，如果是负数，则表示不缓存，Cache-Control的值为no-cache，如果为正数，例如 expires 60，表示Cache-Control的值为max-age=60
- off: 不设置缓存相关请求头

## 静态伺服

```
location /assets {
    # 请求/assets/*，会去/etc/nginx/assets/*下找资源
    alias /etc/nginx/assets/;
}

location /assets {
    # 请求/assets/*， 会去/etc/nginx/assets/*下找资源
    root /etc/nginx;
}
```

## 负载均衡

```
http {
    upstream backserver {
        server localhost:3003;
        server localhost:3004;
        server localhost:3005;
    }

    location ^~ /rest/serviceA {
        # 给代理的请求添加额外请求头
        proxy_set_header X-Real-IP  $remote_addr;
        # 反向代理到后端服务器
        proxy_pass http://backserver;
    }
}
```

负载均衡策略：

- 轮询（默认）：

```
upstream backserver {
    server localhost:3003;
    server localhost:3004;
    server localhost:3005;
}
```

- 按权重分配负载：

```
upstream backserver {
    server localhost:3003 weight=1;
    server localhost:3004 weight=2;
}
```

- 按健康状况分配负载：

```
upstream backserver{
    server 192.168.0.1  max_fails=1 fail_timeout=40s;
    server 192.168.0.2  max_fails=1 fail_timeout=40s;
}
```

涉及两个配置：

- fail_timeout : 设定服务器被认为不可用的时间段以及统计失败尝试次数的时间段，默认为10s
- max_fails : 设定Nginx与服务器通信的尝试失败的次数，默认为：1次

## Gzip

```
server{
    gzip on; # 启动
    gzip_comp_level 1; # 压缩级别，1-10，数字越大压缩的越好，但是越消耗CPU资源，一般1就够用了
    gzip_min_length 100; # 不压缩临界值，大于100字节的才压缩，一般不用改
    gzip_types application/javascript text/css text/html; # 要压缩的文件类型
    gzip_disable "MSIE [1-6]\."; # IE6对Gzip不友好，对IE6禁用Gzip
    gzip_vary on; # 增加响应头Vary: Accept-Encoding
}
```