Nginx 从入门到企业实战（一）

Nginx 基础安装配置

1. 关于 Nginx 介绍

`Nginx` 是异步框架的 `web` 服务器，也可以用作反向代理、负载均衡以及作为缓存服务器。`Nginx` 是目前互联网公司 `web` 服务器的主流技术，用于处理高并发甚至海量并发的网站数据。`Nginx` 具有社区版和商业版，社区版是完全开源的，`Tengine` 就是淘宝在 `Nginx` 基础上进行二次开发，以获取更高的稳定性和并发能力，已经在淘宝、天猫等海量数据的电商网站上经过了“双十一”的技术洗礼，足以证明其稳定性和高性能。

1.1 Nginx 主要特性

- **高并发、高性能** 一台普通的服务器可以轻松处理上万并发连接，一般单台服务器最多建议处理三万左右的并发；
- **模块化设计** 基于模块化设计，具有非常好的扩展性，可以通过加载、卸载某个模块（注意：模块动态加载在 `Nginx V1.9.11` 版本之后才支持）以实现相应的功能；
- **热部署、热更新** `Nginx` 支持配置文件的热更新，版本热升级、动态加载模块、日志热更换；
- **内存低消耗** 据统计在 `10000` 个 `keep-alive` 连接模式下的非活动连接，仅消耗内存 `2.5M`；
- **配置、维护简单** `Nginx` 的配置非常简单，对于运维同学非常友好。

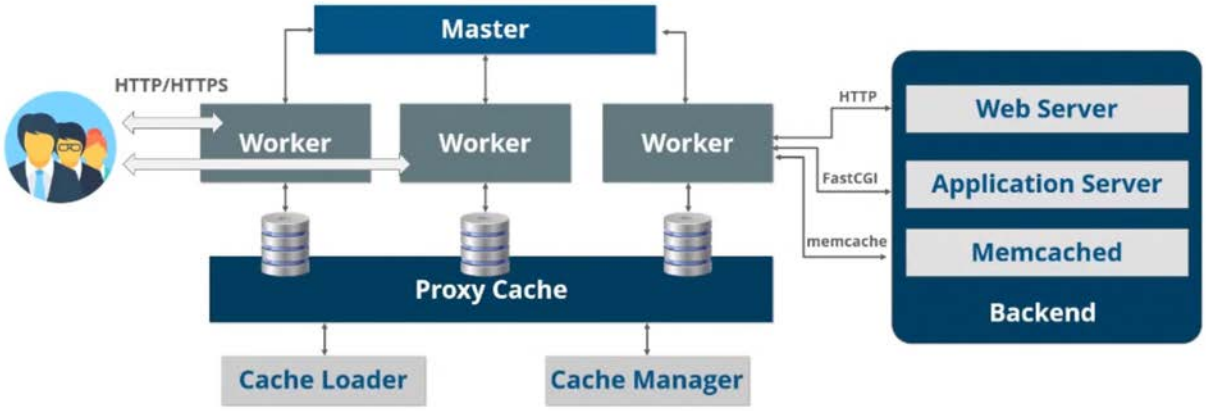
1.2 Nginx 基本功能

- **`Web` 服务器** 这是 `Nginx` 最基本的功能，也是非常重要的功能之一；
- **反向代理服务器** 同样，`Nginx` 可以作为 `http` 协议的反向代理服务器，也是生产环境中最常用的功能之一；
- **`FastCGI(php)`、`uWSGI(python)` 代理服务器** 生产环境上经常使用 `Nginx` 作为客户端请求后端应用服务（注意：此时 `Nginx` 在用户看来依然是反向代理服务器，只不过这里代理的请求不再是 `http` 协议，而是跟后端服务相关的协议，比如后端如果是 `php` 语言开发的，则是 `FastCGI` 协议代理）；
- **`TCP/UDP` 代理服务器**，也即 “**调度器**” 生产环境中，在一些并发量不大的情况下，有时候也会使用 `Nginx` 作为四层调度器；
- **`Mail` 邮件代理服务器** 可以作为邮件代理服务器，几乎不使用；

1.3 Nginx 基础架构

这里我们先不对 `Nginx` 架构做太多介绍，后面有专门的课程进行深入分析，本节我们仅做简单了解。

如下图，`Nginx` 为 `master/workers` 结构，一个 `master` 主进程，负责管理和维护多个 `worker` 进程，真正接收并处理用户请求的其实是 `worker` 进程，`master` 不对用户请求进行处理。即 `master` 主进程负责分析并加载配置文件，管理 `worker` 进程，接收用户信号传递以及平滑升级等功能。另外，`Nginx` 具有强大的缓存功能，其中 `Cache Loader` 负责载入缓存对象，`Cache Manager` 负责管理缓存对象。



2. Nginx 安装与访问测试

这里我们先介绍 `yum` 安装方式，稍后我们再介绍编译安装。需要注意的是 `CentOS` 基础 `yum` 源中是没有 `Nginx` 包的，我们需要添加好 `EPEL` 源，具体添加方式这里不再赘述，目前 `EPEL` 源中最新版本为 `Nginx V1.16.1`，这也是官方最新稳定版。

【安装】

```
[root@web-nginx ~]# yum install nginx -y
```

> **这里安装好的版本为 `nginx.x86_64 1:1.16.1-1.el7`**

我们可以看下 `Nginx` 生成了很多文件，其中 `/usr/sbin/nginx` 为主程序文件，`/etc/nginx/` 为 `nginx` 的配置目录，还有一些是帮助文件、默认网页文件、日志文件等，我们用到的时候再做介绍。我们先来看看 `/usr/sbin/nginx` 文件（这是一个可执行二进制文件），如下：

```
[root@web-nginx ~]# nginx -h
nginx version: nginx/1.16.1
Usage: nginx [-?hvVtTq] [-s signal] [-c filename] [-p prefix] [-g directives]

Options:
  -?,-h         : this help
  -v            : show version and exit
  -V            : show version and configure options then exit
  -t            : test configuration and exit
  -T            : test configuration, dump it and exit
  -q            : suppress non-error messages during configuration testing
  -s signal     : send signal to a master process: stop, quit, reopen, reload
  -p prefix     : set prefix path (default: /usr/share/nginx/)
  -c filename   : set configuration file (default: /etc/nginx/nginx.conf)
  -g directives : set global directives out of configuration file
```

> **`-V`： 该参数用于显示 `Nginx` 版本信息以及编译加载了哪些模块；**

`-t`： 用于 `nginx` 配置文件语法检测，我们在生产上修改好配置文件后，一定要先使用该参数进行语法检测；

`-s`： 给 `nginx` 传递信号，其中 `stop` 强制停止、`quit` 优雅退出；即等待连接关闭之后再退出； `reopen` 用于重新打开日志文件，一般用于日志文件的切割； `reload` 重载 `nginx` 配置文件；

`-g`：用于指定指令，该指令高于配置文件中的设置。如 `nginx -g 'daemon off;'` ，将 `nginx` 设置为前台运行；

【启动】

`Nginx` 启动非常简单，我们可以通过 `systemctl start nginx` 启动，也可以直接使用命令 `nginx` 进行启动，如下：

```
[root@web-nginx ~]# nginx
[root@web-nginx ~]# netstat -nltp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
  PID/Program name
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
 40542/nginx: master
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN
 1206/sshd
tcp6       0      0 :::80                   :::*                    LISTEN
 40542/nginx: master
tcp6       0      0 :::22                   :::*                    LISTEN
 1206/sshd
```

注意：为了避免防火墙以及 'selinux' 对我们的实验产生影响，先关闭掉防火墙 'firwalld/iptables' 以及 'selinux'。

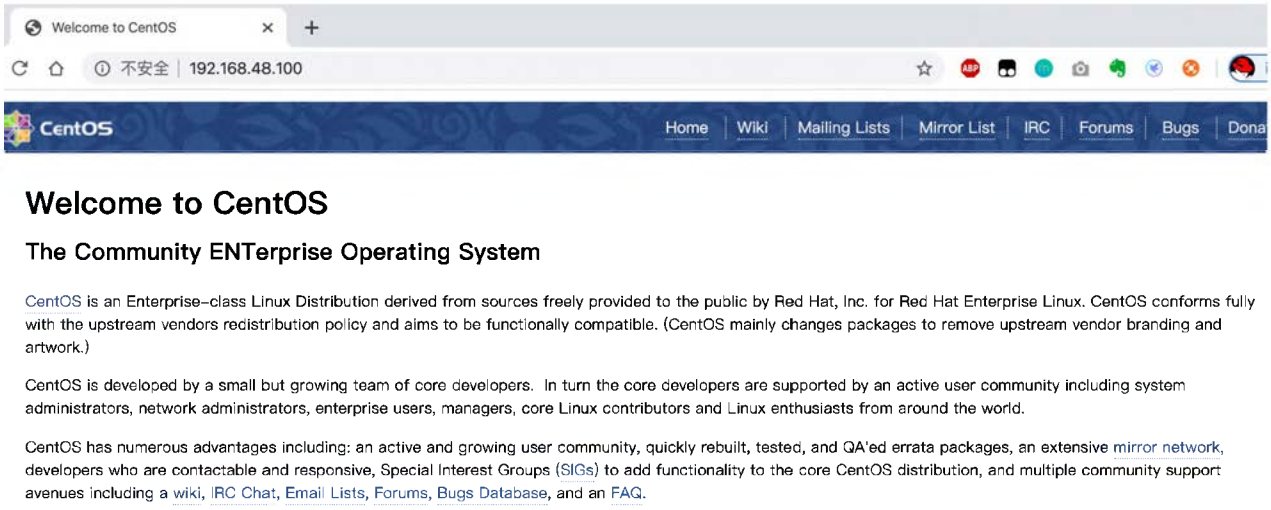
```
[root@web-nginx ~]# setenforce 0
[root@web-nginx ~]# sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/sysconfig/selinux

[root@web-nginx ~]# systemctl disable firewalld
Removed symlink /etc/systemd/system/multi-user.target.wants/firewalld.service.
Removed symlink /etc/systemd/system/dbus-org.fedoraproject.FirewallD1.service.
[root@web-nginx ~]# systemctl stop firewalld
```

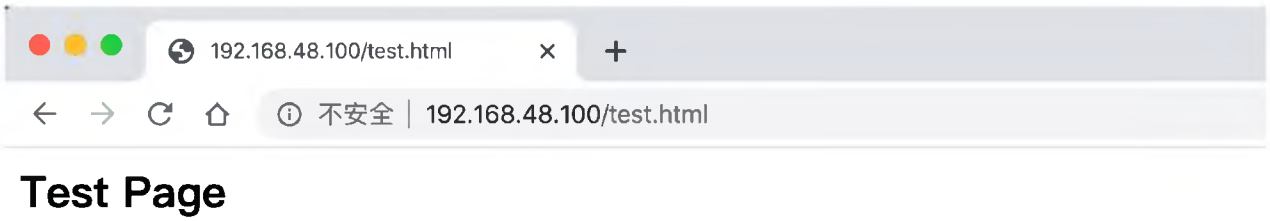
【访问】

```
❯ curl -I 192.168.48.100

HTTP/1.1 200 OK
Server: nginx/1.16.1
Date: Tue, 17 Mar 2020 09:23:26 GMT
Content-Type: text/html
Content-Length: 4833
Last-Modified: Fri, 16 May 2014 15:12:48 GMT
Connection: keep-alive
ETag: "53762af0-12e1"
Accept-Ranges: bytes
```



这样最简单的 `Nginx` `Web` 服务器就已经启动好了。我们可以写个简单的 `test.html` 页面进行测试，如下：



注意，我们要把页面内容放在 `/usr/share/nginx/html` 下，至于为什么要放在这个目录下，可不可以放在其他目录，如何指定网页文件存储位置，我们都将在后面的课程中进行详细的介绍。

---

3. Nginx 简单配置

我们通过 `yum` 方式安装的 `Nginx` 程序，配置文件采用 **"一主多子"** 的配置方式，即一个主配置文件，通过将不同功能或者不同模块的子配置文件进行包含引入的方式，这样可以做到配置文件的清晰、便于管理。这种配置方式的思想也是一个优秀的工程师需要借鉴的地方。下面我们来看下 `Nginx` 默认的主配置文件，注意此处我省略了部分内容。

```
[root@web-nginx nginx]# cat nginx.conf | grep -v -E '^.*#|^$'
```

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
include /usr/share/nginx/modules/*.conf;
events {
    worker_connections 1024;
}
http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';
    access_log  /var/log/nginx/access.log  main;
    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;
    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;
    include /etc/nginx/conf.d/*.conf;
    server {
        listen       80 default_server;
        listen       [::]:80 default_server;
        server_name  _;
        root         /usr/share/nginx/html;
        include /etc/nginx/default.d/*.conf;
        location / {
        }
        error_page 404 /404.html;
            location = /40x.html {
        }
        error_page 500 502 503 504 /50x.html;
            location = /50x.html {
        }
    }
}
```

3.1 main 配置段

其中：

```
user nginx;
worker_processes auto;
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
include /usr/share/nginx/modules/*.conf;
events {...};
```

这些配置为 `nginx` 核心配置（也称为主配置 `main`），可以使用的参数可以参考 官方文档 。这里有一些参数非常重要，涉及到 `Nginx` 的性能优化，我们会在后面的篇幅中进行深入且全面的介绍。此处我们仅介绍默认配置文件中出现的参数：

`user nginx`：用来指定运行 `nginx` 程序的用户，默认为 `nginx`，注意该用户是安装 `nginx` 时程序自动创建的，如果我们想要手动编译安装 `Nginx` 的话，则需要首先手动创建相应的用户。

> **可以看到我们在安装 `nginx` 时，有一个依赖包 `依赖安装 nginx-filesystem-1:1.16.1-1.el7.noarch @epel` 被安装，我们可以查看下该依赖包安装时执行了什么脚本。**
>
> ```
> [root@web-nginx nginx]# rpm -q --scripts  nginx-filesystem
> preinstall scriptlet (using /bin/sh):
> getent group nginx > /dev/null || groupadd -r nginx
> getent passwd nginx > /dev/null || \
>  useradd -r -d /var/lib/nginx -g nginx \
>  -s /sbin/nologin -c "Nginx web server" nginx
> exit 0
> ```
>
> **即：`nginx` 用户就是在这个程序安装时添加的。**

需要注意的是，Nginx 主进程，即 master 进程会以 root 身份运行，worker 进程会以指定的 nginx 用户运行。所以用户的某个请求能否处理，则要看 worker 进程的用户权限是否足够。而能否监听某个端口，则是由 master 用户权限决定的。

`worker_processes auto`：用来指定 `worker` 进程的个数，`auto` 则会根据系统的 `cpu` 个数进行设置，比如我有 4 颗 `cpu`，则 `worker` 进程的个数就为 4。

```
[root@web-nginx html]# lscpu |grep -E '^CPU\(s\)'
CPU(s):                4
[root@web-nginx html]# ps -ef | grep nginx
root      40696      1  0 05:51 ?        00:00:00 nginx: master process nginx
nginx     40697  40696  0 05:51 ?        00:00:00 nginx: worker process
nginx     40700  40696  0 05:51 ?        00:00:00 nginx: worker process
nginx     40701  40696  0 05:51 ?        00:00:00 nginx: worker process
nginx     40702  40696  0 05:51 ?        00:00:00 nginx: worker process
root      40756   2176  0 06:18 pts/0    00:00:00 grep --color=auto nginx
```

我们也可以手动指定 `worker` 进程的个数，或者绑定某个 `worker` 进程在某个 `cpu` 上运行，这部分内容我们会在调优部门深入介绍。

`error_log /var/log/nginx/error.log`：用来定义 `nginx` 错误日志位置，我们也可以对不同的虚拟主机或者 `uri` 定义不同的错误日志，该参数可以在 `main`，`http`，`mail`，`stream`，`server`，`location` 部分应用。

`pid /run/nginx.pid`：用来定义 `nginx` 主进程的 `pid` 文件路径，注意：对于 `nginx` 应用的启动、停止、重载等信号传递操作都是依赖于该文件；

`include /usr/share/nginx/modules/*.conf`：用于指定 `nginx` 模块配置文件所包含的子配置文件；

`events {...}`：用于定义事件驱动相关配置，该配置与连接的处理密切相关，其中最重要的几个指令如下：

```
use method; # 定义 Nginx 使用那种事件驱动类型，在 Redhat/CentOS 中性能最好的是 epoll 模型
worker_connections number; # 定义每个 worker 进程可以处理的连接数
accept_mutex on|off; # 处理新连接的方法，on 是指由各个 worker 进程轮流处理，off 则会通知所有 worker 进程，但是只有一个 worker 进程获得处理连接的权限。在 CentOS7( Linux 3.9+) 将使用 "reuseport" 会有更好性能；
```

> **注意：如果启用了 `accept_mutex on`，则会启用 `accept_mutex_delay 500ms;` 参数，时间可以设置，表示当一个 `worker` 进程在处理新连接时，多长时间以后才会重新接受下一个新的请求。**

这些参数为主配置段的一些常见配置参数，下面的 `http` 配置段则是用来配置 `http` 协议的部分内容，我们将对此进行简单介绍。

3.2 http 模块配置段

在上面的配置文件示例中，处理 `main` 配置段，还有 `http` 模块配置段，这两个参数是平行关系，即不是包含与被包含的关系。于此平行关系的参数还有 `mail`、`stream`，这些我们后面再细说。本节内容我们只 `http` 配置做简单介绍。首先，认识下 `http` 的配置格式，除去一些配置参数，简单的架构如下：

```
http {
    ...
    server {
        server_name www.byte-edu.com;
        ...
        location / {
          ...
        }
        location /img/ {
          ...
        }
    }
    server {
        server_name ke.byte-edu.com;
        ...
        location / {
          ...
        }
        location /course/ {
          ...
        }
```

```
    }
}
```

我们可以看到在 `http` 配置段中，我们可以设置多个 `server` 配置，该 `server` 就是用来配置虚拟主机的，可以基于 `IP` 地址，也可以基于 `PORT`，当然生产上更多的还是使用基于域名的的方式来配置虚拟主机。在 `server` 配置段内还可以在配置多个 `location` 字段，该字段用来配置虚拟主机不同 `uri` 的响应方式，这就是 `nginx` 作为 `web` 应用的最简单配置格式。

无论是 `http` 、`server` 还是 `location` 字段，其中都可以使用很多很多参数，我们会对生产上经常使用的参数进行介绍，如果同学们在使用中遇到一些未见的参数，可以多看看官方文档。本节内容，我们先简单配置两个虚拟主机，后面的课程再作更深入的介绍。

4. 配置虚拟主机

基于域名的最简单的配置：

我们在 `/etc/nginx/conf.d` 下创建子配置文件，至于为什么在这里创建，可以看上面的 `http` 配置段中 `include /etc/nginx/default.d/*.conf;` 。生产环境中，我们一般一个虚拟主机会单独创建一个配置文件，希望大家也有这样的好习惯。

```
# 创建几个备用目录，作为不同虚拟主机的家目录
[root@web-nginx conf.d]# mkdir /data/nginx/{a,b,c} -pv
mkdir: 已创建目录 "/data"
mkdir: 已创建目录 "/data/nginx"
mkdir: 已创建目录 "/data/nginx/a"
mkdir: 已创建目录 "/data/nginx/b"
mkdir: 已创建目录 "/data/nginx/c"
```

【创建域名 `a.com|www.a.com` 的虚拟主机】

```
[root@web-nginx conf.d]# cat a.com.conf
server {
    listen        80;
    server_name   www.a.com a.com;
    root          /data/nginx/a;
}
```

这里我们写了四行配置，其实 `listen` 可以省略，默认监听在 80 端口；`root` 也可以省略，如果我们不再 `server` 中指定，则会继承 `http` 中定义的 `root` 目录作为自己的家目录。

【创建域名 `b.com|www.b.com` 的虚拟主机】

```
[root@web-nginx conf.d]# cat b.com.conf
server {
    server_name   www.b.com b.com;
    root          /data/nginx/b;
}
```

【创建测试页面】

```
[root@web-nginx conf.d]# echo "This is a.com page" > /data/nginx/a/index.html
[root@web-nginx conf.d]# echo "This is b.com page" > /data/nginx/b/index.html
```

【测试基于域名的虚拟主机】

```
[root@web-nginx conf.d]# nginx -s reload
```

```
❯ curl www.a.com
This is a.com page
❯ curl a.com
This is a.com page
❯ curl www.b.com
This is b.com page
❯ curl b.com
This is b.com page
```

> **注意：如果我们想要通过域名测试，记得先在 `hosts` 中指定域名，我的 `host` 列表：**
>
> ```
> 192.168.48.100  a.com b.com c.com www.a.com www.b.com www.c.com
> ```

【基于 IP 地址的虚拟主机】

这种方式很少使用，我们做个简单示例：

创建配置文件：

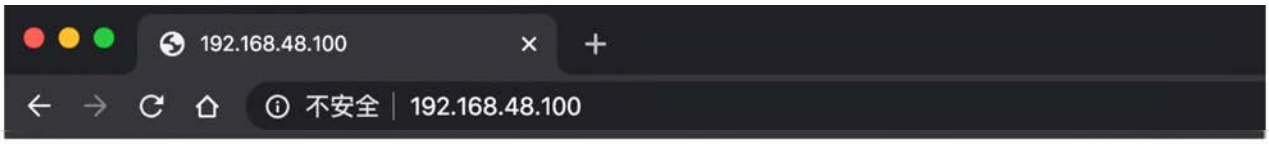
```
[root@web-nginx conf.d]# cat ip.conf
server {
    listen      192.168.48.100;
    root        /data/nginx/ip;
}
```

创建相应的测试目录与测试页面：

```
[root@web-nginx conf.d]# mkdir /data/nginx/ip -p
[root@web-nginx conf.d]# echo "This is IP page." > /data/nginx/ip/index.html
[root@web-nginx conf.d]# nginx -s reload
```

通过 `IP` 地址进行访问测试：

```
❯ curl 192.168.48.100
This is IP page.
```



This is IP page.

还记得我们刚安装好 `nginx` 通过 `IP` 地址访问时的页面吧，那是因为之前的页面作为 `default` 域名进行响应的，现在我们配置好相应的页面，就不再通过默认域名响应。

【基于端口号的虚拟主机】

这种情况也不是很常用，一般跟基于 `IP` 的虚拟主机方式一样，多数都是用于内部应用。

创建相应的配置文件：

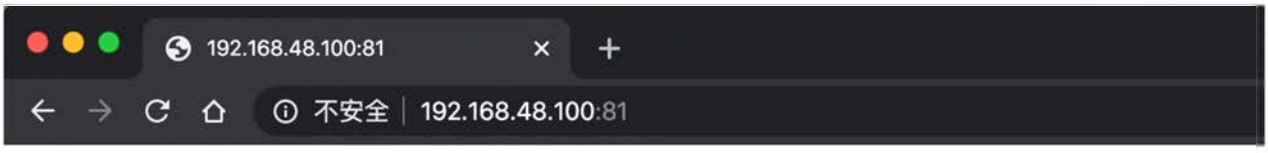
```
[root@web-nginx conf.d]# cat port.conf
server {
    listen      81;
    root        /data/nginx/port;
}
```

创建相应的目录与网页文件：

```
[root@web-nginx conf.d]# mkdir -pv /data/nginx/port; echo "This is Port page."
> /data/nginx/port/index.html
mkdir: 已创建目录 "/data/nginx/port"
[root@web-nginx conf.d]# nginx -s reload
```

基于 `IP` 地址的访问测试：

```
❯ curl 192.168.48.100:81
This is Port page.
❯ curl a.com:81
This is Port page.
❯ curl b.com:81
This is Port page.
```



This is Port page.

结论：所谓基于域名、IP 或者端口的虚拟主机，其实就是指当客户端通过不同的域名、IP 或者端口进行请求时，服务端会给与不同的响应。