1. 常见web服务器：

```
nginx、tengine、httpd、tomcat、IIS、lighttpd

httpd|IIS          政府类，银行用的居多
nginx|tomcat|tengine          社区，电商用的多

nginx          新浪、
httpd          中国人民政府网站、兴业银行...
IIS            招商银行、工商银行、农业银行...
tengine(2012)   简书、csdn、淘宝...
```

2. nginx 的发展：

- 第一个公开版本0.1.0发布**于2004**年10月4日。
- Nginx 的1.4.0稳定版已经于**2013**年4月24日。
- Nginx目前最新的版本是1.18.0，于 2020年4月21号发布。

3. nginx的安装方式：

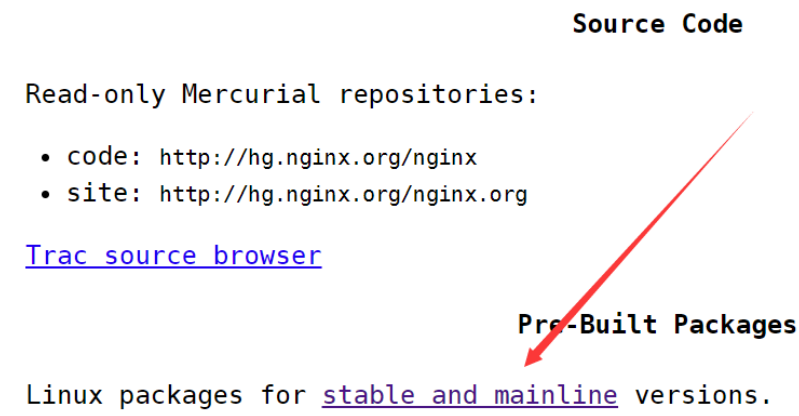
- YUM安装部署
- 源码安装部署

3.1 yum安装：

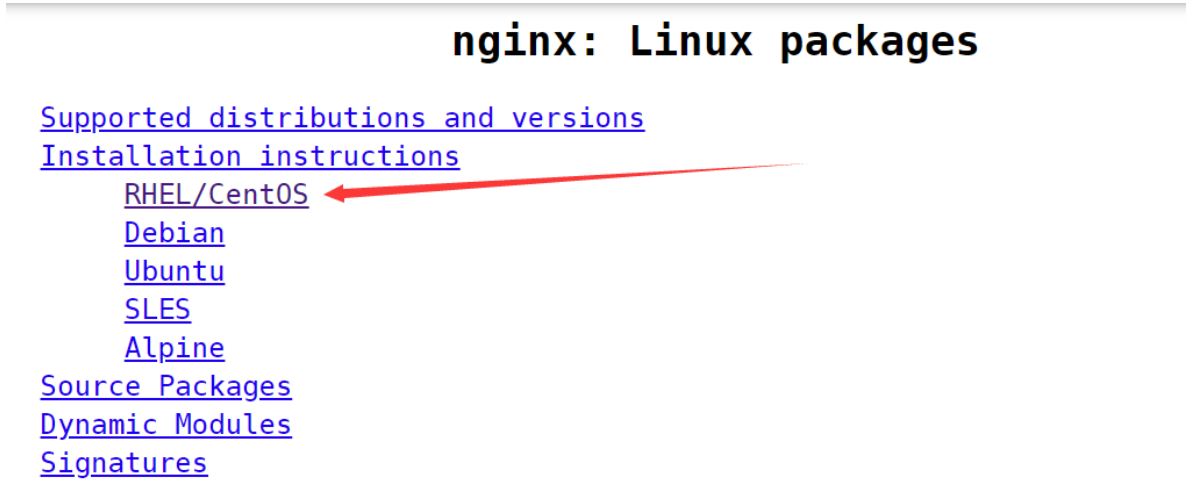
3.1.1 配置yum仓库：

- 登录http://nginx.org官网，点击右侧的**download**：

- 点击 当前页最下方的stable and mainline：



- 然后选择RHEL/CentOS：



- 复制配置文件，创建nginx.repo文件：

```
vim /etc/yum.repos.d/nginx.repo

[nginx-stable]
```

```
name=nginx stable repo
baseurl=http://nginx.org/packages/centos/$releasever/$base
arch/
gpgcheck=1
enabled=1
gpgkey=https://nginx.org/keys/nginx_signing.key

[nginx-mainline]
name=nginx mainline repo
baseurl=http://nginx.org/packages/mainline/centos/$release
ver/$basearch/
gpgcheck=1
enabled=0
gpgkey=https://nginx.org/keys/nginx_signing.key
```

- 现在就可以直接安装了，默认情况是安装稳定版的：

3.1.1 安装稳定版本：

```
[root@localhost ~]# yum install nginx -y
[root@localhost ~]# nginx -v
nginx version: nginx/1.16.0
```

3.1.2 安装主线版本：

```
# 先安装yum的扩展包，开启nginx的主线版本仓库：
yum  install yum-utils -y
yum-config-manager --enable nginx-mainline
yum install nginx -y
```

这个时候nginx 1.16的版本会直接被升级。

```
[root@localhost ~]# nginx -v
nginx version: nginx/1.17.1
```

3.2 源码安装：

登录http://nginx.org官网，点击右侧的**download**：

- 主线版本：也叫开发版本，目前最新但是还没有经过大量测试的版本。
- 稳定版本：稳定版通常是经过大量测试的，相对比较稳定的版本，企业中我们也会使用稳定版。
- 历史版本：通常是往期的稳定版本。

3.2.1 下载包：

```
# 选择想要下载的版本，直接单击右键复制地址下载：
wget http://nginx.org/download/nginx-1.16.0.tar.gz
```

3.2.2 安装依赖：

```
yum install gcc  gcc-c++ pcre pcre-devel zlib zlib-devel
openssl openssl-devel -y
```

3.2.3 解压包：

```
tar xf nginx-1.16.0.tar.gz
cd nginx-1.16.0
```

3.2.4 预编译：

```
预编译主要是用来检查系统环境是否满足安装软件包的条件，
并生成Makefile文件，该文件为编译、安装、升级nginx指明了相应参数。

./configure --help 可以查看预编译参数
--prefix       指定nginx编译安装的目录；
--user=***     指定nginx的属主
--group=***    指定nginx的属主与属组
--with-***      指定编译某模块
--without-**    指定不编译某模块
--add-module   编译第三方模块
```

开始预编译：

```
./configure --prefix=/usr/local/nginx
```

```
[root@www nginx-1.16.0]# cat Makefile

default:    build

clean:
    rm -rf Makefile objs

build:
    $(MAKE) -f objs/Makefile

install:
    $(MAKE) -f objs/Makefile install

modules:
    $(MAKE) -f objs/Makefile modules

upgrade:
        /usr/local/nginx/sbin/nginx -t
    kill -USR2 `cat /usr/local/nginx/logs/nginx.pid`
    sleep 1
    test -f /usr/local/nginx/logs/nginx.pid.oldbin

    kill -QUIT `cat
/usr/local/nginx/logs/nginx.pid.oldbin`

make clean : 重新预编译时，通常执行这条命令删除上次的编译文件
```

```
make build : 编译，默认参数，可省略build参数
make install : 安装
make modules : 编译模块
make  upgrade : 在线升级
```

3.2.5 编译并安装

```
make  &&  make install
```

3.2.6 查看版本

```
/usr/local/nginx/sbin/nginx -v
nginx version: nginx/1.16.0
```

3.2.7 启动nginx:

```
/usr/local/nginx/sbin/nginx
```

3.2.8 检查进程及端口:

```
# 查看进程:
[root@localhost ~]# ps -ef|grep nginx
root      6853     1  0 20:27 ?        00:00:00 nginx:
master process /usr/local/nginx/sbin/nginx
nobody    7839  6853  0 21:35 ?        00:00:00 nginx:
worker process
# 查看端口
[root@localhost ~]# netstat -ntlp|grep 80
tcp        0      0 0.0.0.0:80              0.0.0.0:*
          LISTEN      6853/nginx: master
```

可以看到端口及进程表示nginx WEB服务已经搭建成功!

4. nginx的常用命令:

使用/usr/local/nginx/sbin/nginx -h命令查看可用参数:

```
[root@localhost ~]# /usr/local/nginx/sbin/nginx  -h
nginx version: nginx/1.16.0
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/local/nginx/)
  -c filename   : set configuration file (default:
conf/nginx.conf)
  -g directives : set global directives out of
configuration file
```

4.1 命令解读：

```
-v      可查看nginx的版本。
-V      可查看nginx的详细信息，包括编译的参数。
-t      可用来测试nginx的配置文件的语法错误。
-T      可用来测试nginx的配置文件语法错误，同时还可以通过重定向备份
nginx的配置文件。
-q      如果配置文件没有错误信息时，不会有任何提示，如果有错误，则提
示错误信息，与-t配合使用。
-s      发送信号给master处理：
        stop    立刻停止nginx服务，不管请求是否处理完
        quit    优雅的退出服务，处理完当前的请求退出
        reopen  重新打开日志文件，原日志文件要提前备份改名。
        reload  重载配置文件
-p      设置nginx家目录路径，默认是编译时的安装路径
-c      设置nginx的配置文件，默认是家目录下的配置文件
-g      设置nginx的全局变量，这个变量会覆盖配置文件中的变量。
```

4.2 命令演示：

如果觉得每次都需要输入绝对路径执行命令麻烦，可以通过以下几种方法实现直接使用nginx命令。

```
1、做软连接：
ln  -s /usr/local/nginx/sbin/* /usr/local/sbin
然后重新读取下配置文件
. /etc/profile

ps:软连接做在PATH路径是第一位，因为yum安装的在/usr/sbin/目录下，
which安装PATH的顺序找到第一个，就不找了。

2、配置环境变量：
echo "export PATH=/usr/local/nginx/sbin:$PATH" >
/etc/profile.d/nginx.sh

然后重新读取下配置文件
source /etc/profile

ps:最好写在$PATH前面，否则，如果安装了yum版的nginx，
直接执行nginx会启动yum版的nginx，因为which nginx，会先找
到/usr/sbin/nginx文件

3、设置别名：

alias  nginx='/usr/local/nginx/sbin/nginx'

ps:which优先找别名
```

4.2.1 启动nginx：

```
nginx
```

4.2.2 立即停止nginx：

```
nginx  -s stop
```

4.2.3 优雅停止nginx：

```
nginx -s quit
```

4.2.4 重新打开日志文件

该命令可用于日志切割，定期执行。

```
[root@localhost logs]# ls
access.log  error.log  nginx.pid
[root@localhost logs]# mv access.log{,.bak}
[root@localhost logs]# ls
access.log.bak  error.log  nginx.pid
[root@localhost logs]# /usr/local/nginx/sbin/nginx -s
reopen
[root@localhost logs]# ls
access.log  access.log.bak  error.log  nginx.pid
```

4.2.5 重载配置文件：

修改工作进程数，重启服务，对比前后进程数

```
[root@localhost logs]# ps -ef|grep nginx
root      2685     1  0 23:56 ?        00:00:00 nginx:
master process /usr/local/nginx/sbin/nginx
nginx     2686  2685  0 23:56 ?        00:00:00 nginx:
worker process
root      2691  2532  0 23:57 pts/1    00:00:00 grep --
color=auto nginx
[root@localhost logs]# vim
/usr/local/nginx/conf/nginx.conf
[root@localhost logs]# /usr/local/nginx/sbin/nginx -s
reload
[root@localhost logs]# ps -ef|grep nginx
root      2685     1  0 23:56 ?        00:00:00 nginx:
master process /usr/local/nginx/sbin/nginx
nginx     2694  2685  0 23:58 ?        00:00:00 nginx:
worker process
nginx     2695  2685  0 23:58 ?        00:00:00 nginx:
worker process
nginx     2696  2685  0 23:58 ?        00:00:00 nginx:
worker process
root      2698  2532  0 23:58 pts/1    00:00:00 grep --
color=auto nginx
```

4.2.6 启动指定的配置文件：

在/data/目录下拷贝一份nginx的配置文件，然后修改用户名为www。

注意：配置文件中引用的其他配置文件路径也要做一个修改。

```
[root@localhost logs]# cp /usr/local/nginx/conf/nginx.conf
/data/
[root@localhost logs]# vim /data/nginx.conf
[root@localhost logs]# /usr/local/nginx/sbin/nginx  -c
/data/nginx.conf
[root@localhost logs]# ps -ef|grep nginx
root      2736     1  0 00:05 ?        00:00:00 nginx:
master process /usr/local/nginx/sbin/nginx -c
/data/nginx.conf
www       2737  2736  0 00:05 ?        00:00:00 nginx:
worker process
www       2738  2736  0 00:05 ?        00:00:00 nginx:
worker process
www       2739  2736  0 00:05 ?        00:00:00 nginx:
worker process
root      2741  2532  0 00:05 pts/1    00:00:00 grep --
color=auto nginx
```

4.2.7 设置全局变量

通过设置全局变量，让nginx在前端运行。

```
[root@localhost logs]# /usr/local/nginx/sbin/nginx  -g
"daemon off;"

现在当前nginx在前端运行，
输入ctrl +c，则nginx就退出了。
可以使用ctrl +z放置后台运行。
```

5. nginx配置文件路径：

不同安装方式，nginx的文件存放路径也有所不同。

5.1 源码安装配置文件路径：

在安装目录下的conf目录下，比如我的安装目录是/usr/local/nginx，那么他的配置文件就在/usr/local/nginx/conf目录下。

5.2 yum安装配置文件路径：

在/etc/nginx/目录（主配置文件）与/etc/nginx/conf.d目录下。

6. nginx配置文件的结构：

通常源码安装的nginx的配置文件，会是下面这种结构，yum安装的有细微差异（大致是一样的，只是server是通过include引用的独立配置文件）。

```
...
events {
        ...
}

http {
    ...
    server {
        ....
        location / {
            root html;
            ...
        }
    }
}
```

nginx的配置指令可以分为两大类：指令块与单个指令。

指令块就是像events，http，server等；

单独指令就是像root html;这样的。

nginx规定指令块可以嵌套，如http块中可以嵌套server指令，server块中可以嵌套location指令，指令可以同时出现在不同的指令块，如root指令可以同时出现在http、server和location指令块，需要注意的是在location中定义的指令会覆盖server，http的指令。

7. 解析配置文件：

7.1 全局配置：

```
user  nobody;
worker_processes  1;
#error_log  logs/error.log;
#error_log  logs/error.log  notice;
#error_log  logs/error.log  info;
#pid        logs/nginx.pid;

events {
    use epoll;
    worker_connections  1024;
}
```

user :指定nginx的工作进程的用户及用户组，默认是nobody用户。

worker_processes :指定工作进程的个数，默认是1个。具体可以根据服务器cpu数量进行设置，
比如cpu有4个，可以设置为4。如果不知道cpu的数量，可以设置为auto。
nginx会自动判断服务器的cpu个数，并设置相应的进程数。

error_log :设置nginx的错误日志路径，并设置相应的输出级别。
如果编译时没有指定编译调试模块，那么 info就是最详细的输出模式了。
如果有编译debug模块，那么debug时最为详细的输出模式。这里设置为默认就好了。

pid :指定nginx进程pid的文件路径。

events :这个指令块用来设置工作进程的工作模式以及每个进程的连接上限。
use :用来指定nginx的工作模式，通常选择epoll，除了epoll，还有select,poll。
worker_connections :定义每个工作进程的最大连接数，默认是1024。

ps:进程的最大连接数受Linux系统进程的最大打开文件数限制。

修改文件描述符方式：
临时生效： ulimit -n 65535

在压测的时候，如果遇到报错 apr_socket_recv: Connection reset by peer (104):
解决办法：
临时解决：
加一个-r参数，避免因为套接字错误退出，但是影响测试结果。

# 根本解决：
# vim /etc/sysctl.conf
net.ipv4.tcp_syncookies = 0

然后执行：sysctl -p

7.2 http指令块：

```
http {
    include       mime.types;
    default_type  application/octet-stream;
    charset utf-8;
    #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  logs/access.log  main;
    Sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    #keepalive_timeout  0;
    keepalive_timeout  65;
    keepalive_requests 100;
    #gzip  on;
    server {
        ...
        location {
            root html;
            ...
        }
    }

}
```

```
include    mime.types;  定义数据类型
    如果用户请求lutixia.png，服务器上有lutixia.png这个文件，后缀名是png；
    根据mime.types，这个文件的数据类型应该是image/png；
    将Content-Type的值设置为image/png，然后发送给客户端

default_type          :设定默认类型为二进制流，也就是当文件类型未定义时使用这种方式，
                      例如在没有配置PHP环境时，Nginx是不予解析的，
                      此时，用浏览器访问PHP文件就会出现下载窗口。
charset utf-8;    解决中文字体乱码
log_format        :定义日志文件格式，并默认取名为main，可以自定义该名字。
                   也可以通过添加，删除变量来自定义日志文件的格式。

access_log        :定义访问日志的存放路径，并且通过引用log_format所定义的main名称设置其输出格式。
sendfile    on :用于开启高效文件传输模式。直接将数据包封装在内核缓冲区，然后返给客户，将tcp_nopush和tcp_nodelay两个指令设置为on用于防止网络阻塞；
keepalive_timeout   65     :设置客户端连接保持活动的超时时间。在超过这个时间之后，服务器会关闭该连接。
keepalive_requests 100    :设置nginx在保持连接状态最多能处理的请求数，到达请求数，即使还在保持连接状态时间内，也需要重新连接。

提示：可以用netstat -ntlpa |grep 80   查看链接状态
gzip  on          :开启压缩功能，减少文件传输大小，节省带宽。
gzip_min_length 1k; 最小文件压缩，1k起压。
gzip_types       text/plain text/xml;     压缩文件类型
gzip_comp_level   3;    压缩级别，默认是1。
```

7.3 server指令块：

```
server {
        listen       80;
        server_name  localhost;
        #charset koi8-r;
        #access_log  logs/host.access.log  main;
         index  index.html index.htm;
```

```
        location /
         {
            root   html;
            ...
        }
        #error_page  404              /404.html;
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
        #location ~ \.php$ {
                ...
        #}
        #location ~ /\.ht {
        #    deny  all;
        #}
    }
```

```
server       :用来定义虚拟主机。
listen       :设置监听端口，默认为80端口
server_name      :域名，多个域名通过逗号隔开
Charset      :设置网页的默认编码格式
access_log       :指定该虚拟主机的独立访问日志，会覆盖前面的全局配置。
index         :设置默认的索引文件
location        :定义请求匹配规则。
error_page       :定义访问错误返回的页面，凡是状态码是500 502 503 504 都会返回这个页面。
```

7.4 location指令块：

```
 #location ~ \.php$ {
        #    root           html;
        #    fastcgi_pass   127.0.0.1:9000;
        #    fastcgi_index  index.php;
        #    fastcgi_param  SCRIPT_FILENAME
/scripts$fastcgi_script_name;
        #    include        fastcgi_params;
        #}
```

```
        # deny access to .htaccess files, if Apache's
document root
        # concurs with nginx's one
        #
        #location ~ /\.ht {
        #    deny  all;
        #}
```

```
location ~ \.php$      :凡是以php结尾文件，都会匹配到这条规则。
root         :php文件存放的目录
fastcgi_pass         :指定php-fpm进程管理的ip端口或者unix套接字
fastcgi_index    :指定php脚本目录下的索引文件
fastcgi_param        :指定传递给FastCGI服务器的参数
location ~ /\.ht         :凡是请求类似.ht资源，都拒绝。
```