Caddy使用手册

https://docboon.github.io/

docboon

Caddy使用手册:

https://docboon.github.io/

Mr. Older Flower

出版日期 October 10th,2021

[docID: docboon20221027084713]

修订历史		
修订 0.1	2021年11月2日.	<olderflower[AT]outlook. com>
文档建立		

目录

插图清单

表格清单

排版约定

在正文中会有一些诸如程序代码、系统命令或是屏幕输出一类的信息，为了能清晰地展示各种元素，本手册遵守下列排版约定。

1. **字体约定**，在正文中嵌入的一些系统命令、文件名、函数或是参数等使用有别于正文的字体：

   a. 等宽字体：程序片段、正文中出现的配置选项、变量、函数名等，我们采用等宽字体，样式如下：`serverip`。

   b. 等宽微粗字体：表示由用户输入的系统命令。例如，在linux下，我们查网络接口信息的命令：`ifconfig eth0`。

   c. 等宽斜体：表示由用户输入的值或是一些需要设定的参数的值，例如，用户在浏览器里输入的一个url，用如下的样式显示：*`http://docboon.io`*。

   d. 等宽微粗斜体字体：表示文件名、数据库名及新的术语等。例如，文件的名称用如下的样式来显示：***`myfirst.xml`***。

2. **提示、告诫类信息**（admonitions），有一些内容是正文的补充或是对用户的一种提醒。为了能清晰地展示这部分内容，采用独立的“信息块”来呈现。一共有5种，分别为：

   - 

     提示

     可以使用`apt-get upgrade`来升级您的ubuntu或debian Linux系统。

   - 

     注意

     在运行`apt-get upgrade`之前，需要先运行用`apt-get update`命令。

   - 

     重要

     请及时升级系统补丁程序，这是保证系统处于安全状态的好方法。

   - 

     小心

     如果系统升级还没有完成，请不要重新启动您的电脑。

   - 

     警告

     在系统升级过程中千万不要关闭电源。

3. **屏幕内容围栏及程序代码围栏：**

   - 屏幕内容围栏，有时候我们需要整块显示在屏幕上输出的内容，或者是键入的命令。例如，在linux下，我们使用`ifconfig eth0`命令来查网络接口信息：

```
$ ifconfig eth0
eth0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether 98:fa:9b:db:b9:9f  txqueuelen 1000  (以太网)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
```

```
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 16  memory 0xea300000-ea320000
```

- 程序代码围栏，对于一些程序代码，使用带有序号的样式来展示，例如，一段php代码：

```
 1 <?php
 2   class SimpleBook {
 3     private $title;
 4
 5     function __construct($title_in) {
 6       $this->title  = $title_in;
 7     }
 8
 9     function getTitle() {return $this->title;}
10   }
11 ?>
```

第 1 章 关于caddy

1. 简介

caddy目前是ZeroSSL在维护的一个开源项目。ZeroSSL[1]和Let's Encrypt[2]一样，是一个SSL证书颁发机构。Caddy属于web服务器界的新秀。用go语言开发。与其他的web服务器相比，因为有了ZeroSSL的加持，caddy最大的一个特点就是它自带TLS，能非常方便的部署HTTPS。按照官方步骤安装好了以后无需任何配置，就已经自带了https功能。同时，caddy也非常好地支持成诸如代理、负载均衡等功能。在HTTP/2协议上，caddy表现也非常好。和很多的开源软件一样，caddy的文档也并不是特别的全面，有一些选项和指令需要自己去实验才能明白具体如何使用。

caddy的另外一个特点，是可以通过nginx-adapter[3]模块来使用nginx的配置文件。caddy第一个版本发布于2015年4月，截至本文截稿，caddy的最新版本是2.4.5。

https://caddyserver.com/

2. 安装

caddy针对不同的操作系统和平台都有不同的发行包。也有静态的已经编译好的二进制文件,可以直接下载使用。详细的安装请参考这个页面：https://caddyserver.com/docs/install。在这里我们仅给出二进制文件和deb的安装方式。

2.1. 二进制静态包

二进制包安装方式并不会将caddy安装为一个服务（service），仅仅是一个可以执行的程序。这种方式更适合开发和调试。直接到caddy的下载站点：https://caddyserver.com/download下载适合自己操作系统的压缩包就可以了。如果是Linux系统，将解压缩后的caddy命令复制到*/usr/local/bin*目录下就可以了。例如：

```
cp caddy_linux_amd64 /usr/local/bin/caddy
```

[1]https://zerossl.com/

[2]https://letsencrypt.org/

[3]https://github.com/caddyserver/nginx-adapter

2.2. deb包方式

在Debian、Ubuntu或Raspbian基于debian包的系统可以使用deb包的安装方式。这种方式会将caddy安装为一个**systemd service**，同时，`caddy-api`这个服务也会被自动安装。

1. **Stable版本系统**

```
$ sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https
$ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | \
      sudo tee /etc/apt/trusted.gpg.d/caddy-stable.asc
$ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | \
      sudo tee /etc/apt/sources.list.d/caddy-stable.list
$ sudo apt update
$ sudo apt install caddy
```

2. **Testing版本系统**

```
$ sudo apt install -y debian-keyring debian-archive-keyring apt-transport-https
$ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/testing/gpg.key' | \
      sudo tee /etc/apt/trusted.gpg.d/caddy-testing.asc
$ curl -1sLf 'https://dl.cloudsmith.io/public/caddy/testing/debian.deb.txt' | \
      sudo tee /etc/apt/sources.list.d/caddy-testing.list
$ sudo apt update
$ sudo apt install caddy
```

3. caddy的配置方式

作为概览，在本节我们从一些基础的方面展示如何使用caddy。

caddy的配置和大多数的web server不太一样。有三种不同的配置方式：

1. **JSON**

   caddy的原生配置是JSON格式。使用JSON格式来配置caddy可以实现它的全部功能。如果不喜欢JSON的配置方式，也可是使用caddy所具备的“适配器（adapter）”功能来使用其他的配置格式。其中，Caddyfile就是“adapter”的一种。另外，caddy也可以通过使用“nginx adapter”来使用nginx的配置文件。

2. **Caddyfile**

   这是一种比较实用和便捷的方式。在这种方式下，使用*`Caddyfile`*这个文件来配置系统。这个文件的配置语法比较容易阅读，并且也能够胜任大部分的使用场景。它的样子如下：

```
1 example.com
2
3 root * /var/www/wordpress
4 php_fastcgi unix//run/php/php-version-fpm.sock
5 file_server
```

   这个文件可以保存于系统中的任何位置。只需在启动时使用如下命令进行指定就可以了。

```
caddy adapt --config /path/to/Caddyfile
```

caddy使用的配置文件是JSON格式的，其他格式的文件，例如Caddyfile，需要使用配置适配器将其转换为JSON格式。caddy利用这种机制可以使用其他类型的配置文件，目前支持8种：Caddyfile、nginx的配置文件、jsonc、josn5、yaml、cue、toml、及hcl

3. caddyAPI

   caddy可以通过一个管理端点（administration endpoint）来配置，这个管理端点是一套通过HTTP协议使用来使用的REST API。这个“管理端点”可以在配置文件中进行配置。它的默认使用地址是：`localhost:2019`

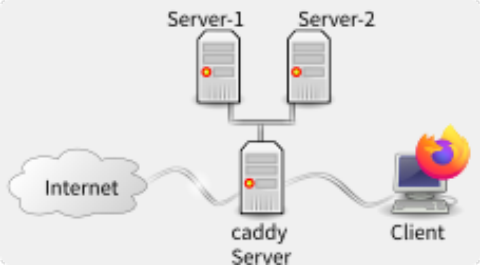
上面所说到的，是caddy的三种配置和管理的方式。在后面的章节会陆续进行说明。

如果是使用某种包管理方式安装的caddy，那么caddy已经是一个系统服务了，可能已经启动了。在开始这节的内容之前，建议先将caddy停止运行。可以使用`caddy stop`命令来停止caddy服务。下面，我们通过如下的几个实验来对caddy进行一个大致的了解。

第 2 章 使用入门

为了能更清晰地描述，我们在局域网中设立一台caddy服务器，我们称为**Server**，其内网IP地址是192.168.31.2/24，外网IP地址是192.168.1.2/24。我们在其上安装Debian Linux（V12）。

图 2.1. caddy服务器网络拓扑图



同时，我们需要在客户端（客户端使用了Ubuntu桌面系统）和caddy server端安装两个工具：

1. curl

   curl，即client-URL，可以认为是一个命令行的浏览器，在Debian/Ubuntu下使用apt-get install curl命令来安装。它的官方站点是https://curl.se/

2. jq

   jq 是一个基于命令行的、轻量级JSON文件处理工具，caddy的原生配置文件是JSON文件，我们需要用jq命令来将其格式化，以便于阅读。在Debian或Ubuntu下可通过apt-get install jq来安装。关于其更多内容请访问：https://stedolan.github.io/jq/。

1. 运行守护进程（Run the daemon）

在终端输入caddy命令，会显示help信息。如果想让caddy作为守护进程来运行，有两个子命令可以使用：run和start。在终端中输入caddy run或caddy start即可。run和start的区别是：

1. run：开启caddy进程，可以随时使用（Ctrl+C）或关闭终端来结束其运行。
2. start：开启caddy进程，并在后台运行。可以使用caddy sotp来结束其运行。

在配置文件为空的情况下，运行caddy run或caddy start之后，系统会开启一个本地的2019端口，这个端口就是caddy的API管理接口。可以通过这个接口对caddy完成很多的操作。

下面的内容就是使用caddy start启动系统之后的屏幕内容，通过使用netstat -ptuln命令，可以看到，caddy开启了本地的2019管理端口：

```
root@lognovel:~# caddy start
```

```
2021/11/08 05:46:38.499 INFO admin admin endpoint started {"address": "tcp/localhost:2019", \
         "enforce_origin": false, "origins": ["localhost:2019", "[::1]:2019", "127.0.0.1:2019"]}
2021/11/08 05:46:38.499 INFO serving initial configuration
Successfully started Caddy (pid=1850) - Caddy is running in the background
root@lognovel:~# netstat -ptuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address    State       PID/Program name
tcp        0      0 127.0.0.1:2019          0.0.0.0:*          LISTEN      1850/caddy
... ...
```

注意

caddy所启动的2019端口L并非我们的web站点，而是“管理端点（administration endpoint）”，也就是caddy的API。在默认情况下，运行在本机的2019端口，并且只允许本机访问。可以使用`curl localhost:2019/config/`命令来查看系统配置。

2. Caddyfile配置文件

2.1. Hello World! (Caddyfile)

在caddy的配置方式中，Caddyfile文件对人来说是比较容易阅读的。我们编辑一个文件，名为*Caddyfile*（没有扩展名，C字母大写，其他字母小写！）。在*Caddyfile*中加入如下内容：

```
:2015
respond "Hello, world!"
```

现在停止caddy的运行，使用（Ctrl+C），如果是运行在守护进程状态下的话可以使用`caddy stop`命令。然后，使用如下的命令再次运行caddy

```
caddy run
```

这个命令会在当前目录下寻找*Caddyfile*文件，并直接加载它。如果*Caddyfile*这个文件保存在其他位置，那需要指明具体的路径：

```
caddy run --config /path/to/Caddyfile
```

实际上，上面的命令是下面一行命令的缩写，省掉了`--adapter caddyfile`，因为默认情况下caddy使用Caddyfile adapter，

```
caddy run --adapter caddyfile --config /path/to/Caddyfile
```

接下来，在另外一个终端中输入：`curl https://localhost`，就会看到”Hello World!“。

2.2. 配置多站点 (Caddyfile & API)

现在我们更改*Caddyfile*，在其中定义两个站点，内容如下：

```
localhost {
 respond "Hello, world!"
}

localhost:2016 {
 respond "Goodbye, world!"
}
```

我们使用API的方式将这些内容加载到系统中，使用如下的命令：

```
curl localhost:2019/load \
 -X POST \
 -H "Content-Type: text/caddyfile" \
 --data-binary @Caddyfile
```

然后使用`curl https://localhost:2016`命令，便会看到“Goodbye, world!”

3. 尝试使用caddyAPI & JSON

3.1. Hello World! (Heredoc)

使用`caddy run`或`caddy start`启动caddy，不要带任何配置文件。现在caddy已经在“空转”了，我们使用Heredoc[1]的方式，通过API接口给它加载一些配置：

```
curl localhost:2019/load \
    -X POST \
    -H "Content-Type: application/json" \
    -d @- << EOF
    {
        "apps": {
            "http": {
                "servers": {
                    "hello": {
                        "listen": [":2015"],
                        "routes": [
                            {
                                "handle": [{
                                    "handler": "static_response",
                                    "body": "Hello, world!"
                                }]
                            }
                        ]
                    }
                }
            }
        }
    }
EOF
```

把这个配置加载到caddy以后，我们在客户端使用浏览器访问caddy服务器的2015端口，*http://192.168.31.2:2015*，或者也可以使用命令：`curl http://192.168.31.2:2015`。我们会看到“Hello, world!”

3.2. Hello World! (JSON)

用Heredoc字符流的方式加载配置还是有点费劲的，前面的例子只是让我们知道caddy的这种工作方式。更常用的方式还是把配置写在一个文件中，然后再加载。

接下来我们把前面的配置内容写入到一个简单的JSON配置文件中，然后同样使用`curl`命令完成加载。文件名称为*caddy.json*，其内容如下：

[1]Heredoc：向交互式命令传递多行文本或代码的一种方式，第一行是以可选命令开始，紧接着是重定向符号 <<, 你可以使用任何字符串作为分隔符，我们最常用的是 EOF(End Of File) 或者 END

```
{
    "apps": {
        "http": {
            "servers": {
                "hello": {
                    "listen": [":2015"],
                    "routes": [
                        {
                            "handle": [{
                                "handler": "static_response",
                                "body": "Hello, world!"
                            }]
                        }
                    ]
                }
            }
        }
    }
}
```

现在使用如下的命令加载这个配置文件：

```
curl localhost:2019/load \
 -X POST \
 -H "Content-Type: application/json" \
 -d @caddy.json
```

同样，我们可以在客户端通过浏览器或curl来访问一下2015端口：

```
curl http//192.168.31.2:2015
```

如果能看到Hello,world!那就代表刚才的配置文件已经生效了。

提示

实际上caddyAPI在大多数情况下，是为其他程序所设计的一种接口。

3.3. 配置多站点 (JSON & API)

接下来我们更新一下*caddy.json*文件，定义两个站点，一个在2015端口，一个在2016端口：

```
{
    "apps": {
        "http": {
            "servers": {
                "hello": {
                    "listen": [":2015"],
                    "routes": [
                        {
                            "handle": [{
                                "handler": "static_response",
                                "body": "Hello, world!"
                            }]
                        }
                    ]
                },
                "bye": {
                    "listen": [":2016"],
                    "routes": [
                        {
                            "handle": [{
                                "handler": "static_response",
```

```
                                "body": "Goodbye, world!"
                            }]
                        }
                    ]
                }
            }
        }
    }
}
```

然后再次使用如下的命令来加载这个文件：

```
curl localhost:2019/load \
 -X POST \
 -H "Content-Type: application/json" \
 -d @caddy.json
```

然后在客户端分别访问：*http://192.168.31.2:2015*和*http://192.168.31.2:2016*，会看到“Hello, world!”及“Goodbye, world!”。

提示

*caddy.json*这个文件也可以通过命令行来使用：`caddy start --config caddy.json`

使用API可以做很多的事情，包括导出配置和对配置进行细粒度的更改（而不是更新整个内容）。建议阅读更完整的API教程：https://caddyserver.com/docs/api-tutorial及https://caddyserver.com/docs/api

4. 配置测试和adapter

4.1. 对配置进行测试（Test config）

在caddy启动以后，我们随时都可以在caddy server上使用`curl`命令通过GET请求来查看配置：

```
curl localhost:2019/config/
```

这个命令直至输出的结果是JSON格式的，非常不易读，可以使用`jq`命令来将其格式化：

```
curl localhost:2019/config/ | jq
```

可以看出，输出的结果和我们的配置是一致的。可以使用这个命令将当前的配置导出。

4.2. adapter

我们会看到一个JSON输出，caddy使用caddyfile adapter（config adapter），将*Caddyfile*转换为JSON（Caddy's native JSON structure）格式。

在使用`caddy run`或`caddy start`启动caddy以后，可以在另外一个终端中运行下面的命令：

```
curl localhost:2019/config/
```

通过这个命令，可以查看当前的配置。

5. JSON vs. Caddyfile

现在我们已经知道Caddyfile会被转换为JSON格式。实际上Caddyfile的配置方式对人来说更友好、更简单一些。具体是使用JSON还是Caddyfile，要取决于具体的场景。caddy的官方网站有一张表格，可以供参考：

表 2.1. JSON vs. Caddyfile

JSON	Caddyfile
Full range of Caddy functionality	Most common parts of Caddy functionality
Easy to generate	Easy to craft by hand
Easily programmable	Difficult to automate
Extremely expressive	Moderately expressive
Allows config traversal	Cannot traverse within Caddyfile
Partial config changes	Whole config changes only
Can be exported	Cannot be exported
Compatible with all API endpoints	Compatible with some API endpoints
Documentation generated automatically	Documentation is hand-written
Ubiquitous	Niche
More efficient	More computational
Kind of boring	Kind of fun

其实，从上面的表格可以看出，如果比较精通JSON配置的话，似乎使用JSON还是更好的。

6. API vs. Config files

另外一点，就是在caddy配置管理中是使用API还是使用它的Caddyfile。其实可以同时使用，但他们官方并不推荐，他们推荐最好只使用一种。一种比较常见的组合是：JSON+API 或者 Caddyfile+CLI 。下面的表格可以让我们理解API和Caddyfile的一些区别：

表 2.2. API vs. config files

JSON	Caddyfile
Make config changes with HTTP requests	Make config changes with shell commands
Easy to scale	Difficult to scale
Difficult to manage by hand	Easy to manage by hand
Really fun	Also fun

7. 静态文件服务器

启用静态的文件服务器(file server)有两种简单的方式：一种是通过命令行，一种是通过Caddyfile。

7.1. command line

使用命令启动文件服务器，可以在终端中切换到站点所在的目录，例如：*/var/www/html*，然后在这个目录下执行：`caddy  file-server`，这样一个基本的静态文件服务器就在80端口开始运行了。如果在*/var/www/html*目录下有个*index.html*文件的话，在客户端的浏览器中输入：*http://192.168.31.2/*就可以直接看到这个文件了。

如果想要caddy运行在其他的端口，可以使用`caddy file-server --listen :2015`这个命令。

如果目录中没有*index.html*文件，并且想能够列出*/var/www/html*中的文件，可以使用`caddy  file-server --browse`这个命令。

如果想要使用指定的目录作为根目录，可以使用`caddy file-server --root /path/to/yourSite`这个命令。

7.2. Caddyfile

使用Caddyfile配置文件的方式实现文件服务器也非常简单，如果只是简单的启动它，下面的两行配置就够了：

```
localhost
file_server
```

如果想要变换端口，只需要将`localhost`更改为`localhost:2015`就可以了。

如果想要目录下的文件可列表浏览，可以将`file_server`替换为`file_server browse`

如果想要从客户端的浏览器中可以访问，我们需要下面的配置：

```
192.168.31.2
root * /var/www/
file_server browse
```

8. 反向代理

为了说明反向代理，我们需要在Server1上安装一个web服务器。ip地址设置为192.168.31.10/24。

8.1. command line

使用如下的命令：

```
caddy reverse-proxy --from :2016 --to 192.168.31.10:80
```

当我们在客户端的浏览器中输入*http://192.168.31.2:2016*时，就会看到Server1站点的内容。

8.2. Caddyfile

更改*Caddyfile*的内容，如下：

```
:2016
```

```
reverse_proxy 192.168.31.10:9000
```

然后使用caddy run命令来启动caddy，再次在客户端的浏览器中输入*http://192.168.31.2:2016*时，就会看到Server1站点的内容。

9. HTTPS

HTTPS对于caddy来说是一件非常容易的事情，当caddy安装好以后默认就带HTTPS了，只要有一个域名，就可以启用HTTPS了。我们使用lognovel.com这个域名来说明如何配置HTTPS。

9.1. Caddyfile

建立一个根目录*/usr/local/work/lognovel.com*，在这个目录里建立一个*index.html*文件，*Caddyfile*配置如下：

```
lognovel.com www.lognovel.com {
 root * /usr/local/work/lognovel.com
 file_server
}
```

在浏览器中输入*http://lognovel.com*或*http://www.lognovel.com*，会看到*index.html*里的内容。我们可以通过浏览器地址栏左的小锁icon来查看https证书的情况。

图 2.2. caddy https

9.2. 使用命令

使用下面的命令也可以达到上面配置文件相同的效果：

```
caddy file-server --domain lognovel.com --root /usr/local/work/lognovel.com
```