如何使用 HTTP 代理?

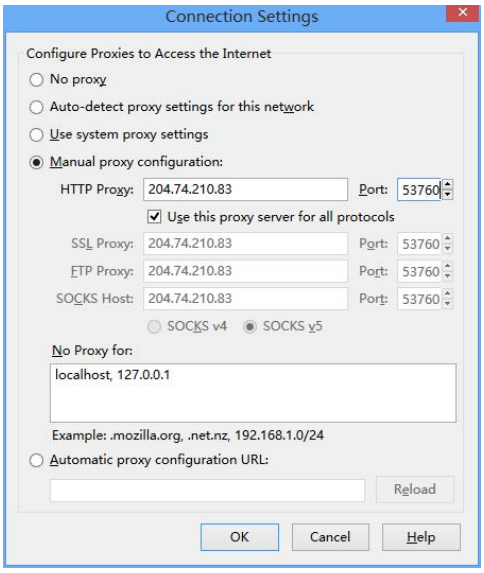
IPRENT.CN

标准的 HTTP 代理格式是这样的”用户名:密码@IP:端口”，其中的用户名和密码是用来进行代理权限认证的，认证协议一般采用标准的 HTTP Basic Authentication。网上大部分公开的免费代理是不需要认证的，意味着任何人都可以使用，稳定性和速度都没保障，俗称“野代理”，常见的野代理格式为”IP:端口”。

一、 在浏览器中使用代理

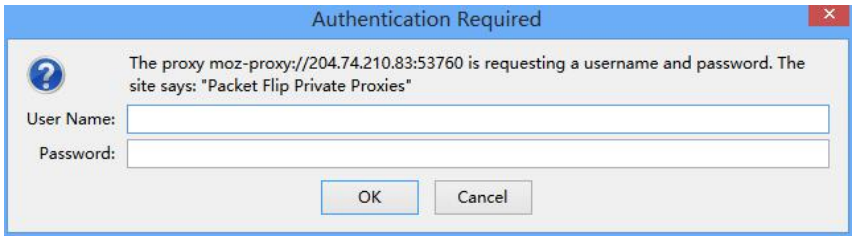
1. 火狐

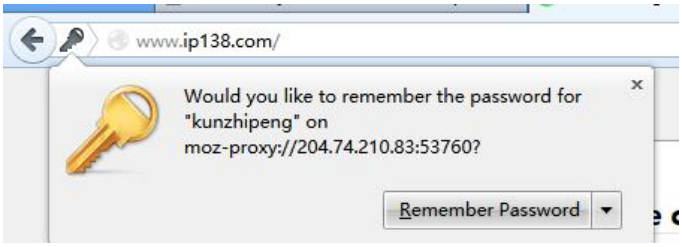
依次点击打开 “菜单 -> 选项(Options) -> 高级（Advanced） -> 网络（Network） -> 设置 （Settings）”，如下图所示：



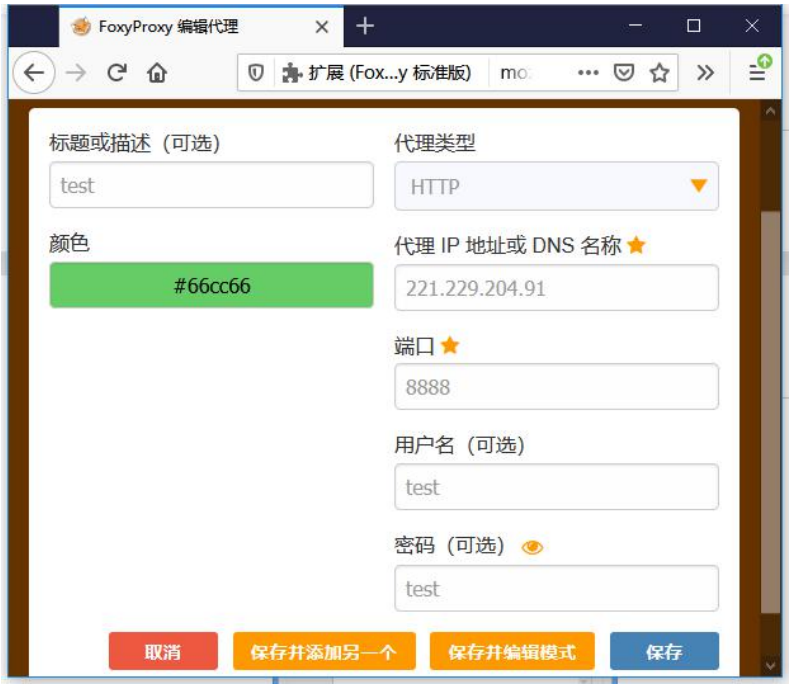
输入代理 IP 和端口，勾选“为所有协议使用相同代理”，确定。

然后打开任意网站，会出现“Authentication Required”窗口（如下图示），输入代理的用户名和密码，确定即可。可点击“记住密码”按钮，这样下次就不用再次重复输入用户名和密码。



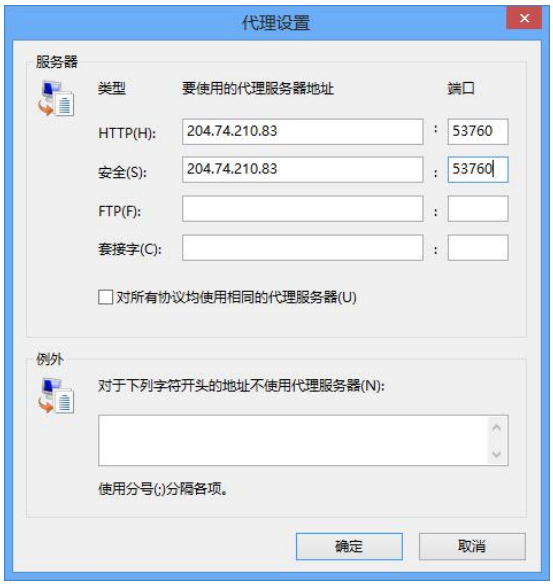


另外，强烈推荐通过安装火狐代理插件来管理代理，例如”FoxyProxy Standard”。插件能够支持设置用户名密码（如下图所示），这样后面使用的时候就不会再弹出代理权限认证窗口了，而且插件能够支持同时添加多个代理，需要的时候一键切换。



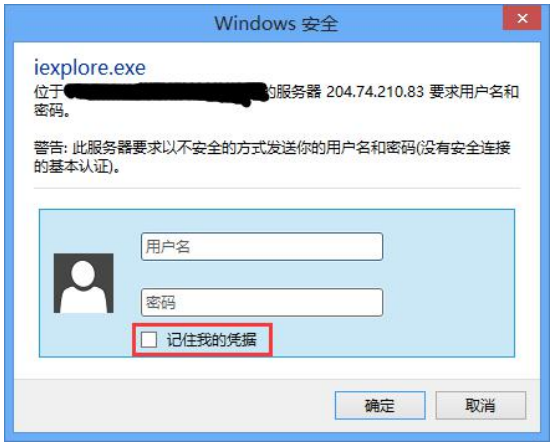
2. IE/360

依次点击打开 “菜单 -> 工具 -> Internet 选项 -> 连接 -> 局域网设置”，勾选“为 LAN 使用代理服务器”，点击高级按钮。如下图所示：



输入 HTTP 代理 IP 和端口，如果你的代理也支持 HTTPS，在”安全（S）”里也输入相同的 IP 和端口（这一步很重要），确定。

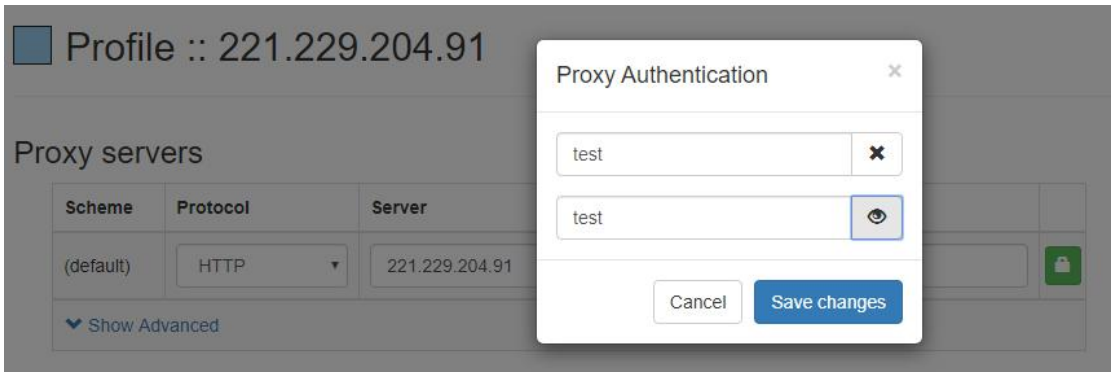
然后打开任意网站，会出现“Windows 安全”窗口（如下图示），输入代理的用户名和密码，勾选“记住我的凭据”确定即可。



3. Chrome

谷歌默认使用系统代理，设置系统代理的方法和 IE 一样。

强烈推荐通过安装 Chrome 代理插件来管理代理，例如"SwitchyOmega"，最新版的 SwitchyOmega 插件已经支持设置保存代理认证的用户名和密码，如下图所示。



二、 在自己编写的程序中使用代理

1. Python 语言

参见如下示例代码：

示例一：使用 requests

```
import requests

proxies = {
  'http': 'http://username:password@ip:port',
  'https': 'http://username:password@ip:port',
}

print requests.get('http://httpbin.org/ip', proxies=proxies).content
```

示例二：使用 urllib2

```
import urllib2

# 你的HTTP代理
proxy = 'username:password@ip:port'
# 你要访问的网址
url = 'http://httpbin.org/ip'
```

```
opener = urllib2.build_opener()
if url.lower().startswith('https://'):
    opener.add_handler(urllib2.ProxyHandler({'https' : proxy}))
else:
    opener.add_handler(urllib2.ProxyHandler({'http' : proxy}))

request = urllib2.Request(url)
response = opener.open(request)
content = response.read()
print content
```

示例三：使用 Selenium

1) + Phantomjs

```
from selenium import webdriver

service_args = ['--proxy-auth=username:password', '--proxy=ip:port']
driver = webdriver.PhantomJS(service_args=service_args)
driver.get('http://httpbin.org/ip')
print driver.page_source
```

2) + Chrome

```
from selenium import webdriver
options = webdriver.ChromeOptions()
options.add_argument('--proxy-server="http://ip:port"')
driver = webdriver.Chrome(chrome_options=options)
driver.get('http://httpbin.org/ip')
print driver.page_source
```

PS：“Selenium + Chrome”也支持用户名和密码认证，我们在这里介绍了解决方案：
http://www.site-digger.com/html/articles/20160803/129.html

3) + FireFox

```
from selenium import webdriver
fp = webdriver.FirefoxProfile()
fp.set_preference('network.proxy.type', 1)
fp.set_preference('network.proxy.http', 'ip')
fp.set_preference('network.proxy.http_port', int('port'))
driver = webdriver.Firefox(firefox_profile=fp)
driver.get('http://httpbin.org/ip')
print driver.page_source
```

PS：“Selenium + Firefox”也支持用户名和密码认证，我们在这里介绍了两种解决方案：
(1) http://www.site-digger.com/html/articles/20180822/662.html;
(2) http://www.site-digger.com/html/articles/20190115/695.html;

示例四：使用 Scrapy

```
import base64
```

```
# 设置代理
request.meta['proxy'] = "http://ip:port"
# 添加认证头
user_pass = base64.encodestring('username:password')
request.headers['Proxy-Authorization'] = 'Basic ' + user_pass
```

2. PHP 语言

参见如下示例代码：

```
<?php
function curlFile($url,$proxy_ip,$proxy_port,$loginpassw)
{
    //$loginpassw = 'username:password';
    //$proxy_ip = 'ip';
    //$proxy_port = 'port';
    //$url = 'http://httpbin.org/ip';

    $ch = curl_init();
    curl_setopt($ch, CURLOPT_URL, $url);
    curl_setopt($ch, CURLOPT_HEADER, 0);
    curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
    curl_setopt($ch, CURLOPT_PROXYPORT, $proxy_port);
    curl_setopt($ch, CURLOPT_PROXYTYPE, 'HTTP');
    curl_setopt($ch, CURLOPT_PROXY, $proxy_ip);
    curl_setopt($ch, CURLOPT_PROXYUSERPWD, $loginpassw);
    $data = curl_exec($ch);
    curl_close($ch);

    return $data;
}
?>
```

3. C#语言

参见如下示例代码：

```
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(URL);
IWebProxy proxy = request.Proxy;
if (proxy != null)
{
   Console.WriteLine("Proxy: {0}",
proxy.GetProxy(request.RequestUri));
}
else
{
   Console.WriteLine("Proxy is null; no proxy will be used");
}

WebProxy myProxy = new WebProxy();
Uri newUri = new Uri("http://ip:port");
// Associate the newUri object to 'myProxy' object so that new
myProxy settings can be set.
myProxy.Address = newUri;
// Create a NetworkCredential object and associate it with the
```

```
// Proxy property of request object.
myProxy.Credentials = new NetworkCredential("user", "password");
request.Proxy = myProxy;
```

4. Java 语言

方法一:

```
HttpClient httpClient = new HttpClient();
httpClient.getHostConfiguration().setProxy("proxy ip", port);
httpClient.getParams().setAuthenticationPreemptive(true);
// 这里设置代理的用户名密码
httpClient.getState().setProxyCredentials(AuthScope.ANY, new
UsernamePasswordCredentials("username","password"));
```

方法二:

```
System.setProperty("http.proxyHost", "proxy ip");
System.setProperty("http.proxyPort", "port");

URL url = new URL("http://www.site-digger.com/");
URLConnection uc = url.openConnection();
String encoded = new String
      (Base64.base64Encode(new
String("username:password").getBytes()));
// 添加一个"Proxy-Authorization"头处理代理认证
uc.setRequestProperty("Proxy-Authorization", "Basic " + encoded);
uc.connect();
```

方法三:

```
import java.net.Authenticator;

class ProxyAuthenticator extends Authenticator {

    private String user, password;

    public ProxyAuthenticator(String user, String password) {
        this.user = user;
        this.password = password;
    }

    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(user,
password.toCharArray());
    }
}

Authenticator.setDefault(new ProxyAuthenticator("user", "password"));
System.setProperty("http.proxyHost", "proxy ip");
System.setProperty("http.proxyPort", "port");
```

5. Go 语言

```
package main

import (
    "encoding/base64"
    "fmt"
    "io/ioutil"
    "log"
    "net/http"
    "net/http/httputil"
    "net/url"
)

func main() {

    // 代理 IP 和端口
    proxyStr := "http://proxy_ip:proxy_port"
    proxyURL, err := url.Parse(proxyStr)
    if err != nil {
        log.Println(err)
    }

    // 要访问的目标 URL
    urlStr := "http://httpbin.org/ip"
    url, err := url.Parse(urlStr)
    if err != nil {
        log.Println(err)
    }

    //给 Transport 设置代理
    transport := &http.Transport{
        Proxy: http.ProxyURL(proxyURL),
    }

    // http Client
    client := &http.Client{
        Transport: transport,
    }

    // 构造 HTTP GET 请求
    request, err := http.NewRequest("GET", url.String(), nil)
    if err != nil {
        log.Println(err)
    }

    // 添加 proxy authentication 头
    auth := "user:password"
    basicAuth := "Basic " +
base64.StdEncoding.EncodeToString([]byte(auth))
    request.Header.Add("Proxy-Authorization", basicAuth)
    // 为了支持 HTTPS(Web 隧道)协议，以下两行不能少
```

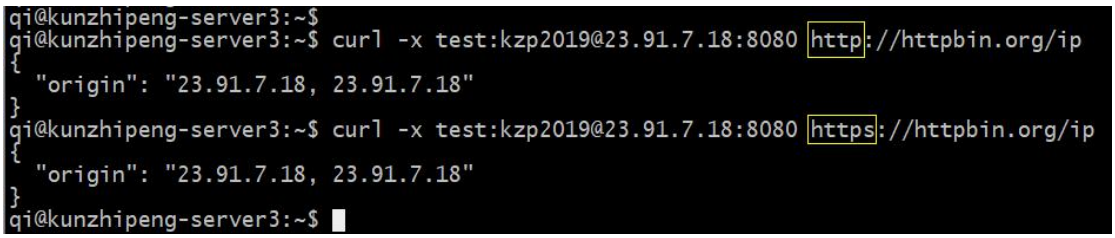
```
    transport.ProxyConnectHeader = http.Header{}
    transport.ProxyConnectHeader.Add("Proxy-Authorization",
basicAuth)

    // 打印请求数据
    dump, _ := httputil.DumpRequest(request, false)
    fmt.Println(string(dump))

    // 发出请求
    response, err := client.Do(request)
    if err != nil {
        log.Println(err)
    }

    log.Println(response.StatusCode)
    log.Println(response.Status)
    // 获取应答数据
    data, err := ioutil.ReadAll(response.Body)
    if err != nil {
        log.Println(err)
    }
    // 打印应答数据
    log.Println(string(data))
}
```

6. curl 命令和 wget 命令

curl 命令示例：

```
curl -x username:password@ip:port  “http://httpbin.org/ip”
```

curl 命令测试如下图所示（HTTP 和 HTTPS 均支持）：



wget 命令示例：

```
wget -e "http_proxy=http://username:password@ip:port"
“http://httpbin.org/ip”
wget -e "https_proxy=http://username:password@ip:port"
“https://httpbin.org/ip”
```

Wget 命令测试如下图所示（HTTP 和 HTTPS 均支持）：

附：哪种代理适合用于 Web 数据采集?

在 Web 数据采集中为了避免被服务器封锁而通过代理下载的情况很常见。但是，并非所有的代理都适合于 Web 数据采集。下面是鲲鹏数据的技术人员给出的说明。

根据 HTTP 代理的匿名性可以将其分为以下几种：

1. 透明代理（Transparent Proxies）

目标服务器能够检测到真实的源 IP。

目标服务器根据 HTTP 请求头进行检测，判断依据：

REMOTE_ADDR = 代理服务器 IP

HTTP_VIA = 通常为代理服务器 IP（或代理软件名称，也可能无此头）

HTTP_X_FORWARDED_FOR = 真实源 IP（不用代理时，无此头或值为空）

PS：该类型代理不适合用于 Web 数据采集。

2. （普通）匿名代理（Anonymous Proxies）

目标服务器无法检测到真实的源 IP，但能够检测到使用了代理。

检测依据：

REMOTE_ADDR = 代理服务器 IP

HTTP_VIA = 通常为代理服务器 IP（或代理软件名称，也可能无此头）

HTTP_X_FORWARDED_FOR = 代理服务器 IP（知道你使用了代理，但无法得知真实源 IP）

PS：该类型代理可以用于 Web 数据采集，但有被检测到的风险。

3. 高匿名代理（High Anonymity Proxies -Elite proxies）

目标服务器无法检测到你在是使用代理。

检测依据：

REMOTE_ADDR = 代理服务器 IP

HTTP_VIA = 值为空或无此头

HTTP_X_FORWARDED_FOR = 没数值或无此头

PS：该类型的代理非常适合用户 Web 数据采集。鲲鹏数据的付费代理方案提供的全部为高匿名类型的代理。

另外，不使用代理时发出的头：

REMOTE_ADDR =真实源 IP

HTTP_VIA = 值为空或无此头

HTTP_X_FORWARDED_FOR = 没数值或无此头

不过，在检测严格的情况下，即使没有 HTTP_VIA 头和 HTTP_X_FORWARDED_FOR 头，如果存在 HTTP_PROXY_CONNECTION 头，会被认为在使用普通匿名代理。

我们提供了一个代理类型检测接口，在浏览器中访问该接口即可显示出你当前使用的代理类型（如下图）：

http://proxies.site-digger.com/proxy-detect/

鲲鹏在线 HTTP 代理测试工具：

http://proxies.site-digger.com/proxy-test/

鲲鹏数据提供多种代理方案，详情请查看这里：

http://www.site-digger.com/html/proxies.html