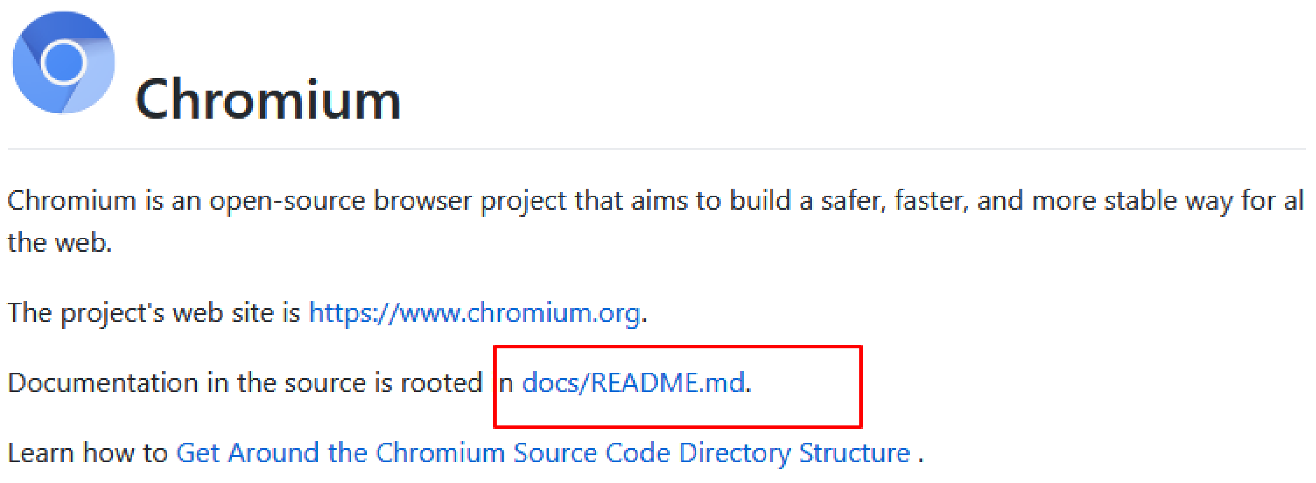
chromium编译手记

1.阅读官方文档，准备环境

在github官网搜索chromium，找到项目仓库地址如下

https://github.com/chromium/chromium.git

在docs/readme.txt中查看项目简介



Chromium

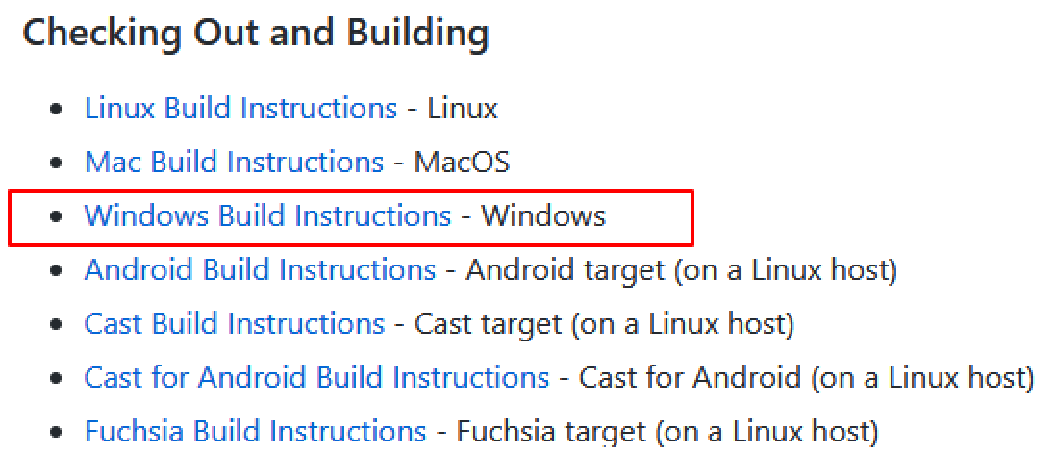
Chromium is an open-source browser project that aims to build a safer, faster, and more stable way for al the web.

The project's web site is https://www.chromium.org.

Documentation in the source is rooted n docs/README.md.

Learn how to Get Around the Chromium Source Code Directory Structure .

在readme中可以找到在各个系统构建chrome的说明



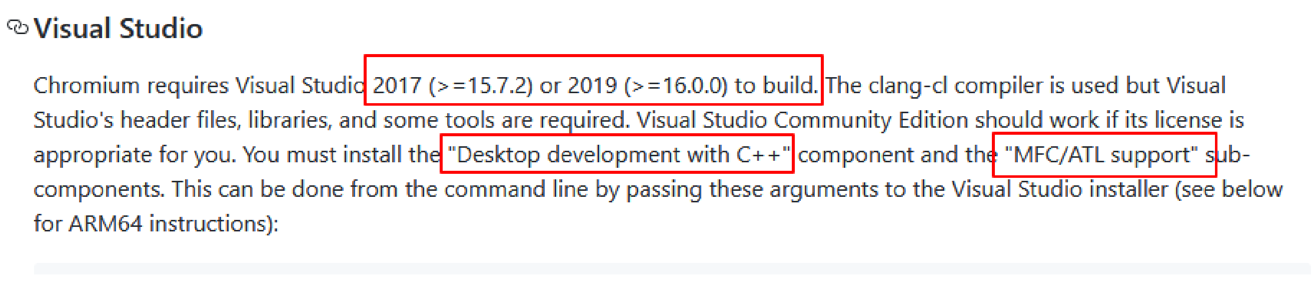
Checking Out and Building

- Linux Build Instructions - Linux
- Mac Build Instructions - MacOS
- Windows Build Instructions - Windows
- Android Build Instructions - Android target (on a Linux host)
- Cast Build Instructions - Cast target (on a Linux host)
- Cast for Android Build Instructions - Cast for Android (on a Linux host)
- Fuchsia Build Instructions - Fuchsia target (on a Linux host)

2. 配置编译环境

2.1 vsual studio

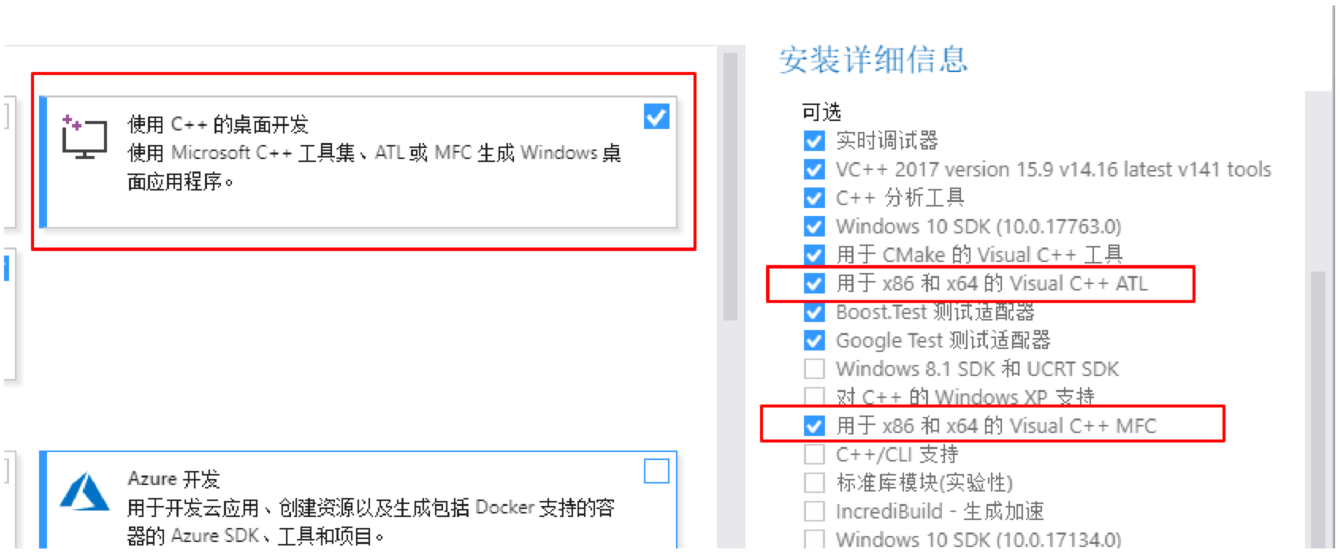
版本与库的要求如下图，如果未安装的话，通过visual studio安装程序安装即可



Visual Studio

Chromium requires Visual Studio 2017 (>=15.7.2) or 2019 (>=16.0.0) to build. The clang-cl compiler is used but Visual Studio's header files, libraries, and some tools are required. Visual Studio Community Edition should work if its license is appropriate for you. You must install the "Desktop development with C++" component and the "MFC/ATL support" sub-components. This can be done from the command line by passing these arguments to the Visual Studio installer (see below for ARM64 instructions):

如果已经安装了visual studio，在开始菜单中可以找到visual studio installer，打开之后点击修改，随后应用即可安装需要的工具



使用 C++ 的桌面开发 ✓
使用 Microsoft C++ 工具集、ATL 或 MFC 生成 Windows 桌面应用程序。

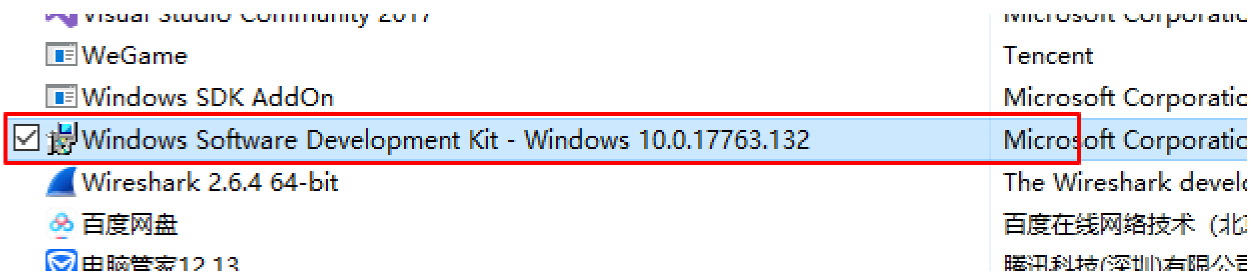
Azure 开发 ☐
用于开发云应用、创建资源以及生成包括 Docker 支持的容器的 Azure SDK、工具和项目。

安装详细信息

可选
- ✓ 实时调试器
- ✓ VC++ 2017 version 15.9 v14.16 latest v141 tools
- ✓ C++ 分析工具
- ✓ Windows 10 SDK (10.0.17763.0)
- ✓ 用于 CMake 的 Visual C++ 工具
- ✓ 用于 x86 和 x64 的 Visual C++ ATL
- ✓ Boost.Test 测试适配器
- ✓ Google Test 测试适配器
- ☐ Windows 8.1 SDK 和 UCRT SDK
- ☐ 对 C++ 的 Windows XP 支持
- ✓ 用于 x86 和 x64 的 Visual C++ MFC
- ☐ C++/CLI 支持
- ☐ 标准库模块(实验性)
- ☐ IncrediBuild - 生成加速
- ☐ Windows 10 SDK (10.0.17134.0)

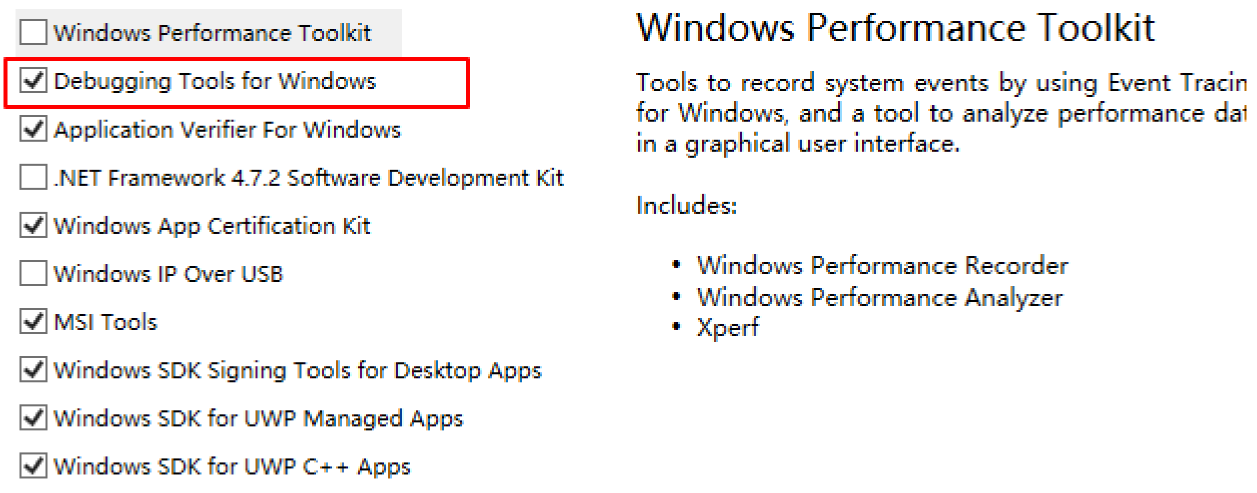
还需要安装sdk debugging tools，win10sdk已经通过visual studio安装以后，可以通过如下途径安装sdk debugging tools

控制面板->software->program and features->Windows Software Development Kit->



Visual Studio Community 2017	Microsoft Corporatio
WeGame	Tencent
Windows SDK AddOn	Microsoft Corporatic
Windows Software Development Kit - Windows 10.0.17763.132	Microsoft Corporatic
Wireshark 2.6.4 64-bit	The Wireshark devel
百度网盘	百度在线网络技术（北
电脑管家12.13	腾讯科技(深圳)有限公司

right click->change->change->next->



- [ ] Windows Performance Toolkit
- [x] Debugging Tools for Windows
- [x] Application Verifier For Windows
- [ ] .NET Framework 4.7.2 Software Development Kit
- [x] Windows App Certification Kit
- [ ] Windows IP Over USB
- [x] MSI Tools
- [x] Windows SDK Signing Tools for Desktop Apps
- [x] Windows SDK for UWP Managed Apps
- [x] Windows SDK for UWP C++ Apps

Windows Performance Toolkit

Tools to record system events by using Event Tracin for Windows, and a tool to analyze performance dat in a graphical user interface.

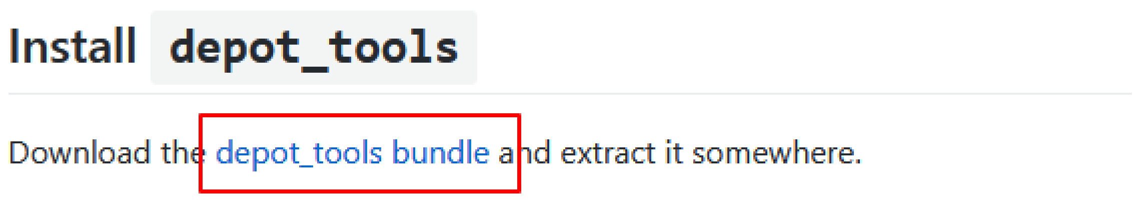
Includes:

- Windows Performance Recorder
- Windows Performance Analyzer
- Xperf

change，勾选debugging tools for windows，安装即可

2.2 安装dept_tools

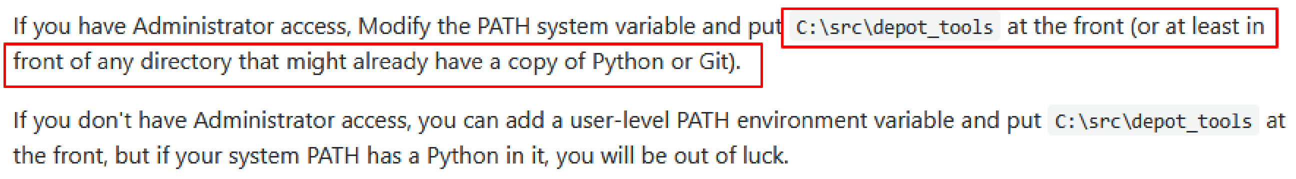
根据文档给出的链接，直接下载dept_tools bundle



Install `depot_tools`

Download the depot_tools bundle and extract it somewhere.

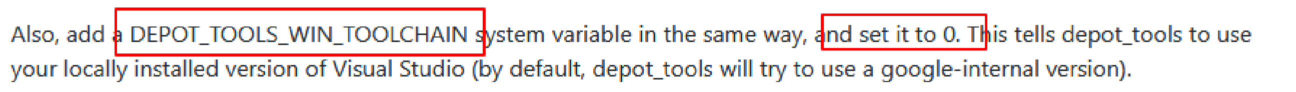
下载完成后，解压至某目录，此处解压至（d:\dev_envir\dept_tools)，随后将该目录添加到系统path环境变量的开头，如下图中的说明，dept_tools中包含有python与git工具，在后续使用过程中，要使用dept_tools目录中的python与git命令，所以需要将dept_tools目录添加进path变量的开头，屏蔽系统本来的python与git



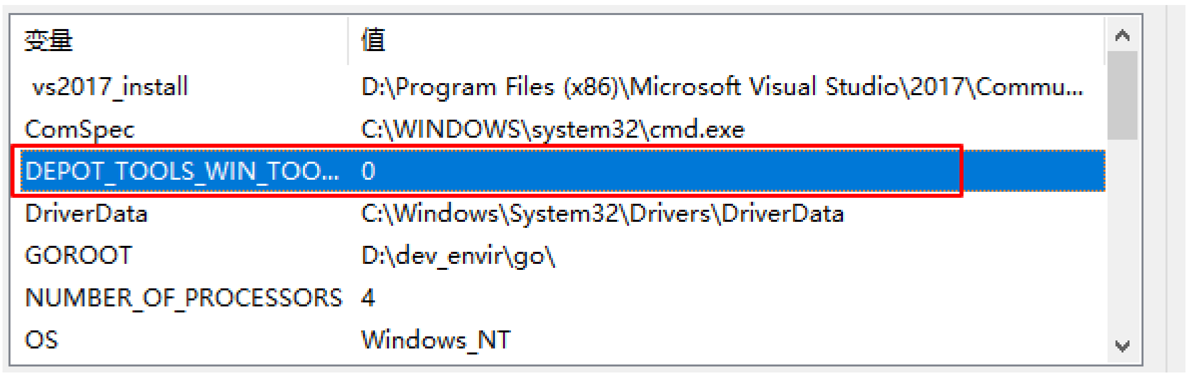
If you have Administrator access, Modify the PATH system variable and put `C:\src\depot_tools` at the front (or at least in front of any directory that might already have a copy of Python or Git).

If you don't have Administrator access, you can add a user-level PATH environment variable and put `C:\src\depot_tools` at the front, but if your system PATH has a Python in it, you will be out of luck.

添加DEPOT_TOOLS_WIN_TOOLCHAIN



Also, add a DEPOT_TOOLS_WIN_TOOLCHAIN system variable in the same way, and set it to 0. This tells depot_tools to use your locally installed version of Visual Studio (by default, depot_tools will try to use a google-internal version).



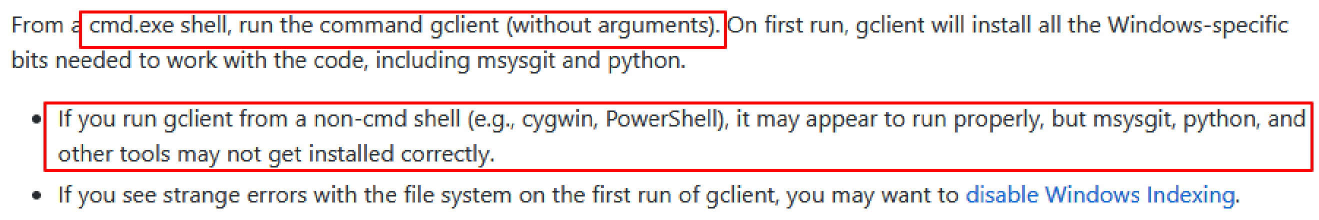
变量	值
vs2017_install	D:\Program Files (x86)\Microsoft Visual Studio\2017\Commu...
ComSpec	C:\WINDOWS\system32\cmd.exe
DEPOT_TOOLS_WIN_TOO...	0
DriverData	C:\Windows\System32\Drivers\DriverData
GOROOT	D:\dev_envir\go\
NUMBER_OF_PROCESSORS	4
OS	Windows_NT

设置GYP_MSVS_VERSION与GYP_MSVS_OVERRIDE_PATH，前者指明visual studio版本，后者指明visual studio安装目录(可以添加进环境变量中，也可以在后续运行gclient命令时执行下面两条指令设置visual studio环境)

```
set GYP_MSVS_VERSION=2017
set GYP_MSVS_OVERRIDE_PATH=D:\Program Files (x86)\Microsoft Visual Studio\2017\Community
```

2.3 获取gclient

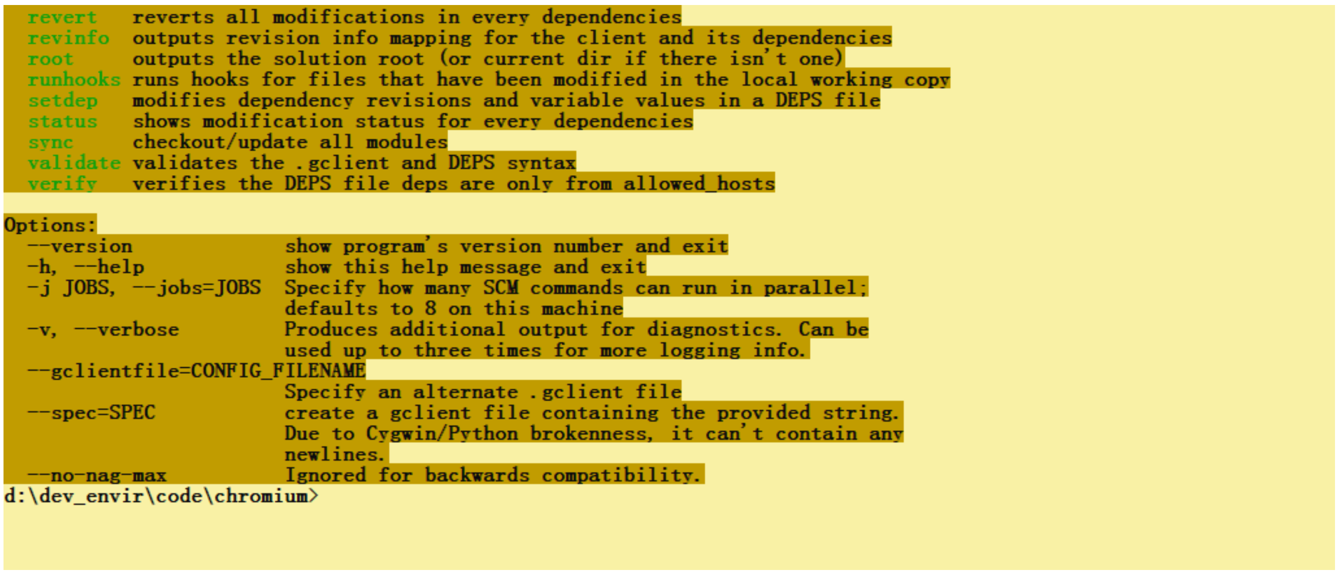
随后在cmd窗口中运行gclient命令，根据文档说明，不能在cmd以外的环境比如powershell中运行，否则会达不到预期安装要求



From a cmd.exe shell, run the command gclient (without arguments). On first run, gclient will install all the Windows-specific bits needed to work with the code, including msysgit and python.

- If you run gclient from a non-cmd shell (e.g., cygwin, PowerShell), it may appear to run properly, but msysgit, python, and other tools may not get installed correctly.
- If you see strange errors with the file system on the first run of gclient, you may want to disable Windows Indexing.

该命令执行完成后，会显示gclient的usage信息，表示gclient安装成功



```
  revert    reverts all modifications in every dependencies
  revinfo   outputs revision info mapping for the client and its dependencies
  root      outputs the solution root (or current dir if there isn't one)
  runhooks  runs hooks for files that have been modified in the local working copy
  setdep    modifies dependency revisions and variable values in a DEPS file
  status    shows modification status for every dependencies
  sync      checkout/update all modules
  validate  validates the .gclient and DEPS syntax
  verify    verifies the DEPS file deps are only from allowed_hosts

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -j JOBS, --jobs=JOBS  Specify how many SCM commands can run in parallel;
                        defaults to 8 on this machine
  -v, --verbose         Produces additional output for diagnostics. Can be
                        used up to three times for more logging info.
  --gclientfile=CONFIG_FILENAME
                        Specify an alternate .gclient file
  --spec=SPEC           create a gclient file containing the provided string.
                        Due to Cygwin/Python brokenness, it can't contain any
                        newlines.
  --no-nag-max          Ignored for backwards compatibility.
d:\dev_envir\code\chromium>
```

3. 获取chromium源码

3.1 配置git

在cmd窗口（任意目录下）配置git参数（仅仅是获取源码的话，git账户信息应该不是必须的，理论上可以跳过，但是后面三个选项还是要设置的）

```
$ git config --global user.name "My Name"
$ git config --global user.email "my-name@chromium.org"
$ git config --global core.autocrlf false
$ git config --global core.filemode false
$ git config --global branch.autosetuprebase always
```

user.name与user.email分别对应用户名与邮箱地址

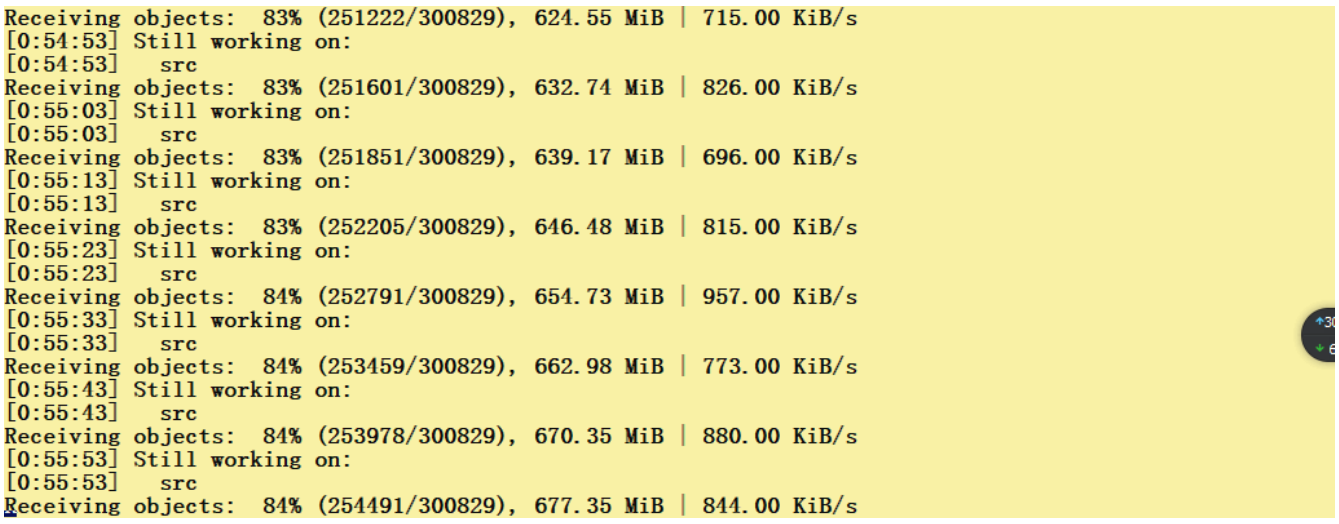
3.2 获取chromium源码

创建chromium文件夹，切换到该文件夹中，尝试获取chromium源码，本次chromium文件夹位于

d:\dev_envir\code\chromium切换到该文件夹下使用fetch chromium命令获取chromium源码，chromium源码非常巨大，可以在fetch命令后加上--no-history跳过历史版本，可以节约一点时间

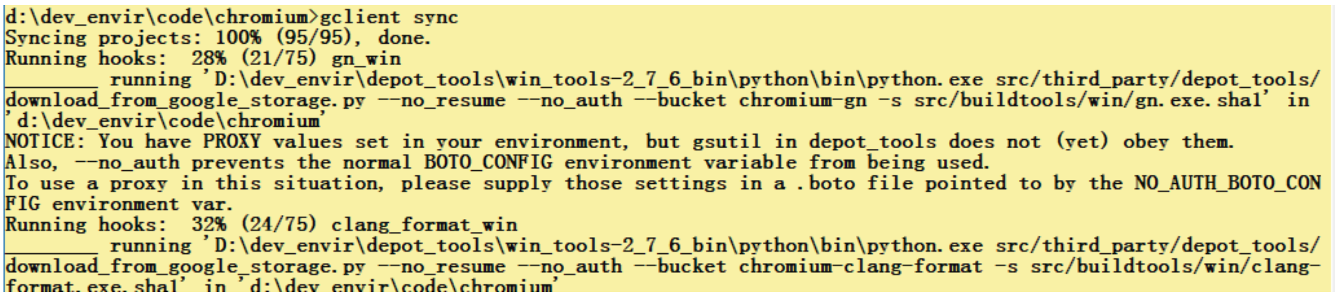
```
fetch chromium --no-history
```

耗时较长，如果中间断了，可以使用gclient sync同步



```
Receiving objects:  83% (251222/300829), 624.55 MiB | 715.00 KiB/s
[0:54:53] Still working on:
[0:54:53]   src
Receiving objects:  83% (251601/300829), 632.74 MiB | 826.00 KiB/s
[0:55:03] Still working on:
[0:55:03]   src
Receiving objects:  83% (251851/300829), 639.17 MiB | 696.00 KiB/s
[0:55:13] Still working on:
[0:55:13]   src
Receiving objects:  83% (252205/300829), 646.48 MiB | 815.00 KiB/s
[0:55:23] Still working on:
[0:55:23]   src
Receiving objects:  84% (252791/300829), 654.73 MiB | 957.00 KiB/s
[0:55:33] Still working on:
[0:55:33]   src
Receiving objects:  84% (253459/300829), 662.98 MiB | 773.00 KiB/s
[0:55:43] Still working on:
[0:55:43]   src
Receiving objects:  84% (253978/300829), 670.35 MiB | 880.00 KiB/s
[0:55:53] Still working on:
[0:55:53]   src
Receiving objects:  84% (254491/300829), 677.35 MiB | 844.00 KiB/s
```

在执行gclient sync的过程中，发现执行到gclient runhooks经常由于网络原因无法进行，此时单步执行gclient runhooks，成功后再执行glicent sync，执行完之后源码就算是获取完成（获取源码完成后，其目录结构与内容与github上的chromium镜像仓库内容相同，可以直接下载github源码镜像，解压至chromium目录后，在chromium目录下执行gclient sync步骤，同步google代码仓库）



```
d:\dev_envir\code\chromium>gclient sync
Syncing projects: 100% (95/95), done.
Running hooks:  28% (21/75) gn_win
________ running 'D:\dev_envir\depot_tools\win_tools-2_7_6_bin\python\bin\python.exe src/third_party/depot_tools/
download_from_google_storage.py --no_resume --no_auth --bucket chromium-gn -s src/buildtools/win/gn.exe.sha1' in
'd:\dev_envir\code\chromium'
NOTICE: You have PROXY values set in your environment, but gsutil in depot_tools does not (yet) obey them.
Also, --no_auth prevents the normal BOTO_CONFIG environment variable from being used.
To use a proxy in this situation, please supply those settings in a .boto file pointed to by the NO_AUTH_BOTO_CON
FIG environment var.
Running hooks:  32% (24/75) clang_format_win
________ running 'D:\dev_envir\depot_tools\win_tools-2_7_6_bin\python\bin\python.exe src/third_party/depot_tools/
download_from_google_storage.py --no_resume --no_auth --bucket chromium-clang-format -s src/buildtools/win/clang-
format.exe.sha1' in 'd:\dev_envir\code\chromium'
```

在chromium目录下可以看到有src文件夹，后续步骤进入该文件夹中执行

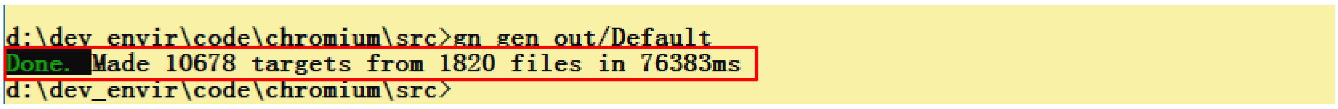
```
cd src
```

4. 使用gn编译chromium

使用gn命令编译chrome之前，最好查阅步骤六，设置google api key，这样编译出的chrome程序不会有缺少google api key的提示

4.1 使用gn生成.ninja 文件

```
gn gen out/Default
```

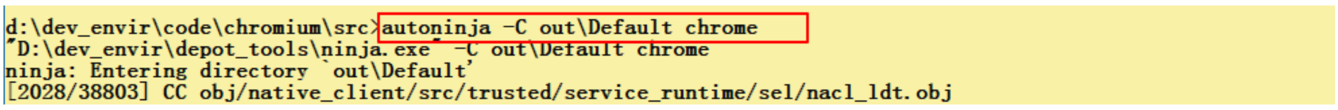


```
d:\dev_envir\code\chromium\src>gn gen out/Default
Done. Made 10678 targets from 1820 files in 76383ms
d:\dev_envir\code\chromium\src>
```

4.2 使用autoninja编译chromium

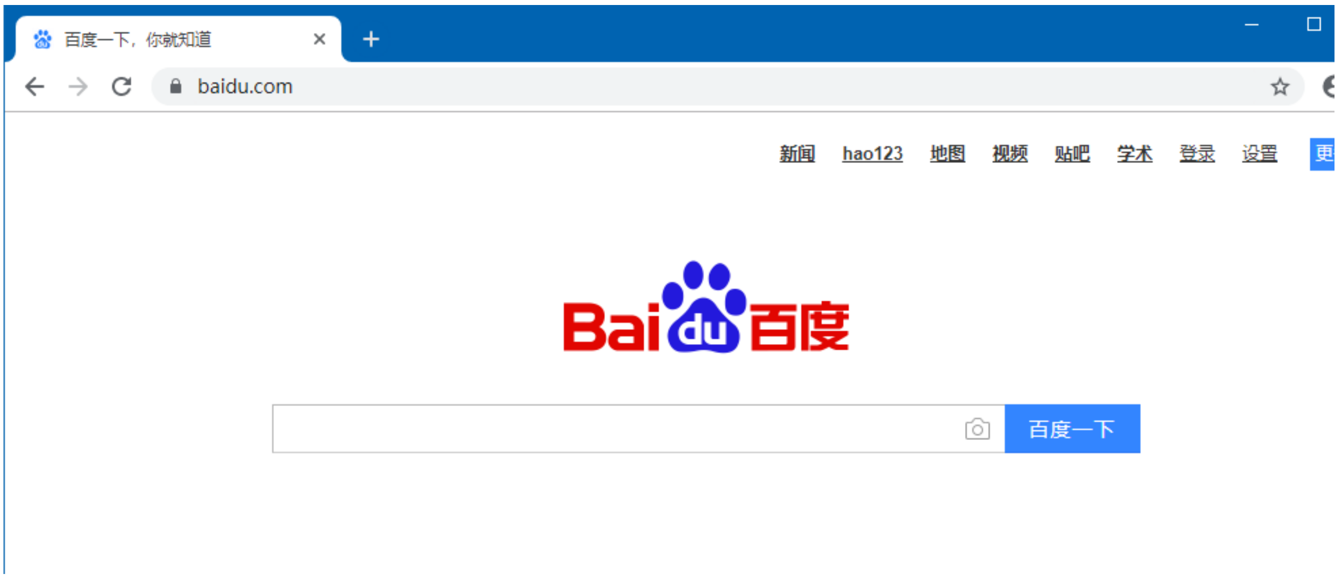
```
autoninja -C out\Default chrome # 非常耗时
```

等待编译完成



```
d:\dev_envir\code\chromium\src>autoninja -C out\Default chrome
"D:\dev_envir\depot_tools\ninja.exe" -C out\Default chrome
ninja: Entering directory `out\Default'
[2028/38803] CC obj/native_client/src/trusted/service_runtime/sel/nacl_ldt.obj
```

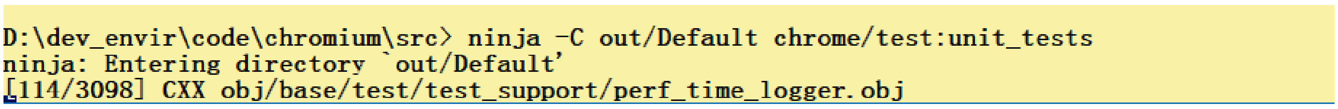
编译完成后，会在out\Default\目录下生成chrome.exe可执行文件，打开即可看到chromium窗口



4.3 生成并运行单元测试

生成单元测试使用如下命令(run with cmd in folder src),编译过程较长，需耐心等待

```
ninja -C out/Default chrome/test:unit_tests
```



```
D:\dev_envir\code\chromium\src> ninja -C out/Default chrome/test:unit_tests
ninja: Entering directory `out/Default'
[114/3098] CXX obj/base/test/test_support/perf_time_logger.obj
```

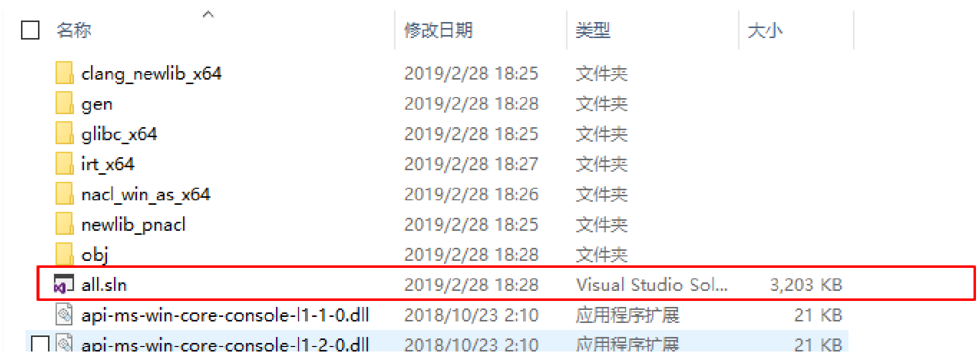
编译完成后在out/Default/下生成unit_tests.exe，运行该测试，检验编译出现的问题

5. 生成visual studio解决方案

chromium源码十分庞大，生成的解决方案中包含很多子项目，导致性能不太好的计算机（我的笔记本）打开非常耗时，电脑性能好的同学可以尝试

```
$ gn gen --ide=vs out\vstudio
$ devenv out\Default\all.sln    # 使用vstudio命令行打开该解决方案，先打开visual studio，再在vs ide 中打开该项目亦可，所以这条命令可以不执行
```

gn命令执行完毕后，在out\studio目录下可已找到all.sln，使用visual studio打开即可，在visual studio中build all



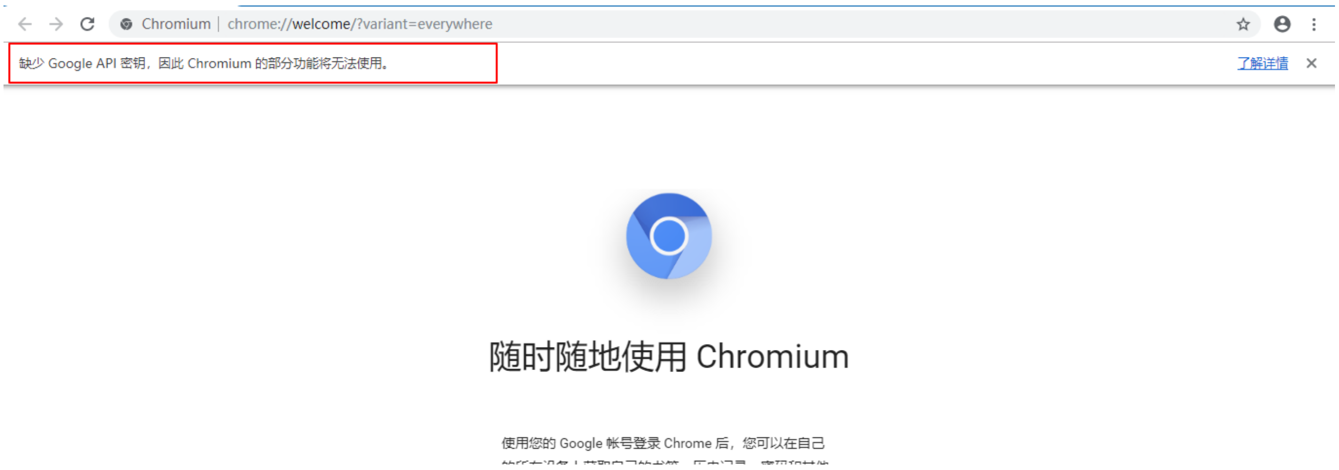
名称	修改日期	类型	大小
clang_newlib_x64	2019/2/28 18:25	文件夹	
gen	2019/2/28 18:28	文件夹	
glibc_x64	2019/2/28 18:25	文件夹	
irt_x64	2019/2/28 18:27	文件夹	
nacl_win_as_x64	2019/2/28 18:26	文件夹	
newlib_pnacl	2019/2/28 18:25	文件夹	
obj	2019/2/28 18:28	文件夹	
all.sln	2019/2/28 18:28	Visual Studio Sol...	3,203 KB
api-ms-win-core-console-l1-1-0.dll	2018/10/23 2:10	应用程序扩展	21 KB
api-ms-win-core-console-l1-2-0.dll	2018/10/23 2:10	应用程序扩展	21 KB

6. 配置google api key

http://www.chromium.org/developers/how-tos/api-keys

6.1 申请google api key

随上述步骤编译出程序后，打开浏览器会看到一条警告信息



Chromium | chrome://welcome/?variant=everywhere

缺少 Google API 密钥，因此 Chromium 的部分功能将无法使用。

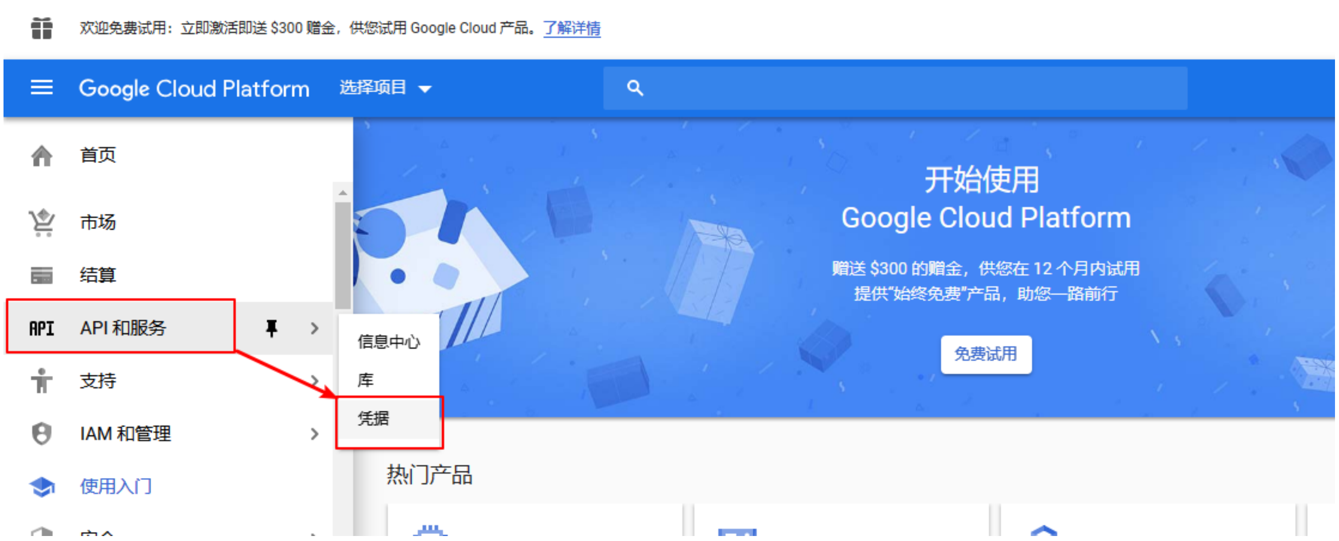
了解详情

随时随地使用 Chromium

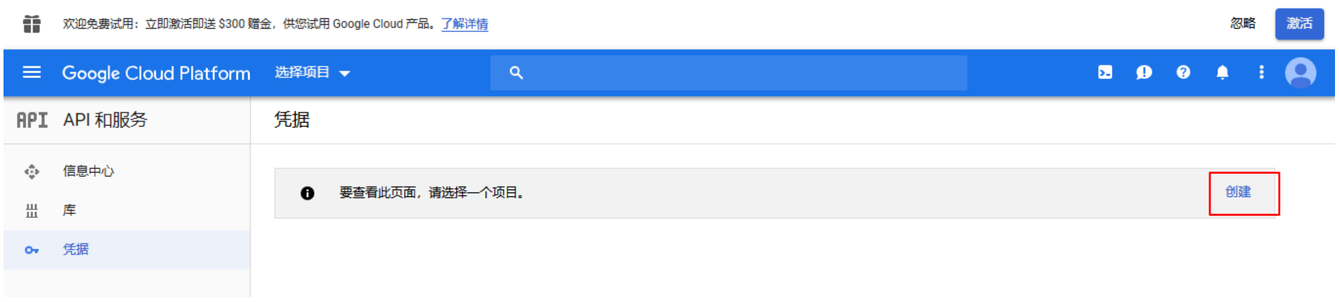
使用您的 Google 帐号登录 Chrome 后，您可以在自己

由于缺少google api key，所以有关google账户的功能都不可以使用，下面尝试申请google api key

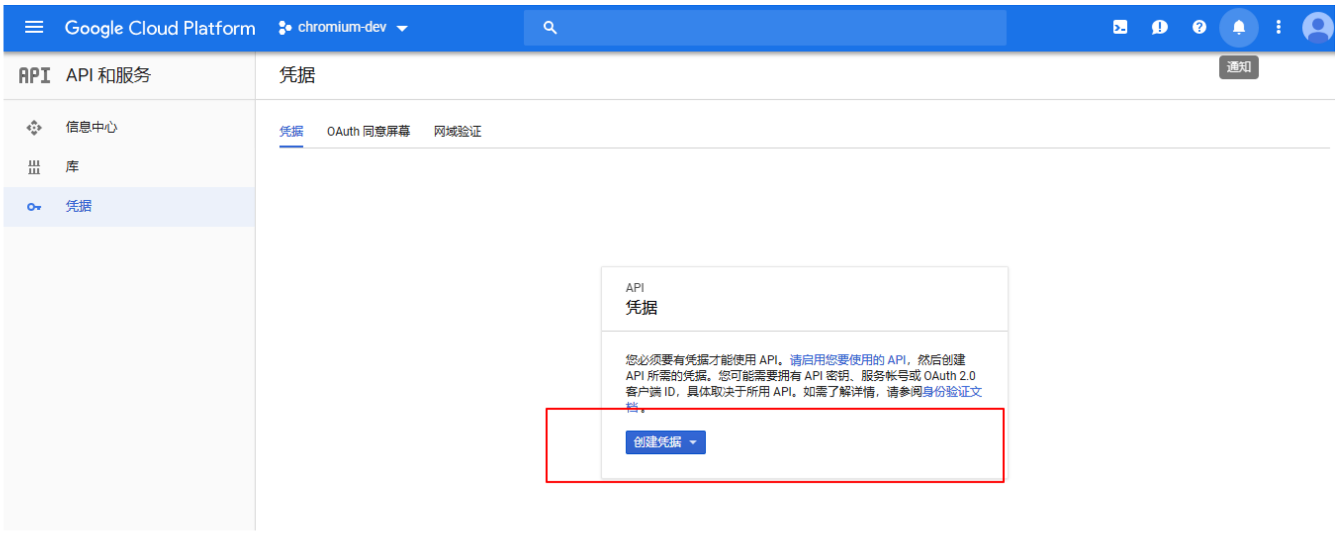
访问https://console.cloud.google.com，API和服务->凭据


欢迎免费试用：立即激活即送 $300 赠金，供您试用 Google Cloud 产品。了解详情
Google Cloud Platform
选择项目
首页
市场
结算
API 和服务
支持
IAM 和管理
使用入门
信息中心
库
凭据
开始使用
Google Cloud Platform
赠送 $300 的赠金，供您在 12 个月内试用
提供"始终免费"产品，助您一路前行
免费试用
热门产品

在页面中点击创建项目


欢迎免费试用：立即激活即送 $300 赠金，供您试用 Google Cloud 产品。了解详情
忽略
激活
Google Cloud Platform
选择项目
API 和服务
凭据
信息中心
库
凭据
要查看此页面，请选择一个项目。
创建

在创建页面中，名称填chromium-dev（随便填一个）其余默认，创建成功后点击创建凭据

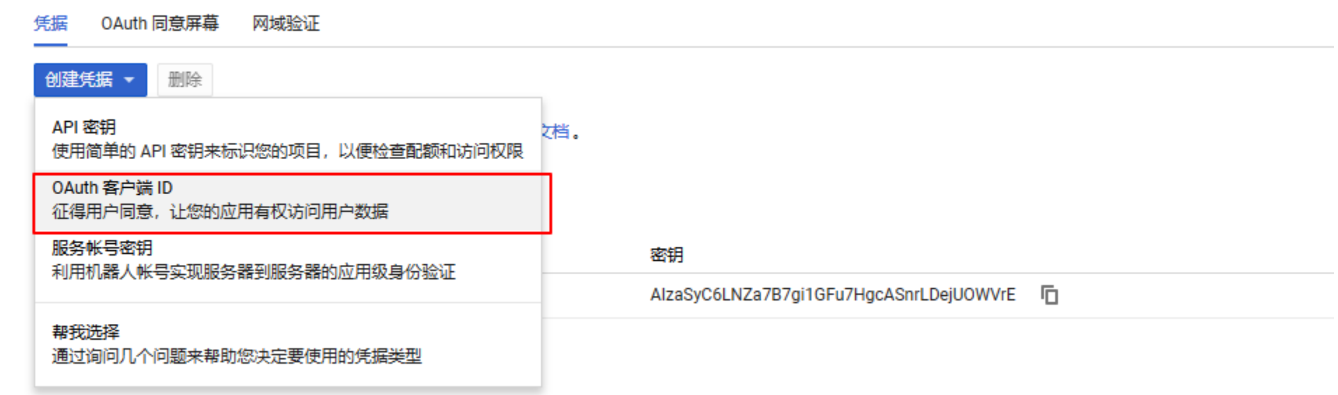


您必须要有凭据才能使用 API。请启用您要使用的 API，然后创建 API 所需的凭据。您可能需要拥有 API 密钥、服务帐号或 OAuth 2.0 客户端 ID，具体取决于所用 API。如需了解详情，请参阅身份验证文档。

类型选择为api key，随后即可生成api-key，此次生成的api-key如下

> AIzaSyC6LNZa7B7gi1GFu7HgcASnrLDejUOWVrE

再点击创建凭据，创建**oauth客户端id**类型的凭据


凭据 OAuth 同意屏幕 网域验证

创建凭据 删除

API 密钥
使用简单的 API 密钥来标识您的项目，以便检查配额和访问权限

OAuth 客户端 ID
征得用户同意，让您的应用有权访问用户数据

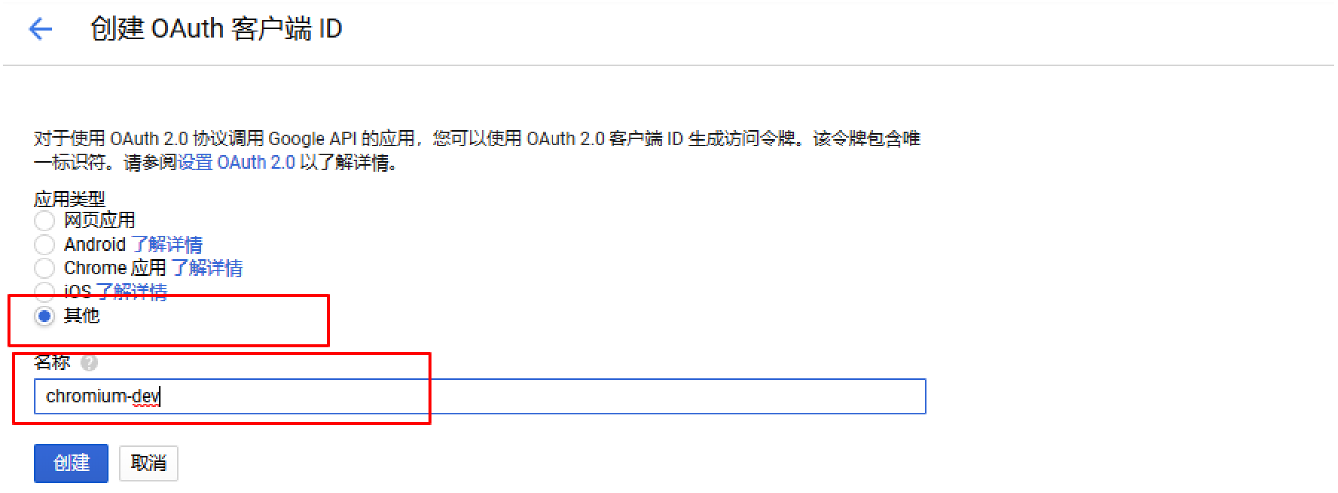
服务帐号密钥
利用机器人帐号实现服务器到服务器的应用级身份验证

帮我选择
通过询问几个问题来帮助您决定要使用的凭据类型

文档。

密钥
AIzaSyC6LNZa7B7gi1GFu7HgcASnrLDejUOWVrE

创建该类型的凭据之前，需要先设置Oauth同意屏幕，填写一个名字即可，点击保存，随后来到oauth客户端id创建页面，选择类型其他，随意填写一个名称


创建 OAuth 客户端 ID

对于使用 OAuth 2.0 协议调用 Google API 的应用，您可以使用 OAuth 2.0 客户端 ID 生成访问令牌。该令牌包含唯一标识符。请参阅设置 OAuth 2.0 以了解详情。

应用类型
- 网页应用
- Android 了解详情
- Chrome 应用 了解详情
- iOS 了解详情
- [✓ 其他]

名称
chromium-dev

创建 取消

创建以后可以得到

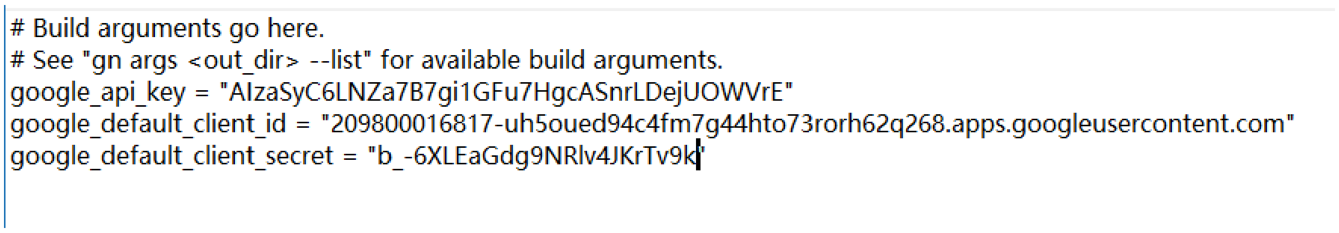
> 客户端id：209800016817-uh5oued94c4fm7g44hto73rorh62q268.apps.googleusercontent.com
>
> 客户端密钥：b_-6XLEaGdg9NRlv4JKrTv9k

6.2 在编译时指定google api key信息

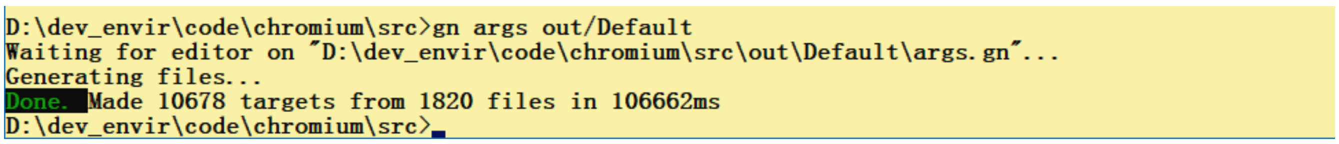
google api key得到以后，可以在编译程序时指定api key，使用如下命令打开参数文件

```
gn args out/Default
```

将key信息填入



```
# Build arguments go here.
# See "gn args <out_dir> --list" for available build arguments.
google_api_key = "AIzaSyC6LNZa7B7gi1GFu7HgcASnrLDejUOWVrE"
google_default_client_id = "209800016817-uh5oued94c4fm7g44hto73rorh62q268.apps.googleusercontent.com"
google_default_client_secret = "b_-6XLEaGdg9NRlv4JKrTv9k"
```



```
D:\dev_envir\code\chromium\src>gn args out/Default
Waiting for editor on "D:\dev_envir\code\chromium\src\out\Default\args.gn"...
Generating files...
Done. Made 10678 targets from 1820 files in 106662ms
D:\dev_envir\code\chromium\src>
```

随后再编译程序，就可以使用google的全部功能

6.3 在运行时指定key信息

使用命令行的setx工具永久设置相关环境变量

```
setx GOOGLE_API_KEY 生成的API密钥
setx GOOGLE_DEFAULT_CLIENT_ID 生成的客户端ID
setx GOOGLE_DEFAULT_CLIENT_SECRET 生成的客户端密钥
```

随后在打开chrome警告即消失

参考

官方文档

https://github.com/chromium/chromium/blob/master/docs/windows_build_instructions.md

segmentfault链接（推荐）

https://segmentfault.com/a/1190000016921832

google api key应用说明

http://www.chromium.org/developers/how-tos/api-keys

关于网络连接

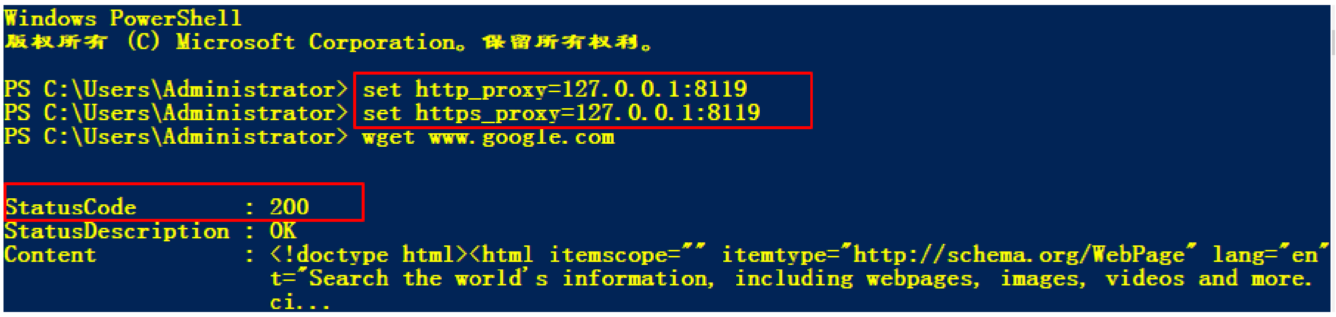
由于获取源码与工具的过程中需要访问googlecode，在gfw内无法直接访问，推荐使用shadowsock+privoxy的方式设置http代理，shadowsocks是一款socks协议代理工具，相信大家并不陌生，privoxy可以将socks代理转换为http代理，在windows下下载privoxy，安装步骤一路默认，安装完成后，默认配置文件是位于安装目录的config.txt，在config.txt中添加

```
listen-address 127.0.0.1:8119 # privoxy监听地址
forward-socks5 / localhost:1080 . # 所有请求转发到本机1080端口的socks服务器
```

具体配置文件的规则需要根据privoxy的版本做出一些相应的调整，设置完成后，查看privoxy的启动日志有无异常，如果启动成功，通常没有提示信息，如果有错误，则会给出相应提示。设置好以后，在cmd窗口中设置http_proxy与https_proxy环境变量，变量的值均指向本机privoxy代理

```
set http_proxy=127.0.0.1:8119
set https_proxy=127.0.0.1:8119
```

这种设置方式只在当前cmd窗口中有效，可以使用curl或wget工具进行网络测试，访问www.google.com



```
Windows PowerShell
版权所有 (C) Microsoft Corporation。保留所有权利。

PS C:\Users\Administrator> set http_proxy=127.0.0.1:8119
PS C:\Users\Administrator> set https_proxy=127.0.0.1:8119
PS C:\Users\Administrator> wget www.google.com


StatusCode        : 200
StatusDescription : OK
Content           : <!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="en"
                    t="Search the world's information, including webpages, images, videos and more.
                    ci...
```

wget是cygwin中的一个小工具，如果没有安装的话，可以在浏览器中配置http代理指向privoxy，测试代理是否配置成功