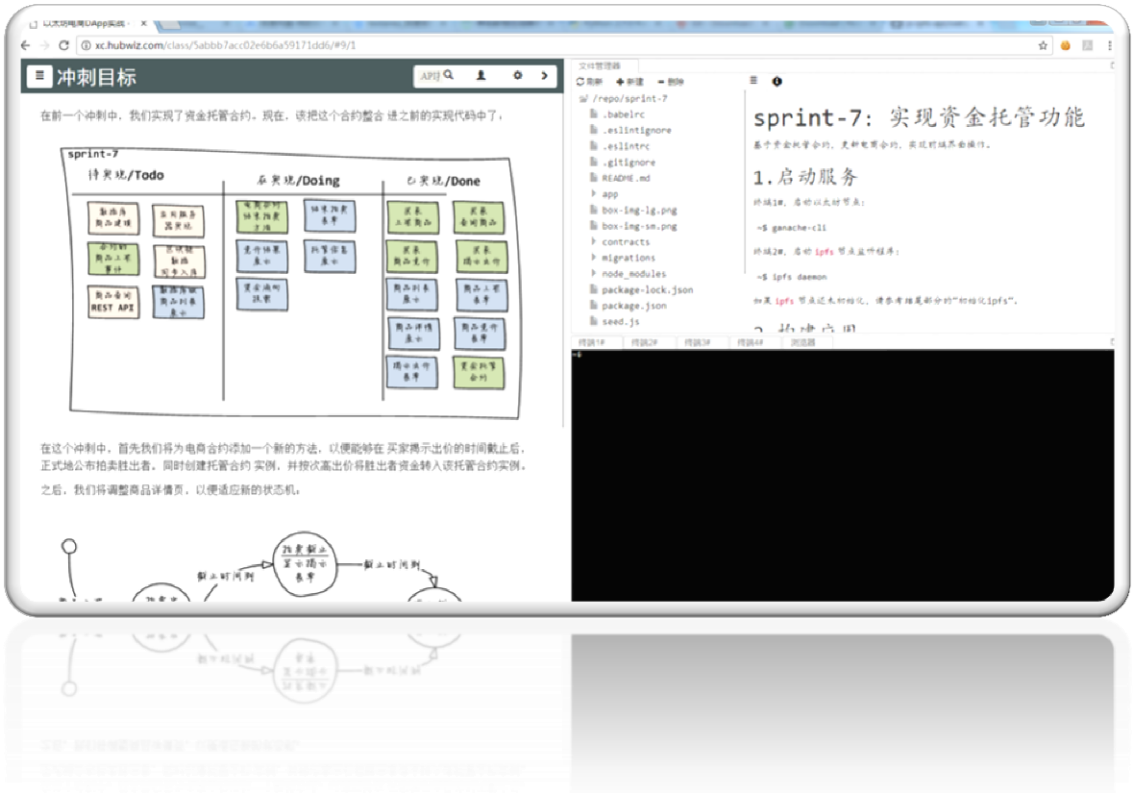
导读

本电子书由汇智网（http://www.hubwiz.com）创作，适用于 Windows 平台(Win7/Win10)下 ipfs 开发环境的搭建。

汇智网推出了在线交互式以太坊电商 DApp 实战开发课程，以去中心化电商应用为课程项目，内容涵盖以太坊合约开发、ipfs 去中心化文件系统、链下 mongodb 存储等诸多知识点，同时采用敏捷开发模式，通过 8 个冲刺周期的详细讲解与在线实践，将区块链的理念与去中心化思想贯穿于课程实践过程中，为希望快速入门区块链开发的开发者提供了一个高效的学习与价值提升途径。读者可以通过以下链接访问《以太坊电商 DApp 实战》在线教程：

http://xc.hubwiz.com/course/5abbb7acc02e6b6a59171dd6?affid=ipfs7878

教程预置了开发环境。进入教程后，可以在每一个知识点立刻进行同步实践，而不必在开发环境的搭建上浪费时间：



一、ipfs 节点安装与使用

1.1 下载节点软件

到官网下载 windows 版的 ipfs 节点软件：32 位，64 位

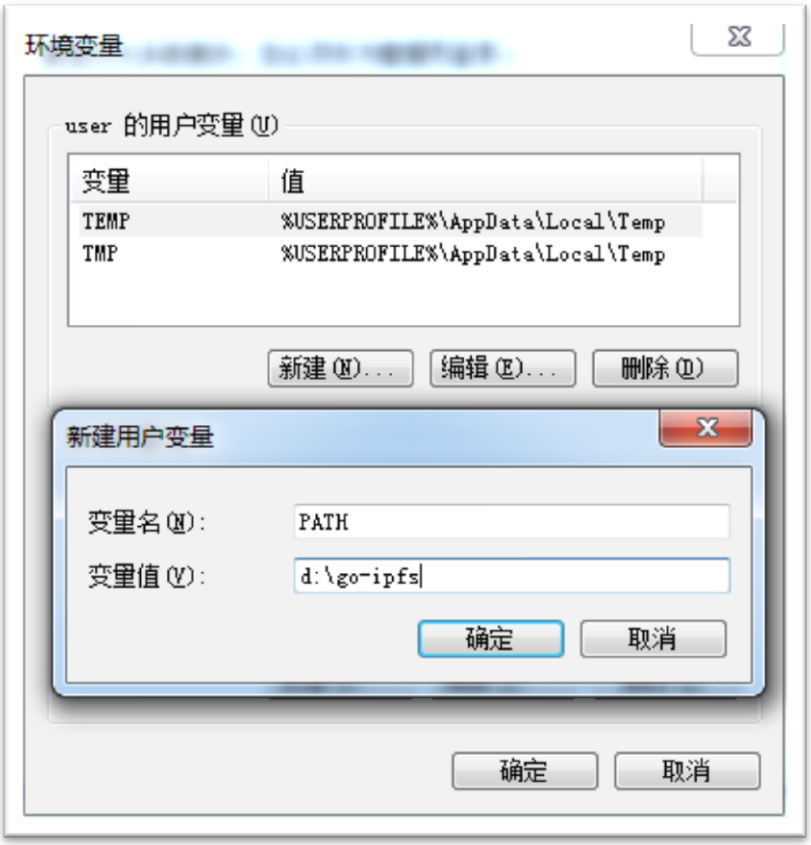
如果你不能访问官网，可以使用百度云盘镜像：32 位，64 位

1.2 解压节点软件

下载后解压到指定目录，例如 d:\go-ipfs，开一个控制台窗口，测试：

```
D:\go-ipfs > ipfs version
Ipfs version 0.4.14
```

可以将该目录加入环境变量 PATH，



或者将 d:\go-ipfs\ipfs.exe 拷贝到 windows 系统目录，以便在任何目录中

可以启动 ipfs.exe。

1.3 初始化本地仓库

和 git 类似，ipfs 节点也需要先初始化一个本地仓库。执行 init 子命令来初始化本地仓库：

```
D:\go-ipfs> ipfs init
Initializing IPFS node at C:\Users\hubwiz\.ipfs
generating 2048-bit RSA keypair...done
peer identity: QmQaTgU1TLNHPBEvLGgWK1G9FgVByyUZNVhDs789uWPtku
to get started, enter:

    ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv/readme
```

默认情况下，ipfs 将在当前用户主目录（例如：对于 hubwiz 用户，其主目录就是 C:\Users\hubwiz）下建立.ipfs 子目录，作为本地仓库的根目录。

如果你的 C 盘空间不够大，或者你就是希望使用其他目录作为本地仓库根目录，可以设置 IPFS_PATH 环境变量，使其指向目标路径，例如 D:\my_ipfs_root

1.4 重新初始化

如果你期望重新初始化节点，会提醒你不能这么做，否则会改写你的密钥：

```
D:\go-ipfs> ipfs init
Initializing IPFS node at C:\Users\hubwiz\.ipfs
Error: ipfs configuration file already exists!
```

```
Reinitializing would overwrite your keys.
```

这挡不住我们。如果你必须重新初始化的话，先删除原来的仓库根目录即可：

```
D:\go-ipfs> del C:\users\hubwiz\.ipfs
```

1.5 将文件添加到本地仓库

使用 add 子命令将指定的文件添加到本地仓库，例如将当前目录的 README.md 文件添加到本地仓库：

```
D:\go-ipfs> ipfs add README.md
465 B / ? [-------------------------------------------------------=--]
added QmXBpD37vBm5537pqHwyJRGSaX7hMrkHyp866wqEVU2BE8 README.md
```

ipfs 会根据文件的内容生成一个哈希值，例如：

```
QmXBpD37vBm5537pqHwyJRGSaX7hMrkHyp866wqEVU2BE8
```

你需要记录下这个编码，因为需要使用它来访问本地仓库（或 ipfs 网络）中的文件。

注意：ipfs 并不会无节制地将你本地仓库中的文件分布到其他 ipfs 节点中，如果没有其他的 ipfs 节点搜索你的文件（的哈希值），那么你本地仓库中的文件将始终只存在于本地。

1.6 访问 ipfs 文件

Ipfs 网络中只能通过内容的哈希值来访问文件，例如对于上面的 README.md 文件，我们使用 cat 子命令通过其哈希值来查看其内容：

```
D:\go-ipfs> ipfs cat QmXBpD37vBm5537pqHwyJRGSaX7hMrkHyp866wqEVU2BE8
```

控制台将输出内容：

```
# ipfs commandline tool


This is the [ipfs](http://ipfs.io) commandline tool. It contains a full ipfs
node.
…
```

1.7 将节点接入网络

执行 daemon 子命令将节点接入 ipfs 网络：

```
D:\go-ipfs> ipfs daemon
Initializing daemon...
...
Daemon is ready
```

只有当启动监听后，节点才能够接受 ipfs 网络中的内容检索请求，参与内容的交换与分布。

可以按 Ctrl+C 退出监听状态。

二、ipfs-api 安装与使用

Ipfs 节点提供和 REST API 接口，可供我们在程序代码中操作节点进行文件的上传等操作。不过大多数情况下，我们并不需要直接操作这个 REST 开发接口，而是使用经过封装的更友好的 ipfs-api，一个 nodejs 包。

2.1 安装 nodejs

到官网下载 nodejs 安装包：32 位，64 位。下载后双击进行安装即可。

开一个控制台窗口，测试：

```
C:\Users\hubwiz> node -v
V8.11.1
```

2.2 安装 ipfs-api

Ipfs-api 的安装需要 git 命令行，因此我们先安装 git。从官网下载 git 安装包：32 位，64 位。下载后双击安装即可。

执行 git 命令测试：

```
D:\test-ipfs-api> git version
git version 2.16.2.windows.1
```

ipfs-api 需要编译原生 node 模块，因此需要安装 VisualStudio 2015 和 python27。

官网下载 VisualStudio 2015 社区版，双击安装即可。

官网下载 Python27 安装包，双击安装并设置 PATH 环境变量使 python 可用。

重新开一个控制台，使环境变量生效。现在安装 ipfs-api：

```
D:\test-ipfs-api> npm install ipfs-api
+ ipfs-api@20.0.1
```

```
added 1 package, updated 1 package and moved 1 package in 22.138s
```

2.3 测试代码 – nodejs

在 D:\test-ipfs-api 目录下创建一个测试脚本 test.js：

```
const ipfsAPI = require('ipfs-api')
const ipfs = ipfsAPI('localhost', '5001', {protocol: 'http'})
const buffer = Buffer.from('this is a demo')
ipfs.add(buffer)
    .then( rsp => console.log(rsp[0].hash))
.catch(err => console.error(err))
```

执行这个脚本：

```
D:\test-ipfs-api> node test.js
QmfQS4vm9YZTAyGZEkDqm81xripwsK3NgqfNkbCdoeEw5i
```

也就是说，我们将内容`this is a demo`添加到本地仓库后，得到哈希值 QmfQS4vm9YZTAyGZEkDqm81xripwsK3NgqfNkbCdoeEw5i。现在可以使用 cat 子命令来查看这个哈希值对应的内容：

```
D:\test-ipfs-api> ipfs cat QmfQS4vm9YZTAyGZEkDqm81xripwsK3NgqfNkbCdoeEw5i
```

控制台会输出我们之前上传的内容：

```
this is a demo
```

ipfs 进入监听状态后，提供了一个 http 网关，让我们可以使用浏览器来访问 ipfs 上的内容。网关默认在本机（127.0.0.1）的 8080 端口监听，因此使用你的浏览器访问这个 URL：

```
http://127.0.0.1:8080/ipfs/QmfQS4vm9YZTAyGZEkDqm81xripwsK3NgqfNkbCdoeEw5i
```

同样可以看到我们之前上传的内容。

注意：需要首先启动监听器（ipfs daemon）并且你的浏览器和 ipfs 节点在同一台计算机。

三、在浏览器中访问 ipfs

ipfs-api 也支持在 browser 使用。最简单的方法是使用专门针对浏览器的封装库，在 html 中引用即可：

```
<script src="https://unpkg.com/ipfs-api/dist/index.js"></script>
```

这个特别封装的库会创建一个全局对象 ipfsAPI，我们在浏览器脚本中可以直接使用，例如：

```
var ipfs = window.IpfsApi('localhost', '5001')
```

这种方法比较简单，因此下文不再描述。接下来我们将使用更加工程化的方法，采用 webpack 来直接在前端脚本中使用 ipfs-api 的 nodejs 包。

3.1 HTML 页面

在 D:\test-ipfs-api 目录下创建 index.html：

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
</head>
<body>
  <textarea id="content">THIS IS ANOTHER DEMO</textarea>
  <button id="upload">Upload</button>
  <script src="./bundle.js"></script>
</body>
</html>
```

我们的目标是，当点击按钮时，我们将文本框的内容上传到 ipfs

3.2 前端脚本

在 D:\test-ipfs-api 目录下编写脚本 app.js：

```
import ipfsAPI from 'ipfs-api'
const ipfs = ipfsAPI('localhost', '5001', {protocol: 'http'})
```

```
window.addEventListener('load', function() {
  let btn = document.querySelector('#upload')
  let txt = document.querySelector('#content')
  btn.addEventListener('click',()=>{
    let buffer = Buffer.from(txt.value, 'utf-8');
    ipfs.add(buffer)
      .then( rsp => console.log(rsp[0].hash))
      .catch(err => console.error(err))
  })
})
```

3.3 webpack 配置

在 D:\test-ipfs-api 目录下编写配置文件 webpack.config.js：

```
const webpack = require('webpack');
const path = require('path');
const CopyWebpackPlugin = require('copy-webpack-plugin');

module.exports = {
  entry: './app.js',
  output: {
    path: path.resolve(__dirname),
    filename: 'bundle.js'
  },
  module: {
    loaders: [
      {
        test: /\.js$/,
        exclude: /(node_modules|bower_components)/,
        loader: 'babel-loader',
        query: {
          presets: ['es2015'],
          plugins: ['transform-runtime']
```

```
      }
    }
  ]
 }
}
```

3.4 前端脚本打包

执行 webpack 打包：

```
D:\test-ipfs-api> webpack
```

3.5 配置 ipfs 的 CORS 策略

由于需要从网页中访问 ipfs 节点，这就引入了跨域安全问题，因此我们需要配置 ipfs 节点使其允许跨域请求：

```
D:\>ipfs config --json API.HTTPHeaders.Access-Control-Allow-Origin '["*"]'
```

3.6 配置 ipfs 的 API 监听地址

由于 ipfs 节点默认在本机（127.0.0.1）的 5001 端口监听 API 请求，因此如果你的浏览器和 ipfs 节点不在同一台机器上，需要让 ipfs 节点监听公开地址：

```
D:\> ipfs config --json Addresses.API '"/ip4/0.0.0.0/tcp/5001"'
```

当然，如果你的浏览器和 ipfs 节点在同一台机器上，就不需要进行这个配置了。

3.7 配置 ipfs 的网关的监听地址

由于 ipfs 节点的 http 网关默认在本机（127.0.0.1）的 8080 端口监听 http 请求，因此如果你的浏览器和 ipfs 节点不在同一台机器上，就需要让 ipfs 网关监听公开地址：

```
D:\> ipfs config --json Addresses.Gateway '"/ip4/0.0.0.0/tcp/8080"'
```

3.8 测试网页

首先启动 ipfs 监听：

```
D:\>ipfs daemon
```

然后在测试目录下启动 web 服务器，这里使用 python 内置的简单服务器，当然你可以使用任何熟悉的 web 服务器：

```
D:\test-ipfs-api> python -m SimpleHTTPServer
Serving HTTP on 0.0.0.0 port 8000 ...
```

现在打开你的浏览器，访问 http://127.0.0.1:8000/，一切顺利的话，你可以看到一个文本框和一个按钮，点击按钮，即可将文本框的内容上传到 ipfs 节点。