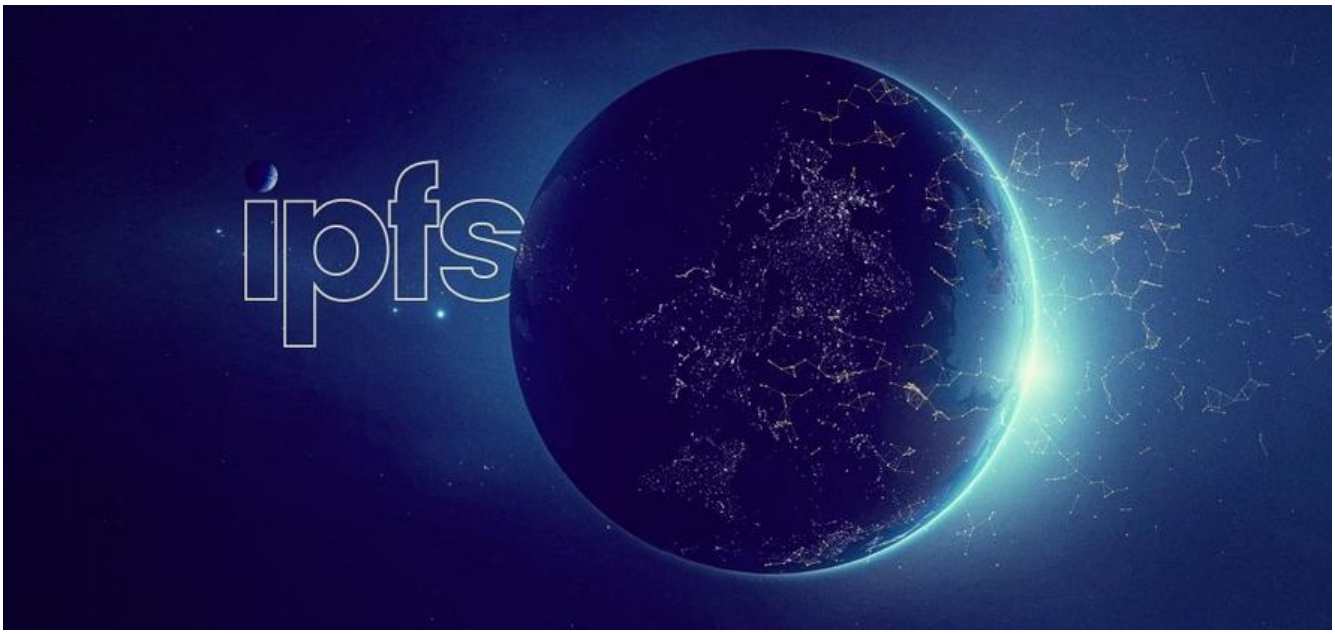
链滴

IPFS 入门笔记

作者：88250

原文链接：https://ld246.com/article/1511015097370

来源网站：链滴



IPFS 是什么

IPFS（InterPlanetary File System，星际文件系统）是永久的、去中心化保存和共享文件的方法，这一种内容可寻址、版本化、点对点超媒体的分布式协议。

- 内容可寻址：通过文件内容生成唯一哈希值来标识文件，而不是通过文件保存位置来标识。相同内的文件在系统中只会存在一份，节约存储空间
- 版本化：可追溯文件修改历史
- 点对点超媒体：P2P 保存各种各样类型的数据

可以把 IPFS 想象成所有文件数据是在同一个 BitTorrent 群并且通过同一个 Git 仓库存取。

总之，它集一些成功系统（分布式哈希表、BitTorrent、Git、自认证文件系统）的优势于一身，是一很厉害的文件存取系统。

IPFS 使用场景

IPFS 的发明者 Juan Benet（juan@benet.ai）在 IPFS 技术白皮书中假设了一些使用场景：

- 在 /ipfs 和 /ipns 下挂载全球文件系统
- 挂载的个人同步文件夹，拥有版本功能
- 文件加密，数据共享系统
- 可用于所有软件的带版本的包管理器（已经实现了： https://github.com/whyrusleeping/gx）
- 可以作为虚机的根文件系统
- 可以作为数据库：应用可以直接操作 Merkle DAG，拥有 IPFS 提供的版本化、缓存以及分布式特性
- 可以做（加密）通讯平台
- 各种类型的 CDN
- 永久的 Web，不存在不能访问的链接

我觉得作为数据库这一点对应用开发者来说会很有用。

安装与初始化

下载 go-ipfs 解压（下面的示例我是在 Windows 10 上做的，解压目录为 D:\o-ipfs），然后到解压目录执行命令 ipfs init，将在用户 home（~）下建立 .ipfs 目录存放数据，默最大存储 10G。init 命令可以带参，比如修改最大存储、目录等，具体参考 ipfs init help。

继续执行命令 ipfs daemon 启动节点服务器：

- 加入 IPFS 网络
- 本地 HTTP 服务器，默认 8080 端口
- 处理后续 ipfs 的客户端命令

新开一个命令行，执行命令 ipfs id 以查看当前节点标识:

```
{
  "ID": "QmSYF1HZxhPUWWGrz5bMn16tdD73AeMVhp7pNSHkVCMF7R",
  "PublicKey": "....",
  "Addresses": [
    "/ip4/169.254.40.215/tcp/4001/ipfs/...",
    ....
  ],
  "AgentVersion": "go-ipfs/0.4.12/",
  "ProtocolVersion": "ipfs/0.1.0"
}
```

浏览器访问 http://localhost:5001/webui 进入管理界面，查看系统状态、管理文件以及配置系统。

配置

除了使用 Web 管理界面修改配置外，也可以直接用命令行 ipfs config show > ipfs.conf 先导出当配置（JSON 格式，配置项不多且含义明显），改完后使用 ipfs config replace ipfs.conf 更新配置重启服务器就生效了。当然，修改配置也可以直接用 ipfs config edit。

服务器最终使用的配置文件保存在 ~/.ipfs/config 中，对比刚刚导出的文件我们发现导出的文件只比个 config 少了一项 Identity.PrivKey，即节点初始化时自动生成的 RSA 私钥。

密钥对

节点初始化时会自动生成 RSA 密钥对，并且私钥没有设置密码。

公钥通过多重哈希得到节点 id（即上面的 QmSYF1HZxhPUWWGrz5bMn16tdD73AeMVhp7pNSHVCMF7R），节点服务器启动后会和其他节点交换公钥，后续通讯时使用对方公钥加密数据，通过多哈希对方公钥、对比对方节点 id 来确认是否正在和正确的节点交互。

私钥用来解密接收到的数据，也用于 ipns 来绑定文件名。整个过程没有引入证书，仅是使用了 PKI 制。

总之，我觉得可以暂时不用关心密钥对，可能只有在一些使用场景下面才需要吧。

添加文件

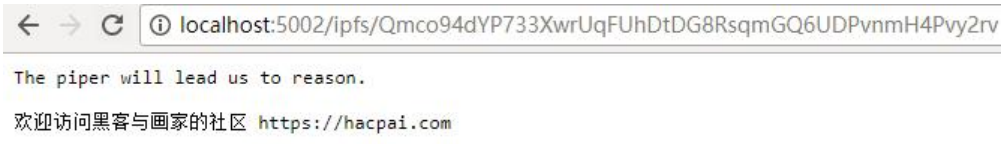
我当前目录结构是这样的：

```
D:\GO-IPFS
│  build-log
│  config
│  install.sh
│  ipfs.conf
│  ipfs.exe
│  LICENSE
│  README.md
│
└─b3log
    └─hacpai
            README.md
```

我准备添加的目录是 b3log，执行命令：

```
D:\go-ipfs>ipfs add -r b3log
 94 B / 94 B [=============================================================================================] 100.00% 0s
added Qmco94dYP733XwrUqFUhDtDG8RsqmGQ6UDPvnmH4Pvy2rv b3log/hacpai/README.md
added Qmbkno2HVZdW7XfwsVjmuu9VDKBByczFR8qwsBXMjMrjPQ b3log/hacpai
added QmPxebZuW2pgfzj5JWq22KUzxStmqQ6i7YUK9Sq9xepXT9 b3log
```

这样我们使用 ipfs cat /ipfs/Qmco94dYP733XwrUqFUhDtDG8RsqmGQ6UDPvnmH4Pvy2rv 就以查看 README.md 了。在其他节点上也可以，只要记住这个文件的哈希值就行了。我们可以在自的 HTTP 网关上试试（注意我的端口改成了 5002，你的默认应该是 8080）：



当然也可以用 ipfs 官方的 HTTP 网关：https://ipfs.io/ipfs/Qmco94dYP733XwrUqFUhDtDG8RsqGQ6UDPvnmH4Pvy2rv

获取文件

ipfs get /ipns/QmSYF1HZxhPUWWGrz5bMn16tdD73AeMVhp7pNSHkVCMF7R

将获取刚才我们发布的 b3log 目录。

Pin

IPFS 的本意是让用户觉得所有文件都是在本地的，没有“从远程服务器上下载文件”。Pin 是将文件期保留在本地，不被垃圾回收。

执行 ipfs pin ls 可以查看哪些文件在本地是持久化的，通过 add 添加的文件默认就是 pin 过的。

绑定节点名

每次修改文件后 add 都会返回不同的哈希，这对于网站来说就没法固定访问地址了，所以我们需要过 ipns 来“绑定”节点名。

上面 b3log 目录的哈希值是 QmPxebZuW2pgfzj5JWq22KUzxStmqQ6i7YUK9Sq9xepXT9，我们整个目录作为节点根目录发布：

```
D:\go-ipfs>ipfs name publish QmPxebZuW2pgfzj5JWq22KUzxStmqQ6i7YUK9Sq9xepXT9
Published to QmSYF1HZxhPUWWGrz5bMn16tdD73AeMVhp7pNSHkVCMF7R: /ipfs/QmPxeb
uW2pgfzj5JWq22KUzxStmqQ6i7YUK9Sq9xepXT9
```

然后我们就可以通过 ipns 访问了，**注意是 ipns**：

```
D:\go-ipfs>ipfs cat /ipns/QmSYF1HZxhPUWWGrz5bMn16tdD73AeMVhp7pNSHkVCMF7R/ha
pai/README.md
The piper will lead us to reason.

欢迎访问黑客与画家的社区 https://hacpai.com
```

以后每次更新文件都再 publish 一下就行了。目前（v0.4.12）使用 ipns 访问会很慢，据说 v0.4.14 解决。

DNS 解析

IPFS 允许用户使用现有的域名系统，这样就能用一个好记的地址来访问文件了，比如：

```
D:\go-ipfs>ipfs cat /ipns/ipfs.b3log.org/hacpai/README.md
The piper will lead us to reason.

欢迎访问黑客与画家的社区 https://hacpai.com
```

只需要在 DNS 解析加入一条 TXT 记录：

记录类型	主机记录	记录值
TXT	ipfs	dnslink=/ipns/QmSYFHZxhPUWWGrz5bMn16tdD73AeMVhp7pNSHkVCMF7R

总结

- IPFS 是永久的、去中心化保存和共享文件的方法，这是一种内容可寻址、版本化、点对点超媒体的布式协议
- 我们可以用它来存取文件，数据永不丢失
- 应用可以用它来做数据库，自动拥有版本化、缓存及分布式特性
- 官方参考实现使用 golang 编写，JavaScript、Python、C 等语言在陆续开发中
- 总之，IPFS 是一套非常厉害的文件系统

进一步学习

- IPFS 去中心化数据结构（一） —— 数据结构、寻址和中心化的 Web
- IPFS 去中心化数据结构（二） —— 通过加密哈希进行内容寻址

- IPFS 去中心化数据结构（三） —— 内容标识符（CID）
- IPFS 去中心化数据结构（四） —— 默克尔树和有向无环图（DAG）
- IPFS 数据链接与内容寻址

欢迎大家关注 https://ld246.com/tag/ipfs，我们一起学习和讨论 IPFS 相关知识。

参考资料

- IPFS White Paper
- IPFS Specs
- IPFS Wikipedia
- IPFS 筆記和教學 (繁體中文)
- IPFS：替代HTTP的分布式网络协议