IPFS部署操作手册

### [ipfs命令手册](http://cw.hubwiz.com/card/c/ipfs/) 汇智网http://cw.hubwiz.com/card/c/ipfs/1/14/3/



# 部署环境说明

该部署操作手册适用于ubuntu 20.04 server操作系统，要求有公网连接。是否适用于ubuntu 18.04 server或更高版本需要验证，不确定能否适用于desk版本。

本文只适用于go-ipfs的部署操作。

# 系统与程序环境配置

go-ipfs是基于Go语言的项目，环境要求go语言环境，并在系统变量中进行相应的配置。

## 下载Go语言包(可以打开的网址https://studygolang.com/dl)

（这个网站的下的文件能不能最后编译成功还未知，用的还是以前下载的go1.17的软件包才可以。）

IPFS是基于Go语言的项目，环境要求go version 1.14+。在Go的官方网站下载最新的版本即可[https://golang.org/dl/](https://link.zhihu.com/?target=https://golang.org/dl/" \t "https://zhuanlan.zhihu.com/p/_blank)。现在go的最新版本为go1.17.8， 可以用以下命令下载到服务器本地并解压至指定的目录/usr/local：

wget <https://golang.org/dl/go1.17.8.linux-amd64.tar.gz> 如果这个网址进入不了就用下面别一个网址下载。注意有些网址下载的go包是最后用命令apt install make解释不了的。

**wget <https://studygolang.com/dl/golang/go1.18.linux-amd64.tar.gz>**

wget https://golang.org/dl/go1.17.8.linux-amd64.tar.gz

wget [https://dl.google.com/go/go1.18.3.linux-amd64.tar.gz](https://dl.google.com/go/go1.18.3.linux-amd64.tar.gz" \t "_blank)

#上面看哪个命令地址能下载成功。

**sudo tar -zxvf go1.17.8.linux-amd64.tar.gz -C /usr/local #解压到指定/opt目录用**

注:如果wget失败可以到官网[https://golang.org/dl/go1.17.8.linux-amd64.tar.gz](https://link.zhihu.com/?target=https://golang.org/dl/go1.14.6.linux-amd64.tar.gz" \t "https://zhuanlan.zhihu.com/p/_blank)下载镜像，然后在那个目录下打开终端执行上面的命令的第二句。

如果服务器上下载失败，可以在终端机下载完成后，用Xftp连接服务器，将文件拖过去，至于如何连接，请看这篇文章[保姆级教程——Xshell连接虚拟机中的Ubuntu并通过Xftp传输文件](https://link.zhihu.com/?target=https://blog.csdn.net/ExcaliburUlimited/article/details/107718611" \t "https://zhuanlan.zhihu.com/p/_blank)，Xshell和Xftp连接过程相同。

## 部署和配置Go环境

在进入用户主目录，并新建名为go的文件夹，在go的文件夹中建立三个子目录(名字必须为src、pkg和bin)并赋权。创建目录赋权命令如下:

cd ~ 改为cd /root 为好

sudo mkdir -p go/{src,pkg,bin}

sudo chmod u+x go/{src,pkg,bin}

运行ls -l命令查看目录创建及权限：

|  |
| --- |
|  |

为用户配置go语言环境变量，首先输入：

vi ~/.profile #这里是配置在普通用户目录下/home/$userdir/.profile

打开用户的环境变量配置文件，按”i”键进入编译模式，在最后添加如下内容：

export PATH**=**$PATH:/usr/local/go/bin

export GOROOT**=**/usr/local/go

export GOPATH**=**$HOME/go

export PATH**=**$PATH:$HOME/go/bin

再按”Esc”键退出编译，接着输入:wq，回车就可以保存修改并退出。

输入如下命令即激活修改后的环境变量配置文件：

source ~/.profile

最后验证一下是否成功，输入如下命令查看结果：

go version #（下面用来看ipfs安装好没有就用ipfs version这个命令）

go env

|  |
| --- |
|  |

出现上图结果即表示go语言环境部署成功。

但是这样关掉终端，配置就会失效，所以需要对~/.bashrc中也进行相应的配置，输入以下命令开始编译：

vi ~**/**.**bashrc #这个配置在超级用户目录里, 直接用下面这个命令好。**

**vi /root/.bashrc**

然后在最后面添加：

export PATH**=**$PATH:/usr/local/go/bin

export GOROOT**=**/usr/local/go

export GOPATH**=**$HOME/go

export PATH**=**$PATH:$HOME/go/bin

再按”Esc”键退出编译，接着输入:wq，回车就可以保存修改并退出。

输入如下命令即激活修改后的环境变量配置文件：

source ~/.bashrc

Go环境部署与配置完成。

## 安装工具包

部署ipfs需要使用gcc与make等工具，运行以下命令更新系统并安装最新的工具：

sudo su #先切换到root权限, 防止有些文件还是提示权限问题。要不然可能出现下图情况。



sudo apt update && apt upgrade -y

sudo apt-get install git

sudo apt install gcc

sudo apt install make

系统更新及工具安装完成后，ipfs部署准备完成。

# Ipfs安装部署

## 下载go-ipfs源码

因为go get国内基本上下载不了，所以需要开启并加上镜像进行下载，命令如下：

go env -w GO111MODULE=on

go env -w GOPROXY=https://goproxy.cn,direct

go get -u github.com/ipfs/go-ipfs

下载成功后如下图提示：

|  |
| --- |
|  |

go-ipfs的源码将会下载到~/go/pkg/mod/github.com/ipfs/go-ipfs@v0.12.2/的目录下。

## 编译go-ipfs源码

进入cd /root/go/pkg/mod/github.com/ipfs/go-ipfs@v0.12.0/，运行命令对以下两个文件进行赋权：

chmod 777 bin/check\_go\_version

chmod 777 plugin/loader/preload.sh

然后执行以下命令开始编译go-ipfs：

make install

如果编译正常结束则会出现以下提示：

|  |
| --- |
|  |

编译成功，输入以下命令验证：

ipfs version

如出现以下提示，则表示go-ipfs部署成功：

|  |
| --- |
|  |

注：如果编译时碰到以下问题：

|  |
| --- |
|  |
|  |

则需要对提示Permission denied的文件进行chmod 777赋权，之后再重新运行make install进行编译即可。

## 恢复配置

恢复配置，保持系统运行安全，命令如下：

go env -w GO111MODULE**=**off

Ipfs部署完成。

## Ipfs节点参数配置

使用以下命令对ipfs节点进行初始化

|  |
| --- |
| ipfs init #初始化 |
|  |

初始化后显示上图表示初始化成功。

根据上图初始化后生成ipfs配置目录在/root/.ipfs/下，使用命令切换到ipfs目录下：

|  |
| --- |
| cd /root/.ipfs/ #切换目录  ls -l #查看目录文件 |

使用以下命令打开config主配置文件：

|  |
| --- |
| vim /root/.ipfs/config |



打开文件后显示为上图，找到红框中的API和Gateway选项，将IP更换为本机真实IP地址。

为了防止其他外网ipfs节点的接入，使用以下命令删除外网节点：

如果不删除这些外网节点，之后用命令./ipfs-cluster-ctl peers ls查询时就会显示出一些外网IP的地址如下图红柜中的IP就是其他外网的：, 现在这个命令还查不了，是配置完了后才行。



|  |
| --- |
| ipfs bootstrap rm --all #为了不连接全球的 IPFS 网络，你需要将默认的 bootstrap 的节点信息删除。 |
|  |

上图显示表示成功删除外网节点。

使用以下命令启动ipfs节点：

|  |
| --- |
| nohup ipfs daemon & （#用nohup &方便在后台运行不掉线。） |
|  |

显示以上图的参数表示ipfs节点启动成功。

浏览器上打开 [http://192.168.1.24:5001/webui](http://192.168.1.23:5001/webui) 可以看到IPFS的网站。



从网页上的第2点，看到有需要输入的两条命令提示，照着输入。

ipfs config --json API.HTTPHeaders.Access-Control-Allow-Origin '["http://192.168.1.140:5001", "http://localhost:3000", "http://127.0.0.1:5001", "https://webui.ipfs.io"]'

ipfs config --json API.HTTPHeaders.Access-Control-Allow-Methods '["PUT", "POST"]'

输入完上面命令后，关闭ipfs节点可使用以下命令查看进程：

|  |
| --- |
| ps -ef | grep ipfs |

对上图红框中的进程号使用kill -9 13535杀死该进程表示关闭ipfs节点。

最后再次运行 nohup ipfs daemon &

再次浏览器上打开 [http://192.168.1.24:5001/webui](http://192.168.1.23:5001/webui) 变成如下图，网站提示连接到IPFS。



# Ipfs-cluster（IPFS的集群安装）

## 初始化ipfs-cluster-service

使用以下命令对ipfs-cluster-service源码包进行解压：（目的地目录没要求，解压到随便一个目录下也行, 但是为了接下来的全部统一性，所以软件包的地址统一解压到/opt/ 相对应的目录下。）

|  |
| --- |
| tar -zxvf ipfs-cluster-service\_v0.14.5\_linux-amd64.tar.gz -C /opt/ |

使用以下命令进入目录查看文件：

|  |
| --- |
| cd /opt/ipfs-cluster-service/ #切换目录  ls -l #查看文件 |

上图红框中的文件为本次ipfs节点统一管理工具。

使用以下命令对ipfs-cluster-service工具进行初始化：

|  |
| --- |
| cd /opt/ipfs-cluster-service/ #进入目录  ./ipfs-cluster-service init #初始化管理工具 |

显示上图表示管理工具初始化成功。

使用以上命令初始化后，可以发现在root目录下生成了/root/.ipfs-cluster/目录文件。

使用以下使用以下命令进入目录打开主配置文件：

|  |
| --- |
| cd /root/.ipfs-cluster/ #切换目录  ls -l #查看文件 |

## 配置集群参数

使用以下命令打开配置文件：

|  |
| --- |
| vim /root/.ipfs-cluster/service.json |

将上图红框中的参数复制更换到其他节点的secret上。

我的secret密钥是b509ba9df0d962509ac6a1c058dbe0daa961fcbebb285b8b8c629549977b664d

注意：每台需要加入ipfs集群中的节点，根据上图红框中的secret参数都要一致。

使用命令打开文件编辑：

|  |
| --- |
| vim /root/.ipfs-cluster/service.json |

将上图红框中的三个地方的IP改为本地服务的真实IP地址。

## 启用ipfs管理工具

使用以下命令启动管理工具获取数据：

|  |
| --- |
| cd /opt/ipfs-cluster-service/ #切换目录  ./ipfs-cluster-service daemon #启动ipfs管理工具 |

将红框中的内容复制下来，然后ctrl+c关闭管理节点。

使用以下命令对管理工具重启启动：

|  |
| --- |
| cd /opt/ipfs-cluster-service/ #切换目录  ./ipfs-cluster-service daemon bootstrap /ip4/192.168.1.24/tcp/9096/p2p/12D3KooWJSgxUaxdhFhtAuoAMYCpnPxGphZ7ZrC7PqDBiTX1ikW8 #启动管理工具 |

如上图显示表示启动成功。

注意：运行上图的启动管理工具前，首先需要使用ps -ef | grep ipfs查看ipfs节点是否有启动，没有的话就运行ipfs daemon & 启动ipfs节点。

## 安装ipfs-cluster-ctl

使用以下命令解压ipfs-cluster-ctl源码包：

|  |
| --- |
| tar -zxvf ipfs-cluster-ctl\_v0.14.5\_linux-amd64.tar.gz -C /opt/ |

使用以下命令进入解压目录查看文件：

|  |
| --- |
| cd /opt/ipfs-cluster-ctl/  ls -l |

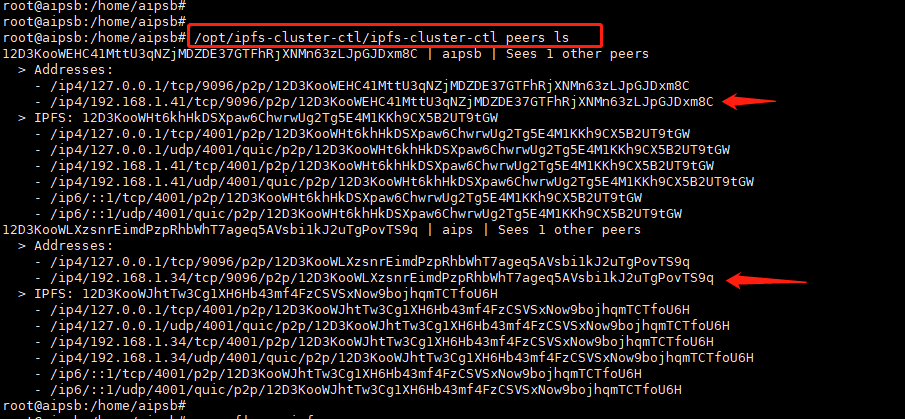
使用以下命令运行上图中的ipfs-cluster-ctl脚本：

|  |
| --- |
| ./ipfs-cluster-ctl id |

显示为上图表示安装成功。

/opt/ipfs-cluster-ctl/ipfs-cluster-ctl peers ls #这个是默认解压到opt目录下面的命令路径。

#这条命令查看集群的命令，显示出总多少台机器在这个集群。如下图这里显示有2台。



## IPFS的上传与下载文件 (安装好集群后，进行某台机器上转文件另一台机器下载文件的试验。)

rz 命令随便上传一个文件到服务器的某个目录。比如到 /home/jy/555.txt

./ipfs-cluster-ctl add 555.txt #文件名,此命令表示上传文件

转着系统自动生成哈希值。如下图，QmNbuSpWHVDuwKX26UkfsSabh5pcb2ME1UK9mgXrSQuJr7就是哈希值。



./ipfs-cluster-ctl pin add 上传文件hash值    #此命令表示数据分发。

然后去到另一台机器使用下载命令ipfs get  哈希值         #只要有ipfs节点都是可以下载的

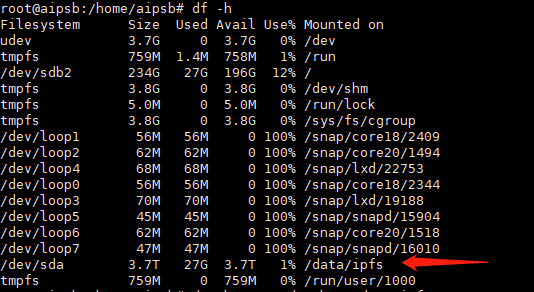
cat 555.txt #查看刚才下载的内容。

1. **IPFS群组的存储路径与管理的问题。**

**每一个节点默认的存储路径为本机的/root/.ipfs 这个时候我们想要更大的空间需要进行挂载硬盘的操作，如下：**

**mount /dev/sdb /root/.ipfs #这.ipfs的目录是系统默认的路径，挂载之后这个路径就可以存储挂载盘的最大容量。注意事项挂载硬盘时要准确到blocks与datastore目录的上一级，**

**又如图mount /dev/sda /data/ipfs，因为blocks与datastore这两个目录可以删除后程序重新启动后可以自动新建，但上级目录不能自动创建。**



**如果这个新挂载的硬盘是全新的，需要进行格式化操作mkfs.xfs /dev/sdb**

**关于存储目录也可以参考以下网：https://blog.csdn.net/wzygis/article/details/114324354?spm=1001.2101.3001.6661.1&utm\_medium=distribute.pc\_relevant\_t0.none-task-blog-2%7Edefault%7ECTRLIST%7EPayColumn-1.pc\_relevant\_default&depth\_1-utm\_source=distribute.pc\_relevant\_t0.none-task-blog-2%7Edefault%7ECTRLIST%7EPayColumn-1.pc\_relevant\_default&utm\_relevant\_index=1**

1. **其中一个节点上传的新文件或数据压缩包，如果只是文件名不一样，IPFS可以识别到里面的东西是一样的，此时每个节点的总存储空间没有变化。但此时这个新文件名的哈希值是不一样。**



1. **修改默认存储路径与存储最大的空间限制。一共要修改两个配置文件。**

**其中一个：改下图的这两个箭头的位置使得改变存储位置。**

**vim /root/.ipfs/config**



另外一个文件要修改

vim /root/.ipfs/datastore\_spec

改下面箭头处两个地方的存储路径。



1. 对之前上传的文件进行删除与空间解放的测试。

先进行 ipfs pin rm 哈唏值 #命令从本地存储中解除指定ipfs对象的固定。

ipfs repo gc #命令扫描仓库中的对象，并依照先后顺序移除没有固定的对象， 以便回收磁盘空间。进行过ipfs pin add 哈希值的数据不会被消除掉。

然后再用du -h --max-depth=1 /root/.ipfs #查IPFS默认的存储空间容量做对比，如果之前没有进行修改存储路径的话。

1. 进行验证文件哈希是否存在。

删除或着上传文件后, 除了用du -h --max-depth=1命令查容量外，还可以配合下面的命令进行验证文件哈希是否存在。

ipfs pin ls ##这个查出本地节点所有已经固定的哈希值。

Ipfs pin ls |wc -l ##这里查的哈希数的数量，上传一个压缩包有可能产生很多数量。

用ipfs pin ls |grep Qmf6r8PYPM158amesk51h5iGQ3jWRLLzjnUZ69en68ZTQc 这个命令查询在不在本机内。

而用/opt/ipfs-cluster-ctl/ipfs-cluster-ctl pin ls ##这个命令查出的固化的哈希值，是一个压缩包文件只会多一个哈希值的数量出来。

1. 对于ipfs add 文件名，与ipfs pin add 文件的哈希值，的使用。这两个命令使用只能把文件上传到单机的存储库上，别的节点机需要下载时也可以通过哈希值下载，但是一旦本机关了或坏了，其他节点机不能下载。

与上面第5节说的命令 ./ipfs-cluster-ctl add 文件名，是不一样的，第5节这个命令是可以即时同步到所有节点机上的存储库上的。（一个只对单机存储容量有影响，一个是对所有节点机存储容量有影响。）

1. 对于大文件的上传比较好的方式用法，对于超过1G或更大文件的上传：

步骤1：ipfs add 文件名 #得到哈希值。

步骤2： 工具包的目录/ipfs-cluster-ctl pin add 哈希值,

这样上传速度会快很多，不会影响到后面的进程速度。要不然一开始用工具包的目录/ipfs-cluster-ctl  add 文件名，0.9G的文件需要用时6-9分钟。

1. 大文件上传到IPFS存储系统上的用时，这里以本机上传到IPFS储存方式计算，所以暂时不考虑外网网速的影响。以下以1.1G与4.2G的压缩包作为测试时间。（硬件平台i5-7400, 8G内存, 固态盘）

Ipfs add 文件名 #上传1.1G的文件用时15-20秒。

/ipfs-cluster-ctl pin add 哈希值1.1G, #这里开始是分发到群组里所有的节点机用时2分钟左右。（也就是所有群组内的节点机都会有相同的数据作为备份的用处。）

Ipfs add 文件名 #上传4.2G的文件用时1分20秒左右。

/ipfs-cluster-ctl pin add 哈希值4.2G, #这里开始是分发到群组里所有的节点机用时7分钟左右。（也就是所有群组内的节点机都会有相同的数据作为备份的用处。）

1. 清空某一台节点机存储内容试验：用命令rm -rf /data/ipfs/\* (主要是对blocks与datastore两个目录内容的删除清空)

清空群组某一台节点机存储内容后，重新启动ipfs进程后，存储内容与容量会从其中一台有数据的节点机自动同步到刚才清空后的节点机上。

1. （硬件平台i5-7400, 8G内存, HDD硬盘）与上面13写的SSD固态盘做对比。

Ipfs add 文件名 #上传1.1G的文件用时1分50秒-2分左右。

1. 配置服务器开机自启动环境，使可以使用下面的命令：

root@aips:/home/aipsb# systemctl restart ipfs

root@aips:/home/aipsb# systemctl restart ipfs-cluster

systemctl enable ipfs ; systemctl enable ipfs-cluster 这两个命令就是设置开机自启。

systemctl start ipfs ; systemctl start ipfs-cluster 这两个命令启动服务进程的，这样ps -ef|grep ipfs 就可以查到进程了。

systemctl stop ipfs ; systemctl stop ipfs-cluster 这两个命令停止服务进程的，ps -ef|grep ipfs 就查不到进程了，但系统重启后进程还会自启。

systemctl disable ipfs ; systemctl disable ipfs-cluster 这两个命令就是设置开机不要自启了，进程虽然还在，但系统重启后进程就不会再自启了。

上面的命令就是下面这些文件配置出来的，可以参考“IPFS设置开机自启动方法文档2.md”这个说明文档。

ls /usr/lib/systemd/system |grep ipfs #查到有下面这两个文件。

ipfs-cluster.service

ipfs.service