IPFS BLOX

Set-up and user guide v 1.0

Table of Contents

ABSTRACT

In this day and age, the internet has become such an important tool in our everyday lives. We use it to consume media, to communicate with friends and colleagues, to learn, to handle our finances, and much more; but the web as we know it, has a problem. Information on the web is centralized; stored in massive server farms which are usually controlled by big companies. Have you ever wondered if sites like Twitter, YouTube or Wikipedia go offline?

IPFS (InterPlanetary File System) is an open source, file sharing system that aims to make the web completely decentralized by running a peer-to-peer (P2P) file storage network that works similarly to how BitTorrent and Blockchain.The large centralized cloud storage providers such as Amazon, Google, Dropbox and many others control the large waves of the market. It's not just the cloud storage market specifically but the whole centralized web hosting industry and the clients' server infrastructure that's built on top of it.

With all these methods of file storage, you have one central point of failure.

Moreover, your files are technically the property of these providers while stored on their servers. Not to mention the rising cost of using these services.Data centralization brings another problem which is censorship. Because content is hosted in just a few servers, it's easy for governments to block access to them. In 2017, Turkey ordered internet providers to block access to Wikipedia because the administration called it a threat to national security.But why do we keep using this centralized web?One reason is because we have high expectations when it comes to the internet. We want pages, images and videos to load instantly and we want them in high quality. Centralizing servers allow companies to have complete control over how fast they can deliver their content.

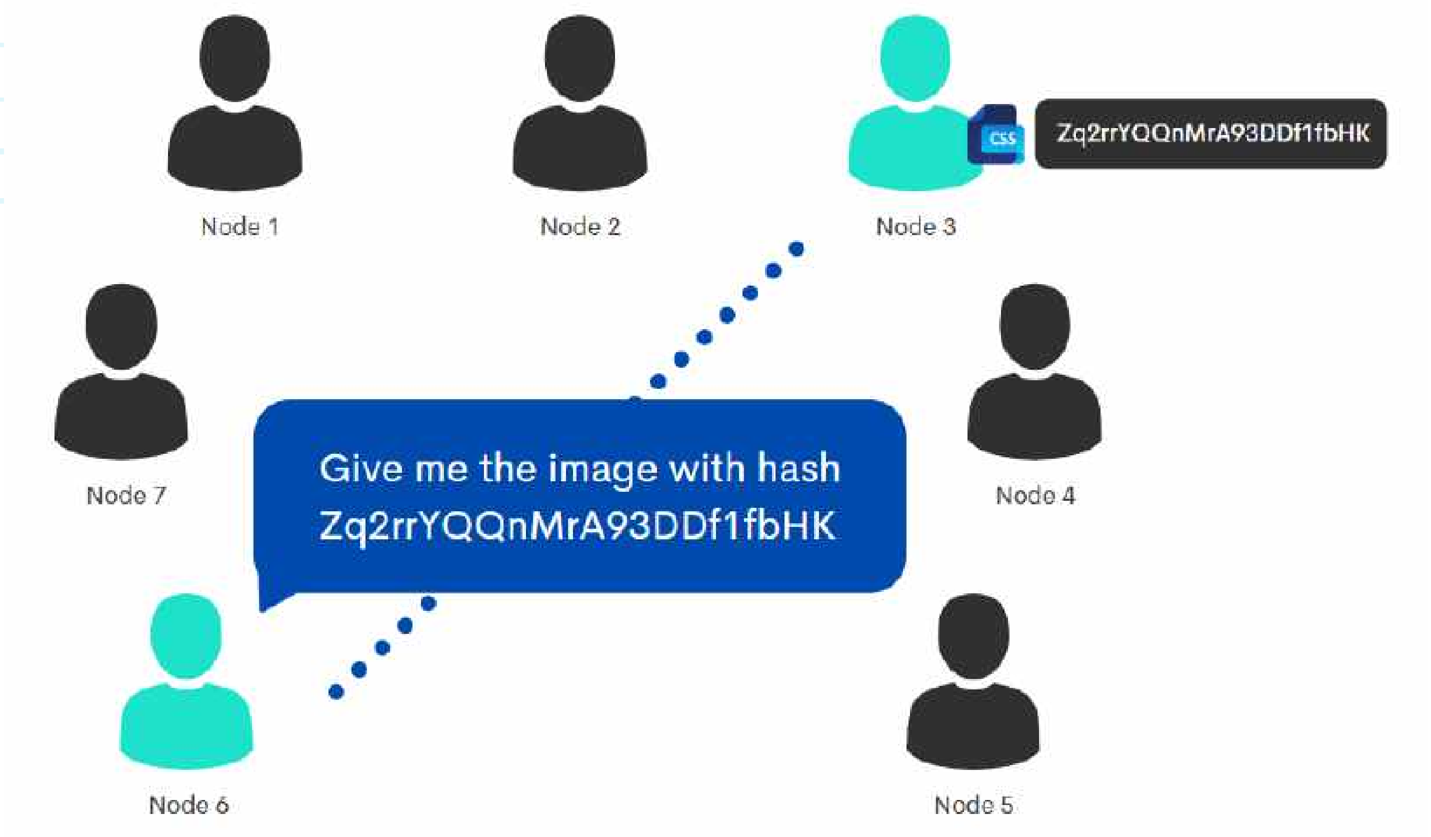
HOW DOES IT WORK?

Let us first understand how we access content on the web right now.

Suppose you want to download a photo from the internet. The location of the photo is the IP address or the domain name and this is called location-based addressing. When you download the photo, you tell the computer exactly where to find it; but if that location is not accessible—if the server is down for instance—you will not be able to get that photo.

When that happens, however, there is still a high chance that someone else has downloaded that photo before, and still has a copy of it; yet your computer won't be able to copy that photo from that person.

To fix this, IPFS moves from location addressing to content-based addressing. Instead of pointing "where" to find the resource, you simply state what it is you want. Every file on the web has a unique hash which can be compared to a fingerprint.



When you want to download a certain file, you just ask the network: "Who has the file with this hash?" and someone on the IPFS network will provide this to you.

BUILT-IN SECURITY FOR IPFS

IPFS uses hash functions to request a file, which means the architectural structure of IPFS is transparent and you can always verify what you have received. Another feature of using hash to address content integrity is deduplication. This happens when multiple people publish the same file on IPFS. To fix this, IPFS will create the file only once which makes the network quite efficient.

STORING AND SERVING FILES ON IPFS

Files are stored inside IPFS objects, and these objects can store up to 256 KB worth of data. They can also contain links to other IPFS objects. A simple “hello world” text file can be stored in a single IPFS object.

Files that are larger than 256 KB—like an image or a video for instance—are split into multiple IPFS objects that are all 256 KB in size. The system will then afterwards create an empty IPFS object that links to all the other pieces of the file.

The data architecture of IPFS is simple and yet it can be very powerful. This architecture allows us to really use it as a File System. Since IPFS uses content-based addressing, once something is added, it can no longer be changed. It is an immutable datastore much like the blockchain.

IPFS supports versioning of your files. If you are working on an important document that you want to share with everyone over IPFS, it will create a new commit object for you. This object is really basic. It simply tells IPFS which commit went before it and it links to the IPFS object of your file.

For instance, if you want to update this file, you can simply add your updated file to IPFS and the software will commit a new commit object for your file. This commit object now links to the previous commit. This process can be repeated endlessly. IPFS will make sure that your file and its entire history is accessible to other nodes on the network.

BASIC INSTALLATION USING AWS

Getting started with EC2

Create basic AWS EC2 with the following initial size

Hostname: IPFS
Instance Type: General Purpose t2.micro
vCPU: 1
Mem GiB4: 1
Storage: 30 GiB

Note: Global IP address will be randomly given by AWS when you allocate Elastic IP address. Only local IP addresses can be statically filled when creating the instance.

Connecting to Instance

Upload the private key file that you created when you launched the instance, (ex ipfs.pem). Change the key permission to ensure that it is not publicly viewable. If you are using Linux, use the ssh command to connect to the instance. For Windows, you can use Putty as client terminal to perform remote access to the server.

Note: Server Global IP address or Public DNS can be used to remotely access the server. Public DNS can be enabled or disabled under VPC Edit DNS hostname.

```
$ chmod 400 ipfs.pem
$ ssh –i ipfs.pem ubuntu@ec2-xx-xx-xx-xx.ap-southeast-1.compute.amazonaws.com
```

BASIC INSTALLATION USING AWS

▶ IPFS installation

1. After using the SSH command to the server, create a user for IPFS and log on. Upgrade the system using *apt-get update/upgrade* to apply the latest security updates and then install go language.

```
$ sudo su - root
$ useradd -s /bin/bash -m -d /home/ipfs -c “ipfs user”
ipfs
$ passwd ipfs
$ usermod —aG sudo ipfs
$ su — ipfs
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install golang-go -y
```

2. Download the latest version available for go-ipfs binary with wget; unpack it using tar -xvfz. Clean it up by removing the downloaded archive using rm. Move the executable file to the directory program with mv and then remove the other unpacked folder using the rm-rf command.

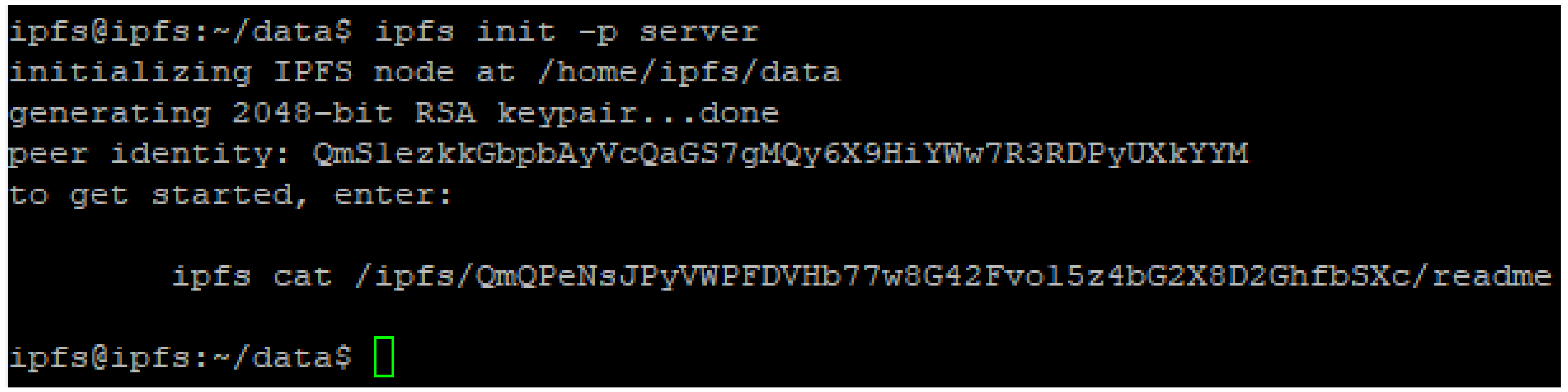
```
$ wget https://dist.ipfs.io/go-ipfs/v0.6.0/go-
ipfs_v0.6.0_linux-amd64.tar.gz
$ tar xvfz go-ipfs_v0.6.0_linux-amd64.tar.gz
$ rm go-ipfs_v0.6.0_linux-amd64.tar.gz
$ sudo mv go-ipfs/ipfs /usr/local/bin/ipfs
$ rm -rf go-ipfs
```

BASIC INSTALLATION USING AWS

▶ IPFS installation

3. Verify your installation using ipfs version command. Initialize the IPFS, and create a new directory (optional) to use for the repo. You can also use the user's home directory. Use mkdir to create a directory for repo, then add the repo path to .bash_profile script and source that can run the code contained in the file.

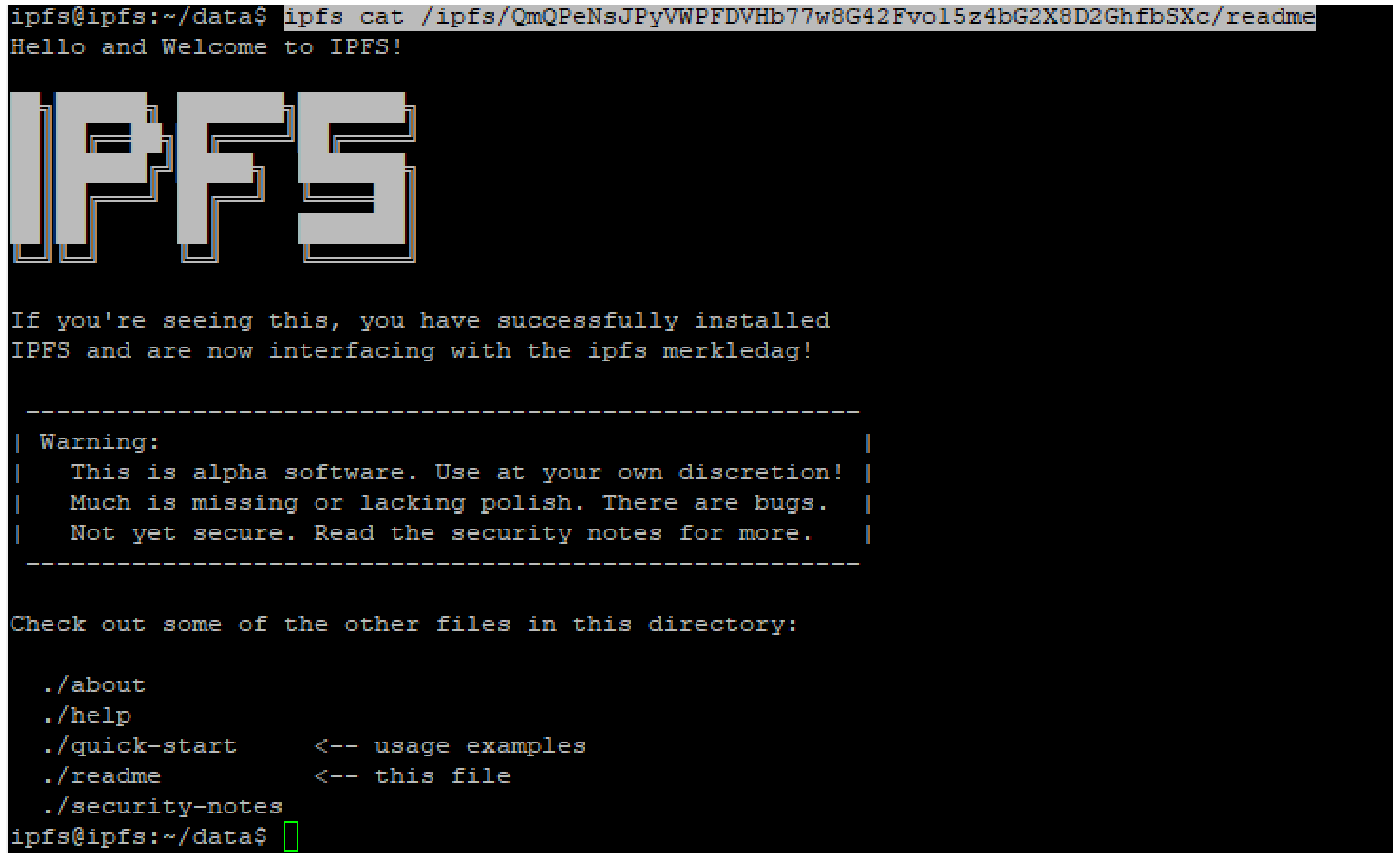
```
$ ipfs version
$ mkdir data
$ cd data
$ echo 'export IPFS_PATH=/home/ipfs/data' »~/.bash_profile
$ source ~/.bash_profile
$ ipfs init -p server
```



```
ipfs@ipfs:~/data$ ipfs init -p server
initializing IPFS node at /home/ipfs/data
generating 2048-bit RSA keypair...done
peer identity: QmSlezkkGbpbAyVcQaGS7gMQy6X9HiYWw7R3RDPyUXkYYM
to get started, enter:

        ipfs cat /ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc/readme

ipfs@ipfs:~/data$
```

4. Once you see the message "initialized successful", it will generate a local IPFS repository, as well as a cryptographic key pair that allows the IPFS node to cryptographically sign the content and messages that will be created later. To get started, after ipfs init command, IPFS provides a content that you can read and match the path you gave. You can view or read the content using the command ipfs cat.

```
$ ipfs cat
/ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc/readme
```

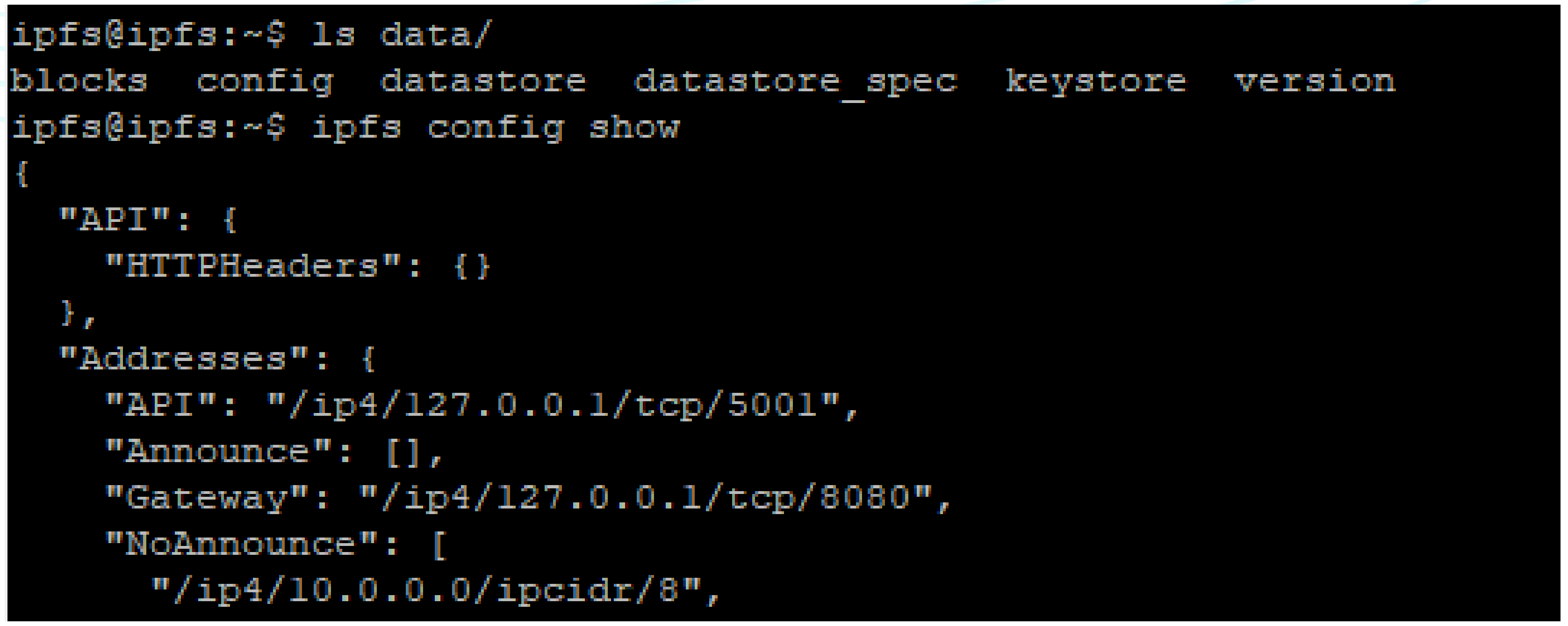


```
ipfs@ipfs:~/data$ ipfs cat /ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc/readme
Hello and Welcome to IPFS!

IPFS

If you're seeing this, you have successfully installed
IPFS and are now interfacing with the ipfs merkledag!

 -------------------------------------------------------
| Warning:                                              |
|   This is alpha software. Use at your own discretion! |
|   Much is missing or lacking polish. There are bugs.  |
|   Not yet secure. Read the security notes for more.   |
 -------------------------------------------------------

Check out some of the other files in this directory:

  ./about
  ./help
  ./quick-start     <-- usage examples
  ./readme          <-- this file
  ./security-notes
ipfs@ipfs:~/data$
```

BASIC INSTALLATION USING AWS

▶ IPFS installation

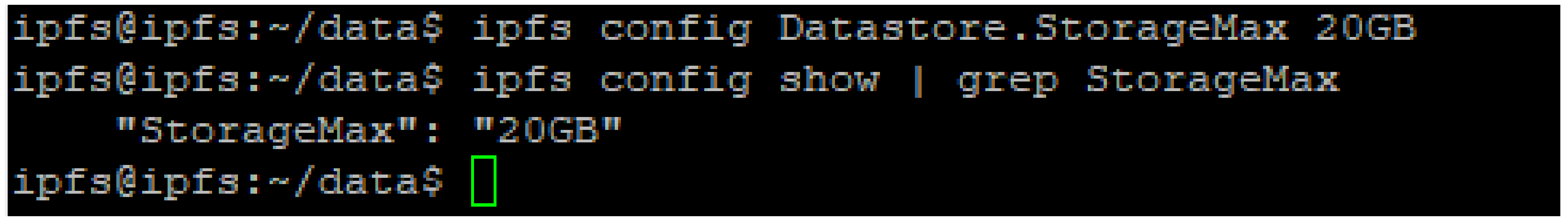
5. Locate where IPFS stores the repository content by using the ls command. All contents of the IPFS repository are stored in the directory that was created. The configuration for the IPFS repo is in *json* file format. You can view the current config using the *ipfs config* show command.



```
ipfs@ipfs:~$ ls data/
blocks  config  datastore  datastore_spec  keystore  version
ipfs@ipfs:~$ ipfs config show
{
  "API": {
    "HTTPHeaders": {}
  },
  "Addresses": {
    "API": "/ip4/127.0.0.1/tcp/5001",
    "Announce": [],
    "Gateway": "/ip4/127.0.0.1/tcp/8080",
    "NoAnnounce": [
      "/ip4/10.0.0.0/ipcidr/8",
```

6. To increase the storage capacity of the repository created, the config file can be edited and assigned a value for the Datastore.StorageMax using the command below:

```
$ ipfs config Datastore.StorageMax 20GB
$ ipfs config show | grep StorageMax
```

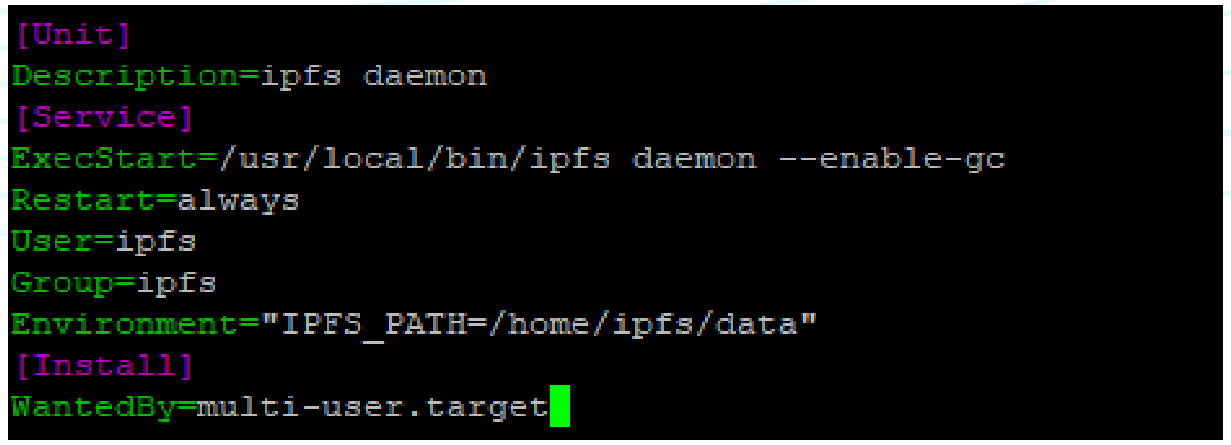


```
ipfs@ipfs:~/data$ ipfs config Datastore.StorageMax 20GB
ipfs@ipfs:~/data$ ipfs config show | grep StorageMax
    "StorageMax": "20GB"
ipfs@ipfs:~/data$ 
```

IPFS installation

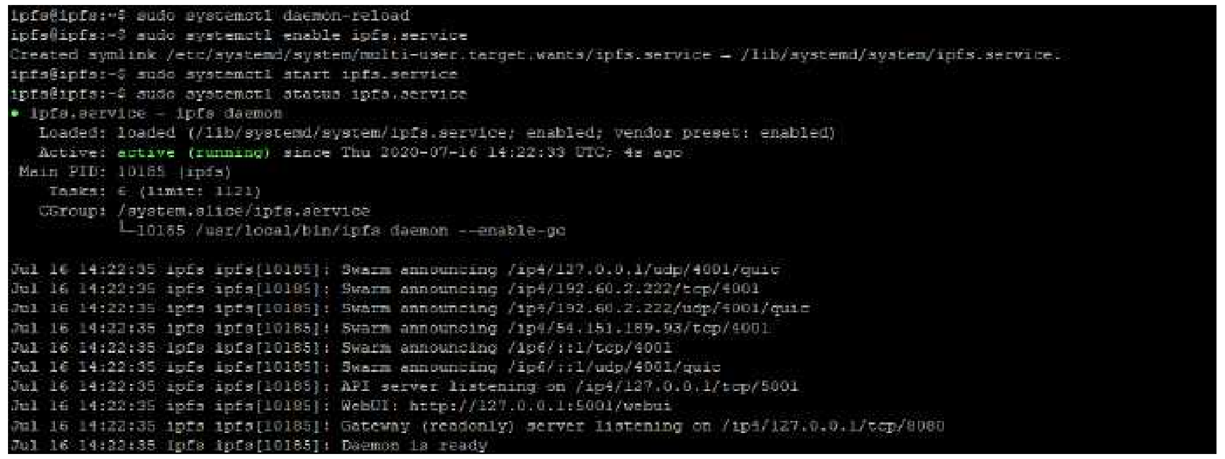
7. Create *systemctl* service to automatically start the IPFS daemon when the instance node gets restarted so you won't have to do it manually. You may then enable the new service and start it using the commands below:

```
$ sudo vim /lib/system/system/ipfs.service
```



```
[Unit]
Description=ipfs daemon
[Service]
ExecStart=/usr/local/bin/ipfs daemon --enable-gc
Restart=always
User=ipfs
Group=ipfs
Environment="IPFS_PATH=/home/ipfs/data"
[Install]
WantedBy=multi-user.target
```

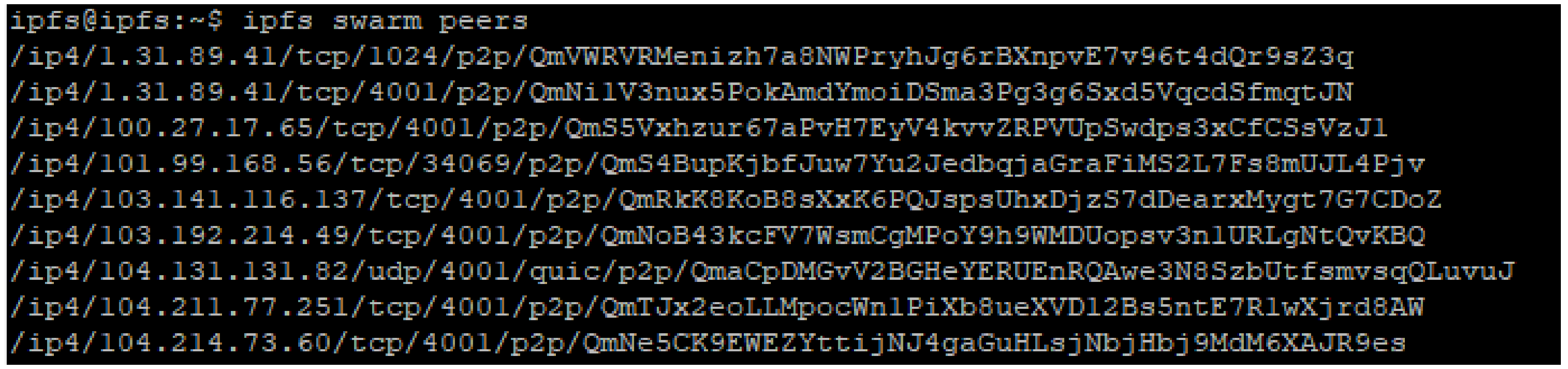
```
$ sudo systemctl daemon-reload
$ sudo systemctl enable ipfs.service
$ sudo systemctl start ipfs.service
$ sudo systemctl status ipfs.service
```



```
ipfs@ipfs:~$ sudo systemctl daemon-reload
ipfs@ipfs:~$ sudo systemctl enable ipfs.service
Created symlink /etc/systemd/system/multi-user.target.wants/ipfs.service → /lib/systemd/system/ipfs.service.
ipfs@ipfs:~$ sudo systemctl start ipfs.service
ipfs@ipfs:~$ sudo systemctl status ipfs.service
● ipfs.service - ipfs daemon
   Loaded: loaded (/lib/systemd/system/ipfs.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-07-16 14:22:33 UTC; 4s ago
 Main PID: 10185 (ipfs)
    Tasks: 6 (limit: 1121)
   CGroup: /system.slice/ipfs.service
           └─10185 /usr/local/bin/ipfs daemon --enable-gc

Jul 16 14:22:35 ipfs ipfs[10185]: Swarm announcing /ip4/127.0.0.1/udp/4001/quic
Jul 16 14:22:35 ipfs ipfs[10185]: Swarm announcing /ip4/192.60.2.222/tcp/4001
Jul 16 14:22:35 ipfs ipfs[10185]: Swarm announcing /ip4/192.60.2.222/udp/4001/quic
Jul 16 14:22:35 ipfs ipfs[10185]: Swarm announcing /ip4/54.151.189.93/tcp/4001
Jul 16 14:22:35 ipfs ipfs[10185]: Swarm announcing /ip6/::1/tcp/4001
Jul 16 14:22:35 ipfs ipfs[10185]: Swarm announcing /ip6/::1/udp/4001/quic
Jul 16 14:22:35 ipfs ipfs[10185]: API server listening on /ip4/127.0.0.1/tcp/5001
Jul 16 14:22:35 ipfs ipfs[10185]: WebUI: http://127.0.0.1:5001/webui
Jul 16 14:22:35 ipfs ipfs[10185]: Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Jul 16 14:22:35 ipfs ipfs[10185]: Daemon is ready
```

8. Once the service is started, you will see the output from the daemon. You will also see other IPFS peers connecting and routing through the newly configured node. Running the command ipfs swarm peers, will view the long list of connected peers to your node.

```
$ ipfs swarm peers
```



```
ipfs@ipfs:~$ ipfs swarm peers
/ip4/1.31.89.41/tcp/1024/p2p/QmVWRVRMenizh7a8NWPryhJg6rBXnpvE7v96t4dQr9sZ3q
/ip4/1.31.89.41/tcp/4001/p2p/QmNi1V3nux5PokAmdYmoiDSma3Pg3g6Sxd5VqcdSfmqtJN
/ip4/100.27.17.65/tcp/4001/p2p/QmS5Vxhzur67aPvH7EyV4kvvZRPVUpSwdps3xCfCSsVzJ1
/ip4/101.99.168.56/tcp/34069/p2p/QmS4BupKjbfJuw7Yu2JedbqjaGraFiMS2L7Fs8mUJL4Pjv
/ip4/103.141.116.137/tcp/4001/p2p/QmRkK8KoB8sXxK6PQJspsUhxDjzS7dDearxMygt7G7CDoZ
/ip4/103.192.214.49/tcp/4001/p2p/QmNoB43kcFV7WsmCgMPoY9h9WMDUopsv3n1URLgNtQvKBQ
/ip4/104.131.131.82/udp/4001/quic/p2p/QmaCpDMGvV2BGHeYERUEnRQAwe3N8SzbUtfsmvsqQLuvuJ
/ip4/104.211.77.251/tcp/4001/p2p/QmTJx2eoLLMpocWn1PiXb8ueXVD12Bs5ntE7R1wXjrd8AW
/ip4/104.214.73.60/tcp/4001/p2p/QmNe5CK9EWEZYttijNJ4gaGuHLsjNbjHbj9MdM6XAJR9es
```

BASIC INSTALLATION USING AWS

▶ **IPFS installation**

9. Open up the gateway to be able to browse its address into a browser, and view the document directory or anything that accessible on the decentralized web. Using the ipfs config Addresses.Gateway command will enable the public gateway and allow files to access it from your repository. You can be using either public DNS or the public IP address to browse the content of the repository. The hash will be the same with the hash that produce after the initialization of the repository.

Use the following URL:

For Linux:

http://ec2-xx-xx-xx-xx.apsoutheast1.compute.amazonaws.com:8080/ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc

http://xxx.xxx.xxx.xxx:8080/ipfs/QmQPeNsJPyVWPFDVHb77w8G42Fvo15z4bG2X8D2GhfbSXc

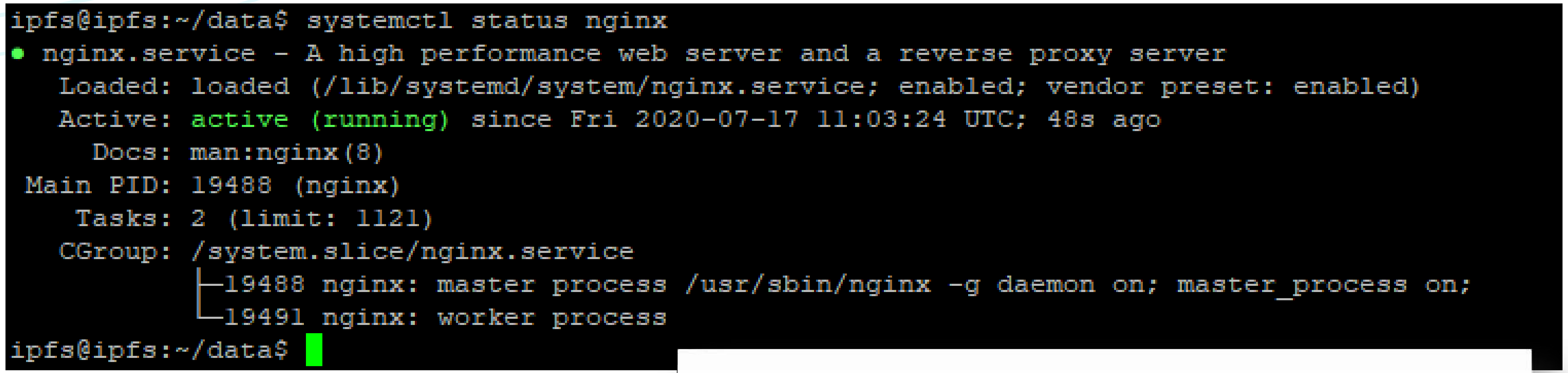
```
$ ipfs config Addresses.Gateway /ip4/0.0.0.0/tcp/8080
```

SETTING UP A REVERSE PROXY WITH NGINX

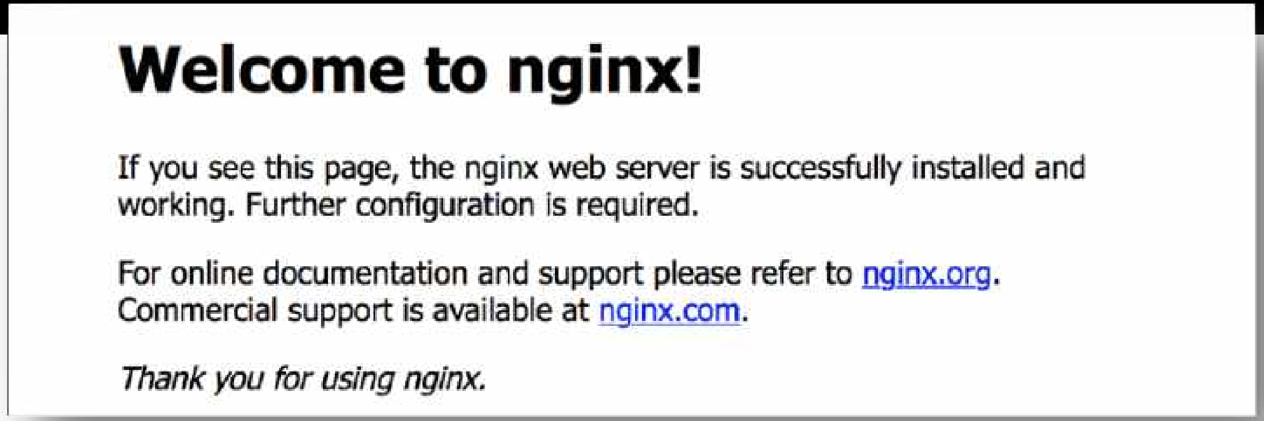
Using NGINX as a reverse proxy for secure web and to be able to run a secure gateway and connect using browser-based peers. We will be using secure protocols to ensure that the traffic between peers and browser-based peers are encrypted and private. The WebCrypto API used by js-ipfs in the browser requires a secure origin and can be achieve it by having secure gateway for accessing to it.

1. Setup the NGINX using our domain and enable the secure websocket connection for ipfs peer node server by reverse proxy. Install NGINX using command below and enter your instance Public DNS (IPV4) name into your favorite browser, you will see the NGINX default landing page.

```
$ sudo apt-get update
$ apt-get -y install nginx-extras
$ sudo apt-get install nginx -y
$ systemctl status nginx
```



```
ipfs@ipfs:~/data$ systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2020-07-17 11:03:24 UTC; 48s ago
     Docs: man:nginx(8)
 Main PID: 19488 (nginx)
    Tasks: 2 (limit: 1121)
   CGroup: /system.slice/nginx.service
           ├─19488 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;
           └─19491 nginx: worker process
ipfs@ipfs:~/data$
```



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

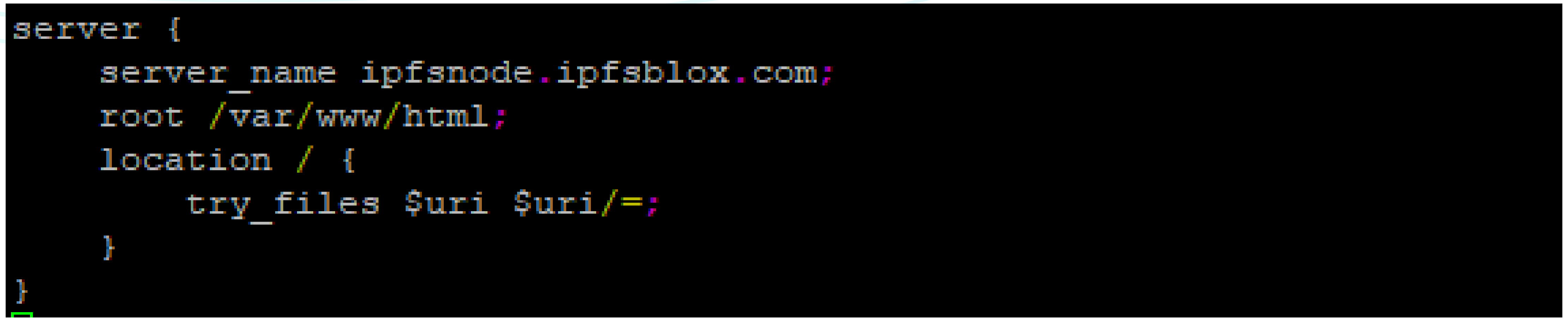
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

SETTING UP A REVERSE PROXY WITH NGINX

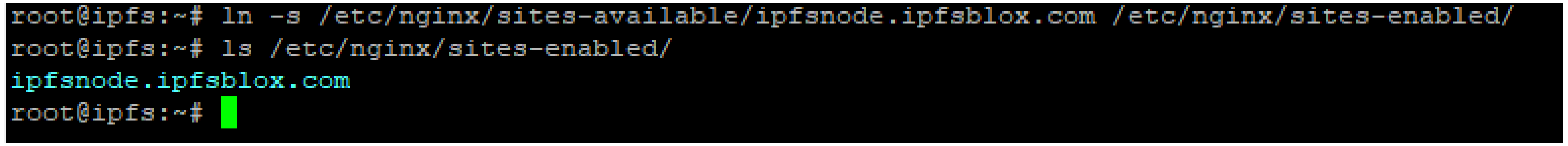
2. Setup domain or sub domain associate with your server public IP address then secure TLS connection that controls the instance of a browser-trusted certificate with SSL on it. We are going to customize the NGINX reverse proxy configuration to connect to our IPFS peer server addresses. Using vi/vim terminal editor, edit the file ipfsnode.ipfsblox.com as related to your sub domain under /etc/nginx/site-available directory then copy the server content below without the SSL certification part, SSL certificate can be install using Let's Encrypt or purchase to any SSL selling company. Link the configuration file to site-enabled using ln -s command.

```
$ vim /etc/nginx/sites-available/ipfsnode.ipfsblox.com
```



```
server {
    server_name ipfsnode.ipfsblox.com;
    root /var/www/html;
    location / {
        try_files $uri $uri/=;
    }
}
```

```
$ ln -s /etc/nginx/sites-available/ipfsnode.ipfsblox.com /etc/nginx/sites-enabled/ $ sudo systemctl restart nginx
```



```
root@ipfs:~# ln -s /etc/nginx/sites-available/ipfsnode.ipfsblox.com /etc/nginx/sites-enabled/
root@ipfs:~# ls /etc/nginx/sites-enabled/
ipfsnode.ipfsblox.com
root@ipfs:~#
```

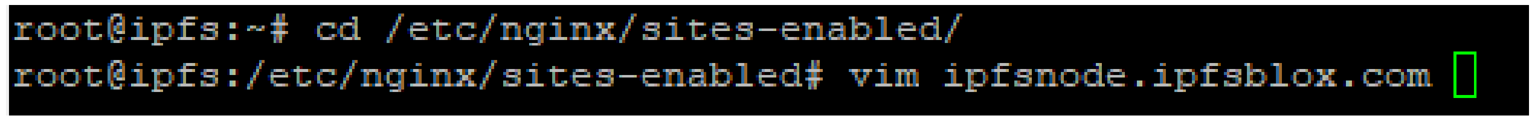
SETTING UP A REVERSE PROXY WITH NGINX

3. The initial configuration set up the welcome page; we will add the entry point of our gateway by changing the location section of the file with configuration below.

```
$ ln -s /etc/nginx/sites-available/ipfsnode.ipfsblox.com /etc/nginx/sites-enabled/ $ sudo systemctl restart nginx
```

4. We will use Let's Encrypt for free and temporary SSL to have secure TLS connections or encrypted between the server and client when browsing our domain, we need to enable HTTPS on our NGINX server. You can search for the Let's Encrypt installation and how to register domain on the web, there are bunch of tutorial on how to setup. But for now let's add the SSL generated by Let's Encrypt to our server by using the command below.

```
$ cd /etc/nginx/sites-enabled/
$ vim ipfsnode.ipfsblox.com
```



```
root@ipfs:~# cd /etc/nginx/sites-enabled/
root@ipfs:/etc/nginx/sites-enabled# vim ipfsnode.ipfsblox.com
```

```
listen [::]:443 ssl ipv6only=on;
  listen 443 ssl; # managed by Certbot
  ssl_certificate
/etc/letsencrypt/live/ipfsnode.ipfsblox.com/fullchain.pem;
  ssl_certificate_key
/etc/letsencrypt/live/ipfsnode.ipfsblox.com/privkey.pem;
   include /etc/letsencrypt/options-ssl-nginx.conf;
   ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
                      }


               server {if ($host = ipfsnode.ipfsblox.com) {
                      return 301 https://$host$request_uri;
                     }
                      server_name ipfsnode.ipfsblox.com;
                  listen 80;
                listen [::]:80 ;
                     return 404;
```

Note: The above config will be generated by let's encrypt when you register the domain and obtain a certificate, then all the traffic should be redirect to HTTPS. To enable the redirecting you should select the option 2 while installing it.

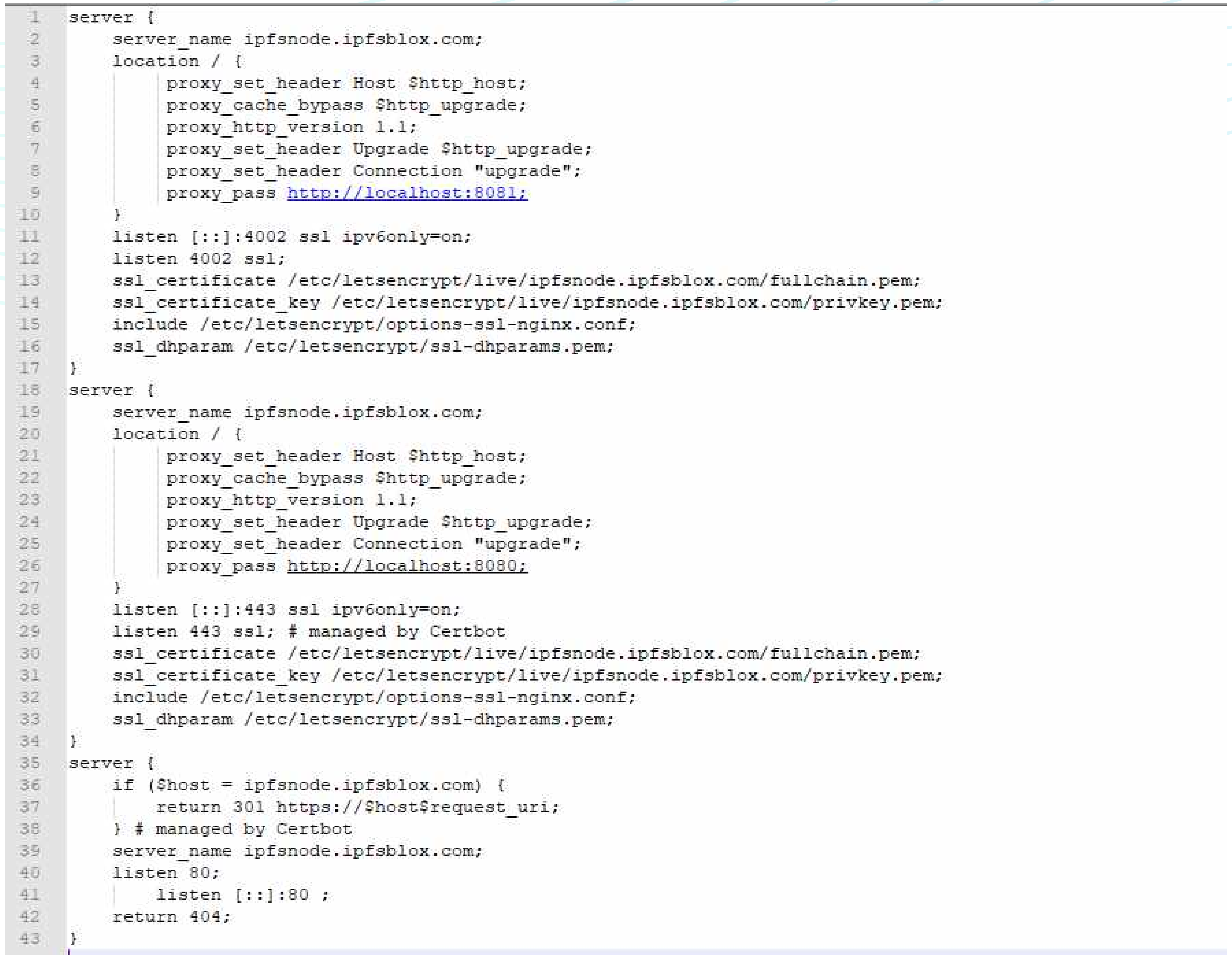
SETTING UP A REVERSE PROXY WITH NGINX

5. With the P2P websocket connection to redirecting secure connection over port 4002 to port 8081, we will add the following configuration below. This includes also the SSL certification location generated by Let's Encrypt.

```
server {

server_name ipfsnode.ipfsblox.com;
location / {
        proxy_set_header Host $http_host;
        proxy_cache_bypass $http_upgrade;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_pass http://localhost:8081;
}



listen [::]:4002 ssl ipv6only=on;
listen 4002 ssl;
ssl_certificate
/etc/letsencrypt/live/ipfsnode.ipfsblox.com/fullchain.pem;
   ssl_certificate_key
/etc/letsencrypt/live/ipfsnode.ipfsblox.com/privkey.pem;
   include /etc/letsencrypt/options-ssl-nginx.conf;
   ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
            }
```

6. For the complete setup of the NGINX, please see below configuration. After saving it, you can check the NGINX if it is correct by using command nginx -t and then sudo systemctl restart nginx to apply the settings.

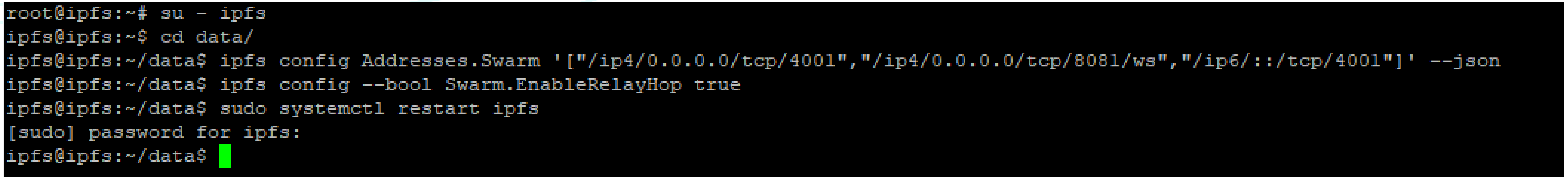
```
$ nginx –t
$ sudo systemctl restart nginx
```



```
server {
    server_name ipfsnode.ipfsblox.com;
    location / {
         proxy_set_header Host $http_host;
         proxy_cache_bypass $http_upgrade;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection "upgrade";
         proxy_pass http://localhost:8081;
    }
    listen [::]:4002 ssl ipv6only=on;
    listen 4002 ssl;
    ssl_certificate /etc/letsencrypt/live/ipfsnode.ipfsblox.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ipfsnode.ipfsblox.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
server {
    server_name ipfsnode.ipfsblox.com;
    location / {
         proxy_set_header Host $http_host;
         proxy_cache_bypass $http_upgrade;
         proxy_http_version 1.1;
         proxy_set_header Upgrade $http_upgrade;
         proxy_set_header Connection "upgrade";
         proxy_pass http://localhost:8080;
    }
    listen [::]:443 ssl ipv6only=on;
    listen 443 ssl; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/ipfsnode.ipfsblox.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/ipfsnode.ipfsblox.com/privkey.pem;
    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;
}
server {
    if ($host = ipfsnode.ipfsblox.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot
    server_name ipfsnode.ipfsblox.com;
    listen 80;
        listen [::]:80 ;
    return 404;
}
```

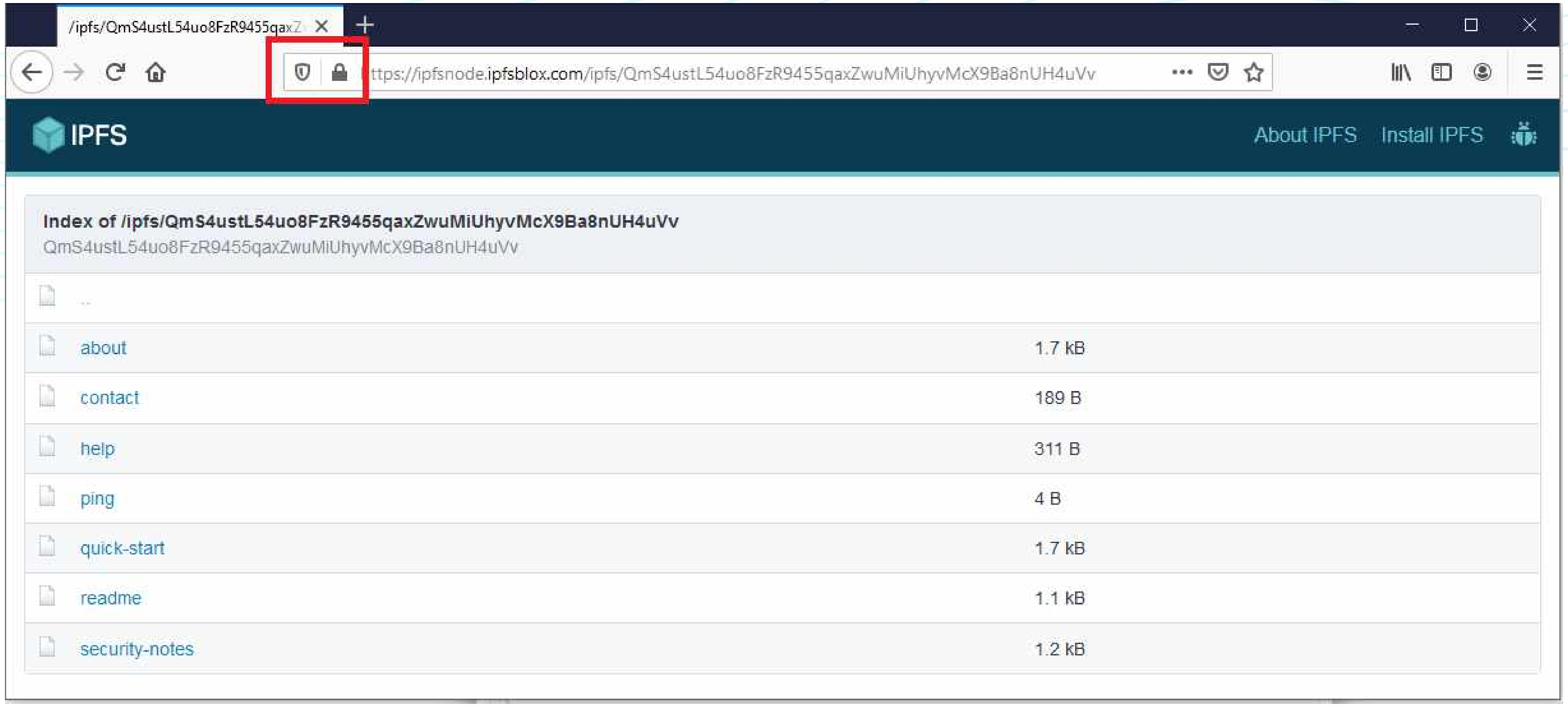
7. Enable the IPFS features to connect into the browser-based peer node by enabling websocket support and relay hopping Swarm.EnableRelayHop to our IPFS node configuration. After adding the configuration, use the command sudo systemctl restart ipfs to apply the changes.

```
$ su – ipfs
$ cd data/
$ ipfs config Addresses.Swarm '[“/ip4/0.0.0.0/tcp/4001”,“/ip4/0.0.0.0/tcp/8081/ws”,“/ip6/::/tcp/4001”]' –json
$ ipfs config –bool Swarm.EnableRelayHop true
$ sudo systemctl restart ipfs
```



```
root@ipfs:~# su - ipfs
ipfs@ipfs:~$ cd data/
ipfs@ipfs:~/data$ ipfs config Addresses.Swarm '["/ip4/0.0.0.0/tcp/4001","/ip4/0.0.0.0/tcp/8081/ws","/ip6/::/tcp/4001"]' --json
ipfs@ipfs:~/data$ ipfs config --bool Swarm.EnableRelayHop true
ipfs@ipfs:~/data$ sudo systemctl restart ipfs
[sudo] password for ipfs:
ipfs@ipfs:~/data$
```

8. Let's test the secure gateway, browse your domain together with the hash that has been generated after the installation of the IPFS. You can check it by running your IPFS node redirecting to HTTPS when browsing it to your favorite browser, you can see the Lock icon that indicates it was secured.


/ipfs/QmS4ustL54uo8FzR9455qaxZ
ttps://ipfsnode.ipfsblox.com/ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv
IPFS
About IPFS
Install IPFS

Index of /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv
QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8nUH4uVv

..	
about	1.7 kB
contact	189 B
help	311 B
ping	4 B
quick-start	1.7 kB
readme	1.1 kB
security-notes	1.2 kB

Thank you.

https://wiki.ipfsjapan.org

END