IPFS BLOX

Set-up and user guide v 1.0

目次

はじめに

ABSTRACT

昨今ではインターネットは毎日の生活において非常に重要なツールになっています。私達は情報を閲覧し、そして友達や同僚とコミュニケーションを取ったり、ファイナンスに関する情報を得たり、またそれに申し込んだりする、、などなど。しかし、WEBは便利だけど皆さんがご存知の通り限界があります。WEBというメディアは中央集権的（一つの大きな機関によってコントロールされている）な配信方法です。すべての情報は管理されたサーバーの中に存在しており、そして多くのWEBサイトは大企業によって運営されています。貴方はツイッターとかYoutueもしくはWikipediaなどのサーバーがダウンしてしまうとどうなるか考えた事があるでしょうか?

IPFS (InterPlanetary File System) はオープンソースのファイル共有システムであり、非中央集権的なしくみを持った、ピアツーピア（P2P）のファイルストレージです。それはBitTorrentやBlockchainと同様の概念で構築されています。

巨大で中央集権的なクラウドストレージはAmazonやGoogle、Dropboxそしてその他の多くの大企業で提供されており現在は大きなトレンドになっています。しかし、それは単なるストレージというだけでなく、WEBサーバーを始めとしたWEB関連のすべてのインフラがそのプラットフォームの上に構築されています。それは構築されたWEBシステムは一つの単一障害地点（one central point of failure）を抱えていると言っても過言ではありません。

その上、貴方がアップしたファイルはテクニカルな表現をすれば、これら大企業が保有する資産の上に存在しており、貴方が希望するしないに関わらず支払いコストがアップしている事もあります。

また、中央集権的なしくみは「検閲」という別の問題を抱えてしまいます。WEBで配信されるファイルは通常はわずか数台のサーバーから配信されているので、政府はたやすくその配信元のサーバーからの配信をストップさせる事が可能です。実際に2017年にはトルコ政府はインターネットプロバイダーに国家安全保障上の問題として、Wikipediaへのアクセス遮断を要請しました。

ではどうしてWEBはこの中央集権的な仕組みが維持されているのでしょうか?

一つの理由として我々はインターネットにあまりにも多くの期待をしている事です。我々はページを表示するだけでなく画像やビデオなどを含めたコンテンツが素早く表示する事を期待しますし、そのコンテンツのクオリティも高い事を要求します。そして、中央集権的なWEBサーバーの仕組みはコンテンツ提供企業に対して素早い配信を完全なコントロールの元に管理して推進する事ができるからです。

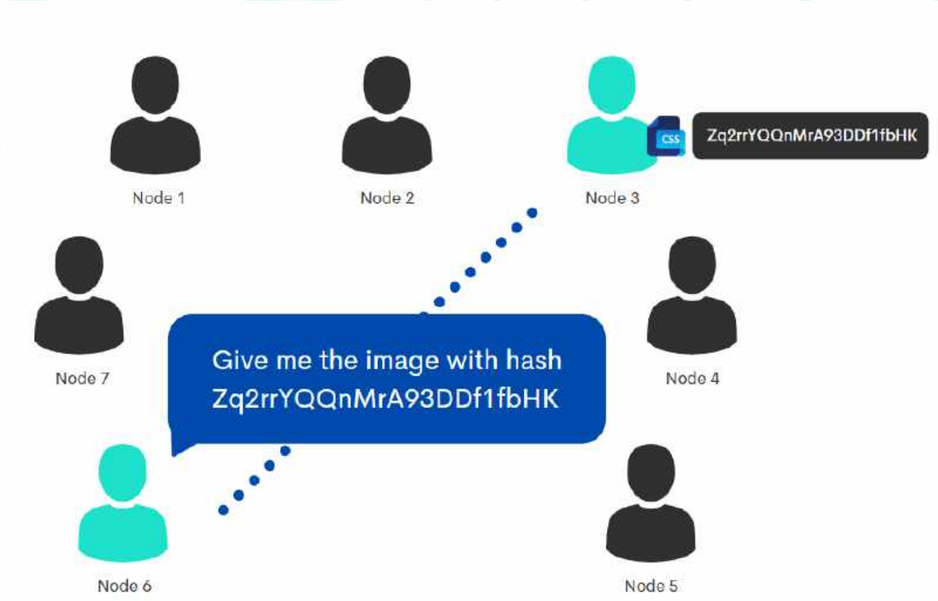
IPFS とはどうのように動くのでしょうか?

まず最初に私達がどのようにWEBにアクセスしているかを見ていきましょう。

もし貴方が一つの写真をインターネットを通じてダウンロードしたいとします。その写真が設定している場所はIPアドレスもしくはドメインネーム（URL）と言った物理的なロケーションベースのアドレスを使って要求をします。そして貴方が目的の写真をダウンロードする時には正確にどの場所（アドレス）に行けばそのコンテンツがあるかを指定する必要があります。しかし、このロケーションという概念では実際にコンテンツが保管されているサーバーがダウンしていると貴方は目的の写真をダウンロードする事はできません。

もし、このようなサーバーダウンが発生した場合にでも、誰が別の人が運良くダウンする前に貴方が必要とする同じ写真をダウンロードして保管している可能性はあります。しかしながら、貴方がそのコピーをその人からもらえるチャンスは、その人がよほど仲の良い友人でもない限り、恐らくないでしょう。

この問題を解消するには、IPFSはロケーションアドレスといういわゆるIPアドレスを主体としたロケーションアドレシングからコンテンツベースアドレシングとい概念に変更する事です。すべてのファイルはユニークなハッシュ値を持っており、そのハッシュキーを元に目的のファイルを探し出します。　このハッシュキーとはファイル単位でユニークであり、ある人間の識別するのにその人特有の指紋を使うのと概念が似ています。そして貴方が目的のファイルを手に入れたければ、「誰かこのハッシュキー（識別子）によって紐付けられているファイルを持ってない? 」と聞くことになります。そうすればIPFSネットワークに参加しているサーバー間で通信が行われ「誰かが持っている目的のファイル」を貴方に届けてくれる事になります。



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.

IPFSの中に組み込まれたセキュリティ

IPFSはハッシュ関数から得るハッシュ値をファイルのリクエストとして使います。それは設計上IPFSの挙動を非常に透明性のあるものとしており、貴方はいつでも取得したファイルが本物である事を確認できます。そしてこのハッシュ値をアドレスとして利用するという方法はデュプリケーション機能を内在させている事になります。もし誰かが全く同じファイルをIPFSにアップしようとしてもハッシュ値が同じなので、何度もそのファイルをIPFSノードにアップさせる事はなく、既存に登録されている唯一のファイルを持ってそれを利用します。そうする事でネットワークの効率的な利用を促進します。

IPFSにおけるファイルのアップと取得

ファイルはIPFSオブジェクトとして保管され、それらのオブジェクトは最大256KBまで単一のファイルとして保管されます。また、関連ファイルへのLinkなども他のIPFSオブジェクトとして保管されます。最も単純な“hello world”のようなテキストファイルはたった一つのIPFSオブジェクトとして保存されます。

画像やビデオのように256KBを超えるファイルは分割されて別々のIPFSオブジェクトとして保管されます。そして、それぞれの分割された保管されるファイルへのLinkがIPFSオブジェクトして埋め込まれます。

このIPFSの設計思想は非常にシンプルながら非常に強力なものです。この仕組は実際のファイルシステムとして利用ができます。IPFSはコンテンツベース・アドレス概念を採用している事から、そのファイルが一旦アップされるとそのファイルに対して何か追記をすると言うことができません。これは一度トランザクションが成立すると後日書き換えが不可能であるBlockchainによく似ています。

一方でIPFSはファイルに対するバージョン管理機能をサポートしています。もし貴方が非常に重要な書類を作成しており、その書類を多くの人と共有したいとします。そしてそのファイルに修正や加筆をした場合に、過去において存在したファイルへのリンクをオブジェクトとして埋め込む事ができます。

例えば、貴方が何かファイルに対して修正を加えると、修正をした後のファイルがIPFSネットワークにアップされる一方で過去に存在していたファイルへのリンクも保存されます。そしてそのような修正作業を行う度にこのプロセスが繰り返されます。そのようにしてIPFSはそのリンクをたどることで過去の修正履歴を追うことができます。

AWSを使った基本的なIPFSのインストール手順

はじめにEC2をセットアップ

基本的なAWS EC2の初期化手順

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

Note: グローバルIPアドレスはAWSからランダムに割当されます。そしてローカルIPアドレスだけこのインスタンス生成手順において静的に自分で割当が可能です。

インスタンスへのコネクション

インスタンスを生成するにあたり、プライベートキーファイルをアップロードします。（例えば、ipfs.pem）そしてキーのパーミッションを変更して、外部の人からは閲覧できない状態にします。もし貴方がLinuxを利用しているのであればsshコマンドを使ってインスタンスへのコネクションを成立させます。もしWindowsを使っているのであればPuttyなど通信が暗号化されたターミナルクライアントアプリを使ってリモートアクセスを行います。

Note: グローバルIPアドレスもしくはパブリックDNSをリモートアクセスに利用できます。パブリックDNSはVPCによるHostnameを編集する事で利用することもしないようにする事もできます。

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

IPFSのインストール

IPFS installation

1. SSHのコマンドを使ってサーバーへアクセスした後にIPFSのユーザーアカウントを作成します。 apt-get update/upgrade を使いセキュリティを最新の状態にした後に、GO言語をインストールします。

```
$ 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. 次に最新バージョンのgo-ipfs のバイナリーを入手してパッケージをtar -xvfz　を使って解凍します。その後、rmを使ってダウンロードしたアーカイブを綺麗に削除します。実行可能なファイルをプログラムディレクトリにmvを使って移動させ、その他の解凍前のフォルダーをrm-rfコマンドを利用して削除します。

```
$ 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. ipfsのバージョンコマンドを使ってベリファイします。そして、repoの為の新しいディレクトリを作成します。また、ユーザーのHomeディレクトリを使う事も可能です。mkdir コマンドを使ってrepoの為の新しくディレクトリを作成します。その後、repoのパスを.bash_profileに通します。スクリプトとソースをこの中で展開します。。

```
$ 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. “initialized successful”のメッセージがでると、システムはIPFSのローカルリポジトリィを作成します。　と同時にIPFSノードへ暗号化コンテンツとメッセージを送信する為の暗号キーを後で作成します。ipfs init コマンドでスタートさせた後はIPFSは貴方が読み出ししたいコンテンツの提供を行います。ipfs cat　コマンドによって、目的とするコンテンツの閲覧を可能とします。

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

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

IPFS installation

5. lsコマンドを使ってIPFSのリポジトリィコンテンツを表示します。ディレクトリに生成されたすべてのIPFSリポジトリィにあるコンテンツを表示します。IPFS repoのコンフィギュレーションは jsonファイルフォーマットで提供されます。貴方は現在のコンフィギュレーションの状態をipfs config showを使って表示する事ができます。

```
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. リポジトリィを生成するディスク容量を増やす為にはconfigファイルを編集して、割当するスペースDatastore.StorageMaxの値を変更する事で調整します。利用するコマンドは下記の通りです。

```
$ 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$
```

BASIC INSTALLATION USING AWS

IPFS installation

7. /system/ipfs.serviceを作成する事で自動的にIPFSデーモンを起動させるように設定する事でシステムが再起動しても毎回手動でデーモンを立ち上げる必要をなくします。　下記のコマンドを使って新しいサービスのエントリーを追加できます。

```
$ 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. 一度サービスが起動すればデーモンからのメッセージを確認します。また、他にもIPFSのpeersコネクションやルーティングなどを新しく設定されたノードを通じて確認できます。ipfs swarm peersコマンドを実行する事でコネクションが成立したpeersが貴方のノードで到達している状況を見ることが可能です。

```
$ 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/QmNilV3nux5PokAmdYmoiDSma3Pg3g6Sxd5VqcdSfmqtJN
/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. ブラウザーにURLを入れる事で表示できるようにするゲートウエイを設定します。これにより、ブラウザによってドキュメントやディレクトリそしてあらゆるコンテンツにアクセスできるようになります。ipfs config Addresses　を使うことでゲートウエイコマンドはパブリックゲートウエイとしても機能するようになり、同時にパブリックゲートウエイを通じて貴方のローカルリポジトリィへのアクセスも可能になります。貴方は公開されたDNSのURLアドレスもしくはIPアドレスを使いコンテンツのリポジトリィを参照できるようになります。リポジトリィの初期化をしてもはじめに生成されたハッシュがそのまま利用される事になります。

下記のようなURLを利用します。:

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
```

NGINXを使ったリバースプロキシ-のセットアップ

NGINXをリバースプロキシとして利用する事でWEBブラウザを使ったセキュアなpeersへのアクセスとゲートウエーの提供を可能とします。　我々は実際のpeersとブラウザーベースのpeersの間にセキュアプロトコルを導入する事でその間のプライベート通信も暗号化されます。js-ipfsによるWebCrypto APIは通信発生元に対してセキュアゲートウエイを使ってセキュア通信を要求します

1. リバースプロキシによってIPFSノードサーバーへのセキュアwebsocket通信をできるようにNGINXをセットアップします。下記のコマンドを使ってNGINXをインストールして、そのインスタンスのパブリックDNS（IPV4）名をブラウザーに入力して表示する事でNGINXの初期ランディングページが表示します。

```
$ 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$
```

SETTING UP A REVERSE PROXY WITH NGINX

2. 貴方が利用しているサーバーのパブリックIPアドレスに紐づくドメイン及びサブドメインをセットアップする事で、ブラウザーが信頼するSSLの証明書をコントロールするセキュアTLSが利用できるようになります。その後、NGINXのリバースプロキシをカスタマイズする事でIPFSのpeersサーバーのアドレスへこれがアクセスできるようにします。vi/vimターミナルを使いipfsnode.ipfsblox.comファイルを編集して、/etc/nginx/site-available directoryに関連付けを行いサーバーのコンテンツをSSL証明証がないパートにコピーをします。SSL証明書はLet's Encryptを使うもしくはSSLキーを販売している会社から購入する事ができます。コンフィギュレーションファイルをln -sコマンドを使ってsite-enabledにリンク付けさせます。

```
$ 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. welcome pageへの初期コンフィギュレーションセットアップ、下記のようなコンフィギュレーションファイルのローカルセクションにゲートウエイサーバーのエンドポイントを追加する事で設定をを行います。

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

4. 今回は無料のLet's Encryptと一時的なSSLを使い、サーバーとクライアント間でのセキュアTLSコネクションを行えるようにします。そこで、NGINXサーバーのHTTPSの状態を利用可能とする必要があります。Let's Encryptのインストール方法はネット上に多数掲載されているので詳細はそれを参照していただくとして、今は下記のコマンドを使って、Let's Encryptによって生成されたSSLをインストールする手順を紹介します。

```
$ 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;
```

上記のコンフィギュレーションはlet's encryptを使って登録されたドメインに対して証明書を発行しています。そしてすべてのアクセスはHTTPSへリダイレクトされる必要があります。このリダイレクトを可能とする為にはインストールする際にオプション２を選択する必要があります。

SETTING UP A REVERSE PROXY WITH NGINX

5. P2PのwebsocketをポートをPort4002から8081へリダイレクトの設定をする事でセキュア通信の設定をします。下記のコンフィギュレーションを追加します。これはLet's EncryptによるSSLコンフィギュレーションも含んでいます。

```
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. NGINXのセットアップは完了させるには下記のコンフィギュレーションを見てください。すべての変更を保存した後にはnginx -t コマンドを使ってNGINXの設定が正しくされているかを確認の上、sudo systemctl restart nginxにてセッチングの反映を行ってください。

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

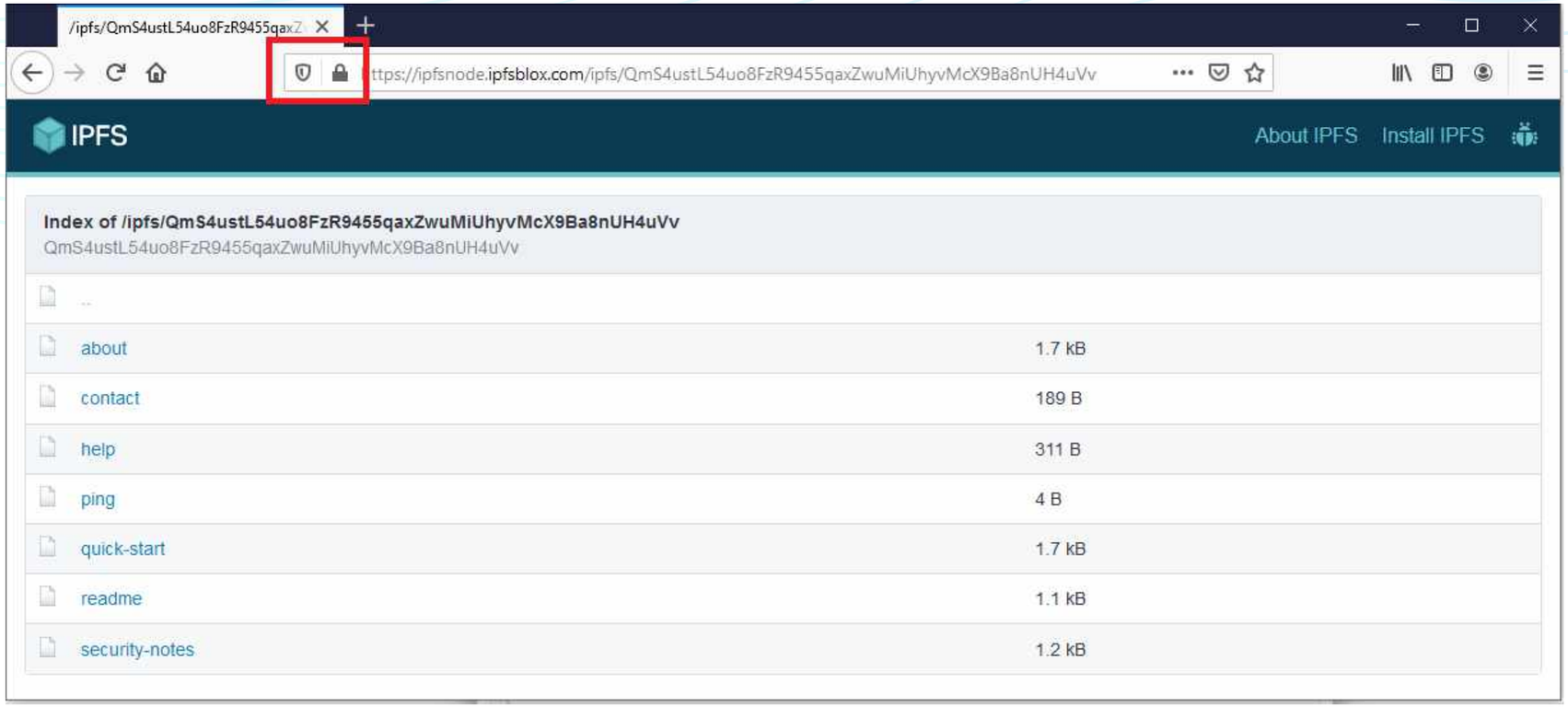
```
server {
    server_name ipfsnode.ipfsblox.com;
    location / {
        proxy_set_header Host $http_host;
        proxy_cache_bypass $http_upgrade;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection "upgrade";
        proxy_pass http://localhost:8081;
    }
    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. IPFSフューチュアーがwebsocketサポートと有効にする事で、ブラウザベースのpeersノードにコネクションできるようにします。そして、Swarm.EnableRelayHopへのリレイができるようにします。この修正を行った後に、sudo systemctl restart ipfsコマンドを使って修正の反映をさせます。

```
$ 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. それではテストをしてみます。IPFSをインストールした時に生成したハッシュを登録されたドメインURLを使ってブラウザで実際にアクセスをしてみます。IPFSノードへのアクセスがHTTPSに正しくリダイレクトされれいる事を確認してください。ブラウザにLock Iconが表示されていればそれはセキュアな通信が実行されています。



ありがとうございました

https://wiki.ipfsjapan.org

END