# **Creating Generative Art NFTs from Genomic Data**

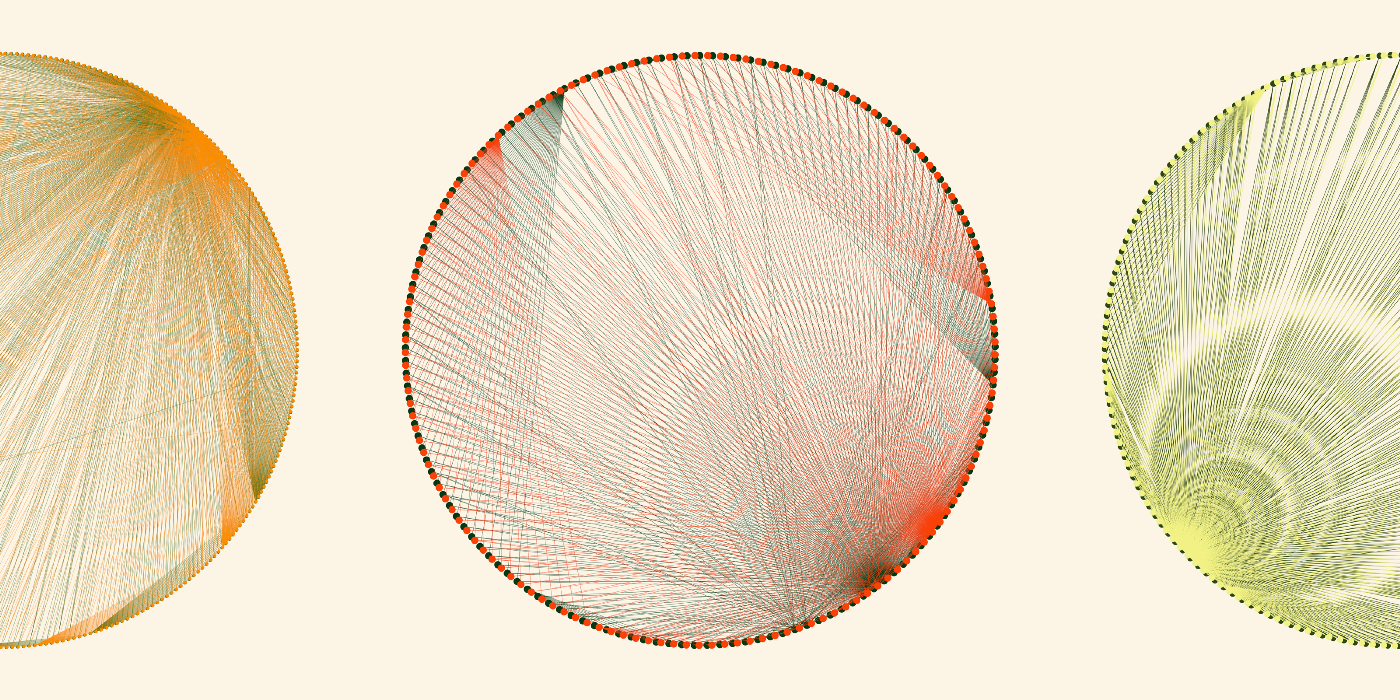
****

Image by author

In this post I outline my journey creating a dynamic NFT on the Ethereum blockchain with IPFS and discuss the possible use cases for scientific data. I do not cover algorithmic generation of static images (you should read Albert Sanchez Lafuente’s neat [step-by-step](https://towardsdatascience.com/algorithmically-generated-image-dataset-71aee957563a) for that) but instead demonstrate how I used Cytoscape.js, Anime.js and genomic feature data to dynamically generate visualizations/art at run time when NFTs are viewed from a browser. I will also not be providing an overview of Blockchain but I highly recommend reading Yifei Huang’s recent post: [Why every data scientist should pay attention to crypto](https://towardsdatascience.com/why-every-data-scientist-should-pay-attention-to-crypto-39b4c25ff319).

While stuck home during the pandemic, I’m one of the [10 million](https://www.nbcdfw.com/news/local/pandemic-gardening-hobby-still-taking-root/2737970/) that tried my hand at gardening on our little apartment balcony in Brooklyn. The Japanese cucumbers were a hit with our neighbors and the tomatoes were a hit with the squirrels but it was the peppers I enjoyed watching grow the most. This is what set the objective for my first NFT: create a depiction of a pepper that ripens over time.

How much of the depiction is visualization and how much is art? Well that’s in the eye of the beholder. When you spend your days scrutinizing data points, worshiping best practices and optimizing everything from memory usage to lunch orders it’s nice to take some artistic license and make something just because you like it, which is exactly what I’ve done here. The depiction is authentically generated from genomic data features but obviously this should not be viewed as any kind of serious biological analysis.

If you’re looking for the final live result you can view it [here](https://opensea.io/assets/0x86944176301532d4432b50838cea252e3b2990bb/1001) and the source on GitHub [here](https://github.com/s1monj/nft-peppers).



## **Preparing the Genomic Data**

There’s nothing new here but I’m going to zip through it for completeness. My first stop was to visit the NCBI [assemblies](https://www.ncbi.nlm.nih.gov/assembly/?term=Capsicum) page with a search for Capsicum. I was surprised to find 8 results and was thinking I could maybe create 8 different peppers but after further digging I found that not all the datasets were annotated and there was some overlap for the same species. After filtering for those that were both annotated and unique species, I arrived at 3 viable datasets:

* [Capsicum annuum](https://www.ncbi.nlm.nih.gov/assembly/GCF_000710875.1) from Jo YD et al [1] with [red peppers](https://www.diark.org/diark/species_list/Capsicum_annuum_cultivar_Zunla-1)
* [Capsicum baccatum](https://www.ncbi.nlm.nih.gov/assembly/GCA_002271885.2) from Kim S et al [2] with [yellow peppers](https://www.diark.org/diark/species_list/Capsicum_baccatum)
* [Capsicum chinense](https://www.ncbi.nlm.nih.gov/assembly/GCA_002271895.2) from Kim S et al [2] with [orange peppers](https://www.diark.org/diark/species_list/Capsicum_chinense)

I found [this paper](https://www.sciencedirect.com/science/article/pii/S0888754318305846) from Dubey et al [3] that listed 11 genomic regions suspected to be involved in the ripening of the pepper fruit and copied the primer nucleotide sequences and lengths from the supplementary materials.

The gff files from NCBI are simply tsv text files that list a feature (eg gene, mRNA, exon etc) along with the start and end coordinates as well as any additional metadata — see the example below. The goal is to subset these files to just those features that fall into the regions involved in the ripening process from the Dubey study.

NC\_029977.1 Gnomon gene 590985 592488 . - . ID=gene-LOC10...

NC\_029977.1 Gnomon mRNA 619017 620755 . + . ID=rna-XM\_016...

NC\_029977.1 Gnomon exon 619829 620755 . + . ID=exon-XM\_01...

In order to extract the gff features from only those regions of interest we need the corresponding coordinates, and this is achieved with the NCBI Basic Local Alignment Search Tool ([BLAST](https://blast.ncbi.nlm.nih.gov/Blast.cgi)). Using this tool we take each of the nucleotide sequences (eg GGCATCGT…) found in the Dubey study and look it up across the sequences of the entire genome for each plant — similar to grepping a very large book for a specific phrase.

BLAST does have a web interface but I wanted to script this step so it could be repeated across the three datasets consistently, so next I downloaded their [command line](https://www.ncbi.nlm.nih.gov/books/NBK569861/) application. I first attempted to use the NCBI servers for the processing by passing the -remote flag. The results were coming back suspiciously fast and I wasn’t getting any hits but then I discovered you need to also pass the -task blastn-short for short sequences, which doesn’t seem to be documented anywhere.

blastn -db “GPIPE/4072/100/GCF\_000710875.1\_top\_level” -query dubey.fsa -out results.out -task “blastn-short” -remote

This now had the opposite effect — the remote process was running for ages and I was getting Error: [blastn] Failed to fetch sequences in batch mode. By this point I decided to byte the bullet and create my own BLAST databases to run the process locally. I downloaded all the fna files from the same NCBI page and scripted a command similar to this:

for g in "${genome\_ids[@]}"; do

for ((i=1;i<=12;i++)); do

makeblastdb -in datasets/ncbi\_dataset/data/${g}/chr${i}.fna

-dbtype nucl -parse\_seqids -out blast/dbs/${g}/chr${i}.blast

done

done

This proved to be worthwhile: I repeated the blastn commands but without the -remote flag and soon after had a list of coordinates that reported perfect hits for each of the gene sequences involved in the ripening process. The final step was to use a bunch of bash commands (see [source](https://github.com/s1monj/nft-peppers) code) to prune and format the data so I was left with a simple 3 column tsv like below for each of the 3 species — the first column is the gene location ID used in the Dubey study to identify the region (11 different LOCs in total), and the second and third columns are the genomic coordinates and features found in the region (with a margin of error) from the gff file.

LOC107843860 213978278 gene

LOC107843860 213989346 mRNA

LOC107843860 213991885 exon

...



## **Generating the Visualization with Cytoscape.js**

I messed around with the Cytoscape desktop application a couple of years ago and I’ve always found the network graphics it produced to be aesthetically appealing. Obviously the data I prepared here is anything but a network, so this is not a typical use case but I took that creative license and chose it because I like the look of it.

Not surprisingly Cytoscape.js needs the data to be in a JSON format, what *was* a little surprising is it can’t read in a table (2D array) and deduce the nodes and edges like the desktop version does. Instead we need to explicitly define each node and edge as a data object like the example below.

const dataAnnuum = [{

"data": {

"id": "6679599"

}

},{

"data": {

"id": "gene"

}

},{

"data": {

"id": "6679599-gene",

"source": "6679599",

"target": "gene"

}

},

...

jq to the rescue:

for g in "${genome\_ids[@]}"; do

for loc in "${loc\_ids[@]}"; do

cat graph/$g-$loc.txt | jq --raw-input --slurp 'split("\n") | map(split(" ")) | .[0:-1] | map( { "data": { "id": .[1] } }, { "data": { "id": .[2] } }, { "data": { "id": (.[1] + "-" + .[2]), "source": .[1], "target": .[2] } } )' >> json/$g.js

cat graph/$g-$loc.txt | jq --raw-input --slurp 'split("\n") | map(split(" ")) | .[0:-1] | map( { "data": { "id": .[0] } }, { "data": { "id": .[1] } }, { "data": { "id": (.[0] + "-" + .[1]), "source": .[0], "target": .[1] } } )' >> json/$g.js

done

done

Once you’ve got your data formatted the only other gotcha for me was that I needed to wrap the constructor in a DOMContentLoaded listener (see below), but apart from that their [Getting Started](https://js.cytoscape.org/#getting-started/initialisation) example nicely works as described.

document.addEventListener("DOMContentLoaded", function () {

cyto1 = cytoscape({

container: document.getElementById("target"),

elements: dataAnnuum,

style: [...],

layout: {

name: "circle",

},

});

});

Cytoscape.js provides an impressive [API for animating](https://js.cytoscape.org/#core/animation) that can be used for dynamically styling nodes and edges (see below) but it’s also worth noting it has an entire feature set for viewport manipulation such as zooming and panning which is handy for proper communicative visualizations that need to step over and narrate diagrams. As a simple animation example, in the code below I find all the edges related to a type of gene feature (gene, exon, etc) and change the color and double the width of the line.

cyto1

.nodes("#" + geneFeature)[0]

.connectedEdges()

.animate({

style: {

lineColor: RIPE\_COLOR[seqNum],

width: SKELETAL\_WIDTH[seqNum] \* 2

},

});



## **Animating with Anime.js**

I wanted to bring the graphic to life a bit but the Cytoscape.js panning and zooming wasn’t really the effect I was after. I’d had a play around with [p5.js](https://p5js.org/) in the past which seems to be popular in the generative art world but I wanted to take this opportunity to give [anime.js](https://animejs.com/) a shot and boy I’m glad I did. With just a few lines of code I was able to pick up the div containing the Cytoscape graphic and smoothly rotate it back and forth with nice start-and-stop easing.

anime({

targets: #graph1,

rotate: [45, -45],

duration: 10000,

easing: "easeInOutSine",

direction: "alternate",

loop: true,

});

I also used the skew effect to grow the graphic onto the page when it first loads. Things get slightly more tricky when you need to start timing and sequencing the effects and anime.js provides a [timeline](https://animejs.com/documentation/#timelineBasics) feature for this. I was moving all my animations onto a timeline when I discovered there is also a simple complete: option that can be provided to fire a function when each animation has finished, which proved to be a much more elegant solution for sequencing this simple animation.



## **Deploying on IPFS**

Non-fungible is an [economic term](https://www.investopedia.com/terms/f/fungibility.asp) that is used to describe things that are unique and not interchangeable with other things of the same type. The concept behind non-fungible tokens is that not only are they unique but their uniqueness can be readily verified by a public record on the Blockchain. When a digital asset, say a single still “photo.jpg” file, is added to the Ethereum Blockchain as an [NFT](https://ethereum.org/en/nft/), a common misconception is the actual bytecode of the file “photo.jpg” is written to a block. With transactions and on-chain data alone, the Ethereum blockchain today just crossed the [1TB](https://ycharts.com/indicators/ethereum_chain_full_sync_data_size) mark so you can imagine if every single NFT jpeg, gif and movie file was added it would quickly balloon out to petabytes of data. Given that the primary motivation for Blockchain is decentralization, if the chain grew to this size it would be prohibitively large for most nodes to maintain, leaving a more centralized network of big boxes.

Instead of storing the NFT data files on-chain, the NFT instead has an immutable tokenURI record that points to a file. If this tokenURI uses location addressing such as a conventional web server eg. <http://myserver.com/photo.jpg> then having an irrevocable tokenURI record is pointless if the file at the address destination can just be switched out, which is [exactly what happened](https://www.theverge.com/2021/3/25/22349242/nft-metadata-explained-art-crypto-urls-links-ipfs) to some unhappy NFT owners.

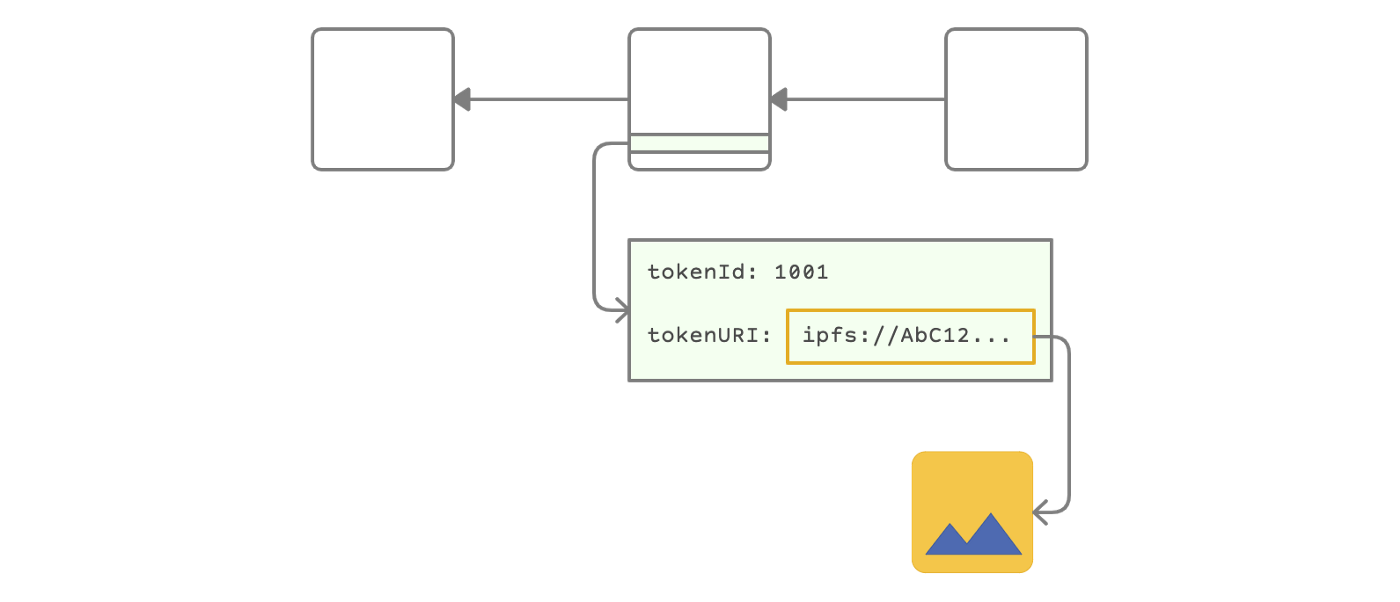


Image by author

This is where [IPFS](https://docs.ipfs.io/concepts/what-is-ipfs/) comes in. The InterPlanetary File System is a peer-to-peer distributed storage network, not unlike BitTorrent, that’s made up of computers all over the globe storing and sharing data. IPFS also uses a cryptocurrency called [FileCoin](https://docs.ipfs.io/concepts/faq/#ipfs-and-filecoin) as an incentive layer so that users are rewarded for storing data on their computers. The key difference here is that instead of using location addressing like conventional web sites, IPFS uses content addressing, which involves generating a hash (“CID”) of the file or directory and using that for retrieval.

Location addressing:<http://myserver.com/photo.jpg>

Content addressing: ipfs://QmP7u9UzoUtfpiNkj3Sa...

To access an IPFS content from a web browser you use a gateway such as *ipfs.io* provided by Cloudflare, eg:

<https://ipfs.io/ipfs/QmP7u9UzoUtfpiNkj3SaL5TLym2ngrUfvsiNhf21uFF3L1>

The beauty of using IPFS for NFT data is that not only is it decentralized and always-on, but if a single byte is changed in a file the CID changes. This means that the immutable tokenURI of the NFT record is always guaranteed to return the exact same data.

Before you start uploading the data to IPFS, there is one additional consideration: the metadata used to display the NFT. Rather than set the tokenURI of the NFT to point directly to the file on IPFS, you instead point it to a metadata JSON file on IPFS, which includes information on the NFT and in turn points to the source file as well as any additional files such as a preview image. Because this metadata file is also stored on IPFS you can be guaranteed it too has not been modified. There is no official Ethereum specification but the [format](https://docs.opensea.io/docs/metadata-standards) described by OpenSea has become standard.

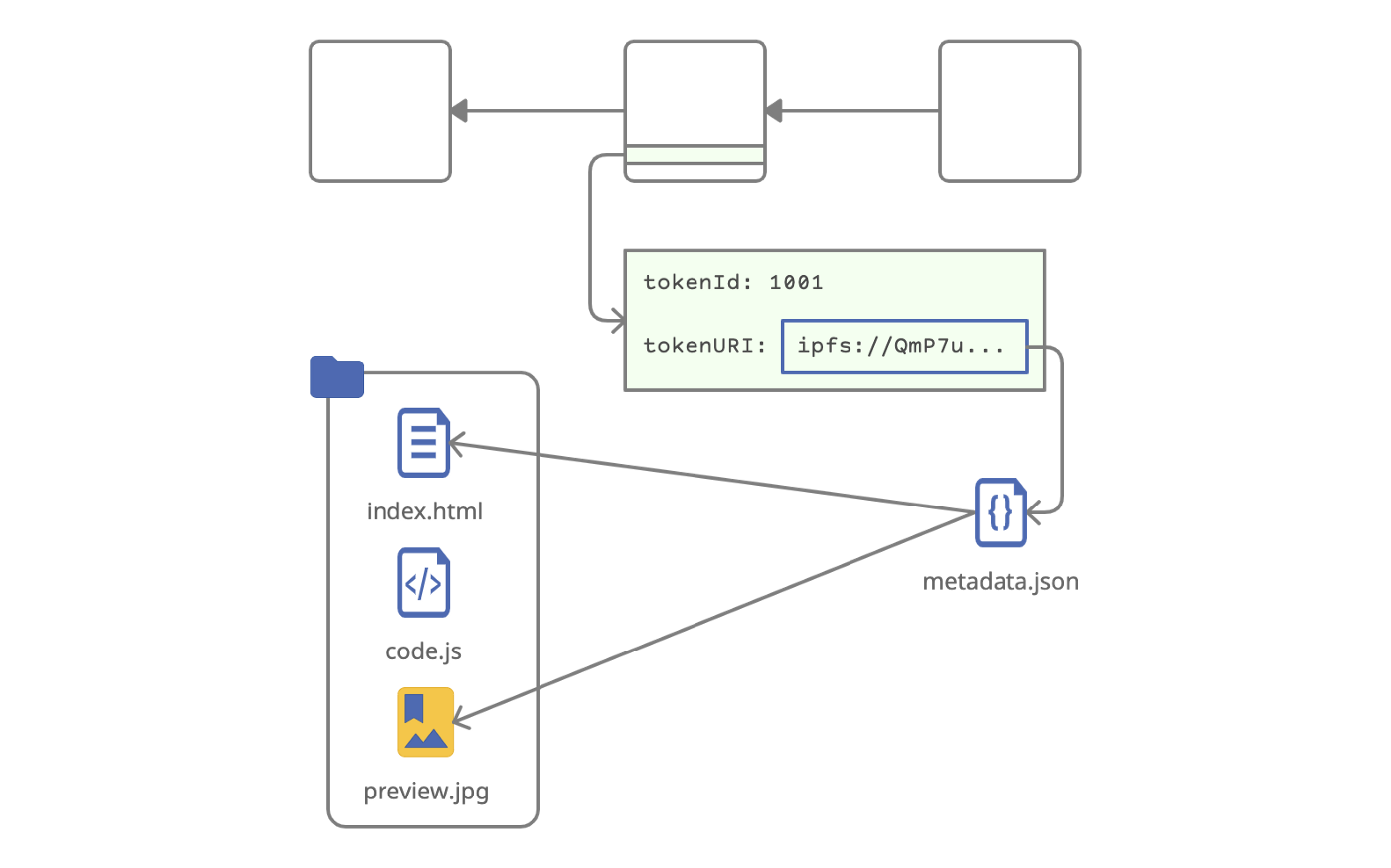


Image by author

To get the files onto IPFS you can download the [Desktop App](https://docs.ipfs.io/install/ipfs-desktop/) and use it to upload the complete source directory, including the index.html, the genomic data, the javascript, the supporting libraries and the preview images. Note that the file paths within the index.html do not need to use content addressing (the IPFS gateway can resolve location addressing at the directory level) but for this reason they do need to be relative.

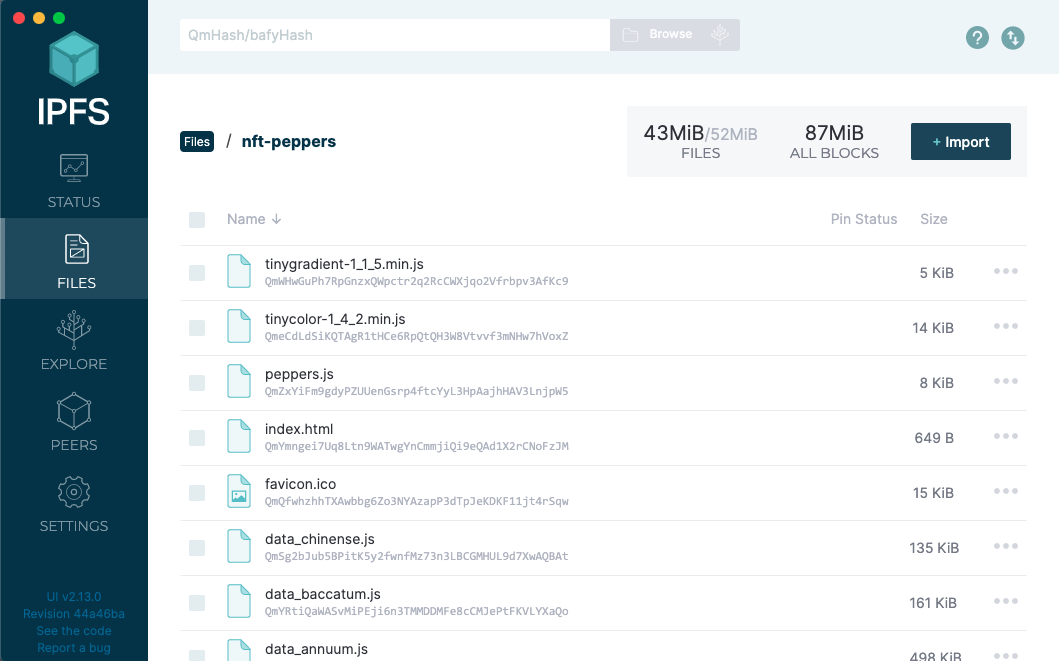


Image by author

With the data now on IPFS, you then copy the CID of the directory and the image preview files and complete the entries in the metadata.json files. Finally you upload the metadata files and copy the CID that will be used with the NFT. Note that the tokenId=1001 parameter passed to the index.html is simply used as a sequence number so that the single source code directory can generate the 3 different variations (species) depending on the URL referenced in the metadata.

{

"token\_id": 1001,

"name": "Capcicum annuum",

"description": "This red pepper depiction is generated...",

"image": "ipfs://QmYpZF5D95THTiF4gVuUuze46eA4SHjwWEZBvFXSn7GE2k/annuum-preview.png",

"background\_color": "fcf5e5",

"external\_url": "<https://nicetotouch.eth.link/peppers>",

"animation\_url": "ipfs://QmYpZF5D95THTiF4gVuUuze46eA4SHjwWEZBvFXSn7GE2k/index.html?tokenId=1001",

"attributes": [

{ "trait\_type": "Species", "value": "Capsicum annuum" },

...

]

}

The IPFS Desktop application is actually serving your files out to the world, so they will only be available so long as your computer is running. To ensure the files are always-on and replicated to other peers (rather than just temporarily cached) we need to “pin” them. There are a few different pinning services out there but I chose [Pinata](https://www.pinata.cloud/) that gives you 1GB free and is extremely straightforward to use. Simply sign up, click to upload/add and paste in the CIDs you want to pin.

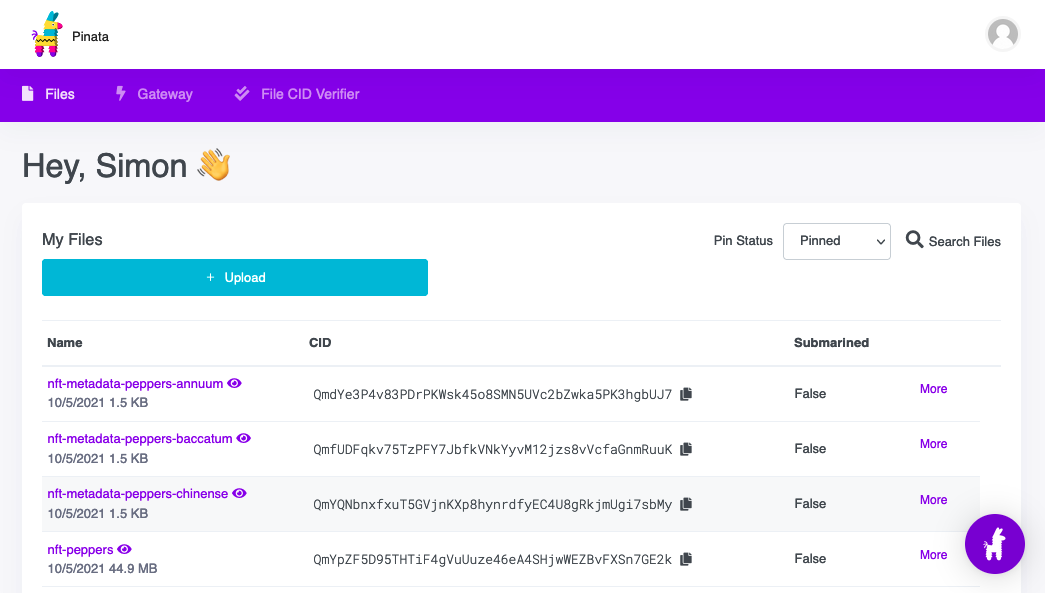


Image by author



## **Minting on the Ethereum Blockchain**

The final step is to claim ownership of your data by creating a public, verifiable record on the Blockchain. There are plenty of NFT aggregators like OpenSea that offer a minting service but unfortunately they only support creating simple, single-file NFTs such as images and movies. [BeyondNFT](https://beyondnft.io/) is the only site I found that allows you to mint dynamic NFTs by uploading a zip file but at the time of writing, creating your own smart contract was not supported and instead you had to use their generic catch-all contract.

To conceptualize the role of the contract, the “token” part of the NFT can be thought of like an instance of an object class, with the object class being an implementation of the [ERC721](https://eips.ethereum.org/EIPS/eip-721) smart contract. The smart contract lives on the Blockchain and has a mint function that creates a unique token with an immutable set of variables (eg a tokenId and tokenURI). For this reason, all tokens created from the same contract are considered a “collection” and are grouped together, so when creating a set of NFTs as in this example you really want to be using your own contract rather than joining some existing large generic collection that you have no control over.

There are a few steps involved in creating your own smart contract and then minting an NFT and luckily for me they were perfectly outlined in [this fantastic guide](https://medium.com/@mondoir/your-very-own-smart-contract-for-nfts-b068b7d65e5d) by Amir Soleymani. I had no prior experience with Solidity or deploying smart contracts but following this post I had the 3 NFTs minted in about half an hour.



## **Use Cases for Scientific Data**

When I was working in bioinformatics I seemed to spend a large amount of time checking, double-checking and triple checking that the files I thought I was storing, moving or processing were actually indeed the correct files. Spreadsheets of sample names with corresponding file names, server locations and directory paths with the words “new” or “latest” or “fixed” haunted me.

If you mix up the naming of a Word document or spreadsheet or JavaScript source file you can pretty quickly determine you have the wrong file just from opening it but this is often not the case with scientific data. Sample data can be so dense and homogeneous that there is no simple guaranteed way of telling file content apart other than by the results, which *IS* the analysis, meaning a file name mix-up can be disastrous.

This is where content addressing data makes a whole lot of sense. Yes checksums are great when you have them but to be useful they really need to be generated right from the source of the data and carried around everywhere the data goes. *Addressing* by checksum however gives us two birds with one stone: we can be guaranteed we’re downloading the correct, original, unmodified file and we can also checksum it on our end to be sure we have transferred it completely.

The decentralized nature of IPFS is also well suited to scientific data. The academic world rarely has the same resources as their corporate counterpart to provide highly available, highly redundant data and ironically most academic data is public and for the public good. If the NCBI website goes offline today, while they may well have their data mirrored on other servers and backed-up on tape, if I can’t access their website I have no way of accessing the data I need. With a content address I can download the data directly from the IPFS, independently from any URL and regardless of if the NCBI node or gateway is down. In addition to this, the peer-to-peer component operates like a content delivery service, so while I’m working with a ginormous genome reference file, if my colleagues next door want to download the same data, they’ll be automatically getting it straight from me rather than hopping across the globe.

Where IPFS is not so perfectly suited to scientific data is on the provenance and lineage front, which is where Blockchain and NFTs step it. At the most basic level, having an immutable record on a highly accessible, fully redundant, secure, decentralized service to describe a dataset or multiple datasets brings a whole lot of value to data science. When the entity responsible for owning the data changes, the NFT can be transacted and the history permanently recorded, and if a record needs to be updated, a new token can be issued with updated metadata that includes a reference to the old token and old data.

On a higher level are the possible applications for a new system of recording citations and intellectual Property rights. I’m not an academic but in today’s information age the system of journal publishing and citing seems antiquated and cumbersome. Richard Forda Burley put out a [paper](https://content.iospress.com/articles/information-services-and-use/isu180017) [4] a few years ago proposing a Bitcoin-inspired decentralized shared citation ledger. He notes that centralized ledgers (eg Scopus and Web of Science) are big business and often “…important decisions on what is and is not indexed are based in many ways on economic considerations” which is a problem for disseminating scientific knowledge. Imagine if instead a research paper, along with all of the accompanying supplementary material, datasets and software was tokeized on the blockchain and the IPFS, owned by the authors and readily available for anyone to read and reproduce. What about indexing? Projects like [The Graph](https://thegraph.com/) have already invented Dapps (decentralized applications) to index Blockchain data and many believe it’s only a matter of time before we are searching and browsing Web3 (the decentralized web) like we use Google today.

Finally, Blockchain is an obvious match for recording IP rights, as [noted](https://www.wipo.int/wipo_magazine_digital/en/2020/article_0002.html) last year by the World Intellectual Property Organization. Although their discussion is focused on designs and trademarks, I also envisage an application for patents that is relevant to this example. Again at a high-level and from my limited understanding of chemical utility patents, a large part of the patent is describing the work along with how it was made and how it can be repeated or reproduced (“enablement”). Imagine if the data was nicely packaged and timestamped on a decenteralized Blockchain, the same package used for publishing and citing, with no conflicting interests or international political interference, then the patent authority could simply point to this record for the world to see.

I hope this has provided some insight into how dynamic visualizations can be added to the Ethereum Blockchain as NFTs and also gets you thinking about possible future use cases for Web3 in data science. You can view the final result [here](https://opensea.io/assets/0x86944176301532d4432b50838cea252e3b2990bb/1001) and the source on GitHub [here](https://github.com/s1monj/nft-peppers).

**Clip Art**

All clip art above was sourced from unDraw ([https://undraw.co](https://undraw.co/)) and used under the the [unDraw Open Source license](https://undraw.co/license).

**References**

[1] Jo, Y. D., Park, J., Kim, J., Song, W., Hur, C. G., Lee, Y. H., & Kang, B. C., Complete sequencing and comparative analyses of the pepper (Capsicum annuum L.) plastome revealed high frequency of tandem repeats and large insertion/deletions on pepper plastome (2011), *Plant cell reports*, *30*(2), 217–229. <https://doi.org/10.1007/s00299-010-0929-2>

[2] Kim S, Park J, Yeom SI, et al., New reference genome sequences of hot pepper reveal the massive evolution of plant disease-resistance genes by retroduplication (2017), *Genome Biol*. 2017;18(1):210. Published 2017 Nov 1. [https://doi.org/10.1186/s13059–017–1341–9](https://doi.org/10.1186/s13059%E2%80%93017%E2%80%931341%E2%80%939)

[3] Meenakshi Dubey, Vandana Jaiswal, Abdul Rawoof, Ajay Kumar, Mukesh Nitin, Sushil Satish Chhapekar, Nitin Kumar, Ilyas Ahmad, Khushbu Islam, Vijaya Brahma, Nirala Ramchiary, Identification of genes involved in fruit development/ripening in Capsicum and development of functional markers (2019), *Genomics*, Volume 111, Issue 6, 2019, <https://doi.org/10.1016/j.ygeno.2019.01.002.>

[4] Burley, Richard Ford, Stable and decentralized? The promise and challenge of a shared citation ledger (2018), *Information Services & Use*, vol. 38, no. 3, pp. 141–148, 2018, <https://doi.org/10.3233/ISU-180017>