Klay-Gacha-Machine Docs

About

Introduction

Klay-Gacha-Machine이란?

Klay-Gacha-Machine은 클레이튼 환경에서 NFT를 발행하기 위한 에코시스템을 지향하며, 클레이튼 기반의 NFT 발행을 원하시는 개발자와 발행인들을 위해 많은 개발도구들을 지원해 개발적 요소를 최소화하는 것을 목표로 합니다.

Klaytn이란?

클레이튼은 카카오의 블록체인 관련 자회사 그라운드X가 개발한 국내 코인으로 세계 1위 거래소 등 다양한 거래소에서 거래가 지원되고 있습니다.

Klay-Gacha-Machine에서는 무엇을 제공하나요?

Klay-Gacha-Machine은 클레이튼 기반의 NFT 발행을 위한 스마트 컨트랙트를 포함한, NFT 발행인이 NFT 발행에 필요한 핵심적인 도구들을 제공하고 있습니다.

Gacha-Machine-CLI - NFT 발행을 위한 이미지 파일의 IPFS, AWS S3 업로드 및 메타데이터 URL 삽입 기능, 그리고 메타데이터의 URL의 IPFS, AWS S3 업로드 및 스마트 컨트랙트 업로드 기능을 제공하여 몇번의 명령어 입력으로 NFT 발행을 위한 모든 준비를 한번에 끝낼 수 있는 기능을 제공합니다.

Mint UI - Gacha-Machine-CLI를 이용해 업로드가 끝난 NFT를 발행하기 위한 Mint UI를 제공합니다. 사용자는 Mint UI를 NFT를 발행/판매할 웹사이트에 올려 NFT 발행/판매를 쉽게 실시할 수 있습니다.

Product

Gacha-Machine-CLI

```
F:\code\gacha_nft_machine>node gacha-cli.mjs upload images -n baobab -i
{ network: 'baobab', ipfs: true }
New collection is successfully made.
Number :  0
Path is images/0.jpg
Path is images/0.json
Number :  1
Path is images/1.jpg
Path is images/1.json
Number :  2
Path is images/2.jpg
Path is images/2.json
Number :  3
Path is images/3.jpg
Path is images/3.json
```

Gacha-Machine-CLI의사용

NFT 발행을 위한 이미지 파일의 IPFS, AWS S3 업로드 및 메타데이터 URL 삽입 기능, 그리고 메타데이터의 URL의 IPFS, AWS S3 업로드 및 스마트 컨트랙트 업로드 기능을 제공하여 몇번의 명령어 입력으로 NFT 발행을 위한 모든 준비를 한번에 끝낼 수 있는 기능을 제공합니다.

Mint UI

민팅버튼UI

Gacha-Machine-CLI를 이용해 업로드가 끝난 NFT를 발행하기 위한 Mint UI를 제공합니다. 사용자는 Mint UI를 NFT를 발행/판매할 웹사이트에 올려 NFT 발행/판매를 쉽게 실시할 수 있습니다.

Overview

왜 Klay-Gahca-Machine인가?

Klay-Gacha-Machine은 클레이튼의 NFT 개발 환경 내에서 표준 프로토콜이 되는 것을 지향하고 있습니다. 특히 Klay-Gacha-Machine은 NFT 개발자와 발행인들이 정보를 공유하고 제안하는 발전적 커뮤니티를 만들고자 합니다.

Klay-Gacha-Machine 사용의 특장점

스마트 컨트랙트를 직접 만들 필요 없이 스마트 컨트랙트를 통한 KIP17 규격의 NFT를 쉽게 발행할 수 있습니다.

NFT를 직접 저장소에 업로드하고, 메타데이터를 직접 관리 및 수정할 필요 없이 단 한번의 명령어로 업로드를 끝낼 수 있습니다.

제공되는 Mint UI를 이용해 웹사이트의 지갑연결 및 NFT 발행까지 쉽게 진행시킬 수 있습니다.

Klay-Gacha-Machine의 향후 계획

Pinata IPFS와 AWS S3 외에도 사용자 친화적인 NFT 데이터 저장소의 업로드 기능 추가 구현

NFT 발행인의 자체적인 마켓플레이스 추가 구현

더 나은 사용자 경험을 위한 기능 업데이트

먼저 설치해야 하는 것들

Node.js

> 다운로드 | Node.js
> Node.js

javascript 기반의 Node.js를 설치하여 프로젝트에 필요한 의존 모듈을 설치하고 관리할 수 있어야 합니다.

개발환경 IDE

> https://code.visualstudio.com/download
> code.visualstudio.com

IDE는 개발자로 하여금 개발에 필요한 툴을 제공하며 편리한 개발을 도와주는 개발환경입니다. 어떠한 IDE(Atom, Bracket 등)를 사용하셔도 좋습니다.

GIT(옵션)

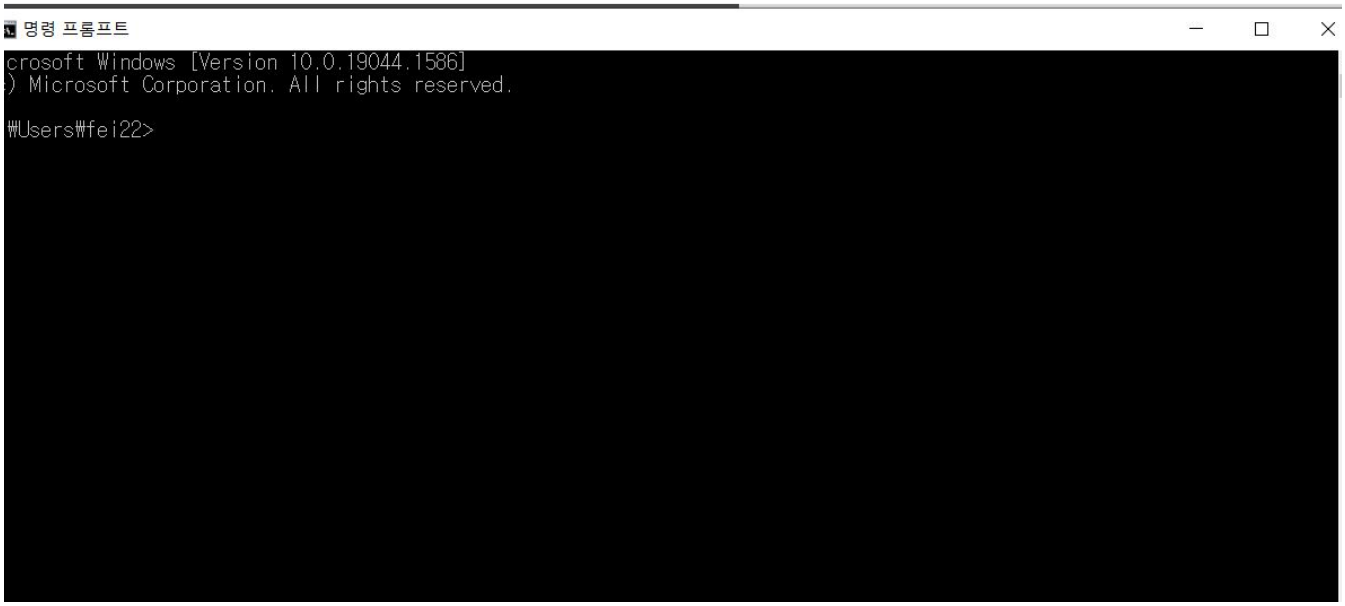
> Git

프로젝트의 버전관리 및 코드 관리를 도와주는 툴입니다. Klay-Gacha-Machine을 Cloning하기 위해 필요합니다만, 코드를 직접 github에서 다운로드받을 수도 있으니 설치하지 않으셔도 무관합니다.

Klay-Gacha-Machine 설치하기

---

git clone을 이용한 설치



명령 프롬프트를 켜고 다음 명령어를 입력합니다.

```
git clone https://github.com/Planet-Us/Klay_Gacha_Machine.git
```

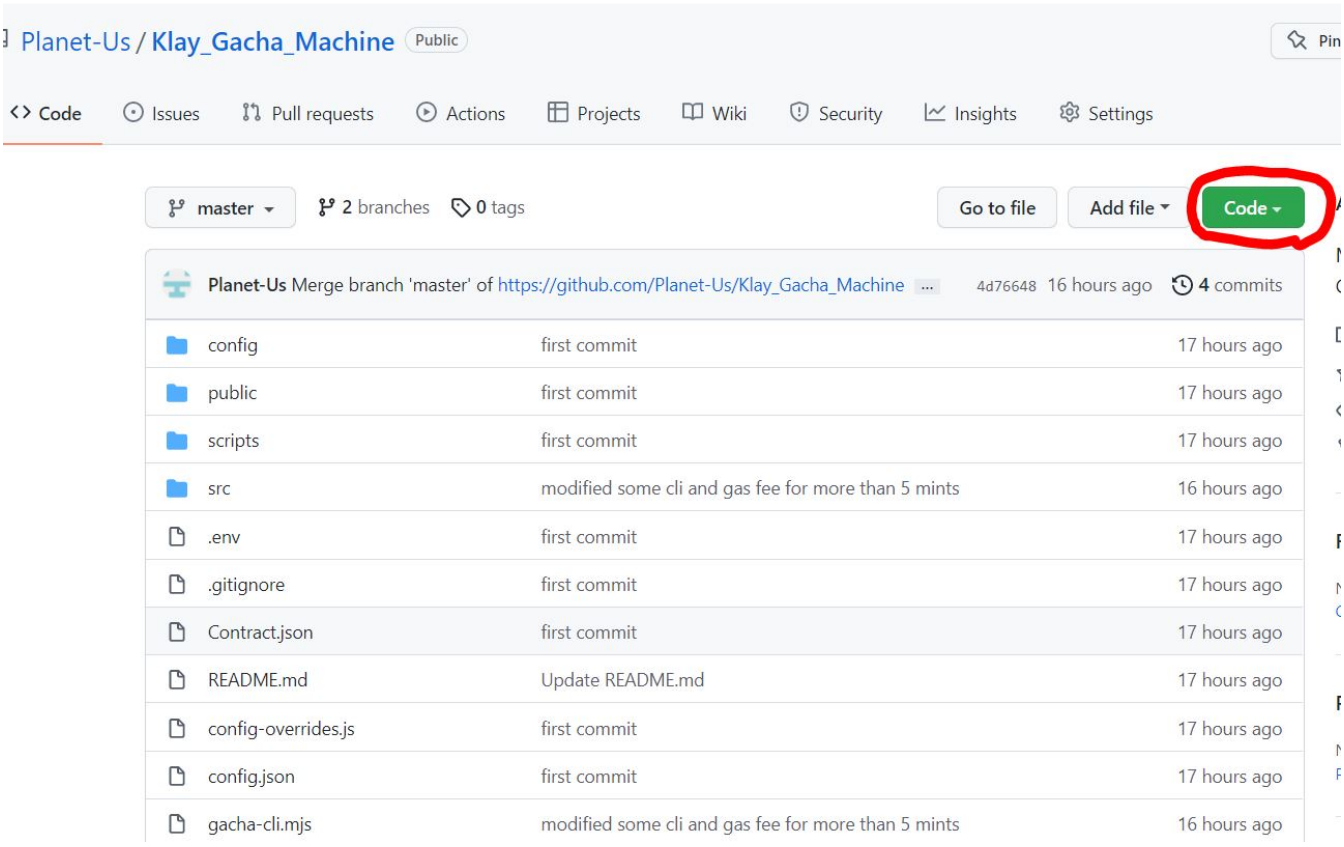
```
cd Klay_Gacha-Machine
```

```
npm install
```

프로젝트를 직접 다운로드해 설치

다음 링크로 접속해 코드를 다운받습니다.

> GitHub - Planet-Us/Klay_Gacha_Machine: Making Klaytn NFTs without coding, with CLIs
> GitHub



Code 버튼을 클릭하면 zip 파일로 다운로드받으실 수 있습니다

다운받은 파일의 압축을 풀고, 해당 폴더 내에서 명령어 프롬프트를 켜고 다음 명령어를 입력합니다.

```
F:₩code>cd Klay_Gacha_Machine

F:₩code₩Klay_Gacha_Machine>npm install
```

npm install을 통해 프로젝트에 필요한 의존 모듈을 설치할 수 있습니다. 이 과정은 몇분가량 소요됩니다.

Klay-Gacha-Machine

준비물

1. **NFT의 대상 작품 1~10000개(갯수는 상관없음)**
2. **NFT 작품 갯수만큼의 Metadata(Json파일)**
3. **Pinata ipfs를 사용하는 경우 - pinata ipfs의 api key와 secret api key**
   AWS S3를 사용하는 경우 - AWS S3의 S3의 모든 권한이 포함된 api key와 secret key
4. **Kaikas 지갑 주소 및 프라이빗 키**

Metadata 표준

Metadata의 standard, 즉 표준은 다음과 같습니다.

```
{
“name”:”NFT의 이름",
“symbol”:”NFT의 토큰심볼”,
“description”:”NFT의 설명, 그러니까 한줄짜리 홍보용 문구”,
“image”:”https://ipfs.io/ipfs/{해시값} 이미지의 주소",
“attributes”:[{“trait_type”:”속성명 ex)hair color”,”value”:”속성값 ex) red”}]
}
```

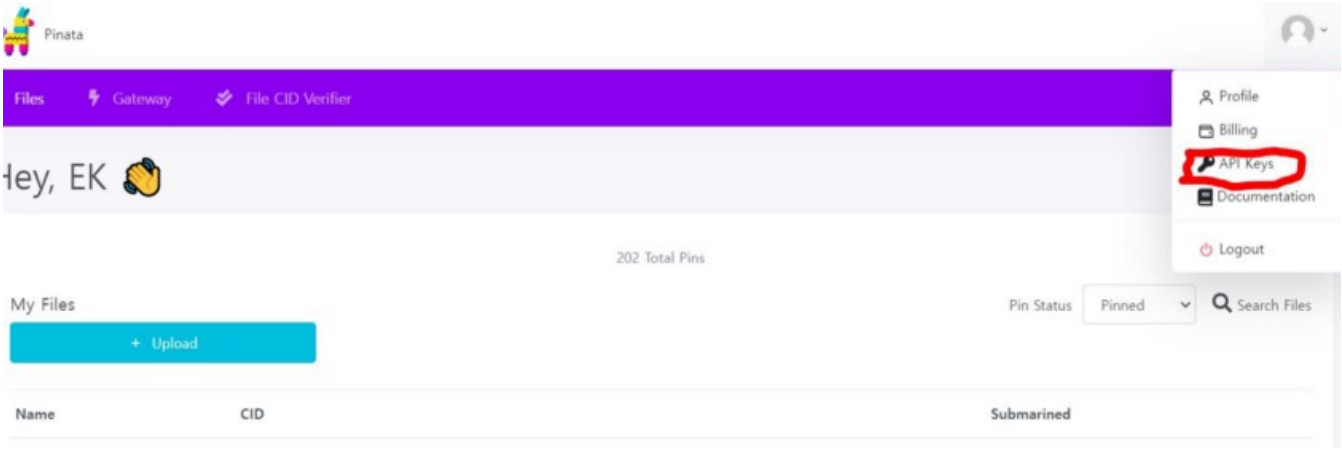
가장 필수적인 것들만 적어놓았습니다. 이 외에도 Json 데이터가 허용하는 값이면 뭐든지 가능합니다.

각 이름은 “NFT이름 #001”처럼 번호를 매기는 것이 일반적이고, 심볼은 넣지 않아도 되지만 구분을 위해 넣는 것이 좋습니다. 속성값(attribute) 또한 넣지 않아도 무방하지만, 제너러티브 NFT를 만드는 이유는 판매하는 것이며, 상품의 특성이 없는 한 구매까지 이어지지도 않으니 보통은 attribute를 넣습니다. 속성값은 몇개를 넣어도 상관은 없지만, 보통 3~6개정도 넣습니다.. 속성값 자체에 Rarity를 common, rare, unique와 같은 방식으로 넣기도 합니다.

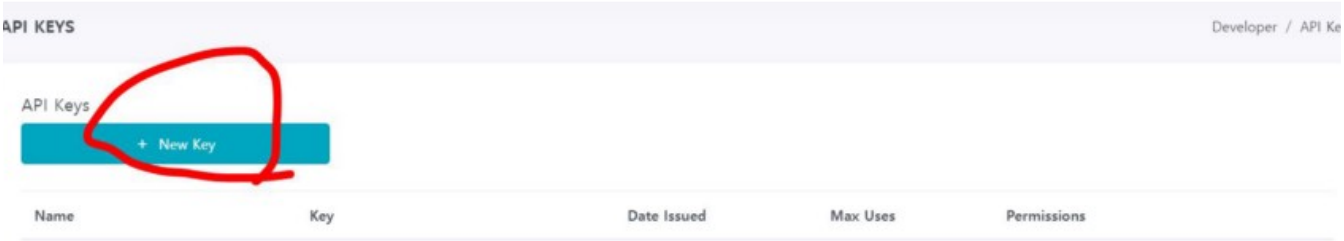
Pinata IPFS Key 가져오기

일단 pinata ipfs의 웹사이트에 접속합니다.

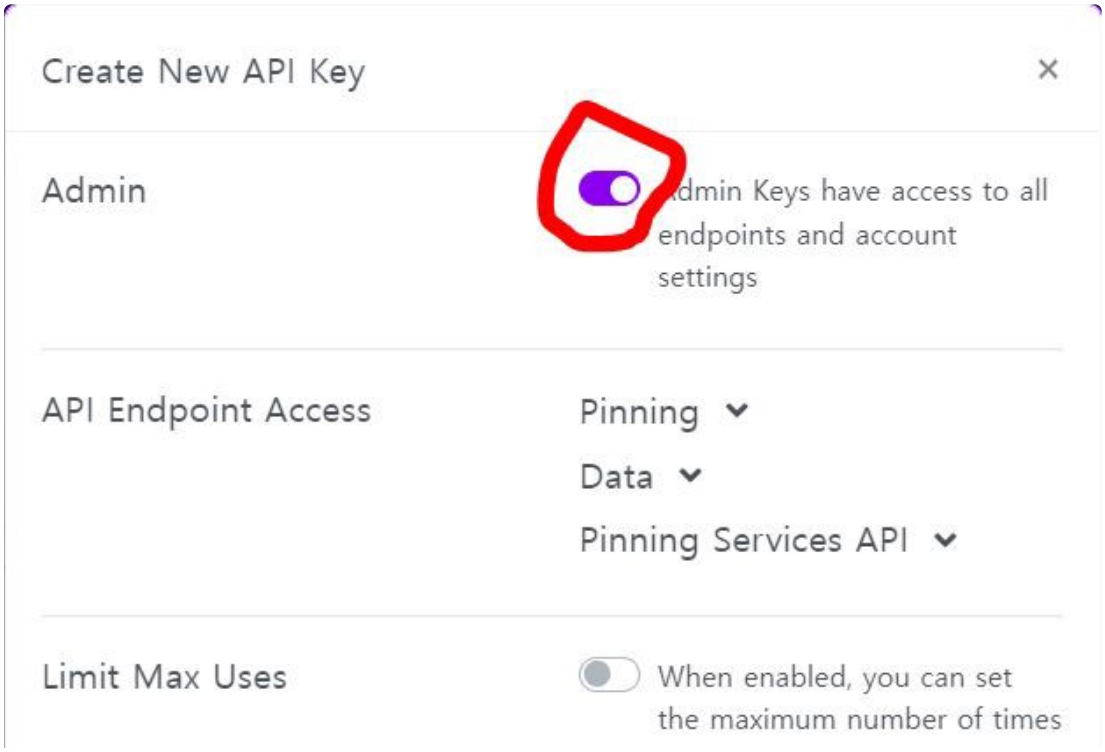
가입을 하고 접속하면 다음과 같은 화면이 뜨는데,

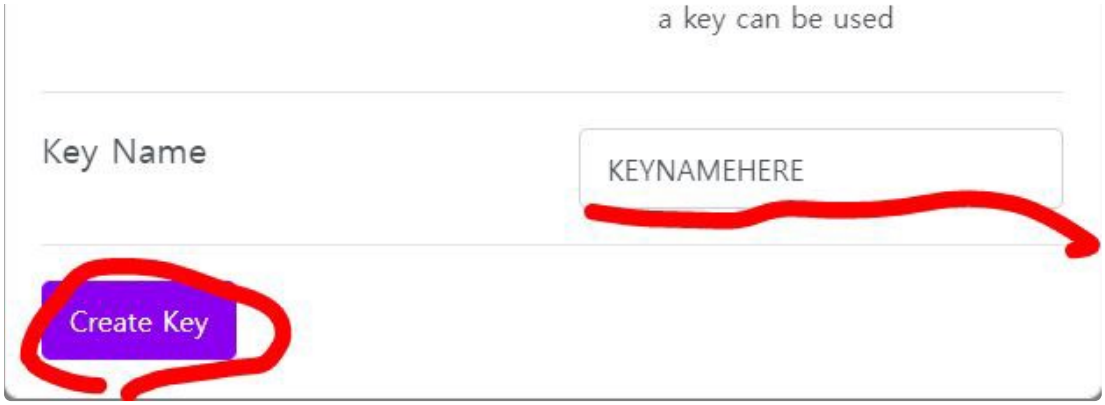


우측 상단에 있는 API Keys 항목에 들어갑다.

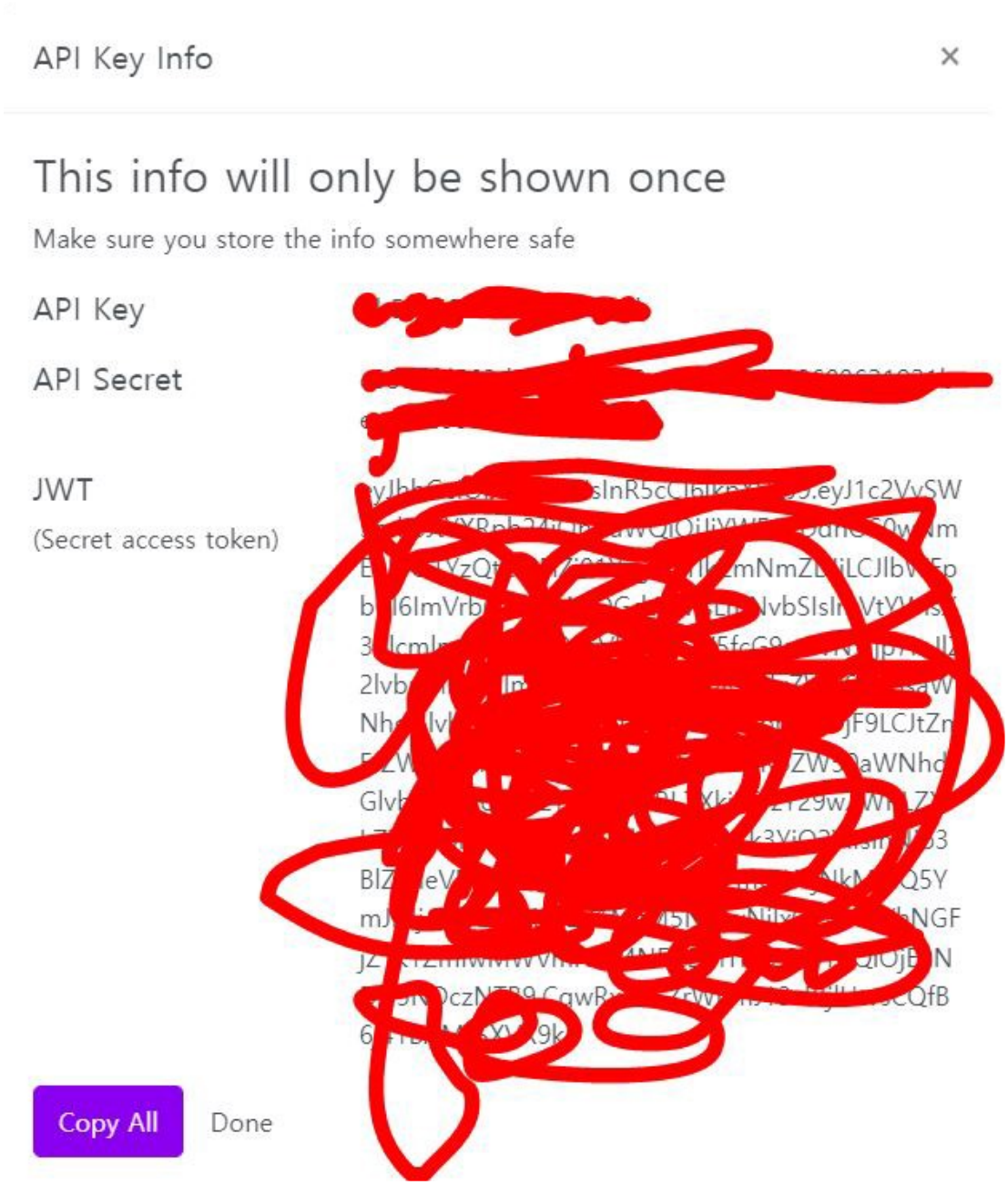


그리고 New Key 버튼을 눌러서 api키를 생성하는데,





Admin permission을 on으로 해두고, Key Name을 설정한 다음, Create Key를 누르면,



요렇게 API Key와 API Secret, 그리고 JWT를 줍니다. **절대로 창을 닫지 말고, Copy All 버튼을 눌러서 전부 카피를 뜬 다음, 메모장 등에 보관해야 한다.**

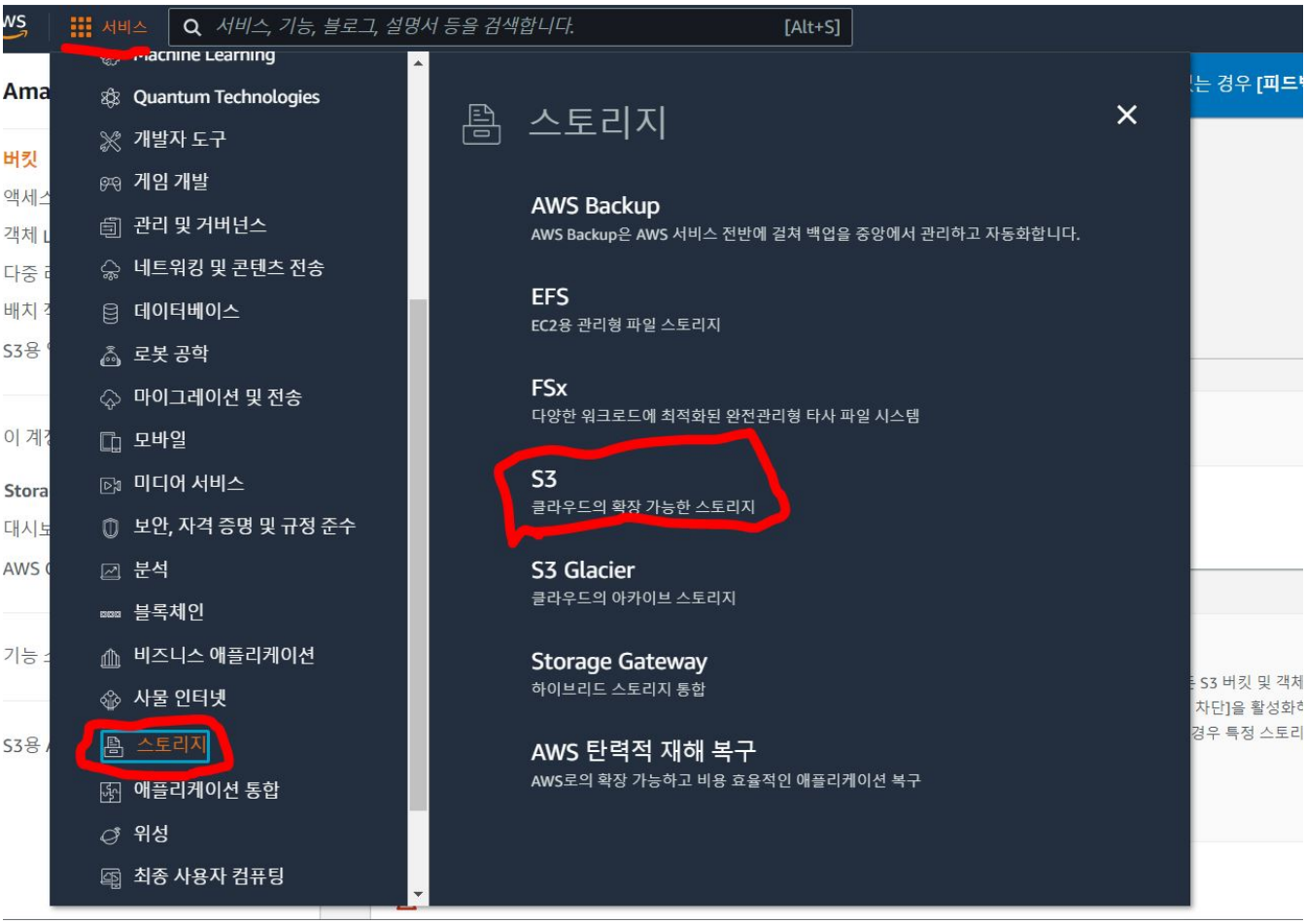
(그냥 닫으면 다시 가져올 방법이 없어서 다시 만들어야 합니다.)

AWS S3 api key 가져오기

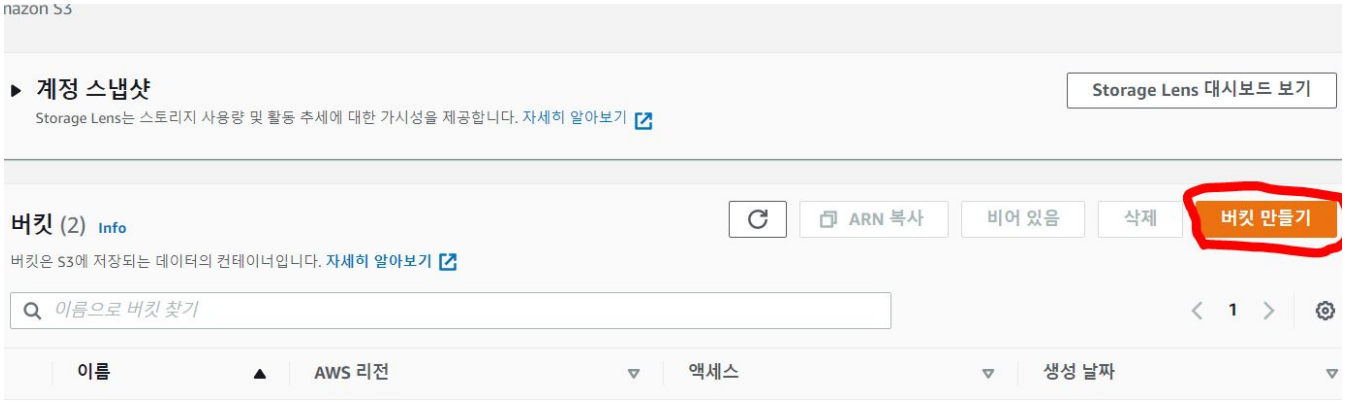
리빌(reveal)용이나 브리딩 시스템(breeding system)을 구상하고 있는 NFT 발행인은 AWS S3 스토리지를 이용해 NFT를 발행할 수 있습니다. 먼저 AWS(amazon web service)에 접속해 계정을 발행합니다.

https://aws.amazon.com/ko/

계정 접속 후 콘솔로 접속합니다.



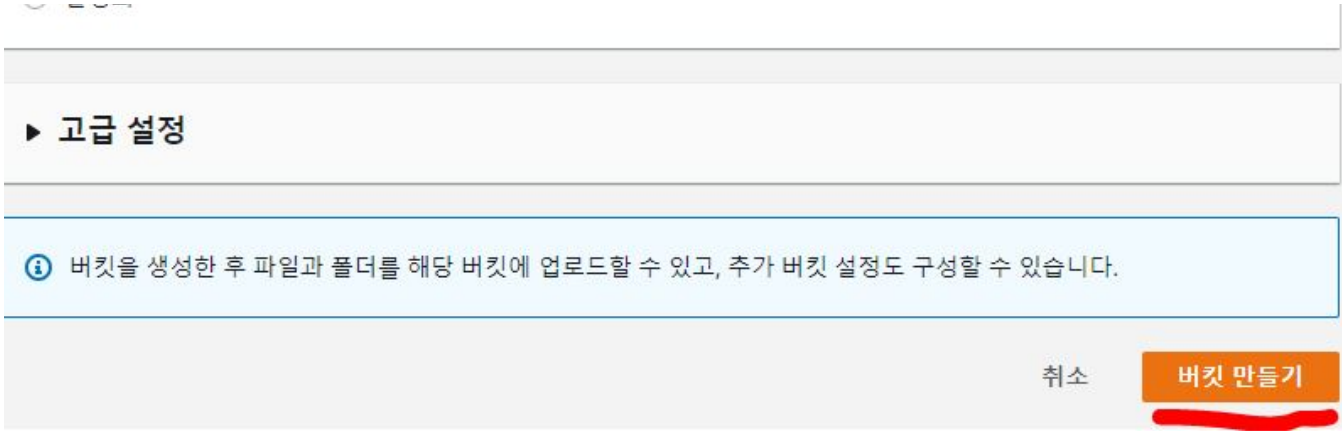
좌측 상단의 서비스목록을 클릭하고, 스토리지 서비스탭의 S3항목으로 들어갑니다.



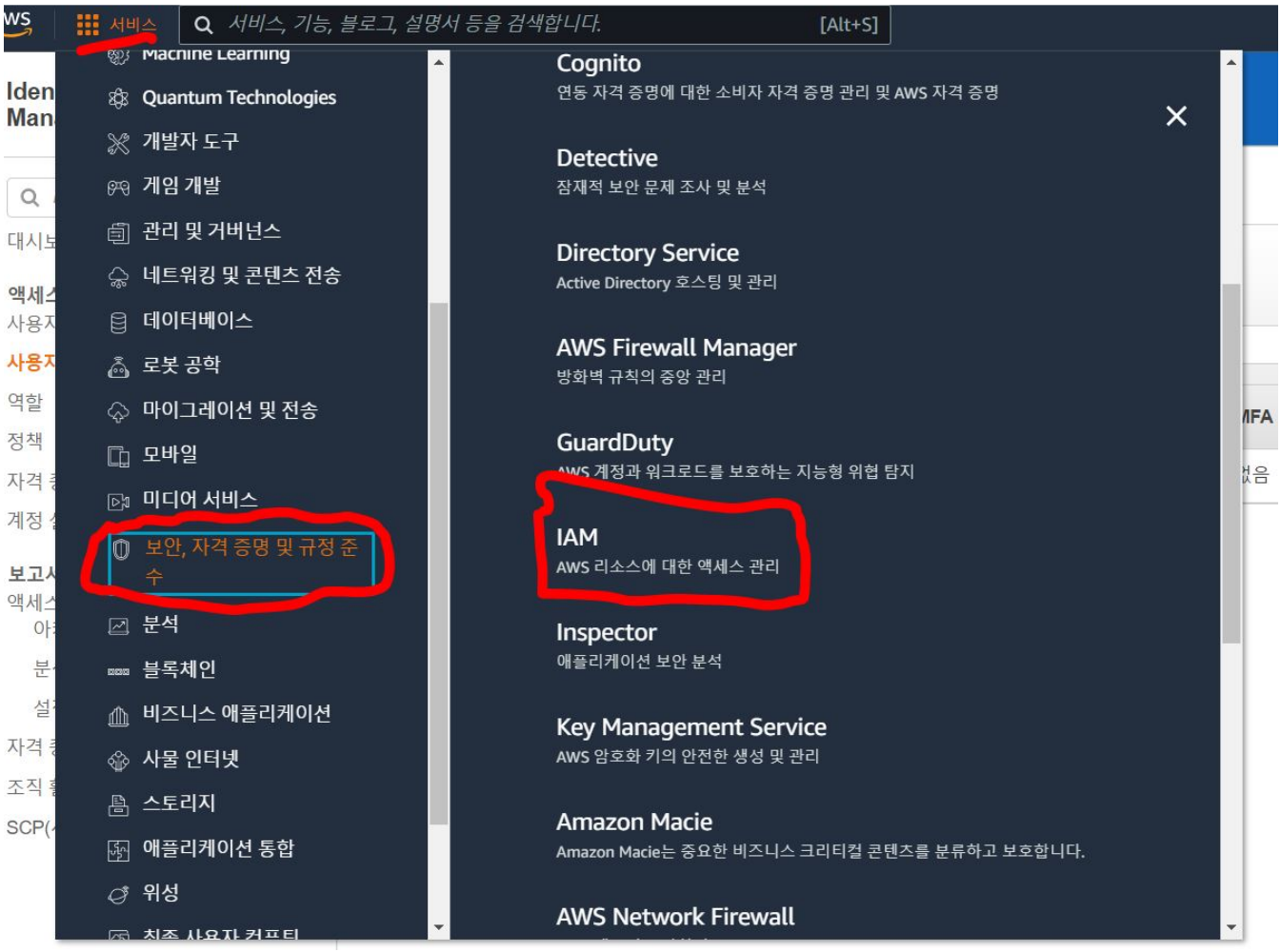
S3항목에 접속한 뒤, 우측의 버킷 만들기를 클릭합니다.

버킷 이름을 입력하고, 리전을 선택합니다. 그리고 객체 소유권의 "ACL 활성화됨"을 클릭합니다.

아래의 "모든 퍼블릭 액세스 차단"의 체크박스를 해제하고, 아래에 있는 "현재 설정으로 인해..."부분의 체크박스를 체크합니다.

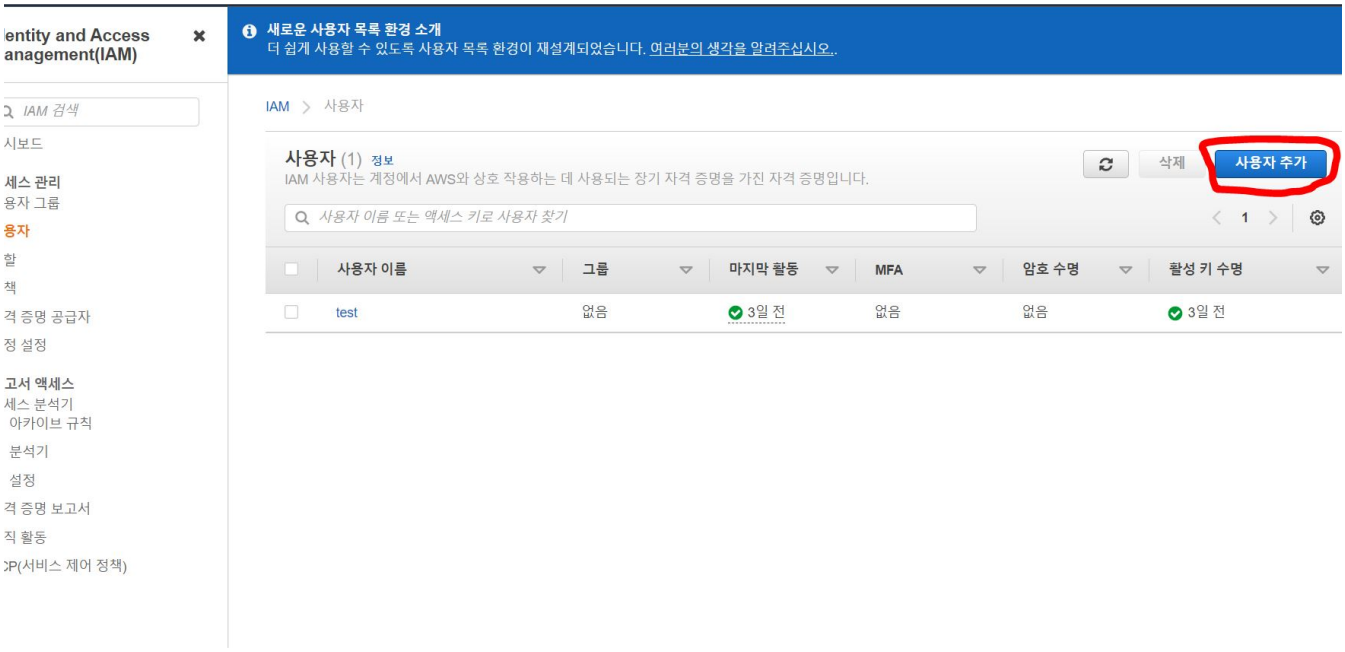


버킷 만들기를 클릭해 버킷을 만듭니다.

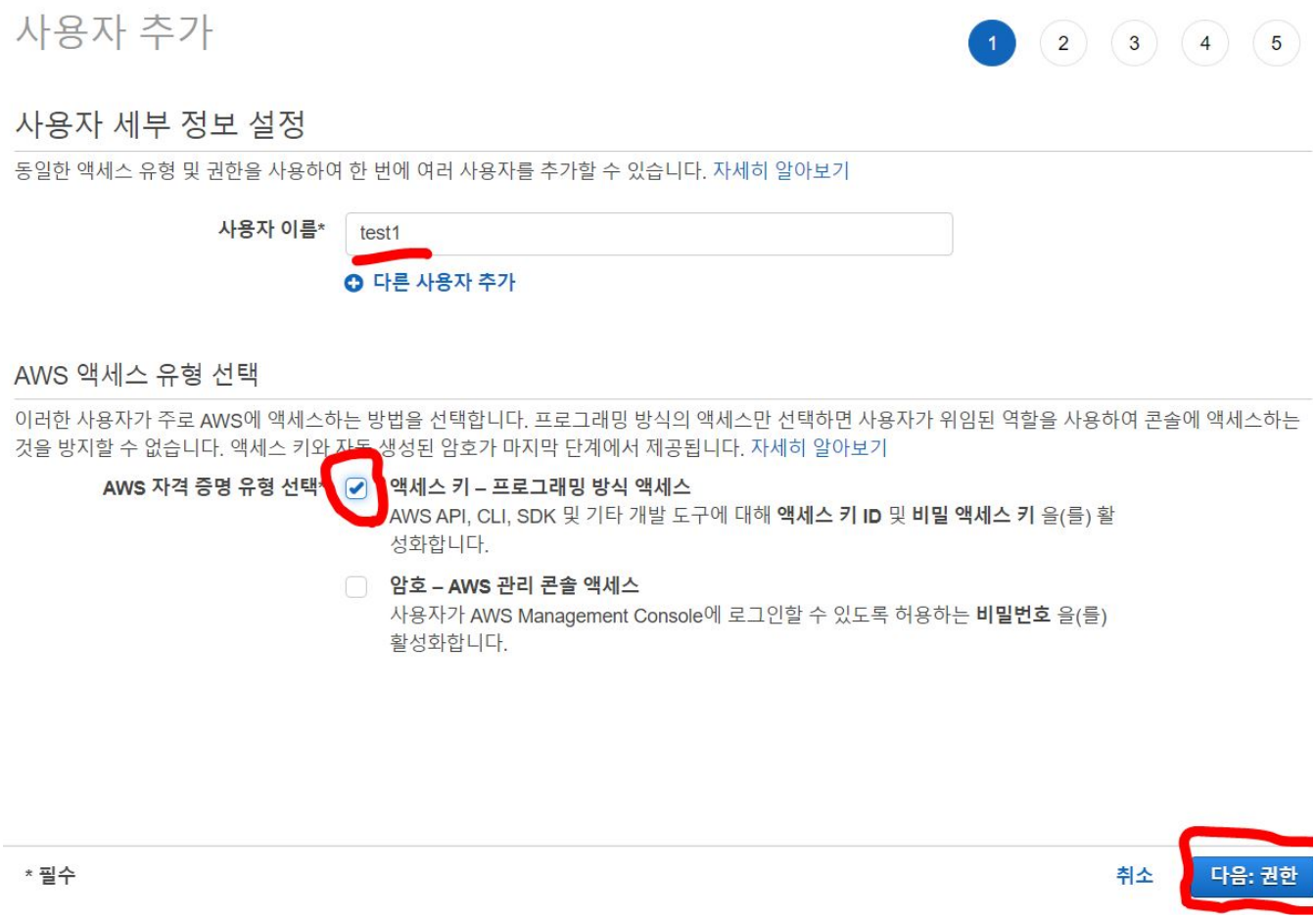


버킷이 생성되었으면, 다시 서비스목록에서 "보안, 자격 증명 및 규정 준수"항목을 클릭하고 우측의 IAM 서비스 항목으로 들어갑니다.

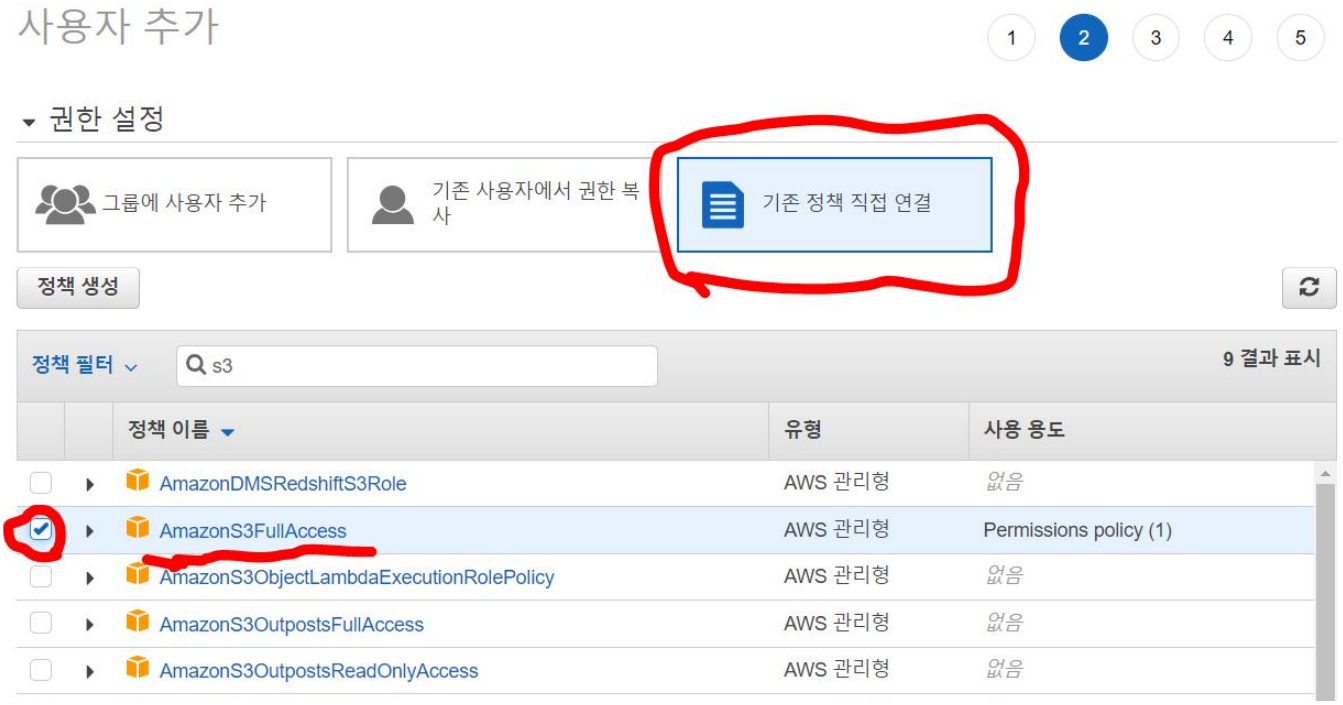
IAM 서비스에 들어가시면, 좌측의 "사용자"탭을 클릭합니다.



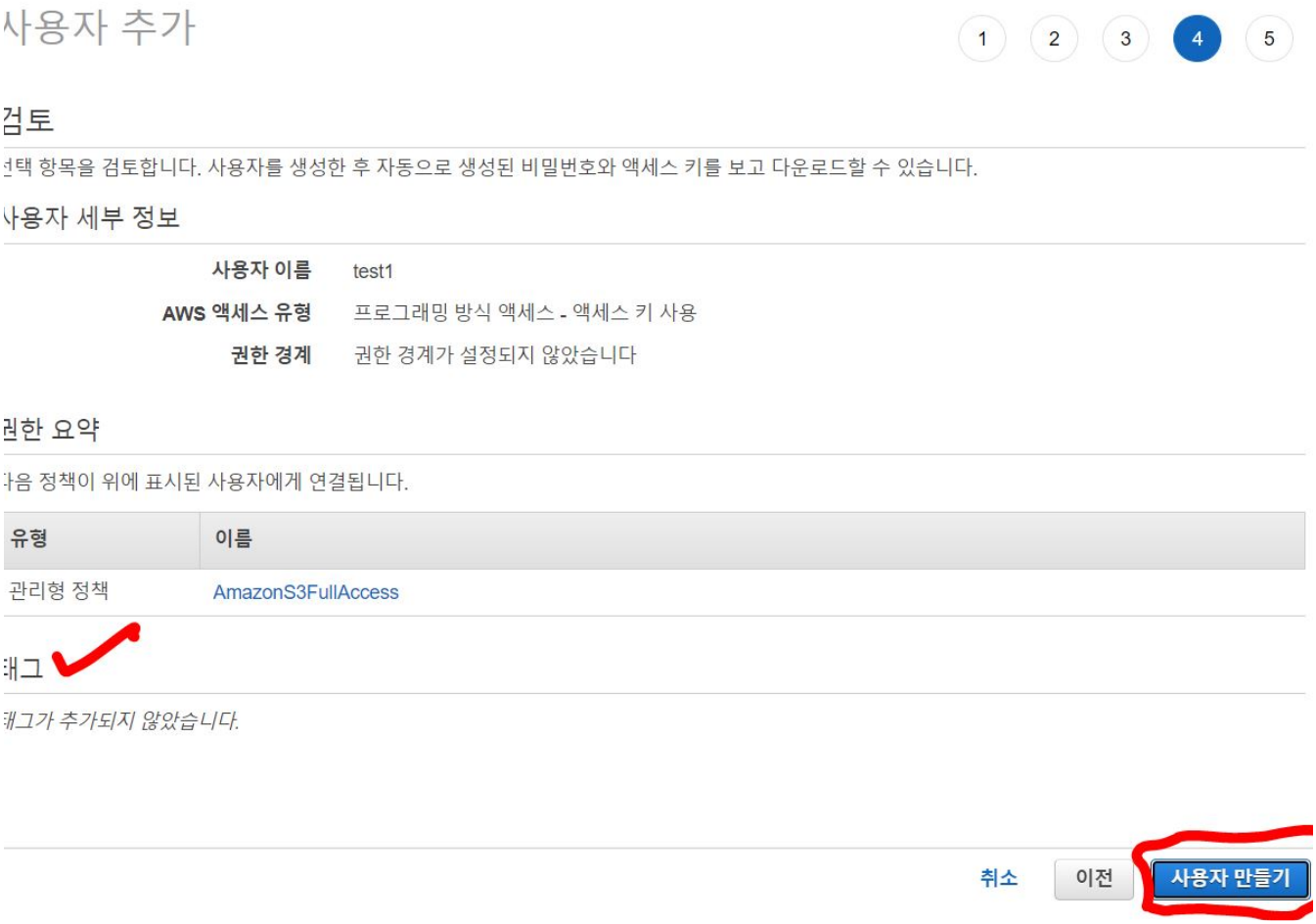
우측의 "사용자 추가"를 클릭합니다.



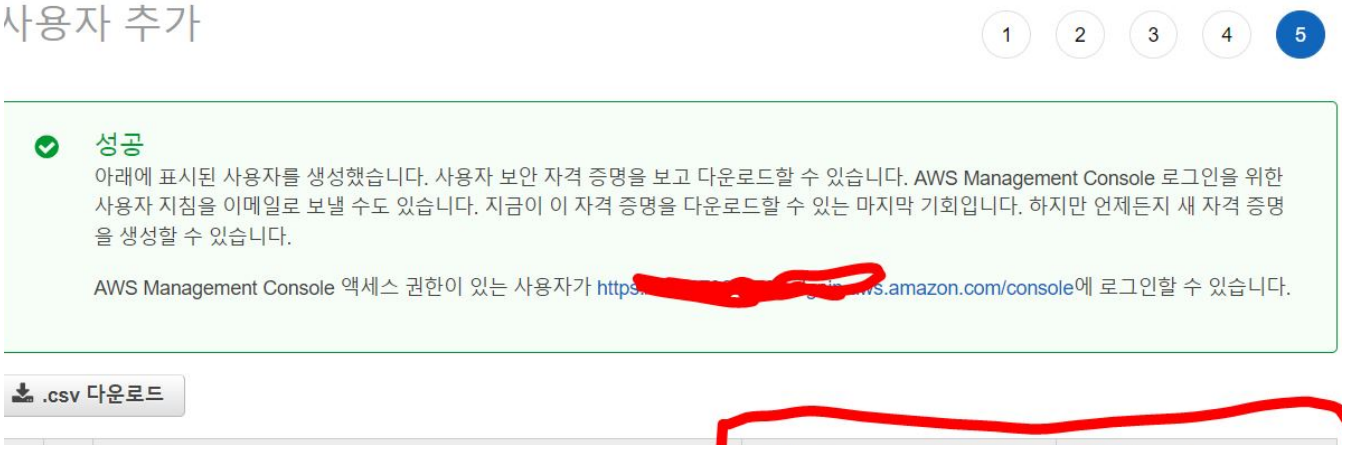
사용자 이름을 입력하고, 중간의 AWS 액세스 유형 선택항목에서 "액세스 키 - 프로그래밍 방식 액세스"를 체크하고 다음 버튼을 클릭합니다.



권한 설정에서 "기존 정책 연결"을 클릭하시고, 정책 필터 내에서 "AmazonS3FullAccess"를 체크하신 뒤, 다음으로 이동합니다. 태그는 입력하지 않고 그대로 다음 항목으로 넘어오시면 됩니다.



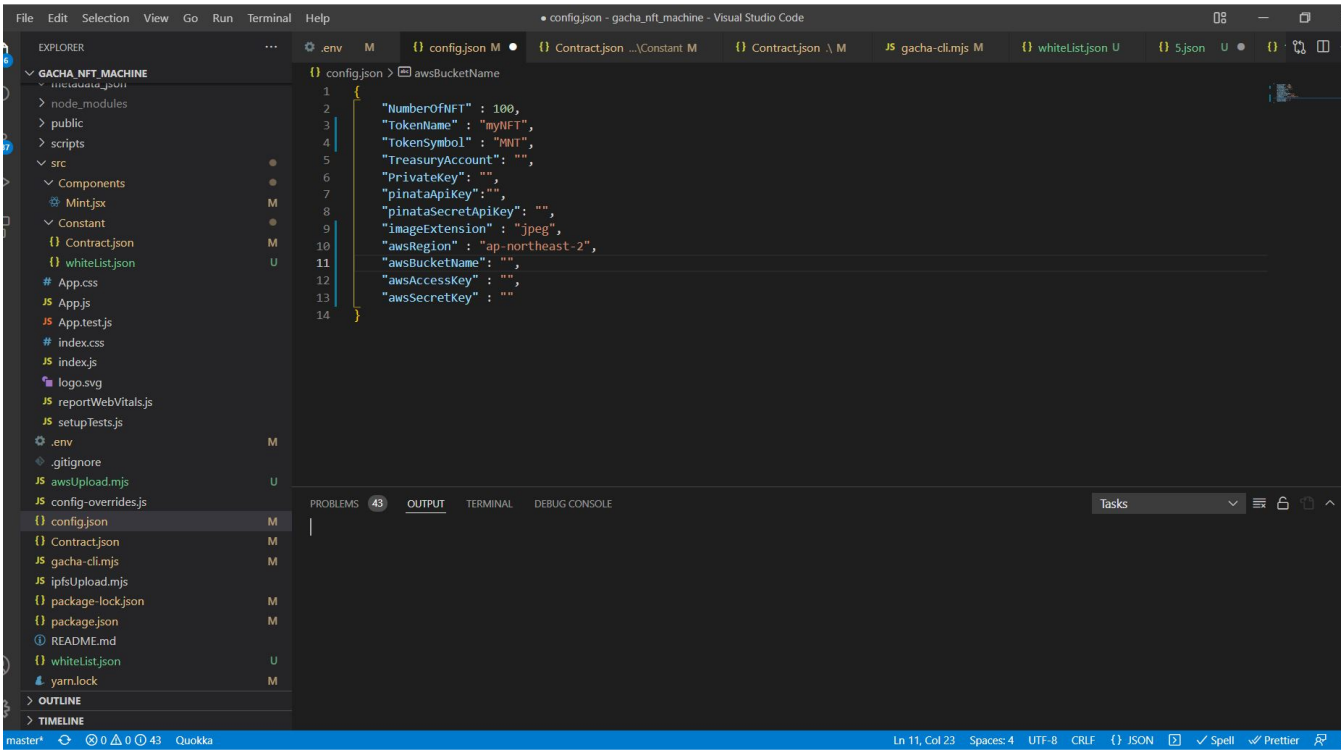
태그는 추가하지 않으셔도 됩니다. 사용자 만들기 버튼을 클릭해 사용자를 만듭니다.



사용자가 추가되면, 중간에 있는 "액세스 키 ID"와 "비밀 액세스 키"를 복사해두시면 됩니다. 비밀 액세스 키는 "표시"를 클릭해 따로 복사해두셔야 합니다. 이 두가지 키와 버킷의 이름, 리전을 이용해 Gacha-CLI가 AWS S3 버킷으로 접근합니다.

Gacha-CLI Config 설정

vscode를 이용해 프로젝트를 열면, 다음과 같이 프로젝트의 파일들이 좌측에 나열됩니다.



프로젝트 내 config.json 파일을 열면 다음과 같은 파일 내용이 나옵니다.

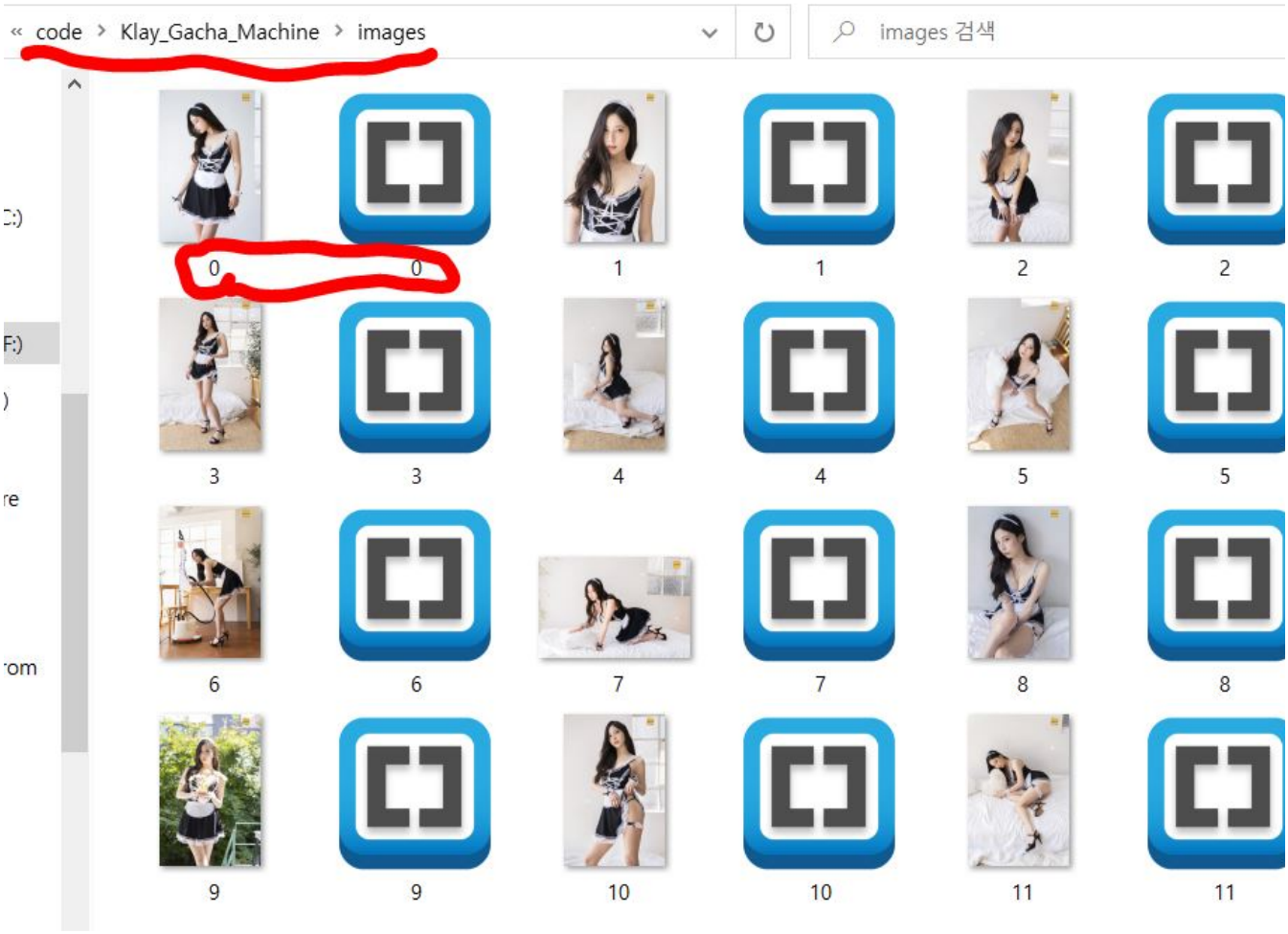
```
{ "NumberOfNFT" : 100, //총 발행갯수
"TokenName" : "myNFT", //NFT의 대표 이름
"TokenSymbol" : "MNT", //NFT의 토큰심볼

"TreasuryAccount": "", //NFT의 발행주소
"PrivateKey": "", //NFT 발행주소의 프라이빗
"pinataApiKey":"", //Pinata의 api key
"pinataSecretApiKey": "", //Pinata의 secret key
"imageExtension" : "jpeg", //이미지 파일의 확장자
"awsRegion" : "ap-northeast-2", //aws s3의 리전이름
"awsBucketName": "", //aws s3의 버킷이름 예) "gacha-machine"
"awsAccessKey" : "", //aws의 액세스키
"awsSecretKey" : "" //aws의secret key }
```

위 내용 중, pinata를 사용하실 분들은 pinataApiKey와 pinataSecretApiKey를 채워야 하고, AWS S3를 사용하실 분들은 awsRegion, awsBucketName, awsAccessKey, awsSecretKey를 채워주시면 됩니다. 그 외의 항목들은 모두 채워주셔야 합니다.

이미지, 메타데이터 옮기기

준비해두신 Image파일과 json 파일들을 한 폴더 안에 옮기신 다음, 해당 폴더를 프로젝트 폴더 내로 이동시킵니다.



파일명과 관련해 필수적으로 지켜야 할 사항

반드시 [이미지파일, json파일]은 각 번호로 매칭시켜둬야 합니다. ex) 0.png, 0.json, 1.png, 1.json

반드시 파일은 0으로 시작해야 합니다. ex)총 1만개라면 0.png~9999.png, 0.json~9999.json

이미지파일, json파일은 꼭 연속된 번호로 되어있어야 합니다.

Gacha CLI Upload

Pinata IPFS를 사용한 방법

Pinata IPFS를 통해 이미지와 메타데이터를 업로드하는 경우

이미지와 메타데이터를 모두 Pinata IPFS에 업로드하는 경우, 다음과 같은 명령어를 통해 이미지와 메타데이터를 Pianata IPFS에 업로드하고, tokenURI를 스마트컨트랙트에 업로드합니다.

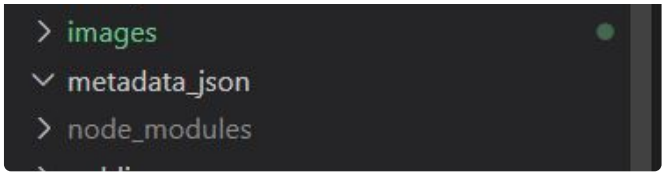
```
node gacha-cli.mjs upload <폴더명> -n <네트워크명> -i

ex) node gacha-cli.mjs upload images -n baobab -i
ex2) node gacha-cli.mjs upload images -n mainnet -i
```

```
F:\code\gacha_nft_machine>node gacha-cli.mjs upload images -n baobab -i
{ network: 'baobab', ipfs: true }
New collection is successfully made.
Number :  0
Path is images/0.jpg
Path is images/0.json
Number :  1
Path is images/1.jpg
Path is images/1.json
Number :  2
Path is images/2.jpg
Path is images/2.json
Number :  3
Path is images/3.jpg
Path is images/3.json
```

```
Number :  9
Path is images/9.jpg
Path is images/9.json
Upload 0-9
Number :  10
Path is images/10.jpg
Path is images/10.json
Number :  11
Path is images/11.jpg
Path is images/11.json
```

업로드가 완료되면, 프로젝트 폴더 내의 .cache폴더가 생성되고, .cache폴더 내 info.json 파일이 생성됩니다.




```
"gachaMachineId": "[redacted]",
"items": [
  {
    "id": 0,
    "link": "ipfs://Qmea3NKHEqB1oie74ipgUnJFgVvbBYVmWkpoFGauYGmv8C",
    "name": "RISA Collection vol.1 #1",
    "onChain": "false"
  },
  {
    "id": 1,
    "link": "ipfs://QmUa38cLkT5BXnNGgXYGeszSFFjbjkj3rguG3xiTtMDdby",
    "name": "RISA Collection vol.1 #2",
    "onChain": "false"
  },
  {
    "id": 2,
    "link": "ipfs://QmSWYx7Zdb9c5MU13yNQvcztEbtsBsj4AHejbVt2Ls612i",
    "name": "RISA Collection vol.1 #3",
    "onChain": "false"
  },
  {
    "id": 3,
    "link": "ipfs://QmQtEcUrQ6gPJ6YcuYyLkdcH4FdzW695iMYeAqRZA6o17n",
    "name": "RISA Collection vol.1 #4",
    "onChain": "false"
  },
```

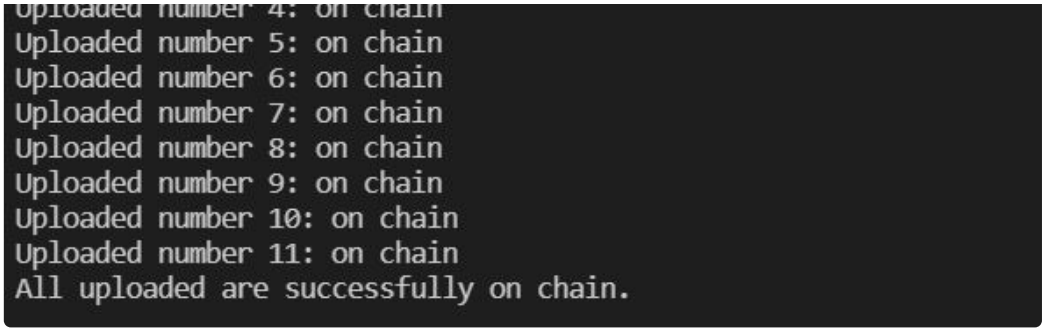

해당 info.json은 프로젝트의 업로드의 캐시파일이며, 각 파일들이 업로드될 때마다 각 NFT의 tokenURI를 저장합니다. 업로드가 중간에 끊기거나, 제대로 업로드가 되지 않은 경우에는 다시 upload 명령어를 입력하여 끊긴 시점부터 다시 업로드를 진행할 수 있습니다.

info.json의 내용을 보면 onChain의 값이 false가 되어있는데, 업로드가 끝나면 다음과 같이 verify 명령어를 통해 업로드가 제대로 되었는지 확인하는 과정이 필요합니다.

```
node gacha-cli.mjs verify -n <네트워크명>

ex) node gacha-cli.mjs verify -n baobab
```

```
F:\code\gacha_nft_machine>node gacha-cli.mjs verify -n baobab
Uploaded number 0: on chain
Uploaded number 1: on chain
Uploaded number 2: on chain
Uploaded number 3: on chain
```



verify가 완료되면, 다음과 같이 info.json의 각 NFT들의 onChain값이 true로 바뀝니다. false인 값은 제대로 업로드가 되지 않은 항목이니 다시 upload 명령어를 입력하시고, verify를 해주시면 됩니다.

AWS S3를 사용한 방법

AWS S3를 사용하고자 한다면 다음 명령어를 통해 이미지와 메타데이터를 AWS S3에 업로드하고, tokenURI를 스마트컨트랙트에 업로드합니다.

```
node gacha-cli.mjs upload <폴더명> -n <네트워크명> -a

ex) node gacha-cli.mjs upload images -n baobab -a
ex2) node gacha-cli.mjs upload images -n mainnet -a
```

나머지 작업은 방법 1과 동일합니다.

이미지가 이미 업로드된 경우

Pinata IPFS

이미지가 이미 저장소에 업로드 되어있는 경우(Pinata ipfs)

Pinata IPFS에 이미 이미지를 업로드하고, 이미지의 URI를 이미 Metadata(json파일)에 삽입해놓은 경우엔, 다음과 같은 명령어를 통해 Metadata를 Pinata IPFS에 업로드하고 tokenURI를 스마트컨트랙트에 업로드하는 작업을 진행합니다.

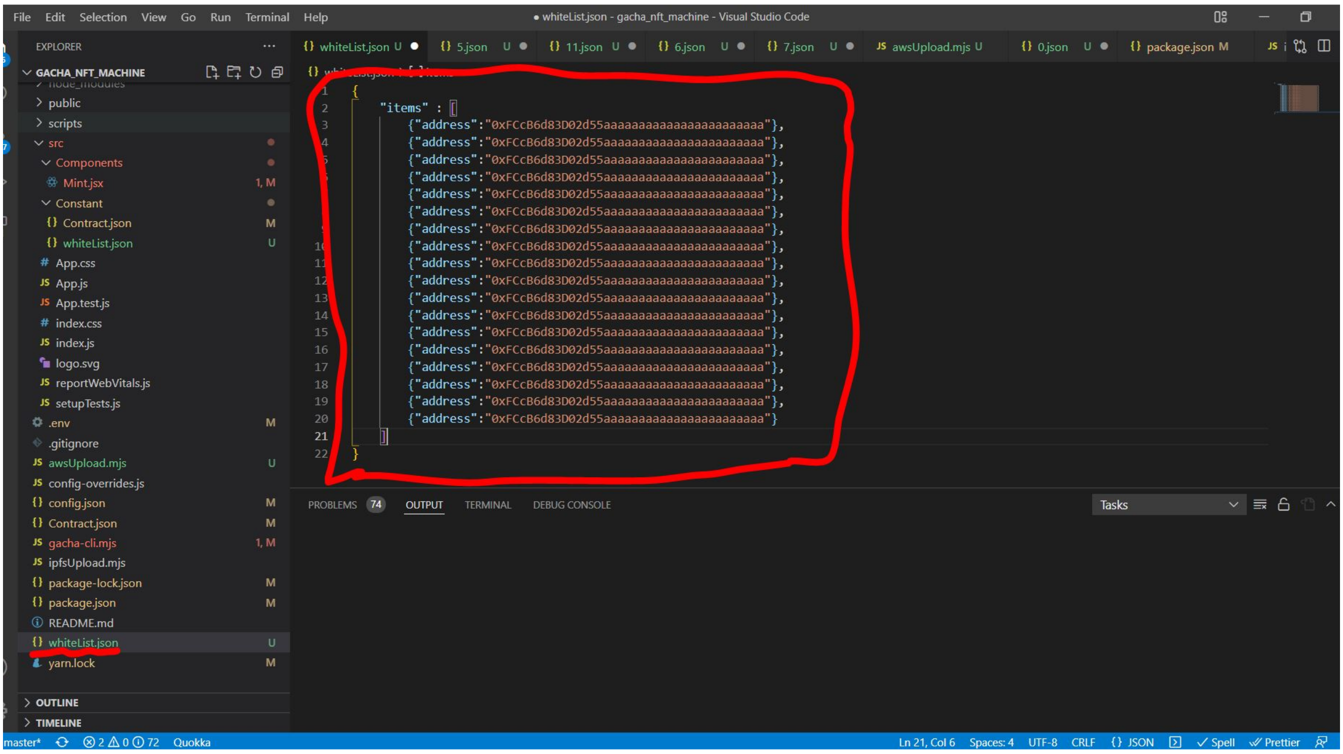
```
node gacha-cli.mjs upload <폴더명> -n <네트워크명>

ex) node gacha-cli.mjs upload images -n baobab
ex2) node gacha-cli.mjs upload images -n mainnet
```

나머지 작업은 방법 1과 동일합니다.

화이트리스트 적용

프로젝트 폴더 내 whiteList.json 파일에 화이트리스트 대상 주소를 입력합니다.



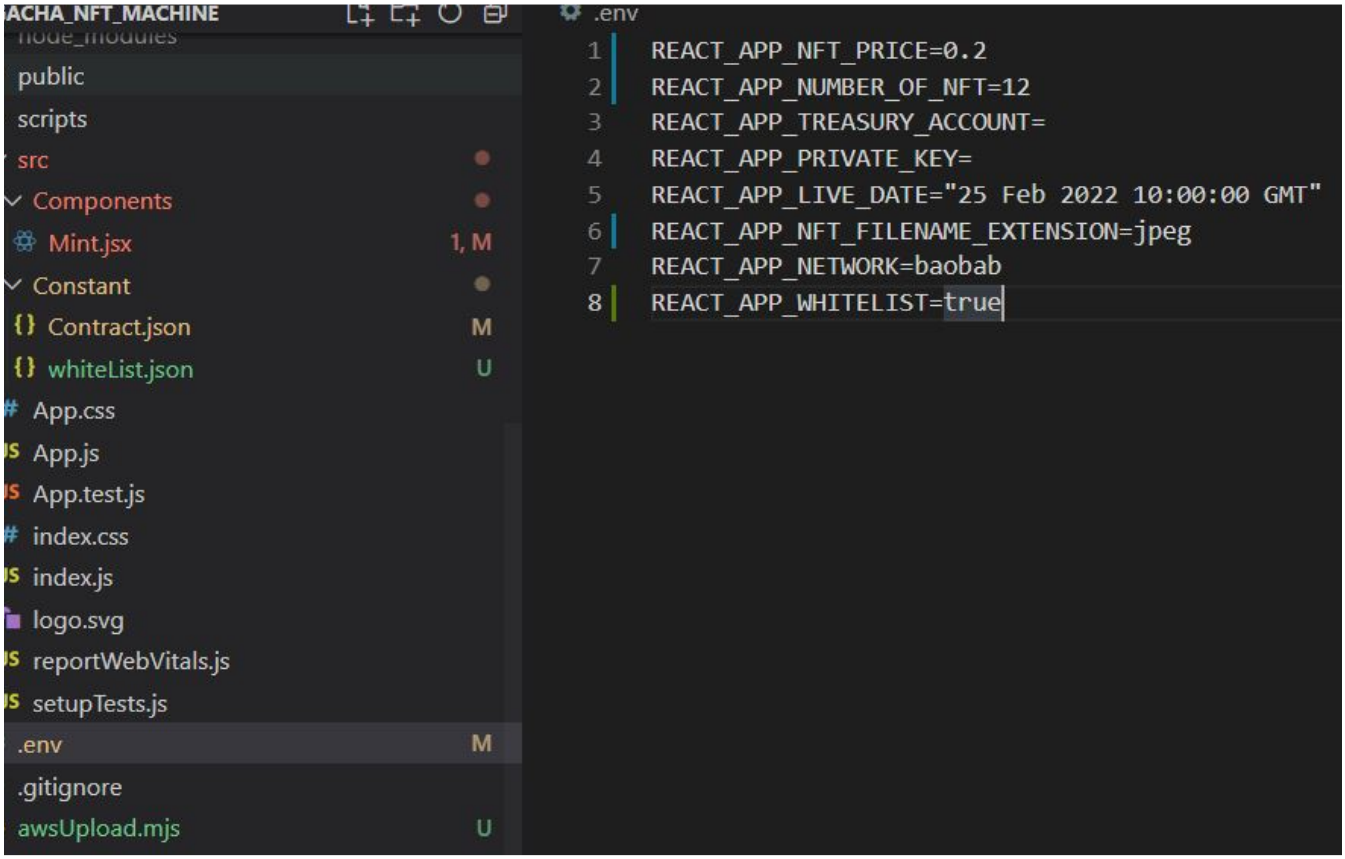
화이트리스트는 json 형식으로 다음과 같이 작성해주셔야 합니다.

```
{
    "items" : [
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"},
        {"address":"0xFCcB6d83D02d55aaaaaaaaaaaaaaaaaaaaaaa"}
    ]
}
```

화이트리스트 명단이 작성되면, 다음의 명령어를 통해 화이트리스트를 업데이트합니다.

```
node gacha-cli.mjs applyWhiteList
```

화이트리스트 업로드가 완료되면, Mint UI에서의 화이트리스트 적용을 위해 .env파일의 화이트리스트 항목을 "true"로 바꿔줍니다.



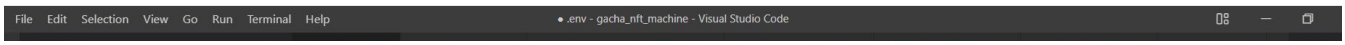
설정이 완료되면 Mint UI에 화이트리스트가 적용이 되고, 민팅 대상이 아닌 주소는 민팅이 불가능합니다.

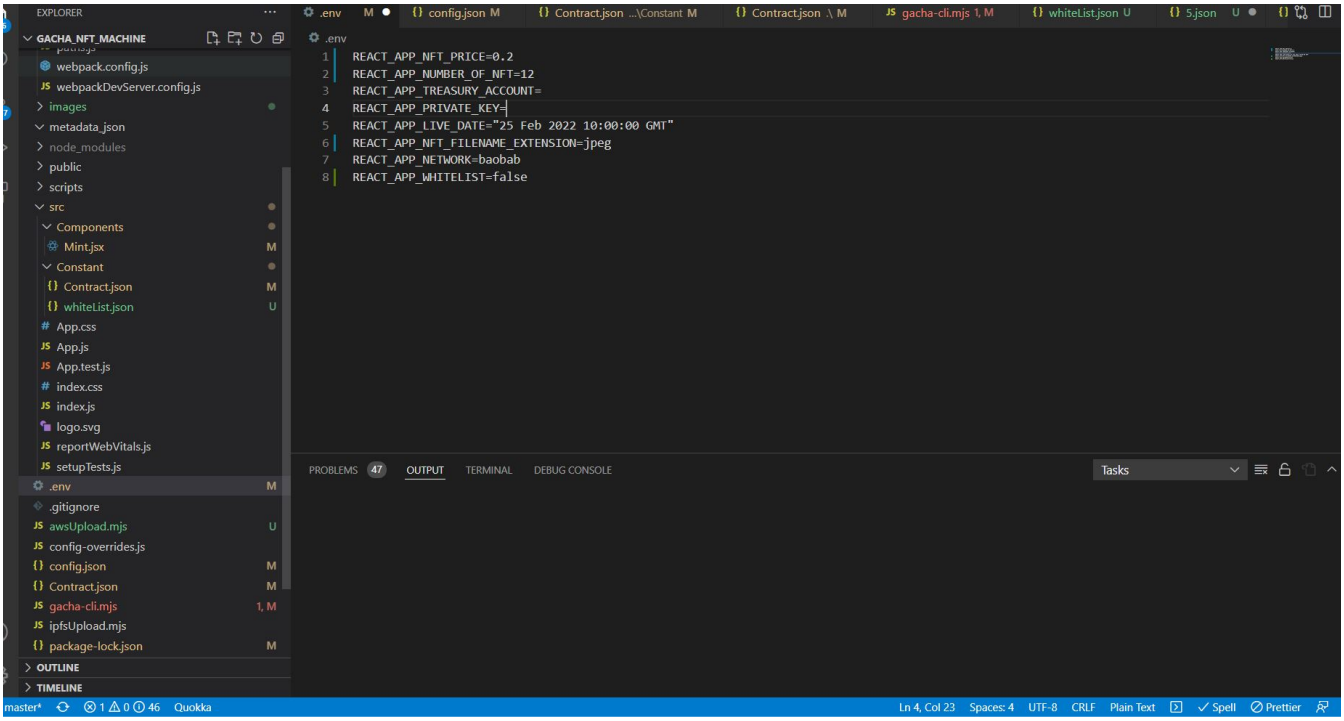
화이트리스트를 전체 물량의 일부만 적용하실 경우(예를 들어, 1만개 민팅물량 중 1천개를 적용하실 경우), 화이트리스트 민팅 일자에 배포하실 때에는 .env파일의 총 물량항목을 해당 일부물량(예를 들어, 1만개가 아니라 1000으로 입력)만 입력하시면 총 1천개만 발행이 됩니다. 이후 퍼블릭 세일에는 총 물량을 다시 전체 물량(예를 들어, 1만개 전체)을 입력하시고, 화이트리스트 항목을 "false"로 입력하신 뒤 배포하시면 됩니다.

Mint-UI

Mint UI 환경파일 설정

프로젝트 내 .env파일을 열면 다음과 같은 내용이 나옵니다. (.env파일이 없는 경우, 파일 이름을 '.env'로 새로 만들어주세요)






```
REACT_APP_NFT_PRICE=0.2 //민팅 가격(0.1Klay 이상으로 설정하셔야 합니다)
REACT_APP_NUMBER_OF_NFT=12 //총 발행량 or 회당 발행량(화이트리스트의 경우)
REACT_APP_TREASURY_ACCOUNT= //NFT 발행 주소
REACT_APP_PRIVATE_KEY= //NFT 발행 주소의 프라이빗 키
REACT_APP_LIVE_DATE="25 Feb 2022 10:00:00 GMT" // NFT 발행을 시작할 날짜
REACT_APP_NFT_FILENAME_EXTENSION=jpeg //NFT 이미지의 형식
REACT_APP_NETWORK=baobab //네트워크명 baobab or mainnet
REACT_APP_WHITELIST=false //화이트리스트 사용 여부
```

Mint UI 테스트

.env파일의 설정이 끝나면, 다음과 같은 명령어를 통해 Mint UI를 로컬 환경에서 테스트해볼 수 있습니다.

```
npm start
```

Kaikas 지갑으로 UI에 연결할 수 있습니다.

주의) 해당 버튼을 누르면 실제로 NFT가 발행됩니다. 테스트는 baobab 네트워크에서만 진행하시고, mainnet 으로 작업하실 경우 발행버튼 클릭에 주의를 기울여 주십시오.

Contact Us

Discord

Github