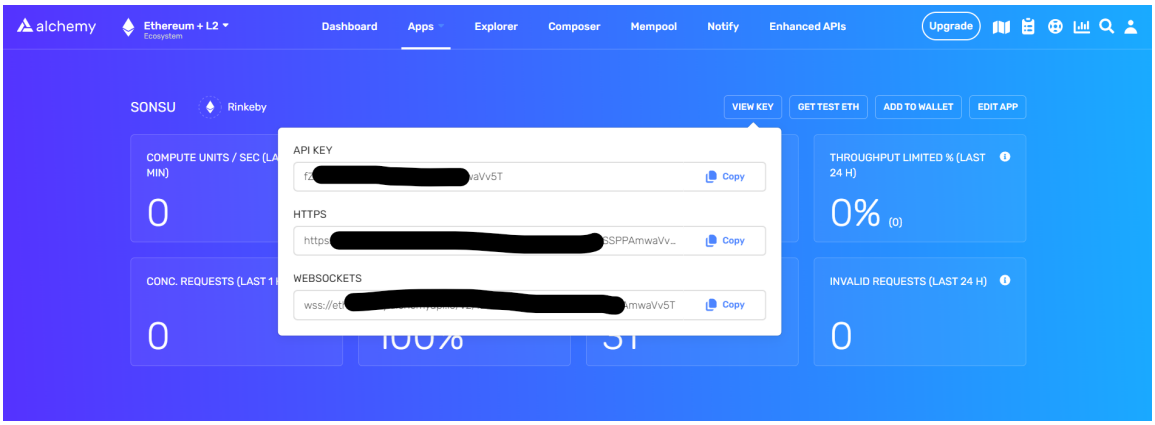
JS를 사용한 NFT 생성

1. alchemy.io 계정 생성 후 APP 생성하기
    - Environment는 Development, Chain은 Ethereuam, Network는 `rinkeby`
    - Free plan, Capped Capacity option을 선택(앱을 처음 만드는 경우에만 선택)
    - 그리고 생성된 앱에서 View Key를 선택하면
        - API KEY
        - HTTPS
        - WEBSOCKETS

    가 보인다. 각 변수들은 여기 값들을 집어넣어주면 된다.



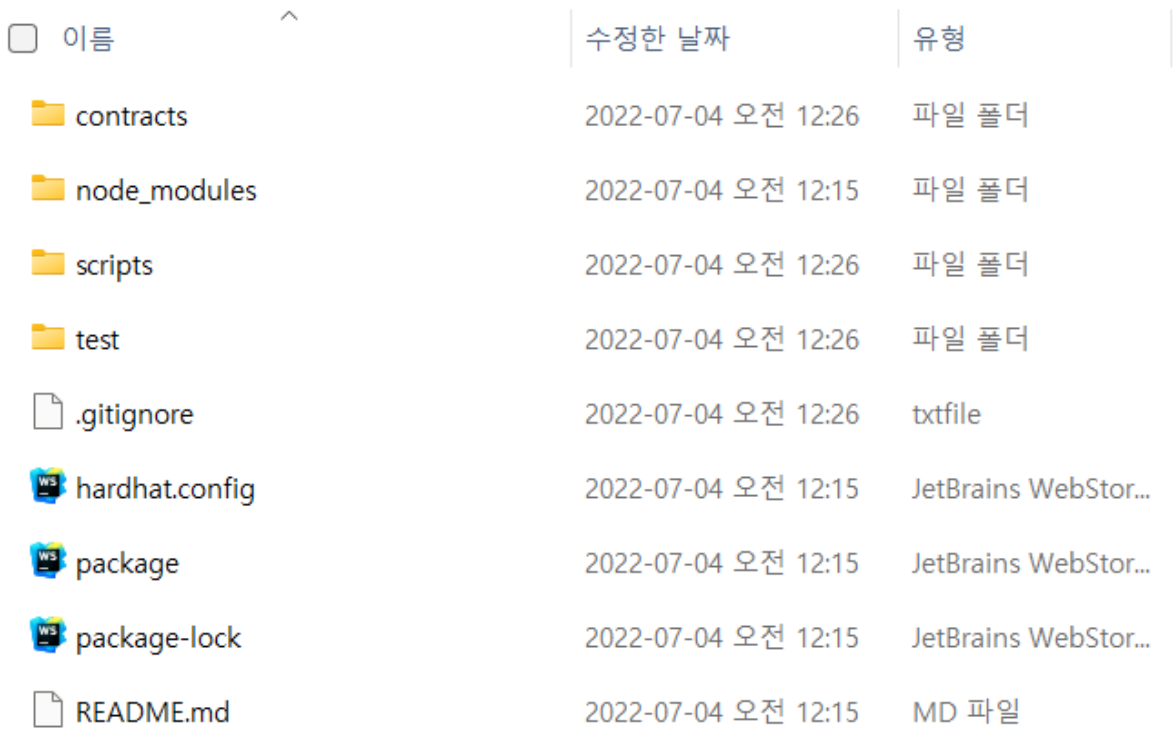
2. Wallet 생성
    - 그냥 메타마스크에서 지갑 생성해서 사용해도 상관없다.
    - 직접 만들 생각이라면 https://www.notion.so/Eth-c8b71dbd8101455e942b83c3a4259020 참고

3. Hardhat 설치 후 실행

```
npm install -D hardhat

npx hardhat
```

hardhat을 설치하고 실행하면 이런 화면이 뜨는데 그냥 엔터 엔터 눌러주면 프로젝트가 생성된다. 그러면



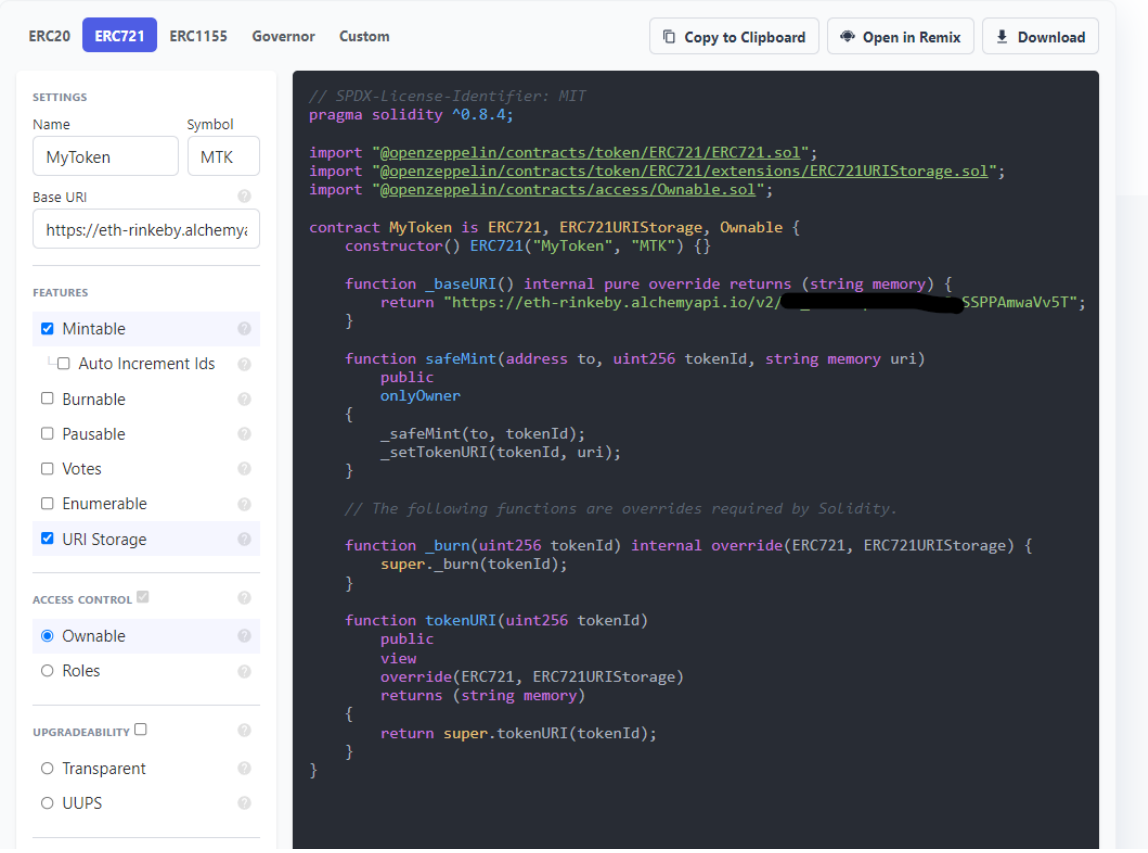
이름	수정한 날짜	유형
contracts	2022-07-04 오전 12:26	파일 폴더
node_modules	2022-07-04 오전 12:15	파일 폴더
scripts	2022-07-04 오전 12:26	파일 폴더
test	2022-07-04 오전 12:26	파일 폴더
.gitignore	2022-07-04 오전 12:26	txtfile
hardhat.config	2022-07-04 오전 12:15	JetBrains WebStor...
package	2022-07-04 오전 12:15	JetBrains WebStor...
package-lock	2022-07-04 오전 12:15	JetBrains WebStor...
README.md	2022-07-04 오전 12:15	MD 파일

이런식으로 `contracts`, `scripts`, `test` 폴더와 `hardhat.config`가 생성된 것을 볼 수 있다.

4. 스마트 컨트랙트 개발을 위한 `OpenZeppelin` 라이브러리 설치

```
npm i @openzeppelin/contracts@4.0.0
```

5. Contracts 폴더에 ***.sol 저장
    - sol 파일의 이름은 임의로 아무거나 정해주자. 딱히 상관없다.
    - 나는 처음 접할땐 이 코드들을 한땀한땀 다 따라쳤는데 https://wizard.openzeppelin.com/#erc721 이런 사이트가 있는 모양.
    - ~~자동완성도 안되는 바람에 따라친다고 개고생했는데 허탈하네~~



설정값 참고

- [ ] ERC721 설정
- [ ] Name(본인 계약명)과 Symbol(계약명 약어)을 알아서 설정
- [ ] Base URI는 Alchemy 사이트의 `View Key - HTTPS` 를 통해 가져오기
- [ ] Mintable 체크
- [ ] URI Storage 체크
- [ ] Ownable 체크

⇒ 각각 무엇을 뜻하는지는 알아서 차후 정리하는 것으로 하고…

6. dotenv 패키지 설치 (Key Secure)

```
npm i dotenv
```

- `node.js`에서의 환경변수 파일인 .env 파일을 사용하기 위핸 패키지
- 사실 연습단계에서는 굳이 쓸 필요 있나 싶긴 한데, 일단 실습 해볼수 있는 건 다 해보자.

7. process.env 파일 생성

```
METAMASK_PRIVATE_KEY="이더 지갑 프라이빗키 붙여 넣으세요"
API_URL="Alchemy.io에서 생성한 앱의 HTTPS를 붙여 넣으세요"
```

- 반드시 root 폴더에 생성해야 한다.
- 여기서 이더 지갑의 프라이빗 키를 붙여넣을 때는 맨 앞의 0x를 빼고 그 뒤부터 붙여넣자. 자세한건 `hardhat.config.js` 파일 생성할때 설명.

8. Ether.js 설치

```
npm i -D @nomiclabs/hardhat-ethers ethers@^5.0.0
```

9. `hardhat.config.js`에 디펜던시 추가
   - 초기 `hardhat.config.js` 상태

```
require("@nomiclabs/hardhat-waffle");

// This is a sample Hardhat task. To learn how to create your own go to
// https://hardhat.org/guides/create-task.html
task("accounts", "Prints the list of accounts", async (taskArgs, hre) => {
  const accounts = await hre.ethers.getSigners();

  for (const account of accounts) {
    console.log(account.address);
  }
});

// You need to export an object to set up your config
// Go to https://hardhat.org/config/ to learn more

/**
 * @type import('hardhat/config').HardhatUserConfig
 */
module.exports = {
  solidity: "0.8.4",
};
```

   - 디펜던시를 추가해서 아래와 같은 상태로 만들어준다

```
require("@nomiclabs/hardhat-waffle");
```

```
// This is a sample Hardhat task. To learn how to create your own go to
// https://hardhat.org/guides/create-task.html
task("accounts", "Prints the list of accounts", async (taskArgs, hre) => {
  const accounts = await hre.ethers.getSigners();

  for (const account of accounts) {
    console.log(account.address);
  }
});

// You need to export an object to set up your config
// Go to https://hardhat.org/config/ to learn more

/**
 * @type import('hardhat/config').HardhatUserConfig
 */

require('dotenv').config({path:__dirname+'/process.env'})
require("@nomiclabs/hardhat-ethers");

const {API_URL, METAMASK_PRIVATE_KEY} = process.env;

module.exports = {
  solidity: "0.8.2",
  defaultNetwork: "rinkeby",
  networks: {
    hardhat: {},
    rinkeby: {
      url: API_URL,
      accounts: [`0x${METAMASK_PRIVATE_KEY}`]
      // url:"https://eth-rinkeby.alchemyapi.io/v2/fZ_RlZZ0MpckxkK9EvQuSSPPAmwaVv5T",
      // accounts: [`0x71daa457c43a8885ba33922d381b383373b9686d685b7a5c85f5d3ebb37a6360`]
    }
  }
};
```

- process.env를 못찾는 경우에는 직접 값을 박아넣은 하드코딩을 진행해 주어야 한다.
- 처음에는 `require('dotenv').config();` 를 임포트할때 경로설정을 따로 해주지 않아 계속 process.env를 찾지 못하는 에러가 발생하여 애를 좀 먹었다.
  - `__dirname` = 현재 경로를 알려주는 변수
- 7번에서 프라이빗키 앞의 0x를 빼고 env파일에 넣어주라는 이유는 `` `0x${}` `` 의 방식으로 변수를 받아왔기 때문이다.

10. waffle 설치

- `require("@nomiclabs/hardhat-waffle");` 를 임포트 시켜야하니, 이 패키지도 설치한다.

```
npm i @nomiclabs/hardhat-waffle
```

10. hardhat 컴파일

```
npx hardhat compile

// 이미 한번 컴파일이 되었거나
// Nothing to compile 문구가 뜰 경우에는
// npx hardhat compile --force 를 입력하여 강제로 재컴파일을 시켜준다.

// npx hardhat compile clean 를 통해 캐시를 초기화시켜주는것도 방법인데...
// 전엔 먹히더니 지금은 안먹히니 일단 패스
```

- 컴파일 과정이 잘 실행되었다면

이런식으로 컴파일 성공 문자가 뜬다.

11. 컨트랙트 배포

- scripts 폴더에 `deploy.js` 파일을 생성하고 코드를 작성한다.

```
async function main() {
    const [deployer] = await ethers.getSigners();
    console.log("Deploying contracts with the account:", deployer.address);

    console.log("Account balance:", (await deployer.getBalance()).toString());
    const TPM = await ethers.getContractFactory("SonsuNFT");
    // "SonsuNFT" = .sol 파일에서 작성한 본인 계약명

    // Start deployment, returning a promise that resolves to a contract object
    const tpm = await TPM.deploy();
    console.log("Contract deployed to address:", tpm.address);
}

main()
    .then(() => process.exit(0))
    .catch(error => {
        console.error(error);
        process.exit(1);
    });
```

▼ 참고

▼ hardhat은 공식적으로 배포시스템을 구현하는 플러그인이 존재하지 않기 때문에 `스크립트를 통한 배포`나 `hardhat 커뮤니티에서 제작한 플러그인`의 사용을 권장하고 있다.

▼ `hardhat 커뮤니티 플러그인`은 contracts 폴더 내에 있는 `Greeter.sol` 을 통해 테스트 스마트 컨트랙트를 작성하고 scripts 폴더 내에 있는 `sample-scripts.js` 를 통해 테스트 배포를 진행한다.

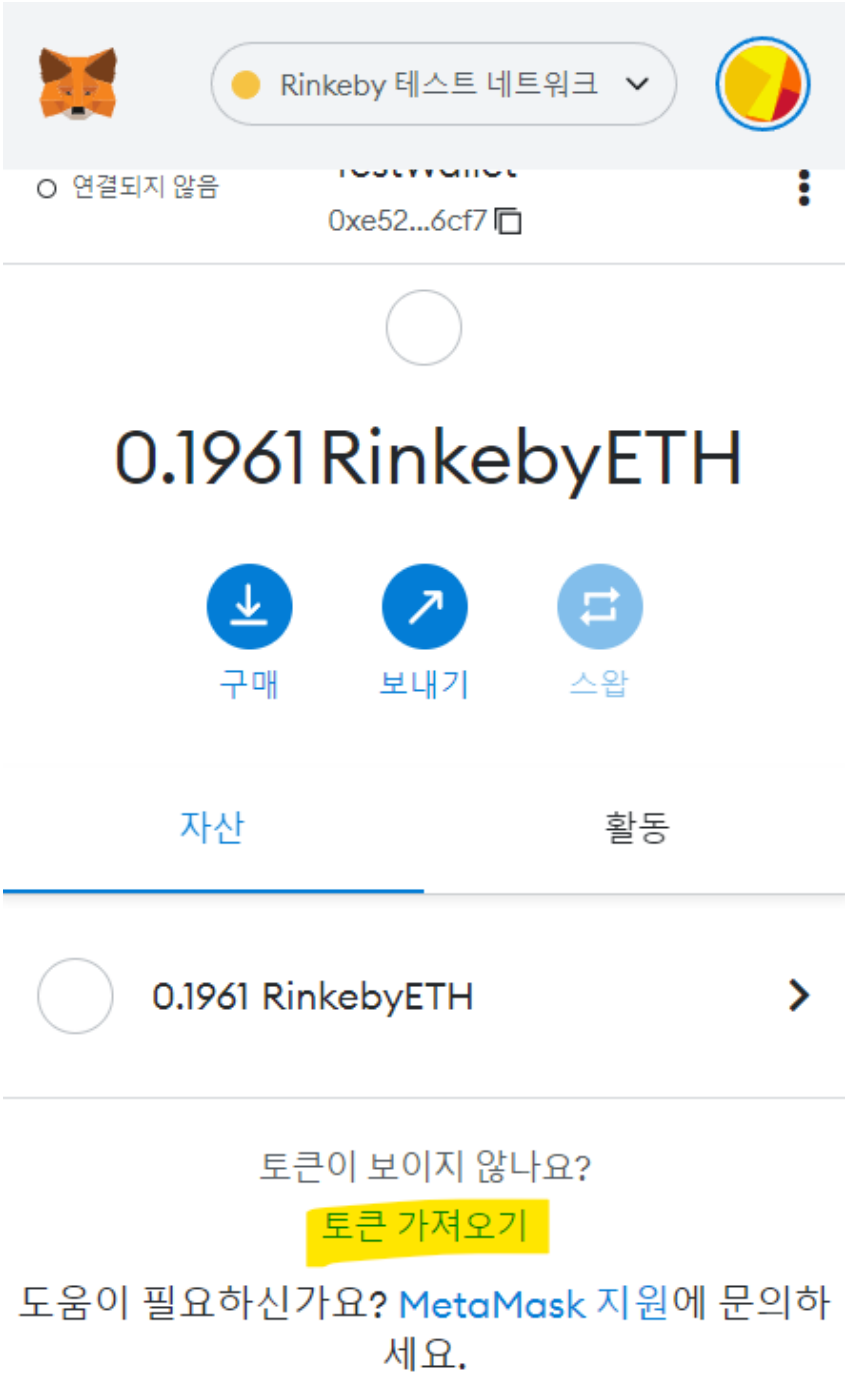
▼ 상세한 내용은 공식문서 https://hardhat.org/guides/deploying 참고

- 배포 실행

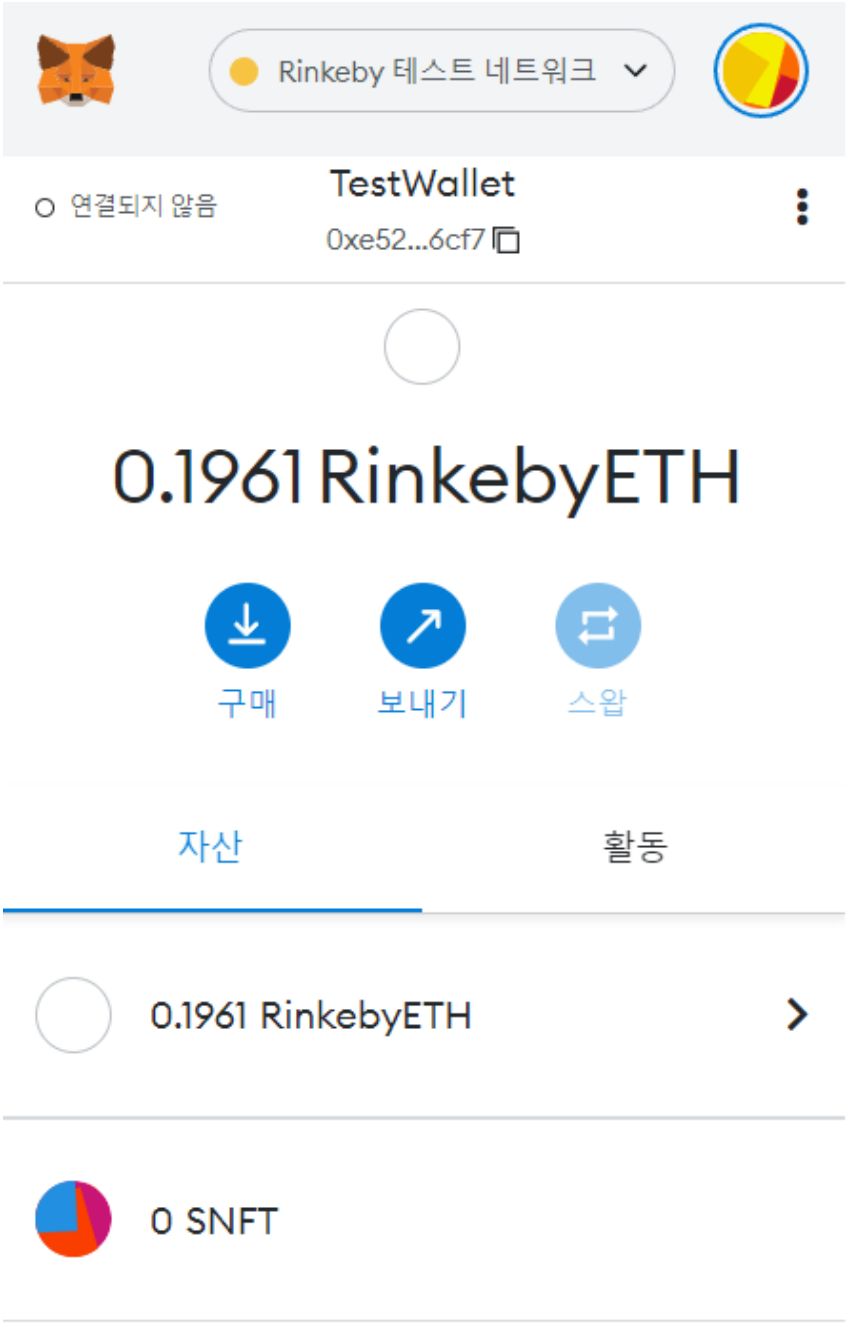
```
npx hardhat run scripts/deploy.js --network rinkeby

// 물론 이더리움 잔액 있어야함.
// 집어넣은 메타마스크 주소를 이더스캔에서 찍어보면 가스피가 빠져나간 것을 확인 가능.

// Contract deployed to address: 0x####  <= 여기 주소를 카피해서
// 메타마스크 토큰 생성란에 붙이기
```


Rinkeby 테스트 네트워크
연결되지 않음
0xe52...6cf7
0.1961 RinkebyETH
구매
보내기
스왑
자산
활동
0.1961 RinkebyETH
토큰이 보이지 않나요?
토큰 가져오기
도움이 필요하신가요? MetaMask 지원에 문의하세요.

12. 토큰 생성 확인


Rinkeby 테스트 네트워크
연결되지 않음
TestWallet
0xe52...6cf7
0.1961 RinkebyETH
구매
보내기
스왑
자산
활동
0.1961 RinkebyETH
0 SNFT

뽕~ 잘나왔쥬?