HERE 포팅 매뉴얼

⚙️ SSAFY 8th 특화프로젝트_B209 HERE 포팅 매뉴얼 ⚙️

목차

기술 스택 버전

🤖 프론트엔드

- React: 18.2.0
- Next.js: 13.2.3
- react-query: 3.39.3
- Redux Toolkit: 1.9.3
- Typescript: 4.9.5
- Tailwind CSS: 3.2.7
- Web3: 1.8.2

백엔드

- java : openJDK version 11.0.18
- mySQL : 8.0.31
- springBoot : 2.7.10
- spring Swagger : 2.9.2
- queryDsl : 5.0.0

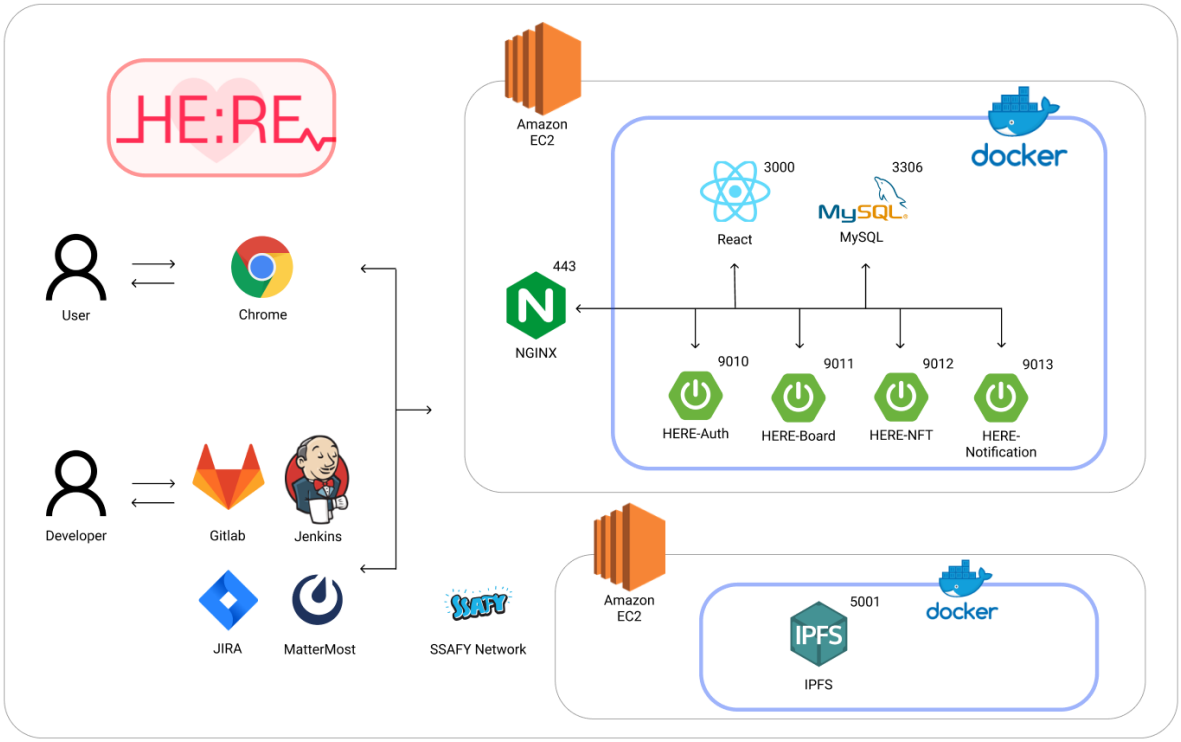
스마트 컨트랙트

- truffle : 5.4.33
- solidity : 0.8.4
- ipfs : 0.10.0

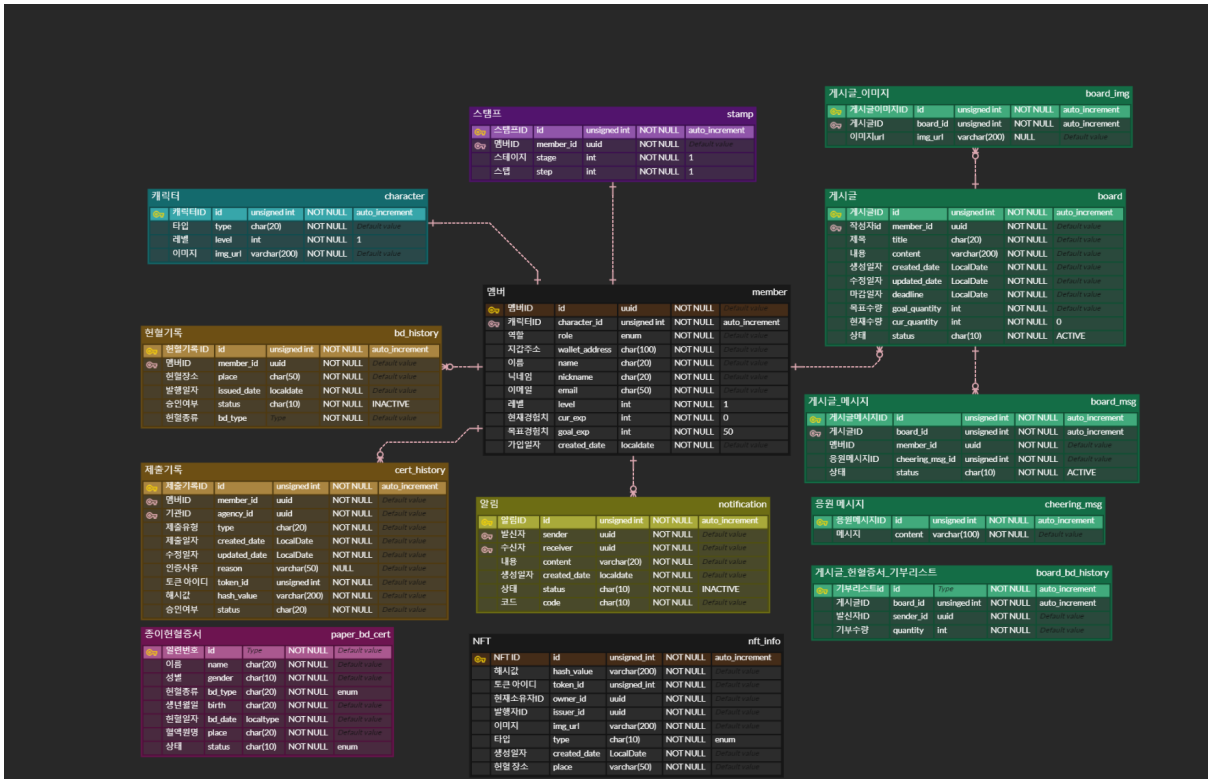
인프라

- docker : 23.0.1
- docker-compose : 1.29.2
- Jenkins : 2.387.1

아키텍처



ERD



Java 설치

```
sudo apt-get update && sudo apt-get upgrade
sudo apt-get install openjdk-11-jdk
java -version
```

Nginx 설치

- Nginx 설치 및 버전 확인

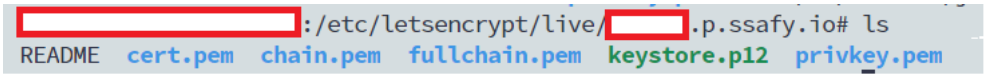
```
apt-get install nginx
nginx -v
```

- Nginx 중지

```
systemctl stop nginx
```

- SSL 인증서 발급

```
apt-get install letsencrypt # Let's Encrypt 설치

sudo letsencrypt certonly --standalone -d [도메인] # 인증서 적용 및 pem키 발급

cd /etc/letsencrypt/live/[도메인] # 발급 경로 확인

vim custom.conf # conf 파일 생성
```



- .conf 파일 작성

```
server {
        # 프론트 연결(포트 번호는 본인의 프론트 포트번호를 입력)
        location /{
                proxy_pass https://localhost:3000;
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Typ
                add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';

                # https websocket
                proxy_set_header       Upgrade $http_upgrade;
                proxy_set_header       Connection "upgrade";
                proxy_set_header Host $host;
        }


    listen 443 ssl http2; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate /etc/letsencrypt/live/{도메인 주소}/fullchain.pem; # managed by Certbot
    # 도메인 이름을 써줘야함
    ssl_certificate_key /etc/letsencrypt/live/{도메인 주소}/privkey.pem; # managed by Certbot
    # include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    # ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    # 도메인 이름을 입력
    if ($host = {도메인 주소}) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name {도메인 주소};
    return 404; # managed by Certbot
}
```

- 심볼링 연결 테스트 및 상태 확인

```
# 심볼릭 링크 연결
sudo ln -s /etc/nginx/sites-available/[파일명].conf /etc/nginx/sites-enabled/[파일명].conf

sudo nginx -t # nginx 테스트

sudo systemctl restart nginx # nginx 재시작

sudo systemctl status nginx # nginx 상태 확인
```

- keystore.p12 파일 생성

```
cd /etc/letsencrypt/live/[도메인주소]/
openssl pkcs12 -export -in fullchain.pem -inkey privkey.pem -out keystore.p12 -name [이름] -CAfile chain.pem -caname root
```

Docker 설치

- apt-transport-https : 패키지 관리자가 https를 통해 데이터 및 패키지에 접근할 수 있도록 한다.
- ca-certificates : certificate authority에서 발행하는 디지털 서명. SSL 인증서의 PEM 파일이 포함되어 있어 SSL기반 앱이 SSL 연결이 되어 있는지를 확인할 수 있다.
- curl : 특정 웹사이트에서 데이터를 다운로드 받을 때 사용
  software-properties-common : PPA(Personal Package Archive)를 추가하거나 제거할 때 사용한다

```
sudo apt update && sudo apt-get upgrade
sudo apt install apt-transport-https ca-certificates
```

```
sudo apt install curl gnupg-agent software-properties-common
```

- Docker 공식 GPG 키 추가

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
```

- stable repository를 세팅하기 위한 명령어 실행
- add-apt-repository: PPA저장소를 추가해준다. apt 리스트에 패키지를 다운로드 받을 수 있는 경로가 추가됨

```
sudo add-apt-repository \
"deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
```

- 가장 최신 버전의 Docker 엔진을 설치한 후, 버전 확인

```
sudo apt update
sudo apt install docker-ce docker-ce-cli containerd.io
docker -v
```

- docker-compose 설치

```
curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-c
chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

MySQL 설치

- MySQL Docker Image 다운로드, 태그 버전 생략하면 최신 버전 다운로드

```
docker pull mysql:8.0.31
```

- MySQL Docker 컨테이너 생성 및 실행

```
docker run --name mysql-container -e MYSQL_ROOT_PASSWORD=B209HERE -v mysql-volume:/var/lib/mysql -d -p 3306:3306 mysql:8.0.31
```

- MySQL docker 컨테이너 접속

```
docker exec -it mysql-container bash

mysql -u root -p
Enter password: B209HERE
```

Jenkins 설치

- 폴더 생성

```
sudo mkdir -p /home/ubuntu/jenkins
```

- Docker에 Jenkins 설치

```
sudo docker run --name jenkins -d -p 9999:9999 -p 50000:50000 -v /home/ubuntu/jenkins:/var/jenkins_home -v /var/run/docker.sock:/var/r
```

- jenkins container 접속

```
sudo docker exec -it jenkins /bin/bash
```

- jenkins 접속

```
http://<your-aws-domain>:<jenkins port> 접속 후 admin password 입력
ex) http://j8b209.p.ssafy.io:9999
```

- jenkins admin password 확인

```
cat /var/jenkins_home/secrets/initialAdminPassword
```

- jenkins 내에 docker 설치

```
# Old Version Remove
apt-get remove docker docker-engine docker.io containerd runc

## Setup Repo
apt-get update

apt-get install ca-certificates curl gnupg lsb-release

mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" | tee /etc/apt/sources.list.d/docker.list > /dev/null

## - Install Docker Engine
apt-get update

apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Ipfs 설치

- docker-compose.yml

```
version: '3'

services:
  ipfs0:
    container_name: ipfs0
    image: ipfs/go-ipfs:latest
    ports:
      - "4001:4001" # ipfs swarm - expose if needed/wanted
      - "5001:5001" # ipfs api - expose if needed/wanted
      - "8080:8080" # ipfs gateway - expose if needed/wanted
    volumes:
      - ./compose/ipfs0:/data/ipfs

  cluster0:
    container_name: cluster0
    image: ipfs/ipfs-cluster:latest
```

```
    depends_on:
      - ipfs0
    environment:
      CLUSTER_PEERNAME: cluster0
      CLUSTER_SECRET: # From shell variable if set
      CLUSTER_IPFSHTTP_NODEMULTIADDRESS: /dns4/ipfs0/tcp/5001
      CLUSTER_CRDT_TRUSTEDPEERS: '*' # Trust all peers in Cluster
      CLUSTER_RESTAPI_HTTPLISTENMULTIADDRESS: /ip4/0.0.0.0/tcp/9094 # Expose API
      CLUSTER_MONITORPINGINTERVAL: 2s # Speed up peer discovery
    ports:
          - "127.0.0.1:9094:9094"
          # - "9096:9096" # Cluster IPFS Proxy endpoint
    volumes:
      - ./compose/cluster0:/data/ipfs-cluster

  ipfs1:
    container_name: ipfs1
    image: ipfs/go-ipfs:latest
    volumes:
      - ./compose/ipfs1:/data/ipfs

  cluster1:
    container_name: cluster1
    image: ipfs/ipfs-cluster:latest
    depends_on:
      - ipfs1
    environment:
      CLUSTER_PEERNAME: cluster1
      CLUSTER_SECRET:
      CLUSTER_IPFSHTTP_NODEMULTIADDRESS: /dns4/ipfs1/tcp/5001
      CLUSTER_CRDT_TRUSTEDPEERS: '*'
      CLUSTER_MONITORPINGINTERVAL: 2s # Speed up peer discovery
    volumes:
      - ./compose/cluster1:/data/ipfs-cluster

  ipfs2:
    container_name: ipfs2
    image: ipfs/go-ipfs:latest
    volumes:
      - ./compose/ipfs2:/data/ipfs

  cluster2:
    container_name: cluster2
    image: ipfs/ipfs-cluster:latest
    depends_on:
      - ipfs2
    environment:
      CLUSTER_PEERNAME: cluster2
      CLUSTER_SECRET:
      CLUSTER_IPFSHTTP_NODEMULTIADDRESS: /dns4/ipfs2/tcp/5001
      CLUSTER_CRDT_TRUSTEDPEERS: '*'
      CLUSTER_MONITORPINGINTERVAL: 2s # Speed up peer discovery
    volumes:
      - ./compose/cluster2:/data/ipfs-cluster
```

- ipfs config

```
"API": {
    "HTTPHeaders": {
  "Access-Control-Allow-Credentials": [
      "true"
    ],
    "Access-Control-Allow-Methods": [
      "PUT",
      "GET",
      "POST",
      "OPTIONS"
    ],
    "Access-Control-Allow-Origin": [
        "*"
    ]
  }
  },
  "Addresses": {
    "API": "/ip4/0.0.0.0/tcp/5001",
    "Announce": [],
    "AppendAnnounce": [],
    "Gateway": "/ip4/0.0.0.0/tcp/8080",
    "NoAnnounce": [],
    "Swarm": [
      "/ip4/0.0.0.0/tcp/4001",
      "/ip6/::/tcp/4001",
      "/ip4/0.0.0.0/udp/4001/quic",
      "/ip6/::/udp/4001/quic"
```

```
    ]
  },
"Gateway": {
    "APICommands": [],
    "HTTPHeaders": {
      "Access-Control-Allow-Headers": [
        "X-Requested-With",
        "Range",
        "User-Agent"
      ],
      "Access-Control-Allow-Methods": [
        "GET"
      ],
      "Access-Control-Allow-Origin": [
        "*"
      ]
    },
    "NoDNSLink": false,
    "NoFetch": false,
    "PathPrefixes": [],
    "PublicGateways": null,
    "RootRedirect": "",
    "Writable": false
  },
```

- 실행

```
docker-compose up -d
```

방화벽 설정

- ufw 상태 확인

```
sudo ufw status verbose
```

- ufw 활성/비활성

```
sudo ufw enable # 활성화
sudo ufw disable # 비활성화
```

- ufw 기본 룰 확인

```
sudo ufw show raw
```

- ufw 포트 허용/거부

```
# 포트
sudo ufw allow 8080
sudo ufw allow 8080/tcp
sudo ufw allow 8080/udp

sudo ufw deny 8080
sudo ufw deny 8080/tcp
sudo ufw deny 8080/udp

# 서비스 이름
sudo ufw allow ssh
sudo ufw deny ssh
```

- 특정 IP 방화벽 허용

```
sudo ufw allow from ${IP_ADDRESS}
```

- 특정 IP 주소와 프로토콜, 포트 허용

```
sudo ufw allow from ${IP_ADDRESS} to any port 22
```

- Allowed Ports

Main EC2		BlockChain EC2	
PORT	이름	PORT	이름
22	SSH	4001	IPFS
80	HTTP	5001	
443	HTTPS	8080	
3000	React, Nginx	8081	
3306	MySQL	9094	IPFS-cluster
9010	HERE-Auth	9095	
9011	HERE-Board	9096	
9012	HERE-Nft		
9013	HERE-Notification		
9999	Jenkins		

프론트엔드 배포

- Dockerfile

```
FROM node:16-alpine AS base

# 만약 컨테이너 안의 이미지의 경로가 /app 이런식으로 되어있다면 작업할 div 경로를 설정할 수도 있다.
# 설정해주면 COPY 의 두번째 경로를 ./ 이것으로 했을 때 자동으로 /app 경로가 된다.
WORKDIR /app

# package.json 파일을 복사한다. 만약 다시 빌드할 때 변경사항이 없을 경우 npm install까지 그냥 넘어간다.
COPY package.json /app

# 이미지를 받으면 npm install을 자동으로 해줌
RUN npm install

# 어떤 파일이 이미지에 들어가야 하는지
# 첫 번째 .은 이 프로젝트의 모든 폴더 및 파일들 (Dockerfile을 제외한)
# 두 번째 .은 파일을 저장할 컨테이너 내부 경로 (ex /app)
COPY . /app

RUN npm run build
EXPOSE 3000

ENV PORT 3000

CMD ["node", "server.js"]
```

- Jenkinsfile

```
pipeline {
    agent any
    environment {
        DOCKER = 'sudo docker'
    }

    stages {
        stage('Clone Repository') {
            steps {
```

```
                checkout scm
                echo 'Checkout Scm'
            }
        }

        stage('env setting') {
            steps{
                sh 'cp /var/jenkins_home/env/.env /var/jenkins_home/workspace/develop-FE-pipeline/here-front'
            }
        }

        stage('Build image') {
            steps {
                sh 'ls -al'
                sh 'npm -v'
                sh 'node -v'
                dir('here-front'){
                    sh 'ls -al'
                    sh 'docker build -t ${HERE_FRONT_IMAGE} .'
                }
                echo 'Build image...'
            }
        }

        stage('Remove Previous image') {
            steps {
                script {
                    try {
                        sh 'docker stop ${HERE_FRONT_CONTAINER}'
                        sh 'docker rm ${HERE_FRONT_CONTAINER}'
                    } catch (e) {
                        echo 'fail to stop and remove container'
                    }
                }
            }

        }
        stage('Run New image') {
            steps {
                dir('here-front'){
                    sh 'ls -al'
                    sh 'docker run --name ${HERE_FRONT_CONTAINER} -d -p 3000:3000 ${HERE_FRONT_IMAGE}'
                    echo 'Run New image'
                }
            }
        }

    }
}
```

- nginx/custom.conf

```
server {
        # 프론트 연결(포트 번호는 본인의 프론트 포트번호를 입력)
        location /{
                proxy_pass https://localhost:3000;
                add_header 'Access-Control-Allow-Origin' '*';
                add_header 'Access-Control-Allow-Methods' 'GET, POST, OPTIONS';
                add_header 'Access-Control-Allow-Headers' 'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Typ
                add_header 'Access-Control-Expose-Headers' 'Content-Length,Content-Range';

                # https websocket
                proxy_set_header       Upgrade $http_upgrade;
                proxy_set_header       Connection "upgrade";
                proxy_set_header Host $host;
        }
}
```

백엔드 배포

- Dockerfile

```
FROM adoptopenjdk/openjdk11
COPY build/libs/here-auth-0.0.1-SNAPSHOT.jar app.jar
ENTRYPOINT ["java", "-jar", "-Dspring.profiles.active=dev", "app.jar"]
```

- Jenkinsfile

```
pipeline {
    agent any
    environment {
        DOCKER = 'sudo docker'
    }

    stages {
        stage('Clone Repository') {
            steps {
                checkout scm
                echo 'Checkout Scm'
            }
        }

        stage('env setting') {
            parallel {
                stage('insert-keystore-auth') {
                    when {
                        anyOf {
                            changeset "here-back/here-auth/**/*"
                        }
                    }
                    steps {
                        sh 'cp /var/jenkins_home/env/keystore.p12 /var/jenkins_home/workspace/develop-BE-pipeline/here-back/here-auth/

                        echo 'Insert keystore...'
                    }
                }

                stage('insert-keystore-board') {
                    when {
                        anyOf {
                            changeset "here-back/here-board/**/*"
                        }
                    }
                    steps {
                        sh 'cp /var/jenkins_home/env/keystore.p12 /var/jenkins_home/workspace/develop-BE-pipeline/here-back/here-board

                        echo 'Insert keystore...'
                    }
                }

                stage('insert-keystore-nft') {
                    when {
                        anyOf {
                            changeset "here-back/here-nft/**/*"
                        }
                    }
                    steps {
                        sh 'cp /var/jenkins_home/env/keystore.p12 /var/jenkins_home/workspace/develop-BE-pipeline/here-back/here-nft/s

                        echo 'Insert keystore...'
                    }
                }

                stage('insert-keystore-notification') {
                    when {
                        anyOf {
                            changeset "here-back/here-notification/**/*"
                        }
                    }
                    steps {
                        sh 'cp /var/jenkins_home/env/keystore.p12 /var/jenkins_home/workspace/develop-BE-pipeline/here-back/here-notif

                        echo 'Insert keystore...'
                    }
                }
            }
        }

        stage('Build image') {
            parallel {
                stage('build-here-auth') {
                    when {
                        anyOf {
                            changeset "here-back/here-auth/**/*"
                        }
                    }
                    steps {
                        sh 'ls -al'
                        dir('here-back/here-auth'){
```

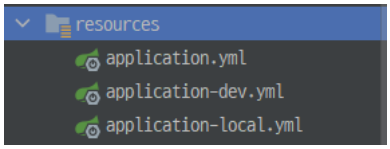
```
                        sh 'ls -al'
                        sh 'chmod +x ./gradlew'
                        sh './gradlew build'
                        sh 'docker build -t ${HERE_AUTH_IMAGE} .'
                    }
                    echo 'Build image...'
                }
            }
            stage('build-here-board') {
                when {
                    anyOf {
                        changeset "here-back/here-board/**/*"
                    }
                }
                steps {
                    sh 'ls -al'
                    dir('here-back/here-board'){
                        sh 'ls -al'
                        sh 'chmod +x ./gradlew'
                        sh './gradlew build'
                        sh 'docker build -t ${HERE_BOARD_IMAGE} .'
                    }
                    echo 'Build image...'
                }
            }
            stage('build-here-nft') {
                when {
                    anyOf {
                        changeset "here-back/here-nft/**/*"
                    }
                }
                steps {
                    sh 'ls -al'
                    dir('here-back/here-nft'){
                        sh 'ls -al'
                        sh 'chmod +x ./gradlew'
                        sh './gradlew build'
                        sh 'docker build -t ${HERE_NFT_IMAGE} .'
                    }
                    echo 'Build image...'
                }
            }
            stage('build-here-notification') {
                when {
                    anyOf {
                        changeset "here-back/here-notification/**/*"
                    }
                }
                steps {
                    sh 'ls -al'
                    dir('here-back/here-notification'){
                        sh 'ls -al'
                        sh 'chmod +x ./gradlew'
                        sh './gradlew build'
                        sh 'docker build -t ${HERE_NOTIFICATION_IMAGE} .'
                    }
                    echo 'Build image...'
                }
            }
        }
    }

    stage('Remove Previous image') {
        parallel {
            stage('remove-here-auth') {
                when {
                    anyOf {
                        changeset "here-back/here-auth/**/*"
                    }
                }
                steps {
                    script {
                        try {
                            sh 'docker stop ${HERE_AUTH_CONTAINER}'
                            sh 'docker rm ${HERE_AUTH_CONTAINER}'
                        } catch (e) {
                            echo 'fail to stop and remove container'
                        }
                    }
                }
            }
            stage('remove-here-board') {
                when {
                    anyOf {
                        changeset "here-back/here-board/**/*"
                    }
                }
```

```
                    steps {
                        script {
                            try {
                                sh 'docker stop ${HERE_BOARD_CONTAINER}'
                                sh 'docker rm ${HERE_BOARD_CONTAINER}'
                            } catch (e) {
                                echo 'fail to stop and remove container'
                            }
                        }
                    }
                }
                stage('remove-here-nft') {
                    when {
                        anyOf {
                            changeset "here-back/here-nft/**/*"
                        }
                    }
                    steps {
                        script {
                            try {
                                sh 'docker stop ${HERE_NFT_CONTAINER}'
                                sh 'docker rm ${HERE_NFT_CONTAINER}'
                            } catch (e) {
                                echo 'fail to stop and remove container'
                            }
                        }
                    }
                }
                stage('remove-here-notification') {
                    when {
                        anyOf {
                            changeset "here-back/here-notification/**/*"
                        }
                    }
                    steps {
                        script {
                            try {
                                sh 'docker stop ${HERE_NOTIFICATION_CONTAINER}'
                                sh 'docker rm ${HERE_NOTIFICATION_CONTAINER}'
                            } catch (e) {
                                echo 'fail to stop and remove container'
                            }
                        }
                    }
                }
            }
        }
        stage('Run New image') {
            parallel {
                stage('run-here-auth') {
                    when {
                        anyOf {
                            changeset "here-back/here-auth/**/*"
                        }
                    }
                    steps {
                        sh 'docker run --name ${HERE_AUTH_CONTAINER} -e KEY_STORE=${KEY_STORE} -e KEY_STORE_PASSWORD=${KEY_STORE_PASSW
                        echo 'Run New image'
                    }
                }
                stage('run-here-board') {
                    when {
                        anyOf {
                            changeset "here-back/here-board/**/*"
                        }
                    }
                    steps {
                        sh 'docker run --name ${HERE_BOARD_CONTAINER} -e KEY_STORE=${KEY_STORE} -e KEY_STORE_PASSWORD=${KEY_STORE_PASS
                        echo 'Run New image'
                    }
                }
                stage('run-here-nft') {
                    when {
                        anyOf {
                            changeset "here-back/here-nft/**/*"
                        }
                    }
                    steps {
                        sh 'docker run --name ${HERE_NFT_CONTAINER} -e KEY_STORE=${KEY_STORE} -e KEY_STORE_PASSWORD=${KEY_STORE_PASSWO
                        echo 'Run New image'
                    }
                }
                stage('run-here-notification') {
                    when {
                        anyOf {
                            changeset "here-back/here-notification/**/*"
                        }
```

```
                    }
                    steps {
                        sh 'docker run --name ${HERE_NOTIFICATION_CONTAINER} -e KEY_STORE=${KEY_STORE} -e KEY_STORE_PASSWORD=${KEY_STO
                        echo 'Run New image'
                    }
                }
            }
        }
    }
}
```

- gitignore로 업로드 되지 않는 env 파일 및 기타 파일들은 jenkins 서버에 저장되어 있어서, clone 받은 이후 cp 명령어를 통해 배포 시점에 해당 파일들을 프로젝트에 복사하여 적용
- 같은 branch로 merge 혹은 push가 진행되었을 때, 4개의 프로젝트 중 이벤트가 일어난 프로젝트만 배포가 진행되도록 조건문 활용

- application.yml
- 개발 환경에 맞춰 yml 파일을 분리하여, 로컬과 배포시점에 맞는 yml파일을 사용하도록 설정함



스마트 컨트랙트 배포

설치

- install npm

```
apt-get install npm
npm install -g npm@6.14.18
```

- install truffle

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs
sudo npm install -g truffle@5.4.33
truffle version
```

- install truffle using docker

```
docker pull trufflesuite/truffle:5.4.33
docker run -name truffle-container -it trufflesuite/truffle:5.4.33
truffle version
```

openzeppelin

- install openzeppelin

```
npm install @openzeppelin/contracts
```

solidity

- install solidity

```
sudo add-apt-repository -y ppa:ethereum/ethereum
sudo apt-get update
sudo apt-get install solc
solc --version
```

- install solidity using docker

```
docker pull ethereum/solidity:0.8.4
docker run -name solidity-container -it ethereum/solidity:0.8.4
solc --version
exit
```

web3.js v1.8.0

- install web3.js

```
curl -sL https://deb.nodesource.com/setup_14.x | sudo -E bash -
sudo apt-get install -y nodejs (이미 설치되어 있으면 생략)
sudo npm install web3@1.8.0 --save
node -e "console.log(require('web3').version)"
```

- install web3.js using docker
  - create dockerfile in project directory

```
FROM node:14
RUN npm install web3@1.8.0 --save
```

  - build the docker image

```
docker build -t my-web3-image .
```

  - run a docker container

```
docker run -it my-web3-image bash
```

  - verify the web3.js installation

```
node -e "console.log(require('web3').version)"
```

ganache

- install ganache

```
npm install -g ganache-cli
```

---

truffle에서 ssafy network 연결

```
npm install truffle-privatekey-provider
```

truffle-config.js

```
var PrivateKeyProvider = require('truffle-privatekey-provider');
const privateKey =
  '개인 pk';

networks: {
  ssafynet: {
        provider: () =>
          new PrivateKeyProvider(privateKey, 'URI'),
```

```
        network_id: '*', // Match any network id
      },
}
```

- 배포

```
truffle compile
truffle migrate --reset --network ssafynet
```

Web3

스마트 컨트랙트 연결

- ABI

```
build\contracts\HereNFT.json  // abi 파일 복사
```

- smart-contract-address

```
truffle console --network ssafynet // ssafynet console 접속

HereNFT.address // 배포한 HereNFT address 확인
```

```
truffle(ssafynet)> HereNFT.address
'0x5b833C9AaFC03dBe665E0b0B76f2bd1618c13c7A'
```

외부 문서

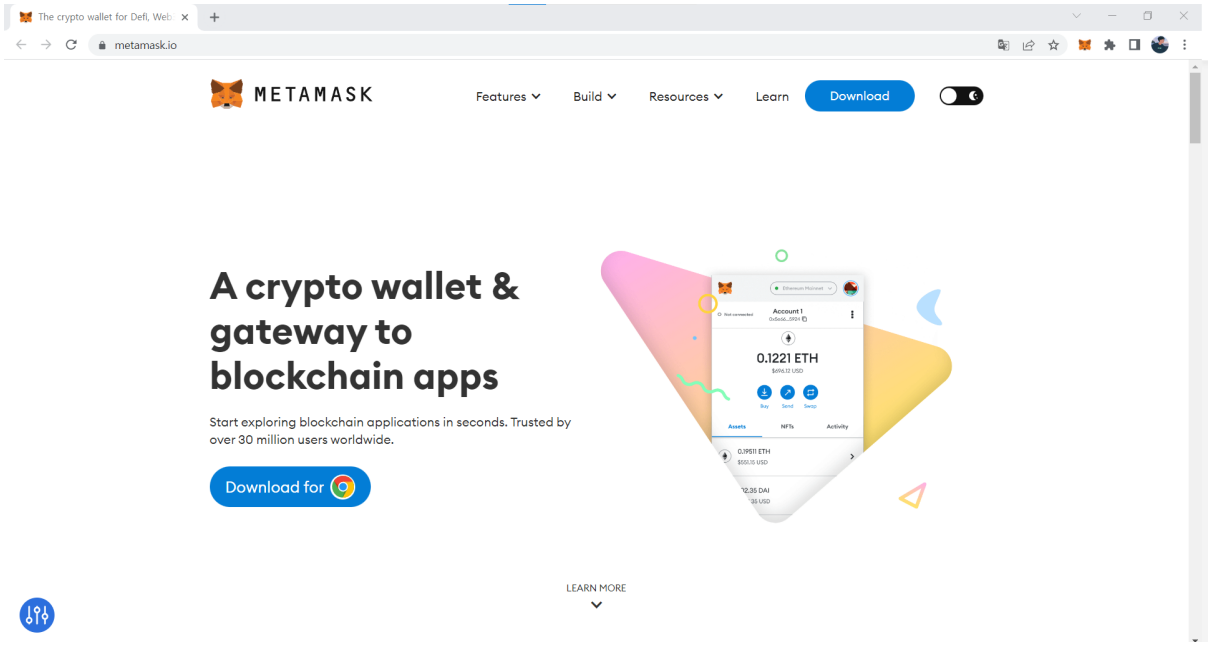
AWS S3

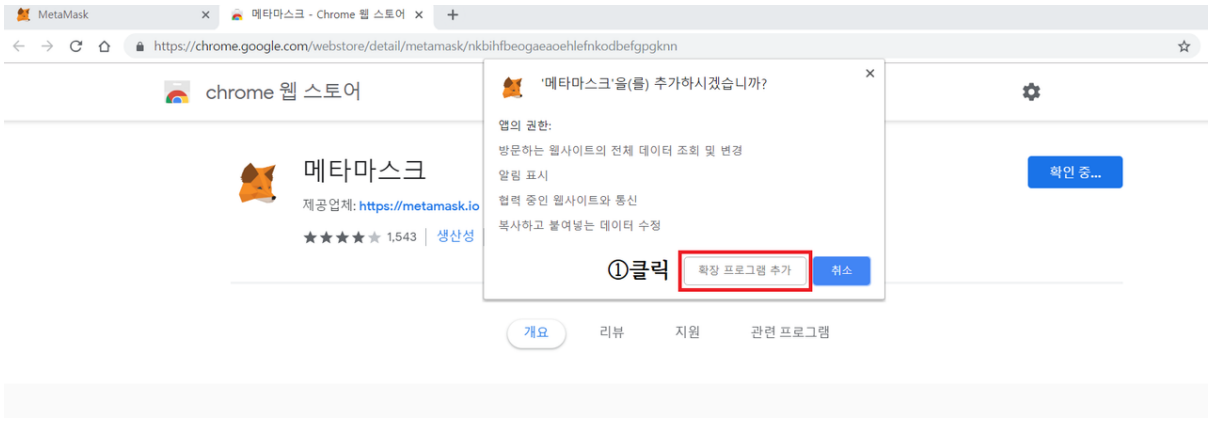
- application.yml

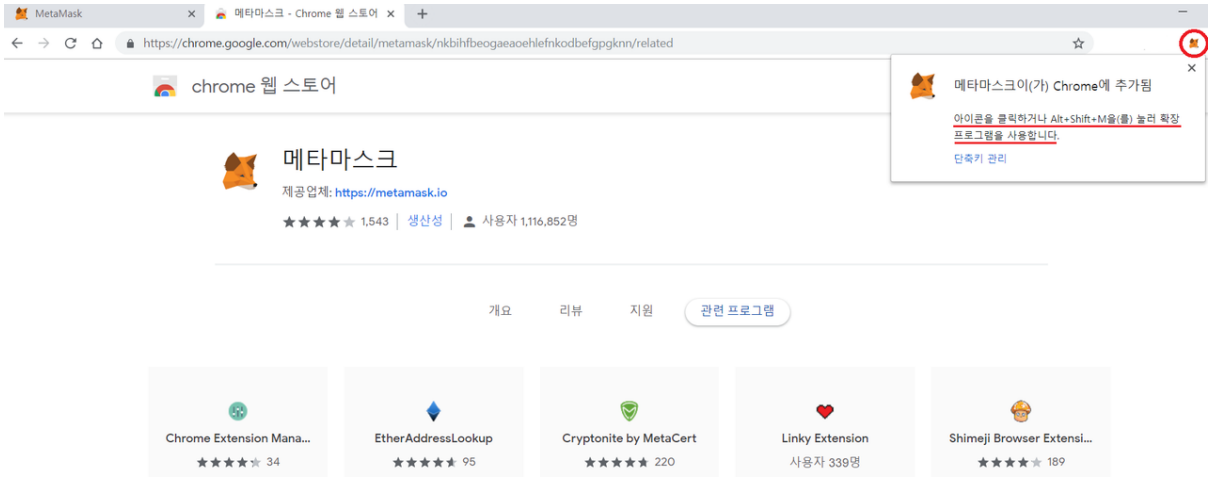
```
cloud:
  aws:
    credentials:
      accessKey: ${S3_ACCESS_KEY}
      secretKey: ${S3_SECRET_KEY}
    s3:
      bucket: ${S3_BUCKET}
    stack:
      auto: false
    region:
      static: ap-northeast-2
```

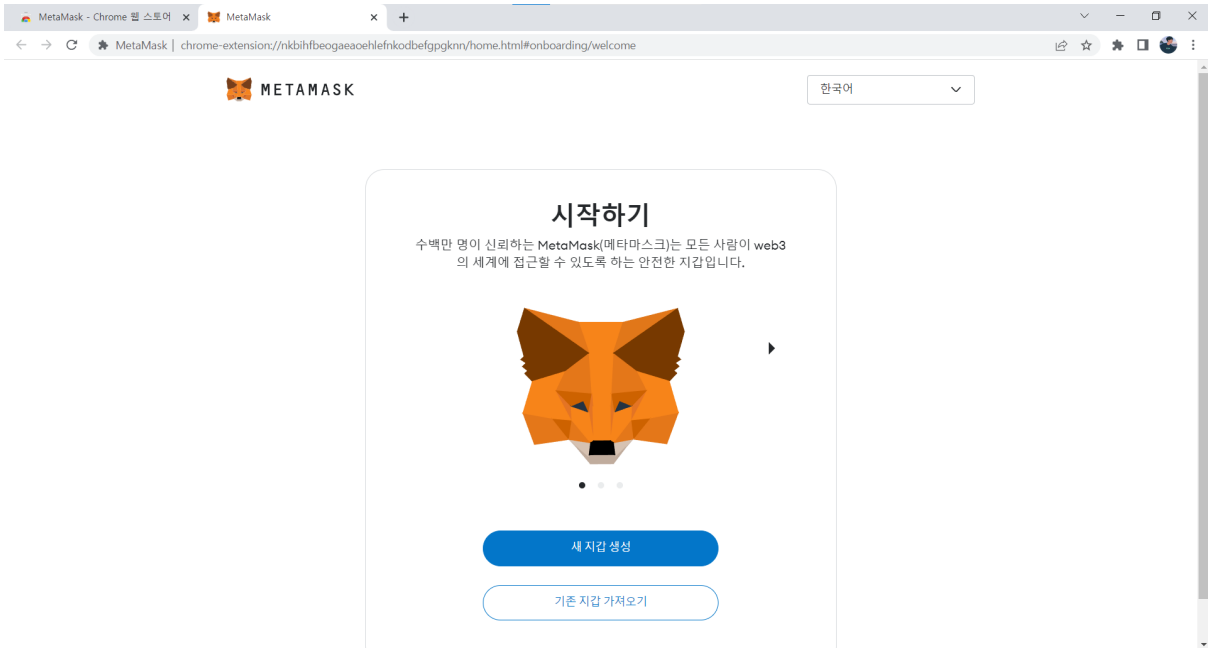
Metamask 설치

- Download for Chrome


METAMASK
Features
Build
Resources
Learn
Download
A crypto wallet & gateway to blockchain apps
Start exploring blockchain applications in seconds. Trusted by over 30 million users worldwide.
Download for
0.1221 ETH
LEARN MORE

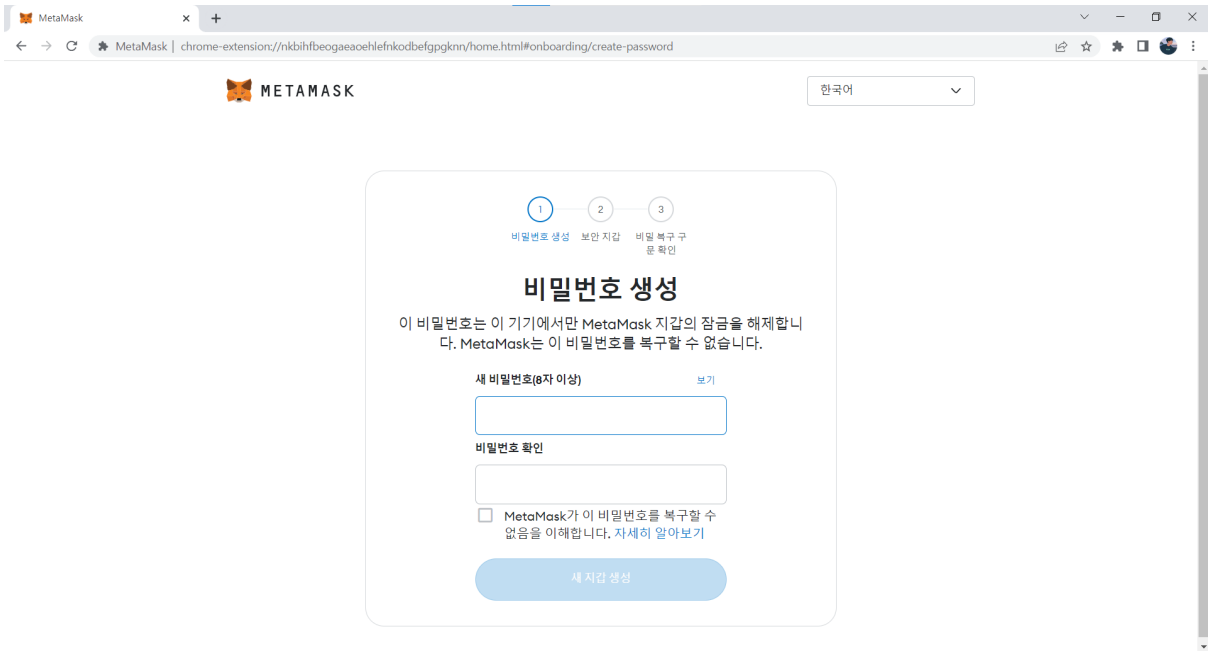

chrome 웹 스토어
'메타마스크'을(를) 추가하시겠습니까?
앱의 권한:
방문하는 웹사이트의 전체 데이터 조회 및 변경
알림 표시
협력 중인 웹사이트와 통신
복사하고 붙여넣는 데이터 수정
①클릭
확장 프로그램 추가
취소
메타마스크
제공업체: https://metamask.io
1,543 | 생산성
확인 중...
개요
리뷰
지원
관련 프로그램

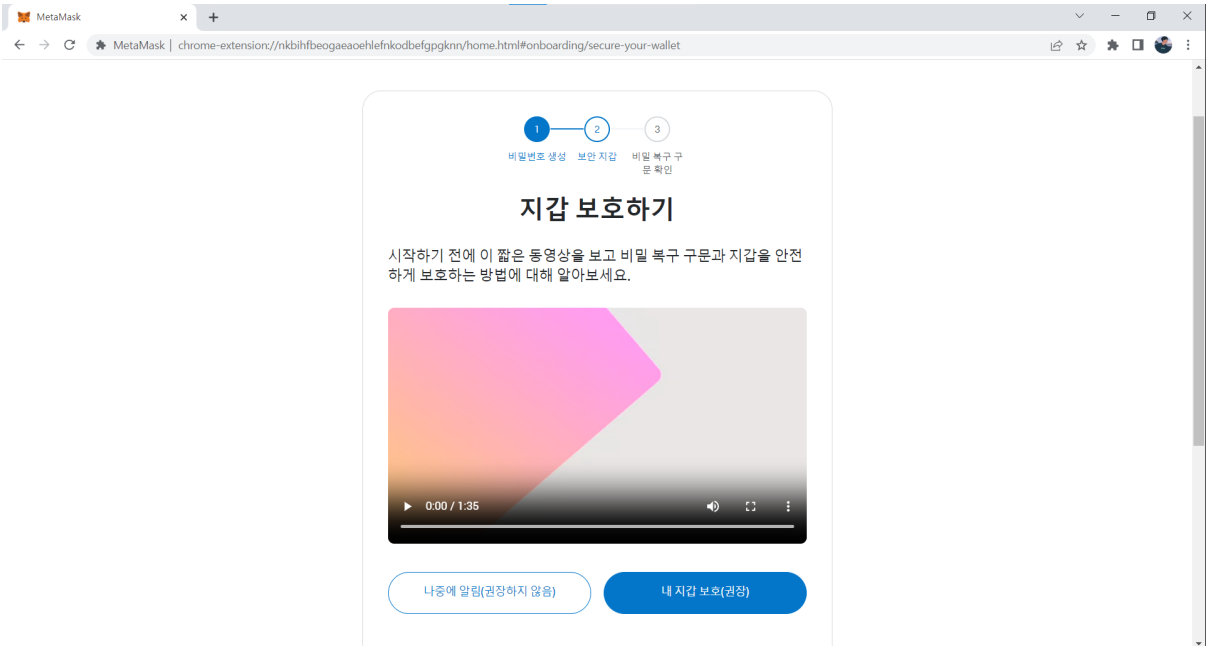

chrome 웹 스토어
메타마스크이(가) Chrome에 추가됨
아이콘을 클릭하거나 Alt+Shift+M을(를) 눌러 확장 프로그램을 사용합니다.
단축키 관리
메타마스크
제공업체: https://metamask.io
1,543 | 생산성 | 사용자 1,116,852명
개요
리뷰
지원
관련 프로그램
Chrome Extension Mana...
EtherAddressLookup
Cryptonite by MetaCert
Linky Extension
사용자 339명
Shimeji Browser Extensi...

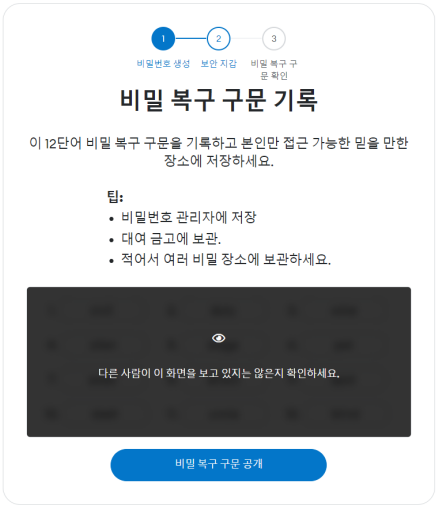


- 처음이라면 → **새 지갑 생성**
- 기존 지갑이 있다면 → 기존 지갑 가져오기

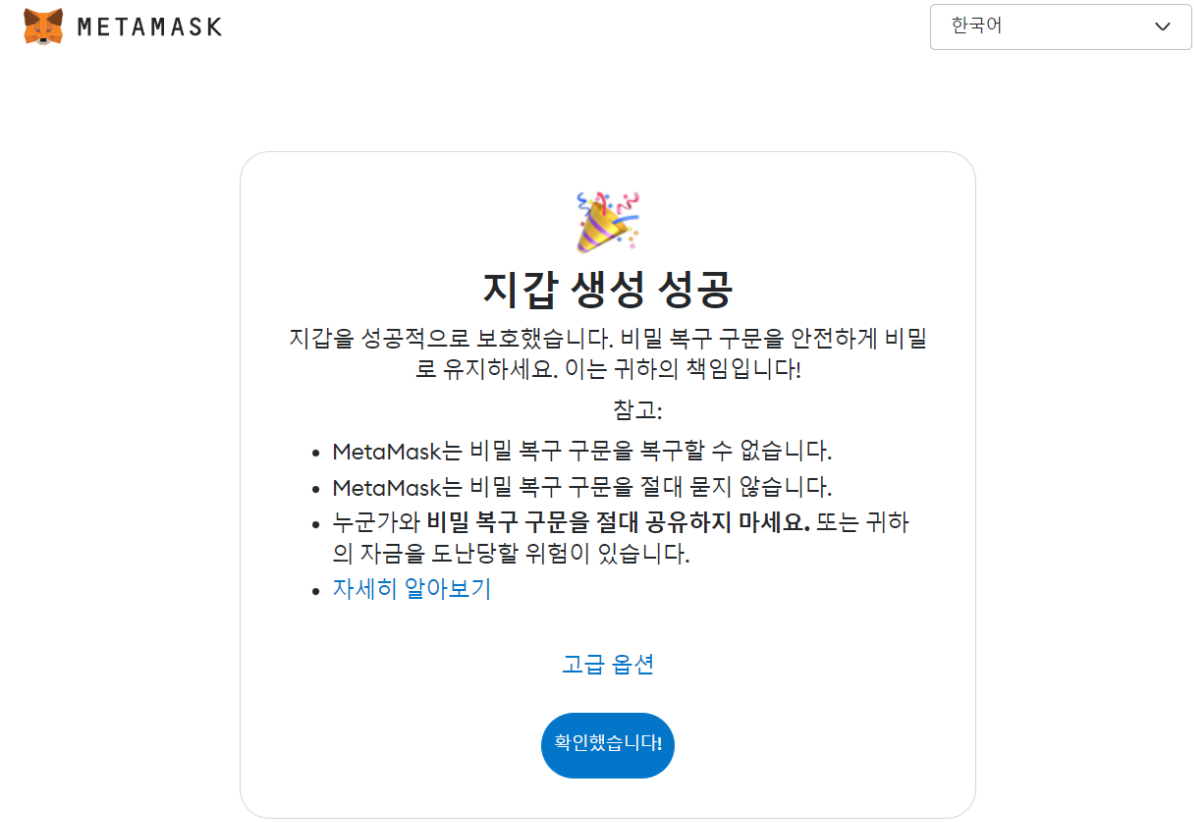
지갑 생성

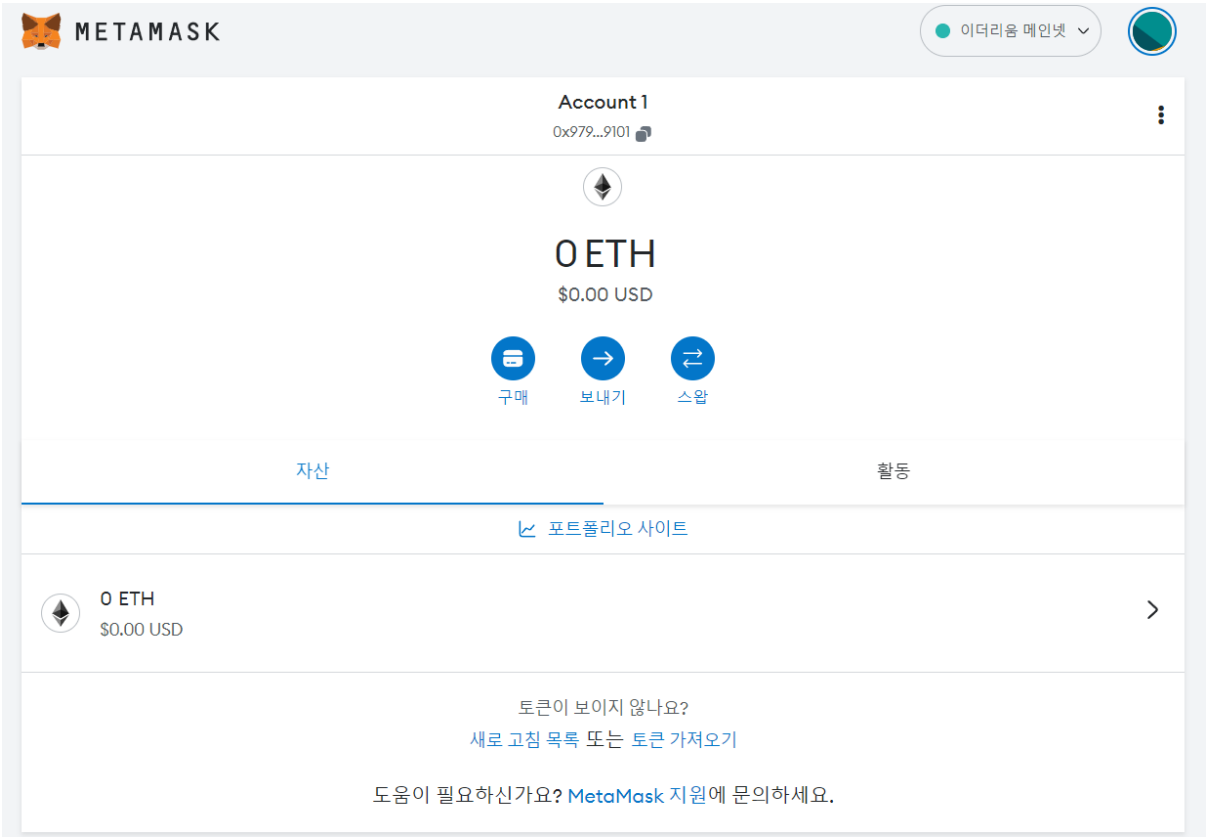






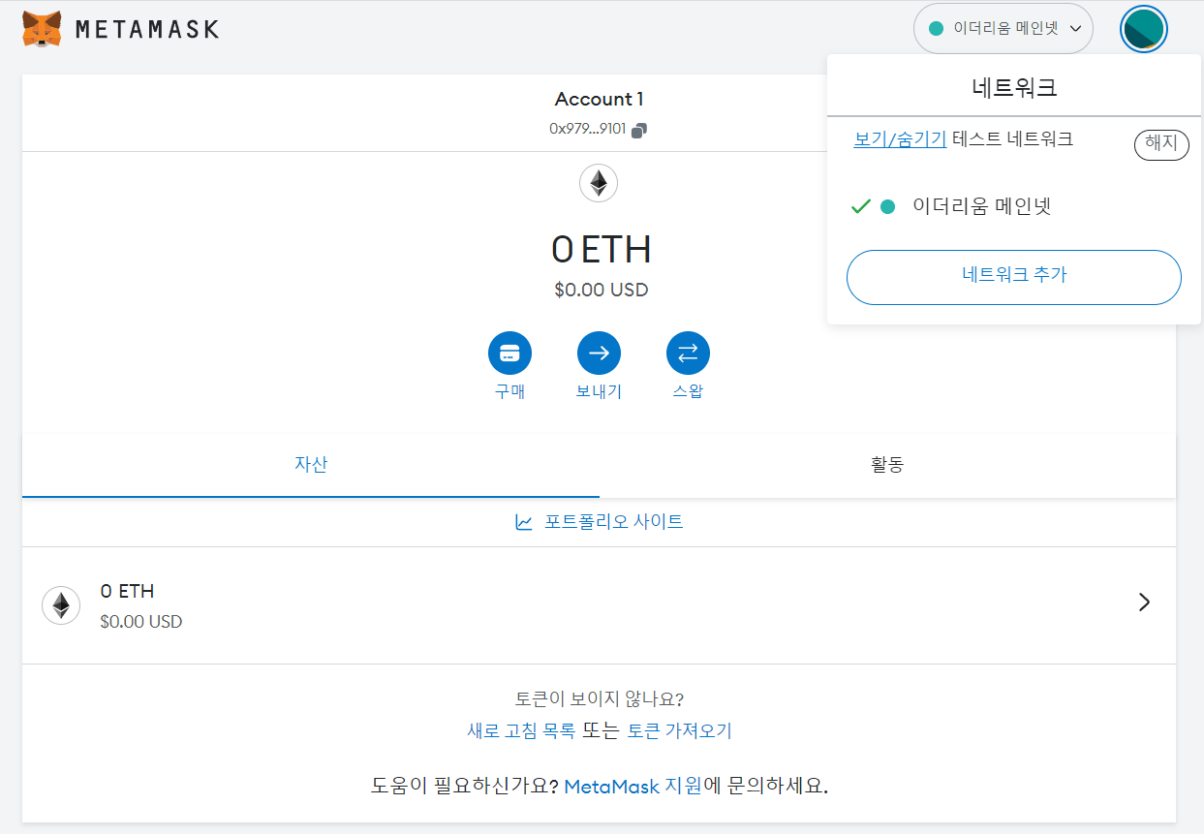
- 비밀 복구 구문을 개인적으로 보관

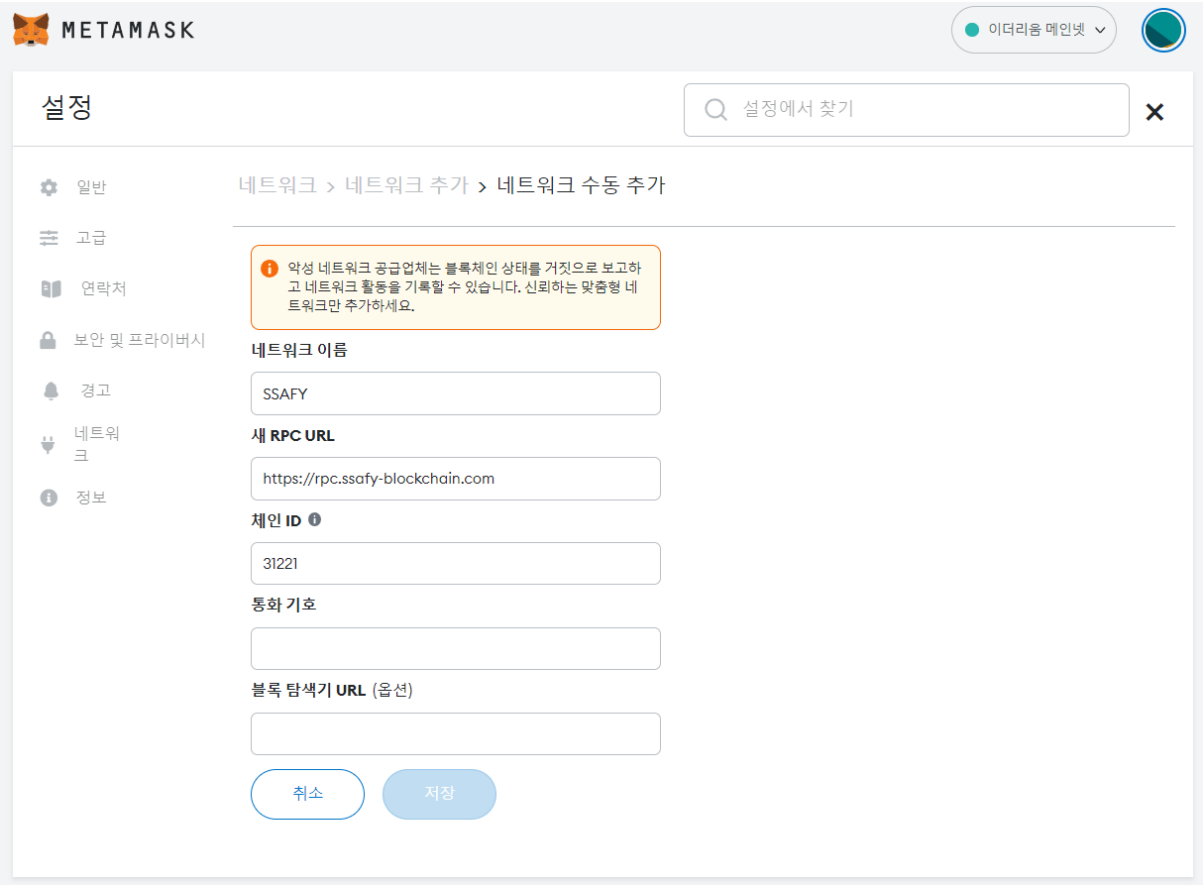




- 완료!

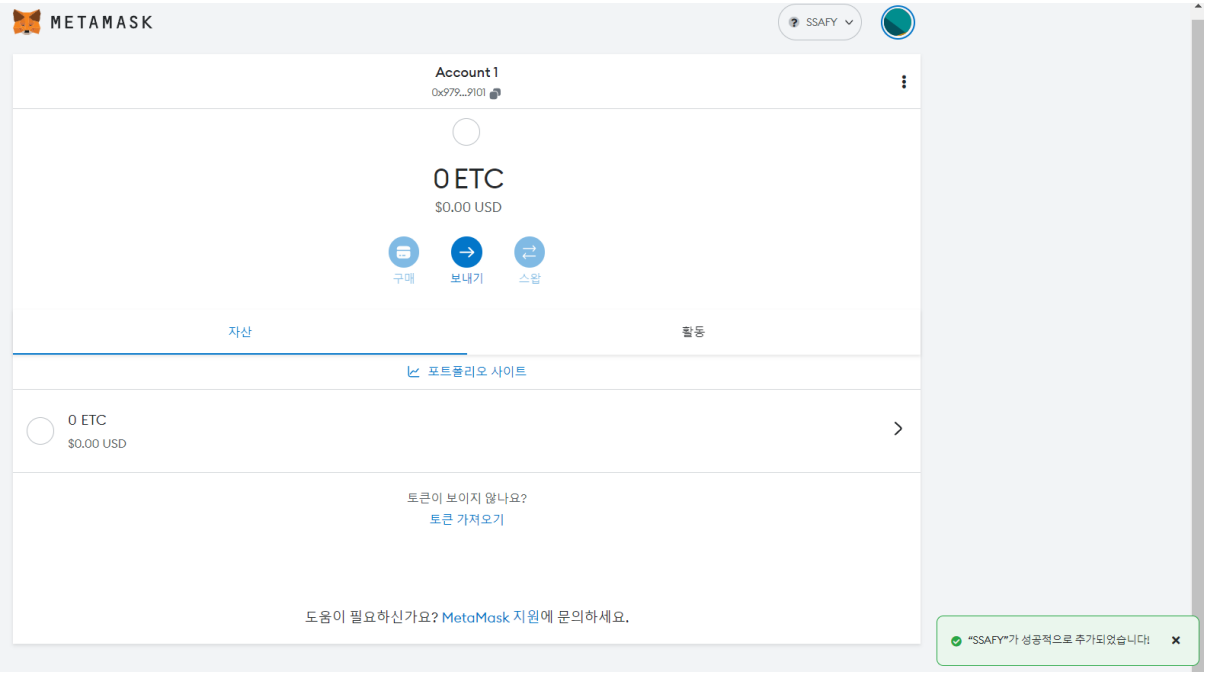
SSAFY Network 추가





- 설정 → 네트워크 → 네트워크 추가 → 네트워크 수동 추가

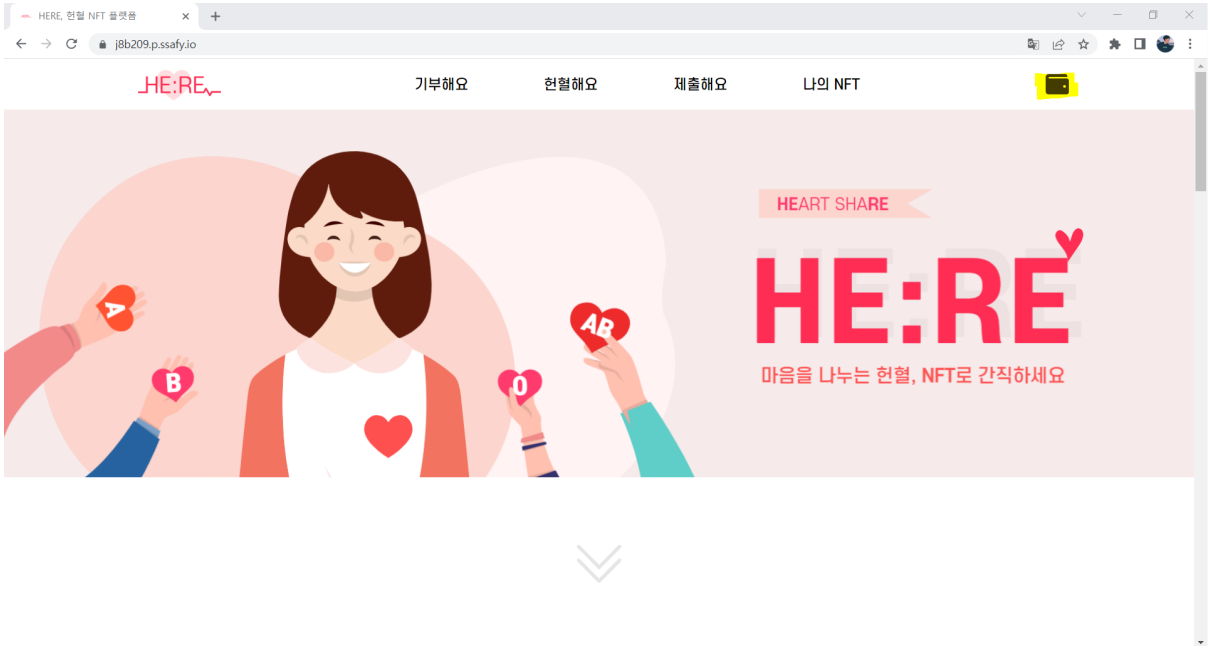
- https://rpc.ssafy-blockchain.com
- chain ID: 31221
- 통화 기호 : SSF

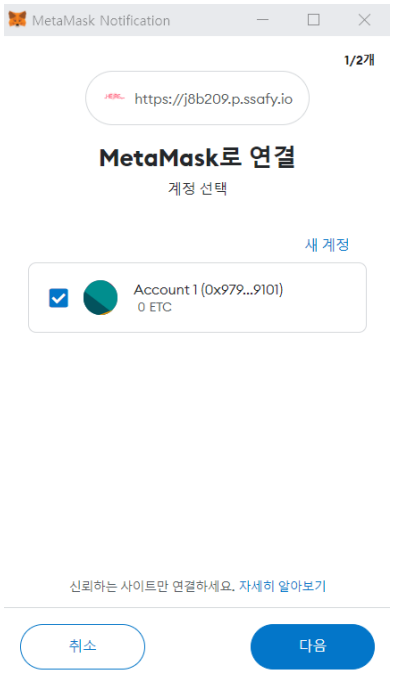


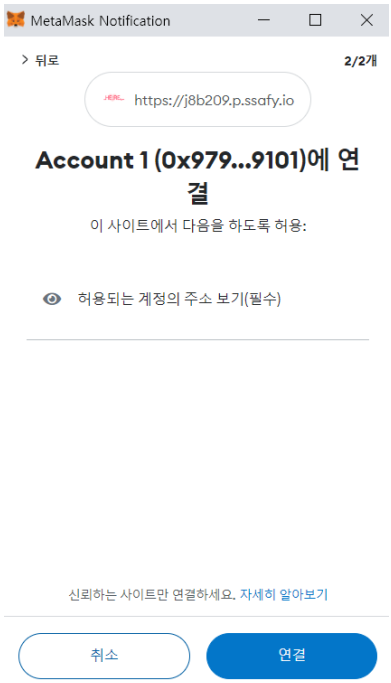
- 완료!

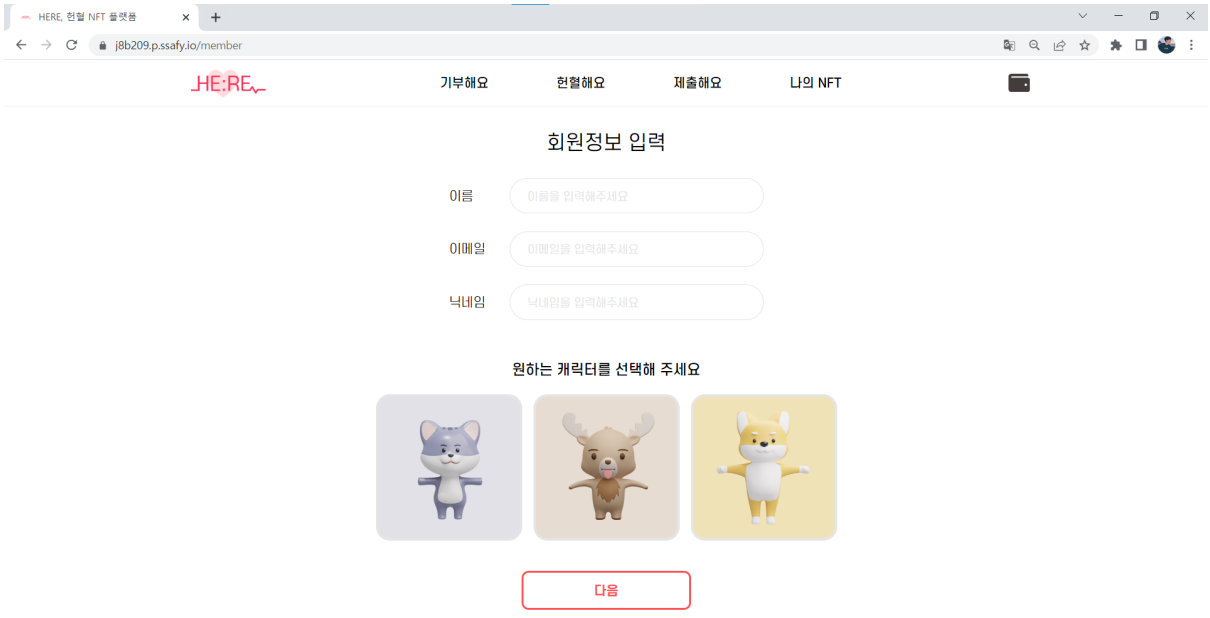
HE:RE 에 METAMASK 연결하기

- 우측 상단 지갑 클릭









- 회원정보 입력
- 지갑 연동 & 회원가입 완료!