



BirdMeal

아기새[:Bird]에게 모이[:Meal]를 주다

포팅 매뉴얼

SSAFY 7기 구미 특화프로젝트 D101팀

강태웅 김지수 배혜연 심세현 이건희 최시령

목차

1. 사용 기술
2. 빌드 및 배포
 - A. Infra
 - B. 스마트 컨트랙트
 - C. 백엔드
 - D. 안드로이드
 - E. 웹 프론트엔드
3. Google Oauth 설정
4. MetaMask 설정

사용 기술

Infra

- AWS EC2, AWS S3, Ubuntu 20.04 LTS, Docker, Jenkins, Nginx, Let's Encrypt

Smart Contract

- Solidity, Geth, Remix, Ethereum, web3

Backend

- SpringBoot, JPA, QueryDSL, Java, Lombok, Gradle, Swagger, Oauth, Mysql

Android

- Dagger Hilt, Retrofit, Coroutine, Coroutine Flow, Navigation, Paging 3, web3J

Frontend

- Vue3, Vue3 Router, web3js, vuetify3, Vite, Pinia, Axios, Vue3-Lottie, vue-cli

Communication

- Mattermost, Notion, Jira, GitLab, Git, Source Tree

빌드 및 배포

Infra 설정

1. AWS EC2 Ubuntu 20.04 LTS 서버에 터미널 접속.
2. 방화벽 설정

```
# 상태 확인
$ sudo ufw status

# ssh
$ sudo ufw allow 22

# http/https
$ sudo ufw allow 80
$ sudo ufw allow 443

# mysql
$ sudo ufw allow 3306

# Api
$ sudo ufw allow 8080

# jenkins
$ sudo ufw allow 9090

# enable
$ sudo ufw enable

# 상태 확인
$ sudo ufw status
```

3. Nginx 설치

```
# 설치
$ sudo apt update
$ sudo apt install nginx

# 실행 확인
$ sudo systemctl status nginx
or
$ sudo service nginx status

# inactive 이면 실행
$ sudo systemctl start nginx
or
$ sudo service nginx start
```

4. Docker 설치

```
# Old version 삭제
$ sudo apt-get remove docker docker-engine docker.io containerd runc

# 필요한 패키지 설치
$ sudo apt-get update
$ sudo apt-get install \
    ca-certificates \
    curl \
    gnupg \
    lsb-release

# docker GPG key 가져오기
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg
--dearmor -o /etc/apt/keyrings/docker.gpg

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

# docker engine 설치
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-
compose-plugin

# 실행 확인
$ sudo systemctl status docker
or
$ sudo service docker status

# inactive 일 경우 실행
$ sudo systemctl start docker
or
$ sudo service docker start

# hello world
$ sudo docker run hello-world
```

5. Nginx SSL 적용

```
# snapd 설치
$ sudo apt update
$ sudo apt install snapd

# snap update
$ sudo snap install core; sudo snap refresh core

# old ver 삭제
$ sudo apt-get remove certbot

# certbot 설치
$ sudo snap install --classic certbot

# 심볼릭 링크 생성
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot

# nginx ssl 설정
$ sudo certbot --nginx

# 이후 Email address, Domain name 입력후 인증서 발급
```

6. Docker Mysql 설치

```
$ docker run --name <container-name> -e MYSQL_ROOT_PASSWORD=<password> -d -p
3306:3306 mysql:latest
```

7. Nginx 설정

```
// Web페이지 Redirect
server {
    client_max_body_size 100m;
    root /var/www/html;

    index index.html index.htm index.nginx-debian.html;
    server_name j7d101.p.ssafy.io; # managed by Certbot

    location / {
        proxy_pass http://localhost:8443;
    }

    location /api {
        proxy_pass http://localhost:8081;
    }

    listen [::]:443 ssl ipv6only=on; # managed by Certbot
    listen 443 ssl; # managed by Certbot

    ssl_certificate /etc/letsencrypt/live/j7d101.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j7d101.p.ssafy.io/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
}
```

```
// Https redirect
server {
    client_max_body_size 100m;
    if ($host = j7d101.p.ssafy.io) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80 ;
    listen [::]:80 ;
    server_name j7d101.p.ssafy.io;
    return 404; # managed by Certbot
}
```

```
// API Redirect
server {
    client_max_body_size 100m;
    server_name j7d101.p.ssafy.io;

    location /api {
        proxy_pass http://localhost:8081;
    }

    location ~ ^/(swagger|webjars|configuration|swagger-resources|v2|csrf) {
        proxy_pass http://localhost:8081;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }

    listen 8080 ssl;
    listen [::]:8080 ssl ipv6only=on; # managed by Certbot

    ssl_certificate /etc/letsencrypt/live/j7d101.p.ssafy.io/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/j7d101.p.ssafy.io/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
}
```


8. Mysql 설정

```
# mysql 계정 생성
mysql> create user 's07d101'@'%' identified by 'ssafy7d101';

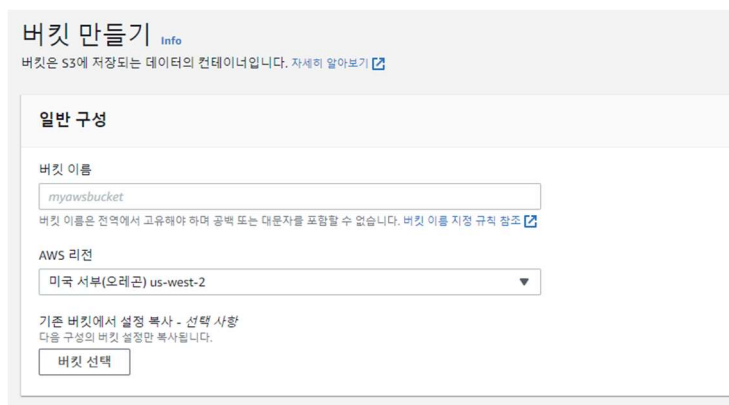
# birdmeal database 생성
mysql> create database birdmeal;

# 접근 권한 설정
mysql> grant all privileges on birdmeal.* to 's07d101'@'%';
mysql> flush privileges;
```

9. AWS S3 설정



버킷 만들기



버킷 이름 설정

객체 소유권 Info

다른 AWS 계정에서 이 버킷에 작성한 객체의 소유권 및 액세스 제어 목록(ACL)의 사용을 제어합니다. 객체 소유권은 객체에 대한 액세스를 지정할 수 있는 사용자를 결정합니다.

☐ ACL 비활성화됨(권장)

이 버킷의 모든 객체는 이 계정이 소유합니다. 이 버킷과 그 객체에 대한 액세스는 정책을 통해서만 지정됩니다.

☒ ACL 활성화됨

이 버킷의 객체는 다른 AWS 계정에서 소유할 수 있습니다. 이 버킷 및 객체에 대한 액세스는 ACL을 사용하여 지정할 수 있습니다.

객체 소유권

☒ 버킷 소유자 선호

이 버킷에 작성된 새 객체가 bucket-owner-full-control 삽입 ACL을 지정하는 경우 새 객체는 버킷 소유자가 소유합니다. 그렇지 않은 경우 객체 라이터가 소유합니다.

☐ 객체 라이터

객체 라이터는 객체 소유자로 유지됩니다.

- ① 새 객체에 대해서만 객체 소유권을 적용하려면 버킷 정책이 객체 업로드에 bucket-owner-full-control 삽입 ACL을 요구하도록 지정해야 합니다. 자세히 알아보기**

객체 소유권 ACL 활성화, 버킷 소유자 선호

이 버킷의 퍼블릭 액세스 차단 설정

퍼블릭 액세스는 ACL(액세스 제어 목록), 버킷 정책, 액세스 지정 정책 또는 모두를 통해 버킷 및 객체에 부여됩니다. 이 버킷 및 해당 객체에 대한 퍼블릭 액세스가 차단되었는지 확인하려면 모든 퍼블릭 액세스 차단을 활성화합니다. 이 설정은 이 버킷 및 해당 액세스 지정에만 적용됩니다. AWS에서는 모든 퍼블릭 액세스 차단을 활성화하도록 권장하지만, 이 설정을 적용하기 전에 퍼블릭 액세스가 없어도 애플리케이션이 올바르게 작동하는지 확인합니다. 이 버킷 또는 내부 객체에 대한 어느 정도 수준의 퍼블릭 액세스가 필요한 경우 특정 스토리지 사용 사례에 맞게 아래 개별 설정을 사용자 지정할 수 있습니다. [자세히 알아보기](#)

☐ 모든 퍼블릭 액세스 차단

이 설정을 활성화하면 아래 4개의 설정을 모두 활성화한 것과 같습니다. 다음 설정 각각은 서로 독립적입니다.

☐ 새 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 새로 추가된 버킷 또는 객체에 적용되는 퍼블릭 액세스 권한을 차단하며, 기존 버킷 및 객체에 대한 새 퍼블릭 액세스 ACL 생성을 금지합니다. 이 설정은 ACL을 사용하여 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 권한을 변경하지 않습니다.

☐ 임의의 ACL(액세스 제어 목록)을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 모든 ACL을 무시합니다.

☐ 새 퍼블릭 버킷 또는 액세스 지정 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 액세스 차단

S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 새 버킷 및 액세스 지정 정책을 차단합니다. 이 설정은 S3 리소스에 대한 퍼블릭 액세스를 허용하는 기존 정책을 변경하지 않습니다.

☐ 임의의 퍼블릭 버킷 또는 액세스 지정 정책을 통해 부여된 버킷 및 객체에 대한 퍼블릭 및 교차 계정 액세스 차단

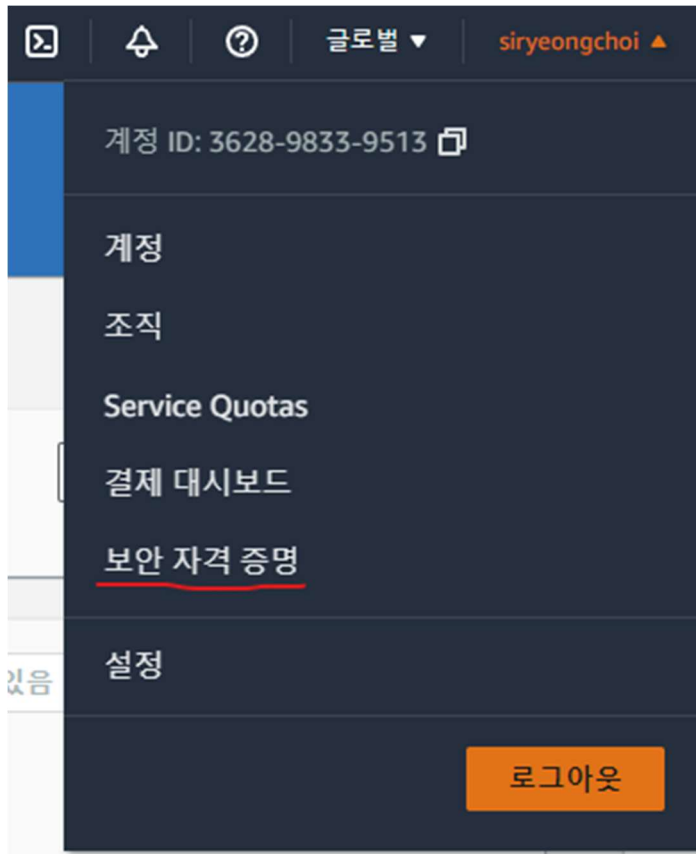
S3은 버킷 및 객체에 대한 퍼블릭 액세스를 부여하는 정책을 사용하는 버킷 또는 액세스 지정에 대한 퍼블릭 및 교차 계정 액세스를 무시합니다.



모든 퍼블릭 액세스 차단을 비활성화하면 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있습니다. 정책 웹 사이트 호스팅과 같은 구체적인 사용 사례에서 퍼블릭 액세스가 필요한 경우가 아니면 모든 퍼블릭 액세스 차단을 활성화하는 것이 좋습니다.

- ☒ 현재 설정으로 인해 이 버킷과 그 안에 포함된 객체가 퍼블릭 상태가 될 수 있음을 알고 있습니다.

버킷의 퍼블릭 액세스 차단 해제



프로필의 보안자격 증명

▼ 액세스 관리

사용자 그룹

사용자

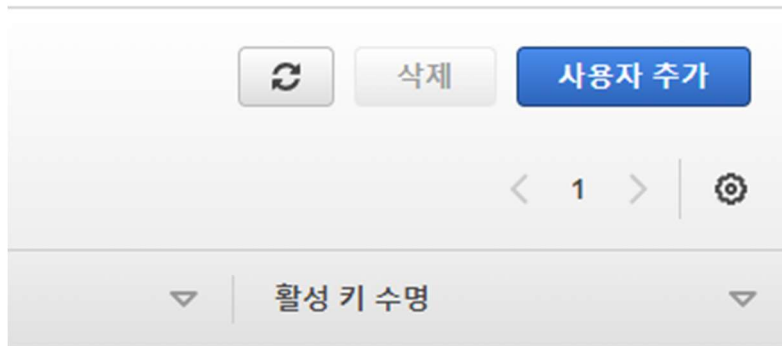
역할

정책

자격 증명 공급자

계정 설정

액세스 관리 → 사용자



사용자 추가

사용자 추가

12345

사용자 세부 정보 설정

동일한 액세스 유형 및 권한을 사용하여 한 번에 여러 사용자를 추가할 수 있습니다. 자세히 알아보기

사용자 이름*

username

다른 사용자 추가

AWS 액세스 유형 선택

이러한 사용자가 주로 AWS에 액세스하는 방법을 선택합니다. 프로그래밍 방식의 액세스만 선택하면 사용자가 위임된 역할을 사용하여 콘솔에 액세스하는 것을 방지할 수 없습니다. 액세스 키와 자동 생성된 암호가 마지막 단계에서 제공됩니다. 자세히 알아보기

AWS 자격 증명 유형 선택*

☒ 액세스 키 – 프로그래밍 방식 액세스
AWS API, CLI, SDK 및 기타 개발 도구에 대해 액세스 키 ID 및 비밀 액세스 키 을(를) 할
성화합니다.

☐ 암호 – AWS 관리 콘솔 액세스
사용자가 AWS Management Console에 로그인할 수 있도록 허용하는 비밀번호 을(를)
활성화합니다.

* 필수

취소

다음: 권한

사용자이름 입력, 액세스 키 체크 후 다음

그룹에 사용자 추가

기존 사용자에서 권한 복사

기존 정책 직접 연결

정책 생성

정책 필터

s3

9 결과 표시

	정책 이름	유형	사용 용도
<input type="checkbox"/>	▶ AmazonDMSRedshiftS3Role	AWS 관리형	없음
<input checked="" type="checkbox"/>	▶ AmazonS3FullAccess	AWS 관리형	Permissions policy (2)
<input type="checkbox"/>	▶ AmazonS3ObjectLambdaExecutionRolePolicy	AWS 관리형	없음
<input type="checkbox"/>	▶ AmazonS3OutpostsFullAccess	AWS 관리형	없음
<input type="checkbox"/>	▶ AmazonS3OutpostsReadOnlyAccess	AWS 관리형	없음
<input type="checkbox"/>	▶ AmazonS3ReadOnlyAccess	AWS 관리형	없음
<input type="checkbox"/>	▶ AWSBackupServiceRolePolicyForS3Backup	AWS 관리형	없음
<input type="checkbox"/>	▶ AWSBackupServiceRolePolicyForS3Restore	AWS 관리형	없음
<input type="checkbox"/>	▶ QuickSightAccessForS3StorageManagementAnalyticsReadOnly	AWS 관리형	없음

기존 정책 직접 연결 → S3FullAccess 체크 후 다음

태그 추가 없이 다음

검토

선택 항목을 검토합니다. 사용자를 생성한 후 자동으로 생성된 비밀번호와 액세스 키를 보고 다운로드할 수 있습니다.

사용자 세부 정보

사용자 이름	username
AWS 액세스 유형	프로그래밍 방식 액세스 - 액세스 키 사용
권한 경계	권한 경계가 설정되지 않았습니다

권한 요약

다음 정책이 위에 표시된 사용자에게 연결됩니다.

유형	이름
관리형 정책	AmazonS3FullAccess

태그

태그가 추가되지 않았습니다.

[취소](#)[이전](#)[사용자 만들기](#)

검토 후 사용자 만들기

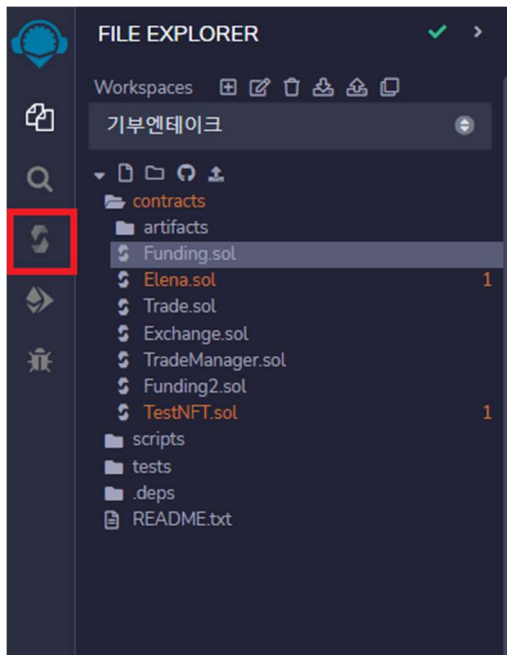
	사용자	액세스 키 ID	비밀 액세스 키
▶	✔ username	AKIAV7TNG247XTLMH5R 	***** 표시

액세스 키 ID 와 비밀 액세스 키 따로 저장해두기. → 나중에 Backend 에서 사용.

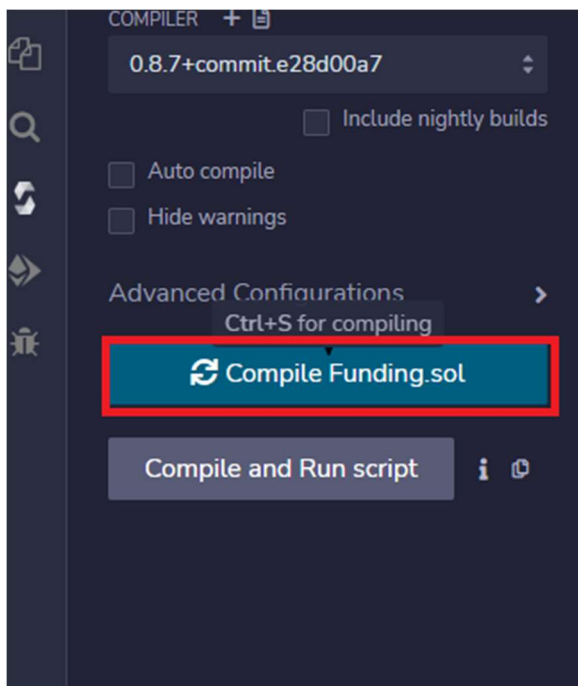
스마트 컨트랙트 빌드 및 배포

1. Remix IDE 에 접속해서 코드 작성 후 컴파일할 파일을 선택 하고

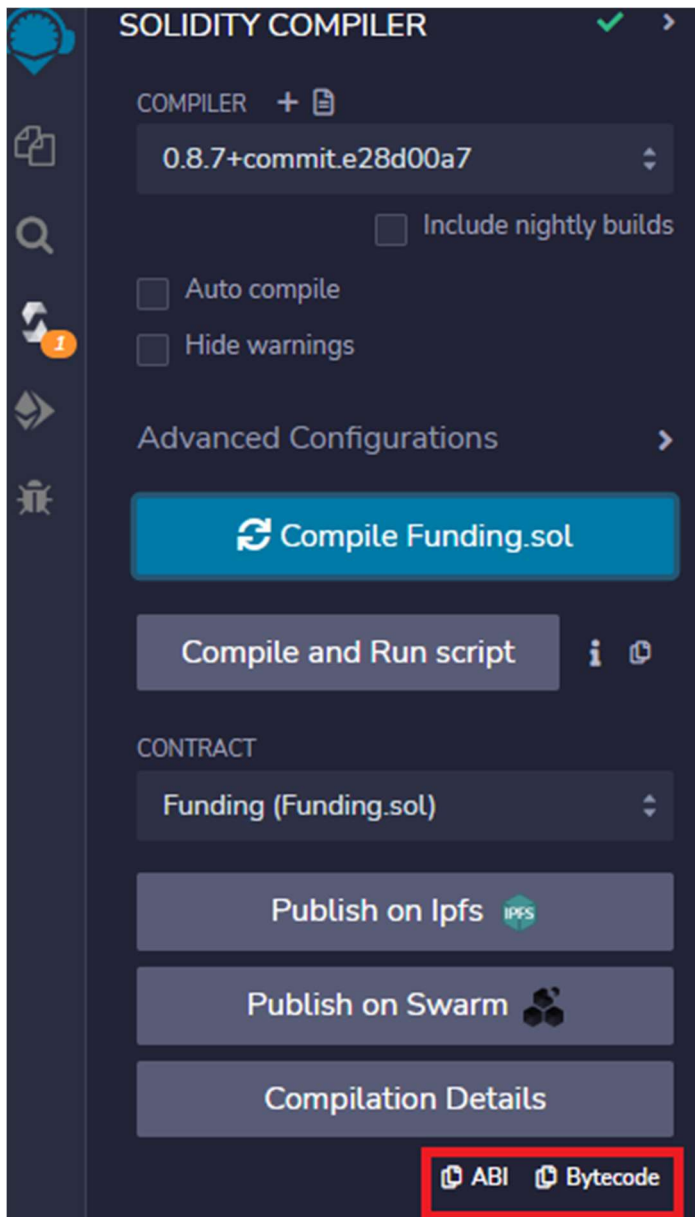
빨간색 박스를 클릭



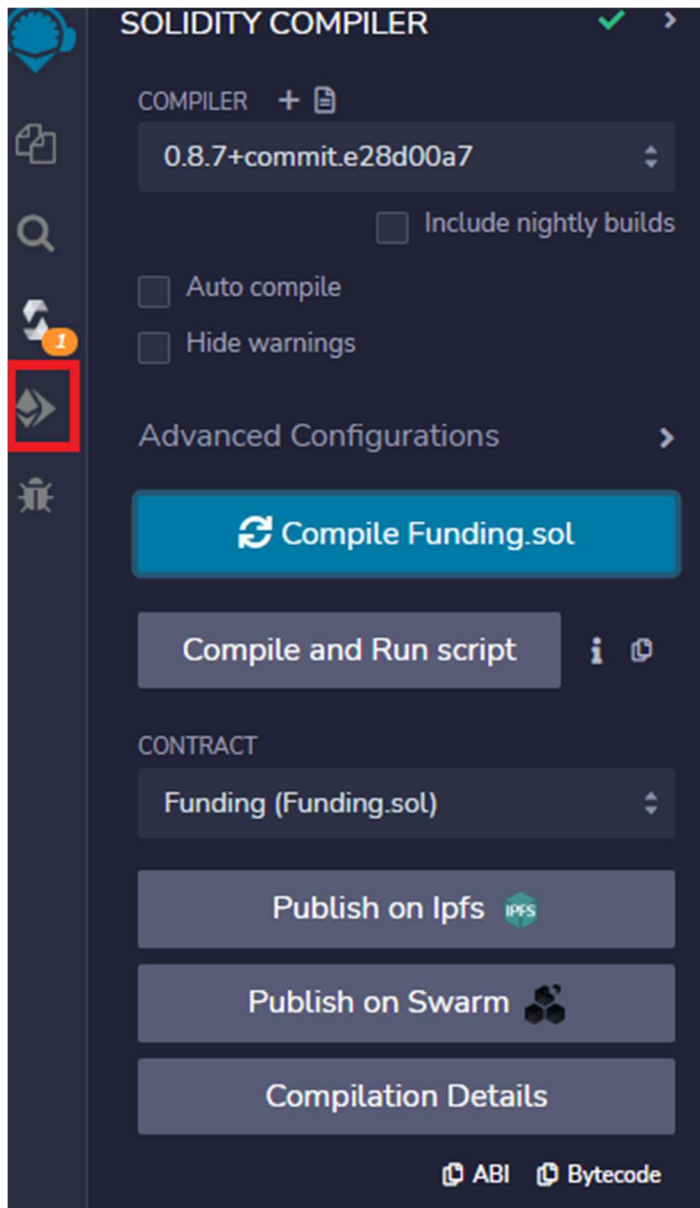
2-1. 컴파일 버튼을 눌러 컴파일



2-2. 컴파일 후 ABI 를 클릭하여 복사 후 따로 .abi 파일로 저장해두기



2-3.우측의 빨간박스를 눌러 배포하러가기

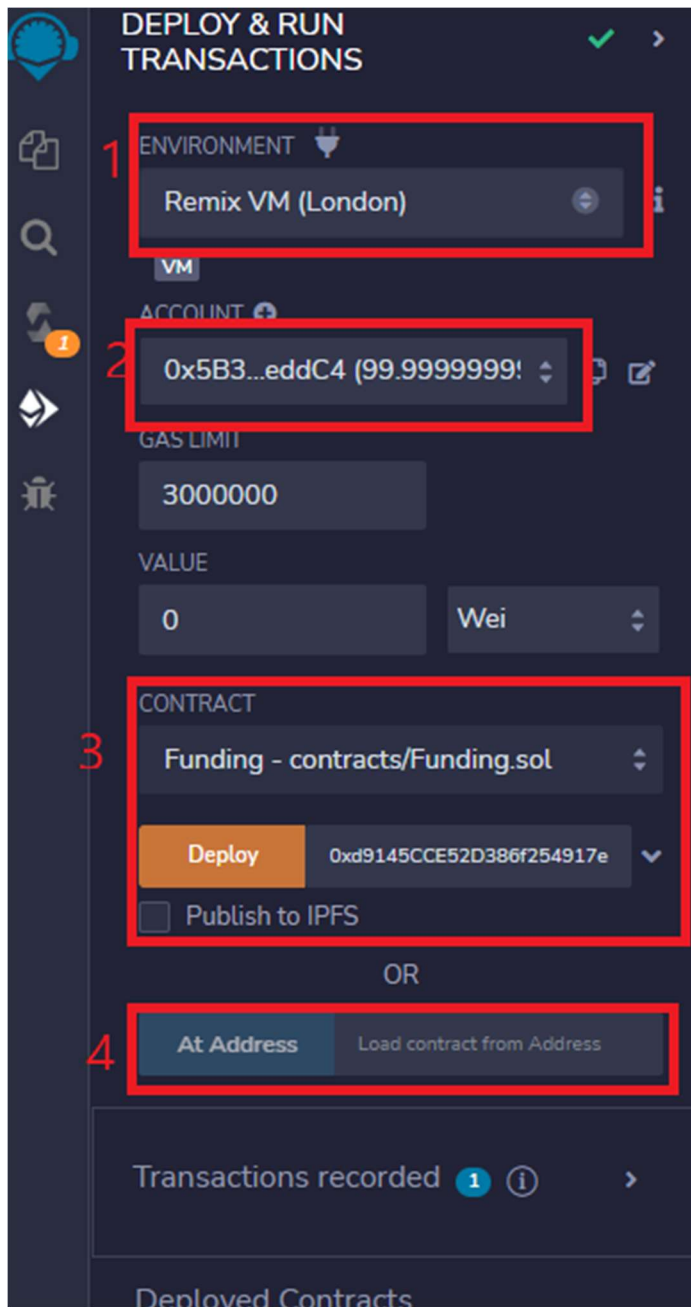


3-1. 1 번박스에 현재 연결할 네트워크 설정(메타마스크, 테스트넷, 가나슈 등)

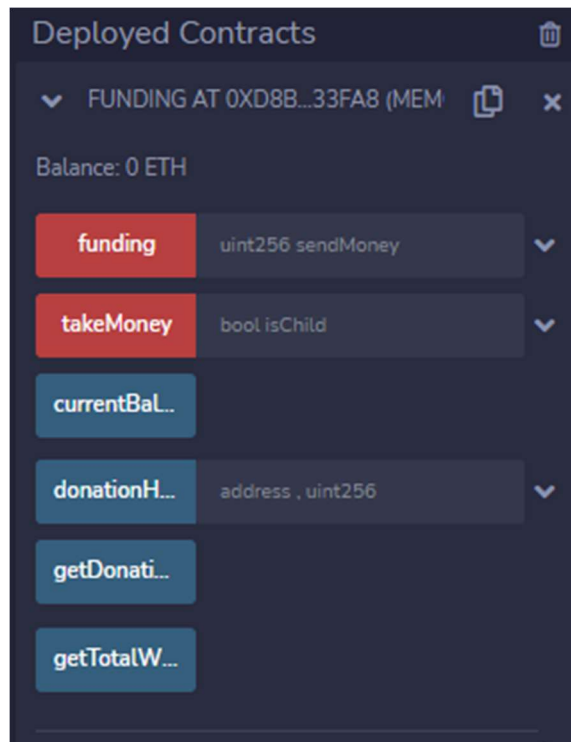
2 번 박스에 현재 컨트랙트를 배포하거나 배포된 컨트랙트에 접근하려는 계정 주소가 나옵니다.

3 번 박스에 생성자에 필요한 인자넣어서 Deploy 버튼을 눌러 배포

4 번 박스에 이미 배포된 컨트랙트 주소를 적어서 호출 가능



3-2. 배포하면 우측하단에 다음과 같이 컨트랙트에 내장된 함수들로 컨트랙트 접근 가능



4. 2-2 에서 만든 abi 파일 solc 로 컴파일

```
Shell
# web3j 를 활용하여 java wrapper 클래스 생성하기
$ web3j generate solidity -a=./Ballot.json -o=./ -p=com.ssafy.test

## option 설명
-a, --abiFile=<abiFile>          # abi 파일 경로
-b, --binFile=<binFile>          # optional, bytecode 파일 경로
-o, --outputDir=<destinationFileDir> # output 경로
-p, --package=<packageName>      # java 패키지 정보
```

백엔드 빌드 및 배포

1. AWS 터미널 접속
2. Git clone
3. Gradle Build 명령어 실행

```
# project 디렉토리로 이동
$ cd birdmeal

# gradlew 실행 권한 부여
$ chmod +x gradlew

# spring boot 프로젝트를 jar 파일로 build
$ ./gradlew clean bootJar
```

4. Dockerfile 작성

```
FROM openjdk:11
ARG JAR_FILE=birdmeal-0.0.1-SNAPSHOT.jar
WORKDIR /birdmeal
COPY ./build/libs/${JAR_FILE} app.jar
ENTRYPOINT ["java", "-jar", "app.jar"]
```

5. Dockerizing

```
# 도커 이미지 만들기
$ sudo docker build -t birdmeal-server-image .

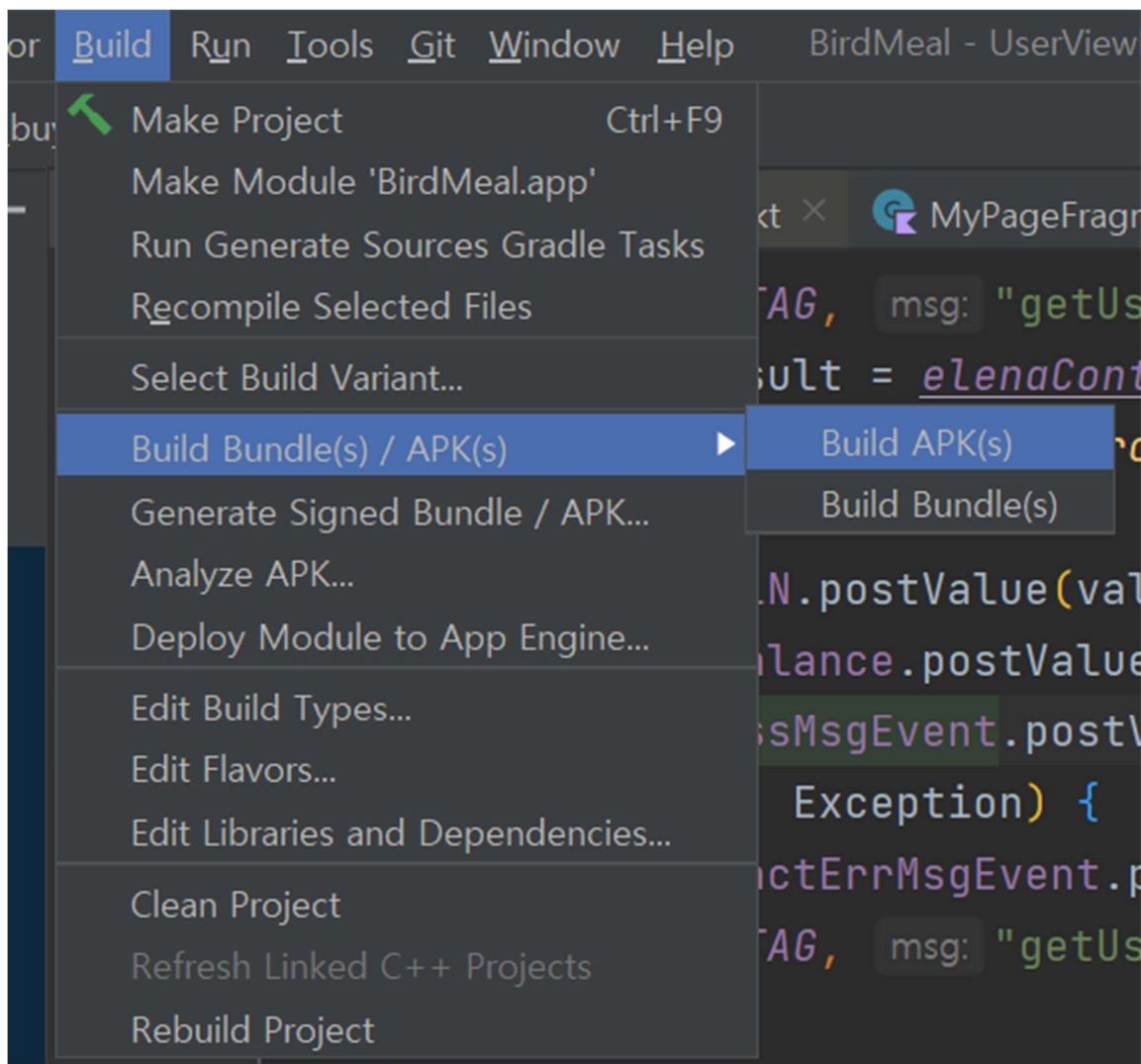
# 컨테이너 실행
$ sudo docker run -p 8081:8080 -d --name birdmeal-server birdmeal-server-image
```

안드로이드 빌드

1. 개발 환경 설정

- Android Studio Bumblebee (2021.1.1 Patch 2)
- targetSDK 30
- minSDK 24
- Kotlin

2. Apk 파일 빌드하는 방법 (Build 탭 -> Build APK(s))



웹 프론트엔드 빌드 및 배포

1. AWS 터미널 접속
2. Git clone
3. cd birdmeal-fe
4. default.conf (nginx 설정파일)

```
server {
    listen      80;
    listen  [::]:80;
    server_name localhost;
    #access_log /var/log/nginx/host.access.log  main;

    location / {
        root    /usr/share/nginx/html;
        index   index.html index.htm;
        try_files $uri $uri/ /index.html;
    }

    #error_page  404              /404.html;
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root    /usr/share/nginx/html;
    }
}
```

5. dockerfile

```
# build stage
FROM node:lts-alpine as build-stage
WORKDIR /app
COPY package*.json ./
RUN npm install
COPY . .
RUN npm run build

# production stage
FROM nginx:stable-alpine as production-stage
COPY default.conf /etc/nginx/conf.d/default.conf
COPY --from=build-stage /app/dist /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

6. Dockerizing

도커 이미지 만들기

```
$ docker build -t birdmeal-web-image .
```

컨테이너 실행

```
$ docker run -p 8443:80 -d --name birdmeal-web birdmeal-web-image
```

- 환경 변수

```
spring:
  # mysql DB 설정
  datasource:
    driver-class-name: com.mysql.cj.jdbc.Driver
    url: DB 주소
    username: 유저 아이디
    password: 유저 비밀번호
  security:
    google_id_token: [ ]
    oauth2:
      client:
        registration:
          google:
            client-id: Client 아이디
            scope: profile, email
  servlet:
    multipart:
      max-file-size: 20MB
      max-request-size: 20MB
```

```
# jpa 설정
jpa:
  database: mysql
  show-sql: true # jpa나 hibernate를 통해 CRUD를 실행하면 해당 CRUD의 sql을 로깅으로 보여준다
  database-platform: org.hibernate.dialect.MySQL5InnoDBDialect # 각기 다른 DB에 맞는 SQL 문법을 처리
  hibernate:
    ddl-auto: update # 서버를 실행할 때 마다 데이터 베이스 초기화 전략
  properties:
    hibernate:
      format_sql: false # 로깅에 표시되는 sql을 보기 좋게 해준다

jwt:
  header: Authorization
  secret: Secret Key
  access-token-validity-in-seconds: access token 유효시간 # 초 단위
  refresh-token-validity-in-seconds: refresh token 유효시간

cloud:
  aws:
    # AWS S3 bucket Info (S3 버킷정보)
    s3:
      bucket: 버킷명
    region:
      static: us-west-2
    stack:
      auto: false
```


Google Oauth 설정

<https://developers.google.com/identity/protocols/oauth2/native-app#android>

1. 구글 사용자 정보 생성
2. **Android** 애플리케이션 유형을 선택합니다.
3. OAuth 클라이언트의 이름을 입력합니다. 이 이름은 프로젝트를 식별하기 위해 프로젝트에 표시되며 [Credentials page](#)에 해당합니다.
4. Android 앱의 패키지 이름을 입력합니다. 이 값은 앱 매니페스트 파일의 [<manifest> 요소의 package 속성](#)에 정의되어 있습니다.
5. 앱 배포의 SHA-1 서명 인증서 디지털 지문을 입력합니다.
 - A. 앱이 [Google Play 앱 서명](#)을 사용하는 경우 Play Console의 앱 서명 페이지에서 SHA-1 디지털 지문을 복사합니다.
 - B. 자체 키 저장소와 서명 키를 관리하는 경우, 자바에 포함된 keytool 유틸리티를 사용하여 사람이 읽을 수 있는 형식으로 인증서 정보를 인쇄합니다. keytool 출력의 Certificate fingerprints 섹션에 SHA1 값을 복사합니다. 자세한 내용은 Android용 Google API 문서의 [클라이언트 인증](#)을 참조하세요.
6. 만들기를 클릭합니다.
7. .apk 파일 또는 GitLab AOS 폴더 파일 Android Studio로 빌드 후 사용

MetaMask 설치

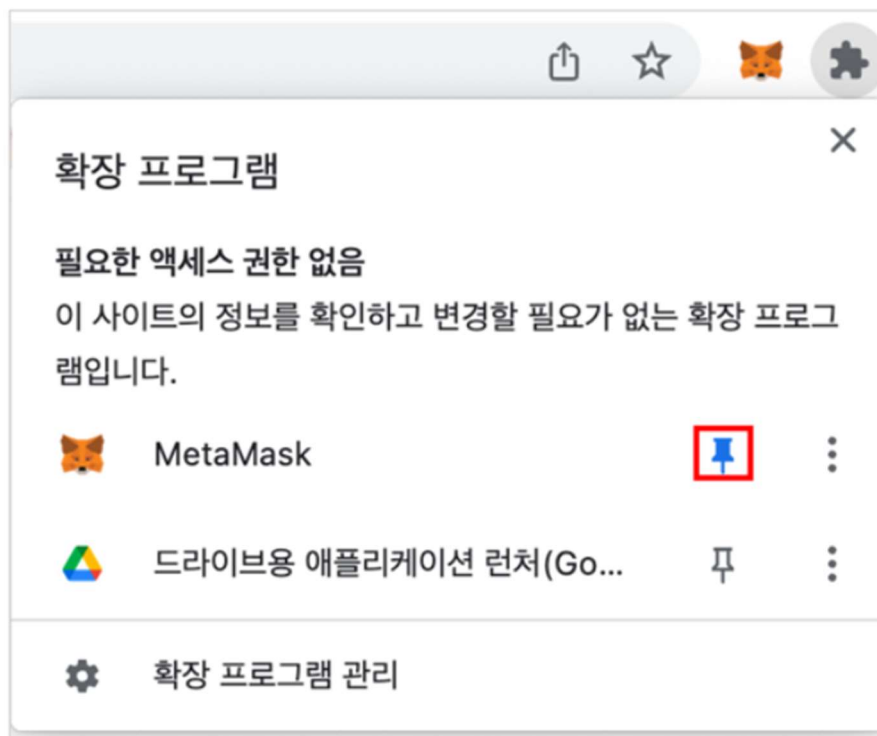
1. 크롬 브라우저에서 확장 프로그램 설치 페이지 접속 및 추가

크롬 브라우저에서 아래의 확장 프로그램 설치 페이지에 접속하고 “Chrome에 추가” 버튼을 클릭해서 확장 프로그램을 설치합니다.




2. 확장 프로그램 탭에서 메타마스크 지갑 고정


메타마스크 지갑을 편리하게 이용하기 위해서 크롬 브라우저 우측 상단의 퍼즐 모양 아이콘(확장프로그램)을 클릭한 뒤, 메타마스크(MetaMask) 우측 핀 모양 버튼을 클릭하여 파란색 모양으로 활성화시킵니다.



3. 메타마스크 지갑 생성


 METAMASK

MetaMask가 처음이세요?




아니요, 이미 비밀 복구 구문이 있습니다.
비밀 복구 구문을 사용하여 기존 지갑 가져오기


지갑 가져오기



설정을 시작하죠!
이렇게 하면 새 지갑과 비밀 복구 구문이 만들어집니다

지갑 생성

 METAMASK



MetaMask 개선에 참여

MetaMask는 사용자가 확장 프로그램 상호작용하는 방식을 자세히 이해하기 위해 사용 데이터를 수집하고자 합니다. 수집한 데이터는 당사의 제품과 이더리움 에코시스템의 사용 편의성 및 사용자 경험을 지속적으로 개선하는 데 활용됩니다.

MetaMask에서는..

- ✓ 언제든지 설정을 통해 옵트아웃할 수 있습니다.
- ✓ 익명화된 클릭 및 페이지뷰 이벤트 보내기
- ✗ 키, 주소, 거래, 잔액, 해시 또는 개인 정보를 절대 수집하지 않습니다.
- ✗ 전체 IP 주소를 절대 수집하지 않습니다.
- ✗ 절대로 수익을 위해 데이터를 판매하지 않습니다!

관참합니다

동의함

이 데이터는 집계 처리된 정보이며 일반 데이터 보호 규정 (EU) 2016/679의 목적에 따라 익명으로 관리됩니다. 당사의 개인정보 보호정책에 관한 자세한 내용은 [개인정보 보호정책](#)을 참조하세요.

4. 비밀번호 설정



비밀번호 만들기

새 비밀번호(8자 이상)

비밀번호 확인

☐ [이용 약관](#)의 내용을 읽고 이에 동의합니다.

생성

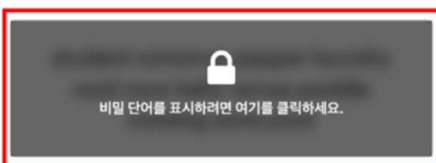
5. 니모닉(비밀 복구 구문) 생성



비밀 복구 구문

비밀 백업 구문을 이용하면 계정을 쉽게 백업하고 복구할 수 있습니다.

경고: 비밀 복구 구문은 절대로 공개하지 마세요. 이 구문이 있는 사람은 귀하의 Ether를 영원히 소유할 수 있습니다.



나중에 알림

다음

팁:

이 구문을 1Password 같은 비밀번호 관리자에 저장하세요.

메모지에 이 구문을 적어 안전한 곳에 보관하세요. 보안을 더욱 강화하고 싶다면 여러 메모지에 적은 다음 2-3곳에 보관하세요.

이 구문을 기억하세요.

이 비밀 복구 구문을 다운로드하여 암호화된 외장 하드 드라이브나 저장 매체에 안전하게 보관하세요.

6. 비밀 백업 구문 확인

 METAMASK

< 뒤로

비밀 백업 구문 확인

각 구문을 선택하여 구문이 올바른지 확인하세요.

확인

7. 지갑 생성 완료

