



Starry Night

PORTING MANUAL

목 차

I. 빌드 및 배포

1. 개발 환경	3
2. 설정 파일 목록	3
3. 설정 파일 및 환경 변수 정보	4
4. Docker 설치	9
5. SSL 인증서 발급	10
6. DB 및 Infra 배포	11
7. Backend CI/CD	14
8. Frontend CI/CD	17
9. Unity 빌드	21

II. 외부 서비스

1. 소셜 로그인	21
2. Photon Network 연동	27

I. 빌드 및 배포

1. 개발 환경

Server : AWS EC2 Ubuntu 20.04 LTS

Visual Studio Code : 1.70.1

IntelliJ IDEA : 2022.1.3(Ultimate Edition) 11.0.15 + 10-b2043.56 amd64

JVM : 11.0.16+8-post-Ubuntu-0ubuntu120.04

Docker : 20.10.21

Node.js : 18.12.1

MySQL : 8.0.31-1.el8

Nginx : 1.23.2

Jenkins : 2.361.3

Unity : 2021.3.11f1

2. 설정 파일 목록

React

- **Dockerfile** : /jenkins/workspace/frontend/frontend

- **.env** : /jenkins/workspace/frontend/frontend

SpringBoot

- **application.properties** :

/jenkins/workspace/backend/backend/src/main/resources

- **application-oauth.properties** :

/jenkins/workspace/backend/backend/src/main/resources

- **Dockerfile** : /jenkins/workspace/backend/backend

- **deploy.sh** : /jenkins/workspace/backend/backend

Docker

- **docker-compose.yml**: /home/ubuntu

Nginx

- **app.conf** : /home/ubuntu/nginx/conf.d

3. 설정 파일 및 환경 변수 정보

React

- **Dockerfile**

```
FROM node:alpine
WORKDIR /usr/src/app
COPY ./package* /usr/src/app/
RUN npm install
COPY ./ /usr/src/app/
CMD ["npm","run","start"]
```

Springboot

- application.properties

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://mysql:3306/dreammungz?useSSL=false&useUnicode=true&serverTimezone=Asia/Seoul
spring.datasource.username={MySQL 사용자 이름}
spring.datasource.password={MySQL 패스워드}

spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true

server.port=8081
```

- application-oauth.properties

```
#Google
spring.security.oauth2.client.registration.google.client-id= 구글
클라이언트 ID
spring.security.oauth2.client.registration.google.client-secret= 구글
클라이언트 SECRET
spring.security.oauth2.client.registration.google.redirect-uri= 구글
리다이렉트 URI
spring.security.oauth2.client.registration.google.scope= profile, email
#Kakao
spring.security.oauth2.client.registration.kakao.client-id=카카오
클라이언트 ID
spring.security.oauth2.client.registration.kakao.client-secret=카카오
클라이언트 SECRET
spring.security.oauth2.client.registration.kakao.redirect-uri=카카오
리다이렉트 URI
```

```

spring.security.oauth2.client.registration.kakao.client-authentication-
method=POST
spring.security.oauth2.client.registration.kakao.authorization-grant-type =
authorization_code
spring.security.oauth2.client.registration.kakao.scope=profile_nickname,
account_email
spring.security.oauth2.client.registration.kakao.client-name=Kakao
#Provider-Kakao
spring.security.oauth2.client.provider.kakao.authorization-
uri=https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-
uri=https://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-
uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=id

```

- Dockerfile

```

FROM openjdk:11-jdk
ARG JAR_FILE=build/libs/*.jar
COPY ${JAR_FILE} app.jar
ENTRYPOINT ["java","-jar","-Duser.timezone=Asia/Seoul","/app.jar"]

```

- deploy.sh

```

echo '실행 시작'

echo 'git pull'

echo 'jar 파일 삭제'
rm build/libs/*.jar

```

```
echo '빌드 전 cleanQuery'
./gradlew cleanQuerydslSourceDir

echo '빌드 시작'
./gradlew build

echo '도커파일 이미지 빌드'
docker build -t springbootapp .

echo '컨테이너 중지'
docker stop springbootapp

echo '컨테이너 삭제'
docker rm springbootapp

echo '컨테이너 실행'
docker run -p 8081:8081 --name springbootapp --network
ubuntu_default -d springbootapp
```

Docker

- docker-compose.yml

```
version: "3"
services:
  mysql:
    image: mysql

    container_name: mysql
    environment:
      MYSQL_DATABASE: {scheme명}
      MYSQL_ROOT_PASSWORD: {root 계정 패스워드}
    volumes:
```

- /mysql:/var/lib/mysql

ports:

- 3306:3306

nginx:

image: nginx

container_name: nginx

ports:

- 80:80
- 443:443

volumes:

- /etc/letsencrypt:/etc/letsencrypt
- ./nginx/conf.d:/etc/nginx/conf.d

jenkins:

image: jenkins/jenkins:lts

container_name: jenkins

volumes:

- /var/run/docker.sock:/var/run/docker.sock
- /jenkins:/var/jenkins_home

ports:

- 9090:8080

privileged: true

user: root

Nginx

- app.conf

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;
    server_name starry-night.kr www.starry-night.kr;
```



```

        return 301 https://$server_name$request_uri;
    }

    server {
        listen 443 ssl;
        listen [::]:443 ssl;
        server_name starry-night.kr;
        access_log off;
        ssl_certificate /etc/letsencrypt/live/starry-night.kr/fullchain.pem;
        ssl_certificate_key /etc/letsencrypt/live/starry-
night.kr/privkey.pem;

        location /api/ {
            proxy_pass http://starry-night.kr:8081/;
            proxy_set_header Host $host;
            proxy_redirect off;
        }

        location / {
            proxy_pass http://starry-night.kr:3000;
            proxy_set_header Host $host;
            proxy_redirect off;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";
        }
    }
}

```

4. Docker 설치

- Docker Install

```
sudo apt-get update
```

```

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

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo 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/ubuntu $(lsb_release -cs) stable" |
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

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

```

- Docker Compose Install

```

sudo curl -L
"https://github.com/docker/compose/releases/download/1.29.2/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

```

5. SSL 인증서 발급

```

sudo apt-get install letsencrypt

sudo letsencrypt certonly --standalone -d www제외한 도메인 이름

```

이메일 작성 후 Agree

뉴스레터 수신 여부 Yes/No

해당 경로에 Key 생성 여부 확인

ssl_certificate /etc/letsencrypt/live/{도메인}/fullchain.pem

ssl_certificate_key /etc/letsencrypt/live/{도메인}/privkey.pem

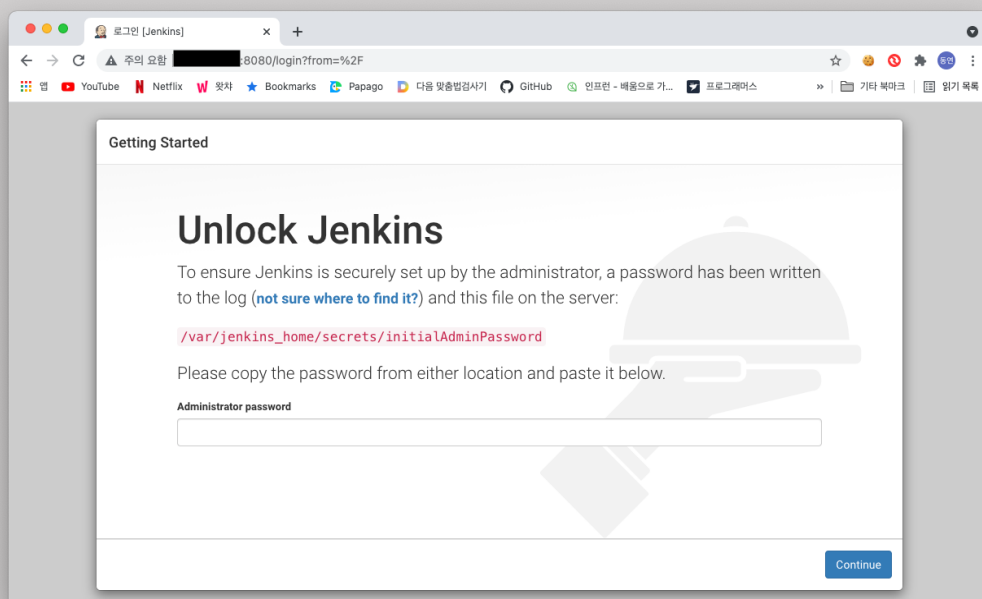
6. Infra 배포

- docker-compose 실행

```
cd /home/ubuntu/mysql,jenkins,nginx (docker-compose.yml 경로에서)
```

```
sudo docker-compose up --build -d
```

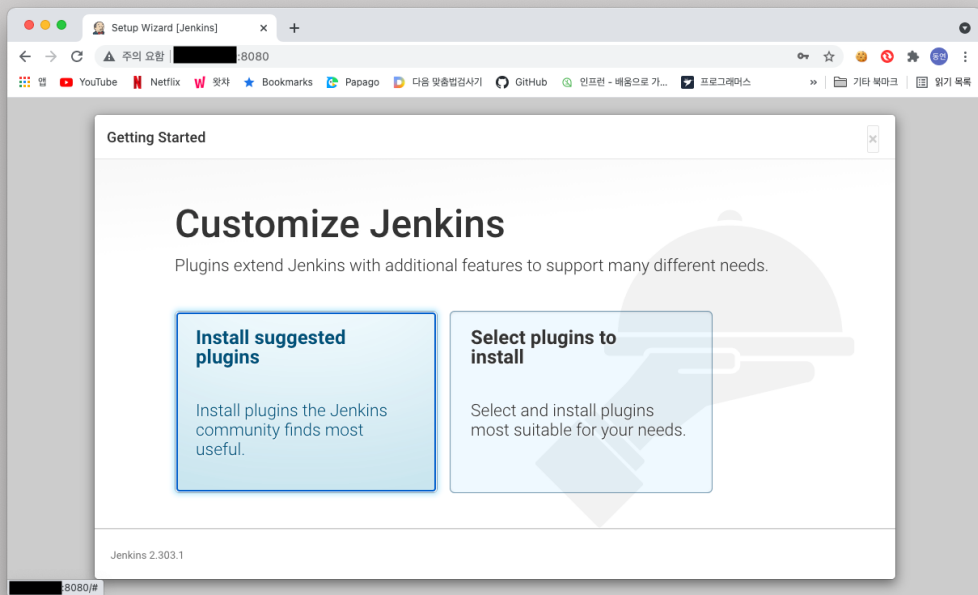
- Jenkins 플러그인 및 docker-in-docker 설치



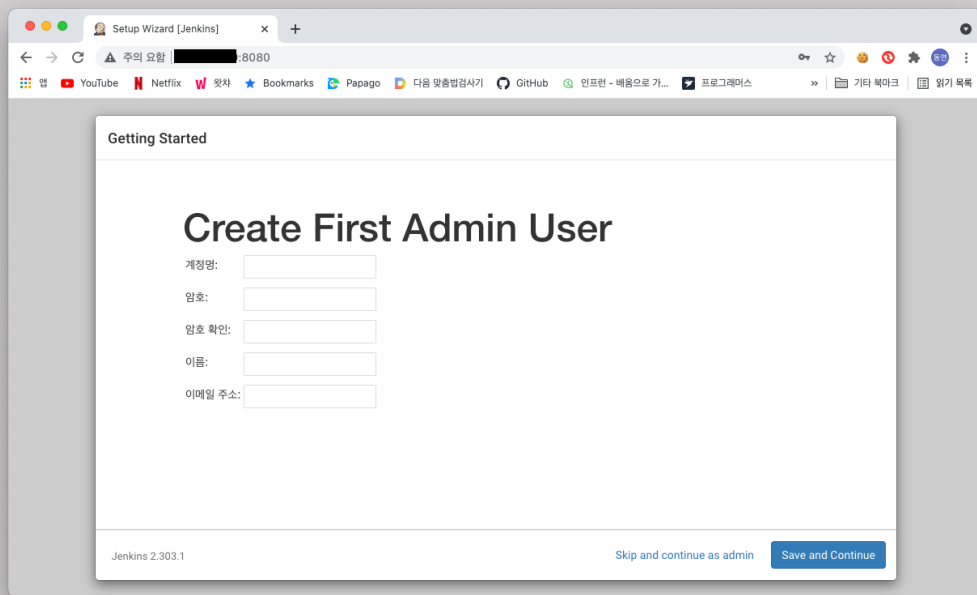
1. http://서비스도메인:9090 으로 접속하여 jenkins 페이지 진입

```
jenkins_cicd | 2021-09-03 10:38:01.280+0000 [id=29] INFO jenkins.InitReactorRunner$1onAttained: System config adapted
jenkins_cicd | 2021-09-03 10:38:01.280+0000 [id=29] INFO jenkins.InitReactorRunner$1onAttained: Loaded all jobs
jenkins_cicd | 2021-09-03 10:38:01.281+0000 [id=29] INFO jenkins.InitReactorRunner$1onAttained: Configuration for all jobs updated
jenkins_cicd | 2021-09-03 10:38:01.509+0000 [id=29] INFO jenkins.install.SetupWizard#init:
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | Jenkins initial setup is required. An admin user has been created and a password generated.
jenkins_cicd | Please use the following password to proceed to installation:
jenkins_cicd | 6d67dc42900?*****
jenkins_cicd | This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | *****
jenkins_cicd | 2021-09-03 10:38:02.043+0000 [id=42] INFO hudson.model.AsyncPeriodicWork$lambda$doRun$0: Started Download metadata
jenkins_cicd | 2021-09-03 10:38:02.049+0000 [id=42] INFO hudson.model.AsyncPeriodicWork$lambda$doRun$0: Finished Download metadata. 2 ms
jenkins_cicd | 2021-09-03 10:38:16.439+0000 [id=28] INFO jenkins.InitReactorRunner$1onAttained: Completed initialization
jenkins_cicd | 2021-09-03 10:38:16.642+0000 [id=22] INFO hudson.WebAppMain$3run: Jenkins is fully up and running
ubuntu@ip-172-31-11-81: ~
```

2. 서버 콘솔에서 `sudo docker logs [Jenkins 컨테이너 이름]`으로 Administrator password를 확인하고 입력

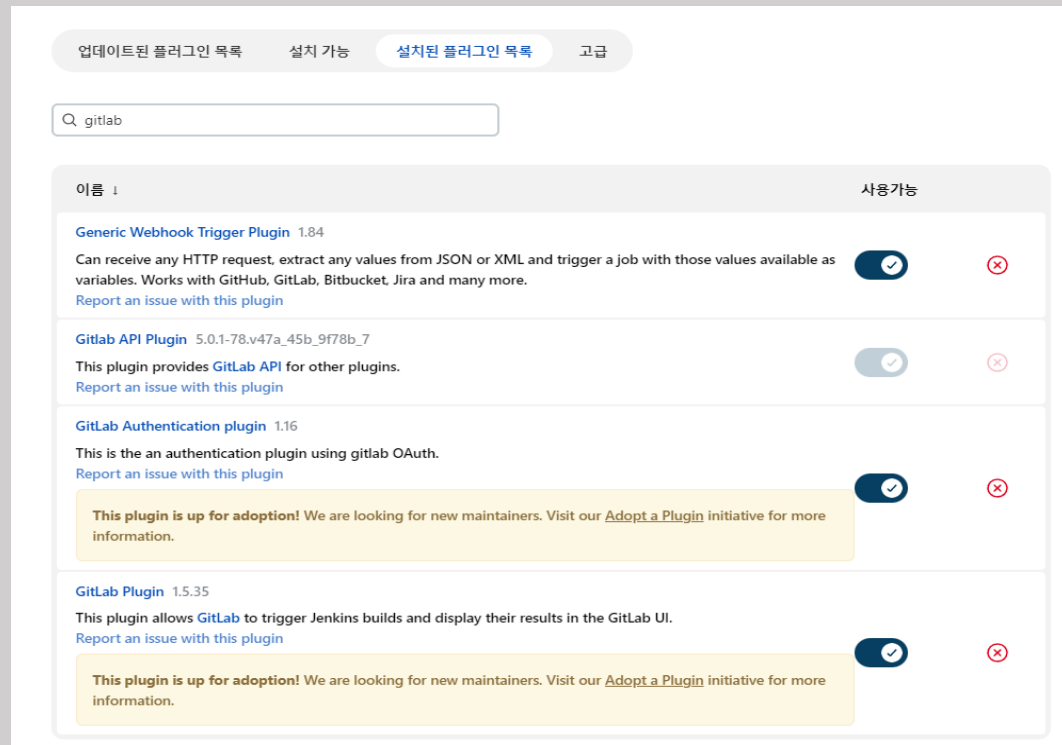


3. Install suggested plugins을 선택하여 플러그인들을 설치



4. 생성할 관리자 계정 정보를 입력하고 Save and Continue

5. Jenkins 접속 URL확인 후 Save and Finish



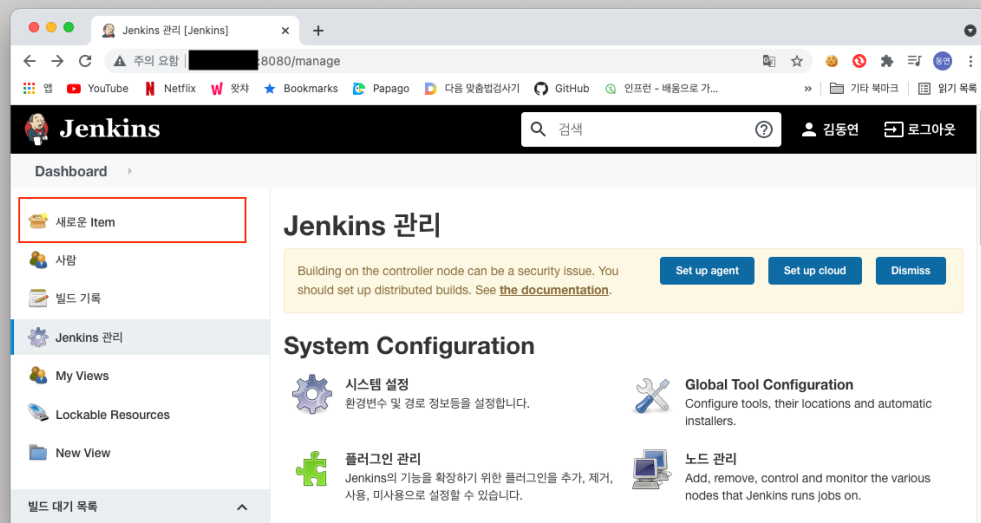
6. 메인 화면에서 DashBoard -> Manager JenKins -> Plugin Manager에서 gitlab, docker에 대해서 플러그인을 설치

7. 서버 콘솔로 돌아가 Jenkins 내부에 docker를 설치

```
sudo docker exec -it {jenkins 컨테이너 이름} /bin/bash
apt-get update -y
apt-get install -y
apt-get install docker.io -y
docker -v
```

7. Backend CI/CD

- Jenkins Job 설정



1. Jenkins 메인 화면 -> Dashboard -> 새로운 Item 클릭

2. FreeStyle project를 선택하고 item name은 backend로 설정 후 OK

소스 코드 관리

☐ None

☒ Git ?

Repositories ?

Repository URL ?

[Redacted]

Credentials ?

[Redacted]

+ Add

고급...

3. Repositories URL에는 프로젝트 레포지토리의 HTTPS Clone 주소 입력
Credentials의 Add를 클릭,

Domain -> Global credentials

Kind -> Username with password

Username -> 레포지토리 접근 권한이 있는 Gitlab 계정 아이디

Password -> Username에 작성한 Gitlab 계정 비밀번호

순서대로 입력하고 드롭박스에서 생성된 Credential을 선택

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/heads/backend

Add Branch

4. backend 브랜치의 내용만을 받아와서 빌드하기 위한 설정

빌드 유발

☐ 빌드를 원칙으로 유발 (예: 스크립트 사용) ?

☐ Build after other projects are built ?

☐ Build periodically ?

☒ Build when a change is pushed to GitLab. GitLab webhook URL: <http://3.39.251.36:9090/project/backend> ?

Enabled GitLab triggers

☒ Push Events

☐ Push Events in case of branch delete

☒ Opened Merge Request Events

☐ Build only if new commits were pushed to Merge Request ?

☐ Accepted Merge Request Events

☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

☒ Approved Merge Requests (EE-only)

☒ Comments

저장 **Apply**

5. 빌드 유발을 다음과 같이 설정
하단의 고급 기능을 눌러 Secret Token을 Generate하여 기록

Build

Execute shell ?

Command

See [the list of available environment variables](#)

```
echo 'jenkinsbuild started...'
pwd
cd backend
chmod +x gradlew
```

고급...

Add build step ▾

6. Build 탭에 Excute shell을 선택하고
cd backend
chmod +x gradlew
chmod +x deploy.sh
./deploy.sh

- Jenkins에 sudo 권한 부여


```
sudo vim /etc/sudoers
```

```
# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL
jenkins ALL=(ALL) NOPASSWD: ALL
```

해당 부분에 jenkins ALL=(ALL) NOPASSWD: ALL 작성

- Gitlab Webhook 설정

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

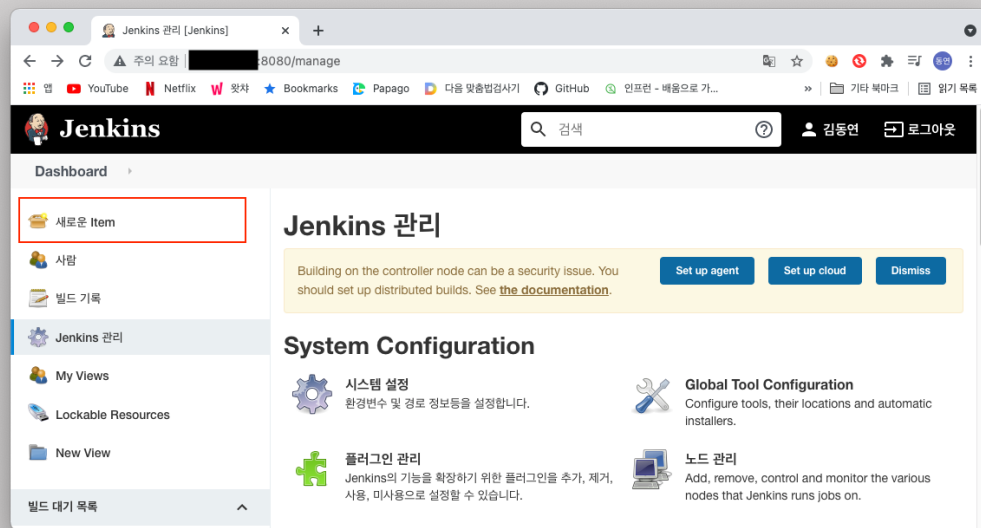
☒ Push events

Push to the repository.

Gitlab 프로젝트 Repository의 Setting > Webhooks 로 이동
 URL에는 Jenkins의 빌드 유발의 Webhook URL을 입력
 Secret Token에는 빌드 유발에서 생성했던 Secret Token을 입력
 Trigger의 Push events를 체크하고 backend 입력 후 Add webhook

8. Frontend CI/CD

- Jenkins Job 설정



1. Jenkins 메인 화면 -> Dashboard -> 새로운 Item 클릭
2. FreeStyle project를 선택하고 item name은 backend로 설정 후 OK



3. Repositories URL에는 프로젝트 레포지토리의 HTTPS Clone 주소 입력
Credentials의 Add를 클릭,
Domain -> Global credentials
Kind -> Username with password
Username -> 레포지토리 접근 권한이 있는 Gitlab 계정 아이디
Password -> Username에 작성한 Gitlab 계정 비밀번호

순서대로 입력하고 드롭박스에서 생성된 Credential을 선택

Branches to build ?

Branch Specifier (blank for 'any') ?

refs/heads/frontend

Add Branch

4. Frontend 브랜치의 내용만을 받아와서 빌드하기 위한 설정

빌드 유발

☐ 빌드를 원칙으로 유발 (예: 스크립트 사용) ?
☐ Build after other projects are built ?
☐ Build periodically ?
☒ Build when a change is pushed to GitLab. GitLab webhook URL: http://3.39.251.36:9090/project/backend ?

Enabled GitLab triggers

☒ Push Events
☐ Push Events in case of branch delete
☒ Opened Merge Request Events
☐ Build only if new commits were pushed to Merge Request ?
☐ Accepted Merge Request Events
☐ Closed Merge Request Events

Rebuild open Merge Requests

Never

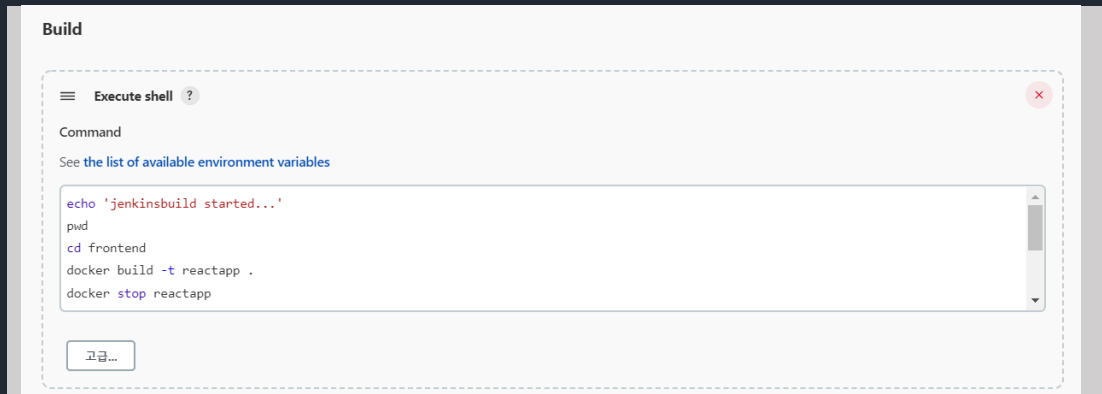
☒ Approved Merge Requests (EE-only)
☒ Comments

저장

Apply

5. 빌드 유발을 다음과 같이 설정

하단의 고급 기능을 눌러 Secret Token을 Generate하여 기록



6. Build 탭에 Execute shell을 선택하고

cd frontend

docker build -t reactapp .

docker stop reactapp

docker rm reactapp

docker run -p 3000:3000 --name reactapp --network ubuntu_default -d reactapp

- Gitlab Webhook 설정

Webhooks

Webhooks enable you to send notifications to web applications in response to events in a group or project. We recommend using an [integration](#) in preference to a webhook.

URL

URL must be percent-encoded if it contains one or more special characters.

Secret token

Used to validate received payloads. Sent with the request in the `X-Gitlab-Token` HTTP header.

Trigger

☒ Push events

Push to the repository.

Gitlab 프로젝트 Repository의 Setting > Webhooks 로 이동

URL에는 Jenkins의 빌드 유발의 Webhook URL을 입력

Secret Token에는 빌드 유발에서 생성했던 Secret Token을 입력

Trigger의 Push events를 체크하고 backend 입력 후 Add Webhook

9. Unity 빌드

- Unity 프로젝트 빌드

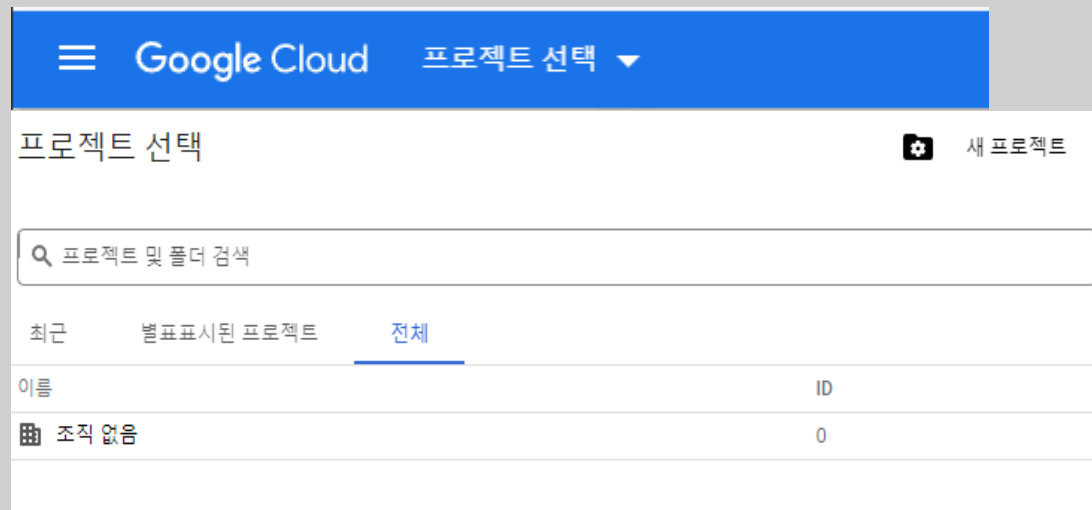
1. Unity 프로젝트를 실행하고 Ctrl + Shift + B
2. 플랫폼을 PC, Mac & Linux Standalone으로 설정
3. Build 클릭

II. 외부 서비스

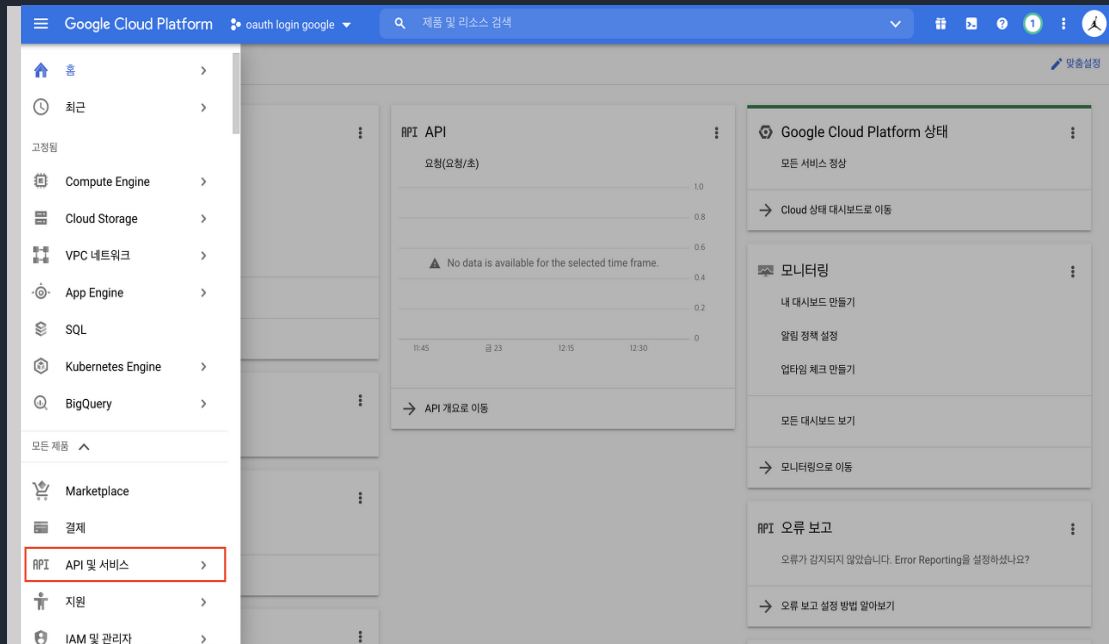
1. 소셜 로그인

- 구글(Google)

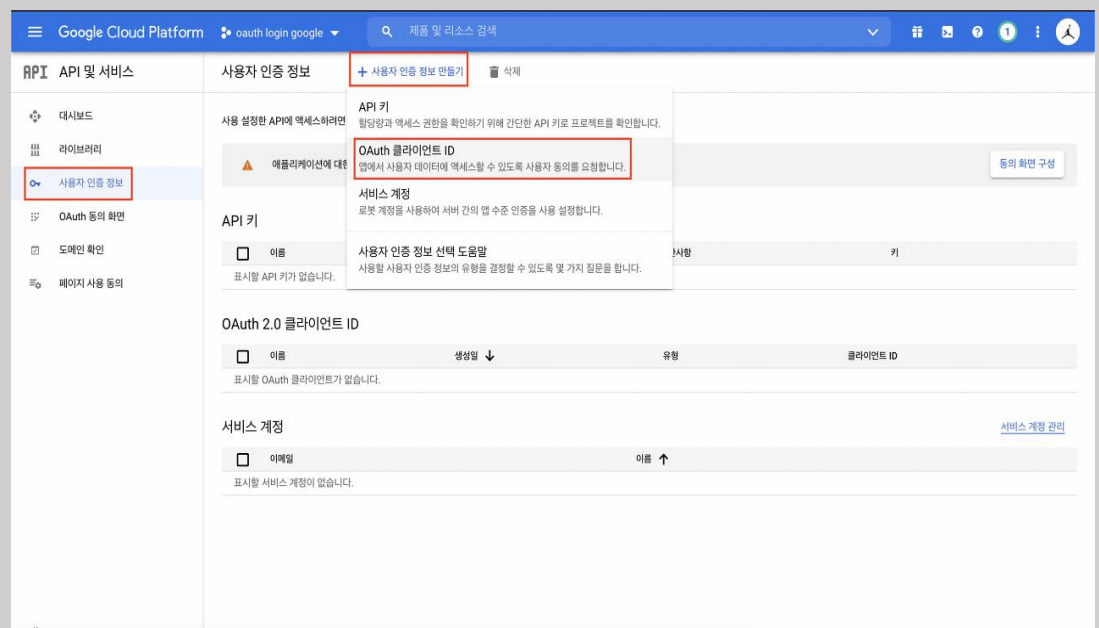
1. <https://console.cloud.google.com/home/dashboard> 로 접속한다.



2. 프로젝트 선택을 클릭하고 모달 우측 상단의 새 프로젝트 클릭



3. API 및 서비스 선택



4. 사용자 인증정보 > 사용자 인증 정보 만들기 > OAuth 클라이언트 ID

OAuth 동의 화면

대상 사용자를 비롯해 앱을 구성하고 등록하려는 방식을 선택하세요. 프로젝트에는 하나의 앱만 연결할 수 있습니다.

User Type

☐ 내부 ?

Google Workspace 사용자가 아니기 때문에 앱을 외부(일반 잠재고객) 사용자에게 제공하는 것만 가능합니다. [외부 앱을 제출할 필요는 없](#)

☒ 외부 ?

Google 계정이 있는 모든 테스트 사용자가 사용할 수 있습니다. 앱이 테스트 모드로 시작되고 테스트 사용자 목록에 추가된 사용자에게만 제공됩니다. 앱을 프로젝트에 푸시할 준비가 되면 앱을 인증해야 할 수도 있습니다.

[사용자 유형 자세히 알아보기](#)

만들기

5. 동의 화면 구성을 클릭하고 해당 화면에서 외부로 선택하여 외부 사용자들이 사용할 수 있도록 설정

앱 정보

동의 화면에 표시되어 최종 사용자가 개발자를 확인하고 문의할 수 있습니다.

앱 이름 *

StarryNight

동의를 요청하는 앱의 이름

사용자 지원 이메일 *

platinadark@gmail.com

사용자가 동의 관련 질문을 위해 문의할 때 이용합니다.

앱 로고

찾아보기

사용자가 앱을 알아보는 데 도움이 되도록 동의 화면에 대한 이미지(1MB 이하 크기)를 업로드합니다. 허용되는 이미지 형식은 JPG, PNG, BMP입니다. 최적의 결과를 위해서는 로고가 120x120픽셀 크기의 정사각형이어야 합니다.



6. 앱 정보를 입력하고, 개발자 연락처 정보에 이메일 작성

Google Cloud zipzoom

API 및 서비스

- 사용 설정된 API 및 서비스
- 라이브러리
- 사용자 인증 정보
- OAuth 동의 화면**
- 도메인 확인
- 페이지 사용 동의

앱 등록 수정

1 OAuth 동의 화면 2 범위 3 테스트 사용자 4 요약

범위는 사용자에게 앱 승인을 요청하는 권한을 나타내며 프로젝트에서 사용자의 Google 계정에 있는 특정 유형의 비공개 사용자 데이터에 액세스하도록 허용합니다. 자세히 알아보기

범위 추가 또는 삭제

민감하지 않은 범위

API	범위	사용자에게 표시되는 설명
...	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
...	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기

Your sensitive scopes

민감한 범위는 비공개 사용자 데이터에 대한 액세스를 요청하는 범위입니다.

API	범위	사용자에게 표시되는 설명
...	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
...	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기

선택한 범위 업데이트

여러개의 사용 설정된 API의 범위가 나와 있습니다. 이 화면에 누락된 범위를 추가하려면 Google API 라이브러리에서 API를 찾아 사용 설정하거나 아래의 '별명'은 범위 엑스트라 설정을 사용하세요. 라이브러리에서 사용 설정한 새 API를 확인하려면 페이지를 새로고침하세요.

필터 속성 이름 또는 값 입력

API	범위	사용자에게 표시되는 설명
...	../auth/userinfo.email	기본 Google 계정의 이메일 주소 확인
...	../auth/userinfo.profile	개인정보(공개로 설정한 개인정보 포함) 보기
...	openid	Google에서 내 개인 정보를 나와 연결
BigQuery API	../auth/bigquery	View and manage your data in Google BigQuery and see the email address for your Google Account
BigQuery API	../auth/cloud-platform	Google Cloud 데이터 확인, 수정, 구성, 삭제 및 Google 계정의 이메일 주소 확인
BigQuery API	../auth/bigquery.readonly	Google BigQuery에서 데이터를 봅니다.
BigQuery API	../auth/cloud-platform.read-only	Google Cloud 서비스 전체의 데이터 조회 및 Google 계정의 이메일 주소 확인
BigQuery API	../auth/devstorage.full_control	Manage your data and permissions in Cloud Storage and see the email address for your Google Account
BigQuery API	../auth/devstorage.read_only	Google 클라우드 저장소에서 데이터 조회
BigQuery API	../auth/devstorage.read.write	Cloud Storage의 데이터 관리 및 Google 계정의 이메일 주소 확인

페이지당 행 수: 10 1 - 10 (전체 25행)

7. 범위 추가 또는 삭제 > email, profile 선택 후 저장 민감하지 않은 범위에 추가된 것 확인

이름 *

StarryNight

OAuth 2.0 클라이언트의 이름입니다. 이 이름은 콘솔에서 클라이언트를 식별하는 용도로만 사용되며 최종 사용자에게 표시되지 않습니다.



아래에 추가한 URI의 도메인이 [승인된 도메인](#)으로 [OAuth 동의 화면](#)에 자동으로 추가됩니다.

승인된 자바스크립트 원본 ?

브라우저 요청에 사용

+ URI 추가

승인된 리디렉션 URI ?

웹 서버의 요청에 사용

URI 1 *

<https://starry-night.kr/api/login/oauth2/code/google>

URI 2 *

<https://K7A208.p.ssafy.io/api/login/oauth2/code/google>

+ URI 추가

8. 사용자 인증정보 > 사용자 인증정보 만들기 -> Oauth 클라이언트 ID
승인된 리디렉션 URI 작성

- 카카오(Kakao)

1. <https://developers.kakao.com/> 로 접속

2. 내 애플리케이션 > 애플리케이션 추가하기 > 앱 이름 입력 > 사업자
명 입력 > 저장

3. 좌측 Nav 바에서 앱 설정 > 요약 정보 > 앱 키 > REST API키

제품 설정 > 카카오 로그인 > 보안 > Client Secret의 코드 발급

Web 플랫폼 수정

사이트 도메인
 JavaScript SDK, 카카오톡 공유, 카카오맵, 메시지 API 사용시 등록이 필요합니다.
 여러개의 도메인은 줄바꿈으로 추가해주세요. 최대 10까지 등록 가능합니다. 추가 등록은 포럼(데브북)으로 문의주세요.
 예시: (O) https://example.com (X) https://www.example.com

https://starry-night.kr
 https://K7A208.p.ssafy.io

기본 도메인
 기본 도메인은 첫 번째 사이트 도메인으로, 카카오톡 공유와 카카오톡 메시지 API를 통해 발송되는 메시지의 Web 링크 기본값으로 사용됩니다.

https://starry-night.kr

취소 저장

4. 좌측 Nav 바에서 앱 설정 > 플랫폼 > Web > Web 플랫폼 등록 필요 정보를 기입

Redirect URI 삭제

Redirect URI	
https://starry-night.kr/api/login/oauth2/code/kakao	
https://K7A208.p.ssafy.io/api/login/oauth2/code/kakao	

- 카카오 로그인에서 사용할 OAuth Redirect URI를 설정합니다. (최대 10개)
- REST API로 개발하는 경우 필수로 설정해야 합니다.

5. 제품 설정 > 카카오 로그인 클릭 Redirect URI를 입력

개인정보			
항목 이름	ID	상태	
닉네임	profile_nickname	필수 동의	설정
프로필 사진	profile_image	필수 동의	설정
카카오계정(이메일)	account_email	선택 동의 [수집]	설정

6. 제품 설정 > 카카오 로그인 > 동의항목에서 위치권 설정

2. Photon Network 연동

1. <https://www.photonengine.com/ko-KR/> 에 접속하여 회원가입



2. 새 어플리케이션 만들기를 클릭

새 어플리케이션 작성하기

어플리케이션의 기본구성은 **무료 플랜**으로 구성되어 있습니다.
언제든지 유료 플랜으로 변경이 가능합니다.

Photon 종류 * 1

PUN

이름 * 2

TestPhoton

어플리케이션 설명

구체적으로 기입해 주십시오. 최대 1024자 까지 입력하실 수 있습니다.

URL

http://enter.your-url.here/

작성하기 3 또는 [뒤로가기](#).

3. Photon 종류를 선택하고 어플리케이션 이름을 입력 후 작성하기

PUN
20 CCU

TestPhoton

어플리케이션 ID: e8c5d71c

최대 수 CCU

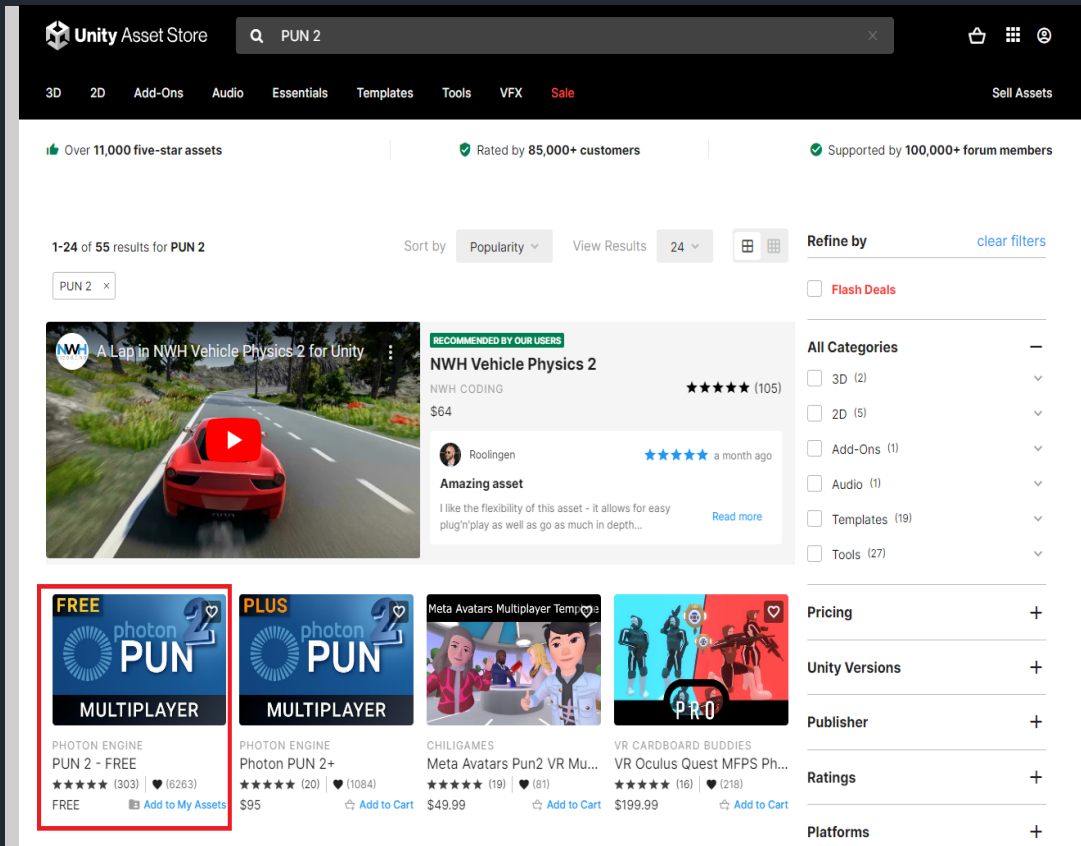
0

트래픽 사용 완료

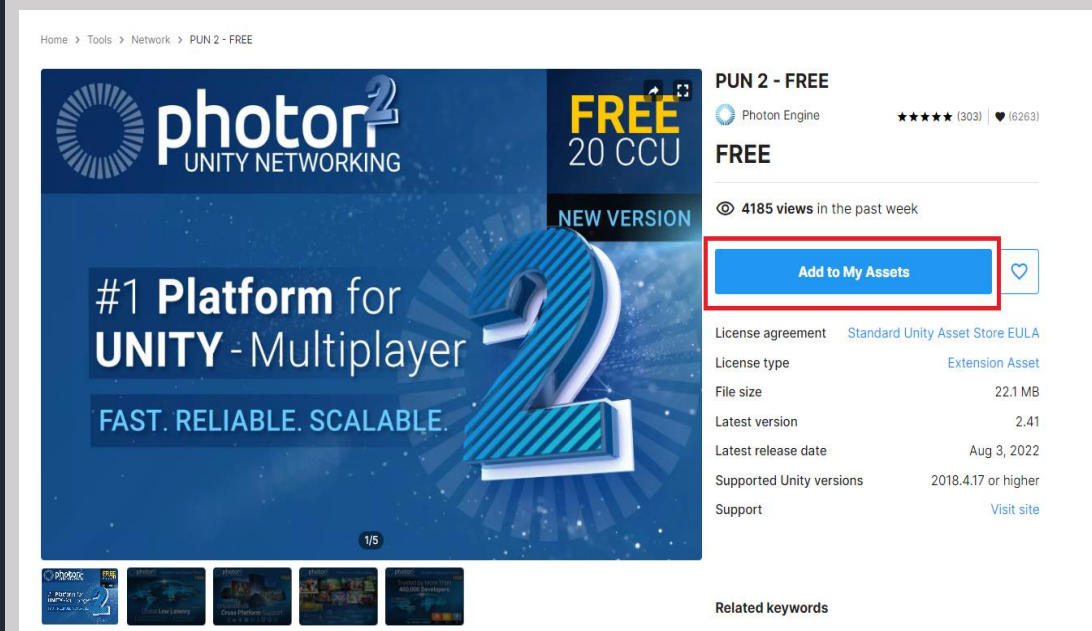
0%

통계 정보
상세 정보
CCU 변경하기

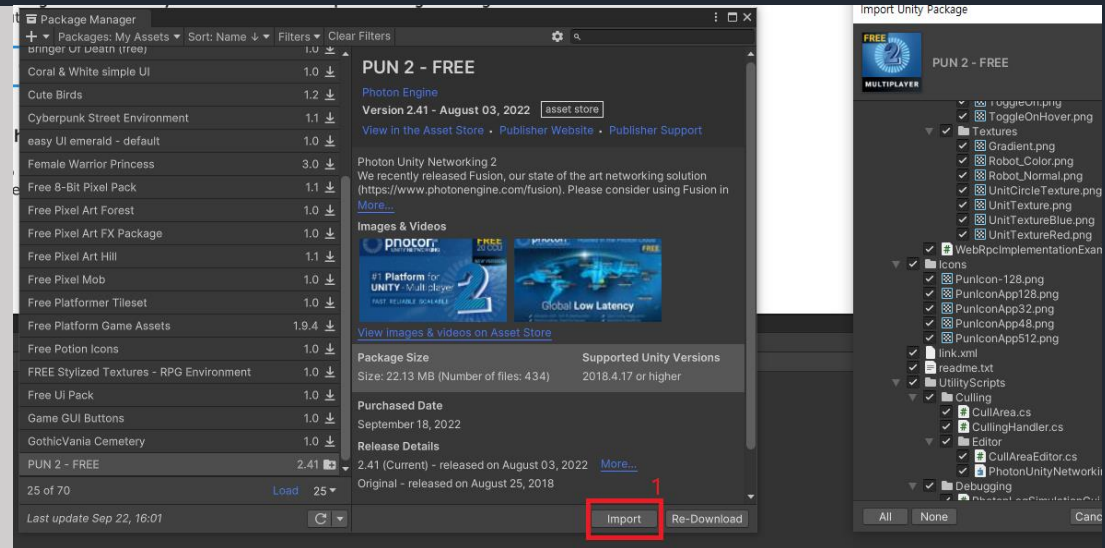
4. 어플리케이션 ID를 복사



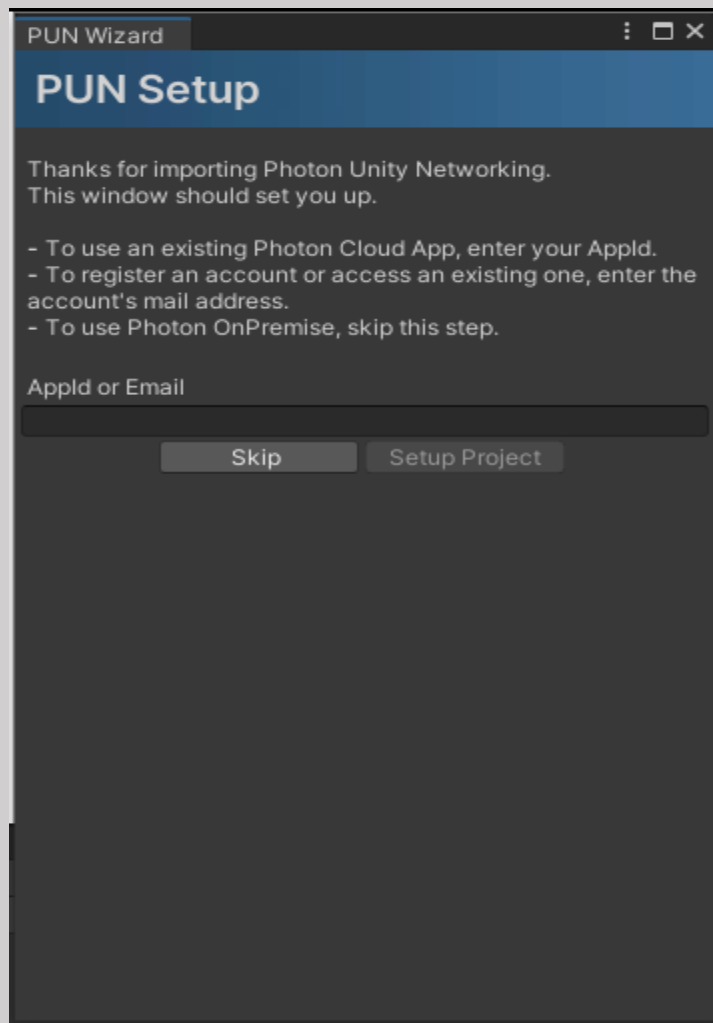
5. <https://assetstore.unity.com/>에서 PUN 2 검색



6. 나의 에셋에 추가



7. 프로젝트 화면에서 해당 에셋을 import



8. AppId or Email 칸에 아까 복사해둔 어플리케이션 ID 입력