



# 포팅매뉴얼

## 개발 정보

### 개발 환경

#### 형상 관리 도구

- Git
- Gitlab
- Plastic SCM

#### 협업 도구

- Jira
- Mattermost
- Notion

#### IDE

- Visual Studio 2022(17.5.4)
- Spring Boot STS(4.17.2)

### 사용 기술 및 버전

#### 서버 공통

Name	Version
Ubuntu	20.04
Docker	23.0.5
Docker-Compose	1.29.2
Jenkins	2.375.3

#### API 서버

Name	Version
Spring Boot	2.7.10
Spring Security	2.7.10
Spring Data JPA	2.7.10
MySQL	8.0.33-debian
Java	11.0.17
Redis	7.0.11

#### Game 서버

Name	Version
Photon Fusion SDK	1.1.6 Stable
Photon Cloud	-

#### Game 클라이언트

Name	Version
Unity Engine	2021.3.23f1
Unity Burst Compiler	1.3.1
C#	4.5.2-3.23171.7
.NET Framework	4.8.04084

### 외부 Unity Assets

Asset Name	License
GUI Parts	<a href="#">Standard Unity Asset Store EULA</a>
Ancient Undead Pack	<a href="#">Standard Unity Asset Store EULA</a>

Asset Name	License
Deadly Dungeons - Mayan Ruins	<a href="#">Standard Unity Asset Store EULA</a>
Realistic Effects Pack v4	<a href="#">Standard Unity Asset Store EULA</a>
100 Special Skills Effects Pack	<a href="#">Standard Unity Asset Store EULA</a>
UIModal	<a href="#">Standard Unity Asset Store EULA</a>
Dynamic Floating Text	<a href="#">Standard Unity Asset Store EULA</a>
MonoBehaviourTree	<a href="#">Standard Unity Asset Store EULA</a>
UltimateCharacterController	<a href="#">Standard Unity Asset Store EULA</a>
100 Special SkillsEffects Pack	<a href="#">Standard Unity Asset Store EULA</a>
Magical Combat VFX 2	<a href="#">Standard Unity Asset Store EULA</a>
Modular Dungeon Catacombs - Mobile	<a href="#">Standard Unity Asset Store EULA</a>
Free Night Sky	<a href="#">Standard Unity Asset Store EULA</a>
The Medieval Civilians & Soldiers	<a href="#">Standard Unity Asset Store EULA</a>
Stylized Rocks with Magic Rune	<a href="#">Standard Unity Asset Store EULA</a>

## Jenkins Plugin

- Git
- Gitlab
- Pipeline: API

## 외부 서비스 정보

---

### 게임 서버 (Photon Server) 정보

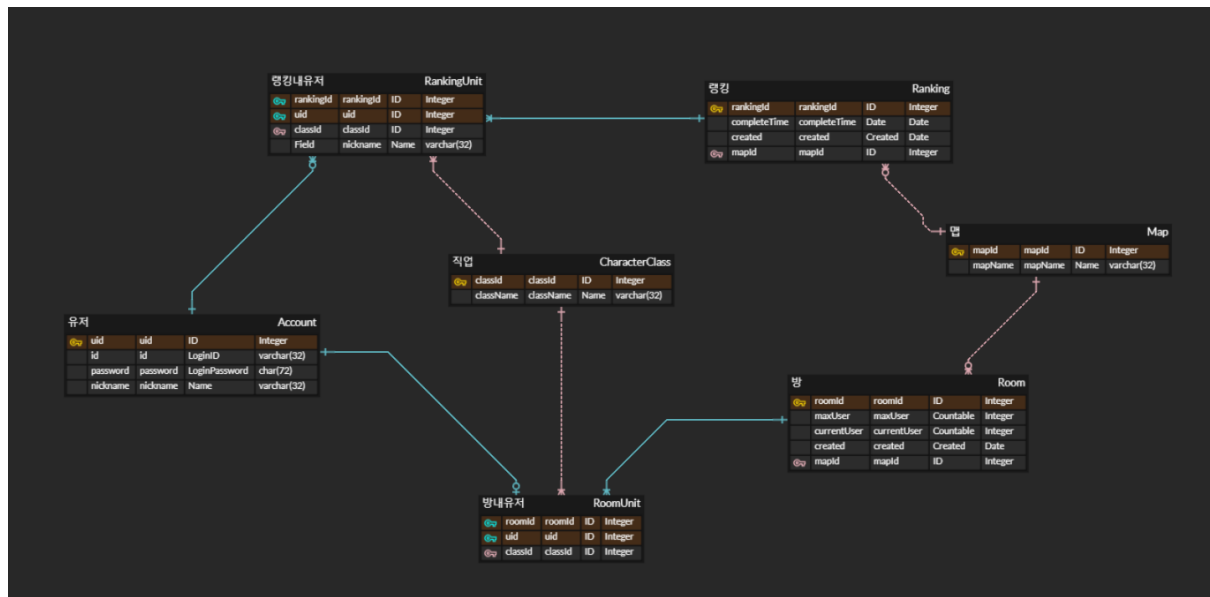
설정이름	설정값
AppID	cb2f42f7-b528-43fd-9c89-f137f30adbac
Regions Allowlist:Allowed Regions	kr;
Plugins	Fusion;
Custom Server Provider	<a href="http://k8a809.p.ssafy.io:8081/login">http://k8a809.p.ssafy.io:8081/login</a>

## 내부 서비스 정보

---

### 데이터베이스

#### Database ERD



## API 정보

### DTO 객체 샘플

#### 1. LoginRequestDTO

```
{"id" : "SampleID", "password": "SamplePassword"}
```

#### 2. ResponseDTO

```
{
  "resultCode": "0", "Message": "Authentication incomplete."
}
{"resultCode": "1", "Message": "Authentication complete.", "AccountData": {"uid": 1, "nickname": "SampleNickname"}}
{"resultCode": "2", "Message": "Authentication failed."}
{"resultCode": "3", "Message": "Invalid parameters."}
```

#### 3. MultiDataResponse

```
{"itemCount": "0", "items": []}
```

#### 4. Ranking

```
{"rank": 1, "rankingId":10001, "mapId":1, "completeTime": 17041, "created":"2023-05-17", "rankingUnit":[{"uid":1, "characterClassId":1}
```

## API Docs

## 1. Account

Name	URI Patterns	Method	RequestType	ResponseType	NeedAuthority
login	/login	POST	Body : ( LoginRequestDTO LoginRequest )	LoginResponseDTO	X
logout	/logout	GET			O
createUser	/account	POST	Body : ( Account account )	ResponseDTO	X
checkIdDuplication	/account/id	GET	Query : ( String id )	ResponseDTO	X

## 2. Ranking

Name	URI Patterns	Method	RequestType	ResponseType	NeedAuthority
getAllRanking	/ranking	GET	Query : ( int mapId, String nickname, int start , int count )	MultiDataResponse<Ranking>	O
postRanking	/ranking	POST	Body : ( Ranking ranking )	Ranking	O

## Redis 설정 정보

redis-compose.yml

```
version: '3.7'
services:
  redis:
    image: redis
    command: redis-server /usr/local/etc/redis/redis.conf
    container_name: redis
    hostname: redis
    labels:
      - "name=redis"
      - "mode=standalone"
    expose:
      - 6379
    ports:
      - "6379:6379"
    volumes:
      - ./redis.conf:/usr/local/etc/redis/redis.conf
```

## 빌드 및 배포 정보 문서

### 사용 포트

#### SSH SFTP 필수 포트

Port	Protocol
22	tcp

## Dedicate 서버

Port	Protocol
4530	udp
4531	udp
4532	udp
5055	udp
5056	udp
5057	udp
27000	udp
27001	udp
27002	udp
9090	udp

## API Server 공통

Port	Protocol
3306	tcp
8080	tcp
6379	tcp

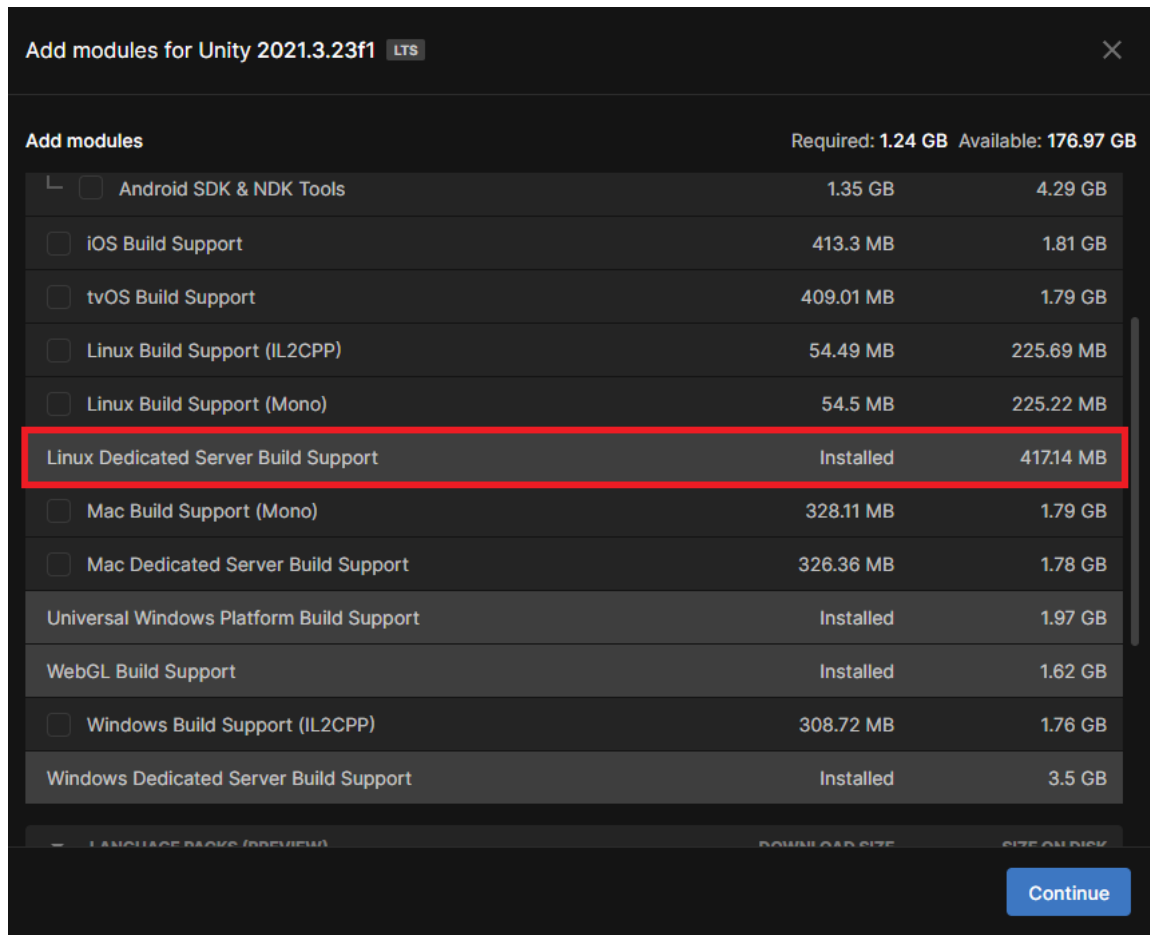
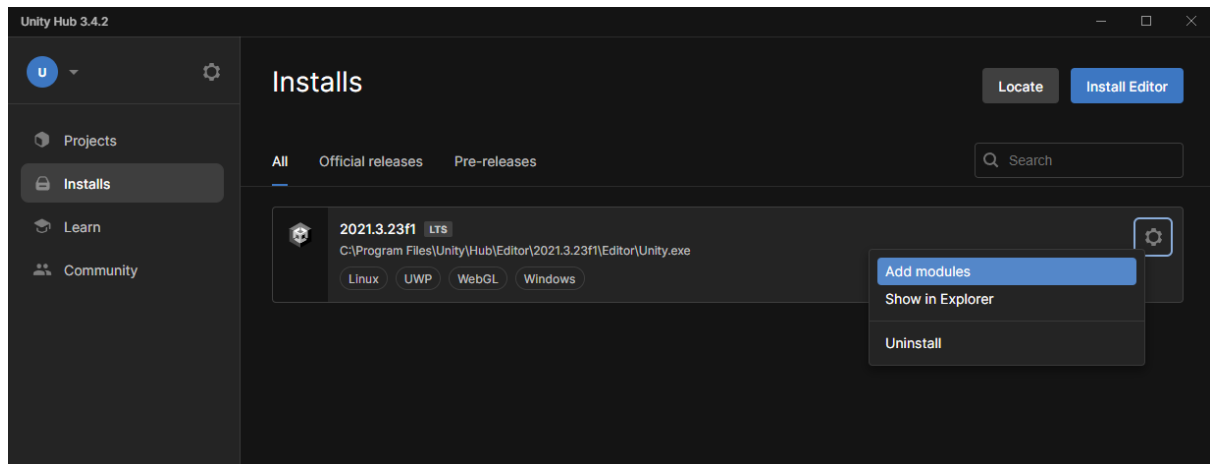
## API 배포 서버 설정

설명	Port	Protocol
Auth-server	8081	tcp
Ranking-server	8082	tcp

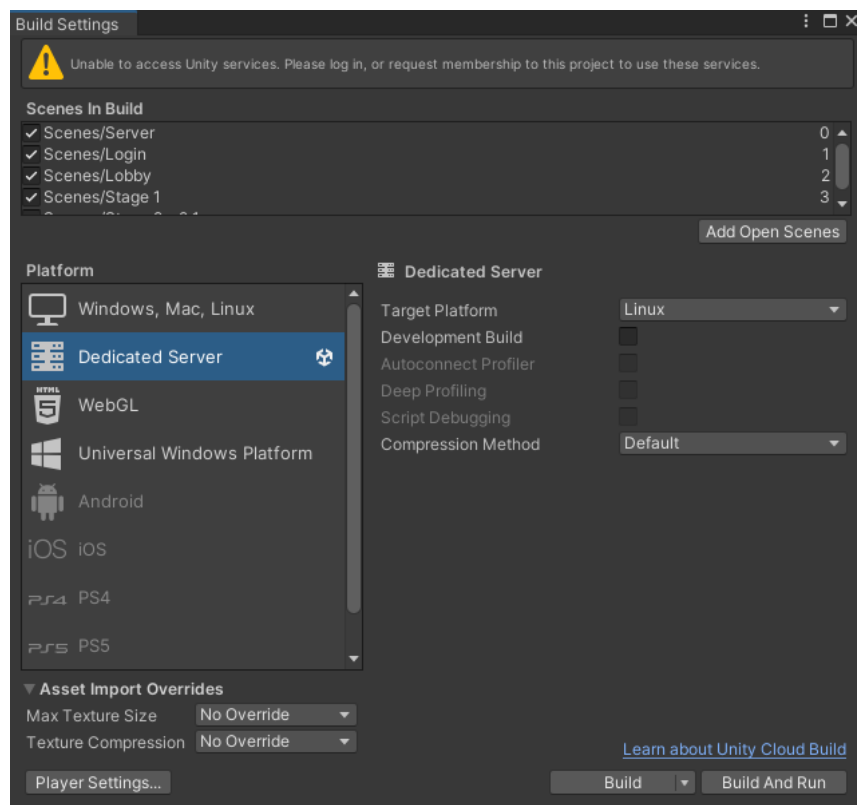
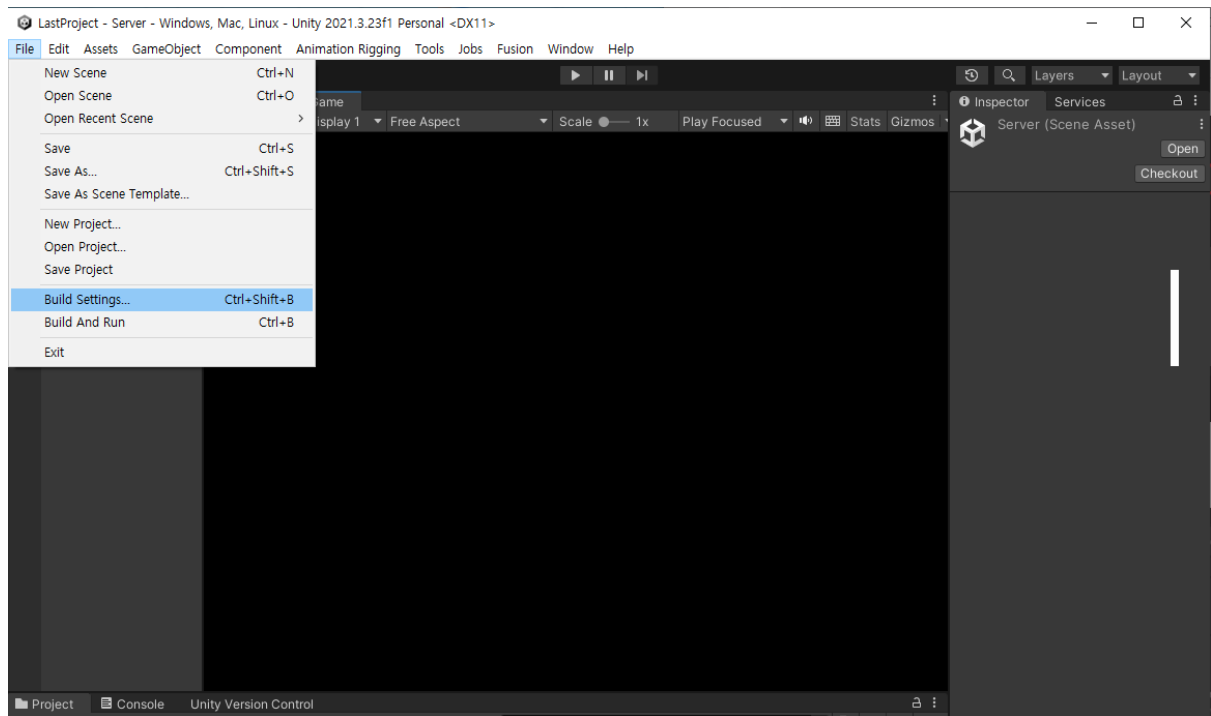
## Dedicate 게임서버 배포

### # 빌드

- Unity Hub > Installs > All에서 설치된 Unity내 Add Modules를 열어  
**Linux Dedicated Server Build Support**가 설치되어 있는지 확인한다.



- Unity Project내에서 Build Settings > Dedicated Server 선택, Linux 설정 뒤 Build를 한다.



## # 배포

- 빌드 산출물을 점검한다.
- UnityPlayer.so, server\_x86\_64, server\_Data, server\_BurstDebugInformation\_DoNotShip
- 빌드 산출물을 전부 서버내 전송한다.
- Dedicate Server 실행 스크립트를 같은 디렉토리에 정의하고, 실행한다.

start-server.sh

```
echo "Starting Fusion Dedicated Server"

while getopts s:r:l:i:p: flag
do
    case "${flag}" in
        s) session="-session ${OPTARG}";;      # custom session name
        r) region="-region ${OPTARG}";;        # custom region
        l) lobby="-lobby ${OPTARG}";;          # custom lobby
        i) publicip="-publicip ${OPTARG}";;     # custom public ip
        p) publicport="-publicport ${OPTARG}";; # custom public port
        esac;
    done

    echo "Connecting to session: $session"

    # Run Server
    cd ~/bin
    chmod +x ./server.x86_64
    ./server.x86_64 -batchmode -nographics $session $region $lobby $publicip $publicport -logFile &

    # Store server execution Exit Code
    status=$?

    if test $status -eq 0
    then
        echo "Server exited normally"
    elif test $status -eq 1
    then
        echo "Server exited by timeout with no players"
    else
        echo "Server exited with code: $status"
    fi

    echo "Done"
```

- Deditace Server는 백그라운드에서 실행되므로 top 명령어를 통해 server.x86\_64의 PID를 확보하고 kill명령어를 통해 종료한다.
- 다음은 서버 종료 예시이다.

```
ubuntu@server:~$ top

top - 07:34:29 up 5 days, 1:29, 1 user, load average: 0.01, 0.04, 0.03
Tasks: 157 total, 1 running, 156 sleeping, 0 stopped, 0 zombie
%Cpu(s): 2.4 us, 0.3 sy, 0.0 ni, 97.1 id, 0.0 wa, 0.0 hi, 0.1 si, 0.2 st
MiB Mem : 15999.0 total, 6291.4 free, 7071.9 used, 2635.7 buff/cache
MiB Swap: 0.0 total, 0.0 free, 0.0 used. 8602.1 avail Mem

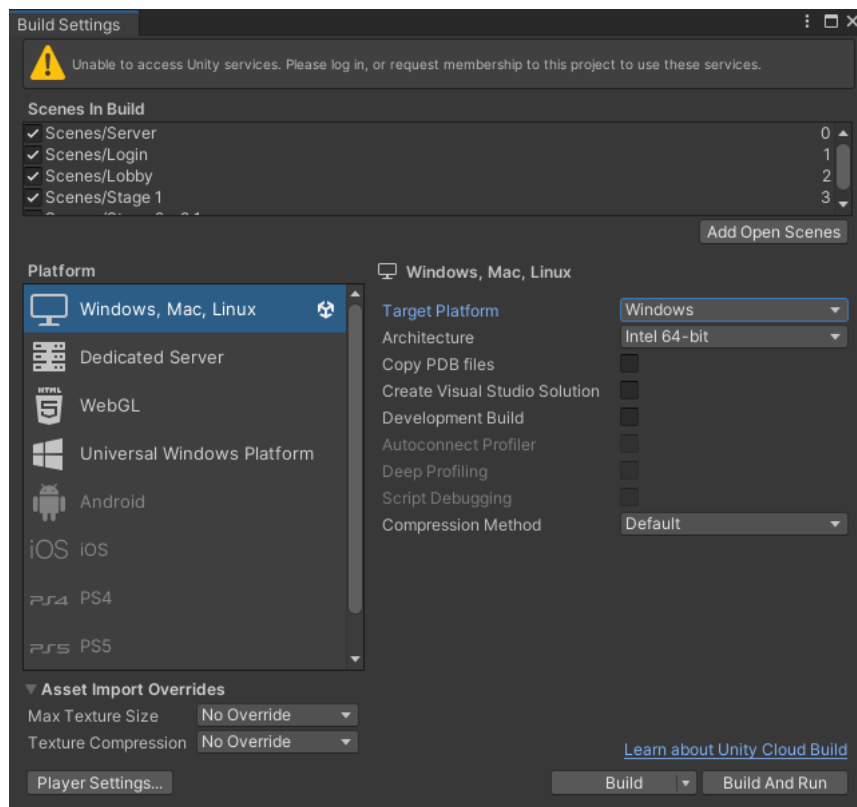
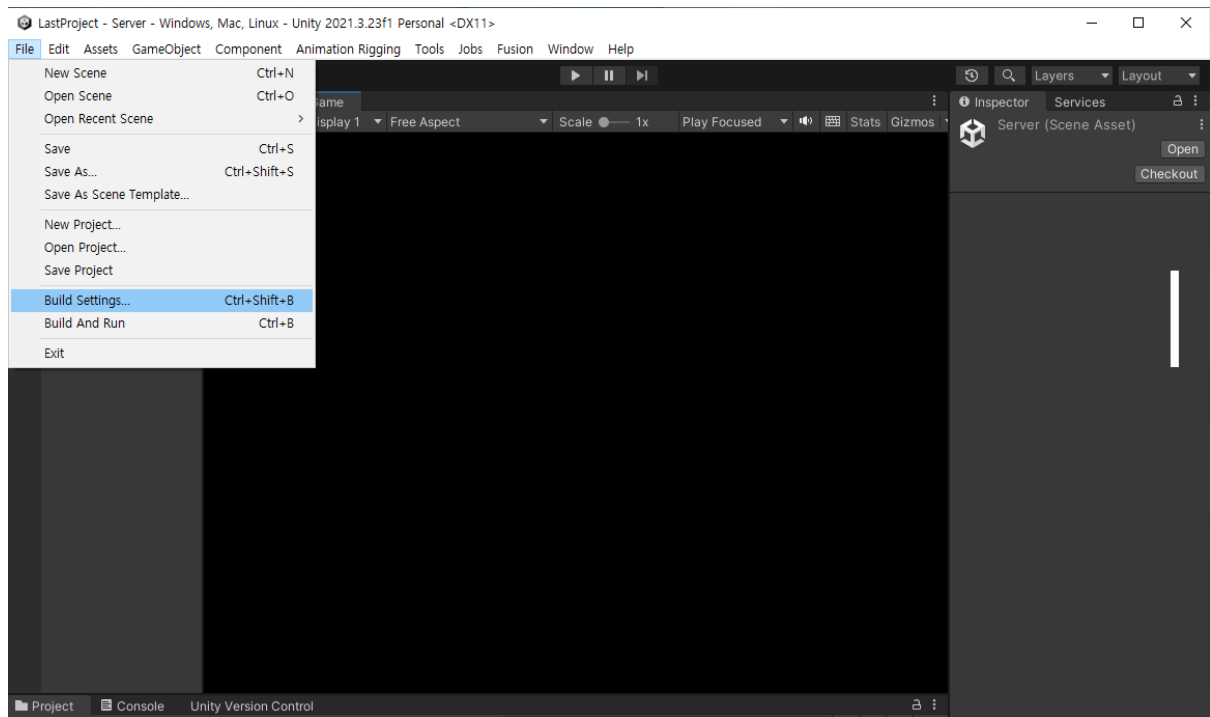
  PID USER      PR  NI   VIRT   RES   SHR S  %CPU  %MEM    TIME+  COMMAND
 57113 root        20   0 7330116 879736 48952 S   10.6   5.4   16:46.33 server.x86_64
 1089 ubuntu      20   0 7953716  4.2g 39284 S    0.7  26.7   30:57.46 java
....

ubuntu@server:~$ kill 57113
```

## Client 빌드 및 배포










- Unity Project내에서 Build Settings > 플랫폼을 Windows, Mac, Linux을 선택한다.



Build를 눌러 빌드한다.

생성된 빌드물을 압축하여 사용자에게 배포하면 된다. 다음은 Windows 배포파일의 예시이다.

	LastProject_BurstDebugInformation_Do...	2023-05-16 오후 1:44	파일 폴더	
	LastProject_Data	2023-05-16 오후 1:44	파일 폴더	
	MonoBleedingEdge	2023-05-16 오후 1:44	파일 폴더	
	LastProject.exe	2023-05-16 오후 1:44	응용 프로그램	639KB
	UnityCrashHandler64.exe	2023-05-16 오후 1:44	응용 프로그램	1,098KB
	UnityPlayer.dll	2023-05-16 오후 1:44	응용 프로그램 확장	48,239KB
	WinPixEventRuntime.dll	2023-05-16 오후 1:44	응용 프로그램 확장	33KB

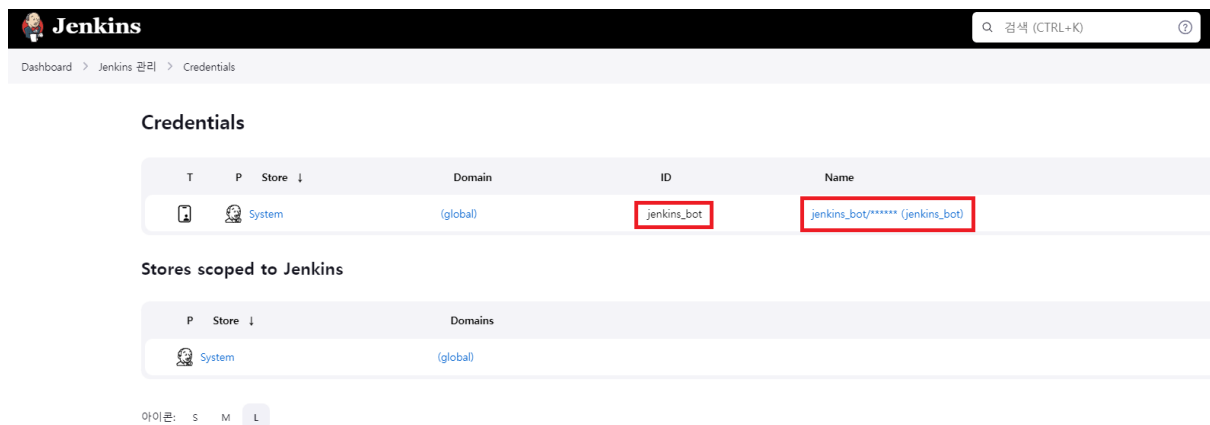
## Jenkins이용 게임 API서버 배포

### # 빌드 및 배포





Jenkins 배포 및 내 필수 플러그인이 설치되어 있다고 가정한다.

먼저 Jenkins내 Gitlab 레포지토리 pull 권한을 획득한 유효한 Credential을 등록한다.




Jenkins Dashboard > Jenkins 관리 > Credentials

Credentials

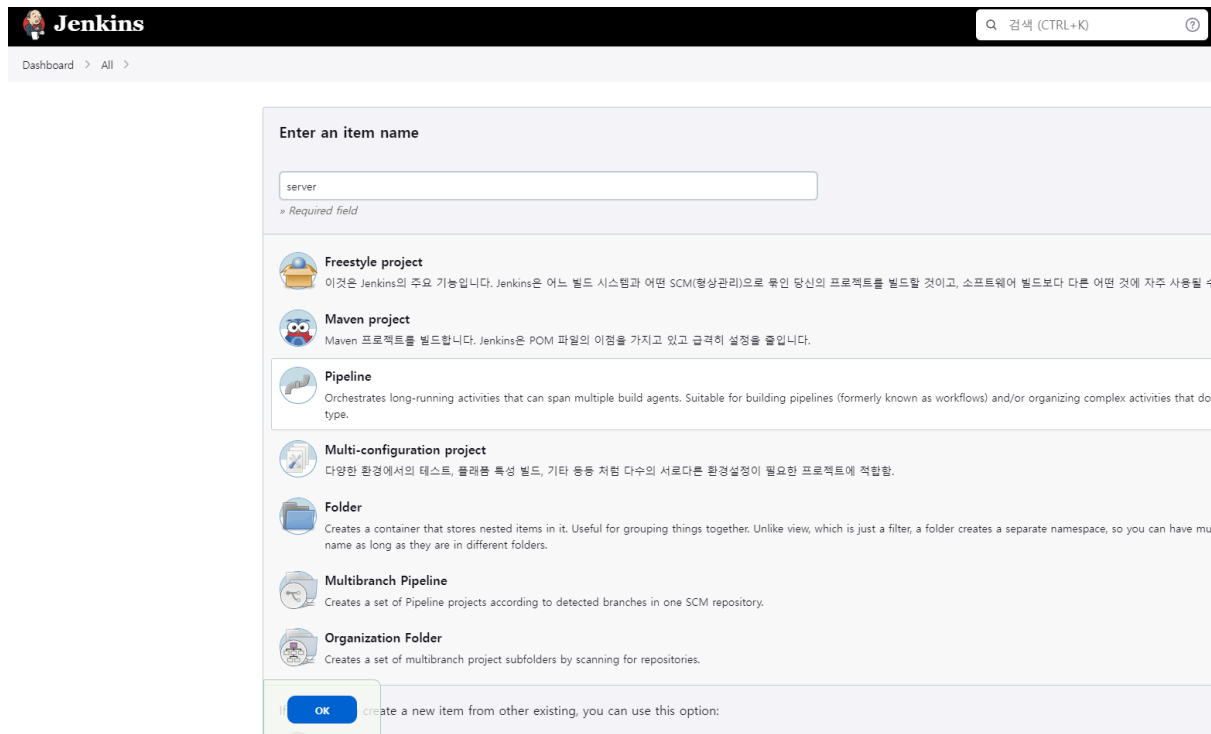
T	P	Store ↓	Domain	ID	Name
		System	(global)	jenkins_bot	jenkins_bot/***** (jenkins_bot)

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

아이콘: S M L

Jenkins Dashboard에서 New Item을 고르고 다음과 같이 Pipeline을 고르고, 이름을 정한뒤 OK를 누른다.



- Auth 및 Ranking서버 두개를 만들어야 한다. 만드는 방법은 같다.

Pipeline Script를 각각 작성한다.

auth-server pipeline code

```
def repository = <gitlab address>
def credentials = <credential name>

def branch = <branch name>;
def branch_decorator = 'prod'

def service = 'auth_server'

def container = "springboot-${service}-${branch_decorator}"
def location = './auth-server'
def port = 8081

pipeline {
    agent any

    stages {
        stage('Git Checkout') {
            steps {
                git branch: "${branch}", credentialsId: "${credentials}", url: "${repository}"
                sh("chmod +x ${location}/gradlew")
            }
        }

        stage('Build') {
            when {
                expression { sh( returnStdout: true, script: "git show --oneline --first-parent -- ${location}").trim().length() > 0 }
            }
            stages {
                stage('Gradle Build') {
                    steps {
                        dir("${location}") {
                            sh "./gradlew init"
                            sh "./gradlew build"
                        }
                    }
                }
            }
        }
    }
}
```



그 뒤, 빌드를 진행하면 된다.

## 배포 특이사항

- auth-server application.yml
- ranking-server application.yml
- redis.conf

만약 jenkins\_home의 위치가 /var/lib/docker/volumes/jenkins\_home/ 이고,  
Jenkins 에서 생성한 Item의 이름이 Auth Server인 경우

포팅매뉴얼

와 같이 복사하여 사용한다.

**sql table파일, dump파일은 exec 폴더 내 존재한다. 그러나, schema에 대한 정의가 빠져 있다. database를 만드려면, 다음과 같이 만들면 된다.**

```
CREATE SCHEMA <DB_NAME> DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4_bin ;
```

**API 서버 호스트 주소에 따라 소스코드 일부를 변경해야 한다.**

Unity Project내 코드중 Assets/Scripts/Network/ApiConfig.cs 파일에서

```
public class ApiConfig
{
    public static string Host { get; set; } = "http://k8a809.p.ssafy.io";
}
```

부분에서 Host값을 포팅 위치에 Api서버로 변경한다.

만약, Auth및 Ranking서버의 위치가 다른 경우.

AccountHttpManager.cs, RankingHttpManager.cs 내 클래스 AccountHttpManager, RankingHttpManager의 필드변수 static string uri를 직접 수정해야 한다.