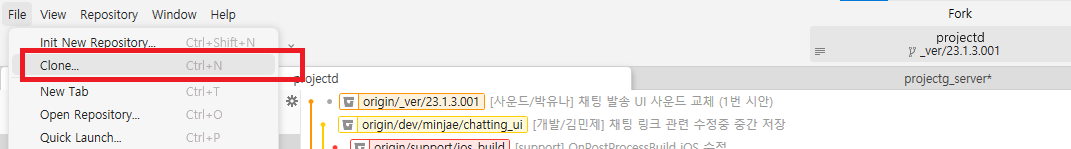
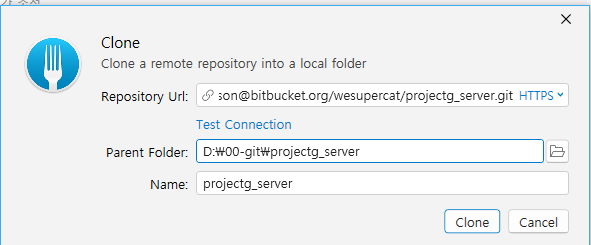
서버 개발하기 위한 환경 구성

# 1. 서버 git clone

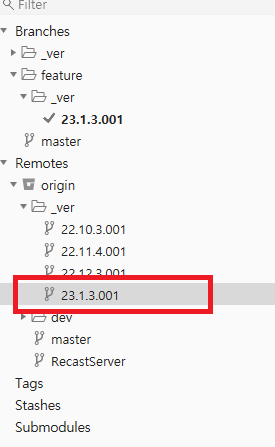
1. Fork 설치 후, 소스 clone



Repository url: <https://bonghoson@bitbucket.org/wesupercat/projectg_server.git>



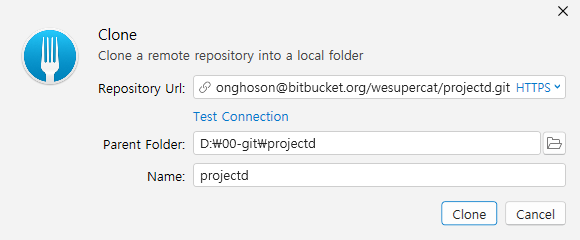
1. master에서 최신 브랜치로 변경하기



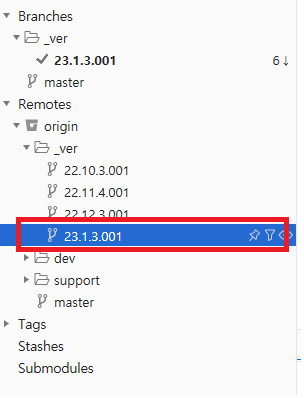
# 2. 클라이언트 git clone

1. git clone 받기

Repository url: <https://bonghoson@bitbucket.org/wesupercat/projectd.git>



1. master에서 최신 브랜치로 변경하기



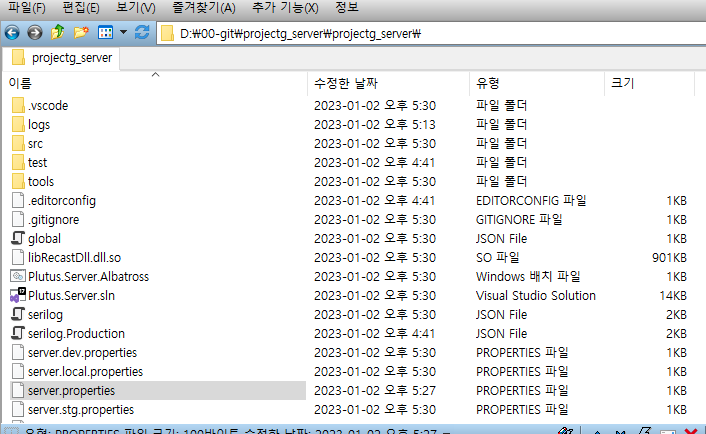
# 3. 서버 구성하기 위한 환경 구성

1. 비쥬얼 스튜디오 2022 또는 Rider, Visual Code IDE 준비
2. 닷넷 7.x SDK 이상 설치

개발 서버 실행

| https://bonghoson@bitbucket.org/wesupercat/projectd.git  https://bonghoson@bitbucket.org/wesupercat/projectg\_server.git  https://bonghoson@bitbucket.org/wesupercat/rnd.git |
| --- |

# 서버 config 수정 - server.properties



| #Local|Development|Stage|Production ENVIRONMENT=Local #환경명을 적어줍니다.. resource.path=D:/00-git/projectd/projectd/ #클라이언트 경로를 적어줍니다. |
| --- |

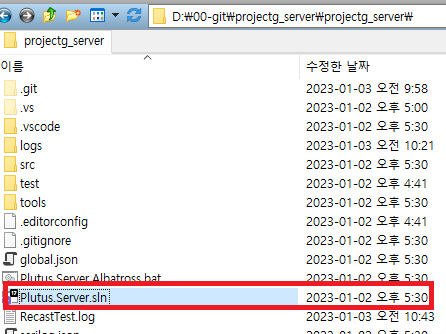


# 방화벽 관련 해결법

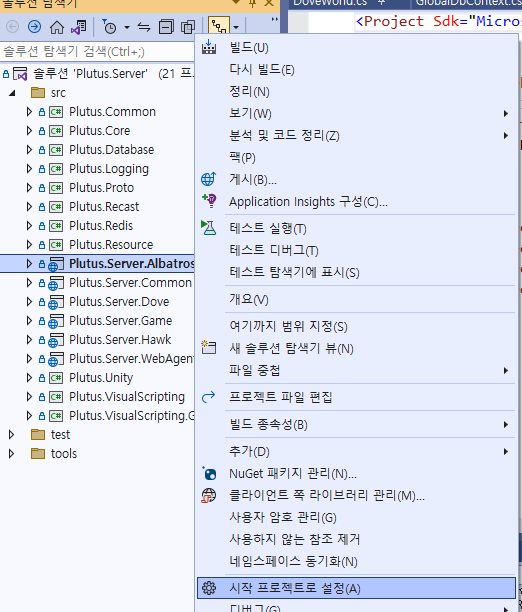
| dotnet dev-certs https --clean 로 기존 인증서 삭제  dotnet dev-certs https --trust 로 새로 설치 |
| --- |

# 프로젝트 로드

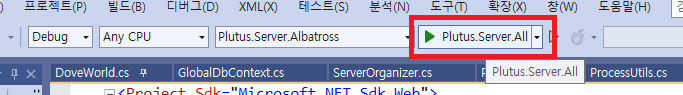
1. 아래 프로젝트 클릭



1. 시작 프로젝트 설정

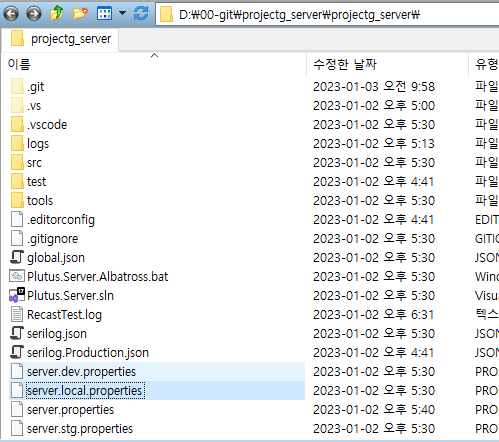


1. 빌드 실행



서버 분석

# DB 연결 주소



| db.host=tigeronline-dev.cluster-c28miexazfvq.ap-northeast-2.rds.amazonaws.com  db.port=3306  db.name=p\_dev\_global  db.user.name=plutus  db.user.password=keiusu22372A  db.connection.pool.size=2 |
| --- |

# redis연결 주소

| public Task<IEnumerable<ServerRedisConfigRow>> GetAllAsync() {  var sql = @"SELECT \* FROM `server\_redis\_configs`";  return \_node.DoAsync(conn => conn.QueryAsync<ServerRedisConfigRow>(sql)); } |
| --- |

| pg-dev.supercat.co.kr |
| --- |

# AlBatross 프로젝트 시작점

* 개발용 프로젝트로 한번에 게임 서버군을 실행하기 위한 프로젝트

| public static int Main(string[] args) {  Protos.Registry();  GameZonedDateTime.Initialize();   Env.Shared.Initialize("serilog.json", "server.properties");   // 리소스 로드  // ...  Logger.Info($"========================================================");  Logger.Info($"====================== START({nameof(ProgramForAlbatross)}) ===========================");  Logger.Info($"========================================================");   Logger.Info($"running - workingPath({Directory.GetCurrentDirectory()})");   var dove = ProgramForDove.Start(args);  var game = ProgramForGameServer.Start(args);   var webAgent = ProgramForWebAgent.Start(args);    var waiter = Env.Shared.Process.CancelOrProcessExitAsync();  waiter.Wait();   webAgent.Stop();  game.Stop();  dove.Stop();   Logger.Info($"========================================================");  Logger.Info($"====================== STOP({nameof(ProgramForAlbatross)}) ===========================");  Logger.Info($"========================================================");    Thread.Sleep(TimeSpan.FromSeconds(3));    return 0; } |
| --- |

# 

## ProgramForDove(커뮤니티 서버) 관련

| public DoveWorld() : base(nameof(DoveWorld)) {  // 서비스 생성  Resource = AddSystem<ResourceSystem>();  Executor = AddSystem<ExecutorSystem>();  Timer = AddSystem<WheelTimerSystem>();  Maintenance = AddSystem<MaintenanceSystem>();  Db = AddSystem<DatabaseSystem>();  Dashboard = AddSystem<ServerDashboardSystem>();  Redis = AddSystem<RedisSystem>();  Alive = AddSystem<ServerAliveSystem>();  Variant = AddSystem<ServerVariantSystem>();   Unique = AddSystem<ServerUniqueSystem>();  Word = AddSystem<WordSystem>();  Shard = AddSystem<ShardSystem>();   Players = AddSystem<DovePlayerSystem>();  PlayerWorker = AddSystem<XWorkerSystem<DovePlayer>>();   Lobby = AddSystem<DoveLobbySystem>();  Zone = AddSystem<DoveZoneSystem>();   GroupMember = AddSystem<DoveGroupMemberSystem>();  Group = AddSystem<DoveGroupSystem>();  GroupWorker = AddSystem<XWorkerSystem<DoveGroup>>();   Sessions = AddSystem<DoveSessionSystem>();  SessionWorker = AddSystem<XWorkerSystem<DoveSession>>();   Server = AddSystem<DoveServerSystem>();  Web = AddSystem<WebSystem>();   Topic = AddSystem<ServerTopicSystem>();   Channel = AddSystem<DoveChannelSystem>();  ChannelWorker = AddSystem<XWorkerSystem<DoveChannel>>();   ZoneFront = AddSystem<ZoneFrontSystem>();  ZoneFrontWorker = AddSystem<XWorkerSystem<ZoneFront>>(); } |
| --- |

| public void Start() {  var runningOs = Env.Shared.Property.RunningOs;  var privatePort = Env.Shared.Property.DovePrivatePort;  var embededWebPort = Env.Shared.Property.DoveEmbeddedWebPort;   Executor.Start();  GroupWorker.Start(TimeSpan.FromMicroseconds(1000 \* 1));  SessionWorker.Start(TimeSpan.FromMilliseconds(1000d / 8));  PlayerWorker.Start(TimeSpan.FromMilliseconds(1000 \* 1));  ChannelWorker.Start(TimeSpan.FromMilliseconds(1000 \* 1));  ZoneFrontWorker.Start(TimeSpan.FromMilliseconds(1000 \* 5));   Server.Start(runningOs, privatePort);  Dashboard.Start();  Alive.Start();  Shard.Start();  Web.Start<StartupWebDove>(this, embededWebPort, embededWebPort);   Topic.Start(DoveTopicSession.RouterOf);  Channel.Start();   ZoneFront.Start(); } |
| --- |

## ProgramForGameServer(게임 서버) 관련

| public GameWorld() : base(nameof(GameWorld)) {  // 서비스 생성  Res = AddSystem<ResourceSystem>();  Executor = AddSystem<ExecutorSystem>();  Timer = AddSystem<WheelTimerSystem>();  Maintenance = AddSystem<MaintenanceSystem>();  Db = AddSystem<DatabaseSystem>();  Dashboard = AddSystem<ServerDashboardSystem>();  Redis = AddSystem<RedisSystem>();   Variant = AddSystem<ServerVariantSystem>();  Unique = AddSystem<ServerUniqueSystem>();  Word = AddSystem<WordSystem>();  Shard = AddSystem<ShardSystem>();  Recast = AddSystem<GameRecastSystem>();   Sessions = AddSystem<GameSessionSystem>();  SessionWorker = AddSystem<XWorkerSystem<GameSession>>();   Server = AddSystem<GameServerSystem>();  Web = AddSystem<WebSystem>();   Topic = AddSystem<ServerTopicSystem>();  Relay = AddSystem<WorldRelaySystem>();  PlayerProfile = AddSystem<PlayerProfileSystem>();   HotTime = AddSystem<GameEventHotTimeSystem>();  OnTime = AddSystem<GameEventOnTimeSystem>();  EventInGame = AddSystem<GameEventInGameSystem>();   Cron = AddSystem<GameCronSystem>();  ServerQueue = AddSystem<ServerQueueSystem>();   Player = AddSystem<GamePlayerSystem>();  PlayerWorker = AddSystem<XWorkerSystem<GamePlayer>>();   GroupProxy = AddSystem<DoveGroupProxySystem>();  GroupProxyWorker = AddSystem<XWorkerSystem<DoveGroupProxy>>();   PromissoryNote = AddSystem<PromissoryNoteSystem>();  Mail = AddSystem<GameMailSystem>();   Actor = AddSystem<GameActorSystem>();   Zone = AddSystem<GameZoneSystem>();  ZoneWorker = AddSystem<XWorkerSystem<GameZone>>();   Boss = AddSystem<GameBossSystem>();   Account = AddSystem<AccountSystem>();  Azit = AddSystem<AzitSystem>();  Lobby = AddSystem<GameLobbySystem>();  Cheat = AddSystem<GameCheatSystem>(); } |
| --- |

| public void Start() {  var runningOS = Env.Shared.Property.RunningOs;  var privatePort = Env.Shared.Property.GamePrivatePort;  var embededWebPort = Env.Shared.Property.GameEmbeddedWebPort;   Executor.Start();  SessionWorker.Start(TimeSpan.FromMilliseconds(1000d / 8));  PlayerWorker.Start(TimeSpan.FromMilliseconds(500));  GroupProxyWorker.Start(TimeSpan.FromMilliseconds(500)); // 0.5초로 셋팅  Server.Start(runningOS, privatePort);  Dashboard.Start();  Shard.Start();  Web.Start<StartupWebGame>(this, embededWebPort, embededWebPort);   // TODO : Test를 위해 100ms로 설정해서 진행함. ( 차후 변경예정 )   ZoneWorker.Start(TimeSpan.FromMilliseconds(100));  Zone.Start();  Topic.Start(GameTopicSession.RouterOf);   HotTime.Start();  OnTime.Start();  EventInGame.Start();   Cron.Start();   // for Test  Boss.StartBossEventSchedule();   ServerQueue.Start();  ServerQueue.StartStampQueue(); } |
| --- |

## ProgramForWebAgent(로드 밸런싱, 대기열 발급, 어느 게임서버에 접속할지) 관련

* 로드 밸런싱, 대기열 발급
* 세션 관리는 하지 않음 (로그인 서버로 부르기가 애매함)

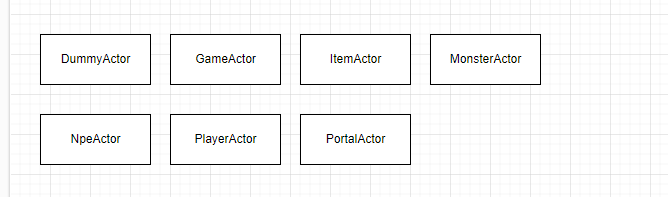
| private WebAgentWorld() : base(worldName: nameof(WebAgentWorld)) {  Res = AddSystem<ResourceSystem>();  Executor = AddSystem<ExecutorSystem>();  Timer = AddSystem<WheelTimerSystem>();  Maintenance = AddSystem<MaintenanceSystem>();  Db = AddSystem<DatabaseSystem>();  Dashboard = AddSystem<ServerDashboardSystem>();  Redis = AddSystem<RedisSystem>();  Shard = AddSystem<ShardSystem>();  Variant = AddSystem<ServerVariantSystem>();  Word = AddSystem<WordSystem>();  Unique = AddSystem<ServerUniqueSystem>();  Alive = AddSystem<ServerAliveSystem>();   Account = AddSystem<AccountSystem>();  PlayerProfile = AddSystem<PlayerProfileSystem>();  ServerQueue = AddSystem<ServerQueueSystem>();   Web = AddSystem<WebSystem>();  WebWorld = AddSystem<WebWorldSystem>();  WebLobby = AddSystem<WebLobbySystem>();  WebPlayer = AddSystem<WebPlayerSystem>();  WebCoupon = AddSystem<WebCouponSystem>();   HubSession = AddSystem<HubSessionSystem>();  HubSessionWorker = AddSystem<XWorkerSystem<HubSession>>();  HubPlayer = AddSystem<HubPlayerSystem>();  HubChatChannel = AddSystem<HubChatChannelSystem>();  HubLobby = AddSystem<HubChatLobbySystem>(); } |
| --- |

| public void Start() {  var privatePort = Env.Shared.Property.WebAgentPrivatePort;  var publicPort = Env.Shared.Property.WebAgentPublicPort;   Executor.Start();  HubSessionWorker.Start(TimeSpan.FromMilliseconds(1000 / 8));   Dashboard.Start();  Shard.Start();  Alive.Start();  //Web.Start(this, privatePort, publicPort);  Web.Start<StartupWebAgent>(this, privatePort, publicPort);  ServerQueue.Start(); } |
| --- |

## ProgramForHawk (운영툴)

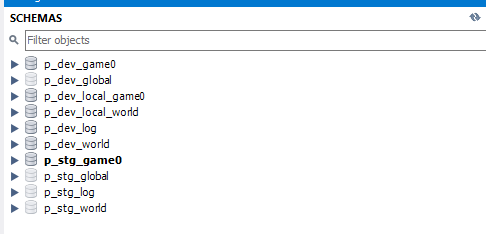
* 운영툴 개발을 위한 프로젝트

게임 서버내 actor 들



질문 사항

1. 프로젝트 g의 개발 일정은 어떻게 되나요?
2. 현재 프로젝트 g의 서버 구조는 어떻게 되어있나요?
3. AlBatross 에 있는 ProgramForDove(커뮤니티 서버), ProgramForGameServer, ProgramForWebAgent 의 간단한 역할은 무엇인가요?
   1. webAgent의 역할은?
4. 클라이언트 접속 방법
   1. 유니티 라이센스가 필요
5. db 구조에 대해 어떻게 되나요?



1. redis 구조는 어떻게 되나요?

