**Setting Up Dedicated Servers**

<https://docs.unrealengine.com/4.26/en-US/InteractiveExperiences/Networking/HowTo/DedicatedServers/>

## **How to set up and package a dedicated server for your project.**

The **server-client** model used by **Unreal Engine** represents network multiplayer games with one server acting as the **host** of the game, and players joining that game as **clients**. The true game state is moderated by the server, while each player controls their pawns remotely with an **autonomous proxy**. The server then **replicates** changes out to each connected client so that all players experience a very close approximation of the game being played on the server.

Where a **listen server** represents a player hosting a game on their machine, a **dedicated server** is a server that runs headlessly. A headless server does not render any visuals, and there is nobody playing on it locally. This enables a dedicated server to focus on gameplay logic and moderating incoming information from clients, making the most of its resources for hosting a game. Additionally, this ensures a level playing field between all players participating in a multiplayer game. While a listen server is often acceptable for casual multiplayer and cooperative play, dedicated servers are ideal for large-scale or competitive games.

This How-To will show you how to build and package a dedicated server for your own multiplayer games.

## **Required Setup**

To follow the steps in this how-to, your project needs to satisfy the following requirements:

You must be using a source build of Unreal Engine, which you can download from the Epic Games GitHub.

If your project is using a binary build from the Epic Games Launcher, you will need to migrate it to a Github source build.

You must have a C++ project that can support server-client multiplayer gameplay.

If you are using a Blueprint project, you will need to convert it to a C++ project before you can proceed.

Configure Windows Defender Firewall Inbound and Outbound Rules

By default, UnrealEngine generates buildable projects for all available platforms that the engine detects SDKs for so that you will be able to build and debug console and mobile platforms as well. If you want to generate projects only for the current platform you are running on (for example, Windows.), run the:

**GenerateProjectFiles.bat** **-CurrentPlatform**

The project files will be a little lighter.

Since a source build of the engine is required for building a **Server** based version of any project, download the project files from GitHub, run Setup.bat, and GenerateProjectFiles.bat **but do not build the engine**. The engine can be built at the same time the project is built.

|  |
| --- |
| PS E:\Epic> **git clone** [**https://github.com/EpicGames/UnrealEngine**](https://github.com/EpicGames/UnrealEngine) **4.27.1**Cloning into '4.27.1'...remote: Enumerating objects: 3998524, done.remote: Counting objects: 100% (2964/2964), done.remote: Compressing objects: 100% (886/886), done.remote: Total 3998524 (delta 2062), reused 2954 (delta 2056), pack-reused 3995560Receiving objects: 100% (3998524/3998524), 11.57 GiB | 39.76 MiB/s, done.Resolving deltas: 100% (2604395/2604395), done.Updating files: 100% (116638/116638), done.PS E:\Epic> cd 4.27.1PS E:\Epic\4.27.1> git statusOn branch releaseYour branch is up to date with 'origin/release'.nothing to commit, working tree cleanPS E:\Epic\4.27.1> git remote -vorigin https://github.com/EpicGames/UnrealEngine (fetch)origin https://github.com/EpicGames/UnrealEngine (push)PS E:\Epic\4.27.1> **./Setup.bat**Checking dependencies...Updating dependencies: 100% (64310/64310), 12156.1/12156.1 MiB | 18.87 MiB/s, done.Registering git hooks...Installing prerequisites...PS E:\Epic\4.27.1> **./GenerateProjectFiles.bat**Setting up Unreal Engine 4 project files...Binding IntelliSense data... 100%Writing project files... 100% |

Open the Visual Studio Solution, make sure that UE4 is set as the Startup Project, press F5 to build and run the UE4Editor.

It is worth noting that the cloned repository offers more Visual Studio Solution target environments than what the Epic Games Launcher does. I use the Epic Games Launcher for quick tests but prefer building with the cloned version of the source.

|  |  |
| --- | --- |
| Source Build Configurations | Epic Games Launcher Build Configurations |
| Graphical user interface, text, application  Description automatically generated | Graphical user interface, text, application, chat or text message  Description automatically generated |

Using the source is the best approach as it allows you to debug in finer detail if needed. The full engine build consists of 4,568 source files and libraries. We can shorten this list by focusing on the environment we are running in by providing the following command line parameter: ./**GenerateProjectFiles.bat** **-CurrentPlatform**

Create a game project from the UE Project Templates. If you already have an existing game you need to Switch Unreal Engine Version and select the Source Build:

|  |  |
| --- | --- |
| Graphical user interface, application  Description automatically generated | Graphical user interface, application  Description automatically generated |

Copy the [Project].Target.cs file, rename it [Project]Server.Target.cs, add it to the Visual Studio Solution, and make the following changes:

|  |  |
| --- | --- |
| Test.Target.cs | TestServer.Target.cs |
| // Copyright Epic Games, Inc. All Rights Reserved.using UnrealBuildTool;using System.Collections.Generic;public class TestTarget : TargetRules{ public TestTarget(TargetInfo Target) : base(Target) { Type = TargetType.Game; DefaultBuildSettings = BuildSettingsVersion.V2; ExtraModuleNames.Add("Test"); }} | // Copyright Epic Games, Inc. All Rights Reserved.using UnrealBuildTool;using System.Collections.Generic;public class TestServerTarget : TargetRules{ public TestServerTarget(TargetInfo Target) : base(Target) { Type = TargetType.Server; DefaultBuildSettings = BuildSettingsVersion.V2; ExtraModuleNames.Add("Test"); }} |

Exit the UE4Editor and rebuild the Visual Studio solution. The solution has not yet been configured for the new Server build.



Once the server target is included in the solution you can build the game or server versions from within the UE4Editor or with Visual Studio.



Make to set the correct Maps & Modes for the Game Default Map and the Server Default Map:



Package both the game and the server. Once the server is completed create a shortcut for it and add -log to the end otherwise it will be a silent server.

|  |  |
| --- | --- |
| Graphical user interface, text, application  Description automatically generated | Graphical user interface, text, application  Description automatically generated |

Note: The solution will build the engine and your project at the same time for the different configurations: **Development Editor**, and **Development Server**. Subsequent builds of the engine should be nil unless something in the engine changes.

Create a shortcut for the Server and add “-log” to give you some visibility

Launch an instance of the game - Open the console and change the resolution: ~ r.SetRes 1080x800w

Connect to the server: ~ Open 127.0.0.1

Repeat the process for another game window.

To recap:

Clone the UnrealEngine from GitHub

PS E:\Epic> git clone <https://github.com/EpicGames/UnrealEngine> 4.27.1

Cloning into '4.27.1'...

remote: Enumerating objects: 3998524, done.

remote: Counting objects: 100% (2964/2964), done.

remote: Compressing objects: 100% (886/886), done.

remote: Total 3998524 (delta 2062), reused 2954 (delta 2056), pack-reused 3995560

Receiving objects: 100% (3998524/3998524), 11.57 GiB | 39.76 MiB/s, done.

Resolving deltas: 100% (2604395/2604395), done.

Updating files: 100% (116638/116638), done.

Run Setup.bat

PS E:\Epic\4.27.1> ./Setup.bat

Checking dependencies...

Updating dependencies: 100% (64310/64310), 12156.1/12156.1 MiB | 18.87 MiB/s, done.

Registering git hooks...

Installing prerequisites...

Run GenerateProjectFiles.bat

PS E:\Epic\4.27.1> ./GenerateProjectFiles.bat

Setting up Unreal Engine 4 project files...

Binding IntelliSense data... 100%

Writing project files... 100%

Build the solution

Add the Server.Target.cs

Build the Game

Build the Server

Create a shortcut for the Server and add “-log” to give you some visibility

Launch an instance of the game - Open the console and change the resolution: ~ r.SetRes 1080x800w

Connect to the server: ~ Open 127.0.0.1

Repeat the process for another gamer