Erupe Monster Hunter Frontier Experimental Server Emulator Setup Guide (1/30/2021)

New quest information available here

The erupe server is still heavily in development and you should expect numerous bugs, crashes, and other unintended behavior during use.

Comments are welcome for edit requests, but please include why in your comment.

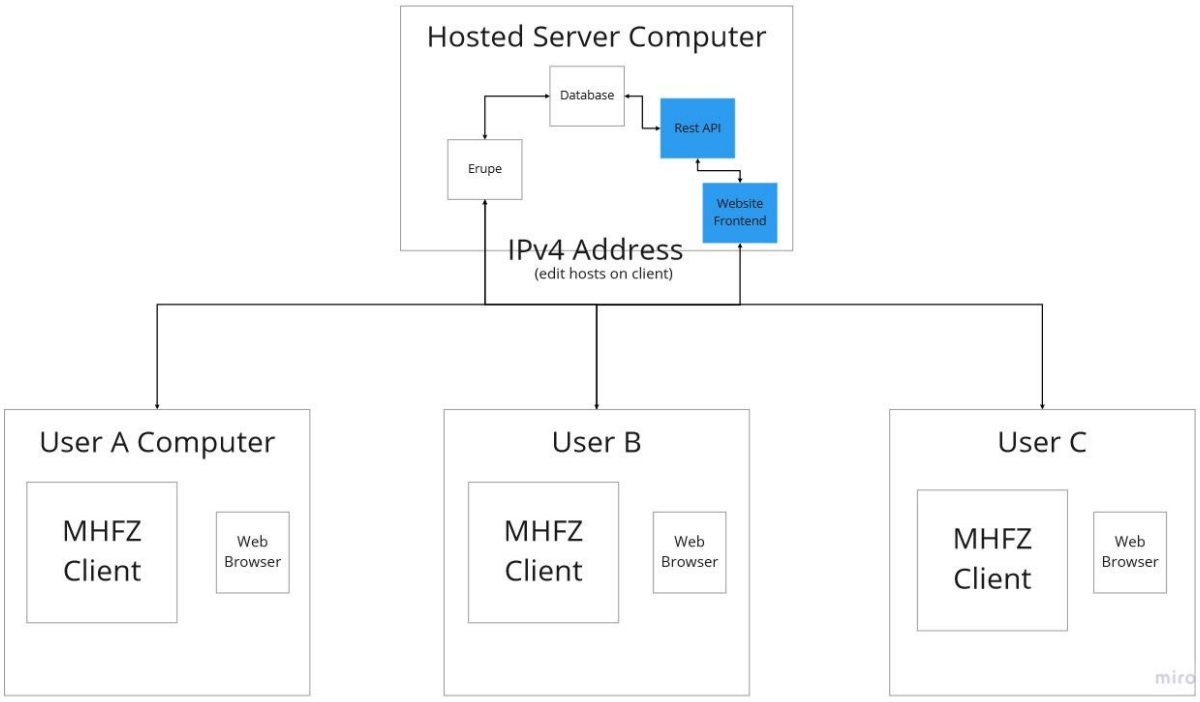
About

This guide aims to be a bit more complete list of steps taken in order to set up the Erupe MHF private server emulator. This tutorial is largely aimed at less technically inclined users, so in any situation where I can avoid extra tooling I will. Expect a dockerized variant (if you don't know what this is just forget it) at some point in the near future. These instructions are specifically targeting **Windows** users so your steps will vary if you're installing under a linux environment. I have linked all relevant programs/tools when they are first mentioned so that you can discover any differences in your installation process if you aren't working in a Windows environment.

Application Architecture (Optional Information to better understand what this actually does):

Rough Network Topology:

- Blue squares represent in-progress work by me
    - Frontend (Likely React) for server management
    - Rest API for db comm (will probably re-route account creation in client to this eventual utility)



The Erupe server emulator is made up of two services, the 4 service servers (used for client-server communication) and the PostgreSQL database engine (for persisting/retrieving information from/for the http servers).

Things that will help you contribute to development efforts

Programming Language:

- Golang
  - Key Go Packages:

gorilla/mux - http server used for request/response communication
net - various network uses throughout the project
sqlx - extension of the database/sql (this is NOT an ORM)
zap - logging

Database Technology:

PostgreSQL - Open-source object-relational database

Key concepts:

Client-server model

Server Services:

The application that sends and receives information to/from clients and reads/writes to the database to persist information

- It is a collection of 4 Golang services that facilitate client<->server communication
  - **launcher_server**: gorilla/mux http request router and dispatcher. Used to deliver
    - Routes
      - setupServerlistRoutes
        - mhf-n.capcom.com.tw
          - /server/unique.php
            - TODO: Supposed to make a check against the db for character name uniqueness in func serverUniqueName
          - /server/serverlist.xml
            - Uses serverHandlerFunc passing in the server instance and func serverList in order to return an xml representation of the server
        - srv-mhf.capcom-networks.jp
          - /serverlist.xml
            - Uses serverHandlerFunc passing in the server instance and func serverList in order to return an xml representation of the server

```
<?xml version="1.0"?>
<server_groups>
        <group idx='0' name='Erupe' ip='%s' port="%d"/>
</server_groups>
```

Clairty Question: Is there a typo in the todo where we should actually be checking for unique combinations among server/port [referencing hostname] or is this actually ensuring a character is unique among a server?

- setupOriginalLauncerRoutes : Only used when config.json[“launcher”][“UseOriginalLauncherFiles”] = true
    - mhfg.capcom.com.tw
        - /
            - Serves the original TW HTML launcher page to the client launcher (requires configuring original launcher files in the erupe/www/tw/)
    - cog-members.mhf-z.jp
        - /
            - Serves the original JP HTML launcher page to the client launcher (requires configuring original launcher files in erupe/www/jp/)
    - www.capcom-onlinegames.jp
        - /auth/
            - /
                - Serves the original JP HTML auth page to the client launcher (requires configuring the original launcher files in erupe/www/jp)
            - launcher/login
                - Defers to func jpLogin which pulls the submitted FormValue[“pw”] and creates a json:

                  {“result”: “Ok”,
                  “skey”: FormValue[“pw”],
                  “Code”: “000”,
                  “msg”: “”}

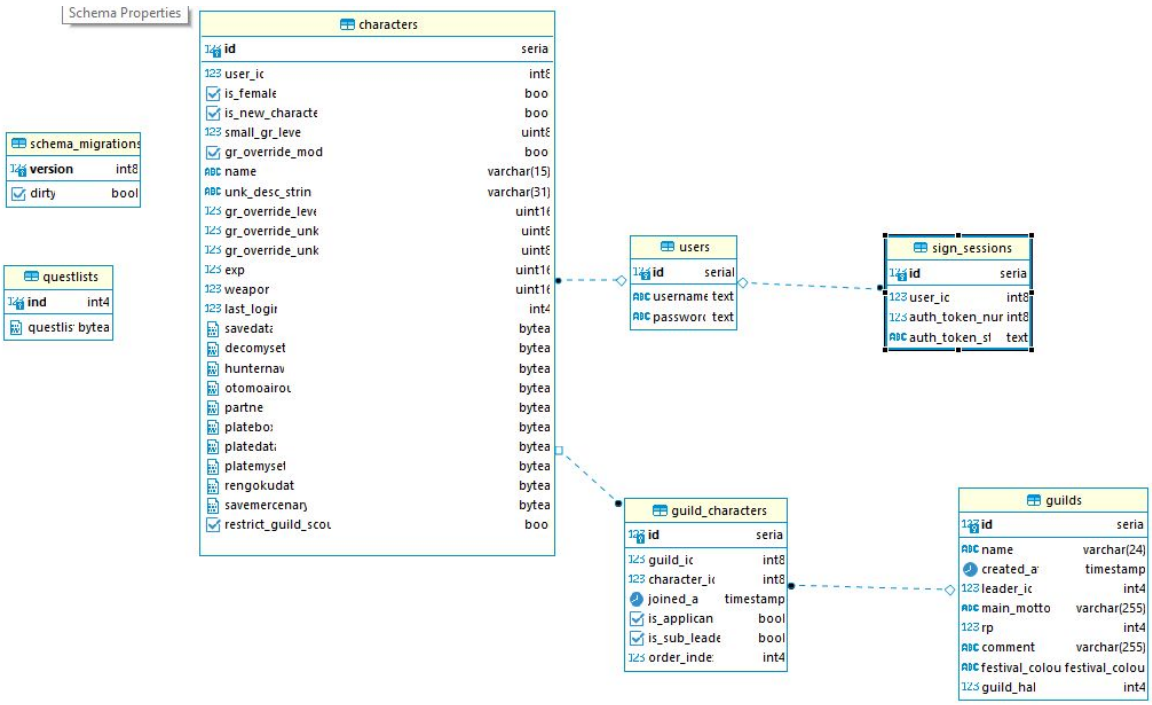
                - jpLogin attaches this json to the <input id=”result” value=THIS> which then makes a post request on cog-members.mhf-z.jp (which is a route on the sign server)
- setupCustomLauncherRoutes
    - mhfg.capcom.com.tw
        - /g6_launcher/
            - Serves the custom TW HTML auth page to the client launcher (file served from www/erupe/
    - cog-members.mhf-z.jp

        - /launcher
            - Serves the custom JP HTML auth page to the client launcher (file served from www/erupe/

    - **entrance_server**
    - **sign_server**
    - **Channel_server**

The database that persists long-term data to disk

- A PostgreSQL instance with a database named erupe
- ERD:



Server Side Installation Steps

Step 1: Download the Server Files (The server)

- Download the code repository (repo) using the dropdown within the green “Code” button and choose: Download ZIP
- Extract the contents of the folder into the directory of you choice (I would recommend avoiding Program Files due to possible permission errors)

Step 2: Download and Install PostgreSQL (The database)

- Download the PostgreSQL installer for your operating system and choose the latest PostgreSQL version
- Run the installer
    - Ensure these components are checked to install
        - PostgreSQL Server (the actual database technology)
        - pgAdmin 4 (a GUI management tool for our PostgreSQL server)
        - Command Line Tools (a command line tool for managing PostgreSQL servers) - not used in this tutorial but you’ll want it for some management stuff in the future
    - Select the Data Directory of your choice (the default pre-filled option should be fine)
    - Create a password for the default database user (postgres)
    - Select a port for the database server to listen to (the default 5432 should be fine)
    - Select the Locale (the default should be fine)
    - Finish the installation

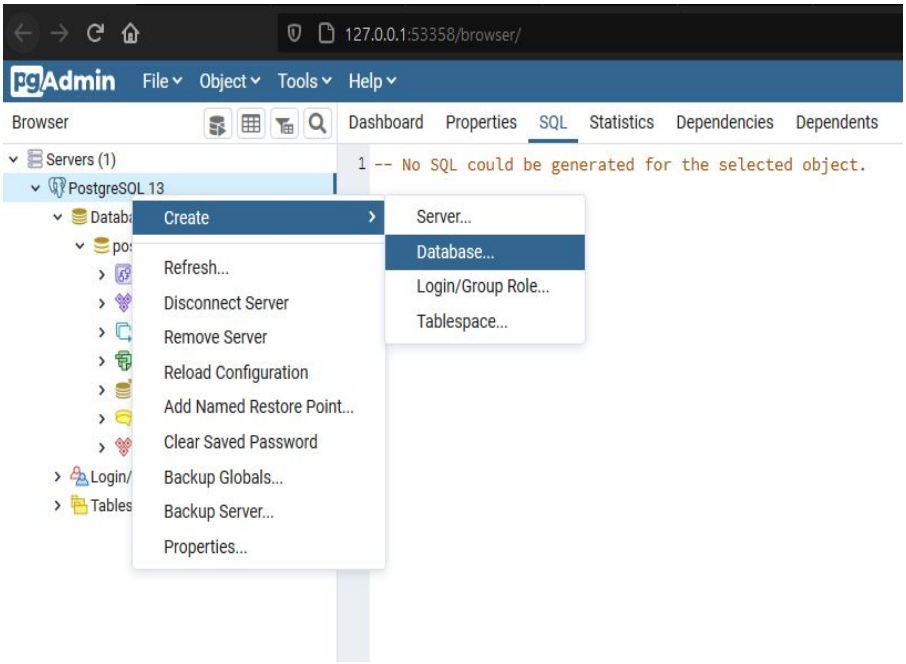
Step 3: Download and Install Golang (The programming language the server is written in)

- Download the Golang installer for your operating system
- Run the installer using the default options

Step 4: Prepare the Database and Download and Install Migration Tools (These facilitate setting up the tables to store information in the database)

Phase 1: Create the database erupe on our PostgreSQL instance

- Open pgAdmin 4 and connect using the credentials you supplied in the Download and Install PostgreSQL steps (the username is probably postgres if you forgot)
    - Right click on your PostgreSQL server and select Create -> Database (photo below for assistance)
    - Fill in the Database field with the value: erupe
        - Click save
    - You should now see a second database called erupe created in the browser on the left



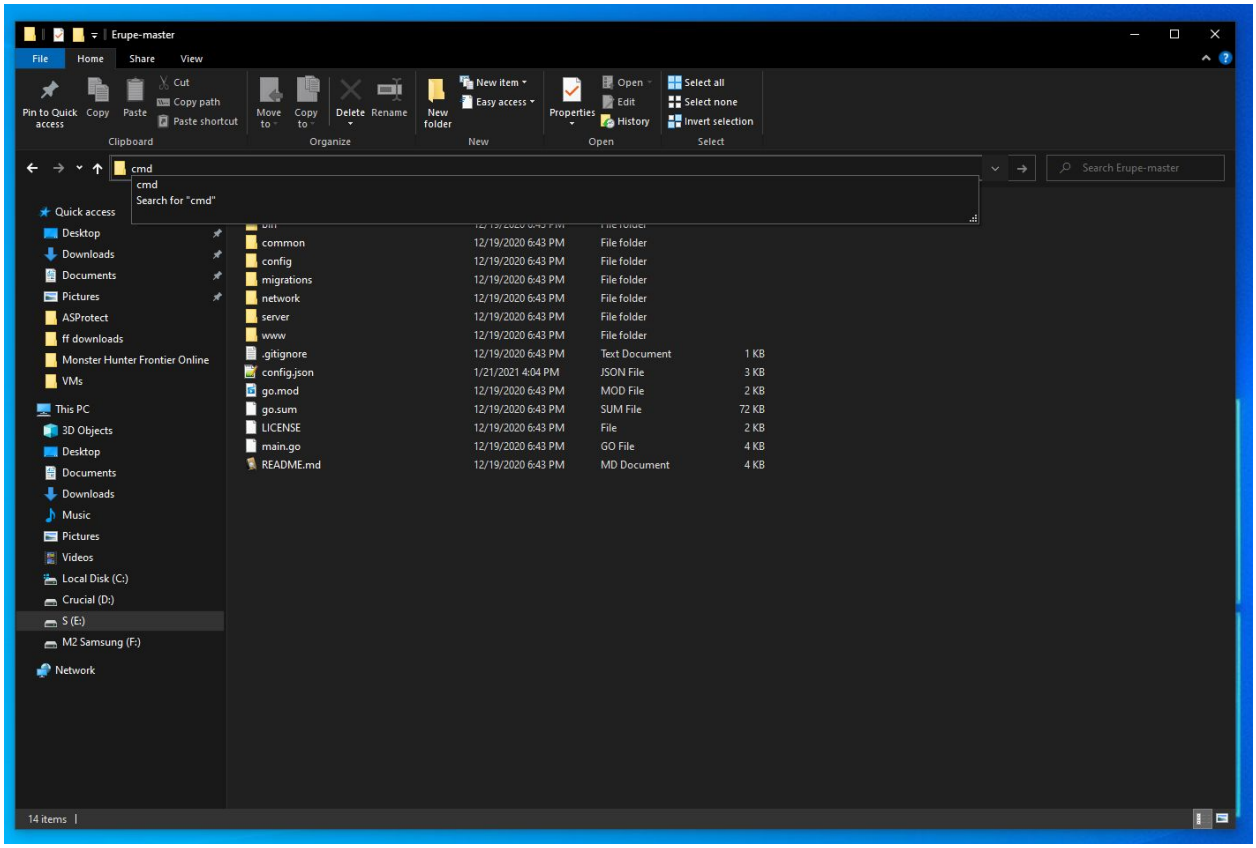
Phase 2: Get and install the programs and tools to install golang-migrate

- Download the latest version of Powershell from its Github releases page
    - Most users will want the file under assets named something like: PowerShell-7.0.4-win-x64.msi (the version may change depending on when you access this page)
    - Run the installer
- Open Powershell and use it to install Scoop by typing the command:
    - iwr -useb get.scoop.sh | iex
- Install golang-migrate by opening a new Powershell window and typing the command:
    - scoop install migrate

Phase 3: Run the migrations in the erupe server files to create the database tables

- Navigate to the root directory of your erupe server files (image for context below)
- In the file browser path, type in cmd and then press enter to launch a cmd prompt at this location (image below shows the cmd prompt being launched from the correct directory)

- Run the command:
    - migrate -database postgres://postgres:password@localhost:5432/erupe?sslmode=disable -path migrations up
- **NOTE:** Replace the blue password with the password you set up during the database installation step. Also if you changed the default port from 5432 during the database installation step, replace the red highlighted text with whatever port you chose



Step 5: Edit the config.json

- Open the config.json file with the text editor of your choice (notepad or notepad++)
    - Under the “database” section, find the port, username, and password fields and change their values to be whatever you chose during the database installation steps

Note: You may not need to change the port value if you kept the default 5432 port. Also the postgres default user is named postgres

- **PLEASE READ:** As I am unable to currently get the game client working I am unable to provide instructions on how to edit the host_ip and ip fields in the config.json
  - I believe you simply need to just replace the value 127.0.0.1 (otherwise known as localhost) with your external ipv4 address which can be found here

Step 5.5: Forward the Ports Used by the Various Services (Unique process to every user)
Note: These steps are going to be unique to most Internet Service Provider/Modem & Router OR Gateway combinations, so I'll be providing some generalities that you can start google searching around with.

Youtube Video that helps de-mystify port-forwarding (start at time 2:15 for quick summary)

Your network is set to block incoming requests by default (so that malicious actors aren't able to get into your network), but sometimes we want users outside of our network to be able to reach services (for example, the server software) without being stopped by our network security. The solution to this is port forwarding, where we essentially say "Hey, if users try to reach resources on this port, let them through".

In the config.json, there are port entries for the following services (these values can change if you've edited your config to use non-default values):

Database: 5432

Launcher: 80
Sign: 53312
Channel: 54001
Entrance: 53310

You will want to forward ports for everything below the line. You could forward the port for the database, but you likely don't want external users having access directly to the PostgreSQL database because they interact with the server which interacts with the database on their behalf. If you forward the port for the database, the only thing stopping users from doing anything they want on your database instance is not knowing your database user (which is the default postgres so they actually do know that) and its password (which they really shouldn't know). If that password is compromised, all bets are off. In short, don't port forward the database port unless you have a reason to and know what you're doing.
Once you've forwarded these ports for the various services that use them, outside clients **IN THEORY** should be able to connect to your server (once it's running in Step 6)

Step 6: Run the server!

- Navigate to the root directory of your erupe server files
- In the file browser path, type in cmd and then press enter to launch a cmd prompt at this location (image above from phase 3 shows the cmd prompt being launched from the correct directory)
- Enter the following command: go run .
    - Assuming everything has been set up correctly, you should now have a functioning server that clients are able to connect to (they need to follow the section about adding entries to their hosts files here, but instead of 127.0.0.1 they should be entering the ip you found in step 5.5
        - In short, what this does is change the outbound request for the urls (example being mhfg.capcom.com.tw) routing from that actual location to the newly specified IP address. This enables the server to get this request and act as if it's the (now offline) Capcom server.
        - If you want to play the game on the same machine you're hosting the server on, you also need to do the host entries instructions above but you WILL KEEP the 127.0.0.1 entries
- To stop the server, press ctrl+c

Client Side Installation

Step 1: Download the Japanese Pre-Installed Client Files

- Get the MHF-ZZ_Installed_Files.zip from archive.org
- Extract the contents of this folder wherever you want the game

Step 2: Download Python and Install Frida

- Download Python (latest version- I used 3.9.1 successfully)
- Install and **MAKE SURE YOU SELECT THE OPTION TO ADD PYTHON TO PATH**
- Open a new cmd prompt (not a python shell)
- Type: pip install frida

Step 3: Download/Copy the Client Patcher

- Within your MFH-ZZ_Installed_Files folder (or whatever you named the folder containing mhf.exe) either download this script or copy and paste it into the same named file within this directory.
    - Note: This has to be done for two reasons, there are protections on the mhf.exe itself called AsProtect (please reach out if you can provide step-by-step instructions in text or video form on how to remove AsProtect and patch out GameGuard) and an Anti-Cheat program called GameGuard. Both prevent us from being unable to connect to private servers, and this python script bypasses both to allow us to successfully launch the game.

Step 4: Edit your Hosts File

- We need to fool our client into thinking it's reaching out to the official capcom jp or tw servers when in reality it's connecting to our local ip (127.0.0.1) or an external host ip (the external ipv4 address of whomever is hosting)
- Navigate to C:/Windows/System32/drivers/etc/hosts and add the following entries:

  127.0.0.1 mhfg.capcom.com.tw
  127.0.0.1 mhf-n.capcom.com.tw
  127.0.0.1 cog-members.mhf-z.jp
  127.0.0.1 www.capcom-onlinegames.jp
  127.0.0.1 srv-mhf.capcom-networks.jp

  Note: Any time you're messing with files in System32, really take an extra minute to verify you know what operation you're doing and why. I imagine a lot of less technically inclined people will try this, and in general anytime you find yourself in some part of the System32 directory ensure that you're not being led horribly astray. That's why throughout this guide I try to give you the "why" of what you're doing.

Step 5: Run the Client Patcher

- For this to work, the server has to be running in order to authenticate against our private server
- Open a cmd prompt as an **administrator** and navigate to the root directory of your MHF-ZZ_Installed_Files folder (or wherever you extracted its contents) and enter the following: no_gg_jp.py
    - The game launcher should now open and bring you to a screen with a username and password. Enter anything for these as the server will create a new entry in the db if the user doesn't exist..

- Select the premade character and enter the game. This will allow you to make your actual character in-game.

Misc things: