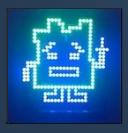


Weasel's Game-Administrator / Server-Manager (wGASM), for Linux - v2

By [WL] Weasel (Probably AFK)

Note:

Please click the **like / thumbs-up button** if you found this guide useful or interesting. Hundreds of hours of work went into the guide and the related dedicated game-server management system - combined with its predecessors, more like over a thousand hours.

Support:

For questions and support with wGASM please use this Steam Community Discussions area:
https://steamcommunity.com/groups/WeaselsLair/discussions/20/

Features:

- Detailed instructions for **preparation** and **installation** in the guide.
- Easy-to-use script, to **prepare** Debian Linux for the system. It installs all required packages, offers to create the unprivileged user, offers to install Webmin [webmin.com] , etc.
- Easy-to-use script, to **install** the system under the unprivileged user. It downloads the system from GitHub, puts everything in the expected folders, offers to install SteamCMD for you, and offers to download and install the latest "**Stencils**" archive.
- Some *moron-detection* to prevent you from running things under the wrong Linux user (i.e. do NOT run game-servers under **root**, etc.)
- Many configuration options are driven by a *text-based* configuration file.
- Server details, game-types and other data stored in easy-to-edit plain-text and **tab-separated-value** (TSV) files.
- No requirement to know how to use SteamCMD yourself - It does it all for you!
- Supports **30+** games on Steam!
- Of those, **10** have been *fully tested* and included in the examples.
- Optional integration with Webmin [webmin.com] - to enable operating everything in a web-browser!
- Even the Webmin [webmin.com] stuff is driven by easy-to-edit *plain-text* files.
- *Minimal Linux experience* required = know how to access with SSH, enter commands in the shell, not much else!
- *Minimal system requirements* = depends mostly on what games you are hosting. Otherwise, the *minimums* would probably be something like:
  - 2x CPU (**virtual** CPU's or physical CPU "**cores**").
  - 2-GB of **Memory**
  - 100-GB of **Storage**.
- *Minimal access requirements* = SSH access, seperate **root** (or *privileged* **sudo-enabled**) and *unprivileged* **game-servers** users.

- Supports creating and using "**Stencils**" (canned-configurations) that maybe be "**Painted**" onto (applied to) game-servers - making setting them up quicker. Examples are available for the 10 tested games.
- Little (if any) *script editing* required - maybe just for scheduled cron jobs if you're into that.
- If you want to setup scheduled cron [en.wikipedia.org] jobs to do maintenance (scheduled **backups**, graceful **automatic updates**, and automated **health-checks**), there are functions for doing-so, which you can schedule! (and examples in the guide).
- Inexpensive support! ("best effort") :salt:

About the System:

The system is written in Bash [en.wikipedia.org] , with the support of several standard Linux command-line text-processing utilities. The system allows for easily completing many actions associated with hosting dedicated game-servers. This includes actions such as:

- **Installing** the game-servers in a consistent manner.
- Applying "(**painting**") game-servers with various canned-configurations ("**stencils**").
- **Running** game-servers interactively at the console - for testing / troubleshooting.
- **Starting** game-servers as disconnected / background processes - so that they can run unattended.
- Sending **commands** to game-servers while running as disconnected / background processes
- **Stopping** disconnected / background processes of the game-servers.
- Manually **updating** game-servers if needed.
- **Checking** for any updates available for game-servers, and conditionally stopping game-servers, updating them, and restarting them - *only if an update is needed*.
- **Monitoring** game-servers running as disconnected / background processes for fatal conditions, and automatically restarting them - *only if intervention is needed*.
- Generating lists of "stock" content to **exclude** from backups.
- Making **backups** of game-servers.
- **Restoring** game-servers from backups.

Where to get stuff:

The source for everything in this system (except the "**Stencils**") is available in this GitHub repository:

https://github.com/Mecha-Weasel/wgasm

The example '**Stencils**" are too big (and perhaps not appropriate) to host in GitHub, Consequently, the latest archive of example '**Stencils**" is hosted in my **Internet Archive** account and also in my public **DropBox** folder. Also hosted in that same location(s), are example **virtual machines** for:

- VirtualBox [en.wikipedia.org] (Linux, Windows and many other systems)
- VMware Workstation [en.wikipedia.org] (Windows)
- VMware Fusion [en.wikipedia.org] (Mac, use same image as **WMware Workstation** above)

To find the resources in the **Internet Archive**, search using this URL:

```
https://archive.org/details/@weasel_steamid_155/lists/1/wgasm
```

The URL for my public **DropBox** is:

```
https://www.dropbox.com/scl/fo/qr0nxjefykegw3ix9p98s/AGHvx9t3sqoH_l88uR9FCso?rlkey=l0h4kyqz3wbmdne4rbw0exc1r&st=s823n1fe&dl=0
```

Hash-tags / Keywords:

- #wgasm, #[WL], #Weasel, #WeaselsLair, #WeaselsLair.com,
- #GoldSrc, #HLDS, #HLDedicatedServer, #MetaMod, #AMXModX, #AMX-Mod-X,
- #Source, #SRCDS, #SourceDedicatedServer, #MetaMod:Source, #MetaModSource, #MetaMod-Source, #SourceMod, #Source-Mod
- #Linux, #Debian, #bash, #webmin, #github
- #VirtualBox, #Virtual-Box,

- #VMWare, #VMwarePlayer, #VMware-Player, #VMwareWorkstation, #VMware-Workstation,

0 - Revision History

Be sure to check here for anything that may have changed, since the last time you read-through or used this guide.

NOTE: Latest updates at the top,

Revision Date	Revision Notes
20250104	Updated notes regarding **Stencils** to reflect migration of official sample stencils archive from **DropBox** to **Internet Archive**.
20241207	Updated **Assumptions** section with note regarding efforts to get the system working under **Rocky Linux 9.x**.
20240811	Updated/fixed related DropBox URL for example virtual machines.
20240726	More minor typographical and grammatical fixes.
20240708	Minor typographical and grammatical fixes.
20240628	Minor typographical and grammatical fixes.
20240626	Example **virtual machine** section(s) completed! Updated **stencils** section to include a new stencil for Day-of-Defeat with Marine-Bot.
20240625	Minor corrections. Still not *completely* done with the example **virtual machine** section(s).
20240621	Guide roughly 90-something% complete. Just example **virtual machine** section(s) remaining!
20240616	Guide roughly 95% complete. Just **Webmin** and example **virtual machine** sections remaining!
20240615	Guide roughly 90% complete. Just **cron jobs** and **virtual machine** sections remaining.
20240614	Guide roughly 85% complete. However, still "under construction".
20240613	Guide roughly 80% complete. However, still "under construction".
20240612	Initially made public / published. However, still "under construction".
20240601	Initial creation, pre-publication.

1 - License

MIT License

Copyright (c) 2023-2024 weasel.steamid.155@gmail.com

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

```
THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT
NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
```

CREATED BY

[WL] Weasel (Probably AFK)
Offline

Category: Modding or Configuration, Multiplayer

Languages: English

Posted Jun 12, 2024 @ 6:35pm
Updated Jan 4 @ 8:44pm

121 Unique Visitors
7 Current Favorites

GUIDE INDEX

2 - About the Author, Credits & Thanks

I have been hosting dedicated game-servers for various Valve software game titles, since late 1999 or early 2000 - long before there was any such thing a "Steam". I switched from attempting to host game-servers on Windows to Linux, some time in the early 2000's (circa 2007?).

In early 2024, completely re-wrote all my Linux (bash) scripts used for management of my dedicated game-servers When I re-wrote everything, I tried to make them less proprietary, and more easily re-usable by other people needing similar scripts. These became effective the *first (published) generation* of the scripts.

In mid 2024, I created this *newest generation* (**v2**) as more of a complete *system* rather than just a bunch of random *scripts*. In particular, *everything is now driven entirely by* ***text-based data files***, including *both* a **central configuration file** and several **tab-delimited tables** . There should be little if any reason for anyone using them to edit the scripts themselves - other than the optional "cron" scripts (meant to be become the basis for scheduled cron [en.wikipedia.org] jobs).

I would especially like to thank the people behind the following free or paid products services and resources noted below. There have been many over the years that have also personally assisted me with various issues - too many people, and on too many occasions to cite.

Various (Free) Products / Services / Resources:

- Valve Mailing-Lists: Thanks to all the helpful members on various lists that I have corresponding with over the last 20+? years.
- Valve VDC: Various documentation about Valve's games, game-engines and related tools for dedicated servers, map-making, etc.
- VERC [twhl.info] : The older version of Valve's reference material.
- MetaMod [metamod.org] : A system for having multiple server-side mods coexist on the same game-server running the ancient GoldSrc [en.wikipedia.org] game-engine.
- AMX-Mod-X [www.amxmodx.org] : An administrative server-side modification for game-servers running the ancient GoldSrc [en.wikipedia.org] game-engine. Allows for an extensive system of "plug-ins", many of which are community-created.
- MetaMod:Source [www.sourcemm.net] : A system for having multiple server-side mods coexist on the same game-server running the older Source [en.wikipedia.org] game-engine.
- SourceMod [www.sourcemod.net] : An administrative server-side modification for game-servers running the older Source [en.wikipedia.org] game-engine. Allows for an extensive system of "plug-ins", many of which are community-created.
- Allied-Modders [forums.alliedmods.net] : The organization to publishes MetaMod [metamod.org] , MetaMod:Source [www.sourcemm.net] , AMX-Mod-X [www.amxmodx.org] and SourceMod [www.sourcemod.net] . And thanks to all the helpful users on their forums that I have corresponding with over the last 20+? years.
- Bots-United [www.bots-united.com] : The clearing-house for "bots" that work with a wide variety of games. Forums also populated by helpful people knowledgeable of such things!
- FoF Saloon: A dedicated Steam community for Fistful of Frags with an active Discord [discord.gg] .
- Webmin [webmin.com] : A web-based management interface for various flavors of Linux, which makes it much easier to learn and use.
- GitHub [github.com] : For all the code they host. Thanks also to all the folks who have published their work up on there for others to see and use.

Various (Paid) Products / Services:

- Valve [en.wikipedia.org] : The company that publishes various games that I have played and/or hosted dedicated game servers for - going back to Half-Life [http//Half-Life] (HL1).
- NFO Servers [www.nfoservers.com] : A game-server hosting provider, with excellent DDoS detection and remediation services.
- Google One [one.google.com] : The expanded (paid) version of Google Mail and Drive with greater storage limits.

- InsyncHQ [www.insynchq.com] : Publishers of *both* a desktop-mode Google Drive client for Linux (and other platforms), and a "headless" version for remote Linux servers. Neither free of course, but good products!
- DropBox [www.dropbox.com] : A web-based content host, used for stuff that is too big or outside of the scope for GitHub [github.com] .
- GitKracken [www.gitkraken.com] : An excellent Linux GUI front-end for Git-repositories (including GitHub [github.com]). There is a free/non-commercial version available.
- Sophos [www.sophos.com] : An anti-malware system for various platforms. They have an excellent and relatively inexpensive "home" product for Windows, as well as several enterprise products.

---

3 - Getting Started

Obviously, you ***should*** *read and comprehend* ***everything*** *in this guide* before proceeding. In reality, many will likely just skim this guide, and rush-ahead trying to just install and run this system - perhaps coming back to this guide later to figure things out. For those with little or no patience, the next part is for you.

Quick-Start: (for the impatient)

- Acquire, rent or build your own **Linux** computer - using Debian [www.debian.org] .
- Read the **Planning Requirements** section.
- Read the **Knowledge Requirements** section.
- Read the **Assumptions** section.
- Do the stuff in the **Preparing the OS** section (**prepare-debian.sh --nike**).
- Do the stuff in the **Downloading the System** section (**install-wgasm.sh --nike**).
- Edit the **game-servers** table - to add your servers, or just tinker with the examples.
- **Install** one or more game-servers (using **GameServerID**'s from the game-servers table).
- **Paint** one or more game-servers with a *base "starter" configuration*.
- **Run** one or more of the game-servers *interactively* - to make sure it works.
- **Start** one or more servers in the background and try to connect & play on it.
- Use other scripts as desired to understand their functionality.
- If using Webmin [webmin.com] , make sure the related "**Custom Commands**" are installed.
- If using Webmin [webmin.com] , also update your "lists" in the **~/wgasm/webmin** folder - being especially sure to update **list-backups-all.txt** and **list-servers-all.txt** (which are used by the Webmin [webmin.com] "**Custom Commands**" feature).
- If you want to schedule some *automated maintenance*, check-out the "**cron**" scripts.

---

4 - Planning Requirements

WARNING: Do NOT skip this section! *Everything else in this guide is dependent on it!*

For **each** game server you are planning to host (*even if its just a* ***single*** *game-server*), you will need to plan ahead with **three** (**3**) pieces of information:

1. A ***unique*** *alphanumeric* "**gameserverid**". This is basically a short name that the scripts will use to identify which specific game-server you are trying to operate on. This could be as simple as "gameserverid1", or more complex such as "hl1server1", "hl1server2, "cs1server1", "cs1server2", etc. These names must be **unique** to each server instance, and must be entirely *lower-case alpha-numeric*: Only containing characters **a-z** and/or **0-9**, *no upper-case letters, no spaces nor any punctuation allowed.* Given that file systems on Linux are generally *case-sensitive*, keeping all of these *lower-case* prevents a wide-range of possible issues. Update: *Technically*, the scripts generally allow for a **hyphen** ("**-**") in the **gameserverid** also. *But that is the* ***only*** *punctuation allowed.*

2. A ***unique*** *numeric* network **port number** for each game-server. Obviously, this must be an all numeric value - Only containing characters **0-9**, *no letters, spaces or any punctuation allowed.* The default is typically 27015. if you are hosting multiple game-servers, they must each have their own unique port number assigned. Obviously, they can not *all* be 27015. Generally, a value somewhere in the **27000-27999** range is *typical for these types of game-servers*. However, any number in the 1024-49151 range *should* work, *assuming that number is not already used* by some other process on your Linux host. Due to some

bugs / features of how some game-engines work, it is *recommended* to *skip at least one number* between game-servers. A typical pattern might be something like this:

Server	Port Number
First game-server	27115
Second game-server	27117
Third game-server	27119
Fourth game-server	27121

3. The Steam **AppID** for that *game*, *and* it's *dedicated server* AppID. In some cases, the AppID of the game and its dedicated server are the same. In other cases, they are *completely different*. In many cases the game's ***dedicated server* AppID** is used for installing and updating the game-server, while the game's ***client* AppID** is used to check for availability of updates.

   Where possible, the sample **game-types** table provided ***already has this information properly included***. The example **game-types** file already includes:

   - Over **40** games, based on their information from SteamDB [steamdb.info] .
   - Of those, **10** have been tested/validated with this system.
   - The rest of those are untested / unvalidated, but *should* work with this system.
   - Additionally, also listed *just for reference* are several (10+) games that are known **not** to work - for various reasons (as noted in the **game-types** table).

   Steam AppID's for some common game titles are included in the table below:

Game (Valve)	Game AppID	Server AppID
Half-Life (Hl1)	70	90
Deathmatch Classic (DMC)	40	90
Team Fortress Classic (TFC)	20	90
Team Fortress 2 (TF2)	440	232250
Counter-Strike 1.6 (CS1)	10	90
Counter-Strike:Source (CSS)	240	232330
Counter-Strike 2 (CS2)	730	730
Day of Defeat (DoD)	30	90
Day of Defeat:Source (DoDS)	300	232290
Game (3rd-party)	**Game AppID**	**Server AppID**
Fistful of Frags (FoF)	265630	295230

5 - Knowledge Requirements

To effectively implement and utilize this system, you will need to know how to perform certain (common) actions on Linux. These include:

- **Must** know how to **login** to the host Linux installation using SSH or some other mechanism to access the Linux command-line.
- **Must** know how to **navigate** the folder structure, at the command-line. Typically, this involves commands such as "**pwd**" and "**cd**".
- **Nice** to know how to **install** and update **packages** for your host operating system. Typically, this is accomplished with various "**dpkg**" and "**apt**" commands. There is an installation script available (**prepare-debian.sh**) *that will make this **easier***, but some experience might be needed for troubleshooting.
- **Nice** to know how to change **permissions** on files. Typically, this involves commands such as "**chmod +x**", or through a web-interface such as "**Webmin**". In this guide, sample commands

for doing this are included. So, not really a big deal if you have never done this before.

- **Nice** to know how to **upload** files to the appropriate folders of your Linux installation. This may be accomplished any number of ways, including: "**wget**" to pull files directly from the internet, "**scp**" over and SSH session, or web-interfaces such as "**Webmin**". If you are using Webmin [webmin.com] , *this will obviously be a lot easier*. In this guide, sample commands to download the required files (preparation and installation scripts) from the command-line are included. So, not really a big deal if you have never done this before.

- **Nice** to know how to **edit** text files. You will be doing this quite a lot. This typically involves using either command-line editors such as "**nano**", or editing files through a web-interface such as "**Webmin**". In this guide, sample commands for doing this are included. So, not really a big deal if you have never done this before.

- **Ideally**, know how to create symbolic links [wiki.debian.org] . You may need to know how to create **symbolic links** to work-around bugs in the SteamCMD installation process. Sometimes, it does not automatically create all the symbolic links to various **.so** files that the game-servers require. Typically symbolic links are created either at the command-line using the "**ln -s**" command, or through a web-interface such as "**Webmin**". There is an installation script available (**install-steamcmd.sh**) *that should make this easier*, but some experience might be needed for troubleshooting.

6 - Assumptions

- This system is designed to run under the common Linux command-line shell called bash [en.wikipedia.org] . This system requires that everything is being performed under some distribution (flavor) of **Linux**. ***None*** *of this is intended to or likely to work under* ***Windows****. If you want to try "WSL" or some similar situation, you are on your own (****period****).*

- The specific distribution (flavor) of Linux being used is assumed to be Debian [www.debian.org] . Other distributions in the Debian Family [en.wikipedia.org] of Linux (such as Ubuntu [ubuntu.com] , or Mint [www.linuxmint.com]) **should** work.

- However, any distribution *outside* of the Debian Family [en.wikipedia.org] *likely will* ***not*** *work*. In particular, *prepare* and *installation* components rely heavily on *package-management* commands (such as **dpkg** and **apt**).

- This is can be particularly problematic with any of the Red-Hat Family [en.wikipedia.org] of Linux distributions - since they typically use a *completely different package-management system* (typically **dnf** or **rpm** and **yum**) and unavailability of certain packages. Of course, you are welcome try. But, you do so at your own risk.

- **NOTE:** I have been tinkering with getting this system working with Rocky Linux [rockylinux.org] (part of the broader Red-Hat family [en.wikipedia.org] of distributions), but I am stuck on at least one package that is *critical* to (i.e. ***extensively*** *used by*) this system. That package is colorized-logs [github.com] , specifically the **ansi2txt** utility. If anyone knows *exactly* how to get that installed on Rocky Linux 9.x [rockylinux.org] , ***please contact me!***

- This guide also assumes not only that the Linux distribution being used is Debian, but also that the specific installation being used is the *minimal* Debian installation - which has no Graphic-User-Interface (GUI) elements. This is sometimes referred to as "network installer" boot-CD image. It allows the creation of Debian for "Servers" - which are typically "headless", meaning they are intended to operate without any local keyboard and/or monitor attached. If you are installing Debian yourself, select the image to download for "**amd64**" from: https://www.debian.org/CD/netinst/

- This system requires that all package dependencies have been installed previously using the Linux "**root**" user, or a similarly *privileged* ***sudo****-enabled* user. *Except for the* ***SteamCMD*** utility - as noted below.

- The SteamCMD utility ***and*** the game-servers will be running under a *different Linux user* - something *other than* "**root**" - and that *does not* have any *privileged* ***sudo****-access*. This system and all examples shown here assume *this user* will be named: "**game-servers**".

- This system only works with common games using either Valve's older "**Source**" (HL2-ish) game-engine, or the even older/*ancient* "**GoldSrc**" (HL1-ish) game-engine. Accommodation for **Source2**-based Counter-Strike 2 is included - but is *somewhat experimental*. As more **Source2**-based games become available, this system may be updated accordingly.

- It should be possible to adapt this system to any other games - as long as they are based upon either the **Source** or **GoldSrc** game-engines. This should also be the case for any 3rd-party total-conversion "mods" of such games. Just adding their details into the **game-types** table *should* be enough - "fingers-crossed".

---

7 - Preparing the OS, Part-1

Linux user requirements:

As has already been mentioned, to be successful in installing and using this system, you will need ***two (2)*** *distinct Linux users*:

- You will need to login to Linux initially as either "**root**" or some user with equivalent privileges using "**sudo**". If you installed Linux yourself, you should already know the details (password) for the "**root**" user. If you are renting a server on the Internet, then your hosting provider may have provided you with the "**root**" user details, **or** provided you with a different user *that has **root**-like privileges by way of the "**sudo**" command*.

  You will use this ***root-like/privileged/sudo*** user (or **root** itself) to:

  - Install various operating system packages and generally make sure the operating system (presumably Debian [www.debian.org]) is ready.
  - Create the second *non-**root**/non-**privileged**/non-**sudo*** user (as noted below).
  - Optionally, install Webmin [webmin.com] - which is *highly recommenced*, since Webmin "Custom Commands" are available that can front-end this system.

- You will also need a *second*, normal (*non-**root**/non-**privileged**/non-**sudo***) user. This will be the user that you login as to:

  - Install this system.
  - Install SteamCMD.
  - Configure this system.
  - Install various game-servers.
  - Use this system to stop/start/manage game-servers.

NOTE: Other than the **prepare-debian.sh** script, if you attempt to run any other parts of this system under **root** or another ***root-like/privileged/sudo*** user, it will *display and error message and stop*. This feature is known as "**Moron Detection**".

Ensuring "sudo" function exists:

Obviously, if you are using a rental Virtual Private Server (VPS), and your hosting provider gave you an alternative user login with root-like privileges via "sudo", then it must already be installed. Therefore, you can *skip this part*.

For those installing Debian Server themselves, they may find that new installations *do not automatically install* the "**sudo**" utility by default.

Consequently, if you installed Debian Server directly yourself, before you can go any further, you will need make sure it is installed - since it is invoked extensively in the **prepare-debian.sh** script. You can can install it (logged-in as "**root**"), by entering the following command:

```
apt-get install -y sudo;
```

Ensuring you have remote/network SSH access:

Obviously, if you are using a rental Virtual Private Server (VPS), and your hosting provider gave you user login credentials and an IP address to connect to, then that user must already have SSH permissions. Therefore, you can *skip this part*.

For those installing Debian Server themselves, they may *also* find that new installations of some Linux distributions *do not automatically allow* the"**root** user to connect to the server using SSH over the network by default.

Consequently, if you installed Debian Server directly yourself, you may need to enable the **root** user to access the computer over the network with SSH. To do so, you must use a text-editor (such as nano [en.wikipedia.org] to edit the **/etc/ssh/sshd_config** file.

Enter this command, to ensure nano [en.wikipedia.org] is installed:

```
apt-get install -y nano;
```

Enter this command, to run nano [en.wikipedia.org] , and edit the **/etc/ssh/sshd_config** file:

```
nano /etc/ssh/sshd_config;
```

Inside nano, you can use the arrow-keys on the keyboard to navigate up and down the contents of the file. You will need to find the line that contains this text:

```
PermitRootLogin prohibit-password
```

That line will need to be changed to read:

```
PermitRootLogin yes
```

Once you have made those changes, press **CTRL** + **O** (for "output") and then **Y** to save the changes. Once the change is saved, you may then **CTRL** + **X** (for "exit") to close nano [en.wikipedia.org] .

You may then *reboot the computer* to ensure SSH access is working now for **root**, using the following command:

```
reboot;
```

BTW: If you find that you need to cleanly shut-down the machine from within the OS, you may do so using the following command:

```
shutdown -h now;
```

Ensuring other perquisites for this system are in-place:

In order to ensure all other components that the system requires are actually installed, the **prepare-debian.sh** script is provided in the related GitHub repository:

https://github.com/Mecha-Weasel/wgasm

The "**prepare-debian.sh**" script performs the following tasks:

- Ensures you are running it under a user with sufficient privileges ("**root**" or similarly privileged **sudo**-*enabled* user.
- Ensures you are using a compatible Linux distribution (i.e. something in the Debian Family [en.wikipedia.org]).
- Enables the **Intel/AMD *32-bit*** architecture for applications and code-libraries - required by some game-servers. This architecture support can be installed alongside any existing **Intel/AMD *64-bit*** architecture.
- Installs the latest updates for your Linux installation.
- Runs the installation command for various required packages. Of course, many of these may already be present in your existing Linux installation.
- Runs a check to ensure that the commands and utilities it expects are actually available after the installation/update attempt.

- If the **game-servers** user does not already exist, it will offer to create it for you.
- If the **game-servers** user does *now* exist then it will offer to pre-download the **install-wgasm.sh** installation script into the home-directory of the **game-servers** user.
- If Webmin [webmin.com] is not already installed, it will offer to install it for you.
- If Webmin [webmin.com] is *now* installed, it will offer to setup the **game-servers** as a Webmin [webmin.com] user.

Obviously, if any of the checks fail during **prepare-debian.sh** script, it will display an error message, and you will have to troubleshoot why the command or utility listed is not available or not working.

You may download the **prepare-debian.sh** script with the following command:

```
wget -O prepare-debian.sh https://github.com/Mecha-Weasel/wgasm/raw/main/prepare-debian.sh;wait;
```

Once downloaded, mark the **prepare-debian.sh** script as executable with this command:

```
chmod +x prepare-debian.sh;
```

Once downloaded and marked executable, you may run the **prepare-debian.sh** script with the following commands:

```
sudo ./prepare-debian.sh;
```

If you want the **prepare-debian.sh** script to assume "**Y**" to all prompts, you may add the "**--nike**" parameter, to tell it to "*just do it*". Invoking "Nike Mode", the command would instead be:

```
sudo ./prepare-debian.sh --nike;
```

---

8 - Preparing the OS, Part-2

Creating the 2nd user:

If you did not opt to have the **prepare-debian.sh** script automatically create the 2nd user for you (which would have been named "**game-servers**"), you will need to do that manually. This can be accomplished with the following commands:

```
sudo useradd -m -s /bin/bash game-servers;
sudo passwd game-servers;
```

You may select a different name for this user instead of the default "**game-servers**". However, if you do so you *will not be able utilize the related* Webmin [webmin.com] *"**Custom Commands**" setup* - since those required hard-coded paths to find various components.

Installing Webmin: (optional)

If you did not opt to have the **prepare-debian.sh** script automatically install Webmin [webmin.com] , you may want to do that manually.

While this system does not directly depend upon Webmin [webmin.com] to operate from the Linux command-prompt over SSH or on the console of a server, no direct web-interface is provided. If you want to use this system from a web-browser, a related package of Webmin [webmin.com] "**Custom Commands**" is available.

To install Webmin [webmin.com] , first install its repository settings, using these commands:

```
wget -O setup-repos.sh https://raw.githubusercontent.com/webmin/webmin/master/setup-repos.sh;
chmod +x ./setup-repos.sh -f;
sudo ./setup-repos.sh -f;
```

You may need to respond to a prompt with "Y", to allow the installation of the Webmin [webmin.com] repository settings.
Once the Webmin [webmin.com] repository settings are installed, use this command to actually install Webmin [webmin.com] :

```
sudo apt-get install -y --install-recommends ntpdate iptables webmin;
```

Installing "Custom Commands" for Webmin: (optional)

If you did not opt to have the **prepare-debian.sh** script automatically install this systems "**Custom Commands**" into Webmin [webmin.com] , you may do that manually by downloading the archive (**/webmin-custom.tar.gz**) and extracting its contents into the **/etc/webmin/custom** folder. This may be accomplished with the following commands:

```
cd /etc/webmin;
wget -O webmin-custom.tar.gz https://github.com/Mecha-Weasel/wgasm/raw/main/webmin-
custom.tar.gz:wait;
tar xfv webmin-custom.tar.gz;
```

Setting-up "game-servers" as a Webmin user: (optional)

If you did not opt to have the **prepare-debian.sh** script automatically setup the **game-servers** Linux user as a Webmin [webmin.com] user, you should *just go back and let the script do that for you*.

There is *way too much* to explain here to get it right **manually**. Just run the script **prepare-debian.sh** script, and add the "**--nike**" command-line paramater to tell it to "*just do it*". Invoking "Nike Mode", the command would be:

```
sudo ./prepare-debian.sh --nike;
```

If you really want to know how to do that manually (at a command-line anyway, which is probably not the easiest way), you are welcome to read through the **prepare-debian.sh** script. The section involved starts around **line 577** or so. Just search for the comment text:

```
Offer to setup game-servers as a Webmin user
```

---

9 - Downloading the System

Downloading and running the installer:

The **install-wgasm.sh** script is provided in the related Git-Hub repository:

https://github.com/Mecha-Weasel/wgasm

The **install-wgasm.sh** script performs the following tasks:

- Ensures you are **NOT** running it under a user with **root/sudo** privileges ("**root**" or similarly privileged **sudo**-*enabled* user).
- Runs a check to ensure that the commands and utilities it expects are actually available and working as expected.
- Downloads the content of the system from the Git-Hub repository [github.com] into the *expected location* (**~/wgasm**).
- Offers to download the SteamCMD utility, and install it into its *expected location* (**~/steamcmd**).
- Offers to download the "Stencils" archive from DropBox [www.dropbox.com] , and install it into its *expected location* (**~/stencils**).

Obviously, if any of the checks fail during **install-wgasm.sh** script, it will display an error message, and you will have to troubleshoot why the command or utility listed is not available or not working.

You may download the **install-wgasm.sh** script with the following command:

```
wget -O install-wgasm.sh https://github.com/Mecha-Weasel/wgasm/raw/main/install-wgasm.sh;wait;
```

Once downloaded, mark the **install-wgasm.sh** script as executable with this command:

```
chmod +x install-wgasm.sh;
```

Once downloaded and marked executable, you may run the **install-wgasm.sh** script with the following commands:

```
./install-wgasm.sh;
```

If you want the **install-wgasm.sh** script to assume "**Y**" to all prompts, you may add the "**--nike**" parameter, to tell it to "*just do it*". Invoking "Nike Mode", the command would instead be:

```
./install-wgasm.sh --nike;
```

10 - Installing SteamCMD

SteamCMD is a utility provided by Valve [en.wikipedia.org] that is used for installing dedicated servers for games that have their content stored up in the Steam [en.wikipedia.org] cloud service.

This system will use SteamCMD to install dedicated servers based on any of the following game-engines:

- The ancient GoldSrc [en.wikipedia.org] game-engine, used by games such as:
  - Half-Life (HL1)
  - Deathmatch Classic (DMC)
  - Team Fortress Classic (TFC)
  - Counter-Strike 1.6 (CS1)

- The older Source [en.wikipedia.org] game-engine, used by games such as:
  - Half-Life 2 (HL2),
  - Half-Life Deathmatch:Source (HLDMS),
  - Team Fortress 2 (TF2),
  - Counter-Strike:Source (CSS),

- The newer Source2 game-engine, used by games such as:
  - Counter-Strike 2 [en.wikipedia.org] (CS2).

If you did not opt to have the **install-wgasm.sh** script automatically install SteamCMD, you will need to do that manually. Fortunately, an easy installation script for SteamCMD has also been provided. It should already be present in the **~/wgasm** folder. You can install SteamCMD using the **install-steamcmd.sh** script using the following commands:

```
cd ~/wgasm;
chmod +x install-steamcmd.sh;
./install-steamcmd.sh;
```

11 - Installing Example Stencils

This system implements a feature known as "**Stencils**". These are essentially standard configurations that may be applied to (or "painted" onto) a game-server. They may also be used to add various server-side modifications to an existing game-server configuration. Each **Stencil** is specific to a particular **game-type**.

Several example **Stencils** are available, including:

- For several example games, base "starter" configurations to take a game-server from a blank installation to something that is at least minimally ready to connect to from a game-client.
- For several example games that are based on the GoldSrc [en.wikipedia.org] game-engine, a basic setup for AMX-Mod-X [www.amxmodx.org] - with or without 3rd-party "bots".
- For several example games that are based on the Source [en.wikipedia.org] game-engine, a basic setup for SourceMod [www.sourcemod.net] .

Although the list of example **Stencils** is already included in the **~/wgasm/data/game-stencils.tsv** file - their actual content is too large to house directly inside the Git-Hub repository [github.com] . Consequently, the content for the **Stencils** has been archived and uploaded into this **Internet Archive** URL:

```
https://archive.org/download/stencils-latest.tar/stencils-latest.tar.gz
```

A copy is also typically maintained in my public **DropBox** folder:

```
https://www.dropbox.com/scl/fo/qr0nxjefykegw3ix9p98s/AGHvx9t3sqoH_l88uR9FCso?
rlkey=l0h4kyqz3wbmdne4rbw0exc1r&st=0cyziuuy&dl=0
```

Please note that downloading the example **Stencils** from either source will take quite some time. In my recent testing, this has typically ranged from **180 to 480 seconds**.

If you did not opt to have the **install-wgasm.sh** script automatically install the **Stencils**, you may want to do that manually. Fortunately, an easy installation script for the **Stencils** has also been provided. It should already be present in the **~/wgasm** folder. You can install the **Stencils** using the **install-stencils.sh** script using the following commands:

```
cd ~/wgasm;
chmod +x install-stencils.sh
./install-stencils.sh;
```

Example **Stencils** content in the archive includes:

StencilID	Description	Comment
cs1-base	[CS1] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
cs1-amx	[CS1] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
cs1-amx-bots	[CS1] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus YaPB.
css-base	[CSS] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
css-sm	[CSS] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
cs2-base	[CS2] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dmc-base	[DMC] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dmc-amx	[DMC] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
dmc-amx-bots	[DMC] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus HPB_Bot.
dod-base	[DoD] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dod-amx	[DoD] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
dod-amx-bots	[DoD] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X plus Marine-Bot.
dods-base	[DoDS] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dods-sm	[DoDS] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
fof-base	[FoF] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
fof-sm	[FoF] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
hl1-base	[HL1] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
hl1-amx	[HL1] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.

hl1-amx-bots	[HL1] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus JKBotti.
tf2-base	[TF2] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
tf2-sm	[TF2] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
tfc-base	[TFC] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
tfc-amx	[TFC] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
tfc-amx-bots	[TFC] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus FoxBot.

12 - The Central Configuration File

Many options used by the system are specified in the central configuration file. The configuration file is named "**config.txt**", and MUST be *located in the **same folder** as the main scripts themselves*. Of course the default scripts folder is **$HOME/wgasm** (effectively **~/wgasm**).

The config file is a plain-text file, and as such should be editable via any preferred text editor ("nano", "vim", or any GUI text editor, etc.).

Any lines in the **config.txt** file that start with "#" are comments and will be ignored.

The contents of the **config.txt** file, should looks something like this:

```
#
#       Actual values to use start here ...
#       -----------------------------
#
display_banner=true
scripts_verbose=false
moron_detection=true
purge_temp_installs=true
autobackup_by_check=true
autoexclude_by_check=true
servers_install_folder=$HOME
logs_folder=$HOME/logs
temp_folder=$HOME/temp
tempinstall_folder=$HOME/temp/temp-install
backup_folder=$HOME/backups
stencils_folder=$HOME/stencils
steamcmd_folder=$HOME/steamcmd
backup_configs_folder=$SCRIPTS_FOLDER/backup-configs
data_folder=$SCRIPTS_FOLDER/data
game_types_file=game-types.tsv
game_servers_file=game-servers.tsv
game_stencils_file=game-stencils.tsv
banner_file=banner.txt
steam_login="anonymous"
```

The meaning of each option in the **config.txt** file is detailed in the table below:

Option	Format / Default Value	Description
display_banner	"true" or "false" (Default: "true")	Enable/disable banner display.
scripts_verbose	"true" or "false" (Default: "**false**")	Enable/disable verbose output.
moron_detection	"true" or "false" (Default: "true")	Prevent running things under root/sudo that should NOT be.
purge_temp_installs	"true" or "false" (Default: "true")	Purge "exclude" installs.
autobackup_by_check	"true" or "false" (Default: "true")	Automatically backup before "check" forces an update.

autoexclude_by_check	"true" or "false" (Default: "true")	Automatically update "exclude" when "check" updates.
servers_install_folder	$HOME	Default folder to branch game-server installs from.
logs_folder	$HOME/logs	Folder where scripts and 'GNU screen' will store logs.
temp_folder	$HOME/temp	Folder that 7-zip will use as a working area.
tempinstall_folder	$HOME/temp/temp-install	Folder where "exclude" process will put temp installs.
backup_folder	$HOME/backups	Folder where "backup" scripts will store the backups.
stencils_folder	$HOME/stencils	Folder where "stencils" (canned-configs) are stored.
steamcmd_folder	$HOME/steamcmd	Folder where SteamCMD utility is installed.
backup_configs_folder	$HOME/wgasm/backup-configs	Folder where "include" and "exclude" files are stored.
data_folder	$HOME/wgasm/data	Folder where data files / tables will be stored.
game_types_file	game-types.tsv	The name of the game-types tab-separated-value file, relative to the "data" folder.
game_servers_file	game-servers.tsv	The name of the game-servers tab-separated-value file, relative to the "data" folder.
game_stencils_file	game-stencils.tsv	The name of the game-stencils tab-separated-value file, relative to the "data" folder.
banner_file	banner.txt	The name file with the banner content to display, relative to the "data" folder.
steam_login	"username password" or "anonymous"	Default Steam credentials to use. Be sure to enclose in quotes if not just anonymous.

13 - The Game-Types Table

The **game-types** table is stored in the **data** sub-folder (typically **~/wgasm/data**). The default filename is "**game-types.tsv**". The format of the file is a **tab-separated-value** (**TSV**) text-file. The **game-types** table functions as a database of information about how the system should handle installing and operating various games. The *order of each column is **critical***. The columns in this table are:

Order	Field-Name	Description
1	GameTypeID	ID for this game **type**. Each must be **unique**. These must contain only *lower-case letters* (**a-z**), *numerals* (**0-9**) and if you like a *hyphen* (**-**).
2	GameEngine	Game **engine** type. Currently, the only **engines** that this system supports are: • GoldSrc [en.wikipedia.org] (as "**goldsrc**") • Source [en.wikipedia.org] (as "**source**") • Source2 [en.wikipedia.org] (specifically, the **CS2** version as "**src2cs2**")
3	SteamLogin	If a non-default Steam login is required, the format is "username password" (all in one field).
4	AppIDInstall	Steam AppID to use for installation.
5	AppIDCheck	Steam AppID to use for checking for updates via Steam web API.
6	SteamCMDMod	Steam "mod" value to use for install (if any).
7	SteamCMDOpts	Extra install options for SteamCMD.
8	ModSubFolder	Mod sub-folder, relative to install folder (valve, cstrike, dod, game/csgo, etc.)
9	WarnStopText	Server-side command to warn players (in text) about a restart.
10	WarnStopAudio	Server-side command to warn players (in audio) about a restart.
11	WarnUpdateText	Server-side command to warn players (in text) about an update.
12	WarnUpdateAudio	Server-side command to warn players (in audio) about an update.

13	Description	Plain-text game-type description.
14	Comment	Notes about this game type (if any).

The information used to create the included **game-types** table, was collected from the SteamDB [steamdb.info] web-site - by filtering for which games have a seperate **Dedicated Server** app available for them, ***and*** also had either **Engine.Source** or **Engine.GoldSource** specified in their list of included technologies.

Also included is **Counter-Strike 2** (**CS2**) which uses the newer **Engine.Source2** technology. At this stage, it is not clear if later games built around the Source2 [en.wikipedia.org] game-engine will follow similar conventions with regard to the games executable name, launch parameters / requirements, etc. Consequently, the game-engine defined for **CS2** in the game-type table is named as "**src2cs2**" due to how it is currently implemented - which MIGHT be *unique* to **CS2** (time will tell, as other Source2 [en.wikipedia.org] -based games are released).

The **game-types** table *already includes a **large** selection of games* that (in theory) may be installed and managed via this system. In summary, the updated **game-types** table includes:

- **30+** games that ***likely*** can be installed and managed with this system.
- **10** of which have actually been installed and ***tested*** with this system.
- **13** additional games listed, but that are identified as being ***unusable***.
- **11** are ***unusable***, because *no **Linux**-native version* of their server is available.
- **2** are ***unusable***, because they are simply *not released* (yet).

The "comment" field of each **game-type** has been populated with notes about which game-types are tested, untested, unusable, etc.

In some rare cases, there are a few games that can not be installed using the default "**anonymous**" Steam login with SteamCMD. In those cases, the game must be *installed using the credential of a Steam account that "owns" the game* in Steam (i.e. has it in their Steam "Library"). In these cases, in the **game-types** table the "**steamlogin**" field has been populated with a generic placeholder entry of "**username password**" - to be replaced with a real Steam login *before* you try to setup and manage a server with that specific game. This will override the default Steam login credentials (from the central config file), for *just that specific type of game* - when trying to install or update it.

The list **30+** game-types which are either already tested, or appear LIKELY to be usable with these scripts, currently includes:

1. Counter-Strike 1.6 (CS1 - **Tested!**
2. Deathmatch Classic (DMC) - **Tested!**
3. Day of Defeat (DoD) - **Tested!**, but no Bot "Stencil" available.
4. Half-Life (HL1) - **Tested!**
5. Team Fortress Classic (TFC) - **Tested!**
6. Counter-Strike:Source (CSS) - **Tested!**
7. Day of Defeat:Source (DoDS) - **Tested!**
8. Team Fortress 2 (TF2 - **Tested!**
9. Counter-Strike 2 (CS2) - **Tested**, but no SourceMod "Stencil" available.
10. Fistful of Frags (FoF) - **Tested!**
11. Half-Life: Blue Shift (BShift) - *Untested.*
12. Half-Life: Opposing Force (Op4) - *Untested.*
13. Half-Life 2 Deathmatch (HL2DM) - *Untested.*
14. Half-Life Deathmatch:Source (HLDMS) - *Untested.*
15. Left for Dead (L4D) - *Untested.*
16. Left for Dead 2 (L4D2) - *Untested.*
17. Sven Co-op (Sven) - *Untested.*
18. Age of Chivalry (AoC) - *Untested.*
19. Brain Bread 2 (BB2) - *Untested.*
20. Black Mesa (BMS) - *Untested.*
21. Blade Symphony (BS) - *Untested.*
22. Codename CURE (Cure) - *Untested.*
23. Double Action:Boogaloo (DAB) - *Untested.*
24. Day of Infamy (DoI) - *Untested.*
25. Dystopia (Dys) - *Untested.*
26. Gary's Mod (GMod) - *Untested.*
27. Insurgency (INS) - *Untested.*

28. NSURGENCY: Modern Infantry Combat (InsMIC) - *Untested.*
29. Jabroni Brawl:Episode 3 (JBEp3) - *Untested.*
30. JBMod (JB) - *Untested.*
31. Military Conflict:Vietnam (MCV) - *Untested.*
32. Nuclear Dawn (ND) - *Untested.*
33. Natural Selection 2 (NS2) - *Untested.*
34. Pirates, Vikings, & Knights II (PVK2) - *Untested.*
35. The Ship (Ship) - *Untested.*
36. Zombie Panic:Source (ZPS) - *Untested.*

The common reasons to edit anything in the **game-types** table would be things like:

- Add actual Steam login credentials, in place of the "username password" place-holder.
- Update a game that formerly required Steam login, but no longer requires Steam login.
- Update a game that formerly had no Linux-native dedicated server, but now does..
- Update a game that formerly was not released, but now is released.
- Add a new game.
- Add a "mod" for an existing game.

The ability to setup a "mod" for an existing game, is *theoretical* and *untested* at this point

---

14 - The Game-Stencils Table

The **game-stencils** table is stored in the **data** sub-folder (typically **~/wgasm/data**). The default filename is "**game-stencils.tsv**". The format of the file is a **tab-separated-value** (**TSV**) text-file. The **game-stencils** table functions as a database of information about what **Stencils** are available to be "painted" onto a previously-installed game-server. The *order of each column is **critical***. The columns in this table are:

Order	Field-Name	Description
1	GameStencilID	ID for this **stencil**. Each must be **unique**. These must contain only *lower-case letters* (**a-z**), *numerals* (**0-9**) and if you like a *hyphen* (**-**)
2	GameTypeID	ID for the **game-type** this **stencil** is built for. This must match a **GameTypeID** defined in the **game-types** table.
3	StencilFileName	Filename of the zip file containing this stencils content. These must all be stored in the "Stencils" folder (typically **~/stencils**). Typically, they are named matching the **GameStencilID**.
4	Description	Plain-text stencil description.
5	Comment	Notes about this stencil (if any).

Each **stencil** is associated with **one** game-type. Typically, **stencils** are utilized to *standardize configurations* or apply *server-side modifications*. Effectively, **stencils** are a bunch of "canned" content zipped-up in a manner that facilitates it being unzipped directly into the appropriate game-servers "mod-sub-folder".

It is important to keep in mind that, when "painting" a **stencil** onto an existing game-server, any conflicting files *will be overwritten by the content in the stencil*.

For this reason, **stencils** are usually applied when *building a **new** game-server*, to setup a standard configuration, add some standard server-side mods. This provides an easier starting-point for getting a new game-server customized and ready for play.

For any **stencil** to be utilized *successfully*, two things must be true:

- The **.zip** with the properly zipped content must exist in the "Stencils" folder - *as defined in the central configuration file* (**config.txt**).
- A matching entry must exist in the **game-stencils** table (**~/weaselsscripts/data/game-stencils.tsv**) with the filename matching the actual **.zip** file.

Several example **stencils** are included with this system. The entries for them are already included in the default **game-stencils** table (**game-stencils.tsv**). However, their content is too larger to host in the GitHub repository. Consequently, the **.zip** files (and source files/folders used to create those), are contained in a archive (**stencils-latest.tar.gz**) stored in the related DropBox:

```
https://www.dropbox.com/scl/fo/qr0nxjefykegw3ix9p98s/AGHvx9t3sqoH_l88uR9FCso?
rlkey=l0h4kyqz3wbmdne4rbw0exc1r&st=0cyziuuy&dl=0
```

Example **stencils** included with this system are:

GameSencilD	GameTypeID	StencilFileName	Description	Comment
cs1-base	cs1	cs1-base.zip	[CS1] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
cs1-amx	cs1	cs1-amx.zip	[CS1] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
cs1-amx-bots	cs1	cs1-amx-bots.zip	[CS1] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus YaPB.
css-base	css	css-base.zip	[CSS] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
css-sm	css	css-sm.zip	[CSS] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
cs2-base	cs2	cs2-base.zip	[CS2] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dmc-base	dmc	dmc-base.zip	[DMC] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dmc-amx	dmc	dmc-amx.zip	[DMC] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
dmc-amx-bots	dmc	dmc-amx-bots.zip	[DMC] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus HPB_Bot.
dod-base	dod	dod-base.zip	[DoD] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dod-amx	dod	dod-amx.zip	[DoD] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
dod-amx-bots	dod	dod-amx-bots.zip	[DoD] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus Marine-Bot.
dods-base	dods	dods-base.zip	[DoDS] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
dods-sm	dods	dods-sm.zip	[DoDS] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
fof-base	fof	fof-base.zip	[FoF] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
fof-sm	fof	fof-sm.zip	[FoF] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
hl1-base	hl1	hl1-base.zip	[HL1] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
hl1-amx	hl1	hl1-amx.zip	[HL1] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
hl1-amx-bots	hl1	hl1-amx-bots.zip	[HL1] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus JKBotti.
tf2-base	tf2	tf2-base.zip	[TF2] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.
tf2-sm	tf2	tf2-sm.zip	[TF2] MetaMod:Source, SourceMod	MMS, SourceMod starter setup.
tfc-base	tfc	tfc-base.zip	[TFC] Starter Config	Minimum config with autoexec.cfg, server.cfg, etc.

tfc-amx	tfc	tfc-amx.zip	[TFC] MetaMod, AMX-Mod-X	Metamod, AMX-Mod-X starter setup.
tfc-amx-bots	tfc	tfc-amx-bots.zip	[TFC] MetaMod, AMX-Mod-X, Bots	Metamod, AMX-Mod-X plus FoxBot.

15 - The Game-Servers Table

The **game-servers** table is stored in the **data** sub-folder (typically **~/wgasm/data**). The default filename is "**game-servers.tsv**". The format of the file is a **tab-separated-value** (**TSV**) text-file. The **game-servers** table functions as a database of information about the **game-servers** that will be created by and managed by this system. The *order of each column is **critical***. The columns in this table are:

Order	Field-Name	Description
1	GameServerID	ID for game-server. Each must be **unique**. These must contain only *lower-case letters* (**a-z**), *numerals* (**0-9**) and if you like a *hyphen* ([/b]-[/b]). The GameServerID will be used for naming the Linux directory (folder) that the server will be installed into, as well as for naming disconnected / background GNU "screen" processes.
2	GameTypeID	ID for game-type for this game-server. This must match a GameTypeID defined in the **game-types** table.
3	PortNumber	Network port number for game-server. Each should be **unique**, to keep servers from interfering with each other. Obviously, these must *numerals* (**0-9**) only. Typically this will be something in the 27000-27999 range.
4	Verbose	Enable/disable *verbose* script output. This allows turning-on *verbose* logging for an *individual* server, rather than *globally* (for all servers).
5	Stale	Enable/disable monitoring for *stale* GNU "screen" logging. This should only be enabled if you have a **server-side modification** installed and configured that *ensures that output is still generated periodically to the console* (and consequently the GNU "screen" logging output (*even when the server is empty of players*). The example **stencils** for both AMX-Mod-X [www.amxmodx.org] and SourceMod [www.sourcemod.net] already include such a plug-in, with it configured to issue the "status" command on the game-server console once per minute.
6	Threshold	Threshold for stale-log monitoring (in seconds). If *stale* log monitoring is enabled, this controls how much inactivity is allowed before logging is considered *stale*.
7	Description	Plain-text game-server description.
8	Comment	Notes about this server (if any). This text is not used for any purpose, other than whatever notes you want of keep regarding each server.

Configuration information for several example **game-servers** is included in the default **game-servers** table, as follows:

GameServerID	GameTypeID	PortNumber	Verbose	Stale	Threshold	Description	Comment
server0hl1	hl1	27101	FALSE	FALSE	300	server0hl1 - Half-Life (HL1)	
server1dmc	dmc	27103	FALSE	FALSE	300	server1dmc - Deathmatch Classic (DMC)	
server2tfc	tfc	27105	FALSE	FALSE	300	server2tfc - Team Fortress Classic (TFC)	
server3tf2	tf2	27107	FALSE	FALSE	900	server3tf2 - Team Fortress 2 (TF2)	
server4cs1	cs1	27109	FALSE	FALSE	300	server4cs1 - Counter-Strike 1.6 (CS1)	
server5css	css	27111	FALSE	FALSE	900	server5css - Counter-Strike:Source (CSS)	

server6cs2	cs2	27113	FALSE	FALSE	900	server6cs2 - Counter-Strike 2 (CS2)	
server7dod	dod	27115	FALSE	FALSE	300	server7dod - Day of Defeat (DoD)	
server8dods	dods	27117	FALSE	FALSE	900	server8dods - Day of Defeat:Source (DoDS)	
server9fof	fof	27119	FALSE	FALSE	900	server9fof - Fistful of Frags (FoF)	

You may use whatever text editor you prefer - being sure to preserve the columns. Columns **must** be separated explicitly by **tabs** (*not just spaces*).

Also "*cutting-and-pasting*" between a text editor and a spreadsheet application may be used to assist with keeping the columns intact. Such applications might include:

- Google Sheets [en.wikipedia.org]
- LibreOffice Calc [en.wikipedia.org]
- Microsoft Excel [en.wikipedia.org])

It is *suggested* that you *add* your servers to the **game-servers** table, leaving the examples in the file for later reference.

16 - Components, 1 of 3

Several component scripts are largely meant to be used **one-time** for initial setup purposes. These have already been discussed or at least mentioned before this point. These included:

Component	Description
prepare-debian.sh	Unlike all other components of this system, this script **must** be run under the Linux "**root**" user or some similarly privileged ***sudo**-enabled* Linux user. Its purpose is to prepare the Linux operating system (presumably a "Debian" family distribution), by installing all the packages required by this system. In addition, it will also: • Offer to create the **game-servers** Linux user ,if it does not already exist, • Offer to install Webmin [webmin.com] , if it is not already installed. • Offer to install the wGASM [github.com] -related "Custom Commands" for Webmin [webmin.com] , • Offer to setup the **game-servers** Linux user as a Webmin [webmin.com] user. This script also features a "**nike mode**" ("Just do it.") which may be used to bypass (accept) all prompts, using the "**--nike**" command-line parameter.
install-wgasm.sh	Run under the **game-servers** Linux user, its purpose is to install all the other wGASM [github.com] system components. In addition, it will also: • Offer to install SteamCMD, if it is not already installed. • Offer to install the latest package of **Stencils**. This script also features a "**nike mode**" ("Just do it.") which may be used to bypass (accept) all prompts, using the "**--nike**" command-line parameter.
install-steamcmd.sh	Also run under the **game-servers** Linux user, its purpose is to ensure that SteamCMD is properly installed in the expected location (**~/steamcmd**).
install-stencils.sh	Also run under the **game-servers** Linux user, its purpose is to download the latest package of **Stencils**, and install them into the expected location (**~/stencils**).

17 - Components, 2 of 3

Several component scripts provide largely *diagnostic* or *support* functions, and *do not perform* any direct action on the **game-servers** themselves. These include:

Component	Purpose/Description

scripts-test.sh	Displays information gathered from *both* the central configuration file (**config.txt**), *and* the Linux operating system - including the computer's **local IP address**, and apparent **public IP address**. Useful for understanding if the configuration file is populated correctly, detecting missing critical files or folders, and understanding network connectivity (IP addressing) of the system.
list-running.sh	Displays running GNU screen [en.wikipedia.org] processes. Typically these are **game-servers** - running disconnected in the background.
list-network.sh	Displays utilization of the network interface. Basically, answers the (rare) question, "How busy is the network interface?"
list-ports.sh	Lists what network ports are in use on the computer (right now). Useful to ensure that a network port is not already in use before specifying it for a new **game-server**.
list-ports-game.sh	Lists what network ports are in use, by all **game-servers** (right now). Useful to check which network port(s) *currently running* **game-servers** are utilizing.
list-ports-hlds.sh	Same as above, but more specific to just **game-servers** built-upon the GoldSrc [en.wikipedia.org] engine. These **game-servers** all utilize the HLDS server system.
list-ports-source.sh	Same as above, but more specific to just **game-servers** built-upon the Source [en.wikipedia.org] engine. These **game-servers** all utilize the SrcDS server system.
game-types-list.sh	Lists all the **game-types** that are currently defined in the **game-types** table.
game-types-detail.sh	Displays details for a specific **GameTypeID**. This will also include listing any **stencils** that are built-for that specific **GameTypeID**, as well as listing any **game-servers** that have been setup based on that specific **GameTypeID**.
game-stencils-list-sh	Lists all the **stencils** that are defined in the **game-stencils** table.
game-stencils-detail.sh	Displays details for a specific **GameStencilID**. This will also include listing details of the relevant **GameTypeID**
game-server-list.sh	Lists all the **game-servers** that are defined in the **game-servers** table.
game-server-detail.sh	Displays details for a specific **GameServerID**. This will also include listing details of the **GameTypeID** associated with this **game-server**, as well as any **stencils** relevant to that **game-type**.
game-stencils-package.sh	Uses the content of the "source" sub-folder of the **stencils-folder**, to re-package the stencil **.zip** files.

18 - Components, 3 of 3

Several component scripts perform direct actions on the **game-servers** themselves. These include:

Component	Purpose/Description
game-server-install.sh	Installs a **game-server**, based on data from the **game-servers** table. This will: • Create a folder (named for the **GameServerID** specified), • Lookup the **GameTypeID** associated with that **game-server** • Using information about that **GameTypeID**, have SteamCMD to install the game into the folder. • If the game is already installed in that folder, it will basically update it (perhaps destructively).
game-server-exclude.sh	Builds backup exclusions, based on a **GameServerID**. This will: • Lookup the **GameTypeID** associated with that **game-server** • Using information about that **GameTypeID**, have SteamCMD to install a copy of that game *in a temporary folder*. • Build a "backup exclusion list" file, based on that temporary install - which will **exclude** default "stock" installation for that game from backups.

	• This makes backups a *tiny fraction* of what they might be otherwise, effectively backing-up only the *customization*, and not the "stock" content.
game-server-backup.sh	Backs-up a **game-server**, This requires that *both* an "include" list file *and* "exclude" list file exist in the **backup-configs** folder (typically **~/wgasm/backup-configs**).
game-server-restore.sh	Restores a **game-server** from backups, based on a **GameServerID**. This requires that the relevant backup .zip file exists in the **backups** folder (typically **~/backups**) and will unzip it into the default location all **game-servers** are branched-from.
game-server-paint.sh	Paints a specific **game-server** with a specific **stencil**. Any conflicting content *will be overwritten* by the content in the **stencil**.
game-server-run.sh	Runs a **game-server** *interactively* at the command-prompt (Linux shell). Generally requires entering "**quit**" and possibly **CTRL+C** to break out of the **game-server** to return to the command-prompt. However, doing so closes the **game-server**. This is useful for troubleshooting - especially for the first-time you setup a new game.
game-server-start.sh	Starts a **game-server** *disconnected in "background"*. Basically, *wraps* the **game-server-run.sh** component in GNU screen [en.wikipedia.org] to allow it to continue run as a disconnected background process - that will continue to operate when the shell is closed or user logs-off.
game-server-stop.sh	Stops a **game-server** if it is already started. This will: • Attempt to warn any players about the shutdown, with an in-game ***text message***. • Attempt to warn any players about the shutdown, with an in-game ***audible message***. • Sends commands to the GNU screen [en.wikipedia.org] that wraps the **game-server** - attempting to gracefully shut it down, by first sending the "**quit**" command, then **CTRL+C**, and then (as a last-resort) "**killing**" the Linux process.
game-server-update.sh	Updates a **game-server**, restarting if needed. This will: • Check if the **game-server** is already running. • If the **game-server** is already running, invoke the **game-server-stop.sh** component, to shut-down the **game-server** gracefully. • Invoke the **game-server-install.sh** component, to update the game content. • If the **game-server** was running before, invoke the **game-server-start.sh** component, to start the server back up again.
game-server-check.sh	Check for, and gracefully apply, any new updates for a **game-server**. This will: • Check to see if the existing **game-server** has the latest version of the game installed. • If the version installed does not match the version available on Steam (presumably newer), invoke the **game-server-update.sh** component.
game-server-monitor.sh	Monitors a **game-server**, restarting if needed. This will: • Check if the **game-server** is already running. • Check the GNU screen [en.wikipedia.org] output log, for several potentially unrecoverable (fatal) error conditions. • If any of those conditions is detected, invoke the **game-server-stop.sh** and then **game-server-start.sh** components. Tests that are performed against the GNU screen [en.wikipedia.org] output log include: • Detect "fatal error" • Detect "segmentation fault" • Detect that GNU screen [en.wikipedia.org] logging is *stale* (i.e. no recent output).
game-server-command.sh	Sends a command to a **game-server**'s console. This allows a user (from the Linux shell) to send a command to the **game-server**'s console - without manually connecting to it.

19 - Installing Servers

Requirements:

- Before game-servers may be installed, the details for it **must** be populated into the **game-servers** table. Before attempting to install a game-server, refer to the earlier section of this guide about the **game-servers** table to properly add the information for your new game-server into that table with some sort of text-editor.
- You should have some idea what **size** your game-server installation will be, to ensure you have enough drive-space to accommodate it. For example, CS2 can consume about **36-GB** of space. And you need to allow double-that to accommodate potential *temporary installs* for the "backup exclusion" process.

Using the **game-server-install.sh** component:

The installation of game-servers is performed with the **game-server-install.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it install. For example:

```
~/wgasm/game-server-install.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-install.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- Search through the rows of the **game-servers** table, looking for a record with the matching **GameServerID**.
- If no matching record is found, display and error message and exit.
- If a matching record is found, create a folder matching **GameServerID**.
- Use information for the record to install the game (using SteamCMD) into that folder.

If you want to install *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-install.sh gameserver1 gameserver2 gameserverN;
```

Of course, all **GameServerID**'s specified ***must*** *already be included* in the **game-servers** table. If any of them do not, the **game-server-install.sh** component will display and error message and stop (when it gets to trying to install that **GameServerID**).

Example of using the **game-server-install.sh** component:

Command to install the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-install.sh server0hl1;
```

Output of the example command:

```
game-servers@wGASM:~$ ~/wgasm/game-server-install.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-install.sh, at Thu Jun 13 03:38:39 PM PDT 2024            |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
/home/game-servers/wgasm/game-server-install.sh: line 154: /home/game-
servers/server0hl1/valve/install-in-progress.txt: No such file or directory
Full command-line to send to SteamCMD:
/home/game-servers/steamcmd/steamcmd.sh +force_install_dir /home/game-servers/server0hl1 +login
anonymous +app_set_config 90 mod valve +app_update 90  +quit
Redirecting stderr to '/home/game-servers/Steam/logs/stderr.txt'
Logging directory: '/home/game-servers/Steam/logs'
[  0%] Checking for available updates...
[----] Verifying installation...
UpdateUI: skip show logoSteam Console Client (c) Valve Corporation - version 1716584438
-- type 'quit' to exit --
Loading Steam API...OK
Connecting anonymously to Steam Public...OK
Waiting for client config...OK
Waiting for user info...OK
{NOTE: A bunch of sample output removed here for brevity}
Success! App '90' fully installed.
Size of game-server installation (including any mods):
```

```
900M    /home/game-servers/server0hl1/
+------------------------------------------------------------------------------+
| End: game-server-install.sh, at Thu Jun 13 03:39:29 PM PDT 2024 (50 secs)    |
+------------------------------------------------------------------------------+
game-servers@wGASM:~$
```

20 - Painting Servers with Stencils

Requirements / Warnings:

- Before game-servers may be painted, the details for the game-server **must** already be populated into the **game-servers** table, *and* the details for the **stencil must** already be populated into the **game-stencils** table, *and[/b] the **stencil's content (.zip file) must already exist in the stencils folder (typically ~/stencils).***
- When stencil is painted-onto (applied-to) a server, it will overwrite any conflicting files. Consequently, painting a game-server with a stencil is usually only done when performing the initial setup of a server - *before* any manual customization work is performed.

Using the **game-server-paint.sh** component:

The painting of stencils onto a game-server is performed with the **game-server-paint.sh** component. The **two (2)** command-line parameters that it **requires** are *both* the **GameServerID** of the game-server you wish to be painted, *and* the **GameStencilID** of the stencil you wish to paint with. For example:

```
~/wgasm/game-server-paint.sh SomeGameServerIDHere SomeGameStencilIDHere;
```

Example-1 of using the **game-server-paint.sh** component:

Command to paint the example Half-Life (HL1) game-server with the relevant "base" stencil:

```
~/wgasm/game-server-paint.sh server0hl1 hl1-base;
```

Output of the example:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-paint.sh server0hl1 hl1-base;
+------------------------------------------------------------------------------+
| Begin: game-server-paint.sh, at Thu Jun 13 11:23:32 AM PDT 2024              |
+------------------------------------------------------------------------------+
Game-server to 'paint' the stencil on: [ID=server0hl1]
Game-stencil to 'paint' with:          [ID=hl1-base]
Temp folder (for zipping process):     /home/game-servers/temp
Input File:                            /home/game-servers/stencils/hl1-base.zip
Paint Folder:                          /home/game-servers/server0hl1/valve

Stats/Details BEFORE painting:
Size of server install:                900M     /home/game-servers/server0hl1
Size of server game/mod folder:        516M     /home/game-servers/server0hl1/valve
Stats for server install:              127 directories, 5429 files
Stats for server game/mod folder:      81 directories, 4248 files

Paint command being used:
nice -n 19 7za x -mmt=off -aoa -o/home/game-servers/server0hl1/valve -w/home/game-servers/temp
/home/game-servers/stencils/hl1-base.zip

Painting the server with stencil ...

Stats/Details AFTER painting:
Size of server install:                934M     /home/game-servers/server0hl1
Size of server game/mod folder:        550M     /home/game-servers/server0hl1/valve
Stats for server install:              129 directories, 5465 files
Stats for server game/mod folder:      83 directories, 4284 files
```

```
+------------------------------------------------------------------------------+
| End: game-server-paint.sh, at Thu Jun 13 11:23:33 AM PDT 2024 (1 secs)       |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

Example-2 of using the **game-server-paint.sh** component:

Command to paint the example Half-Life (HL1) game-server with the relevant stencil that installs **AMX-Mod-X** and **bots**:

```
~/wgasm/game-server-paint.sh server0hl1 hl1-amx-bots;
```

Output of the example:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-paint.sh server0hl1 hl1-amx-bots;
+------------------------------------------------------------------------------+
| Begin: game-server-paint.sh, at Thu Jun 13 11:24:23 AM PDT 2024              |
+------------------------------------------------------------------------------+
Game-server to 'paint' the stencil on: [ID=server0hl1]
Game-stencil to 'paint' with:          [ID=hl1-amx-bots]
Temp folder (for zipping process):     /home/game-servers/temp
Input File:                            /home/game-servers/stencils/hl1-amx-bots.zip
Paint Folder:                          /home/game-servers/server0hl1/valve

Stats/Details BEFORE painting:
Size of server install:                934M     /home/game-servers/server0hl1
Size of server game/mod folder:        550M     /home/game-servers/server0hl1/valve
Stats for server install:              129 directories, 5465 files
Stats for server game/mod folder:      83 directories, 4284 files

Paint command being used:
nice -n 19 7za x -mmt=off -aoa -o/home/game-servers/server0hl1/valve -w/home/game-servers/temp
/home/game-servers/stencils/hl1-amx-bots.zip

Painting the server with stencil ...

Stats/Details AFTER painting:
Size of server install:                970M     /home/game-servers/server0hl1
Size of server game/mod folder:        586M     /home/game-servers/server0hl1/valve
Stats for server install:              148 directories, 5750 files
Stats for server game/mod folder:      102 directories, 4569 files
+------------------------------------------------------------------------------+
| End: game-server-paint.sh, at Thu Jun 13 11:24:24 AM PDT 2024 (1 secs)       |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

---

21 - Running Servers (Interactively)

Requirements / Notes:

- Before game-servers may be *run*, the details for it **must** be populated into the **game-servers** table. Before attempting to *run* a game-server, refer to the earlier section of this guide about the **game-servers** table to properly add the information for your new game-server into that table with some sort of text-editor.
- For a game-server to be *ran* it must already be *installed*. That should be obvious.
- Since this is an *interactive* process, to exit the running game-server process (which is effectively its own command-prompt), you may need to enter the "**quit**" command. Even doing after so, the game-server may attempt to restart itself (usually true of games built on the older HLDS system for GoldSrc [en.wikipedia.org] games). In which case you will have probably a 10-second window in which the press **CTRL+C** to break out of the game-server's command-prompt, and return to the Linux command-prompt ("shell").

Using the **game-server-run.sh** component:

The *running* of game-servers is performed with the **game-server-run.sh** component. The only command-line parameter that it requires (or accepts) a *single* **GameServerID** of the game-server you wish to have it run. For example:

```
~/wgasm/game-server-run.sh SomeGameServerIDHere;
```

Example of using the **game-server-run.sh** component:

Command to install the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-run.sh server0hl1;
```

Output of the command (and using "**quit**" and **CTRL+C** to return to the Linux shell):

```
game-servers@wGASM:~/wGASM$ ~/wgasm/game-server-run.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-run.sh, at Thu Jun 13 11:42:50 AM PDT 2024                |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Start-up command-line:
nice -n 10 ./hlds_run -game valve -secure -port 27101 +ip 192.168.187.96
Starting game server ...
nice -n 10 ./hlds_run -game valve -secure -port 27101 +ip 192.168.187.96
Auto-restarting the server on crash

Console initialized.
Using breakpad crash handler
Setting breakpad minidump AppID = 70
Forcing breakpad minidump interfaces to load
Looking up breakpad interfaces from steamclient
Calling BreakpadMiniDumpSystemInit
Protocol version 48
Exe version 1.1.2.2/Stdio (valve)
Exe build: 23:25:09 Dec  9 2023 (9907)
STEAM Auth Server
Server logging data to file logs/L0613000.log
L 06/13/2024 - 11:42:50: Log file started (file "logs/L0613000.log") (game "valve") (version
"48/1.1.2.2/Stdio/9907")
Server IP address 192.168.187.96:27101

   Metamod version 1.21p38 Copyright (c) 2001-2013 Will Day
     Patch: Metamod-P (mm-p) v38 Copyright (c) 2004-2018 Jussi Kivilinna
   Metamod comes with ABSOLUTELY NO WARRANTY; for details type `meta gpl'.
   This is free software, and you are welcome to redistribute it
   under certain conditions; type `meta gpl' for details.

L 06/13/2024 - 11:42:50: [META] Metamod v1.21p38  2018/02/11
L 06/13/2024 - 11:42:50: [META] by Will Day
L 06/13/2024 - 11:42:50: [META]    http://www.metamod.org/
L 06/13/2024 - 11:42:50: [META]  Patch: Metamod-P (mm-p) v38
L 06/13/2024 - 11:42:50: [META]  by Jussi Kivilinna
L 06/13/2024 - 11:42:50: [META]     http://metamod-p.sourceforge.net/
L 06/13/2024 - 11:42:50: [META] compiled: Feb 11 2018, 11:05:06 EET (optimized)
L 06/13/2024 - 11:42:50: [META] Recognized game 'valve'; Autodetection override; using dllfile
'hl.so'
L 06/13/2024 - 11:42:50: [META] Game DLL for 'hl.so (autodetect-override)' loaded successfully
L 06/13/2024 - 11:42:50: [META] ini: Begin reading plugins list: /home/game-
servers/server0hl1/valve/addons/metamod/plugins.ini
L 06/13/2024 - 11:42:50: [META] ini: Read plugin config for: <amxmodx_mm_i386.so>
L 06/13/2024 - 11:42:50: [META] ini: Read plugin config for: <jk_botti_mm_i386.so>
L 06/13/2024 - 11:42:50: [META] ini: Finished reading plugins list: /home/game-
servers/server0hl1/valve/addons/metamod/plugins.ini; Found 2 plugins to load
```

```
L 06/13/2024 - 11:42:50: [META] dll: Loading plugins...

   AMX Mod X version 1.8.2 Copyright (c) 2004-2006 AMX Mod X Development Team
   AMX Mod X comes with ABSOLUTELY NO WARRANTY; for details type `amxx gpl'.
   This is free software and you are welcome to redistribute it under
   certain conditions; type 'amxx gpl' for details.

L 06/13/2024 - 11:42:50: [META] dll: Loaded plugin 'AMX Mod X': AMX Mod X v1.8.2 Feb 14 2013, AMX
Mod X Dev Team
L 06/13/2024 - 11:42:50: [META] dll: Note: plugin '<jk_botti_mm_i386.so>' is using an older
interface version (5:7), not the latest interface version (5:13); there might be an updated version
of the plugin
JK_Botti: plugin attaching
L 06/13/2024 - 11:42:50: [JK_BOTTI] JK_Botti: plugin attaching
L 06/13/2024 - 11:42:50: Server cvar "jk_botti_version" = "1.42"
L 06/13/2024 - 11:42:50: [META] dll: Loaded plugin 'JK_Botti': JK_Botti v1.42 Jan  7 2009, Jussi
Kivilinna
L 06/13/2024 - 11:42:50: [META] dll: Finished loading 2 plugins
[jk_botti] Standard HL1DM assumed.
[jk_botti] Loading valve/addons/jk_botti/jk_botti_names.txt...
[jk_botti] Loading valve/addons/jk_botti/jk_botti_logo.cfg...
[jk_botti] Loading valve/addons/jk_botti/jk_botti_chat.txt...
L 06/13/2024 - 11:42:51: Log file closed
Server logging data to file logs/L0613001.log
{NOTE: A bunch of sample output removed here for brevity}
L 06/13/2024 - 11:42:51: Server cvar "amxmodx_version" = "1.8.2"
[jk_botti] Loading waypoint file: valve/addons/jk_botti/waypoints/gasworks.wpt
[jk_botti] - loaded: 1014 waypoints, 11156 paths
[jk_botti] [matrix load] - loading jk_botti waypoint path matrix
couldn't exec maps/gasworks_load.cfg
[AMXX] Loaded 1 admin from file
{NOTE: A bunch of sample output removed here for brevity}
Could not find steamerrorreporter binary. Any minidumps will be uploaded in-process[jk_botti] Added
total 0 map-object based waypoints and updated flags for 10!
[jk_botti] Executing valve/addons/jk_botti/jk_botti.cfg
L 06/13/2024 - 11:42:51: Started map "gasworks" (CRC "24041791")
{NOTE: A bunch of sample output removed here for brevity}
Connection to Steam servers successful.
   VAC secure mode is activated.
{NOTE: A bunch of sample output removed here for brevity}
quit
couldn't exec maps/gasworks_unload.cfg
hostname:  [Test] Half-Life (Hl1)
version :  48/1.1.2.2/Stdio 9907 secure  (70)
tcp/ip  :  192.168.187.96:27101
map     :  gasworks at: 0 x, 0 y, 0 z
players :  8 active (32 max)

#      name userid uniqueid frag time ping loss adr
# 1 "Nomad[bot]" 1 BOT   0 00:15    0    0
# 2 "R2D2[bot]" 2 BOT   0 00:15    0    0
# 3 "FemBot[bot]" 3 BOT   0 00:15    0    0
# 4 "Robby_The_Robot[bot]" 4 BOT   0 00:15    0    0
# 5 "Hymie[bot]" 5 BOT   0 00:15    0    0
# 6 "Lain_of_the_Wired[bot]" 6 BOT   0 00:15    0    0
# 7 "Project_2501[bot]" 7 BOT   0 00:15    0    0
# 8 "EVE[bot]" 8 BOT   0 00:15    0    0
8 users
{NOTE: A bunch of sample output removed here for brevity}
Reason:  Server shutting down
Server IP address 192.168.187.96:27101
L 06/13/2024 - 11:43:06: Server shutdown
L 06/13/2024 - 11:43:06: Log file closed
L 06/13/2024 - 11:43:06: Server shutdown
```

```
Thu Jun 13 11:43:06 AM PDT 2024: Server restart in 10 seconds
^C
game-servers@wGASM:~/wGASM$
```

---

22 - Starting Servers

Requirements / Notes:

- Before a game-server may be *started*, the details for it **must** be populated into the **game-servers** table.
- For a game-server to be *started* it must already be *installed*. That should be obvious.
- Going forward, we will differentiate *starting* a server from merely *running* a server. We will use the term *running* to refer to running it *interactively* directly in the Linux shell. Were as, we will use the term *starting* to refer to having the game-server operate as a *disconnected background* process - under GNU screen [en.wikipedia.org] .
- If you attempt to *start* a game-server that is already *started*, it will be effectively *restarted*.

Using the **game-server-start.sh** component:

The *starting* of game-servers is performed with the **game-server-start.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it *start*. For example:

```
~/wgasm/game-server-start.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-start.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- If the game-server matching that **GameServerID** is already *started*, invoke the **game-server-stop.sh** component to *stop* that game-server.
- Wrap the **game-server-run.sh** component in GNU screen [en.wikipedia.org] allowing it to operate as a *disconnected background* process.
- Assign a GNU screen [en.wikipedia.org] process name (based on the **GameServerID**) - which can be used to *reconnect* to it, send *commands* to it, or *stop* it later.

If you want to start *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-start.sh gameserver1 gameserver2 gameserverN;
```

Of course, all **GameServerID**'s specified ***must*** *already be included* in the **game-servers** table. If any of them do not, the **game-server-start.sh** component will display and error message and stop (when it gets to trying to start that **GameServerID**).

Example of using the **game-server-start.sh** component:

Command to *start* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-install.sh server0hl1;
```

Output of the command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-start.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-start.sh, at Thu Jun 13 12:17:40 PM PDT 2024              |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
No background instance of this game server detected.
Starting game with the 'screen' utility ...
Displaying list of 'screen' processes, AFTER start attempt ...
There is a screen on:
        71982.server0hl1 (06/13/2024 12:17:40 PM) (Detached)
1 Socket in /run/screen/S-game-servers.
+------------------------------------------------------------------------------+
| End: game-server-start.sh, at Thu Jun 13 12:17:40 PM PDT 2024 (0 secs)       |
```

```
+------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

---

23 - Stopping Servers

Requirements / Notes:

- Before a game-server may be *stop*, the details for it **must** be populated into the **game-servers** table.
- Going forward, we will differentiate a server that is *started* from a server that is merely *running*. We will use the term *running* to refer to it being running *interactively* directly in a Linux shell. Were as, we will use the term *started* to refer to having the game-server operating as a *disconnected background* process - under GNU screen [en.wikipedia.org] .
- If you attempt to *stop* a game-server that is **not** already *started*, it will attempt to stop it anyway - but no harm done.

Using the **game-server-stop.sh** component:

The *stopping* of game-servers is performed with the **game-server-stop.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it *start*. For example:

```
~/wgasm/game-server-stop.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-stop.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- Connect to the GNU screen [en.wikipedia.org] matching that **GameServerID**.
- Send a command to that process. to display a text message to any players - warning about the shut-down.
- Send a command to that process, to play and audible message to any players - warning about the shut-down.
- Wait a few seconds, to allow time for the audible message to play.
- Send the "**quit**" command to that process.
- Wait a few seconds, and then send **CTRL+C** to that process.
- Wait a few more seconds, and then use the Linux "**kill**" command to terminate the process (if it is still running at that point).

If you want to stop *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-stop.sh gameserver1 gameserver2 gameserverN;
```

Example of using the **game-server-stop.sh** component:

Command to *stop* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-stop.sh server0hl1;
```

Output of the example command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-stop.sh server0hl1;
+------------------------------------------------------------------------+
| Begin: game-server-stop.sh, at Thu Jun 13 01:04:50 PM PDT 2024         |
+------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Displaying in-game notification ...
Playing in-game audible alert ...
Allowing 15-sec for alert to play ...
Sending the 'quit' command ...
Allowing 3-sec for graceful exit ...
Sending the CTRL+C to game server ...
Allowing 1-sec before resorting to Linux 'kill' command ...
Using 'kill' command on game server process ...
```

```
Displaying of 'screen' processes, AFTER stop attempt ...
No Sockets found in /run/screen/S-game-servers.
+------------------------------------------------------------------------------+
| End: game-server-stop.sh, at Thu Jun 13 01:05:09 PM PDT 2024 (19 secs)       |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

24 - Sending Commands to Servers

Requirements / Notes:

- Before you can send commands to a game-server, the details for it **must** be populated into the **game-servers** table.
- Before you can send commands to a game-server, it must already be *started*.
- We differentiate a server that is *started* from a server that is merely *running*. We use the term *running* to refer to it being running *interactively* directly in a Linux shell. Were as, we use the term *started* to refer to having the game-server operating as a *disconnected background* process - under GNU screen [en.wikipedia.org] .

Using the **game-server-command.sh** component:

Sending commands to game-servers is performed with the **game-server-command.sh** component. The command-line parameters that it requires are a *single* **GameServerID** of the game-server you wish to have it run, followed by whatever command(s) should be sent to that game-server's "console". Generally, it is best to enclose the command(s) in double-quotes to ensure it gets passed correctly to the game-server process For example:

```
~/wgasm/game-server-command.sh SomeGameServerIDHere "Some commands here";
```

Example of using the **game-server-command.sh** component:

Command to install the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-command.sh server0hl1 "say This is a TEST!";
```

Output of the example command:

```
game-servers@wGASM:~$ ~/wgasm/game-server-command.sh server0hl1 "say This is a TEST!";
+------------------------------------------------------------------------------+
| Begin: game-server-command.sh, at Thu Jun 13 03:24:40 PM PDT 2024            |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Command to send: ...
say This is a TEST!
Sending command ...
+------------------------------------------------------------------------------+
| End: game-server-command.sh, at Thu Jun 13 03:24:40 PM PDT 2024 (0 secs)     |
+------------------------------------------------------------------------------+
game-servers@wGASM:~$ screen -r server0hl1
```

25 - Generating Backup Exclusion Lists

About backup exclusion lists:

Essentially **backup exclusion lists** are lists of all the default "stock" content for a given game. These lists are then used by the **backup-game-server.sh** component to *exclude* all the game's default "stock" content from the backup process. This makes the backup process both *faster* and the resultant backup files *smaller* by only backing-up the customized content, such as updated configuration files, any custom maps added, any custom server-side mods, etc. The difference between backing an entire game-server folder, compared to just backing-up the customization can be huge. The table below shows some real-world examples:

Game	Full server installation	Just the mod sub-folder	Just customization
Half-Life (moderately customized)	1.4-GB	973-MB	135-MB
Counter-Strike 1.6 (moderately customized)	2.5-GB	1.8-GB	250-MB
Team Fortress 2 (heavily customized)	14-GB	14-GB	783-MB
Counter-Strike 2 (minimally customized)	36-GB	33-GB	222-MB

Requirements / Notes:

- Before a backup exclusion list for a game-server may be generated, the details for it **must** be populated into the **game-servers** table.
- The *downside* to using backup exclusion lists, is that you need as much ***spare*** *drive space*, as the **largest game** you intend to host. So, as an example, if you planned to host a Counter-Strike 2 (CS2) game-server, you would need an *extra* **36+GB** available - *beyond* what your CS2 game-server itself is consuming.
- While generation of backup exclusion lists for some older games (such as HL1 or CS1) may complete very quickly, other more modern names (such as TF2 or especially CS2) may take a very long time to complete.

Using the **game-server-exclude.sh** component:

Generating backup exclusion lists for game-servers is performed with the **game-server-exclude.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it generate a backup exclusion list for. For example:

```
~/wgasm/game-server-exclude.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-exclude.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- Install another **copy** of the *same game* associated with that game-server, in a ***seperate*** *temporary folder*.
- Generate a backup exclusion list, based on the default "stock" content in that temporary folder.
- Place the newly-generated backup exclusion list in the **backup-configs** folder (typically **~/wgasm/backup-configs**) - named for the **GameServerID** provided.
- Delete that temporary folder to free-up disk space.

If you want to build backup exclusion lists for *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-exclude.sh gameserver1 gameserver2 gameserverN;
```

Of course, all **GameServerID**'s specified ***must*** *already be included* in the **game-servers** table. If any of them do not, the **game-server-exclude.sh** component will display and error message and stop (when it gets to trying to generate a backup exclusion list for that **GameServerID**).

Example of using the **game-server-exclude.sh** component:

Command to generate a backup exclusion list for the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-exclude.sh server0hl1;
```

Output of the command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-exclude.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-exclude.sh, at Thu Jun 13 05:50:37 PM PDT 2024            |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
```

```
Full command-line to send to SteamCMD:
/home/game-servers/steamcmd/steamcmd.sh +force_install_dir /home/game-servers/temp/temp-
install/stock-hl1 +login anonymous +app_config 90 mod valve +app_update 90  +quit
Redirecting stderr to '/home/game-servers/Steam/logs/stderr.txt'
Logging directory: '/home/game-servers/Steam/logs'
[  0%] Checking for available updates...
[----] Verifying installation...
Steam Console Client (c) Valve Corporation - version 1718305764
-- type 'quit' to exit --
Loading Steam API...OK
Connecting anonymously to Steam Public...OK
Waiting for client config...OK
Waiting for user info...OK
 Update state (0x3) reconfiguring, progress: 0.00 (0 / 0)
 Update state (0x61) downloading, progress: 0.01 (50486 / 937135369)
{NOTE: A bunch of sample output removed here for brevity}
 Update state (0x81) verifying update, progress: 69.99 (655902909 / 937135369)
 Update state (0x81) verifying update, progress: 80.71 (756394254 / 937135369)
Success! App '90' fully installed.
Size of temporary installation:
900M    /home/game-servers/temp/temp-install/stock-hl1/
Generating an updated exclude file ...
Purging the temporary game installation ...
+------------------------------------------------------------------------------+
| End: game-server-exclude.sh, at Thu Jun 13 05:51:35 PM PDT 2024 (58 secs)    |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

26 - Backing-Up Servers

Requirements / Notes:

- Before a game-server may be *backed-up*, *both* backup **include** *and* exclude **exclude** files for that game-server must exist in the **backup-configs** folder (typically **~/wgasm/backup-configs**).
- The backup **include** must be named with the **GameServerID** followed by "-include.txt".
- The backup **exclude** must be named with the **GameServerID** followed by "-exclude.txt".
- All paths specified in both the *include* and *exclude* lists are relative to the game-server's installation folder.
- The backup **exclude** is typically generated by the **game-server-exclude.sh** component. If there is no system-generated backup exclusion list, then a file (even if *blank*) must at **exist**.
- The backup **include** should include the list of file/folder paths (relative to the game-server's install folder) to be included in the backup.
- If a thorough *backup exclusion list* has been generated, this *could* be a simple as just the game's **mod-sub-folder**.
- If a thorough *backup exclusion list* is NOT being used, this *should* be more detailed. Including things such as:
  - Any customized configuration files.
  - Any custom sound or music files.
  - Any custom map files.
  - Any additional files that the custom maps require (models, sounds, etc.).
  - Folders for any server-side modifications (e.g. **addons/**)
- For a game-server to be *backed-up* it must already be *installed*. That should be obvious.
- It is recommended that you copy or replicate your **backup-folder** off the computer or synchronize them to a Cloud service. So that, in the event of some kind of catastrophic system failure, you will have copies of all your backups to use for rebuilding everything.

Using the **game-server-backups.sh** component:

The *backing-up* of game-servers is performed with the **game-server-backup.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to *back-up*. For example:

```
~/wgasm/game-server-backup.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-start.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- Use **p7zip** (a Linux implementation of 7-Zip [en.wikipedia.org] to create an archive file.
- Use the backup **include** list to instruct **p7zip** which files/folders to **include** in the backup.
- Use the backup **exclude** list to instruct **p7zip** which files/folders to **exclude** from the backup.
- The resultant archive (.zip) file, will be place into the **backups-folder** (typically **~/backups**);

If you want to back-up *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-backup.sh gameserver1 gameserver2 gameserverN;
```

Of course, all **GameServerID**'s specified ***must*** *already be included* in the **game-servers** table. If any of them do not, the **game-server-backup.sh** component will display and error message and stop (when it gets to trying to start that **GameServerID**).

Example of using the **game-server-backup.sh** component:

Command to *back-up* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-backup.sh server0hl1;
```

Output of the command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-backup.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-backup.sh, at Thu Jun 13 06:28:37 PM PDT 2024             |
+------------------------------------------------------------------------------+
Game-server (or backup config) to backup:  [ID=server0hl1]
Listing any previous backups ...
Deleting any previous backup ...
Creating a new backup ...
Backup command being used:
nice -n 19 7za a -tzip -mmt=off -snl -w/home/game-servers/temp /home/game-
servers/backups/server0hl1-backup.zip @/home/game-servers/wgasm/backup-configs/server0hl1-
include.txt -xr@/home/game-servers/wgasm/backup-configs/server0hl1-exclude.txt
 ... should be done. Listing backups ...
-rw-rw-r-- 1 194K Jun 13 18:28 /home/game-servers/backups/server0hl1-backup.zip
+------------------------------------------------------------------------------+
| End: game-server-backup.sh, at Thu Jun 13 06:28:37 PM PDT 2024 (0 secs)      |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

---

27 - Restoring Servers from Backups

Requirements / Notes:

- Before game-servers may be **restored**, the details for it **must** be populated into the **game-servers** table.
- The backup archive (.zip) file must be located in the **backup-folder** (typically **~/backups**).
- The backup contents will be restored to a sub-folder named for the **GameServerID**, branched from the default server install folder (typically **~/**).

Using the **game-server-restore.sh** component:

The *restore* of game-servers from backups, is performed with the **game-server-restore.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to *restore* from backups. For example:

```
~/wgasm/game-server-restore.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-restore.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- Restore the contents to a sub-folder named for the **GameServerID**, branched from the default server install folder (typically *~/*).

If you want to restore *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-restore.sh gameserver1 gameserver2 gameserverN;
```

Of course, all **GameServerID**'s specified ***must*** *already be included* in the **game-servers** table. If any of them do not, the **game-server-restore.sh** component will display and error message and stop (when it gets to trying to restore that **GameServerID**).

Example of using the **game-server-restore.sh** component:

Command to *restore* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-restore.sh server0hl1;
```

Output of the command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-restore.sh server0hl1;
+-------------------------------------------------------------------------+
| Begin: game-server-restore.sh, at Thu Jun 13 08:41:00 PM PDT 2024       |
+-------------------------------------------------------------------------+
Game-server (or backup config) to restore: [ID=server0hl1]
List any matching backups available ...
-rw-rw-r-- 1 194K Jun 13 18:29 /home/game-servers/backups/server0hl1-backup.zip
Restoring the backup ...
Restore command being used:
nice -n 19 7za x -mmt=off -aoa -o/home/game-servers -w/home/game-servers/temp /home/game-
servers/backups/server0hl1-backup.zip
 Listing restored folder (listing limited to 2 levels deep) ...
/home/game-servers/server0hl1
{NOTE: A bunch of sample output removed here for brevity}
├── valve
{NOTE: A bunch of sample output removed here for brevity}
33 directories, 114 files
+-------------------------------------------------------------------------+
| End: game-server-restore.sh, at Thu Jun 13 08:41:00 PM PDT 2024 (0 secs)|
+-------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

---

28 - Updating Servers

Requirements / Notes:

- Before a game-server may be *updated*, the details for it **must** be populated into the **game-servers** table.

Using the **game-server-update.sh** component:

The *updating* of game-servers is performed with the **game-server-update.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it *update*. For example:

```
~/wgasm/game-server-update.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-update.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- If the game-server is started, use the **game-server-command.sh** component to display a text message to any players - warning about the update.
- If the game-server is started, use the **game-server-command.sh** component to play and audible message to any players - warning about the update.
- If the game-server is started, Wait a one minute.
- If the game-server is started, invoke the **game-server-stop.sh** component on that server.
- Invoke the **game-server-install.sh** component to update that server.
- If the game-server was started before, invoke the **game-server-stop.sh** to start the server again.

If you want to update *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-update.sh gameserver1 gameserver2 gameserverN;
```

Example of using the **game-server-update.sh** component:

Command to *update* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-update.sh server0hl1;
```

Output of the example command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-update.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-update.sh, at Fri Jun 14 10:16:53 AM PDT 2024             |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
No background instance of this game server detected.
Updating content of this game-server ...
+------------------------------------------------------------------------------+
| Begin: game-server-install.sh, at Fri Jun 14 10:16:53 AM PDT 2024            |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Full command-line to send to SteamCMD:
/home/game-servers/steamcmd/steamcmd.sh +force_install_dir /home/game-servers/server0hl1 +login
anonymous +app_set_config 90 mod valve +app_update 90  +quit
Redirecting stderr to '/home/game-servers/Steam/logs/stderr.txt'
Logging directory: '/home/game-servers/Steam/logs'
[  0%] Checking for available updates...
[----] Verifying installation...
Steam Console Client (c) Valve Corporation - version 1718305764
-- type 'quit' to exit --
Loading Steam API...OK
Connecting anonymously to Steam Public...OK
Waiting for client config...OK
Waiting for user info...OK
Success! App '90' already up to date.
Size of game-server installation (including any mods):
900M    /home/game-servers/server0hl1/
+------------------------------------------------------------------------------+
| End: game-server-install.sh, at Fri Jun 14 10:16:57 AM PDT 2024 (4 secs)     |
+------------------------------------------------------------------------------+
+------------------------------------------------------------------------------+
| End: game-server-update.sh, at Fri Jun 14 10:16:57 AM PDT 2024 (4 secs)      |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

29 - Checking Servers (for Updates)

Requirements / Notes:

- Before a game-server may be *checked*, the details for it **must** be populated into the **game-servers** table.
- For a game-server to be *checked* it must already be *installed*. That should be obvious.
- For the *check* function to be truly useful, it should be scheduled to occur regularly for each server being hosted.
- This is best performed by adding various **game-server-check.sh** commands to the related **cron** scripts - and having those scripts added to the Linux cron [en.wikipedia.org] scheduler.
- For the *check* function, it is best scheduled *hourly* - to minimize server down-time.

Using the **game-server-check.sh** component:

The *checking* of game-servers is performed with the **game-server-check.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it *check*. For example:

```
~/wgasm/game-server-check.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-check.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- Compare the version of the game installed, to the version available in Steam.
- If the versions are the same, *do not do anything else, just exit*.
- If the versions are different, assume an update is needed and *proceed as follows*.
- Invoke the **game-server-backup.sh** component, to make a backup of the game-server first.
- Invoke the **game-server-update.sh** component, to apply the latest update to the game-server.
- Invoke the **game-server-exclude.sh** component, to regenerate the backup exclusion list - since the content of the game has changed.

If you want to check *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-check.sh gameserver1 gameserver2 gameserverN;
```

Example of using the **game-server-check.sh** component:

Command to *check* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-check.sh server0hl1;
```

Output of the example command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-check.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-check.sh, at Fri Jun 14 10:30:32 AM PDT 2024              |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
URL to use for update-check:
http://api.steampowered.com/ISteamApps/UpToDateCheck/v1?appid=70&version=1.1.2.2&format=xml
Checking the current version against the update-check URL now ...
Evaluating the SteamAPI update check output ...
[X] SteamAPI update check was successful.
[X] Server installation is already up-to-date.
Server already up-to-date, NOT attempting update.
+------------------------------------------------------------------------------+
| End: game-server-check.sh, at Fri Jun 14 10:30:32 AM PDT 2024 (0 secs)       |
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

---

30 - Monitoring Servers (for Issues)

Requirements / Notes:

- Before a game-server may be *monitored*, the details for it **must** be populated into the **game-servers** table.
- For a game-server to be *monitored* it must already be *started*. That should be obvious.
- For the *monitor* function to be truly useful, it should be scheduled to occur regularly for each server being hosted.
- This is best performed by adding various **game-server-monitor.sh** commands to the related **cron** scripts - and having those scripts added to the Linux cron [en.wikipedia.org] scheduler.
- For the *monitor* function, it is best scheduled at least *hourly*, or better more *often* - to minimize server down-time.
- The two most common errors detected are the very generic *fatal error* or *segmentation fault* conditions.
- Unfortunately, sometimes game-servers just *stop working / responding, without actually displaying any error message* on the console, but also do not properly *self-terminate*. When this happens, they are still technically *operating* (i.e. the Linux *process* continues), but they are not doing anything useful- and there is no further console activity.
- The *stale log* monitoring feature can detect this, by examining the **GNU screen** logs validate they show activity in the last specified *threshold*.
- However, an empty server with no players and no bots (or *hibernating* bots) may also result in a **false-positive *stale-log* condition**. Consequently, you should only enable the *stale-log* monitoring, if you have a server-side modification loaded the ensures that there is always some kind of console activity - at least once per minute.
- The example **stencils** that provide **AMX_Mod-X** or **SourceMod** include such a plug-in.

Using the **game-server-monitor.sh** component:

The *monitoring* of game-servers is performed with the **game-server-monitor.sh** component. The only command-line parameter that it requires (or accepts) is the **GameServerID** of the game-server you wish to have it *monitor*. For example:

```
~/wgasm/game-server-monitor.sh SomeGameServerIDHere;
```

When properly invoked, the **game-server-monitor.sh** component, will:

- If no **GameServerID** is passed as a parameter, display and error message and exit.
- If the server is not *started*, assume it was shut-down intentionally, and exit.
- Access the GNU screen [en.wikipedia.org] output log for the **GameServerID** specified.
- Detect if the log ends with a *fatal error* message.
- Detect if the log ends with a *segmentation fault* message.
- If *stale-log* monitoring is enabled for this server, detect if the age of the log exceeds the specified *threshold*.
- If any of those conditions are detected, invoke the **game-server-start.sh** component.

If you want to monitor *multiple servers* as part of the same command, you may specify more than one **GameServerID** as command-line parameters, separated by **spaces**. For example:

```
~/wgasm/game-server-monitor.sh gameserver1 gameserver2 gameserverN;
```

Example-1 of using the **game-server-monitor.sh** component:

Command to *monitor* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-monitor.sh server0hl1;
```

Output of the example command:

```
game-servers@wGASM:~/wgasm$ screen -r server0hl1
[detached from 139371.server0hl1]
game-servers@Desktop3:~/wgasm$ ~/wgasm/game-server-monitor.sh server0hl1;
+------------------------------------------------------------------------------+
| Begin: game-server-monitor.sh, at Fri Jun 14 11:17:43 AM PDT 2024            |
+------------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Is related screen process running?:    true
Monitor for stale 'screen' logging?:   false
Most recent 'screen' log actvity:      2 seconds ago.
+------------------------------------------------------------------------------+
```

```
| End: game-server-monitor.sh, at Fri Jun 14 11:17:43 AM PDT 2024 (0 secs)    |
+-----------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

Example-2 of using the **game-server-monitor.sh** component:

Command to *monitor* the example Half-Life (HL1) game-server:

```
~/wgasm/game-server-monitor.sh server0hl1;
```

Output of the example command:

```
game-servers@wGASM:~/wgasm$ ~/wgasm/game-server-monitor.sh server0hl1;
+-----------------------------------------------------------------------------+
| Begin: game-server-monitor.sh, at Fri Jun 14 11:26:03 AM PDT 2024           |
+-----------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Is related screen process running?:    true
Monitor for stale 'screen' logging?:    true
Most recent 'screen' log actvity:      503 seconds ago.
__        __              _
\ \      / /_ _ _ __ _ __ (_)_ __   __ _ _
 \ \ /\ / / _` | '__| '_ \| | '_ \ / _` (_)
  \ V  V / (_| | |  | | | | | | | (_| |_
   \_/\_/ \__,_|_|  |_| |_|_|_| |_|\__, (_)
                                   |___/
Stale 'screen' log-file detected!
Restarting this game server ...
+-----------------------------------------------------------------------------+
| Begin: game-server-start.sh, at Fri Jun 14 11:26:04 AM PDT 2024             |
+-----------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Stopping background instance of this game server ...
+-----------------------------------------------------------------------------+
| Begin: game-server-stop.sh, at Fri Jun 14 11:26:04 AM PDT 2024              |
+-----------------------------------------------------------------------------+
Game-server selected: [ID=server0hl1] server0hl1 - Half-Life (HL1)
Displaying in-game notification ...
Playing in-game audible alert ...
Allowing 15-sec for alert to play ...
Sending the 'quit' command ...
Allowing 3-sec for graceful exit ...
Sending the CTRL+C to game server ...
Allowing 1-sec before resorting to Linux 'kill' command ...
Using 'kill' command on game server process ...
Displaying of 'screen' processes, AFTER stop attempt ...
No Sockets found in /run/screen/S-game-servers.
+-----------------------------------------------------------------------------+
| End: game-server-stop.sh, at Fri Jun 14 11:26:23 AM PDT 2024 (19 secs)      |
+-----------------------------------------------------------------------------+
Stopped background instance of this game server.
Starting game with the 'screen' utility ...
Displaying list of 'screen' processes, AFTER start attempt ...
There is a screen on:
        140637.server0hl1       (06/14/2024 11:26:23 AM)        (Detached)
1 Socket in /run/screen/S-game-servers.
+-----------------------------------------------------------------------------+
| End: game-server-start.sh, at Fri Jun 14 11:26:23 AM PDT 2024 (19 secs)     |
+-----------------------------------------------------------------------------+
+-----------------------------------------------------------------------------+
| End: game-server-monitor.sh, at Fri Jun 14 11:26:23 AM PDT 2024 (20 secs)   |
```

```
+------------------------------------------------------------------------------+
game-servers@wGASM:~/wgasm$
```

---

31 - Webmin

Requirements / Notes:

- Webmin [webmin.com] is a web-based management interface for Linux, that provides an easy way to administer a Linux installation using a web-browser.
- All the following examples assume that you are using Google Chrome [en.wikipedia.org] as your web-browser. If you are using some other web-browser, everything should still work as expected, but error messages and some other aspects of the screens may look a bit different.
- If you allowed the **prepare-debian.sh** script to install Webmin [webmin.com] for you, *and* allowed it to install the related "Custom Commands" they should already be available. However, they should really only be operated while logged-in as the "**game-servers**" Linux user.

Determining IP Address:

You will need to know the **IP Address** of your server to access Webmin [webmin.com] . If you are renting a VPS from a hosting provider, this is the same **IP address** they provided to use for SSH access.

Alternatively, if you installed Debian [www.debian.org] yourself on a physical computer or virtual machine, you can determine its current **IP address** by logging-into the console as **root**, and entering the following command:

```
ifconfig;
```

That command will respond with output somewhat like this:

```
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
 --> inet 192.168.187.101  netmask 255.255.255.0  broadcast 192.168.187.255
        inet6 fe80::a00:27ff:fe88:d8fe  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:88:d8:fe  txqueuelen 1000  (Ethernet)
        RX packets 1311293  bytes 435409763 (415.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 815177  bytes 194294624 (185.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

The **IP address** will show in a similar location in the output of the **ifconfig** command on your Debian installation.

If you did not (yet) run the **prepare-debian.sh** script, you should really go do that, or *most things in this guide are not going to work*. However, if while trying to use the **ifconfig** command, it says that it is not installed or available, you may need to install the **net-tools** package manually. You can install the **net-tools** package using this command:

```
apt-get install -y net-tools;
```

Accessing Webmin:

If you installed Webmin [webmin.com] *manually* yourself, then it should be reachable with a web-browser on another computer using the local IP address of your Linux computer (or virtual machine), on the *default* port number **10000**. For example, if the local IP address of your Linux computer were **w.x.y.z**, the URL to enter in your web-browser would be:

```
https://w.x.y.z:10000
```

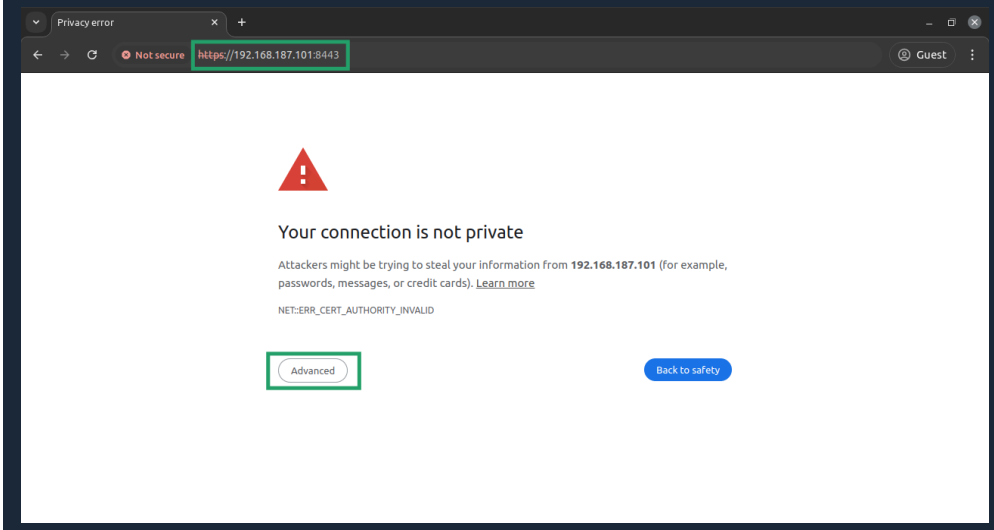
If you allowed the **prepare-debian.sh** script to install Webmin [webmin.com] for you, then then it should be reachable with a web-browser on another computer using the local IP address of your Linux computer, on the *alternative secure-web* port number **8443**. For example, if the local IP address of your Linux computer were **w.x.y.z**, the URL to enter in your web-browser would be:

```
https://w.x.y.z:8443
```

From here forward, all examples assume that you allowed the **prepare-debian.sh** script to install Webmin [webmin.com] for you, and is consequently using the *alternative secure-web* port number **8443**,

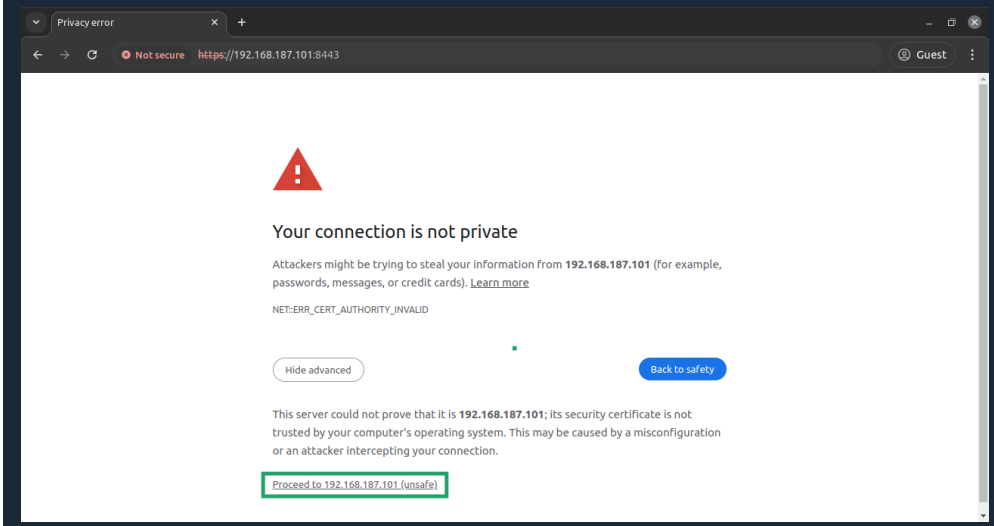
Logging-into Webmin:

Your web-browser will probably complain that the web-site you are accessing is not "secure". This is expected behavior since Webmin [webmin.com] does not have a "real" SSL certificate installed. Since this is essentially your own web-site, you can safely ignore this error, and safely proceed to Webmin [webmin.com] by clicking the **Advanced** button.



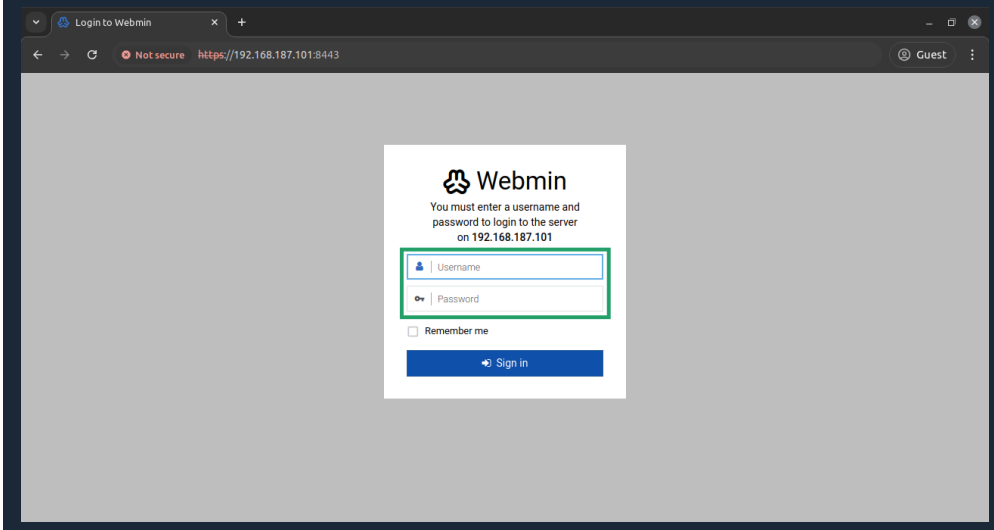
.

On the next screen, it will likely warn *again* about the web-site not having a "real" SSL certificate installed. you can safely click the **proceed** option.



.

You should then be presented with the login screen for Webmin [webmin.com] :



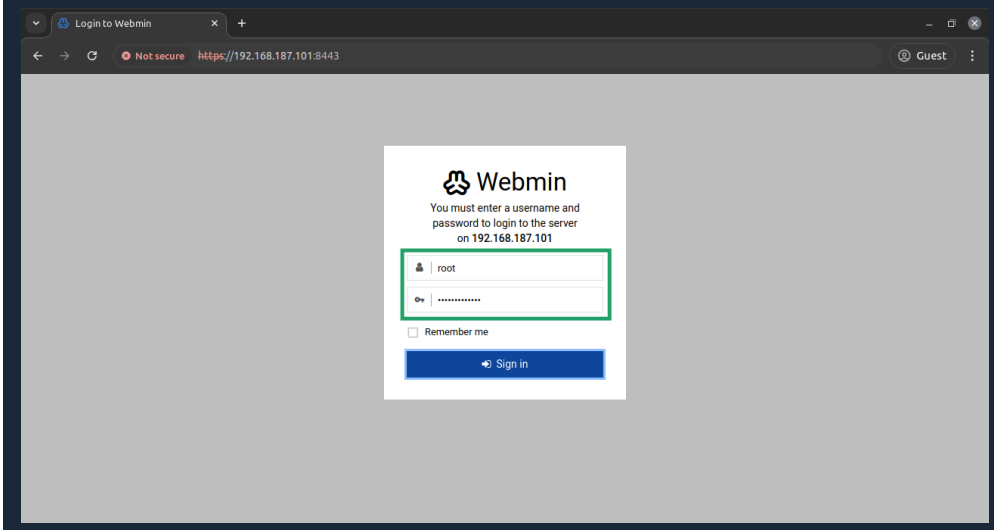
.

Depending on which Linux user you log-into Webmin [webmin.com] with, you may be presented with different menu options.

Logging-in as **root**:

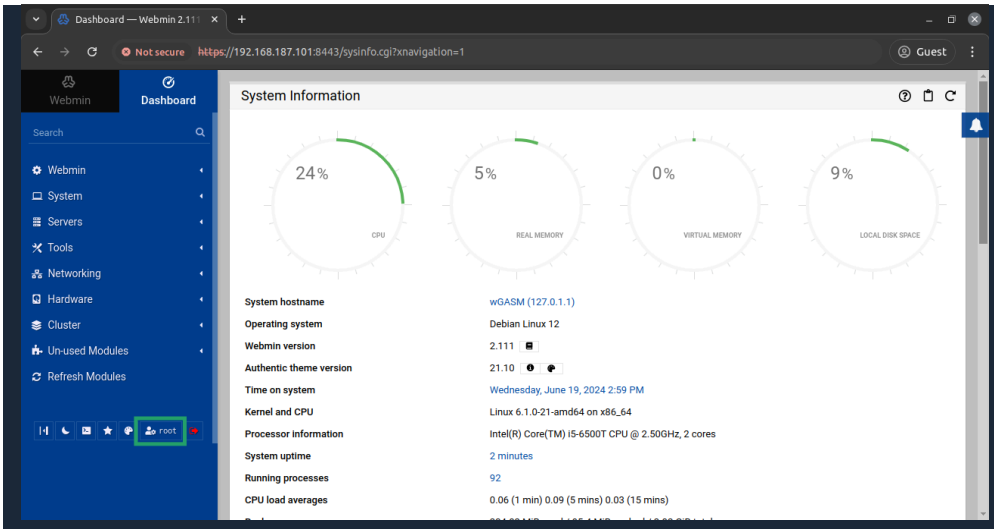
Logging-in with either **root** or another *privileged* ***sudo****-enabled* user (either of which are effectively Linux system administrators), is useful for tasks such as:

- Setting-up scheduled maintenance tasks, using Cron [en.wikipedia.org] jobs.
- Adding and/or updating system packages.
- Setting the clock / time-zone settings for the system.
- Configuring Linux firewall features, such as IP Tables [en.wikipedia.org] .
- Possibly, setting-up an Apache [en.wikipedia.org] web-server, to enable some game's FastDL feature.



.

If you log-in with either **root** or another *privileged* ***sudo****-enabled* user, you will see menu items for **all** installed modules - effectively allowing you administrative access to the entire Linux operating system.

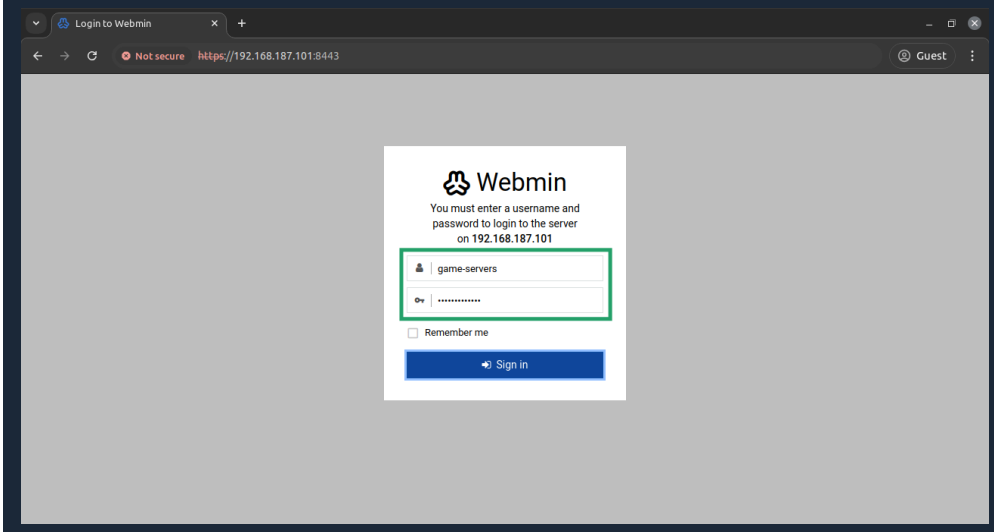


.
Note that at the bottom of the menus in the left-side panel, it will show which user you are currently logged-in with.

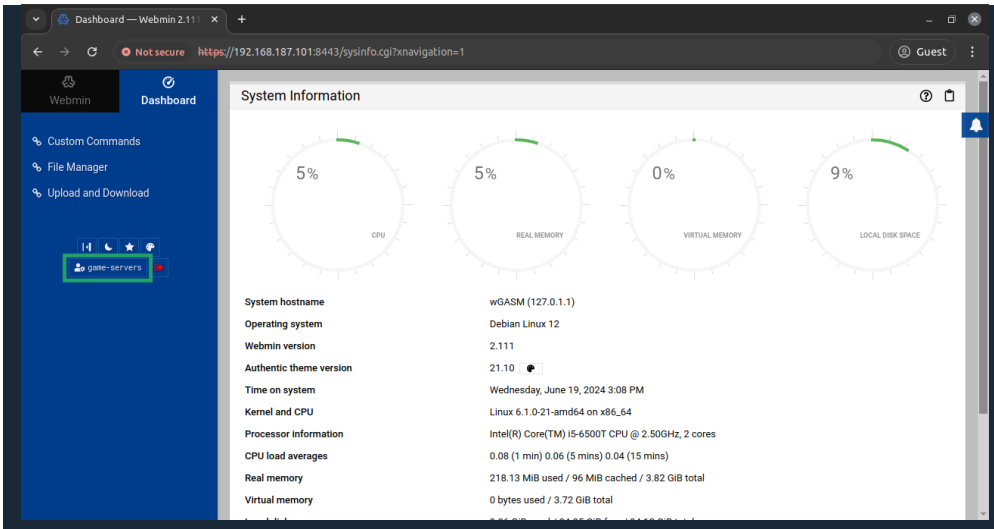
Logging-in as **game-servers**:

Logging-in with the *unprivileged* (*non*-administrator) user, such as the "**game-servers**" user, is useful for tasks such as:

- Accessing the "Custom Commands", which provides access to all **wGASM** functions.
- Uploading files (such as custom maps, and server-side modifications) for game-servers.
- Editing configuration files for game-servers.



.
if you log-in with the *unprivileged* (*non*-administrator) user, such as the "**game-servers**" user, you will be presented with a more limited set of options on the menu - effectively limited to just tasks associated with administration and maintenance of game-servers.



.

Note that at the bottom of the menus in the left-side panel, it will show which user you are currently logged-in with.

---

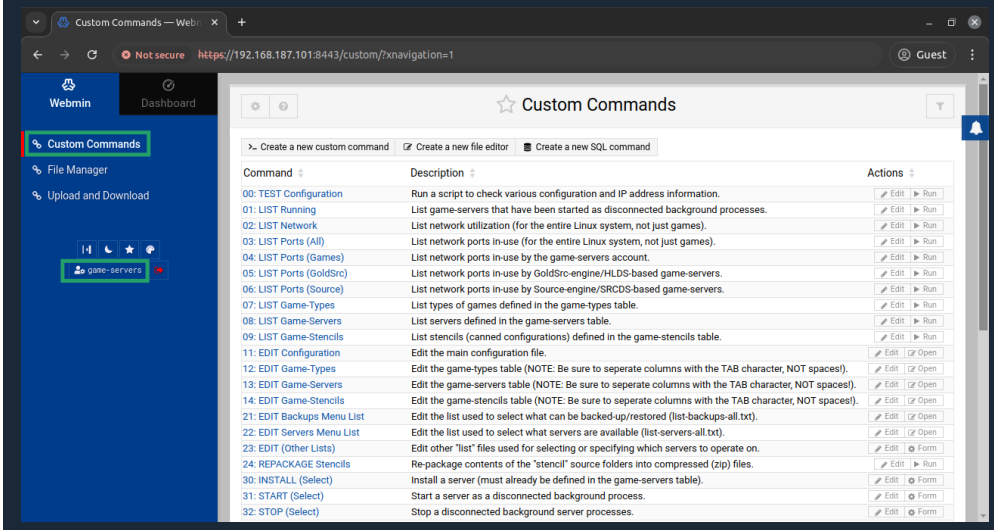
32 - Webmin "Custom Commands"

Requirements / Notes:

- Webmin [webmin.com] must be installed.
- The **wGASM**-provided "Custom Commands" must be installed.
- You must login to Webmin [webmin.com] as **game-servers**

Accessing **Custom Commands**:

A set of "Custom Commands" for Webmin [webmin.com] are made available, to effectively act as a web-interface for **wGASM** features. They are accessed (while logged-in as **game-servers**) by selecting the **Custom Commands** option on menu in the left-side panel.



.

The list of **wGASM**-related **Custom Commands** is long, but can generally be broken-down into several *categories*:

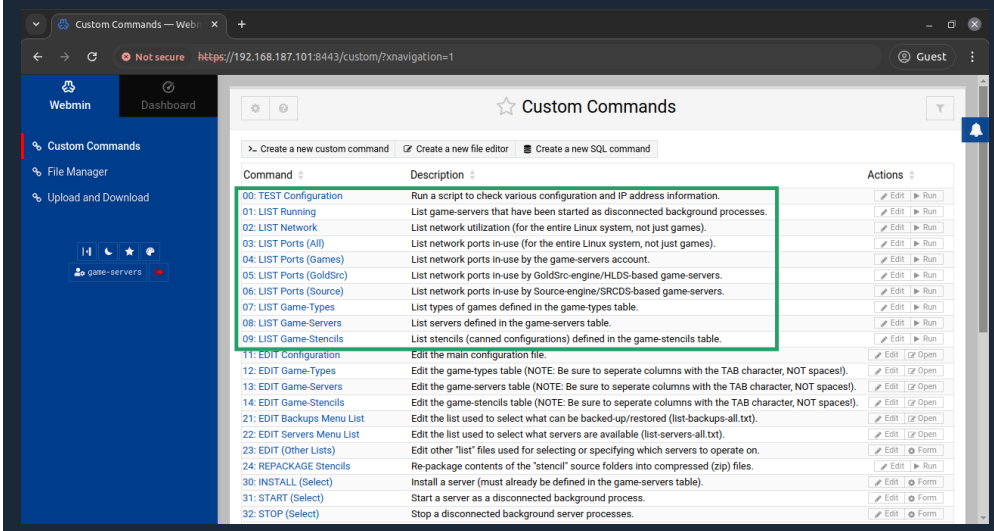
- Diagnostics.
- Configuration.
- Performing functions on a *single* server.
- Performing functions on *multiple* servers.

Each component is also *numbered* to make it easy to refer-to later.

Diagnostic **Custom Commands**:

These options are provided largely for diagnostic purposes, and to not perform any direct action on the game-servers themselves, or (normally) alter any configuration data. Generally, they display information about the system as a whole, game-types, stencils as well as the game-servers. Each option has an adjacent description that should make it obvious what each does.

Of particular note, is the (relatively new) option "**00 - TEST Configuration**" - which will display configuration information that **wGASM** uses along with *both* the computers **local** *and* **public** IP addresses. If you are renting a virtual private server (VPS) from a hosting provider, these are probably the same. Whereas, if you are hosting this yourself at home, they are probably different.



.

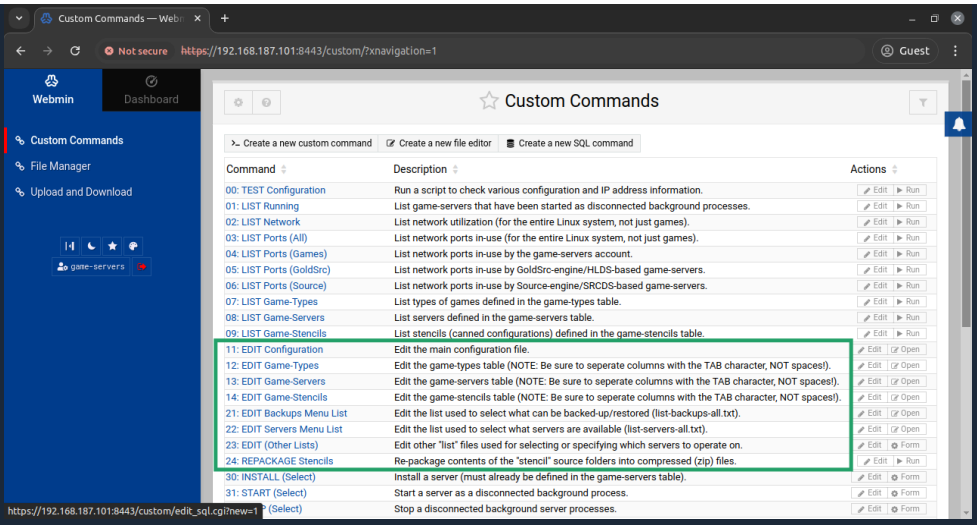
Configuration **Custom Commands**:

The next set of options provide easy editing of various configuration options, such as:

- Editing the **wGASM** central configuration file (**config.txt**).
- Editing the **wGASM** game types table (**game-types.tsv**).
- Editing the **wGASM** game servers table (**game-servers.tsv**).
- Editing the **wGASM** game stencils table (**game-stencils.tsv**).
- Editing various "lists" used for Webmin [webmin.com] menus that follow.

In particular there are two "list" files that are *critical* to the rest of the **Custom Commands**:

- The "Servers Menu List" (**~/wgasm/webmin/list-servers-all.txt**): This file contains a list of **GameServerID**'s that will be used by the other options that have a "Select" option to select a game-server. This list is initially populated with the **GameServerID** of each example game-server - using the same **GameServerID** from game servers table (**game-servers.tsv**). As you add your own servers to the game servers table (**game-servers.tsv**), you *must also add their **GameServerID** to this list* - otherwise they will not be displayed when using other **Custom Commands** that allow picking the server from a "list.
- The "Backups Menu List" (**~/wgasm/webmin/list-backups-all.txt**): This list file should have the same content as the The "Servers Menu List" (**list-servers-all.txt**) noted above, *plus a few extra entries* for things than can be *backed-up* (and possibly *restored*, but are *not game-servers*. This is typically an option the back-up the entire **~/wgasm** folder, specific configuration **data** for **wGASM**, the **~/stencils** folder, etc. Of course, all the game-servers must be listed here as well - to allow them to be selected for backups, restores, etc.

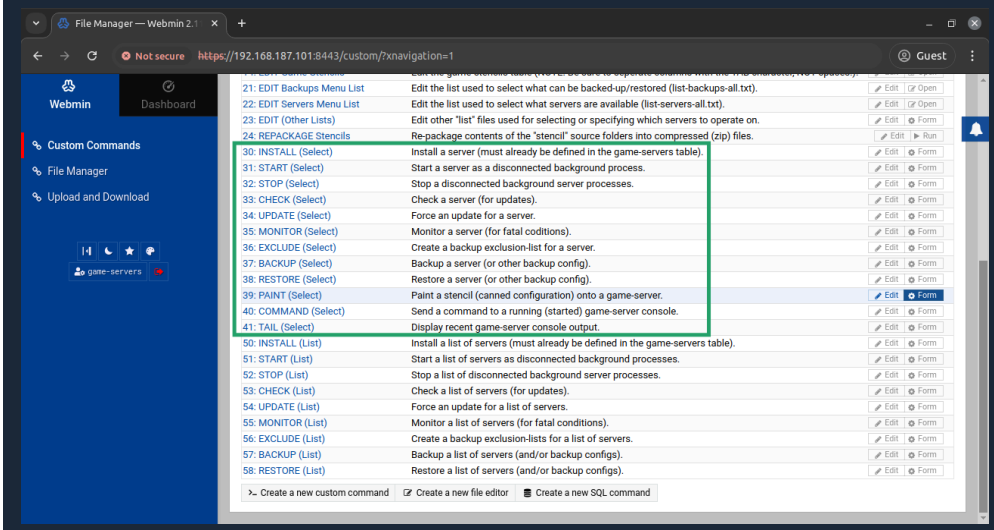


.

Single-Server **Custom Commands**:

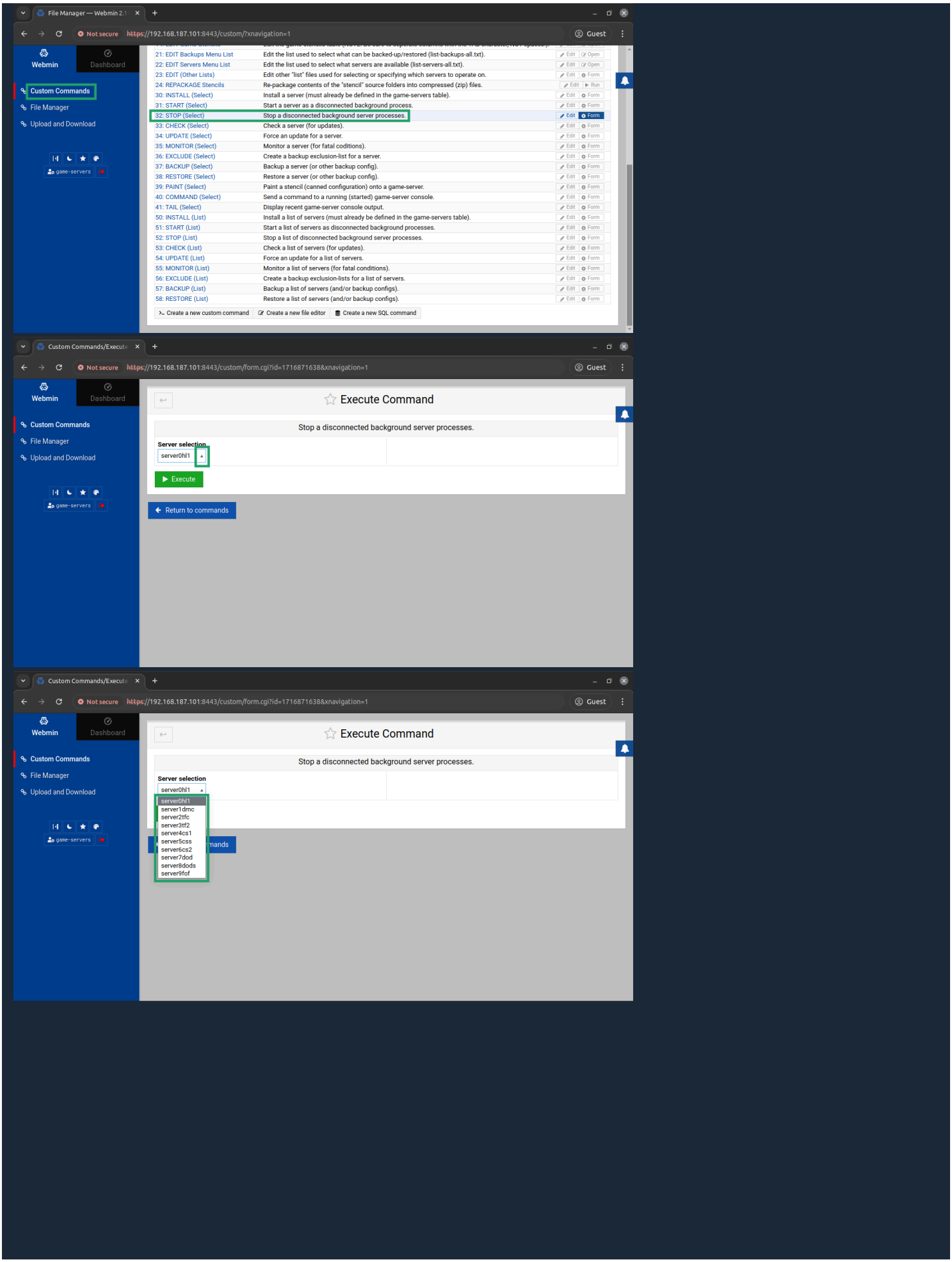
The next set of options are for functions that are performed for a *single* specific game-server, using the "Servers Menu List" (**list-servers-all.txt**) to provide a drop-down list of game-servers to select from. These include commonly-used functions, such as:

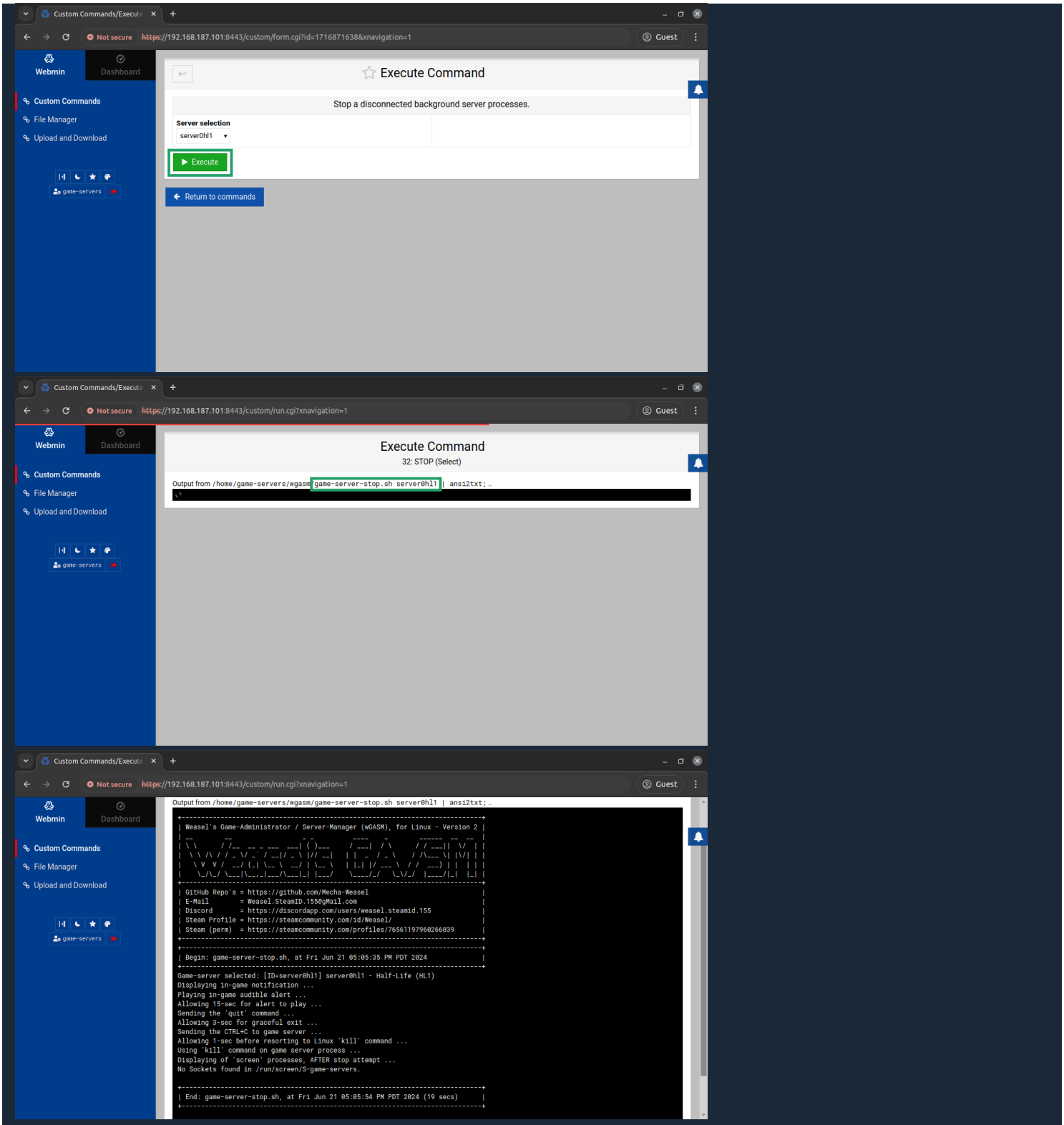
- Start a *single* game-server.
- Stop a *single* game-server.
- Forcibly update a *single* game-server.
- Send a command to a game-server's console.



.

As an example, using option "**32: STOP (Select)**". This will select which game-server to stop, as depicted below:


File Manager — Webmin 2.1
Not secure https://192.168.187.101:8443/custom/?xnavigation=1
Guest
Webmin
Dashboard
Custom Commands
File Manager
Upload and Download
game-servers
21: EDIT Backups Menu List Edit the list used to select what can be backed-up/restored (list-backups-all.txt).
22: EDIT Servers Menu List Edit the list used to select what servers are available (list-servers-all.txt).
23: EDIT (Other Lists) Edit other "list" files used for selecting or specifying which servers to operate on.
24: REPACKAGE Stencils Re-package contents of the "stencil" source folders into compressed (zip) files.
30: INSTALL (Select) Install a server (must already be defined in the game-servers table).
31: START (Select) Start a server as a disconnected background process.
32: STOP (Select) Stop a disconnected background server processes.
33: CHECK (Select) Check a server (for updates).
34: UPDATE (Select) Force an update for a server.
35: MONITOR (Select) Monitor a server (for fatal coditions).
36: EXCLUDE (Select) Create a backup exclusion-list for a server.
37: BACKUP (Select) Backup a server (or other backup config).
38: RESTORE (Select) Restore a server (or other backup config).
39: PAINT (Select) Paint a stencil (canned configuration) onto a game-server.
40: COMMAND (Select) Send a command to a running (started) game-server console.
41: TAIL (Select) Display recent game-server console output.
50: INSTALL (List) Install a list of servers (must already be defined in the game-servers table).
51: START (List) Start a list of servers as disconnected background processes.
52: STOP (List) Stop a list of disconnected background server processes.
53: CHECK (List) Check a list of servers (for updates).
54: UPDATE (List) Force an update for a list of servers.
55: MONITOR (List) Monitor a list of servers (for fatal conditions).
56: EXCLUDE (List) Create a backup exclusion-lists for a list of servers.
57: BACKUP (List) Backup a list of servers (and/or backup configs).
58: RESTORE (List) Restore a list of servers (and/or backup configs).
Create a new custom command
Create a new file editor
Create a new SQL command
Custom Commands/Execute
https://192.168.187.101:8443/custom/form.cgi?id=1716871638&xnavigation=1
Execute Command
Stop a disconnected background server processes.
Server selection
server0hl1
Execute
Return to commands
server0hl1
server1dmc
server2tfc
server3tf2
server4cs1
server5css
server6cs2
server7dod
server8dods
server9fof

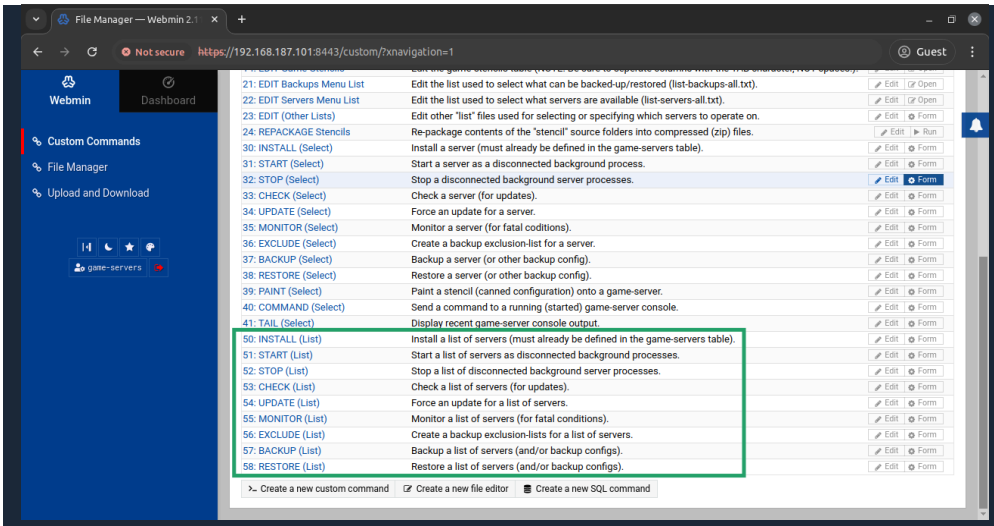


.

Multi-Server **Custom Commands**:

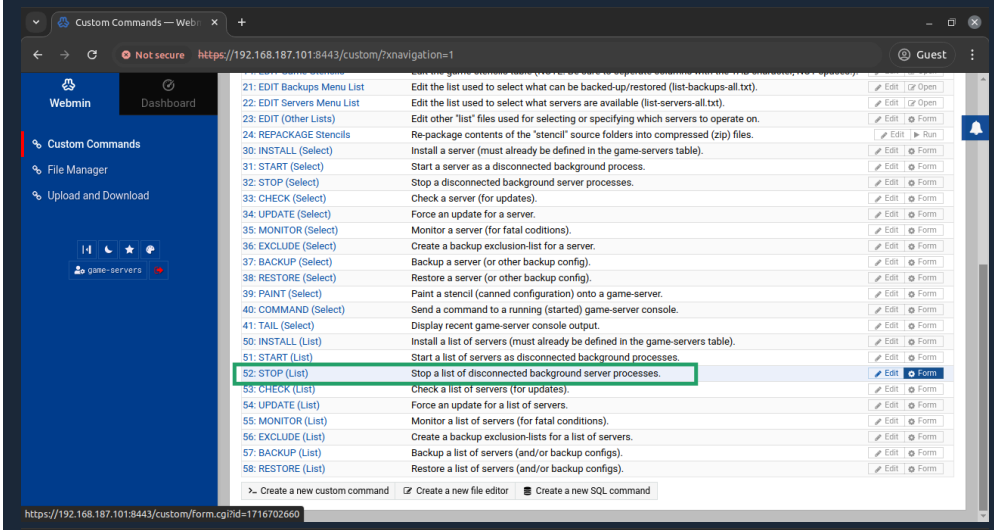
The last set of options are for functions that are performed for *multiple* game-servers. When clicked, each option will allow you to browse for a "list" text file, and will run the selected option for *all game-servers in that list*. For the list, you may use:

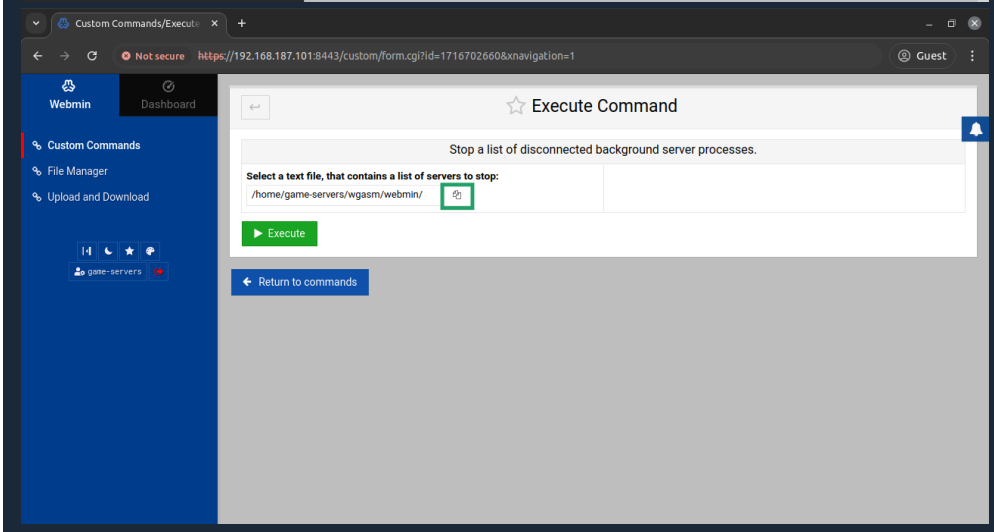
- The "Servers Menu List" (**~/wgasm/webmin/list-servers-all.txt**).
- The "Backups Menu List" (**~/wgasm/webmin/list-backups-all.txt**).
- Any other plain-text list file that *you care to create* in the **~/wgasm/webmin** folder.

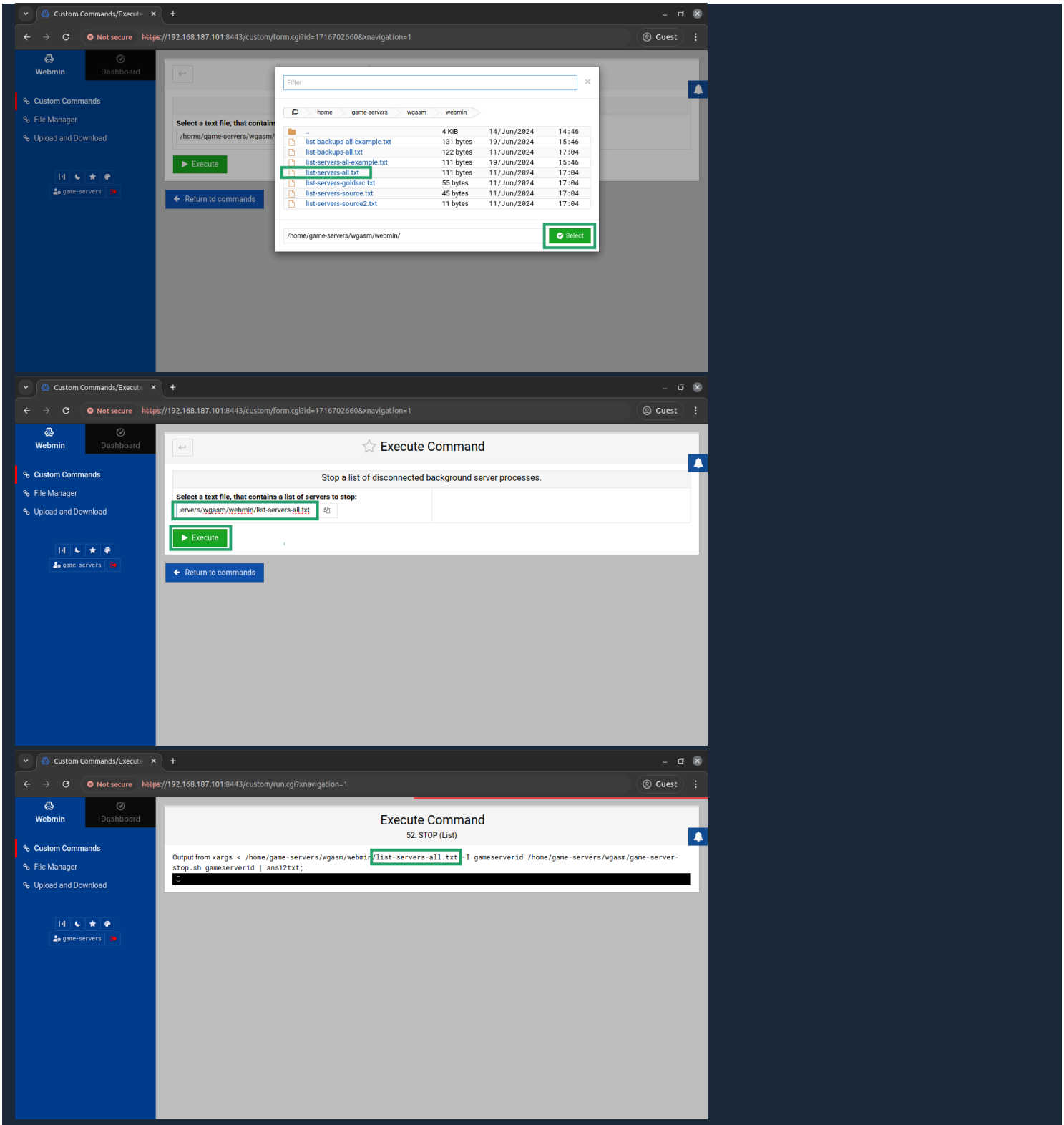


.

As an example, using option "**52: STOP (List)**", selecting the **~/wgasm/webmin/list-servers-all.txt** list. This will effectively stop all servers listed in that file, which in this case is any of the *example* game-servers that may be running, as depicted below:







33 - Scheduled "Cron" Jobs

{work-in-progress}

Requirements / Notes:

- Maintenance tasks may be performed automatically on a scheduled basis, by using the Linux Cron [en.wikipedia.org] job-scheduler.
- Any such scheduled tasks should be configured to operated under the *unprivileged* / non-*root* / non-*sudo* Linux user (i.e. "**game-servers**").
- Do **NOT** schedule the **cron** scripts *directly* in Cron [en.wikipedia.org] .

- Instead, the **cron-wrapper.sh** component to "wrap" the scripts.
- The **cron-wrapper.sh** component implements some basic conflict avoidance to prevent scheduled scripts from running at the same time - potentially interfering with each other.
- This can be particularly problematic if one of the scripts sometimes runs longer than it does at other times.
- For example: The **game-server-check.sh** component might normally complete *quickly* (in under one minute), but will take *much longer* (30-45 minutes?) if an update for Counter-Strike 2 needs to be installed.

Tasks to schedule:

Tasks they you may want to schedule to run automatically might include:

- Monitoring game-servers for unrecoverable conditions, using **game-server-monitor.sh**.
- Automatically applying any new updates for game-servers, using **game-server-check.sh**.
- Backing-up game-servers, using **game-server-backup.sh**.
- Clearing-out old logs.

Schedule recommendations:

The recommended schedule would be **four** (**4**) different jobs - each schedule in a way that minimizes overlapping:

- **Weekly**: A job scheduled *once per week*, which performs the longest-running tasks (backups, etc.), at the least active time of day.
- **Daily**: A job scheduled *every day* (except the day the **weekly** job runs, schedule to run at the same time of day the **weekly** job is scheduled.
- **Hourly**: A job scheduled to run *every hour of every day* - except the hour of day that the **weekly** and **daily** jobs are scheduled.
- **Often**: A job scheduled to run *several times an hour* - except the hour of day that the **weekly** and **daily** jobs are scheduled.

Job recommendations:

The recommended tasks to run during each schedule are:

- **Weekly**:
  - Use **game-server-stop.sh to stop all the game-servers normally left started.**
  - Clear all the logs from the **logs** folder (typically **~/logs**)
  - Use **game-server-backup.sh** to backup your **wGASM** configuration files.
  - Use **game-server-backup.sh** to backup all your installed game-servers.
  - Use **game-server-start.sh** to restart all the game-servers normally left started.
- **Daily**:
  - Use **game-server-backup.sh** to backup your **wGASM** configuration files.
  - Use **game-server-start.sh** to restart any servers that tend to have issues if you leave them running for multiple days at a time (for example: Team Fortress 2).
- **Hourly**:
  - Use **game-server-check.sh** for each game-server that you normally leave started (to ensure they keep updated with the latest version).
- **Often**:
  - Use **game-server-monitor.sh** for each game-server that you normally leave started (to ensure they keep operating).

Example schedule:

An example schedule based on that might be:

- **Weekly**: **03:00 AM** on *Mondays*.
- **Daily**: **03:00 AM** on *Tuesdays*, *Wednesdays*, *Thursdays*, *Fridays*, *Saturdays* and *Sundays*.
- **Hourly**: 23-hours per day (effectively every hour *except* **03:00 AM**) at the "top" of the hour (**__:00**).
- **Often**: *Several times* every hour (*except* the **03:00 AM** hour) - also avoiding the "top" of the hour to prevent conflicting with the **hourly** job.

Example scripts:

For each recommended scheduled job, there is an example script in the **cron** sub-folder (typically **~/wgasm/cron**).

- **Weekly**: Example: **games-*weekly*.sh**
- **Daily**: Example: **games-*daily*.sh**

- **Hourly**: Example: **games-*hourly*.sh**
- **Often**: Example: **games-*often*.sh**

Example Cron jobs:

Implementing the example scripts using the example schedule, would result in a **crontab** configuration such as this:

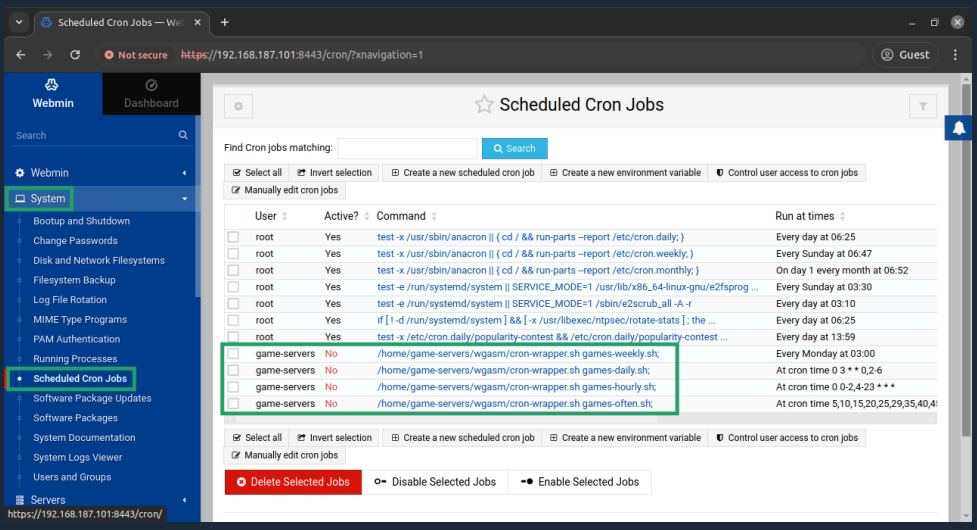
```
# 0 3 * * 1 /home/game-servers/wgasm/cron-wrapper.sh games-weekly.sh; #Game-Server Maintenance: WEEKLY
# 0 3 * * 0,2-6 /home/game-servers/wgasm/cron-wrapper.sh games-daily.sh; #Game-Server Maintenance: DAILY
# 0 0-2,4-23 * * * /home/game-servers/wgasm/cron-wrapper.sh games-hourly.sh; #Game-Server Maintenance: HOURLY
# 5,10,15,20,25,29,35,40,45,50,55 0-2,4-23 * * * /home/game-servers/wgasm/cron-wrapper.sh games-often.sh; #Game-Server Maintenance: OFTEN
```

Example Cron jobs (in Webmin):

If you have Webmin [webmin.com] installed, it is easy to create an manage scheduled Cron [en.wikipedia.org] jobs - using it's "Scheduled Cron Jobs" module.

- Log-into Webmin [webmin.com] in a web-browser as a *privileged* Linux user (i.e. "**root**" or some **sudo**-enabled user).
- Expand the "**System**" menu in the left-panel.
- From the expanded "**System**" menu, select the "**Scheduled Cron Jobs**" option.
- Click the "**Create new scheduled cron job**" button.
- For the new job, ensure the **user** specified is "**game-servers**".
- Select the option to make the job as either "**Inactive**" or "**Active**" depending on whether or not you would like them to start *immediately*.
- In the "**Command**" field, enter the full path to the **cron-wrapper.sh** script, followed by the name of the script (in the **cron** folder) for the wrapper to use.
- In the "**Description**" field, enter the relevant description.
- Select the schedule details for that job - day(s) of the week, hour(s) of the day, minutes of those hour(s), etc.
- Click the "**Create**" button.

The result should look something like this image:



35 - Example VM(s), Part-1

Requirements / Notes:

- This guide assumes you already have *either* VirtualBox [en.wikipedia.org] or VMware [en.wikipedia.org] *installed*, and
- Know *how to use it* - how to create virtual machines, where the default file locations for virtual machines are located, etc.
- While it may be possible to port the example virtual machine from **VirtualBox** and/or **VMware** to other virtualization platforms (such Hyper-V [en.wikipedia.org] , or KVM [en.wikipedia.org]).

However, doing so is *beyond the scope of this guide*.

What is available:

Example virtual machines (VM's) are available for two different virtualization platforms (aka "hypervisors"). These are:

- **VirtualBox**: VirtualBox [en.wikipedia.org] is a free virtualization product, which is available for several different desktop operating systems including: **Windows**, **Linux** and **MacOS**. You can download VirtualBox [en.wikipedia.org] from the official download site [www.virtualbox.org] .
- **VMware**: VMware [en.wikipedia.org] is a *commercial* virtualization system. However, they do offer *free-for-**non-commercial-use*** versions of their desktop virtualization products, specifically VMware Workstation [en.wikipedia.org] (for **Windows** and **Linux**), and VMware Fusion [en.wikipedia.org] for **MacOS**. However, although now available on a *free-for-**non-commercial-use*** basis, frankly it is *harder than it needs to be* to navigate your way through the VMware / Broadcom maze of web-sites to register for an account and then finally find the right download links. I have included a VMware [en.wikipedia.org] image mostly for people who may already be familiar with and have VMware [en.wikipedia.org] installed.

Downloading the VM(s):

The size of the VM image (compressed with 7-zip [www.7-zip.org]) is approximately **2-GB**. Both the VirtualBox [www.virtualbox.org] *and* VMWare [en.wikipedia.org] drive images are uploaded to the **Internet Archive**, and may be found with this link:

```
https://archive.org/search?query=wGASM
```

Copies are also uploaded to my public **DropBox**, here:

```
https://www.dropbox.com/scl/fo/qr0nxjefykegw3ix9p98s/AGHvx9t3sqoH_l88uR9FCso?
rlkey=l0h4kyqz3wbmdne4rbw0exc1r&st=ma5xh5kw&dl=0
```

Purpose of the VM(s):

The example VM(s) were created primarily as a **learning tool**. However, they could also be used to host games *locally* at your residence or office. They are not really intended for hosting games on the Internet (at your residence or office). Doing that would involve additional issues, such as:

- Potential *asynchronous* Internet bandwidth issues. This refers to your Internet connection having a much bigger ***download*** speed, compared to ***upload*** speed. This is particularly common for *residential* or *small-business* Internet connections using cable-modems [en.wikipedia.org] , DSL [en.wikipedia.org] or wireless broadband [en.wikipedia.org] connections. To host game-servers that are *reasonably playable from **others** on the Internet*, you really need good **upload** speed - basically matching your **download** speed. Otherwise, while the game may play fine for you locally, for others on the Internet it will *lag too painfully to be playable*.
- Network port-forwarding / firewall issues. This refers to your Internet connection's router/firewall feature not (by default) allowing **inbound** traffic for various types of **servers**. In the local router / firewall that connects your at home/small-office to the Internet (which might be *yours*, or *rented* from your Internet Service Provider) you will need to *allow* whatever "ports" are being used for your game-servers (typically **27000-27999**), and ensure they are "forwarded" to the inside IP address of the VPS.

It is *beyond the scope of this guide* to cover either of those issues.

Specifications of the VM(s):

The example VM(s) are built with just enough **storage space** and **memory** to be able to have several game-servers *installed*, but only **one** (**1**) or *maybe* **two** (**2**) *running* at any given time.

The **specifications** of the virtual machine(s), are as follows:

Specification	Details
Platform(s):	VirtualBox [www.virtualbox.org] **VDI** format for **Windows**, **Linux** or **MacOS**. ~ or ~ VMware [en.wikipedia.org] **VMDK** format, which should work with any of these products (and possibly some others):

	• **VMware Player** for **Windows** or **Linux**, • **VMware Workstation** for **Windows** or **Linux**, • **VMware Fusion** for **MacOS**, • **VMware ESX/ESXi** (their data-center product).
Processor:	2x Virtual CPU.
Network Interface:	"Bridged" to host computer network (shows on the local network as its own computer, and should get its own IP address dynamically).
Memory:	4-GB
Storage:	100-GB virtual drive size. Approximately 10-GB of space already utilized. Compressed down (using 7-zip [www.7-zip.org]) about **2-GB** for the initial download.

Features of the VM(s):

The **features** included in the virtual machine(s), are as follows:

Feature	Details
Base Operating System:	Debian version 12.5 "Bookworm" (Link [www.debian.org]) Non-GUI "headless-server" installation. SSH & Webmin [webmin.com] access enabled.
Linux-native Steam-based game-server installer:	SteamCMD pre-installed,
Management System:	wGASM [github.com] pre-installed.
Linux Management:	Webmin [webmin.com] , with the **wGASM**-related "Custom Commands" pre-installed. NOTE: Webmin is configured on port **8443** (*not* the default of 10000)
Linux Firewall	**IP Tables** configured with some default filtering rules, such as: • Allow game-servers on ports **27000** to **27999**. • Allow SSH only from local network (**192.168.0.0/16**). • Allow **Webmin** only from local network (**192.168.0.0/16**).

36 - Example VM(s), Part-2

Logging into the example VM:

After you get the virtual machine (VM) downloaded, extracted, placed into the proper folder, and added to VirtualBox [en.wikipedia.org] or VMware [en.wikipedia.org] , you can log into it's **virtual console** directly in as either of these two Linux users:

Linux User	Password	Description
root	**n0t.Password!**	System administrative user, for installing/updating packages, etc. *Do NOT use for game-servers!* **NOTE**: You should change this password *immediately*.
game-servers	**n0t.Password!**	All example game-servers, SteamCMD, and all **wGASM** functions should be run under this user. **NOTE**: You should change this password *immediately*.

Both of those Linux users should also have the ability to login with *both* SSH [en.wikipedia.org] *and* through the Webmin [webmin.com] interface in a web-browser on your host computer.

To find the IP address of the virtual machine, login to it's virtual console as "**root**", and use this command:

```
ifconfig;
```

It should display some output like this:

```
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    --> inet 192.168.187.132  netmask 255.255.255.0  broadcast 192.168.187.255
        inet6 fe80::20c:29ff:fe54:f843  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:54:f8:43  txqueuelen 1000  (Ethernet)
        {more crap here}
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        {more crap here}
```

Note this line in the output:

```
inet 192.168.187.132  netmask 255.255.255.0  broadcast 192.168.187.255
```

In that case, the local IP address of your virtual machine would be:

```
192.168.187.132
```

Example Dedicated Servers:

The example virtual machine (VM) implements one sample game-server for *each* of the **ten** (**10**) game-types that that have been specifically tested with **wGASM**.

Although the example VM has example servers for *all* of the **ten** (**10**) game-types *configured*, it does not have sufficient memory to have more than **one** (**1**) or *maybe* **two** (**2**) actually *running at the **same time***.

The **gameserverid** and **portnumber** assigned to each sample is listed in the table below:

GameServerID	Network Port	Game
server**0**hl1	27101	Half-Life
server**1**dmc	27103	Deathmatch Classic
server**2**tfc	27105	Team Fortress Classic
server**3**tf2	27107	Team Fortress 2
server**4**cs1	27109	Counter-Strike 1.6
server**5**css	27111	Counter-Strike:Source
server**6**cs2	27113	Counter-Strike 2
server**7**dod	27115	Day of Defeat
server**8**dods	27117	Day of Defeat:Source
server**9**fof	27119	Fistful of Frags

Shutting-down the example VM:

To gracefully shut-down the example virtual machine (VM), you should:

- Be sure to ***stop** any game server(s)* you previously had running.
- Log-out of Webmin [webmin.com] .
- Log-out of any **SSH** sessions.
- Go to the **virtual** console for the example VM.
- Login to the console as **root**.

- Use this command to shut-down the virtual machine:

```
shutdown -h now;
```

37 - Example VM(s), Part-3

As noted previously, the example virtual machine (VM) is primarily intended as a **learning tool**. However, if you are going to have it *turned-on for an extended period of time* (perhaps to host a *LAN-game*), you should consider some additional actions to prevent it being exploited either from other computers on the same local network, or from the Internet.

Changing the default passwords:

The first and most important basic change, should be *changing the default passwords* for both the **root** *and* **game-servers** Linux users.

This can be accomplished by

- Logging in as **root** to the **virtual console** (or over the network using **SSH**), and then using the passwd [man7.org] command to change the password for **root**.
- Logging in as **game-servers** to the **virtual console** (or over the network using **SSH**), and then using the passwd [man7.org] command to change the password for **game-servers**.

Host-based firewall:

Please note also that the example VM does *already have* a rudimentary *host-based* firewall feature enabled - called IP Tables [en.wikipedia.org] . Some basic rules have been setup - which can be viewed in **Webmin** while logged-in as **root**.

This existing host-based firewall implementation does already include these rules (along with some other basic stuff):

- The SSH server (on **TCP port 22**) may only be accessed from *local networks* (specifically **192.168.0.0/16** in CIDR notation).
- The "Webmin" interface (on **TCP port 8443**) may only be accessed from *local networks* (specifically **192.168.0.0/16** in CIDR notation).
- Game-servers must be assigned ports in the **27000-27999** port-range.

NOTE: Although the current IP Tables [en.wikipedia.org] setup is configured to permit game-servers operating in the **27000-27999** port-range - each of the *existing example* game-servers already configured, are assigned ports from a smaller subset range of **27100-27199** range. This is intended to keep any ports used by the example game- **servers** from *interfering with local game-**clients** that might be running the same local network.*

8 Comments

[WL] Weasel (Probably AFK) [author] Dec 31, 2024 @ 3:28pm
NOTE: For questions and support with **wGASM** please use this Steam Community Discussionions area:

https://steamcommunity.com/groups/WeaselsLair/discussions/20/

[WL] Weasel (Probably AFK) [author] Dec 7, 2024 @ 10:31pm
NOTE : I have been tinkering with getting this system working with Rocky Linux [rockylinux.org] (part of the broader Red-Hat family [en.wikipedia.org] of distributions), but I am stuck on at least one package that is *critical* to (i.e. ***extensively*** *used by*) this system. That package is colorized-logs [github.com] , specifically the **ansi2txt** utility. If anyone knows *exactly* how to get that installed on Rocky Linux 9.x [rockylinux.org] , ***please contact me!***

[WL] Weasel (Probably AFK) [author] Sep 11, 2024 @ 10:52am
Yeah, but don't hold your breath waiting for Valve to provide community dedicated server support - if how it went with CS2 is any indication. Although, having this giant semi-open

alpha testing seems like they are taking feedback seriously.

Cabben06 Sep 10, 2024 @ 4:44am
Ohhh Deadlock servers would be fun!

[WL] Weasel (Probably AFK) [author] Sep 9, 2024 @ 2:10pm
NOTE: Will update to accommodate Deadlock as soon as Valve releases a **Linux-Native** dedicated-server for it.

Cabben06 Jul 27, 2024 @ 1:40am
Cheers!

[WL] Weasel (Probably AFK) [author] Jul 26, 2024 @ 3:28pm
@Cabben06, fixed in update just pushed to GitHub. Thanks for reporting!

Cabben06 Jul 26, 2024 @ 8:23am
I'm following the guide as I'm typing this. When running install-wgasm.sh and getting to "Deploying data tables..." it failed to cp the 3 example data tables as they were called "game-types **.example** .tsv" when the cp command was searching for "game-types **-example** .tsv

I will now cp the 3 tables manually and then run install-wgasm.sh again. This is a fantastic guide/system all the same and I'm incredibly thankful you made this