

# Tournaments Project Hosting Setup on Hostinger

## Prerequisites

Before you begin, ensure you have the following:

- A Hostinger account
- Domain configured
- SSH credentials
- Vim knowledge (If you don't know Vim, watch the following video: <https://youtu.be/9cC9x-ntNQY?si=phVs66Fv8Ndwzldn>)

## Step 1: Navigate to VPS Panel

1. Log in to your Hostinger account.
2. Navigate to the **VPS** section and click **Manage**

The screenshot shows the Hostinger VPS panel. On the left, there's a sidebar with links: Home, Websites, Domains, Emails, **VPS** (which is highlighted), and Marketplace. The main area has a header with 'VPS' and a 'Get new VPS plan' button. Below it, there's a 'Black Friday sale!' banner with three promotional boxes: 'Get a new VPS plan', 'Get a new hosting plan', and 'Get a new email plan'. Each box shows a discounted price (e.g., \$4.99/mo) and a 'Get deal' button. At the bottom, there's a 'Server Details' section showing 'KVM 2' and 'EXPIRES' status, along with 'SSH Access' and 'Status' dropdowns. The 'Status' dropdown is set to 'Running'. To the right of the status are 'Manage' and '...' buttons, with the 'Manage' button being highlighted with a red box. The bottom right corner features a blue circular icon with a white play symbol.

### 3. Now you are under Hostinger VPS panel!

The screenshot shows the Hostinger VPS Overview page. On the left is a sidebar with links like Main menu, Overview, Kodee AI assistant, Settings, OS & Panel, Backups & Monitoring, Security, and Tutorials. The main area has tabs for Overview, Panel access, SSH access, and Plan details. Under Overview, it shows:

- HestiaCP** Build on Ubuntu 22.04
- Username**: admin
- Password**: (reset)
- Manage panel** button
- VPS information** tab (selected):

  - IP address: 145.223.73.244
  - Hostname: srv634274.hstgr.cloud
  - Status: Running
  - VPS uptime: 5 days 16 hours
  - Current OS: Ubuntu 22.04 with HestiaCP
  - Control panel guide: [Learn about your VPS template in this article](#)
  - Location: United States
  - Node: us-bos-pve-node456

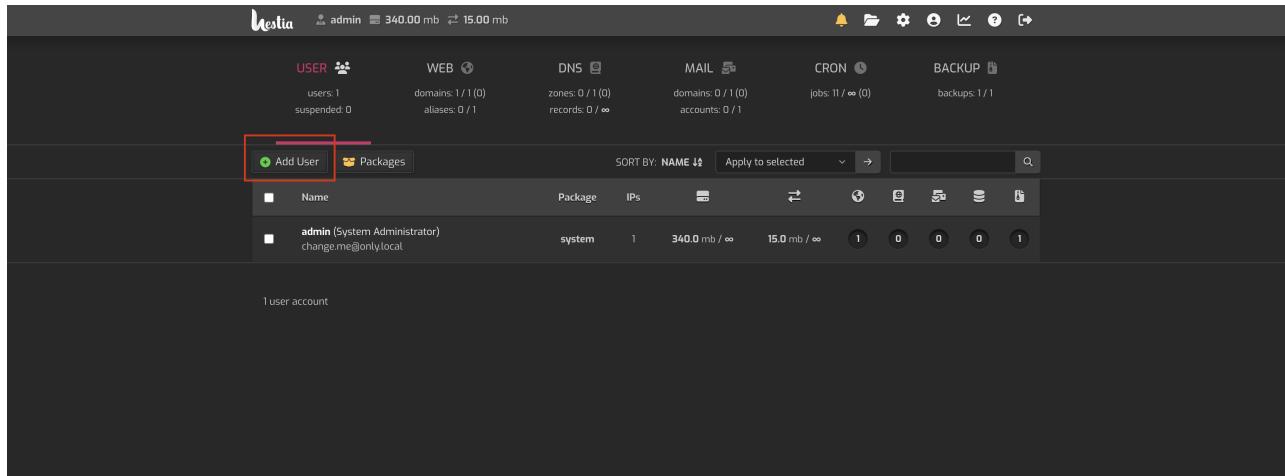
- Reboot VPS** and **Stop VPS** buttons

## Step 2: Create Database

### 1. Click Manage Panel

The screenshot is identical to the previous one, but the **Manage panel** button in the top right corner of the main content area is highlighted with a red box.

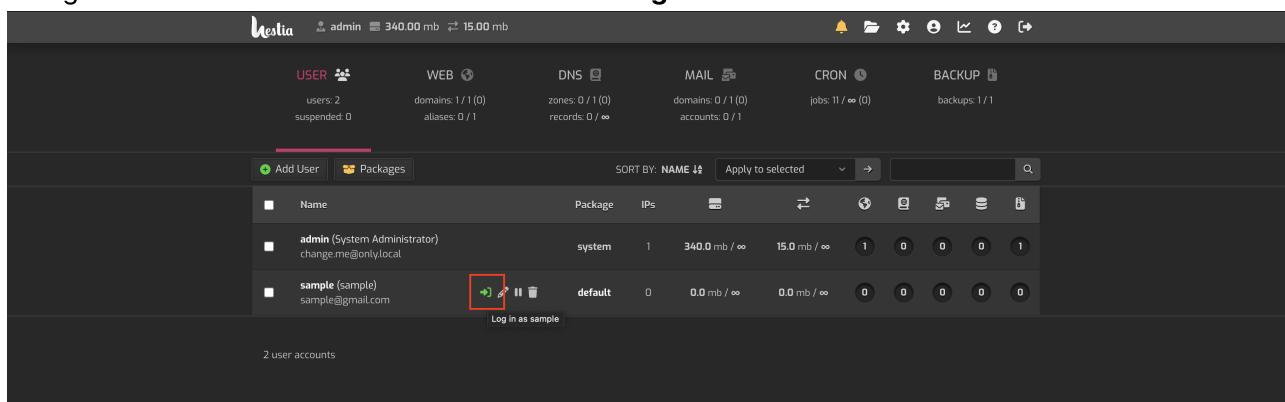
### 2. Now you are in Hestia's control Panel, in order to create a **new Database** you will need to create a **user**, click **Add User**



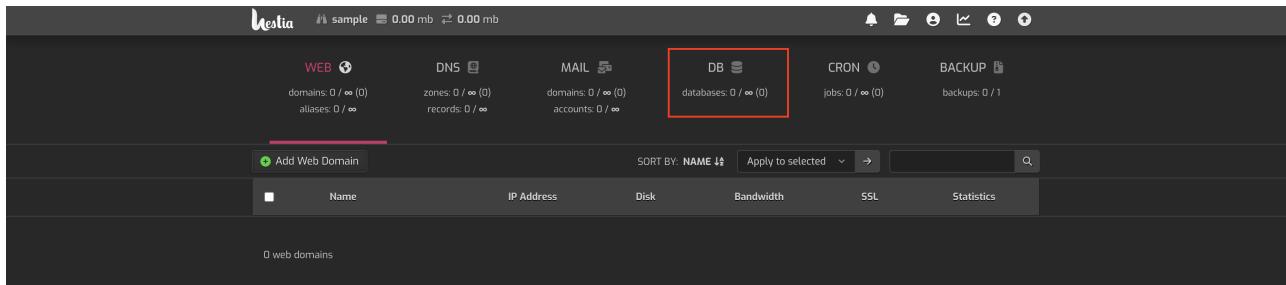
### 3. Now fill out form and click save

The screenshot shows the 'Add User' form. It includes fields for Username ('sample'), Contact Name ('sample'), Email ('sample@gmail.com'), Password (redacted), Language ('English (English)'), Role ('User'), Package ('default'), and Email login credentials to ('sample@gmail.com'). A red box highlights the 'Save' button at the top right of the form.

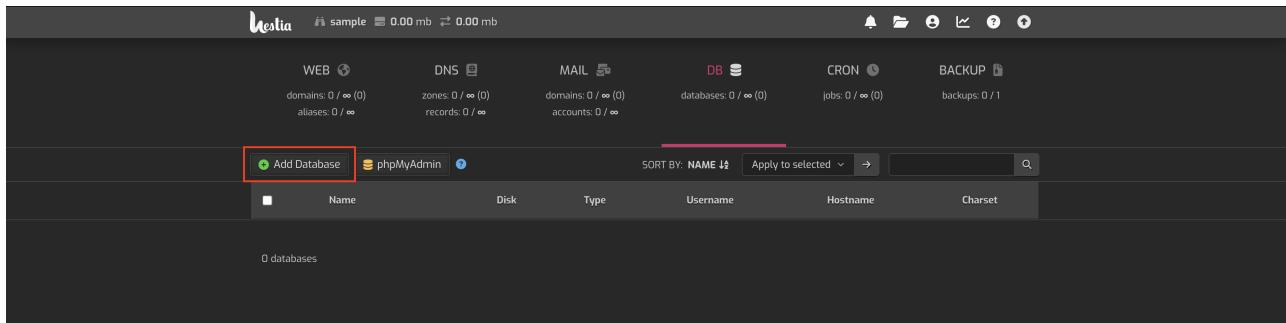
### 4. Navigate back to the Users tab and click on the Login Icon



## 5. Now, Navigate to the DB tab



## 6. Click Add Database



## 7. Now fill out form, remember your Database's credentials and click save

This screenshot shows the 'Add User' form in the Hestia Control Panel. It includes fields for Username (sample), Contact Name (sample), Email (sample@gmail.com), Password (a partially obscured field), and various checkboxes and dropdowns for Language (English), Role (User), and Package (default). A red box highlights the 'Save' button in the top right corner.

## Step 3: Connect to Hestia panel SSH via

SSH (Secure Shell) is a protocol used to securely connect to a remote computer or server over a network. It provides encrypted communication for tasks such as command-line access, file transfers, and remote administration.

1. If you are **NOT** on **Windows** you can **skip** this step. If you are on **Windows** bruh... you are not cooking, Follow this: [Tutorial to setup SSH for windows terminal](#)
2. Go back to your **Hostinger VPS panel**

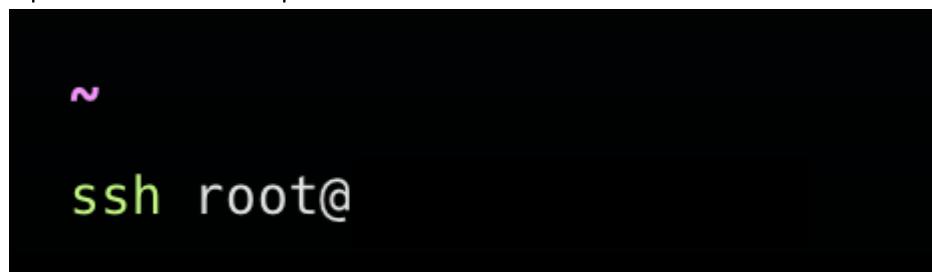
### 3. Click SSH access

The screenshot shows the Hostinger VPS Overview page. On the left, there's a sidebar with links like Main menu, Overview, Kodee AI assistant, Settings, OS & Panel, Backups & Monitoring, Security, and Tutorials. The main area has tabs for Overview, VPS information, Panel access, **SSH access**, and Plan details. The SSH access tab is highlighted with a red border. It contains fields for IP address (disabled), Hostname (disabled), Status (Running), VPS uptime (5 days 16 hours), Current OS (Ubuntu 22.04 with HestiaCP), Control panel guide (link), Location (United States), and Node (us-bos-pve-node456). At the bottom are Reboot VPS and Stop VPS buttons. A purple 'Manage panel' button is also present.

### 4. Copy the SSH terminal credentials

This screenshot is similar to the previous one, showing the Hostinger VPS Overview page. The SSH access tab is selected. The 'Terminal' field, which contains the text 'ssh root@', is highlighted with a red border. Below the terminal field, there are sections for CPU usage (1%), Memory (25%), Disk Usage (5 GB / 100 GB), and Bandwidth (0.002 TB / 8 TB).

### 5. Open a terminal and paste SSH credentials



### 6. Input your **password**

```
~  
ssh root@  
root@ s password: █
```

## Step 4: Project setup

1. Install php: `sudo apt install php8.1 php8.1-cli php8.1-fpm php8.1-mbstring php8.1-xml php8.1-curl php8.1-zip -y` you can **verify** installation by running `php -v`
2. Download Composer installer `curl -sS https://getcomposer.org/installer -o composer-setup.php`
3. Run Composer installer `sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer`
4. Remove the installer `rm composer-setup.php` you can also **verify** installation by running `composer -v`
5. Navigate to your **public\_html domain folder** `cd /home/youruser/web/yourdomain/public_html/`
6. Run: `git clone https://github.com/StarEngineer89/ci_tournament_bracket-generator` **Output:**

```
bash-5.1$ git clone https://github.com/StarEngineer89/ci_tournament_bracket-generator  
Cloning into 'ci_tournament_bracket-generator'...  
remote: Enumerating objects: 4263, done.  
remote: Total 4263 (delta 0), reused 0 (delta 0), pack-reused 4263 (from 1)  
Receiving objects: 100% (4263/4263), 1.70 MiB | 15.53 MiB/s, done.  
Resolving deltas: 100% (3400/3400), done.
```

7. Navigate to your project `cd project`
8. Run `composer update` you will be ask to **Continue as root/super user [yes]?** type yes.  
**Output:**

**composer update**

```

- Installing codeigniter4/settings (v2.2.0): Extracting archive
- Installing codeigniter4/shield (v1.1.0): Extracting archive
- Installing psr/container (2.0.2): Extracting archive
- Installing fakerphp/faker (v1.24.0): Extracting archive
- Installing paragonie/random_compat (v9.99.100): Extracting archive
- Installing paragonie/constant_time_encoding (v3.0.0): Extracting archive
- Installing phpseclib/phpseclib (3.0.42): Extracting archive
- Installing monolog/monolog (3.7.0): Extracting archive
- Installing psr/http-client (1.0.3): Extracting archive
- Installing guzzlehttp/promises (2.0.4): Extracting archive
- Installing guzzlehttp/guzzle (7.9.2): Extracting archive
- Installing psr/cache (3.0.0): Extracting archive
- Installing firebase/php-jwt (v6.10.1): Extracting archive
- Installing google/auth (v1.43.0): Extracting archive
- Installing google/apiclient-services (v0.380.0): Extracting archive
- Installing google/apiclient (v2.18.0): Extracting archive
- Installing mikey179/vfsstream (v1.6.12): Extracting archive
- Installing symfony/process (v7.1.7): Extracting archive
- Installing symfony/polyfill-php80 (v1.31.0): Extracting archive
- Installing symfony/filesystem (v7.1.6): Extracting archive
- Installing norkunas/youtube-dl-php (v2.8.0): Extracting archive
- Installing psr/simple-cache (3.0.0): Extracting archive
- Installing markbaker/matrix (3.0.1): Extracting archive
- Installing markbaker/complex (3.0.2): Extracting archive
- Installing maennchen/zipstream-php (3.1.1): Extracting archive
- Installing phpoffice/phpspreadsheet (2.3.0): Extracting archive
- Installing sebastian/version (4.0.1): Extracting archive
- Installing sebastian/type (4.0.0): Extracting archive
- Installing sebastian/recursion-context (5.0.0): Extracting archive
- Installing sebastian/object-reflector (3.0.0): Extracting archive
- Installing sebastian/object-enumerator (5.0.0): Extracting archive
- Installing sebastian/global-state (6.0.2): Extracting archive
- Installing sebastian/exporter (5.1.2): Extracting archive
- Installing sebastian/environment (6.1.0): Extracting archive
- Installing sebastian/diff (5.1.1): Extracting archive
- Installing sebastian/comparator (5.0.3): Extracting archive
- Installing sebastian/code-unit (2.0.0): Extracting archive
- Installing sebastian/cli-parser (2.0.1): Extracting archive
- Installing phpunit/php-timer (6.0.0): Extracting archive
- Installing phpunit/php-text-template (3.0.1): Extracting archive
- Installing phpunit/php-invoker (4.0.0): Extracting archive
- Installing phpunit/php-file-iterator (4.1.0): Extracting archive
- Installing theseer/tokenizer (1.2.3): Extracting archive
- Installing nikic/php-parser (v5.3.1): Extracting archive
- Installing sebastian/lines-of-code (2.0.2): Extracting archive
- Installing sebastian/complexity (3.2.0): Extracting archive
- Installing sebastian/code-unit-reverse-lookup (3.0.0): Extracting archive
- Installing phpunit/php-code-coverage (10.1.16): Extracting archive
- Installing phar-io/version (3.2.1): Extracting archive
- Installing phar-io/manifest (2.0.4): Extracting archive
- Installing myclabs深深-copy (1.12.1): Extracting archive
- Installing phpunit/phpunit (10.5.38): Extracting archive
Generating optimized autoload files
46 packages you are using are looking for funding.
Use the `composer fund` command to find out more!

```

9. Rename env to .env `mv env .env`

10. Open .env with vim `vim .env`

11. Edit your baseURL to '`https://yourdomain/`'

```
app.baseURL = 'https://yourdomain/'
```

12. Edit your database credentials, use the credentials you created in **First Step**

```
database.default.hostname = localhost
database.default.database = sample_db
database.default.username = sample_root
database.default.password =
database.default.DBDriver = MySQLi
database.default.DBPrefix =
database.default.port = 3306
```

13. Save and exit **vim** editor

14. Before running spark, we will need to setup kint. run `composer remove kint-php/kint && composer require kint-php/kint:^5.0 --dev`

15. Now, run `vim vendor/kint-php/kint/src/Renderer/RichRenderer.php`

16. Then change the following line: `public $richSort = Renderer::SORT_FULL;` to `public $richSort = RichRenderer::SORT_FULL;`

17. To migrate project database run `php spark shield:setup` you will be asked to overwrite type `n` when asked, then you will be ask to `Run spark migrate --all now?` type `y`

```
php spark shield:setup

File 'APPPATH/Config/Auth.php' already exists in destination. Overwrite? [n, y]: n
Skipped APPPATH/Config/Auth.php. If you wish to overwrite, please use the '-f' option or reply 'y' to the prompt.
File 'APPPATH/Config/AuthGroups.php' already exists in destination. Overwrite? [n, y]: n
Skipped APPPATH/Config/AuthGroups.php. If you wish to overwrite, please use the '-f' option or reply 'y' to the prompt.
File 'APPPATH/Config/AuthToken.php' already exists in destination. Overwrite? [n, y]: n
Skipped APPPATH/Config/AuthToken.php. If you wish to overwrite, please use the '-f' option or reply 'y' to the prompt.
Autoload Setup: Everything is fine.
Security Setup: Everything is fine.
Email Setup: Everything is fine.
Run spark migrate --all now? [y, n]: y

Running all new migrations...
    Running: (CodeIgniter\Shield) 2020-12-28-223112_CodeIgniter\Shield\Database\Migrations>CreateAuthTables
    Running: (CodeIgniter\Settings) 2021-07-04-041948_CodeIgniter\Settings\Database\Migrations>CreateSettingsTable
    Running: (CodeIgniter\Settings) 2021-11-14-143905_CodeIgniter\Settings\Database\Migrations>AddContextColumn
    Running: (App) 2024-05-02-053318_App\Database\Migrations>CreateParticipantsTable
    Running: (App) 2024-05-02-053421_App\Database\Migrations>CreateBracketsTable
    Running: (App) 2024-05-08-021145_App\Database\Migrations>CreateTournaments
    Running: (App) 2024-05-27-013123_App\Database\Migrations>CreateLogActionsTable
    Running: (App) 2024-05-28-133051_App\Database\Migrations>CreateShareSettingsTable
    Running: (App) 2024-06-13-160401_App\Database\Migrations>AddDeletedByToBracketsTable
    Running: (App) 2024-06-18-120729_App\Database\Migrations>TournamentShareAccessLogTable
    Running: (App) 2024-06-21-083733_App\Database\Migrations>AddURLInMusicSettingsTable
    Running: (App) 2024-06-28-140454_App\Database\Migrations>AddSearchableColumnInTournamentsTable
    Running: (App) 2024-07-01-204928_App\Database\Migrations>AddArchivedColumnToTournamentsTable
    Running: (App) 2024-07-03-235325_App\Database\Migrations>CreateNotificationsTable
    Running: (App) 2024-07-05-044320_App\Database\Migrations>CreateUserSettingsTable
    Running: (App) 2024-07-11-055946_App\Database\Migrations>AddShuffleEnabledColumnInTournamentsTable
    Running: (App) 2024-07-16-151458_App\Database\Migrations>AddDescriptionColumnInTournamentTable
    Running: (App) 2024-07-17-123407_App\Database\Migrations>AddScoreColumnsInTournamentTable
    Running: (App) 2024-07-19-194457_App\Database\Migrations>AddScoreEnabledColumnInTournamentsTable
    Running: (App) 2024-07-23-053746_App\Database\Migrations>AddIncrementScoreEnabledColumnInTournamentsTable
    Running: (App) 2024-08-05-060741_App\Database\Migrations>AddVisibilityInTournamentsTable
    Running: (App) 2024-08-20-234333_App\Database\Migrations>AlterDescriptionTypeInTournamentsTable
    Running: (App) 2024-08-28-203140_App\Database\Migrations>AddAvailableColumnsInTournamentsTable
    Running: (App) 2024-08-28-213259_App\Database\Migrations>DropForeignkeys
    Running: (App) 2024-08-29-161127_App\Database\Migrations>AddImageToParticipants
    Running: (App) 2024-08-30-040901_App\Database\Migrations>AlterTableBrackets
    Running: (App) 2024-09-10-082042_App\Database\Migrations>AddSessionIDInParticipants
    Running: (App) 2024-09-18-063134_App\Database\Migrations>AddEvaluationColumnsInTournamentsTable
    Running: (App) 2024-09-19-153047_App\Database\Migrations>AddIncrementTypeInTournamentsTable
    Running: (App) 2024-09-19-184830_App\Database\Migrations>AlterIncrementScoreColumnTypeInTournamentsTable
    Running: (App) 2024-09-20-180251_App\Database\Migrations>CreateVotesTable
    Running: (App) 2024-09-20-180903_App\Database\Migrations>AddVotingRetainColumnInTournamentsTable
    Running: (App) 2024-09-24-004043_App\Database\Migrations>AddAllowHostOverrideColumnInTournamentsTable
    Running: (App) 2024-09-25-093234_App\Database\Migrations>CreateScheduleTable
    Running: (App) 2024-10-01-134424_App\Database\Migrations>CreateTournamentRoundSettingsTable
    Running: (App) 2024-10-14-130416_App\Database\Migrations>DropForeignKeyOnVotesTable
    Running: (App) 2024-10-23-021831_App\Database\Migrations>AddUUIDToVotesTable
    Running: (App) 2024-10-23-024238_App\Database\Migrations>AddMarkByHostToBracketsTable
    Running: (App) 2024-10-25-031046_App\Database\Migrations>AddIsDoubleInVotesTable
    Running: (App) 2024-10-25-050701_App\Database\Migrations>AddIsDoubleInBracketsTable
    Running: (App) 2024-10-30-024629_App\Database\Migrations>AddParticipantImageUpdateEnabledToTournaments
    Running: (App) 2024-11-01-110923_App\Database\Migrations>AddKnockoutFinalInBracketsTable
    Running: (App) 2024-11-01-194205_App\Database\Migrations>AddKnockoutSecondInTournamentSettingsTable

Migrations complete.
```

18. Now move **public** folder to **public\_html** `mv public ..`

19. Navigate to **public\_html** `cd ..`

20. Run `mv public/..}* .` this command moves recursively all public content to public's parent (**public\_html**) then run `la` to verify everything moved as expected.

```
mv public/..}* .
mv: cannot move 'public/.' to './': Device or resource busy
mv: cannot move 'public/..' to './..': Device or resource busy

root@          /home/admin/web/          /public_html (0.119s)
la
.htaccess  ci_tournament_bracket-generator  css  favicon.ico  images  index.php  js  public  robots.txt
```

21. Delete public folder `rm -rf public`

22. Now, open `index.php` with `vim index.php` and edit FCPATH to

```
'./ci_tournament_bracket-generator/app/Config/Paths.php'
// This is the line that might need to be changed, depending on your folder structure.
require FCPATH . './ci_tournament_bracket-generator/app/Config/Paths.php';
// ^^^ Change this line if you move your application folder
```

23. Save and exit `vim` editor

-Now we will link writable/uploads to public\_html/uploads

23. Navigate to uploads `cd ci_tournament_bracket-generator/writable/uploads`, run `pwd` and save that path.

24. Navigate to public\_html `cd .../.../...`

25. Now link upload directory `ln -s /path/to/your/codeigniter/writable/uploads /path/to/your/public_html/`

-Now we need to give some permissions to writable folder.

21. Navigate to writable `cd /project/writable`

22. Give permissions to the whole writable folder `sudo chmod -R 0777 .`

-We will now need to update the uploads structure a little.

26. Navigate to uploads `cd uploads`

27. Run the following command:

```
mkdir audios
mkdir audios/local
mkdir audios/url
mkdir videos
mkdir videos/local
mkdir videos/url
mkdir images
mkdir images/participants
mkdir CSV
mkdir CSV/UserLocal
```

-Finally, the only missing thing is setting up cronjob

26. Type `crontab -e` you will open a crontab config file with vim.

27. Copy and paste the following cronjob  
`* * * * * /usr/bin/php /home/admin/web/tourneyforge.com/public_html/ci_tournament_bracket-generator/spark task:run >> /home/admin/web/tourneyforge.com/public_html/ci_tournament_bracket-generator/cronjob.log 2>&1`

28. Make sure to save and exit, the script above.

## What It Does

-This cron job runs a PHP script to generate and update tournament brackets automatically, making sure everything stays up to date without manual input.

## How Often It Runs

-It runs every minute, so updates happen in real-time.

## How It Works

- Executes the PHP script at:

```
/home/admin/web/tourneyforge.com/public_html/ci_tournament_bracket-generator/spark task:run • Logs all output and errors to: /home/admin/web/tourneyforge.com/public_html/ci_tournament_bracket-generator/cronjob.log
```

## Why the Logging?

All output and errors get saved in cronjob.log, making it easy to track what's happening and troubleshoot if needed.

## Command explanation:

- \* \* \* \* \* → Runs every minute
- /usr/bin/php → Runs the script using PHP
- spark task:run → Calls a task in CodeIgniter
- >> cronjob.log 2>&1 → Saves both output and errors to the log file

## Step 5: WebSocket setup

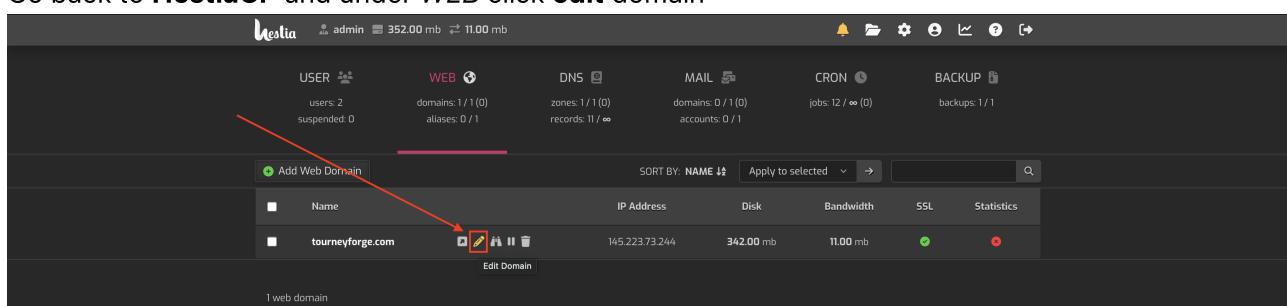
### Context:

Locally, setting up the WebSocket is straightforward; it simply involves running the WebSocket server, and the brackets pop-up feature (which relies on WebSocket) renders the information seamlessly. However, in a production environment where we are deploying to a remote server, the domain must use HTTPS instead of HTTP to provide users with a secure and reliable experience. Since the brackets pop-up feature depends on WebSocket communication and the domain uses HTTPS, the WebSocket connection must also be secure (**WSS** instead of **WS**) to establish bidirectional communication.

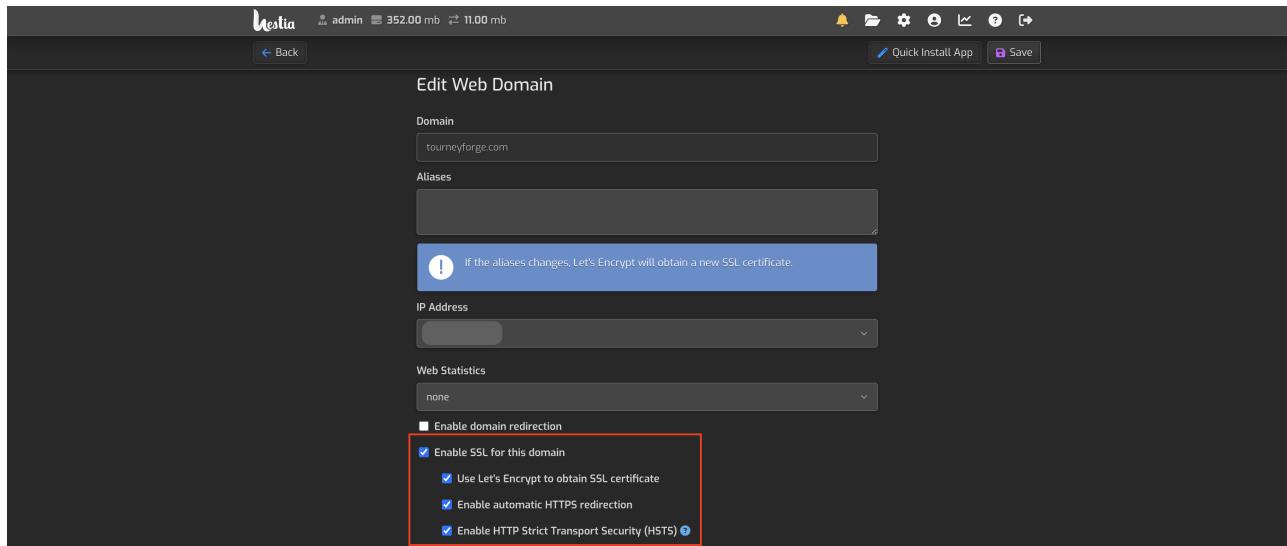
To enable this, we need to configure SSL for the server.

You will need to setup SSL certificates, by default **HestiaCP** provides a *Let's Encrypt to obtain SSL certificate* option, this SSL certificate is already **Authenticated** this will help us to make our domain secure.

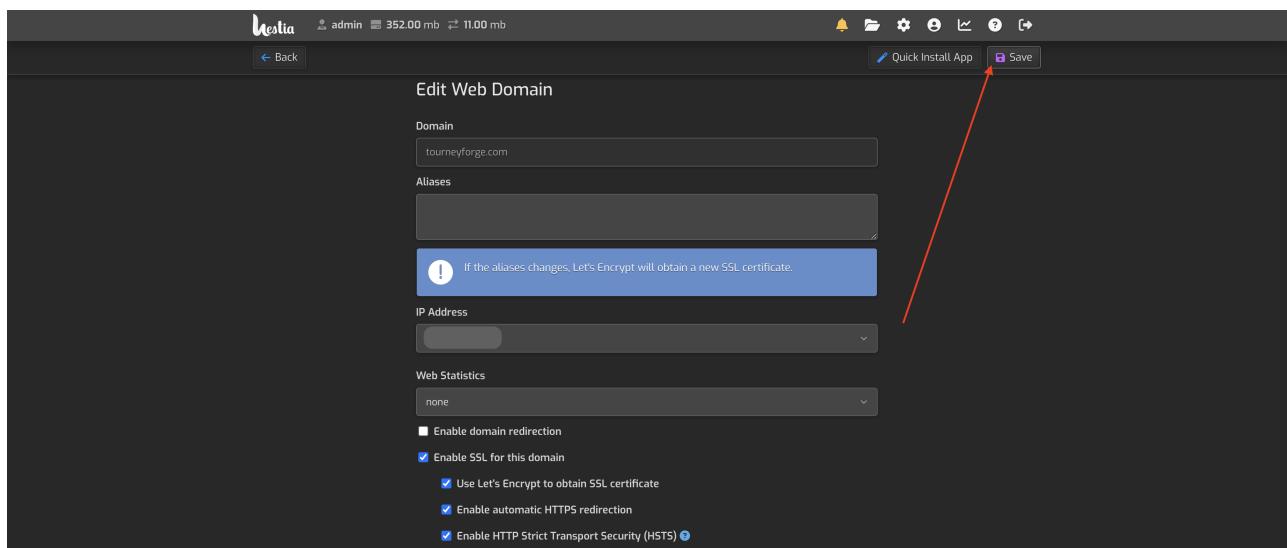
1. Go back to **HestiaCP** and under **WEB** click **edit domain**



2. Enable all 4 options shown below.



3. Click "Save"



4. Now **Go back** to SSH Terminal and you will need to copy the path of your SSL certificates (They are set by default). You can find them under:

`/usr/local/hestia/data/users/admin/ssl/domainname.com.crt` and  
`/usr/local/hestia/data/users/admin/ssl/domainname.com.key`

5. Navigate under the project directory and run `vim ws.php`

6. Edit the `ws.php` as shown below. Don't forget to change the certificates paths

```
<?php

use Ratchet\MessageComponentInterface;
use Ratchet\ConnectionInterface;
use Ratchet\Server\IoServer;
use Ratchet\Http\HttpServer;
use Ratchet\WebSocket\WsServer;
use React\Socket\Server as ReactServer;
use React\Socket\SecureServer;
```

```
require __DIR__ . '/vendor/autoload.php';

class WebSocketsServer implements MessageComponentInterface {
    protected $clients;
    public function __construct() {
        $this->clients = new \SplObjectStorage;
    }
    public function onOpen(ConnectionInterface $conn) {
        $this->clients->attach($conn);
        echo "New connection! {$conn->resourceId}\n";
    }
    public function onMessage(ConnectionInterface $from, $msg) {
        foreach ($this->clients as $client) {
            if ($from !== $client) {
                $client->send($msg);
            }
        }
    }
    public function onClose(ConnectionInterface $conn) {
        $this->clients->detach($conn);
        echo "Connection {$conn->resourceId} has disconnected\n";
    }
    public function onError(ConnectionInterface $conn, \Exception $e) {
        echo "An error has occurred: {$e->getMessage()}\n";
        $conn->close();
    }
}

$cert = '/usr/local/hestia/data/users/admin/ssl/tourneyforge.com.crt';
$key = '/usr/local/hestia/data/users/admin/ssl/tourneyforge.com.key';

$loop = React\EventLoop\Factory::create();
$socket = new ReactServer('0.0.0.0:8092', $loop);

$secureSocket = new SecureServer($socket, $loop, [
    'local_cert' => $cert,
    'local_pk' => $key,
    'allow_self_signed' => true, //Set to false in production
    'verify_peer' => false, //Set to true in production
]);

$wsServer = new HttpServer(
    new WsServer(
        new WebSocketsServer()
    )
);

$server = new Ratchet\Server\IoServer($wsServer, $secureSocket, $loop);
echo "Secure WebSocket server running on wss://localhost:8092\n";
$address = $secureSocket->getAddress();
echo "$address";
$server->run();
```

4. The script above, runs a websocket under the 8092 port by default the hosting already has some ports that receive TCP Requests, however, if we want to add custom ports we will need to configure the firewall with the following command. First we will need to go to our **iptables file** which is under the **etc** directory just run the following command: `cd /etc` then open the `iptables.rules` by running `vim iptables.rules` You will find a file like this:

```
*filter
:INPUT DROP [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:fail2ban-HESTIA - [0:0]
:fail2ban-MAIL - [0:0]
:fail2ban-RECIDIVE - [0:0]
:fail2ban-SSH - [0:0]
:fail2ban-WEB - [0:0]
:hestia - [0:0]
-A INPUT -p tcp -m multiport --dports 80,443 -j fail2ban-WEB
-A INPUT -p tcp -m tcp --dport 8083 -j fail2ban-HESTIA
-A INPUT -p tcp -m multiport --dports 25,465,587,110,995,143,993 -j
fail2ban-MAIL
-A INPUT -p tcp -m tcp --dport 22 -j fail2ban-SSH
-A INPUT -p tcp -m multiport --dports 1:65535 -j fail2ban-RECIDIVE
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -s 145.223.73.244/32 -j ACCEPT
-A INPUT -s 127.0.0.1/32 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 22 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 80,443 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 21,12000:12100 -j ACCEPT
-A INPUT -p udp -m udp --dport 53 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 25,465,587 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 110,995 -j ACCEPT
-A INPUT -p tcp -m multiport --dports 143,993 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8083 -j ACCEPT
-A INPUT -p tcp -m tcp --dport 8092 -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A fail2ban-HESTIA -j RETURN
-A fail2ban-MAIL -j RETURN
-A fail2ban-RECIDIVE -j RETURN
-A fail2ban-SSH -j RETURN
-A fail2ban-WEB -j RETURN
COMMIT
# Completed on Mon Nov 4 06:45:34 2024
```

5. Now you will add at the end of the tcp rules the following rule: `-A INPUT -p tcp -m tcp -- dport 8092 -j ACCEPT`, exit and save the file.
6. You will need to restart the firewall configuration by running the following command `sudo iptables-restore < /etc/iptables.rules`

7. Now you will need to run the ws.php using **tmux**, in order to do that, we will need to install tmux, run **apt install tmux**.
8. Now initialize **tmux** run: **tmux**, now this will create a new instance of a virtual terminal.
9. Make sure you are under the correct directory (wherever ws.php is located). Run: **php ws.php run**
10. Now we have to detach our terminal to the virtual instance (the websocket will continue running). In order to do that you will need to press: **Ctrl-b + d**

The steps above are meant for running the websocket manually, but whenever the server gets rebooted you will need to manually run it again. Follow the next section to make websocket automatically run on reboot.

## Step 6: WebSocket automatization (run on reboot)

In this section we will use resources like **systemd** in order to make the Operating System to always run some scripts to make websocket run.

**It is important to mention what the main script does: It kills ALL processes running on port 8092, usually there shouulnd't be any, but once a port is opened and allowed in our firewall, the Operating System will always try to occupy it. After it kills all processes, it runs the websocket (which we configured before) and sets up the responsible user as root.**

1. **Iptables setup:** If the iptables.rules file is not yet set up or you are unsure, refer to Steps 4, 5, and 6 from the previous section to complete its configuration.

**Note: You do not need to set up the iptables.rules file again if you completed Step 6 of the previous section.**

2. Now, we will need to create a .service file under the **system** direcrotry, in order to do that run: **cd /etc/systemd/system && vim ws-server.service** this command will navigate to the **system** directory and create (or overwrite) a file named **ws-server.service**.
3. Copy and paste the following code:

```
[Unit]
Description=WebSocket Server for TourneyForge
After=network.target

[Service]
Type=simple
ExecStartPre=/bin/bash -c 'fuser -k 8092/tcp || true'
ExecStart/php
/home/admin/web/tourneyforge.com/public_html/ci_tournament_bracket-
generator/ws.php run
Restart=always
User=root
WorkingDirectory=/home/admin/web/tourneyforge.com/public_html/ci_tournamen-
t_bracket-generator
```



6. Type **Y** and then wait, you will have yt-dlp downloaded.
7. Check if it was correctly installed **yt-dlp --version**, Output:

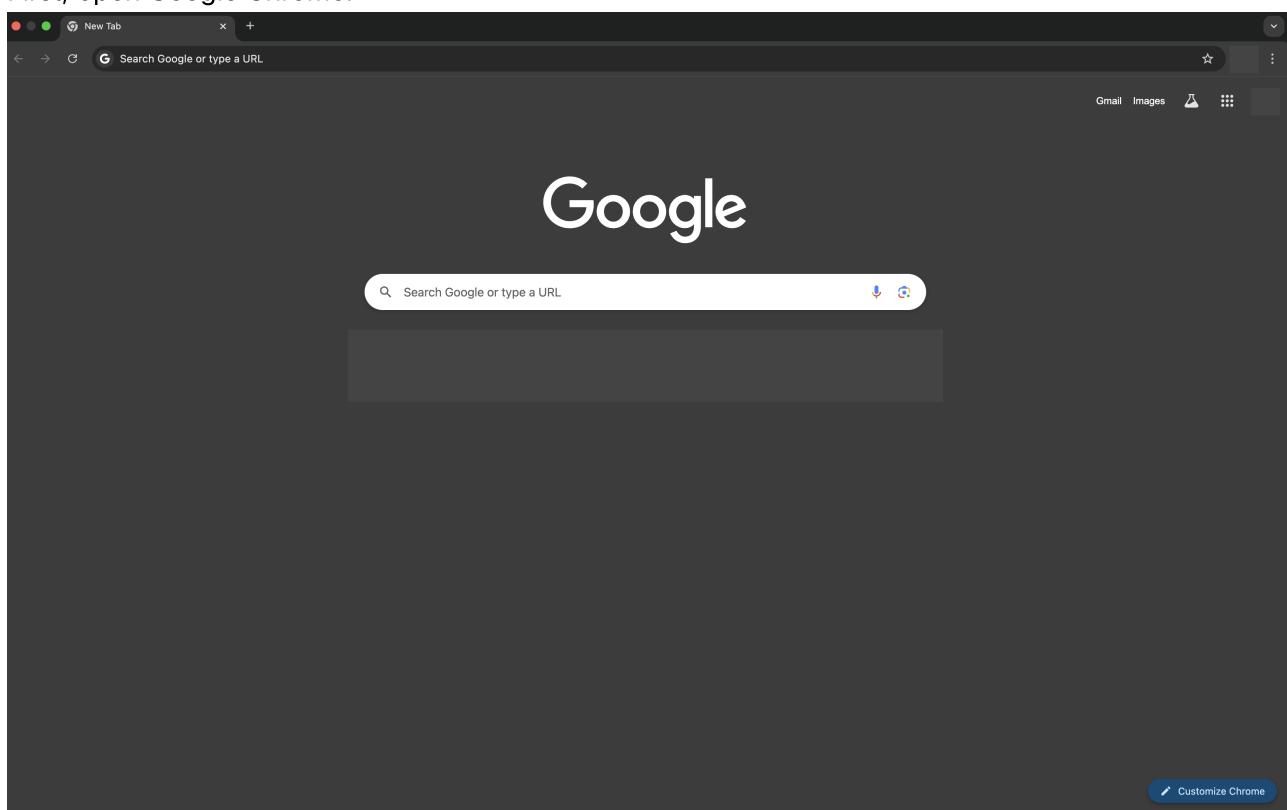
```
yt-dlp --version
```

```
2022.04.08
```

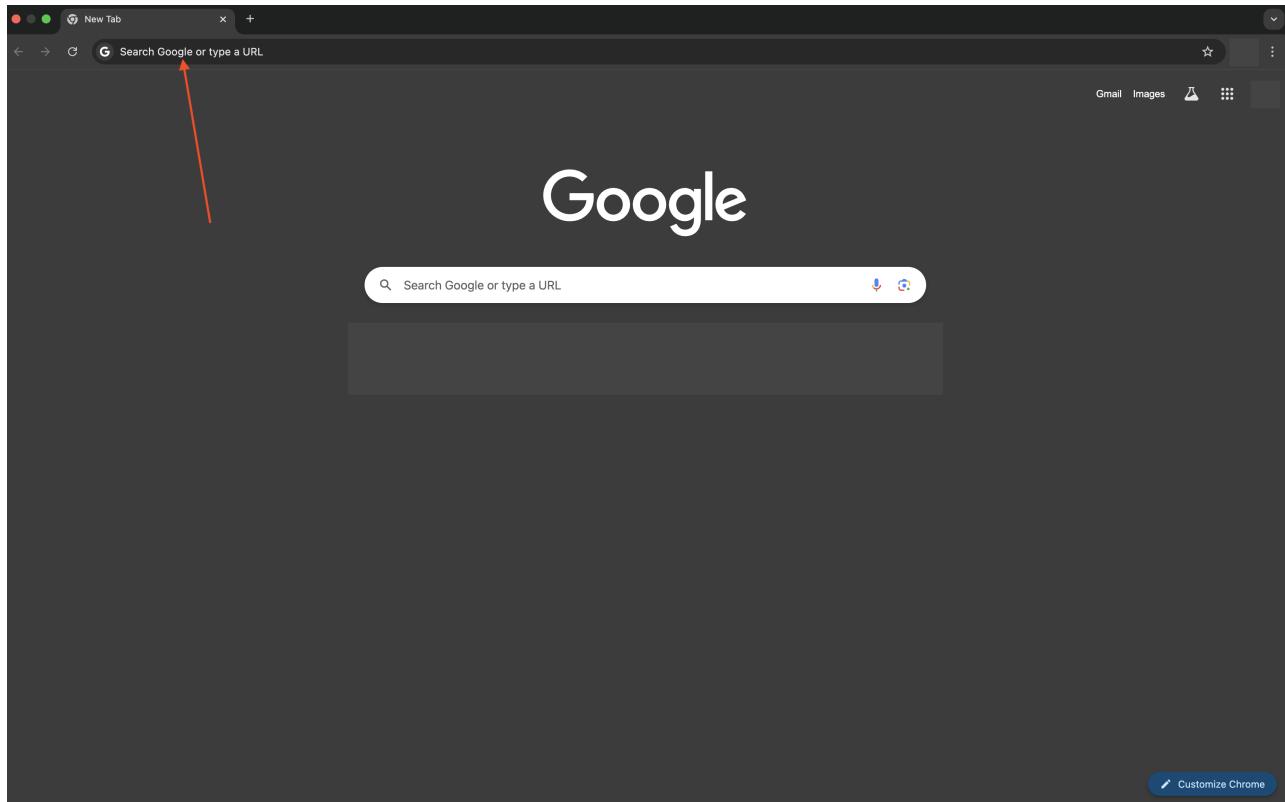
Cookies section:

**In this section you will learn how to copy cookies from any website (Using Chrome)**

5. First, open Google Chrome.



6. Now, copy and paste into the box shown in the image, the following URL:  
<https://chromewebstore.google.com/detail/editthiscookie/jpdpholcdjghlginfadphhefdonkmohg>



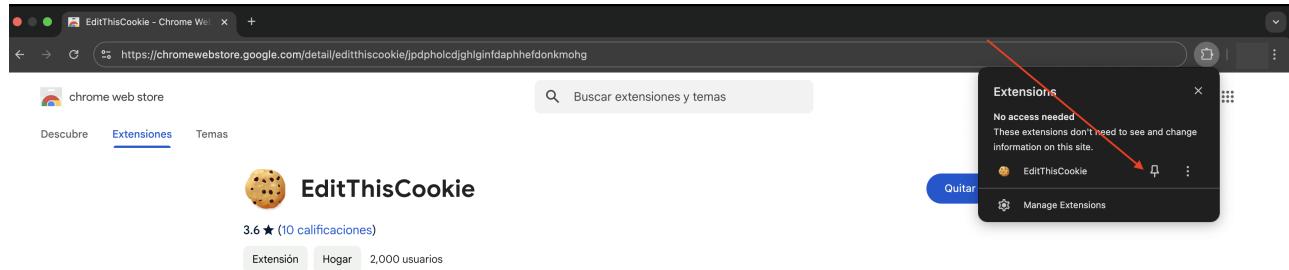
7. After being redirected into Chrome web store, click on "Add to Chrome"

The screenshot shows the Chrome Web Store page for the 'EditThisCookie' extension. The extension has a rating of 3.6 stars (10 reviews) and 2,000 users. A prominent red arrow points to the blue 'Agregar a Chrome' (Add to Chrome) button. Below the extension details, there's a yellow warning box stating: 'Es posible que esta extensión pronto deje de ser compatible porque no sigue las prácticas recomendadas para extensiones de Chrome.' (It's possible that this extension will stop being compatible because it doesn't follow recommended practices for Chrome extensions.) To the right of the warning is a link to 'Más información' (More information). Further down, there's a section titled 'Prueba una de estas alternativas:' (Try one of these alternatives:) featuring four other cookie-related extensions: 'COOKIE EDITOR', 'Cookie Manager', 'Cookie Editor', and 'Broom'.

8. Click on the icon at the top right of the window:

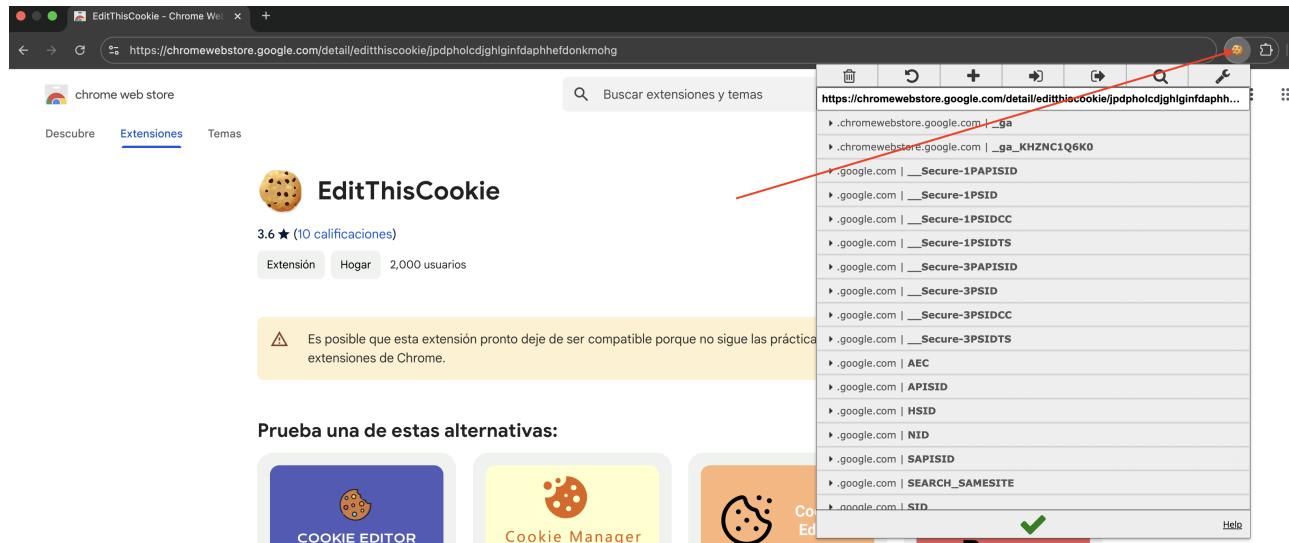
The screenshot shows the same Chrome Web Store page for the 'EditThisCookie' extension. A red arrow points to the top right corner of the window, specifically to the extension icon in the toolbar. A context menu is open, showing options like 'Extensions', 'No access needed', and 'EditThisCookie'. The 'EditThisCookie' option is highlighted with a blue background.

9. Click the pin icon (which will have the Cookies extension pinned at the top right of our browser)

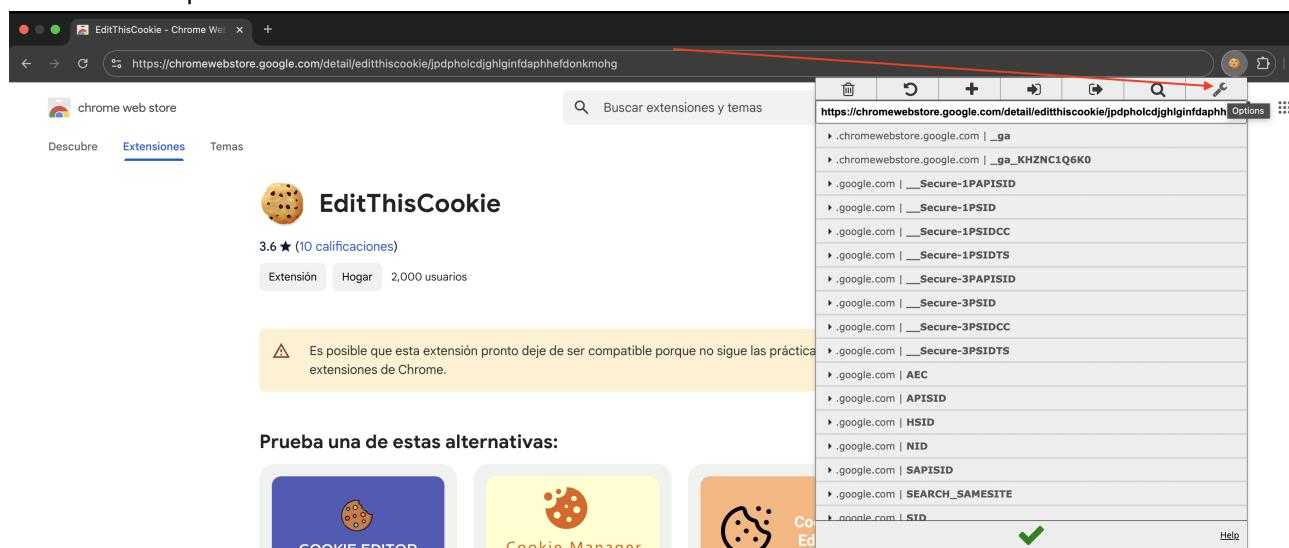


-Now we will need to export the cookies in a certain format that will allow our code to use them

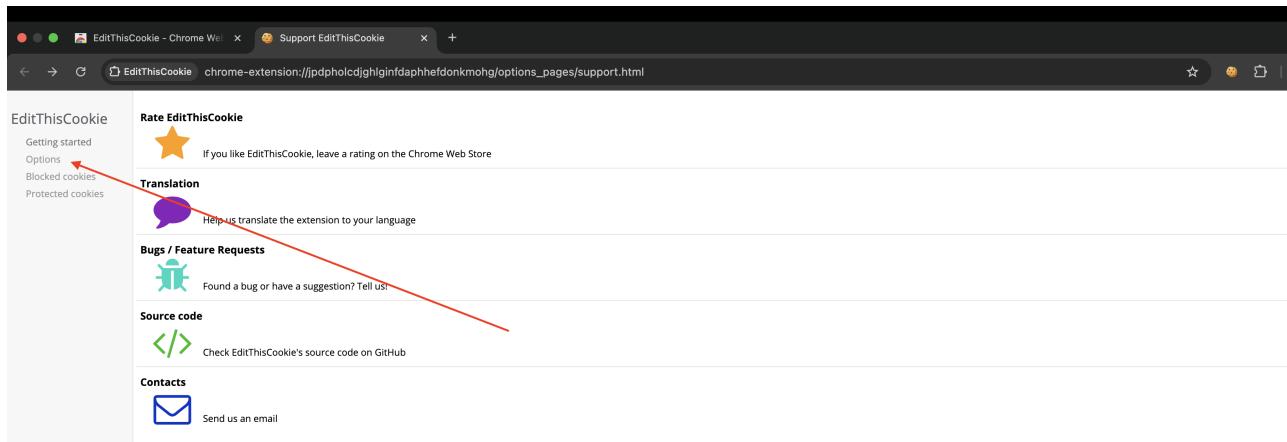
10. Click on the Cookie icon.



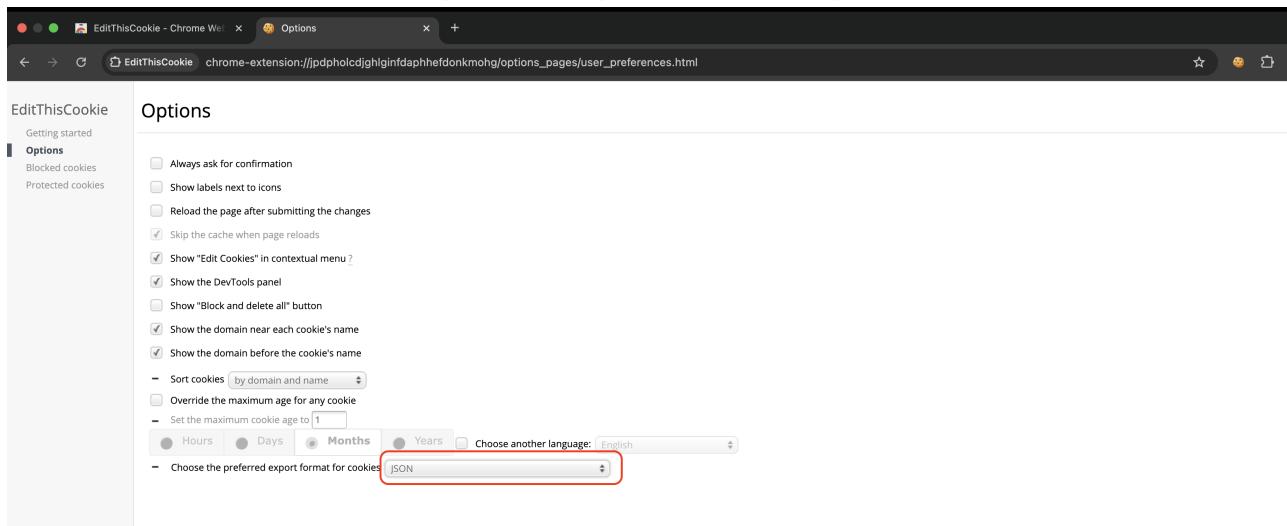
11. Click on the Options icon.



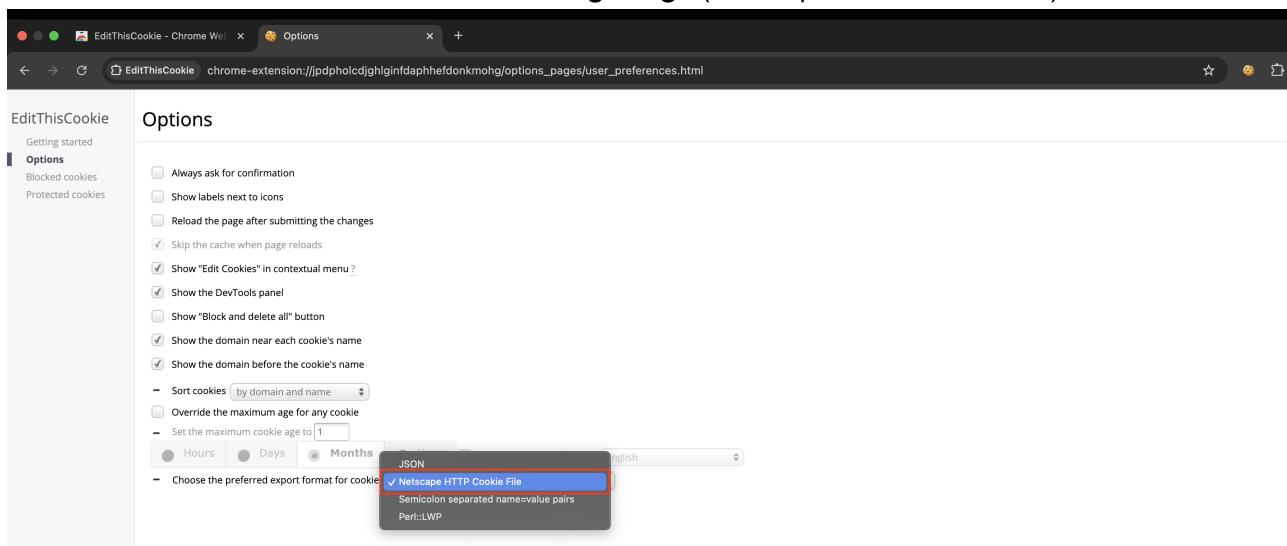
12. You will be redirected to the extension's support page. Click **options**



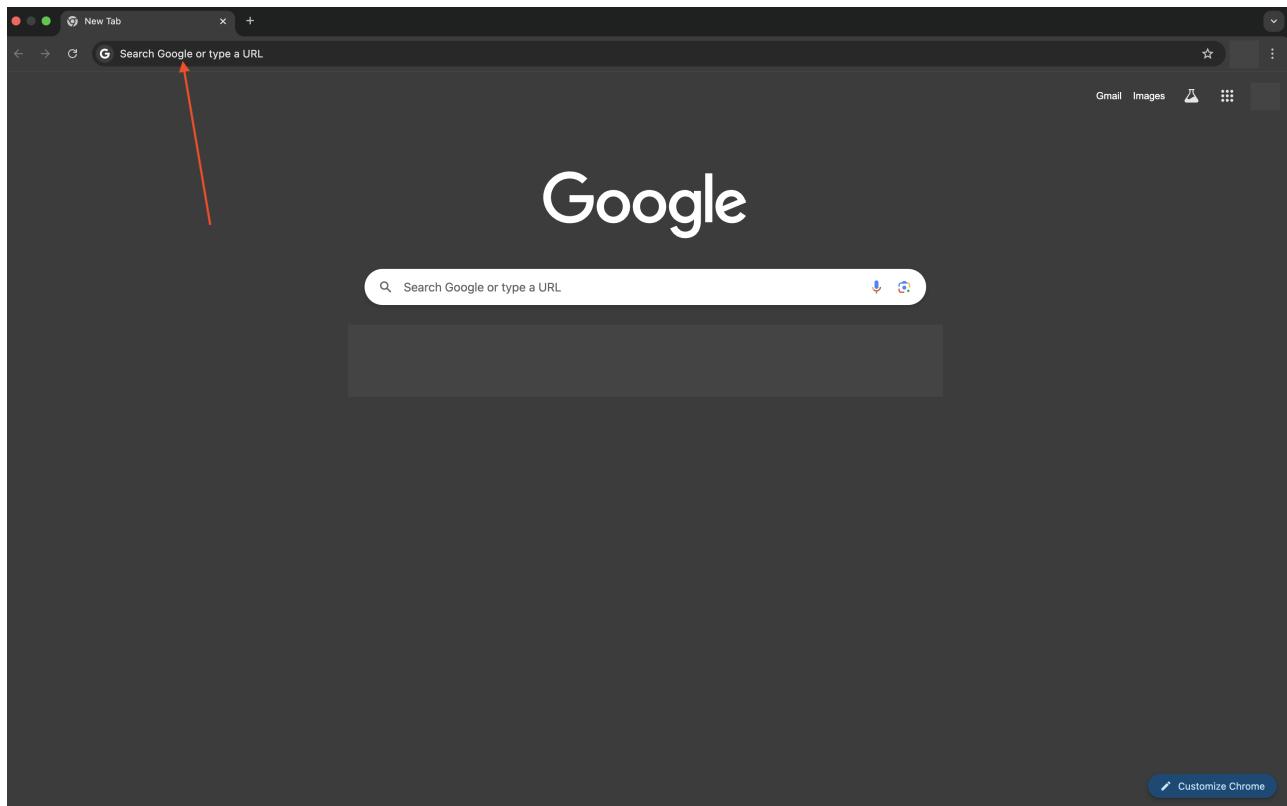
13. Now, you will choose a different export format (it's json by default). Click on the box shown below



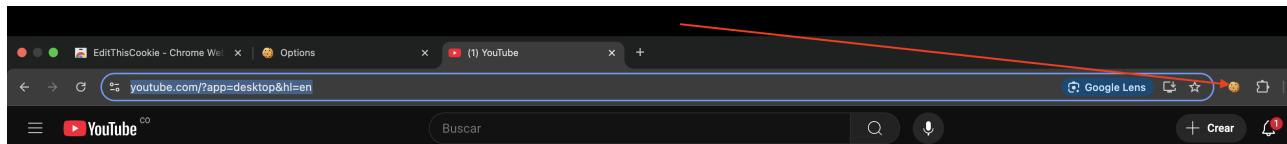
14. Now click on the format shown in the **following image** (Netscape HTTP Cookie file)



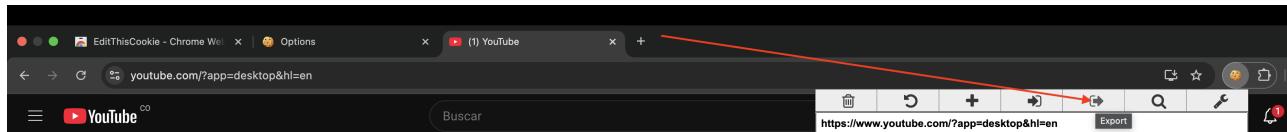
15. You can now open a new tab and go to the following link, by clicking on it or copy and paste it into your browser <https://www.youtube.com/?app=desktop&hl=en>



16. Once on Youtube, click the **cookie icon** at the **top right** of the screen



17. Then, click on Export. You will get the following message - *Cookies copied to clipboard*



18. Save the Cookies wherever you want as **www.youtube.com\_cookies.txt.**

19. Save that path for later use.

End of Cookies section

20. Go back to your ssh terminal, we will be editing two files into the project, the first one is `UploadConfig.php` which is under: `ci_tournamet_bracket-generator/app/Config/UploadConfig.php`

21. Navigate to `cd yourProjectpath/app/Config/`

22. Type this command `which ffmpeg && which yt-dlp`, that will output both `ffmpeg` and `yt-dlp` paths. That will help us for next steps. **Note: both paths should normally be the same**

23. Open `UploadConfig.php` `vim UploadConfig.php`, make sure to edit the `ffmpegPath` and `cookiesPath`, to yours.

```
<?php
```

```

namespace Config;

use CodeIgniter\Config\BaseConfig;

class UploadConfig extends BaseConfig
{
    public $localAudioUploadPath = 'audios/local/';
    public $urlAudioUploadPath = 'audios/url/';
    public $localVideoUploadPath = 'videos/local/';
    public $urlVideoUploadPath = 'videos/url/';
    public $descriptionImagesUploadPath = 'images/description/';
    public $participantImagesUploadPath = 'images/participants/';
    public $csvUploadPath = 'CSV/UserLocal/';
    public $ffmpegPath = '/usr/local/bin/';

}

```

- Development was meant for a windows server, which means some of the paths written into the code shouldn't be the same as a ubuntu server. that is why we will modify the **process** function under TournamentController.php. (ci\_tournamet\_bracket-generator/app/Controllers/Api/TournamentController.php.php)

24. Type: **cd ..**

25. Navigate to the path: **cd Controllers/Api/**

26. Open the file we want to edit: **vim TournamentController.php**

27. Look for the **process** function and modify it with the following updated version of it:

```

public function process($youtubeLink, $type = 'audio')
{
    $uploadConfig = new UploadConfig();

    parse_str( parse_url( $youtubeLink, PHP_URL_QUERY ), $vars );

    if (isset($vars['v'])) {
        $video_id = $vars['v'];
    }

    if (isset($vars['si'])) {
        $video_id = $vars['si'];
    }

    $yt = new Ytubedl();
    $yt->setBinPath($uploadConfig->ffmpegPath . 'yt-dlp');
    if ($type == 'audio') {
        if (file_exists(WRITEPATH . "uploads/$uploadConfig->urlAudioUploadPath/" . $video_id . '.mp3')) {
            return $video_id . '.mp3';
        }
    }
}

```

```

$collection = $yt->download(
    Options::create()
        ->downloadPath(WRITEPATH . "uploads/$uploadConfig-
>urlAudioUploadPath")
        ->extractAudio(true)
        ->audioFormat('mp3')
        ->audioQuality('0') // best
        ->output($video_id)
        ->url($youtubeLink)
        ->cookies($uploadConfig->ffmpegPath . 'cookies.txt')
);

$filetype = '.mp3';
} else {
    if (file_exists(WRITEPATH . "uploads/$uploadConfig-
>urlVideoUploadPath/" . $video_id . '.mp4')) {
        return $video_id . '.mp4';
    }

    $collection = $yt->download(
        Options::create()
            ->downloadPath(WRITEPATH . "uploads/$uploadConfig-
>urlVideoUploadPath")
            ->format('mp4')
            ->output($video_id . '.mp4')
            ->url($youtubeLink)
            ->cookies($uploadConfig->ffmpegPath . 'cookies.txt')
    );

    $filetype = '.mp4';
}

foreach ($collection->getVideos() as $video) {
    if ($video->getError() !== null) {
        echo "Error downloading video: {$video->getError()}." ;
        log_message('debug', $video->getError());
    }
}

return $video_id . $filetype;
}

```

### Change notes:

- We replaced `$yt->setBinPath($uploadConfig->ffmpegPath . 'bin\yt-dlp.exe');` to `$yt->setBinPath($uploadConfig->ffmpegPath . 'yt-dlp');` That's because we needed to set that command for a Ubuntu OS so there is no .exe file for yt-dlp. Also made a little change to the path command (since we also changed the paths a few steps before)
- We changed `->cookies($uploadConfig->ffmpegPath . 'www.youtube.com_cookies.txt')` to `->cookies($uploadConfig->ffmpegPath .`

'cookies.txt') I just wanted name to be shorter lol

**Changes weren't as significant but since you are using vim and changing a specific function is just easier for you to replace the whole function by copying and pasting.**

Congrats! you have successfully set up the project.