USING CRON AND WGET TO HIT A PHP SCRIPT

Version 1.0 Patrick Brunswyck

a manual by

all2all

Moving Art Studio a.s.b.l.

all2all .beagent RIPE NCC Member

Table of Contents

Using cron and wget to hit a php script

What is cron

Cron is a **time-based job scheduler** in Unix-like computer operating systems. **Cron** is short for Chronograph.

Cron enables users to **schedule jobs** (commands or scripts) to run automatically at a certain time or date.

Cron syntax

```
.---------------- minute (0 - 59)
|  .------------- hour (0 - 23)
|  |  .---------- day of month (1 - 31)
|  |  |  .------- month (1 - 12) OR jan,feb,mar,apr ...
|  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7)  OR sun,mon,tue,wed,thu,fri,sat
|  |  |  |  |
*  *  *  *  *  command to be executed
```

More on the syntax: http://en.wikipedia.org/wiki/Cron#crontab_syntax

The wget command

Hitting a PHP script can be achieved by using cron and **wget** with no output; just hit it and die. To achieve this put the following in **/etc/crontab**:

- * * * * * **wget -q** http://mysite.be/index.php **> /dev/null 2>&1**

Wget's **-q** or –**quiet** option turns off wget's output which is exactly what we want since we do not intend to generate content but only **hit** the PHP script.

A common practice when adding entries to crontab is to end the entry like this:

- **>/dev/null 2>&1**

The purpose of this is to suppress any output from the command itself, because we're not interested.

- The first part **>/dev/null**:

Means redirect STDOUT (the standard output stream) to /dev/null (which is basically a blackhole for bits).

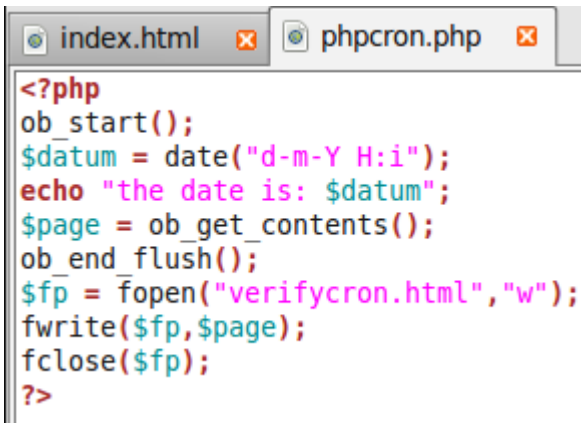
- The second part **2>&1**:

Means redirect STDERR (standard error stream) to the same place as STDOUT (which was just specified). STDOUT has the assigned number 1 and STDERR has the assigned number 2.
This way both STDOUT (1) and STDERR (2) are directed to /dev/null and **all output of the cronned command is suppressed**.

Using cron with Virtualmin

Create a little test script

I have created a little PHP script I named **phpcron.php**. This script will generate a **verifycron.html** file to verify that cron executed the scheduled job as configured:



<?php
ob_start();
$datum = date("d-m-Y H:i");
echo "the date is: $datum";
$page = ob_get_contents();
ob_end_flush();
$fp = fopen("verifycron.html","w");
fwrite($fp,$page);
fclose($fp);
?>

This script will capture all output and store it into a new file called **verifycron.html**. By using the date function we can verify the exact time the verifycron.html page was created. (you can of course just create a simple file and check the timestamp to see when it was created) This way we can be sure the cron scheduled job works.

Just confirm that your script can write to the directory you are going to save your (verifycron.html) file to and is executable (as with any script).

Goal:

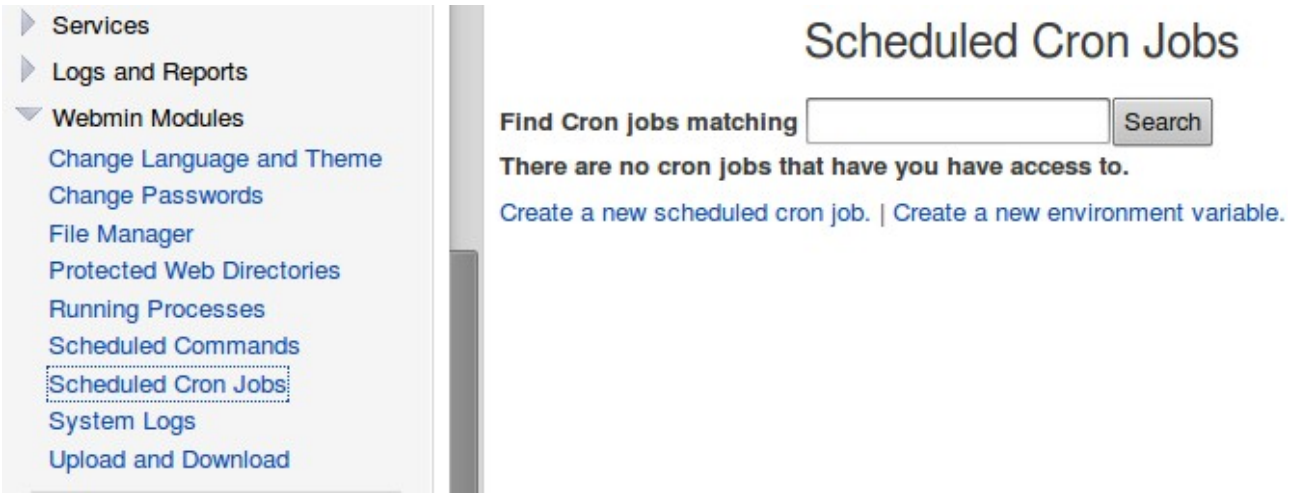
- To have cron hit this page and make it create a verifycron.html file (In this example in the directory: /var/www/htdocs/patrick/public).
  Using this example as a testcase for more useful purposes.

- So in this example the command to be configured in cron is:
  wget -q http://patrick.all2all.org/phpcron.php 2>&1

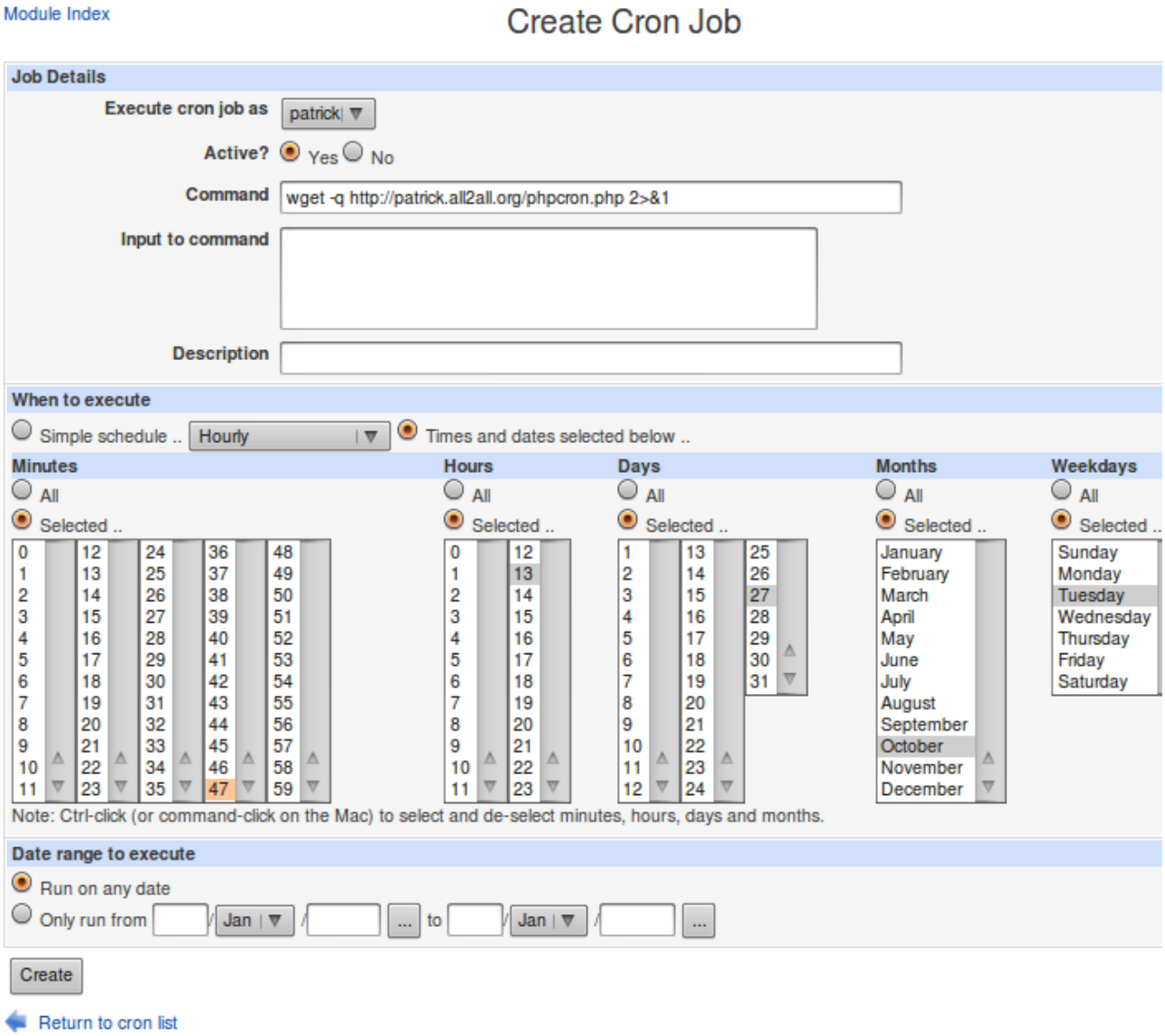
- This wil **hit** the phpcron.php page, execute the script in it and create a verifycron.html page that will show the exact date and time of creation and thus confirm that cron executed my script as configured.

Creating a scheduled cron job

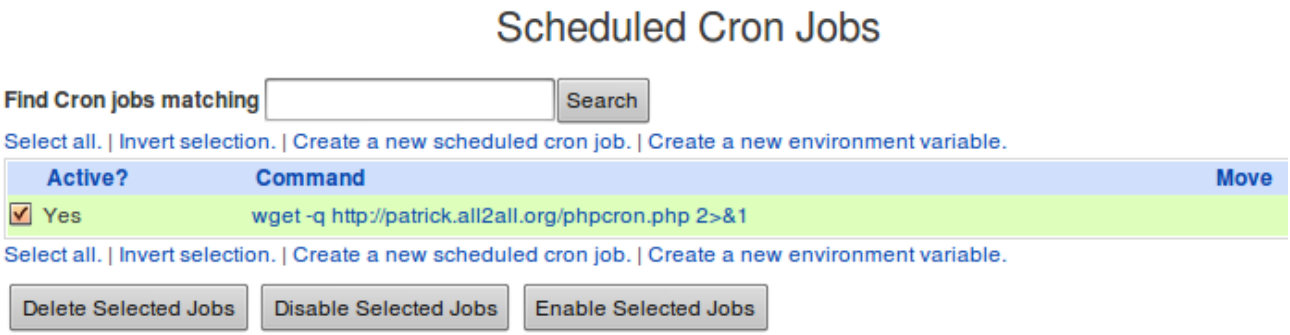
Go to **Virtualmin** click on **Webmin Modules** and then on **Scheduled Cron Jobs**



Now click on **Create a new scheduled cron job**



Now enter the command, schedule it and then click the **create** button.



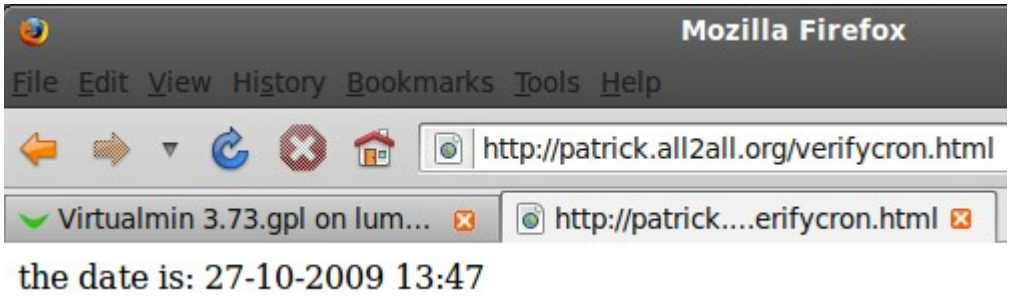
You can now see the scheduled job and others if created. Check the **active** box and click on **Enable Selected Jobs**, it now writes the job to cron.

In this example the **wget -q http://patrick.all2all.org/phpcron.php 2>&1** command will be run:

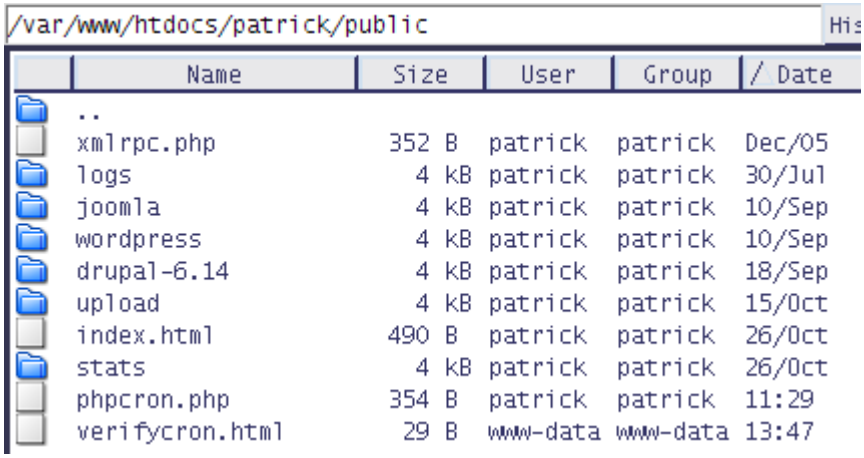
On Tuesday October the 27th at 13hours and 47minutes
Crontab syntax: **47 13 27 10 2 wget -q http://patrick.all2all.org/phpcron.php 2>&1**

I should now see a newly created verifycron.html file in my public directory at 13h47 thanks to the scheduled job!

Indeed when visiting http:/patrick.all2all.org/**verifycron.html**, the by PHP created page shows the correct date and time for the cron job we scheduled:

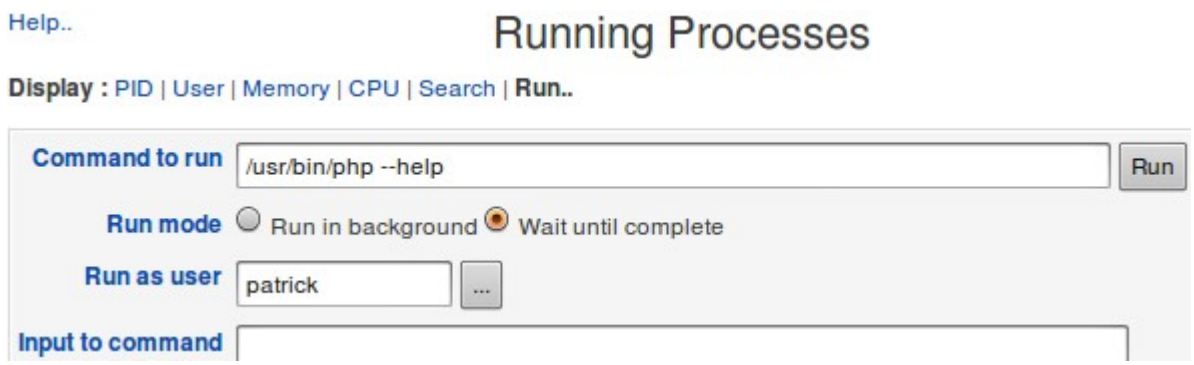


The Virtualmin filemanager confirms the correct time and date too:

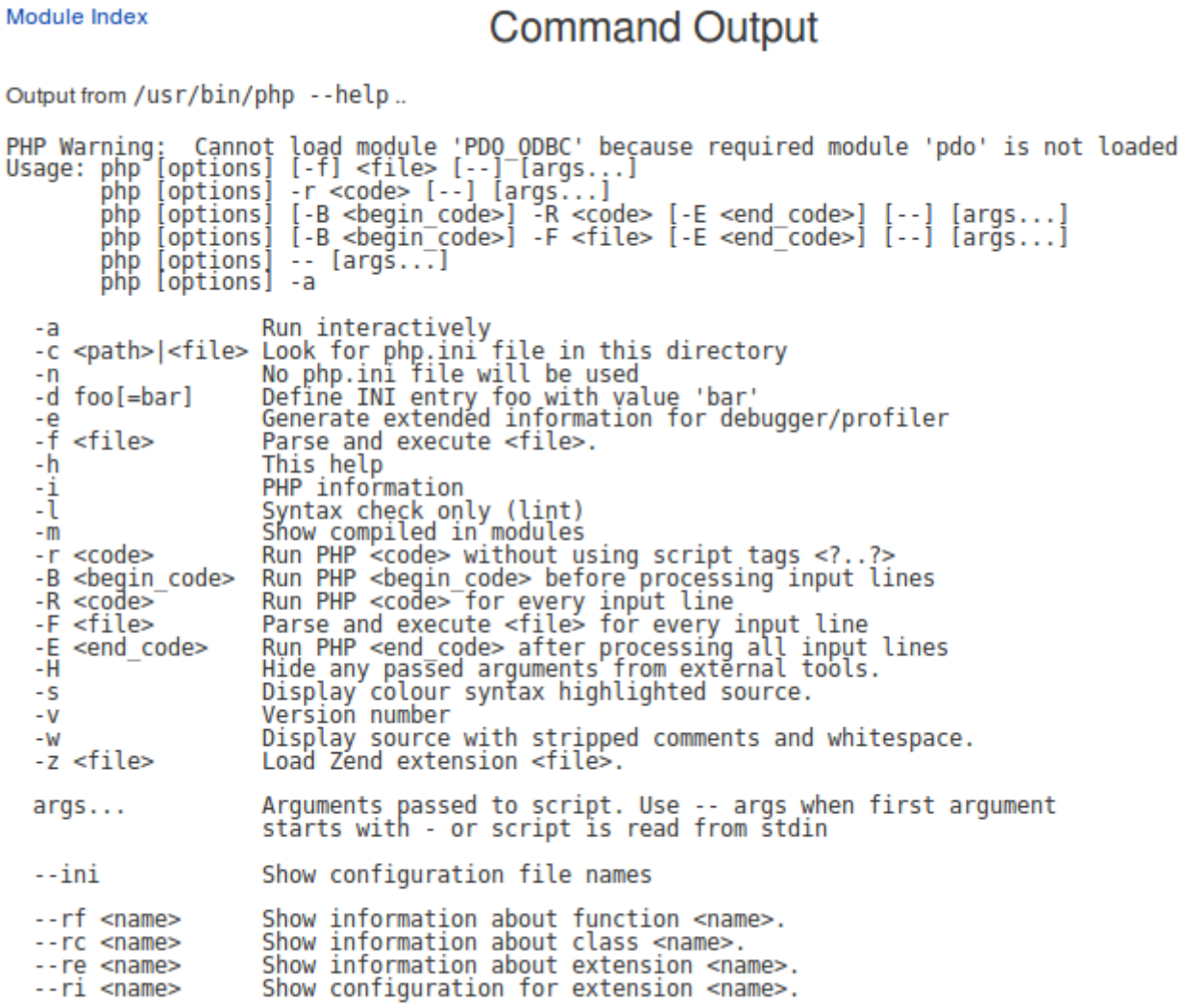


Note that you can test your commands before configuring them as a cron job. To do so click on **Virtualmin** then on **Webmin Modules** and finally on **Running Processes**:

E.g. I will enter and run a command to call help for the PHP binary:



This command results in:



Running PHP cron jobs

Scheduled tasks are a common feature in modern web applications. From cleaning out caches every 24 hours to checking subscription periods and even generating reports, more web applications live by the clock than ever before.

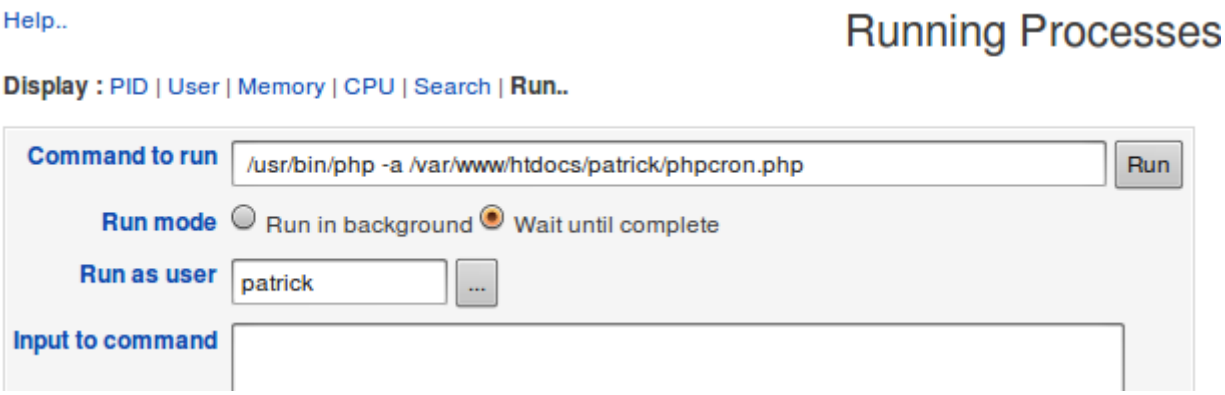
- You can call your PHP scripts via cron using the **PHP binary**. Say your scripts are in the **/var/www/htdocs/mysite/scripts** directory. Your PHP binary is in **/usr/bin/php**, Your command to run your script should be this:

/usr/bin/php /var/www/htdocs/mysite/scripts/runmyscript.php

I will use the phpcron.php file with the date function again to demonstrate how to run a PHP cron job:

- Test your command in **Virtualmin=>Webmin Modules=>Running Processes**:

  /usr/bin/php /var/www/htdocs/patrick/phpcron.php



It shows the date like it should so I can safely configure it as a cron job, knowing the PHP syntax is correct.

Create the Cron Job for your command as shown here.

Versions

Version number	Modifications	Author
1.0 EN	Original version	Patrick Brunswyck