CRON

TABLE OF CONTENTS

Overview

Cron is a time-based task scheduler that executes commands at specified intervals, called ***cron jobs***. Cron is available on Unix, Linux, and Mac servers, and Windows servers use a Scheduled Task to execute commands. Cron jobs are used in Drupal to handle maintenance tasks such as cleaning up log files and checking for updates.

How does it work?

Drupal provides an automated cron system that works with all operating systems because it does not involve the operating system's cron daemon. Instead, it works by checking at the end of each Drupal request to see when the cron last ran. If it has been too long, cron tasks are processed as part of that request.

Module automated_cron subscribes to the onTerminate event for request. You can read more about how this component works in Symfony [here].

File *core/modules/automated_cron/src/EventSubscriber/AutomatedCron.php*:

```
/**
 * Registers the methods in this class that should be listeners.
 *
 * @return array
 *   An array of event listener definitions.
 */
public static function getSubscribedEvents(): array {
  return [KernelEvents::TERMINATE => [['onTerminate', 100]]];
}
```

Drupal then keeps track of when the cron ran and ensures that the next time it runs is only after the configured amount of time has elapsed.

```
/**
 * Run the automated cron if enabled.
 *
 * @param \Symfony\Component\HttpKernel\Event\TerminateEvent $event
 *   The Event to process.
 */
public function onTerminate(TerminateEvent $event): void {
  $interval = $this->config->get('interval');
  if ($interval > 0) {
    $cron_next = $this->state->get('system.cron_last', 0) + $interval;
    if ((int) $event->getRequest()->server->get('REQUEST_TIME') > $cron_next) {
      $this->cron->run();
    }
  }
}
```

So, in essence, if the cron is set to run every hour but the next visitor only comes in three hours, it will only run then.

Enable Drupal Cron

- One way to enable cron is through the administration page. By default, Drupal has a built-in core automated cron system that manages cron. You can access this system by navigating to ***Configuration > System > Cron*** (/admin/config/system/cron). If you have just installed Drupal, this option should be enabled by default. You can confirm this by checking the status of the Automated Cron module at /admin/modules.
- Another way to enable cron is to run it manually from the ***Reports > Status report page***. By default, cron runs every 3 hours, but you can change this to run every hour or every 6 hours. You can also use contributed modules for additional cron functions.
- To run cron using Drush, open a terminal or command prompt and navigate to your Drupal site's root directory. Then, enter the command drush cron. This will run cron for your site.

The cron command

To get Drupal to take care of its maintenance you should have the server execute Drupal's cron periodically. This is done by logging in to the server directly and settings the crontab file.

Crontab (CRON TABle) - is a text file that contains the schedule of cron entries to be run at specified times This file can be created and edited either through the command line interface.

In the following example, the crontab command shown below will activate the cron tasks automatically on the hour:

```
0 * * * * wget -O - -q -t 1 http://www.example.com/cron/<key>
```

In the above sample, the 0 * * * * represents when the task should happen. The first figure represents minutes – in this case, on the 'zero' minute, or top of the hour. The other figures represent the hour, day, month, and day of the week. A * is a wildcard, meaning 'every time'. The minimum is every minute * * * * *.

The rest of the line wget -O - -q -t 1 tells the server to request a URL, so the server executes the cron script.

Here is a diagram of the general crontab syntax, for illustration:

```
# +-------------- minute (0 - 59)
# |  +----------- hour (0 - 23)
# |  |  +-------- day of the month (1 - 31)
# |  |  |  +----- month (1 - 12)
# |  |  |  |  +-- day of the week (0 - 6) (Sunday=0)
# |  |  |  |  |
  *  *  *  *  *   command to be executed
```

Thus, the cron command example above means ping http://www.example.com/cron/<key> at the zero minutes on every hour of every day of every month of every day of the week.

Setting up cron

To edit a crontab through the command line, type:

1. At the Linux command prompt, type: `sudo crontab -e`
2. Add ONE of the following lines:

```
45 * * * * wget -O - -q -t 1 http://www.example.com/cron/<key>
45 * * * * curl -s http://example.com/cron/<key>
```

   This would have a wget or curl visit your cron page 45 minutes after every hour.

3. Save and exit the file. Check the Drupal status report, which shows the time of the cron execution.

> **NOTE**
> Use crontab guru - it's a quick and easy editor for cron schedule expressions.

Disable Drupal cron

For performance reasons, or if you want to ensure that cron can only ever run from an external trigger (not from Drupal), it may be desirable to disable Drupal's automated cron system, in one of three ways:

1. The preferred way to disable Drupal's core `automated cron` module is by unchecking it at `/admin/modules`.
2. To temporarily disable cron, set the 'Run cron every' value to 'Never' (e.g., at ***Administration > Configuration > System > Cron*** (`/admin/config/system/cron`)).
3. For advanced reasons, another way to disable cron in Drupal is to add the following line to your settings.php. Note that this fixes the setting at `/admin/config/system/cron` to 'Never', and administrative users cannot override it.

```
$config['automated_cron.settings']['interval'] = 0;
```

hook_cron()

Gets fired every time the cron runs, so basically, Drupal's cron is a collection of function calls to various modules. For this reason, we must avoid overloading the request with heavy processing; otherwise, the request might crash.

```
function announcements_feed_cron() {
  $config = \Drupal::config('announcements_feed.settings');
  $interval = $config->get('cron_interval');
  $last_check = \Drupal::state()->get('announcements_feed.last_fetch', 0);
  $time = \Drupal::time()->getRequestTime();
  if ($time - $last_check > $interval) {
    \Drupal::service('announcements_feed.fetcher')->fetch(TRUE);
    \Drupal::state()->set('announcements_feed.last_fetch', $time);
  }
}
```

Common inquiries regarding cron jobs

When did the cron job last run?

We can use this in .module files (which don't allow dependency injection) in this way.

```
// Find out when cron was last run; the key is 'system.cron_last'.
$cron_last = \Drupal::state()->get('system.cron_last');
```

Or in another file, we need to use dependency injection.

```
$cron_last = $this->state->get('system.cron_last')
```

How to stop Cron from continuously executing things?

To stop cron from endlessly executing pending cron tasks truncate the queue table e.g. if you have queued up work such as in the salesforce module.

Resolving the ip and name for cron

Here is a Drupal cron job on a prod server where it uses a `--resolve` param to resolve the IP and the name. This task runs every 15 minutes.

```
*/15 * * * * curl -svo /dev/null http://prod.ddd.test.gov:8080/cron/<key> --resolve prod.ddd.test.gov:8080:201.86.28.12
```

Resources:

-

Back to top

Page last modified: May 2 2023.

Edit this page on GitHub