## Post-installation

After installation, auto-cpufreq is available as a binary. Refer to [auto-cpufreq modes and options](https://github.com/AdnanHodzic/auto-cpufreq" \l "auto-cpufreq-modes-and-options) for detailed information on how to run and configure auto-cpufreq.

## Configuring auto-cpufreq

auto-cpufreq makes all decisions automatically based on various factors such as CPU usage, temperature, and system load. However, it's possible to perform additional configurations:

### 1: power\_helper.py script (Snap package install **only**)

When installing auto-cpufreq via [auto-cpufreq-installer](https://github.com/AdnanHodzic/auto-cpufreq" \l "auto-cpufreq-installer), if it detects the [GNOME Power Profiles service](https://twitter.com/fooctrl/status/1467469508373884933) is running, it will automatically disable it. Otherwise, that daemon will cause conflicts and various other performance issues.

However, when auto-cpufreq is installed as a Snap package it's running as part of a container with limited permissions, hence it's highly recommended to disable the GNOME Power Profiles daemon using the power\_helper.py script.

**Please Note:**  
The [power\_helper.py](https://github.com/AdnanHodzic/auto-cpufreq/blob/master/auto_cpufreq/power_helper.py) script is located within the auto-cpufreq repo at auto\_cpufreq/power\_helper.py. In order to access it, first clone the repository:

git clone https://github.com/AdnanHodzic/auto-cpufreq

Navigate to the directory where power\_helper.py resides:

cd auto-cpufreq/auto\_cpufreq

Make sure to have psutil Python library installed before next step:

sudo python3 -m pip install psutil

Then disable the GNOME Power Profiles daemon:

sudo python3 power\_helper.py --gnome\_power\_disable

### 2: --force governor override

By default, auto-cpufreq uses balanced mode which works best for many systems and situations.

However, you can override this behaviour by switching to performance or powersave mode manually. The performance mode results in higher default frequencies, but also higher energy use (battery consumption) and should only be used if maximum performance is needed. The powersave mode does the opposite and extends battery life to its maximum.

See [--force](https://github.com/AdnanHodzic/auto-cpufreq" \l "overriding-governor) flag for more info.

### 3: auto-cpufreq config file

You can configure separate profiles for the battery and power supply. These profiles will let you pick which governor to use, as well as how and when turbo boost is enabled. The possible values for turbo boost behavior are always, auto, and never. The default behavior is auto, which only activates turbo during high load.

By default, auto-cpufreq does not use a config file. If you wish to configure auto-cpufreq statically, we look for a configuration file in the following order:

1. Commandline argument: --config <FILE> if passed as commandline argument to auto-cpufreq
2. User-specific configuration: $XDG\_CONFIG\_HOME/auto-cpufreq/auto-cpufreq.conf
3. System-wide configuration: /etc/auto-cpufreq.conf

#### Example config file contents

# settings for when connected to a power source

[charger]

# see available governors by running: cat /sys/devices/system/cpu/cpu0/cpufreq/scaling\_available\_governors

# preferred governor

governor = performance

# EPP: see available preferences by running: cat /sys/devices/system/cpu/cpu0/cpufreq/energy\_performance\_available\_preferences

energy\_performance\_preference = performance

# EPB (Energy Performance Bias) for the intel\_pstate driver

# see conversion info: https://www.kernel.org/doc/html/latest/admin-guide/pm/intel\_epb.html

# available EPB options include a numeric value between 0-15

# (where 0 = maximum performance and 15 = maximum power saving),

# or one of the following strings:

# performance (0), balance\_performance (4), default (6), balance\_power (8), or power (15)

# if the parameter is missing in the config and the hardware supports this setting, the default value will be used

# the default value is `balance\_performance` (for charger)

# energy\_perf\_bias = balance\_performance

# Platform Profiles

# https://www.kernel.org/doc/html/latest/userspace-api/sysfs-platform\_profile.html

# See available options by running:

# cat /sys/firmware/acpi/platform\_profile\_choices

# platform\_profile = performance

# minimum cpu frequency (in kHz)

# example: for 800 MHz = 800000 kHz --> scaling\_min\_freq = 800000

# see conversion info: https://www.rapidtables.com/convert/frequency/mhz-to-hz.html

# to use this feature, uncomment the following line and set the value accordingly

# scaling\_min\_freq = 800000

# maximum cpu frequency (in kHz)

# example: for 1GHz = 1000 MHz = 1000000 kHz -> scaling\_max\_freq = 1000000

# see conversion info: https://www.rapidtables.com/convert/frequency/mhz-to-hz.html

# to use this feature, uncomment the following line and set the value accordingly

# scaling\_max\_freq = 1000000

# turbo boost setting. possible values: always, auto, never

turbo = auto

# settings for when using battery power

[battery]

# see available governors by running: cat /sys/devices/system/cpu/cpu0/cpufreq/scaling\_available\_governors

# preferred governor

governor = powersave

# EPP: see available preferences by running: cat /sys/devices/system/cpu/cpu0/cpufreq/energy\_performance\_available\_preferences

energy\_performance\_preference = power

# EPB (Energy Performance Bias) for the intel\_pstate driver

# see conversion info: https://www.kernel.org/doc/html/latest/admin-guide/pm/intel\_epb.html

# available EPB options include a numeric value between 0-15

# (where 0 = maximum performance and 15 = maximum power saving),

# or one of the following strings:

# performance (0), balance\_performance (4), default (6), balance\_power (8), or power (15)

# if the parameter is missing in the config and the hardware supports this setting, the default value will be used

# the default value is `balance\_power` (for battery)

# energy\_perf\_bias = balance\_power

# Platform Profiles

# https://www.kernel.org/doc/html/latest/userspace-api/sysfs-platform\_profile.html

# See available options by running:

# cat /sys/firmware/acpi/platform\_profile\_choices

# platform\_profile = low-power

# minimum cpu frequency (in kHz)

# example: for 800 MHz = 800000 kHz --> scaling\_min\_freq = 800000

# see conversion info: https://www.rapidtables.com/convert/frequency/mhz-to-hz.html

# to use this feature, uncomment the following line and set the value accordingly

# scaling\_min\_freq = 800000

# maximum cpu frequency (in kHz)

# see conversion info: https://www.rapidtables.com/convert/frequency/mhz-to-hz.html

# example: for 1GHz = 1000 MHz = 1000000 kHz -> scaling\_max\_freq = 1000000

# to use this feature, uncomment the following line and set the value accordingly

# scaling\_max\_freq = 1000000

# turbo boost setting (always, auto, or never)

turbo = auto

# battery charging threshold

# reference: https://github.com/AdnanHodzic/auto-cpufreq/#battery-charging-thresholds

#enable\_thresholds = true

#start\_threshold = 20

#stop\_threshold = 80

## How to run auto-cpufreq

auto-cpufreq should be run with with one of the following options:

* [monitor](https://github.com/AdnanHodzic/auto-cpufreq" \l "monitor)
  + Monitor and see suggestions for CPU optimizations
* [live](https://github.com/AdnanHodzic/auto-cpufreq" \l "live)
  + Monitor and automatically make (temporary) CPU optimizations
* [install](https://github.com/AdnanHodzic/auto-cpufreq" \l "install---auto-cpufreq-daemon) / [remove](https://github.com/AdnanHodzic/auto-cpufreq" \l "remove---auto-cpufreq-daemon)
  + Install/remove daemon for (permanent) automatic CPU optimizations
* [install (GUI)](https://github.com/AdnanHodzic/auto-cpufreq" \l "install---auto-cpufreq-daemon)
  + Install daemon via GUI for (permanent) automatic CPU optimizations
* [update](https://github.com/AdnanHodzic/auto-cpufreq" \l "update---auto-cpufreq-update)
  + Update auto-cpufreq to the latest release
* [install\_performance](https://github.com/AdnanHodzic/auto-cpufreq" \l "1-power_helperpy-script)
  + Install daemon in "performance" mode
* [stats](https://github.com/AdnanHodzic/auto-cpufreq" \l "stats)
  + View live stats of CPU optimizations made by daemon
* [force=TEXT](https://github.com/AdnanHodzic/auto-cpufreq" \l "overriding-governor)
  + Force use of either the "powersave" or "performance" governor, or set to "reset" to go back to normal mode
* config=TEXT
  + Use config file at designated path
* debug
  + Show debug info (include when submitting bugs)
* version
  + Show currently installed version
* [donate](https://github.com/AdnanHodzic/auto-cpufreq" \l "financial-donation)
  + To support the project
* help
  + Shows all of the above options
* completions=TEXT
  + To support shell completions (current options are "bash", "zsh", or "fish")

Running auto-cpufreq --help will print the same list of options as above. Read [auto-cpufreq modes and options](https://github.com/AdnanHodzic/auto-cpufreq" \l "auto-cpufreq-modes-and-options) for more details.

## auto-cpufreq modes and options

### Monitor

sudo auto-cpufreq --monitor

No changes are made to the system. This is solely to demonstrate what auto-cpufreq could do for your system.

### Live

sudo auto-cpufreq --live

Necessary changes are temporarily made to the system over time, but this process and its changes are lost at system reboot. This mode is provided to evaluate how the system would behave with auto-cpufreq permanently running on the system.

### Overriding governor

sudo auto-cpufreq --force=governor

Force use of either the "powersave" or "performance" governor, or set to "reset" to go back to normal mode. Please note that any set override will persist even after reboot.

### Install - auto-cpufreq daemon

Necessary changes are made to the system over time and this process will continue across reboots. The daemon is deployed and started as a systemd service. Changes are made automatically and live stats are generated for monitoring purposes.

**Install the daemon using CLI ([after installing auto-cpufreq](https://github.com/AdnanHodzic/auto-cpufreq" \l "installing-auto-cpufreq)):**

Installing the auto-cpufreq daemon using CLI is as simple as running the following command:

sudo auto-cpufreq --install

After the daemon is installed, auto-cpufreq is available as a binary and runs in the background. Its stats can be viewed by running: auto-cpufreq --stats

Please note: if the daemon is installed within a desktop environment, then its stats and options can be accessed via CLI or GUI. See "Install the daemon using GUI" below for more details.

**Install the daemon using GUI**

Starting with >= v2.0 [after installing auto-cpufreq](https://github.com/AdnanHodzic/auto-cpufreq" \l "installing-auto-cpufreq), an auto-cpufreq desktop entry (icon) is available, i.e.:

After selecting it to open the GUI, the auto-cpufreq daemon can be installed by clicking the "Install" button:

After that, the full auto-cpufreq GUI is available:

Please note: after the daemon is installed (by any method), its stats and options are accessible via both CLI and GUI.

**auto-cpufreq daemon service**

Installing the auto-cpufreq daemon also enables the associated service (equivalent to systemctl enable auto-cpufreq), causing it to start on boot, and immediately starts it (equivalent to systemctl start auto-cpufreq).

Since the daemon is running as a systemd service, its status can be seen by running:

systemctl status auto-cpufreq

If installed via Snap package, daemon status can be viewed as follows:

systemctl status snap.auto-cpufreq.service.service

### Update - auto-cpufreq update

Update functionality works by cloning the auto-cpufreq repo, installing it via [auto-cpufreq-installer](https://github.com/AdnanHodzic/auto-cpufreq" \l "auto-cpufreq-installer), and performing a fresh [auto-cpufreq daemon install](https://github.com/AdnanHodzic/auto-cpufreq" \l "install---auto-cpufreq-daemon) to provide the [latest version's](https://github.com/AdnanHodzic/auto-cpufreq/releases) changes.

Update auto-cpufreq by running: sudo auto-cpufreq --update. By default, the latest revision is cloned to /opt/auto-cpufreq/source, thus maintaining existing directory structure.

Update and clone to a custom directory by running: sudo auto-cpufreq --update=/path/to/directory

### Remove - auto-cpufreq daemon

The auto-cpufreq daemon, its systemd service, and all its persistent changes can be removed by running:

sudo auto-cpufreq --remove

This does, in part, the equivalent of systemctl stop auto-cpufreq && systemctl disable auto-cpufreq, but the above command should be used instead of using systemctl.

Please note: after the daemon is removed, the auto-cpufreq GUI and desktop entry (icon) are also removed.

### Stats

If the daemon has been installed, live stats of CPU/system load monitoring and optimization can be seen by running:

auto-cpufreq --stats

## Battery charging thresholds

As of [v2.2.0](https://github.com/AdnanHodzic/auto-cpufreq/releases/tag/v2.2.0), battery charging thresholds can be set in the config file. This enforces your battery to start and stop charging at defined values

### Supported devices

* **Lenovo ThinkPad** (thinkpad\_acpi)\*
* **Lenovo IdeaPad** (ideapad\_acpi)\*

\***Please note, your laptop must have an installed ACPI kernel driver specific to the manufacturer.** To check if you have the correct module installed and loaded run lsmod [module]

**To request that your device be supported, please open an**[**issue**](https://github.com/AdnanHodzic/auto-cpufreq/issues/new)**. In your issue, make us aware of the driver that works with your laptop**

### Battery config

Edit the config at /etc/auto-cpufreq.conf

Example config for battery ([already part of example config file](https://github.com/AdnanHodzic/auto-cpufreq/" \l "example-config-file-contents))

[battery]

enable\_thresholds = true

start\_threshold = 20

stop\_threshold = 80

### Lenovo\_laptop conservation mode

this works only with lenovo\_laptop kernel module compatable laptops.

add ideapad\_laptop\_conservation\_mode = true to your auto-cpufreq.conf file

### Ignoring power supplies

you may have a controler or headphones and when ever they may be on battery they might cause auto-cpufreq to limit preformence to ignore them add to you config file the name of the power supply, under [power\_supply\_ignore\_list]

the name of the power supply can be found with ls /sys/class/power\_supply/

[power\_supply\_ignore\_list]

name1 = this

name2 = is

name3 = an

name4 = example

# like this

xboxctrl = {the xbox controler power supply name}

## Troubleshooting

**Q:** If after installing auto-cpufreq you're (still) experiencing:

* high CPU temperatures
* CPU not scaling to minimum/maximum frequencies
* suboptimal CPU performance
* turbo mode not available

**A:** If you're using the intel\_pstate/amd-pstate CPU management driver, consider changing it to acpi-cpufreq.

This can be done by editing the GRUB\_CMDLINE\_LINUX\_DEFAULT params in /etc/default/grub. For instance:

sudo nano /etc/default/grub

# make sure you have nano installed, or you can use your favorite text editor

For Intel users:

GRUB\_CMDLINE\_LINUX\_DEFAULT="quiet splash intel\_pstate=disable"

For AMD users:

GRUB\_CMDLINE\_LINUX\_DEFAULT="quiet splash initcall\_blacklist=amd\_pstate\_init amd\_pstate.enable=0"

Once you have made the necessary changes to the GRUB configuration file, you can update GRUB by running sudo update-grub on Debian/Ubuntu, sudo grub-mkconfig -o /boot/grub/grub.cfg on Arch Linux, or one of the following on Fedora:

sudo grub2-mkconfig -o /etc/grub2.cfg

sudo grub2-mkconfig -o /etc/grub2-efi.cfg

sudo grub2-mkconfig -o /boot/grub2/grub.cfg

# legacy boot method

For systemd-boot users:

sudo nano /etc/kernel/cmdline

# make sure you have nano installed, or you can use your favorite text editor

For Intel users:

quiet splash intel\_pstate=disable

For AMD users:

quiet splash initcall\_blacklist=amd\_pstate\_init amd\_pstate.enable=0

Once you have made the necessary changes to the cmdline file, you can update it by running sudo reinstall-kernels.

### AUR

* If the AUR installer does not work for your system, fallback to auto-cpufreq-installer and open an issue.