

Monitoring

Tutorials

Telegraf + InfluxDB + Grafana

- [TheMickeyMike/raspberrypi-temperature-telegraf: Collect RaspberryPi CPU and GPU temperature with telegraf \(github.com\)](#)
- [Raspberry Pi Monitoring | Grafana Labs](#)
- [System monitoring | Nelis Oostens](#)
- [Simple Pi Stats | Grafana Labs](#)
- [Telegraf / Raspberry Metrics InfluxDB 2.0 \(Flux\) | Grafana Labs](#)
- [Telegraf: system PI dashboard | Grafana Labs](#)
- [?? vcgencmd ???? Raspberry Pi ? CPU ?????????????? - G. T. Wang](#)

Telegraf + InfluxDB

- [Monitoring Your Raspberry Pi System using InfluxDB Telegraf | Random Nerd Tutorials](#)

Dashboard ID 17191

- Link: [Telegraf / Raspberry Metrics InfluxDB 2.0 \(Flux\) | Grafana Labs](#)
- Dependencies: InfluxDB v2, Flux Language, Telegraf

telegraf.conf:

```
[[outputs.influxdb_v2]]
  ## The URLs of the InfluxDB cluster nodes.
  ##
  ## Multiple URLs can be specified for a single cluster, only ONE of the
  ## urls will be written to each interval.
  ## ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
  urls = ["http://your.influxdb.server:8086"]
  ## Token for authentication.
```

```
token = "THIS-IS-TOKEN"
```

```
## Organization is the name of the organization you wish to write to.
```

```
organization = "YOUR-ORG"
```

```
## Destination bucket to write into.
```

```
bucket = "YOUR-BUCKET"
```

```
[[inputs.net]]
```

```
[[inputs.netstat]]
```

```
[[inputs.file]]
```

```
files = ["/sys/class/thermal/thermal_zone0/temp"]
```

```
name_override = "cpu_temperature"
```

```
data_format = "value"
```

```
data_type = "integer"
```

```
[[inputs.exec]]
```

```
#Für ein 32bit System (welches überwacht werden soll)
```

```
commands = ["/usr/bin/vcgencmd measure_temp"]
```

```
#oder für ein 64bit System (welches überwacht werden soll), dann die Raute(#) entfernen und bei 32bit die  
Raute hinzufügen
```

```
# commands = ["/usr/bin/vcgencmd measure_temp"]
```

```
name_override = "gpu_temperature"
```

```
data_format = "grok"
```

```
grok_patterns = ["%{NUMBER:value:float}"]
```

```
# Read metrics about cpu usage
```

```
[[inputs.cpu]]
```

```
## Whether to report per-cpu stats or not
```

```
percpu = true
```

```
## Whether to report total system cpu stats or not
```

```
totalcpu = true
```

```
## If true, collect raw CPU time metrics
```

```
collect_cpu_time = false
```

```
## If true, compute and report the sum of all non-idle CPU states
```

```
report_active = false
```

```
## If true and the info is available then add core_id and physical_id tags
```

```
core_tags = false
```

```
# Read metrics about disk usage by mount point
```

```
[[inputs.disk]]
```

```
## By default stats will be gathered for all mount points.
```

```
## Set mount_points will restrict the stats to only the specified mount points.
```

```

# mount_points = ["/"]
## Ignore mount points by filesystem type.
ignore_fs = ["tmpfs", "devtmpfs", "devfs", "iso9660", "overlay", "aufs", "squashfs"]
## Ignore mount points by mount options.
## The 'mount' command reports options of all mounts in parathesis.
## Bind mounts can be ignored with the special 'bind' option.
# ignore_mount_opts = []
# Read metrics about disk IO by device
[[inputs.diskio]]
## By default, telegraf will gather stats for all devices including
## disk partitions.
## Setting devices will restrict the stats to the specified devices.
# devices = ["sda", "sdb", "vd*"]
## Uncomment the following line if you need disk serial numbers.
# skip_serial_number = false
#
## On systems which support it, device metadata can be added in the form of
## tags.
## Currently only Linux is supported via udev properties. You can view
## available properties for a device by running:
## 'udevadm info -q property -n /dev/sda'
## Note: Most, but not all, udev properties can be accessed this way. Properties
## that are currently inaccessible include DEVTYPE, DEVNAME, and DEVPATH.
# device_tags = ["ID_FS_TYPE", "ID_FS_USAGE"]
#
## Using the same metadata source as device_tags, you can also customize the
## name of the device via templates.
## The 'name_templates' parameter is a list of templates to try and apply to
## the device. The template may contain variables in the form of '$PROPERTY' or
## '${PROPERTY}'. The first template which does not contain any variables not
## present for the device is used as the device name tag.
## The typical use case is for LVM volumes, to get the VG/LV name instead of
## the near-meaningless DM-0 name.
# name_templates = ["$ID_FS_LABEL", "$DM_VG_NAME/$DM_LV_NAME"]
# Get kernel statistics from /proc/stat
[[inputs.kernel]]
# no configuration
# Read metrics about memory usage
[[inputs.mem]]
# no configuration

```

```
# Get the number of processes and group them by status
```

```
[[inputs.processes]]
```

```
  # no configuration
```

```
# Read metrics about swap memory usage
```

```
[[inputs.swap]]
```

```
# Read metrics about system load & uptime
```

```
[[inputs.system]]
```

Set the sudo

```
sudo usermod -aG video telegraf
```

Revision #16

Created 2 February 2024 11:30:02 by Admin

Updated 9 March 2024 10:43:31 by Admin