Installer l'agent Telegraf via Docker

- Se connecter en tant qu'admin : `ssh admin@db-<region>-sinp`
- Se placer dans le dossier : `cd ~/docker/telegraf`
    - Lancer le container afin d'afficher les logs : '' docker compose up ''
    - Corriger les éventuels problèmes de configuration de Telegraf s'affichant dans les logs
    - Lorsque tout fonctionne correctement, lancer le container en tant que service (option `-d`) : `docker compose up -d`

Monitorer différents services spécifiques

- Nginx:
    - status : Doc config du service status dans Nginx
    - access log : ajouter l'utilisateur "telegraf" au groupe "adm" dans le fichier `docker-compose.yml` via `group_add`. Puis configurer Telegraf en ajoutant le support de l'input Tail et du parser Grok (voir telegraf.conf ci-dessous comme exemple).
- Postgresql : ajouter l'utilisateur "telegraf" aux roles Postgresql et au fichier `pg_hba.conf` (Voir config Postgresql).
- Gunicorn : Doc monitorer Gunicorn GeoNature
- Disques additionnels : ajouter les entrées complémentaires à l'input `diskio`.

Monitorer InfluxDB v2

- Pour monitorer InfluxDB v2, il est nécessaire d'utiliser l'url influxdb.<domaine-sinp>/metrics qui retourne des infos au format *prometheus*.
- Il est conseille de stocker ces métriques dans un bucket distinct nommé `oss_metrics`.
- Pour créer le bucket, il est possible via l'interface influxdb.<domaine-sinp> d'ajouter un template qui va le créer pour nous ainsi qu'un dashboard :
    - Aller sur le menu Settings > Templates
    - Ajouter l'url suivante : https://raw.githubusercontent.com/influxdata/community-templates/master/influxdb2_oss_metrics/influxdb2_oss_metrics.yml
    - Vérifier la création du bucket "oss_metrics" et du Dashboard
- Configurer ensuite Telegraf en ajoutant une seconde sortie vers le bucket "oss_metrics", le plugin `inputs.prometheus` et utiliser la mécanique de tag du fichier de conf de Telegraf pour rediriger les métriques prometheus vers la sortie correspondant au bucket "oss_metrics".
- Redémarrer la stack telegraf : '' docker compose down ; docker compose up -d''
- Vérifier sur manager.<domaine-sinp> la sortie des logs de Telegraf.
- Vérifier dans influxdb.<domaine-sinp> que le plugin prometheus alimente bien le bucket "oss_metrics".
- Les métriques renvoyées par le plugin prometheus doivent être en version 1 pour êter compatible avec les Dashboard open source proposé pour Grafana et l'interface web d'InfluxDB.

Exemple de fichier telegraf.conf complet

Exemple avec :

- le support d'InfluxDB v2 accessible sur le port 8086 de l'ip privée 10.0.1.10
- le statut de Nginx sur http://172.18.5.1:9090/nginx_status (voir config Nginx),
- l'analyse des logs de Nginx via Tail et Grok,
- la surveillance de Postgresql
- la surveillance des service du système à l'aide du script *srvstatus*
- la surveillance de Gunicorn pour GeoNature à l'aide de Statsd

```
# Telegraf Configuration
[global_tags]

[agent]
  interval = "10s"
  round_interval = true
  metric_batch_size = 1000
  metric_buffer_limit = 10000
  collection_jitter = "0s"
  flush_interval = "10s"
  flush_jitter = "0s"
  precision = ""
  hostname = ""
  omit_hostname = false
  # WARNING : set to true to debug this config file !
  debug = false

[[outputs.influxdb_v2]]
  urls = ["http://10.0.1.10:8086"]
  ## Token for authentication.
  token = "${INFLUXDB2_TELEGRAF_TOKEN}"
  ## Organization is the name of the organization you wish to write to.
  organization = "${INFLUXDB2_INIT_ORG}"
  ## Destination bucket to write into.
  bucket = "${INFLUXDB2_INIT_BUCKET}"
  # Get all metrics except the one with "influxdb_database" tag with value
equal to "oss_metrics" :
  [outputs.influxdb_v2.tagdrop]
    influxdb_database = ["*"]

[[outputs.influxdb_v2]]
  urls = ["http://10.0.1.30:8086"]
  # Token for authentication.
  token = "${INFLUXDB2_TELEGRAF_TOKEN}"
  # Organization is the name of the organization you wish to write to.
  organization = "${INFLUXDB2_INIT_ORG}"
  # Destination bucket to write into.
  bucket = "oss_metrics"
  # Get only metrics with tag "influxdb_database" with a value equal to
```

```
"oss_metrics" :
  tagexclude = ["influxdb_database"]
  [outputs.influxdb_v2.tagpass]
    influxdb_database = ["oss_metrics"]

[[inputs.conntrack]]
  files = ["ip_conntrack_count", "ip_conntrack_max", "nf_conntrack_count",
"nf_conntrack_max"]
  dirs =
["/host/proc/sys/net/ipv4/netfilter","/host/proc/sys/net/netfilter"]

[[inputs.cpu]]
  percpu = true
  totalcpu = true
  collect_cpu_time = false
  report_active = false

[[inputs.disk]]
  # WARNING : for root path ("/") set the directory inside Docker container
(here it's "/host"). DO NOT add a trailing "/".
  mount_points = ["/host", "/host/data"]
  ignore_fs = ["tmpfs", "devtmpfs", "devfs", "iso9660", "overlay", "aufs",
"squashfs"]

# Monitoring of instance devices
# WARNING : see distinct devices in /dev with `df -h`.
[[inputs.diskio]]
  devices = ["sda", "sdb"]

[[inputs.docker]]
  endpoint = "unix:///var/run/docker.sock"
  gather_services = false
  container_name_include = []
  container_name_exclude = []
  timeout = "5s"
  perdevice = false
  perdevice_include = ["cpu", "blkio", "network"]
  total_include = ["cpu", "blkio", "network"]
  docker_label_include = []
  docker_label_exclude = []

# Monitoring of Systemd services with help of Srvstatus scripts
(inputs.systemd_status not working inside a Docker container)
[[inputs.exec]]
  commands = ["cat /opt/srvstatus/status.json"]
  timeout = "5s"
  name_override = "services_stats"
  data_format = "json"
  tag_keys = ["service"]

[[inputs.internal]]
```

```
[[inputs.interrupts]]
  cpu_as_tag = true
  [inputs.interrupts.tagdrop]
    irq = ["NET_RX", "TASKLET"]

[[inputs.kernel]]

[[inputs.linux_sysctl_fs]]

[[inputs.mem]]

[[inputs.net]]

[[inputs.netstat]]

# Monitoring of Nginx current status
[[inputs.nginx]]
  urls = ["http://172.18.5.1:9090/nginx_status"]
  response_timeout = "5s"

[[inputs.nstat]]
  proc_net_netstat = "/host/proc/net/netstat"
  proc_net_snmp = "/host/proc/net/snmp"
  proc_net_snmp6 = "/host/proc/net/snmp6"
  dump_zeros = true

[[inputs.postgresql]]
address = "host=172.18.5.1 user=telegraf password=<password> dbname=postgres
sslmode=disable"
outputaddress="postgresql-floresentinelle"
max_lifetime = "0s"
databases = ["geonature2db", "gnatlas"]

[[inputs.processes]]

# For InfluxDb metrics
[[inputs.prometheus]]
  urls = ["http://10.0.1.30:8086/metrics"]
  metric_version = 1
  [inputs.prometheus.tags]
    influxdb_database = "oss_metrics"

# Monitoring of Gunicorn for GeoNature, UsersHub, TaxHub and Atlas.
[[inputs.statsd]]
  protocol = "udp"
  max_tcp_connections = 250
  tcp_keep_alive = false
  service_address = ":8125"
  delete_gauges = true
```

```
  delete_counters = true
  delete_sets = true
  delete_timings = true
  percentiles = [50.0, 90.0, 99.0, 99.9, 99.95, 100.0]
  metric_separator = "_"
  parse_data_dog_tags = false
  datadog_extensions = false
  allowed_pending_messages = 10000
  percentile_limit = 1000

[[inputs.swap]]

[[inputs.system]]

# WARNING: disable because we can't use systemd inside a Docker container...
# Use inputs.exec instead with Srvstatus scripts on Host.
# See: https://github.com/influxdata/telegraf/issues/7689
#[[inputs.systemd_units]]

# Nginx access log monitoring
# WARNING : to grant permission to read log files, add "telegraf" user to
"adm" group via docker-compose.yml (group_add)
[[inputs.tail]]
  files = ["/host/var/log/nginx/access.log"]
  data_format = "grok"
  grok_timezone = "Europe/Paris"
  from_beginning = true
  name_override = "nginx_access_log"
  grok_patterns = ["%{CUSTOM_LOG_FORMAT}"]
  # Grock :
https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html
  # Grock pattern :
https://github.com/elastic/logstash/blob/v1.4.2/patterns/grok-patterns
  # Ex :
https://github.com/influxdata/telegraf/blob/master/plugins/parsers/grok/infl
ux_patterns.go
  grok_custom_patterns = '''
      CUSTOM_LOG_FORMAT %{IPORHOST:client_ip} (?:%{NOTSPACE:auth}|-)
(?:%{NOTSPACE:ident}|-) \[%{HTTPDATE:ts}\] "(?:%{WORD:verb}
%{NOTSPACE:request}(?: HTTP/%{NUMBER:http_version:float})?|%{DATA})"
%{NUMBER:resp_code} (?:%{NUMBER:resp_bytes:int}|-) %{QS:referrer}
%{QS:agent} rt="?%{NUMBER:request_time:float}"?
uct="(?:%{NUMBER:upstream_connect_time:float}|-)"
uht="(?:%{NUMBER:upstream_header_time:float}|-)"
urt="(?:%{NUMBER:upstream_response_time:float}|-)"
gzr="(?:%{NUMBER:gzip_ratio:float}|-)"
    '''

# WARNING : only for debugging
#[[outputs.file]]
#  namepass = ["nginx_access_log"]
```

```
#  files = ["/tmp/nginx_access_log.out"]
#  influx_sort_fields = true
```

Tester une métrique

- Pour tester une métrique, il est possible de rajouter une sortie au fichier config comme dans l'exemple ci-dessus présent à la fin du fichier.
- Il est aussi possible de :
  - se connecter au container : `docker exec -it telegraf /bin/bash`
  - d'exectuer un test avec la commande (Ex. ici avec la métrique *disk*) : `telegraf --input-filter=disk --test --debug`

From:
https://sinp-wiki.cbn-alpin.fr/ - **CBNA SINP**

Permanent link:
https://sinp-wiki.cbn-alpin.fr/serveurs/installation/docker-telegraf?rev=1690988227

Last update: **2023/08/02 14:57**