

# Installer, configurer et gérer le sous-domaine "monitor"

- **Notes** : ce domaine hébergera l'outil *Grafana* (avec *Telegraf*, *InfluxDb*) permettant de visualiser l'état des serveurs et de déclencher des alertes si nécessaire.

## Installer le domaine

- Créer un fichier de configuration : `vi /etc/nginx/sites-available/monitor.conf`
  - Y placer le contenu suivant :

```
server {
    listen 80;
    listen [::]:80;

    server_name monitor.<domaine-sinp>;

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $realip_remote_addr;
        proxy_set_header X-Forwarded-Host $host:$server_port;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_pass http://127.0.0.1:3000/;# ATTENTION : bien
mettre un slash final ! Sinon => erreur 404
    }
}
```

- Créer un lien depuis les sites actifs : `cd /etc/nginx/sites-enabled/ ; ln -s ../sites-available/monitor.conf monitor.conf`
  - Tester la config et relancer *Nginx* si OK : `nginx -t && nginx -s reload`
  - Tester l'URL `http://monitor.<domaine-sinp>/` qui doit afficher une erreur 502 car nous n'avons pas encore lancé le container Docker.
- En local, sur votre machine, se placer dans le dépôt Github "*sinp-paca-srv*" récupéré précédemment et si nécessaire resynchroniser le dossier *web-srv/docker* avec le serveur de destination en exécutant la commande *Rsync* indiquée dans le fichier [README.md](#).
- Sur le serveur dans le dossier *docker* de l'utilisateur *admin* :
  - vérifier la présence du réseau Docker spécifique à notre utilisation de type *bridge* nommé *nginx-proxy* (voir fichier *.env*) : `docker network ls`
  - se placer dans le dossier *monitor.silene.eu* : `cd ~/docker/monitor`
  - exécuter la commande : `docker-compose up`
  - vérifier que tout fonctionne à l'adresse : `http://monitor.<domaine-sinp>`
  - arrêter le container : `CTRL+C`
  - relancer le container en tant que service : `docker-compose up -d`

- si besoin de l'arrêter utiliser : `docker compose down`

## Activer le SSL et HTTP2 sur le domaine

- Installer un certificat SSL via Certbot (Letsencrypt) : `certbot --nginx -d monitor.<domaine-sinp>`
  - Répondre : 2
  - Tester ensuite la redirection auto de HTTP vers HTTPS : `http://monitor.<domaine-sinp>/` → doit redirigé vers HTTPS automatiquement
- Tester la configuration SSL : <https://www.ssllabs.com/ssltest/analyze.html?d=monitor.<domaine-sinp>>
- Tester l'URL `https://monitor.<domaine-sinp>/`
- La config finale :

```
server {
    listen 443 ssl http2; # managed by Certbot
    listen [::]:443 ssl http2; # managed by Certbot

    server_name monitor.<domaine-sinp>;

    location / {
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $realip_remote_addr;
        proxy_set_header X-Forwarded-Host $host:$server_port;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;

        proxy_pass http://127.0.0.1:3000/;# ATTENTION : bien mettre un
slash final ! Sinon => erreur 404
    }

    ssl_certificate /etc/letsencrypt/live/monitor.<domaine-
sinp>/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/monitor.<domaine-
sinp>/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by
Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    listen 80;
    listen [::]:80;

    server_name monitor.<domaine-sinp>;

    if ($host = monitor.<domaine-sinp>) {
```

```
    return 301 https://$host$request_uri;
} # managed by Certbot

return 404; # managed by Certbot
}
```

## Configurer InfluxDB v2

- Si ce n'est pas déjà fait, configurer [la zone docker pour le pare-feu FirewallD](#)
- Dans Influxdb v2, la politique de rétention est remplacée par les "buckets". La politique de rétention de "telegraf" est donc défini lors du setup d'Influxdb via des variables d'environnement :
  - INFLUXDB2\_INIT\_BUCKET=telegraf
  - INFLUXDB2\_INIT\_RETENTION=365d : 1 an.

### InfluxDB v2 : connection au shell et requête de config

- Se connecter au container d'InfluxDB :

```
docker exec -it monitor-influxdb /bin/bash
```

- Générer un fichier de config :

```
influx config create --active --org <organisation> --config-name
administrator --host-url http://monitor-influxdb:8086 --token <admin-
token-from-InfluxDB2-interface>
```

- Pour créer l'utilisateur telegraf (impossible depuis l'interface), lancer la commande :

```
influx user create -n telegraf -p <telegraf-password> -o <organisation>
```

- Pour définir la période de rétention du bucket telegraf :
  - lister les buckets pour récupérer l'id du bucket "telegraf" :

```
influx bucket list
```

- définir la période de rétention à 1 an :

```
influx bucket update -i <telegraf-bucket-id> -r 365d
```

- Créer un mapping entre le bucket "telegraf" (v2) et la database/retention policy "telegraf" (v1) pour y accéder avec InfluxQL depuis Grafana :

```
influx v1 dbrp create --db telegraf --rp telegraf --bucket-id <telegraf-
bucket-id> --default
```

## Monitorer InfluxDB v2

- Pour monitorer InfluxDB, il est nécessaire de modifier le fichier de configuration du Telegraf présent sur le serveur hébergeant la base InfluxDB :

```
[[outputs.influxdb_v2]]
  urls = ["http://monitor-influxdb: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}"
```

- Il est aussi recommandé de désactiver le monitoring dans la base *\_internal* d'InfluxDB en modifiant comme suit le paramètre du fichier de configuration d'InfluxDB :

```
[monitor]
  store-enabled = false
```

## Notes sur Telegraf

- Pour tester son fonctionnement : `docker exec -it monitor-telegraf telegraf --test`
- Tester le plugin *Nginx* : `docker exec -it monitor-telegraf telegraf --input-filter nginx --test`

## Configurer Grafana

- Se connecter à *Grafana* sur `https://monitor.<domaine-sinp>`
- Ajouter les sources de données :
  - Cliquer sur le menu gauche "Configuration" > "Data sources"
  - Cliquer sur le bouton "Add data source"
    - Sélectionner l'entrée "InfluxDB" dans la section "Time series databases"
    - Compléter le formulaire en laissant tout par défaut sauf :
      - *Name* : InfluxDB (Flux) - Telegraf
      - *HTTP* :
        - *URL* : `http://monitor-influxdb:8086`
      - *Auth* :
        - *With Credentials* : à activer pour Influxdb v2.
      - *Custom HTTP Headers* :
        - Cliquer sur *Add header* pour InfluxDb v2 et ajouter un entête *Authorisation* avec pour valeur *Token <grafana-api-token>*.
        - Pour obtenir la valeur de *<grafana-api-token>*, connecter vous à l'interface d'InfluxDB v2 et générer un nouveau token (Data > Api



## Notes sur la sauvegarde/restauration de la configuration de Grafana

- **Notes** : afin de sauvegarder et restaurer la configuration de *Grafana* (dashboards, datasources, alert channels, folders), un script Python spécifique est disponible sous forme de Container Docker présent dans la stack "monitor.silene.eu". Le service a pour nom "*monitor-grafana-backup*" et se base sur l'image [ysde/docker-grafana-backup-tool](#).
- Par défaut, ce container n'est pas lancé. Il doit donc être lancé manuellement.
- Pour sauvegarder la configuration de *Grafana*, utiliser la commande : `docker-compose run monitor-grafana-backup`
  - Les sauvegardes sont présentes sous forme de fichier `.tar.gz` dans le dossier `/home/admin/docker/monitor.<domaine-sinp>/grafana/backup`
- Pour restaurer une sauvegarde, utiliser la commande : `docker-compose run monitor-grafana-backup restore _OUTPUT_/<date>.tar.gz`
- Pour automatiser les sauvegardes, ajouter le fichier `cron grafana` dans le dossier `/etc/cron.d`
- Pour réaliser une sauvegarde distante de la configuration de *Grafana*, installer le script `bkp2dbx` et s'assurer que le fichier `/etc/cron.d/bkp2dbx` contient bien une ligne réalisant la sauvegarde du dossier `backup` de *Grafana*.
- Pour fonctionner, il est nécessaire de générer un jeton d'API dans Grafana via le menu : "Configuration" > "API Keys"
  - Nom du jeton : "grafana-backup-tool"
  - Rôle : "Admin"
  - Time to live : "5y"
- Copier ensuite, le jeton obtenu dans le fichier `/home/admin/docker/monitor.<domaine-sinp>/.env` en tant que valeur du paramètre `GRAFANA_TOKEN`

## Configuration des alertes de Grafana

- Les requêtes à utiliser pour les alertes ne peuvent pas utiliser de variables utilisateurs, il est donc nécessaire de mettre des valeurs en dur...
- Pour tester les requêtes, se connecter au shell d'InfluxDB comme indiqué ci-dessous.
- Exemple de requête dans Grafana qui peut être testé au préalable dans le shell d'InfluxDB:

```
SELECT mean(usage_user) AS "user", mean(usage_system) AS "system",
mean(usage_softirq) AS "softirq", mean(usage_steal) AS "steal",
mean(usage_nice) AS "nice", mean(usage_irq) AS "irq",
mean(usage_iowait) AS "iowait", mean(usage_guest) AS "guest",
mean(usage_guest_nice) AS "guest_nice" FROM "cpu" WHERE "host" =~
/^(sinp-paca-(db|web))/ AND "cpu" = 'cpu-total' AND TIME > now() - 5m
GROUP BY TIME($__interval), *
```

- Pour tester la requête précédente dans *InfluxDB*, remplacer `$__interval` par `10s`
- A priori, il est conseillé de ne pas mettre la valeur `NoData` pour le champ "**If no data or all values are null**" qnd on utilise une règle d'alerte utilisant **For**. Vous pouvez utiliser la valeur `Keep Last State`.
- Sur l'interface de *Grafana*, vous pouvez tester l'alerte en cliquant sur le bouton "Test rule" en bas de l'onglet "Alert"

- Si tout se passe correctement, vous devez obtenir un objet pour chaque entrée de la clause SELECT qui contient par exemple :

```
{"message": "Condition[0]: Eval: false, Metric: Alert, Value: 4.287"}
```

- La valeur contient Eval: false qui indique que les conditions ne sont pas réunies pour ce paramètre pour lancer l'alerte
  - Ici Value: 4.287" correspond à la moyenne de la propriété usage\_user
- Autres exemples de requêtes d'alertes :
    - **cpu** :

```
SELECT mean(usage_user) AS "user", mean(usage_system) AS "system",
mean(usage_softirq) AS "softirq", mean(usage_steal) AS "steal",
mean(usage_nice) AS "nice", mean(usage_irq) AS "irq",
mean(usage_iowait) AS "iowait", mean(usage_guest) AS "guest",
mean(usage_guest_nice) AS "guest_nice" FROM "cpu" WHERE "host" =~
/^sinp-paca-(db|web)$/ AND "cpu" = 'cpu-total' AND TIME > now() -
5m GROUP BY TIME($__interval), *
```

- **Load** :

```
SELECT mean(load1) AS short, mean(load5) AS medium, mean(load15)
AS long FROM "system" WHERE host =~ /^sinp-paca-(web|db)$/ AND
TIME > now() - 5m GROUP BY TIME($__interval), * ORDER BY ASC
```

- **Memory** pour *sinp-paca-web* :

```
SELECT mean(used) AS used FROM "mem" WHERE "host" = 'sinp-paca-
web' AND TIME > now() - 5m GROUP BY TIME($__interval), host ORDER
BY ASC
```

- **Memory** pour *sinp-paca-db* :

```
SELECT mean(used) AS used FROM "mem" WHERE "host" = 'sinp-paca-db'
AND TIME > now() - 5m GROUP BY TIME($__interval), host ORDER BY
ASC
```

- **Disk usage** pour *sinp-paca-web* :

```
SELECT mean(used) AS "used" FROM "disk" WHERE "host" = 'sinp-
paca-web' AND path = '/' AND TIME > now() - 5m GROUP BY
TIME($__interval), "host", "path"
```

- **Disk usage** pour *sinp-paca-db* :

```
SELECT mean(used) AS "used" FROM "disk" WHERE "host" = 'sinp-
paca-db' AND path = '/' AND TIME > now() - 5m GROUP BY
TIME($__interval), "host", "path"
```

## Infos sur les graphs et les métriques système

- [Linux Load averages](#)

## Archives

### ▢ Configurer InfluxDB v1

- Créer une nouvelle politique de rétention de données sur la base *Telegraf* :

```
docker exec -it monitor-influxdb influx -execute 'CREATE RETENTION POLICY "telegraf_1_year" ON "telegraf" DURATION 365d REPLICATION 1 SHARD DURATION 7d DEFAULT'
```

- Voir les politiques de rétention de données sur la base *Telegraf* :

```
docker exec -it monitor-influxdb influx -execute 'SHOW RETENTION POLICIES ON "telegraf"'
```

### ▢ InfluxDB v1 : connexion au shell d'InfluxDB et requêtes de base

- Pour se connecter au shell *InfluxDB* du container *monitor-influxdb* : `docker exec -it monitor-influxdb influx`
- Voir les bases de données : `SHOW DATABASES`
- Sélectionner une base de données : `USE "telegraf"`
- Voir les métriques enregistrées : `SHOW MEASUREMENTS`
- Voir les séries d'une métrique (affiche aussi les métadonnées) : `SHOW SERIES FROM <measurement>`
- Voir les clés des métadonnées (= tag) associées à une métrique : `SHOW TAG KEYS FROM <measurement>`
- Voir le contenu de la métadonnée (= tag) *host* pour toutes les métriques : `SHOW TAG VALUES WITH KEY = "host"`
  - Pour une métrique en particulier : `SHOW TAG VALUES FROM <measurement> WITH KEY = "host"`
- Voir les champs d'une métrique : `SHOW FIELD KEYS FROM <measurement>`
- Afficher les données d'une métrique : `SELECT * FROM <measurement>`
- Forcer l'affichage du temps des données sous forme de date et heure formaté : `precision rfc3339`
- Pour supprimer un ancien nom d'hôte apparaissant dans Grafana, il faut supprimer les données en fonction du temps et de la métrique (*measurement* ⇒ *postgresql*) : `DELETE FROM postgresql WHERE time < '2020-01-09 17:00:00'`
- Possible aussi de supprimer globalement pour toutes les métriques : `DELETE WHERE time < '2020-01-09 10:37:00'`

## ☐ Monitorer InfluxDB v1

- InfluxDb v1 :
  - Pour monitorer InfluxDB, il est nécessaire de modifier le fichier de configuration du Telegraf présent sur le serveur hébergeant la base InfluxDB :

```
[[outputs.influxdb]]
  urls = ["http://monitor-influxdb:8086"]
  database = "telegraf"

[[inputs.influxdb]]
  urls = [
    "http://monitor-influxdb:8086/debug/vars"
  ]
```

From:

<http://sinp-wiki.cbn-alpin.fr/> - **CBNA SINP**

Permanent link:

<http://sinp-wiki.cbn-alpin.fr/serveurs/installation/web-srv/docker-grafana?rev=1685267408>

Last update: **2023/05/28 09:50**

