# Agent Telegraf Azure

# Objectifs

- Découvrir Telegraf
- Tester l’agent Telegraf pour envoyer les métriques dans :
  - Prometheus
  - Azure Monitor
- Comparer l’agent natif d’Azure et Telegraf

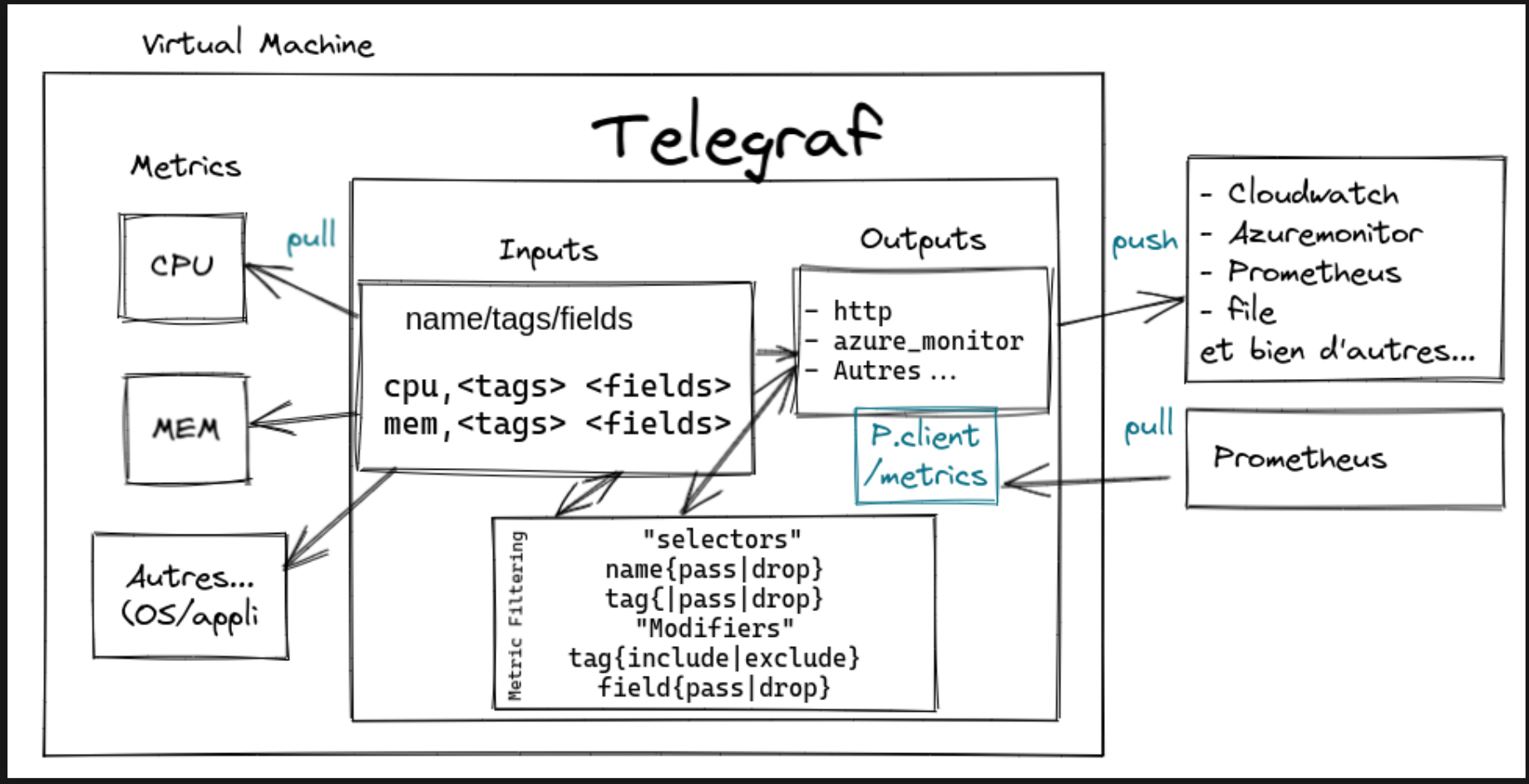
# Prérequis

Nous allons faire dans cette présentation l'impasse sur :

- l'installation/configuration de l'agent Télégraf
- le fonctionnement de Telegraf

qui ont déjà étaient abordés dans l'excellente présentation vidéo, disponible ICI

# Concepts de base Telegraf



# C’est quoi une métrique dans Télégraf ?

Une métrique est composé d’un nom (name), de balise (tags), de champs (fields).

```
> cpu,client=pf_pf-866,cluster=pf_azure,cpu=cpu0,host=pf-791-linux,os=pf
```

Les valeurs sont présentées en key=value avec l’horodatage en dernier.

# Authentification pour envoyer dans Azure Monitor

# Métriques dans Azure Monitor (1/4)

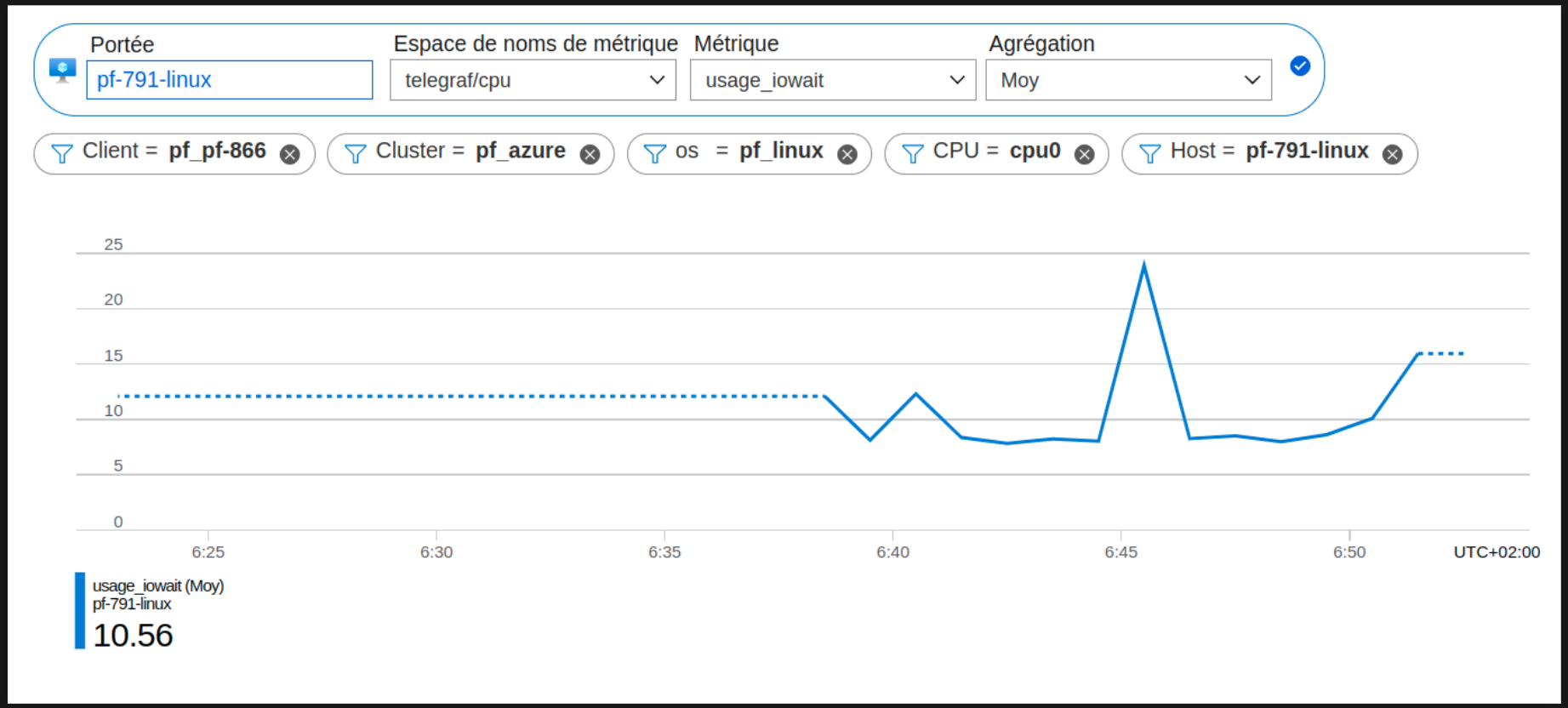
- Le plugin de type output `azure_monitor` envoie les métriques vers Azure Monitor
- Le plugin est encore en preview
- Il aggrége les métriques récupérées en tranches d’une minute
- Ensuite les tranches sont envoyées à Azure Monitor à chaque intervalle de flush

# Métriques dans Azure Monitor (2/4)

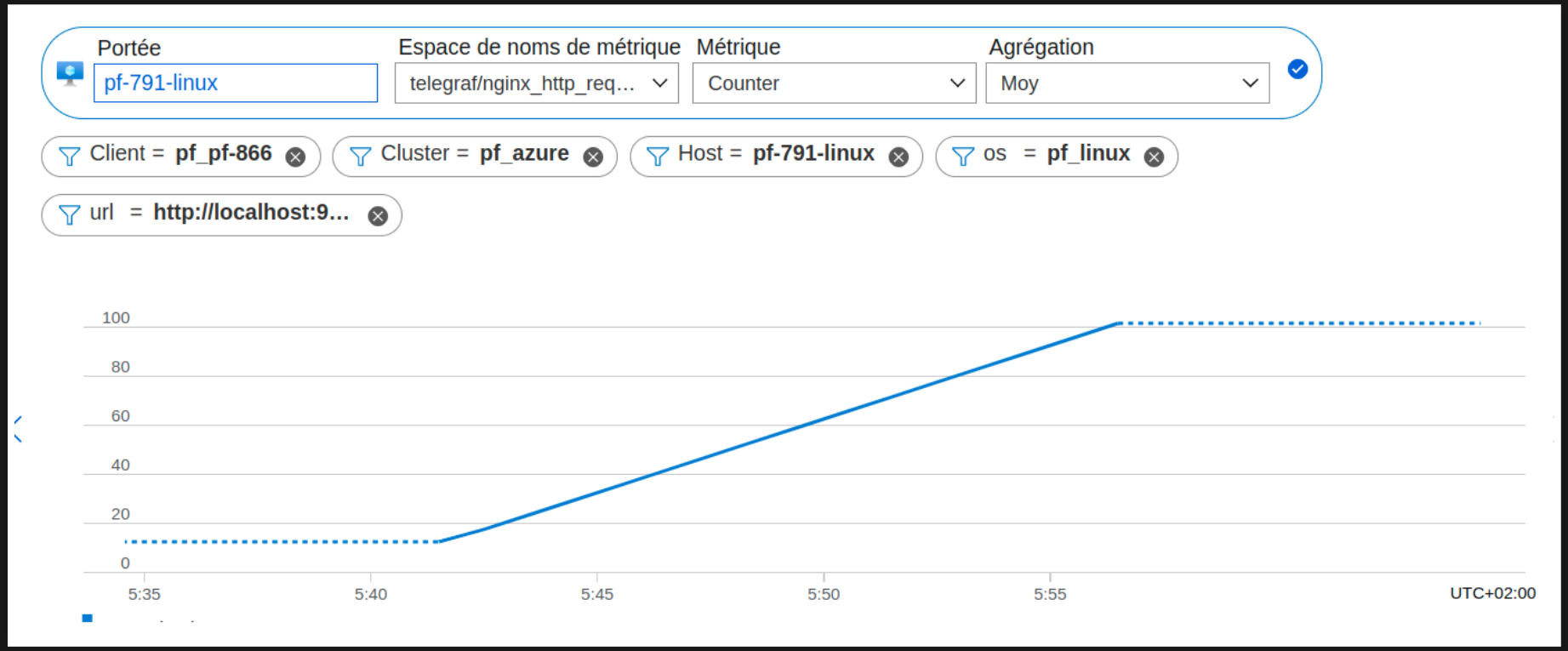
Exemple de configuration

```
[[outputs.azure_monitor]]
  ## Set the namespace prefix, defaults to "Telegraf/<input-name>".
  # namespace_prefix = "Telegraf/"

  ## Azure Monitor doesn't have a string value type, so convert string
  ## fields to dimensions (a.k.a. tags) if enabled
  # strings_as_dimensions = false

  ## Azure Region to publish metrics against.
  ##   ex: region = "southcentralus"
  # region = ""

  # resource_id = ""
  # endpoint_url = "https://monitoring.core.usgovcloudapi.net"
```

# Métriques dans Azure Monitor (3/4)



# Métriques dans Azure Monitor (4/4)

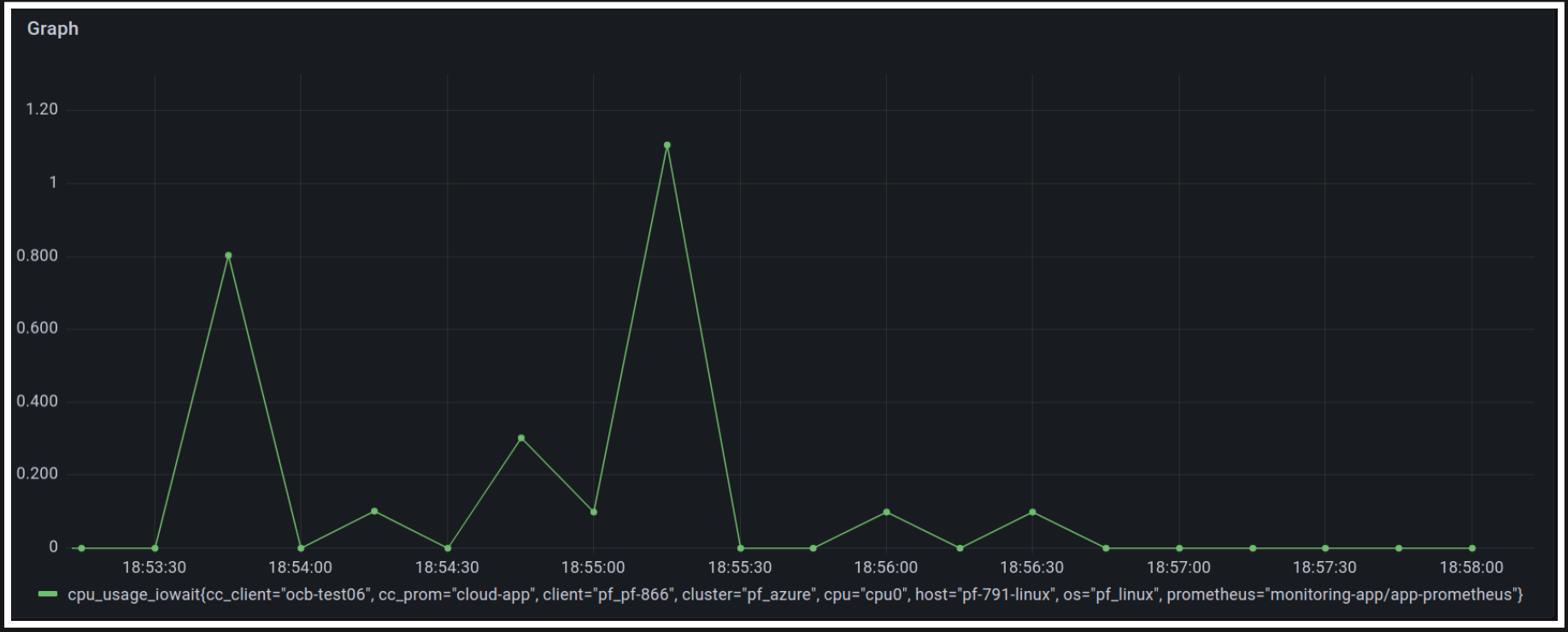


# Métriques dans Prometheus (push) (1/3)

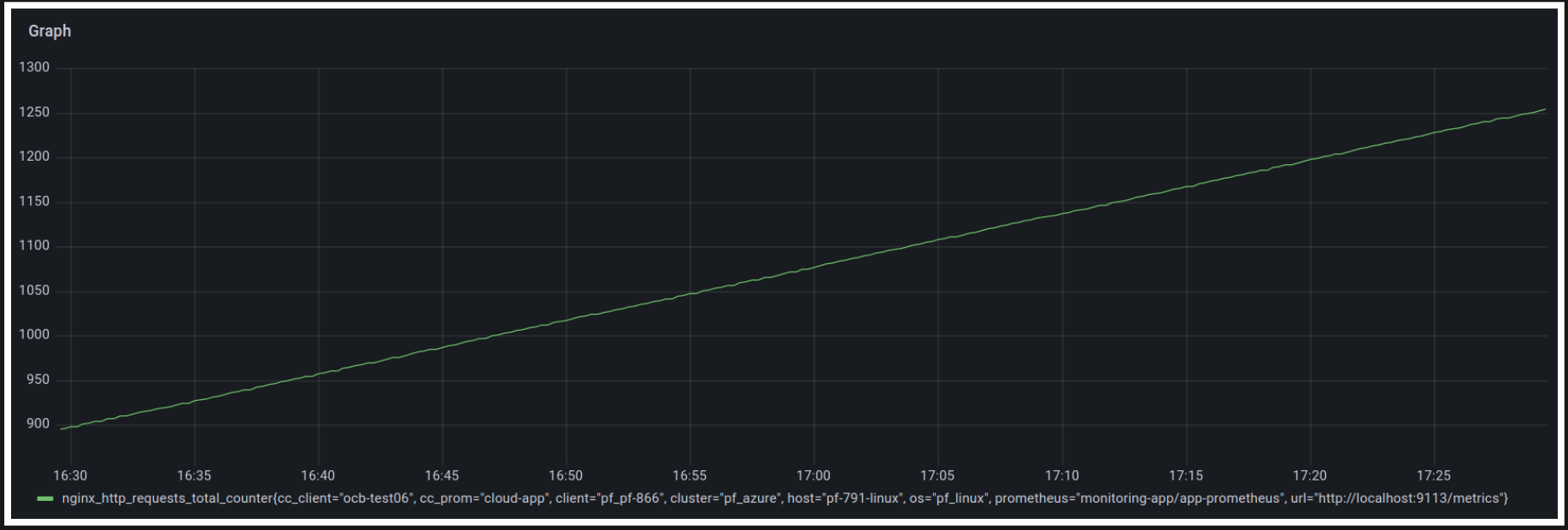
Le plugin de type output `http` envoie les données dans un message http, dans un format `prometheusremotewrite`. Le header `Authorization` associé permet l'authentification dans Rancher.

```
[[outputs.http]]
  alias = "prometheus-0"
  url = "https://rancher.ocb-test06[...]/prometheus-0[...]/v1/write"
  data_format = "prometheusremotewrite"

  [outputs.http.headers]
        Authorization = "Bearer xxx"
        Content-Type = "application/x-protobuf"
        Content-Encoding = "snappy"
        X-Prometheus-Remote-Write-Version = "0.1.0"
```

# Métriques dans Prometheus (push) (2/3)



# Métriques dans Prometheus (push) (3/3)



# Métriques dans Prometheus (pull)

Le plugin de type output `prometheus_client` expose les métriques sur un endpoint http (par défaut : `/metrics`). Un serveur Prometheus va ensuite récupérer les données par pulling.

Ce mode de fonctionnement est à privilégier dans le cas de métriques de type `Summary` et `Histogram`.

```
# Configuration for the Prometheus client to spawn
[[outputs.prometheus_client]]
  ## Address to listen on.
  listen = ":9273"
```

# Monitoring de Telegraf : output health (1/2)

Permet d’avoir un healtcheck http
Différents tests peuvent être réalisés :

- `compares` : ça compare la valeur d’un field à une valeur en dure
- `contains` : ça vérifie que la métrique contient bien un field

Si :

- échec : code retour 503
- succès : code retour 200

# Monitoring de Telegraf : output health (2/2)

## Exemple de configuration

```
[[outputs.health]]
  service_address = "http://localhost:8080"

  namepass = ["internal_agent", "internal_write"]

  [[outputs.health.compares]]
     field = "gather_errors"
     lt = 5.0

  [[outputs.health.contains]]
     field = "buffer_size"
```

# Monitoring de Telegraf : input internal (1/2)

**Permet d’avoir des métriques sur Telegraf lui-même :**

- le nombre de métriques collectées
- le nombre de métriques écrites
- la taille limite du buffer
- la taille du buffer
- des stats sur la mémoire
- ...

# Monitoring de Telegraf : input internal (2/2)

- Configuration :

```
# Collect statistics about itself
[[inputs.internal]]
  ## If true, collect telegraf memory stats.
  # collect_memstats = true
```

- Dashboard upstream disponible (à améliorer)

# Monitoring de Telegraf : mix des deux solutions

Les deux plugins utilisés séparément ne permettent pas de valider la chaine de bout en bout.
On pourrait utiliser les deux plugins pour monitorer Telegraf.

- internal pour récupérer les métriques internes de Telegraf
- health pour vérifier que le plugin internal fonctionne comme attendu

# Problème connu avec Azure Monitor (1/3)

## Description

- Azure Monitor n’accepte pas les valeurs négatives (contrairement à Prometheus ou Cloudwatch)
- Une seule métrique avec une valeur négative -> toutes les écritures sur Azure Monitor sont bloquées

# Problème connu avec Azure Monitor (2/3)

## Solutions

- Filtrer les métriques ayant une valeur négative `fielddrop = ["tcp_maxconn"]`
- Toutes les valeurs négatives -> 0

```
[[processors.execd]]
command = ["sed", "s/=-[0-9]*i/=0i/"]
```

- Faire un patch upstream pour que les valeurs négatives ne soient plus envoyées dans Azure Monitor

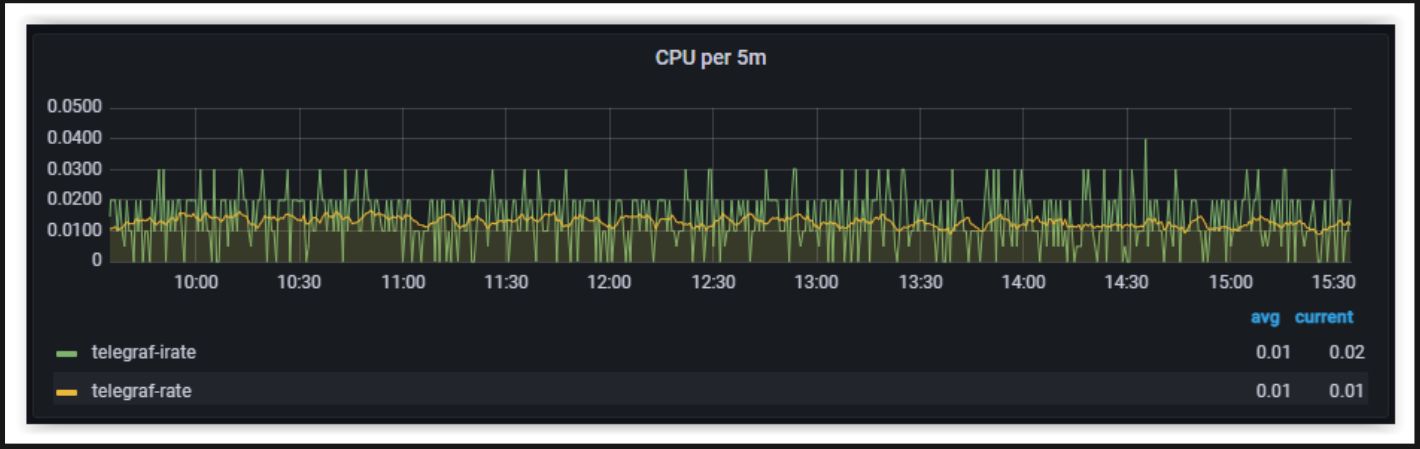
# Problème connu avec Azure Monitor (3/3)

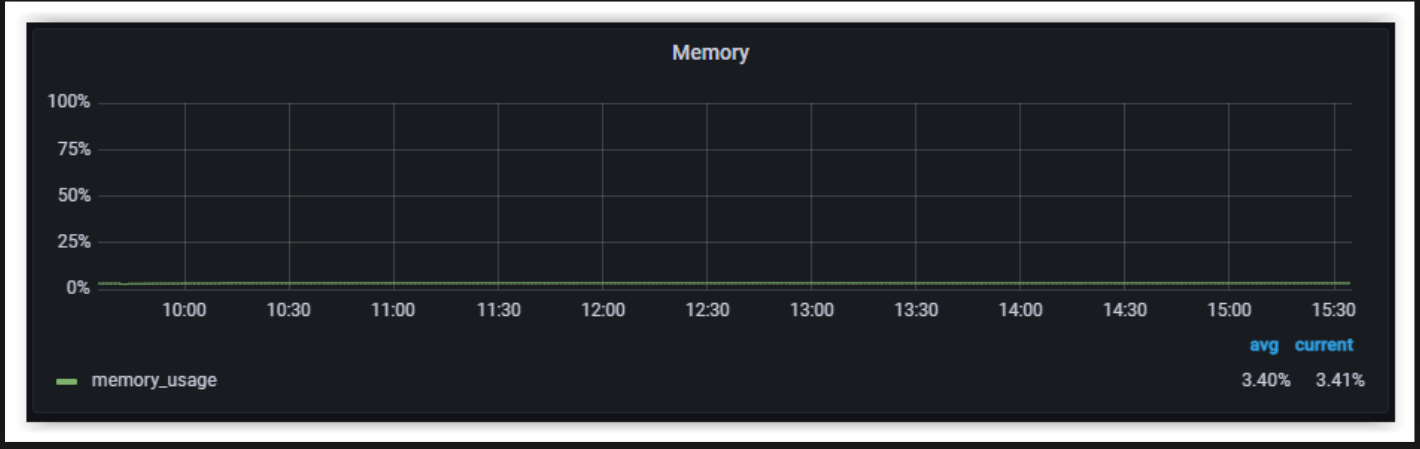
## Supervision de ce problème :

# Limitations

- Le problème des valeurs négatives
- Le plugin output Azure monitor est en preview. Il ne fonctionne pas avec toutes les régions.

# Ressources utilisées (CPU/RAM)



# Comparaison entre Telegraf et l'agent Azure Monitor

| | Azure agent monitor (AMA) | Telegraf |
|---|---|---|
| Linux | ⊕ | ⊕ |
| Windows | ⊕ | ⊕ |
| Empreinte CPU | faible : 2% 1 CPU (linux) | faible |
| Empreinte mémoire | faible : 204 MB (linux) | faible |
| Outside Azure | ⊕ | ⊕ |
| Facilité de configuration / installation | ⊕ | ⊕ |
| Etendue de supervision | ⊖ | ⊕ |
| Filtrage des métriques | ⊕ | ⊕ |
| Configuration flexible : tags custom, transformation des métriques, suppression de tags ... | ⊖ | ⊕ |
| Métriques OS | ⊕ | ⊕ |
| Métriques applicatives | ⊖ | ⊕ |
| Configuration échantillonnage métriques | ✓ | ⊕ |
| Documentation | ⊕ | ⊕ |