Annexe

Technique

CYBER Management School

Monitoring

GROUPE TERCIUM

Benoît ROBART

Table des matières

Sommaire

1. Introduction & Objectifs

OBJECTIFS :

1. **Monitorer l'instance Jitsi (CPU, RAM, disques, etc.).**
2. **Visualiser dans Grafana.**
3. **Superviser la disponibilité du service Jitsi via Prometheus (ou Blackbox Exporter).**
4. **Intégrer Pfsense et ipatable si possible.**
5. **Alerter si nécessaire.**

Ce module de supervision a été mis en œuvre pour suivre l'état et les performances d'une architecture incluant une instance Jitsi Meet, un nœud Linux hébergeant Prometheus et Grafana, ainsi que des cibles distantes comme des services HTTP externes (Google, GitHub). Il inclut :

- Un serveur **Prometheus** avec configuration de règles d'alerte
- Un serveur **Grafana** avec **dashboards JSON** prêts à importer
- Des **exporters** locaux : **Node Exporter, Blackbox Exporter, Telegraf.**

Ni pfsense, ni iptables ne sont supporter sur l'instanciation infomaniak. Des routines pourront être réaliser en bash pour obtenir les résultats escomptés : à voir donc car pour pfsense c'est écris en freeBSD.

2. Architecture du Monitoring

Schéma de supervision déployée :

- **Prometheus** : collecte des métriques (port 9091).
- **Node Exporter** : monitoring système local (port 9100).
- **Blackbox Exporter** : vérification ICMP, HTTP, TCP, SSH (port 9115).
- **Grafana** : visualisation des données collectées (port 3000).
- **Telegraf** : collecte des métriques et des logs (port 9273).

Contrôle des ports : Permissions

sudo ufw allow 9091/tcp	**Prometheus**
sudo ufw allow 9100/tcp	**Node Exporter**
sudo ufw allow 9115/tcp	**Blackbox Exporter**
sudo ufw allow 3000/tcp	**Grafana**
sudo ufw allow 9145/tcp	**Iptables_Exporter**

cd : sudo ufw allow 9091/tcp
sudo ufw allow 9100/tcp
sudo ufw allow 9115/tcp
sudo ufw allow 3000/tcp
sudo ufw allow 9145/tcp ECHEC

Infomaniak ; la gestion des groupes de sécurité pour les ports :

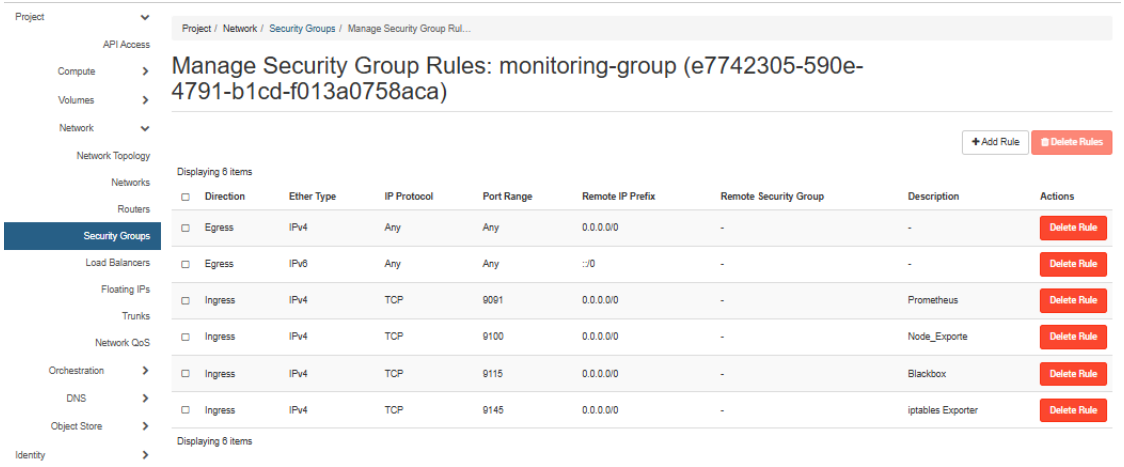
Voici le **tableau récapitulatif des ports essentiels** à ouvrir par service pour ta plateforme Jitsi + Monitoring (Prometheus, Grafana, Blackbox, pfSense Exporter).

✅ Tableau des ports par service

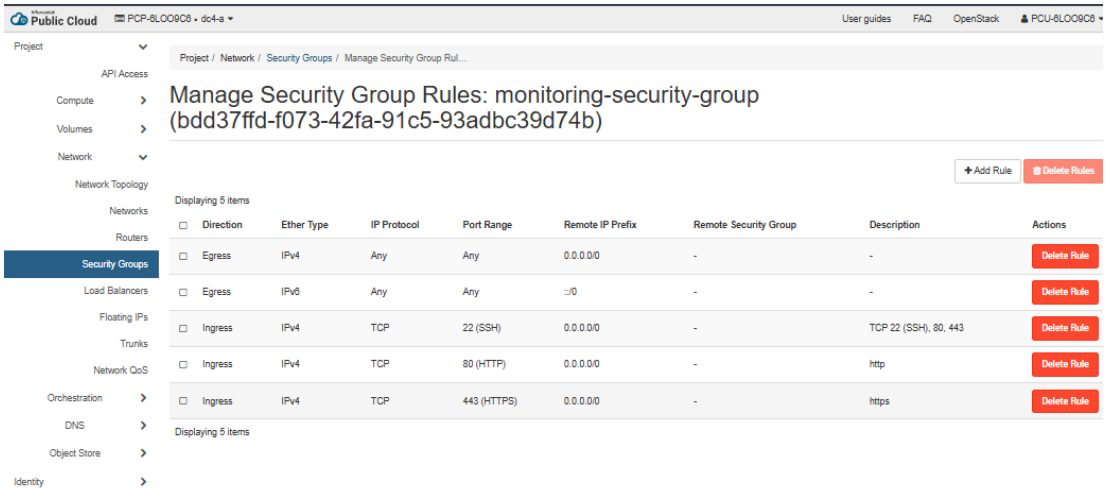
Service	Port	Protocole	Type	Description
SSH (admin)	22	TCP	Ingress	Connexion au serveur
HTTP	80	TCP	Ingress	Web non chiffré (utile pour redirection)
HTTPS	443	TCP	Ingress	Accès web sécurisé (ex. visio.workeezconnect)
Grafana	3000	TCP	Ingress	Interface Web Grafana
Jitsi Media	10000	UDP	Ingress	Canal de flux audio/vidéo UDP (WebRTC)
Prometheus UI	9090 (ou 9091)	TCP	Ingress	Interface Prometheus
Node Exporter	9100	TCP	Ingress	Export métriques serveur Linux
Blackbox Exporter	9115	TCP	Ingress	Tests ICMP/HTTP externes
iptables Exporter	9145	TCP	Ingress	Export règles netfilter/iptables

🗂️ Groupes créés :

Nom du groupe	Ports inclus	Utilisation
`default` (désactivable)	Aucun si retiré	Peut être exclu
`monitoring-group`	9091, 9100, 9115, 9145	Prometheus + Exporters
`monitoring-security-group`	22, 80, 443	Accès standard SSH + HTTP/HTTPS
`visio-monitoring-group`	3000, 10000	Grafana + Jitsi UDP (media)



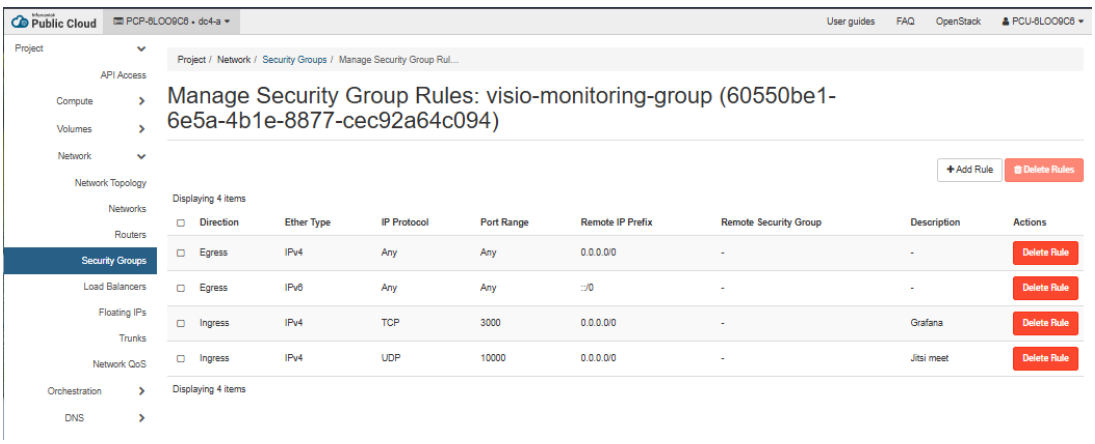
Ce Security Group intègre l'ensemble des ports dédiés aux outils de Monitoring.



Ce Security Group intègre l'ensemble des ports HTTP / HTTPS / SSH.

🛑 Attention

- Le port **10000/UDP** est critique pour que la visioconférence fonctionne.
- **IPv6** peut être ignoré sauf cas spécifique (pas utile pour WebRTC sauf config avancée).
- L'on peut désormais **supprimer "default"** une fois **tous les groupes nécessaires rattachés**.



```
ubuntu@jitsi-tercium:~$ sudo ufw allow 9091/tcp
sudo ufw allow 9100/tcp
sudo ufw allow 9115/tcp
sudo ufw allow 3000/tcp
sudo ufw allow 9145/tcp
Skipping adding existing rule
Skipping adding existing rule (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
Rules updated
Rules updated (v6)
ubuntu@jitsi-tercium:~$
```

***Contrôle des ports :Listen**
cd : sudo ss -tulnp | grep -E '9001|9100|9115|3000|9145'

```
ubuntu@jitsi-tercium:~$ # Vérifie l'écoute
sudo ss -tulnp | grep 9091

# Vérifie la réponse HTTP
curl http://localhost:9091

# Vérifie l'accessibilité depuis l'extérieur
curl http://[IP_PUBLIQUE]:9091
tcp   LISTEN 0      4096                                  *:9091            *:*    users:(("prometheus",pid=258337
,fd=7))
<a href="/graph">Found</a>.

curl: (3) bad range in URL position 9:
http://[IP_PUBLIQUE]:9091
        ^
ubuntu@jitsi-tercium:~$
```

```
ubuntu@jitsi-tercium:~$ sudo ss -tulnp | grep -E '9091|9100|9115|3000|9145'
tcp   LISTEN 0      4096                                  *:9091            *:*    users:(("prometheus",pid=258337
,fd=7))
tcp   LISTEN 0      4096                                  *:9100            *:*    users:(("node_exporter",pid=752
16,fd=3))
```

```
server.service.
ubuntu@jitsi-tercium:~$ sudo ss -tulnp | grep 3000
tcp   LISTEN 0      4096                                  *:3000            *:*    users:(("grafana",pid=265585,fd
=13))
```

Prometheus : Après modifications contenant port et les logs de blackbox_http et node_exporter l'on peut relancer et tester l'interface.

Redémarrer :
cd : sudo systemctl restart prometheus

Vérifier l'état :
cd : sudo systemctl status prometheus
curl http://localhost:9091/targets

Navigateur web :
cd : http://37.156.46.238:9091/targets

3. L'Installation à proprement dite :

3.1 Prometheus : Préparation de l'environnement

Créer les dossiers système :
cd : sudo mkdir -p /etc/prometheus
sudo mkdir -p /var/lib/prometheus

Créer un utilisateur sécurisé :
cd : sudo useradd -rs /bin/false prometheus

Téléchargement et extraction :
cd : sudo wget
https://github.com/prometheus/prometheus/releases/download/v2.52.0/prometheus-2.52.0.linux-amd64.tar.gz
tar -xvf prometheus-2.52.0.linux-amd64.tar.gz

cd : sudo mv prometheus-2.52.0.linux-amd64 /opt/prometheus

Attribution des droits et copie des fichiers :
cd : sudo cp /opt/prometheus/prometheus.yml /etc/prometheus/
sudo cp -r /opt/prometheus/consoles /opt/prometheus/console_libraries /etc/prometheus/
sudo chown -R prometheus: /etc/prometheus /var/lib/prometheus

Création du service systemd :
cd : sudo tee /etc/systemd/system/prometheus.service > /dev/null <<EOF
[Unit]
Description=Prometheus
Wants=network-online.target
After=network-online.target

[Service]
User=prometheus
Group=prometheus
Restart=on-failure
**ExecStart=/opt/prometheus/prometheus **
**--config.file=/etc/prometheus/prometheus.yml **
**--storage.tsdb.path=/var/lib/prometheus **
**--web.console.templates=/etc/prometheus/consoles **
**--web.console.libraries=/etc/prometheus/console_libraries **
--web.listen-address=0.0.0.0:9091 **le port 9090 est pris par Jitsi-Meet**

[Install]
WantedBy=multi-user.target
EOF

Configuration de base de Prometheus :
Chemin : /etc/prometheus/prometheus.yml
Ou
cd : sudo nano : /etc/prometheus/prometheus.yml

yaml :

```
global:
  scrape_interval: 15s
  evaluation_interval: 15s

scrape_configs:
  - job_name: 'prometheus'
    static_configs:
      - targets: ['localhost:9091']
```

Normalement c'est le port 9090 mais il est pris par Jitsi-Meet

Activation et test du service :

cd : sudo systemctl daemon-reload
sudo systemctl enable --now prometheus (redondance car –now active et démarre**).**

Vérification :

cd : sudo systemctl status prometheus
curl http://localhost:9091/metrics

Redémarrer Prometheus si nécessaire :

cd : sudo systemctl restart prometheus

sudo systemctl enable prometheus (pour rendre son lancement automatique**).**

Vérification au terminal : `curl http://localhost:9091/metrics`

À éviter :

1. Ne pas utiliser ****apt install prometheus -y** si vous avez procédé à l'installation manuelle :

   - Risque de conflit de ports
   - Chemins différents
   - Overlap avec /etc/default/prometheus, /usr/bin/prometheus, etc.

2. **Mention non claire des chemins persistants (`/opt`)** :
   Tu pourrais expliciter pourquoi tu copies les fichiers depuis /opt/prometheus/... vers /etc/prometheus — tous les utilisateurs ne comprennent pas que /opt n'est pas scanné automatiquement par Prometheus.

3. Pourquoi certains oublient /etc et /var ?
   a. Parce que beaucoup se basent sur un usage **containerisé (Docker)** ou bien sur des paquets pré-compilés (apt, snap) qui créent **automatiquement** ces chemins.
   b. Or, dans une **installation manuelle depuis binaire**, **rien n'est créé**. C'est **à l'administrateur de déclarer et peupler les chemins manuellement**. C'est pour ça que tu as eu raison de forcer leur création.

Problèmes avec l'installation via apt :

Risque	Détail
Version souvent obsolète	**apt fournit une version < à celle sur GitHub. Problème si tu relies à une instance Prometheus récente.**
Chemins non standardisés	**Binaire dans /usr/bin, service systemd préécrit, sans contrôle ni commentaire de ta part.**
Pas d'options de lancement personnalisées	**Impossible de passer des --web.listen-address, --collector.*, etc., sans réécriture manuelle du service.**
Moins bon contrôle des droits utilisateurs	**Pas d'utilisateur node_exporter isolé créé par défaut.**

3.2 Node Exporter

Objectif
Installer Node Exporter à partir des binaires officiels, dans un chemin contrôlé (/opt/node_exporter), en créant un service dédié avec permissions sécurisées.

Créer un utilisateur système sécurisé :
cd : sudo useradd -rs /bin/false nodeusr

Créer le répertoire cible pour l'installation :
cd : sudo mkdir -p /opt/node_exporter

Télécharger et installer Node Exporter :
cd : /tmp wget
https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz

```
sudo tar -xvf node_exporter-1.8.1.linux-amd64.tar.gz
sudo cp node_exporter-1.8.1.linux-amd64/node_exporter /opt/node_exporter/
sudo chown -R nodeusr: /opt/node_exporter
```

Créer le service systemd :

```
cd : sudo tee /etc/systemd/system/node_exporter.service > /dev/null <<EOF
[Unit]
Description=Node Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=nodeusr
Group=nodeusr
Type=simple
ExecStart=/opt/node_exporter/node_exporter

[Install]
WantedBy=multi-user.target
EOF
```

Activer et démarrer le service :
cd : sudo systemctl daemon-reload
sudo systemctl enable --now node_exporter

Intégration dans Prometheus :
Ajouter au fichier /etc/prometheus/prometheus.yml dans scrape_configs

```
- job_name: 'node_exporter'
  static_configs:
    - targets: ['localhost:9100']
      labels:
        instance: 'node_local'
```

Puis redémarrer Prometheus :
cd : sudo systemctl restart prometheus

On évite quand l'on installe des binaires cette commande : sudo apt install prometheus-node-exporter -y

Distinction claire :

- ✅ **sudo systemctl **enable --now** node_exporter**
  ➤ **Active le service au démarrage et le démarre immédiatement.**

- ✅ **sudo systemctl **enable** node_exporter**
  ➤ **Ne fait que l'activer au démarrage, sans le démarrer tout de suite.**

- ✅ **sudo systemctl **start** node_exporter**
  ➤ **Ne fait que le démarrer, sans l'activer pour le prochain redémarrage.**

On détaille pour la pédagogie :

```
sudo systemctl enable node_exporter    # Active au démarrage
sudo systemctl start node_exporter     # Démarre immédiatement
```

Erreurs fréquentes

Problème	Symptôme	Solution
Port 9100 inaccessible	Curl ne répond pas	Vérifier firewall, redémarrer service node_exporter
Mauvais binaire	Service plante au démarrage	Télécharger la bonne version pour l'architecture
Permission refusée	ExecStart échoue	Vérifier ownership (chown -R nodeusr: sur /opt)
Prometheus ne récupère pas de KPI	Dashboard vide	Vérifier cible Prometheus dans prometheus.yml

Amendement explicatif : Pourquoi pas de binary library ?

Excellente remarque. Voici une analyse précise, étape par étape :

Pourquoi ne crée-t-on pas systématiquement /etc/node_exporter et /opt/node_exporter comme pour Prometheus ou Blackbox Exporter ?

1. Node Exporter = binaire autonome sans configuration

Contrairement à :

- **Prometheus, qui utilise /etc/prometheus/prometheus.yml,**
- **Blackbox Exporter, qui lit un fichier /etc/blackbox_exporter/blackbox.yml,**

→ Node Exporter n'utilise *aucun fichier de configuration par défaut*. Il expose directement les métriques système via /metrics.

2. Pas besoin de /etc/node_exporter

Il ne lit pas de fichiers statiques :

- **Pas de node_exporter.yml,**
- **Pas de modules ou de profils à configurer.**

⚠ Cependant, si tu veux ajouter des paramètres personnalisés (ex. : inclure/exclure certains collectors), alors oui, créer /etc/node_exporter/ pour y déposer un futur node_exporter.flags ou autre est pertinent.

3. Pourquoi créer /opt/node_exporter ?

Cas légitime : si tu télécharges le binaire depuis GitHub :
wget
https://github.com/prometheus/node_exporter/releases/download/v1.8.1/node_exporter-1.8.1.linux-amd64.tar.gz

tar -xvf node_exporter-1.8.1.linux-amd64.tar.gz

sudo mv node_exporter-1.8.1.linux-amd64 /opt/node_exporter

▨ Cela permet de :

- **Avoir une arborescence propre,**
- **Centraliser tous les binaires tiers dans /opt.**

Mais dans la majorité des cas, on extrait directement node_exporter et le place dans /usr/local/bin/ sans conserver /opt/.

✅ **Conclusion :**

Dossier	Obligatoire ?	Utilité / Commentaire
/usr/local/bin	✅ Oui	Répertoire standard pour les binaires système non distribués
/opt/node_exporter	◆ Optionnel	À utiliser si on souhaite garder l'archive extraite proprement
/etc/node_exporter	◆ Optionnel	Utile si on gère des flags ou des fichiers complémentaires

3.3 Blackbox Exporter

Créer un utilisateur système et un répertoire:
cd : sudo useradd -rs /bin/false blackbox
sudo mkdir -p /etc/blackbox_exporter /usr/local/bin /opt

Télécharger :
cd : cd / opt

Source : sudo wget
https://github.com/prometheus/blackbox_exporter/releases/download/v0.24.0/blackbox_exporter-0.24.0.linux-amd64.tar.gz

tar -xvzf blackbox_exporter-0.24.0.linux-amd64.tar.gz
mv blackbox_exporter-0.24.0.linux-amd64 blackbox_exporter
Service systemd : /etc/systemd/system/blackbox_exporter.service

Installation / déplacements des fichiers / permissions :
cd : sudo mv /opt/blackbox_exporter/blackbox_exporter /usr/local/bin/
sudo chmod +x /usr/local/bin/blackbox_exporter
sudo mv /opt/blackbox_exporter/blackbox.yml /etc/blackbox_exporter/
sudo chown -R blackbox: /etc/blackbox_exporter

Adapter le service systemd (/etc/systemd/system/blackbox_exporter.service) :

```
[Unit]
Description=Blackbox Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=blackbox
Group=blackbox
ExecStart=/usr/local/bin/blackbox_exporter
          --config.file=/etc/blackbox_exporter/blackbox.yml
Restart=on-failure
```

NoNewPrivileges=true
ProtectSystem=full
ProtectHome=true
ReadWritePaths=/etc/blackbox_exporter

[Install]
WantedBy=multi-user.target

Recharger les services :
cd : sudo systemctl daemon-reexec / à éviter

cd : sudo systemctl daemon-reload active le service, démarre automatiquement au boot
sudo systemctl enable --now blackbox_exporter id

Si le système tourne déjà :

cd : sudo systemctl daemon-reload
sudo systemctl restart blackbox_exporter

Vérification :
cd : curl -s http://localhost:9115/metrics | head -n 10
ou
cd : curl -s http://localhost:9091/api/v1/targets | jq **'.data.activeTargets[] | select(.labels.job=="blackbox_exporter_http")'**

1. Déplacement du binaire et du fichier de configuration :

```
# Créer un répertoire propre
sudo mkdir -p /usr/local/blackbox_exporter

# Déplacer les fichiers
sudo mv /opt/blackbox_exporter/* /usr/local/blackbox_exporter/

# Assurer les droits
sudo chown -R root:root /usr/local/blackbox_exporter
sudo chmod -R 755 /usr/local/blackbox_exporter
```

2. Service systemd à modifier :
cd : sudo nano /etc/systemd/system/blackbox_exporter.service

```
[Unit]
Description=Prometheus Blackbox Exporter
Wants=network-online.target
After=network-online.target

[Service]
User=ubuntu
ExecStart=/usr/local/blackbox_exporter/blackbox_exporter \
  --config.file=/usr/local/blackbox_exporter/blackbox.yml \
  --web.listen-address="0.0.0.0:9115"
Restart=always

[Install]
WantedBy=multi-user.target
```

3. Recharger, activer et démarrer le service :
cd : sudo systemctl daemon-reexec à éviter
sudo systemctl daemon-reload
sudo systemctl enable blackbox_exporter
sudo systemctl restart blackbox_exporter

4. **Vérification :**

 # Vérifier que ça écoute bien
cd : ss -tulpen | grep 9115

```
ubuntu@jitsi-tercium:/opt$ ss -tulpen | grep 9115
tcp   LISTEN 0      4096                                *:9115             *:*    uid:1002 ino:619735 sk:1039 cgroup:/
system.slice/blackbox_exporter.service v6only:0 <->
```

 # Test d'export
cd : curl http://localhost:9115/metrics | head -n 20

3.4 Grafana

Objectif

Fournir une interface de visualisation des métriques collectées par Prometheus (et autres exporters).

Ajout du dépôt officiel Grafana :

cd : sudo apt-get install -y software-properties-common

sudo apt-get install -y apt-transport-https

sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://packages.grafana.com/gpg.key | sudo gpg --dearmor -o /etc/apt/keyrings/grafana.gpg

**echo "deb [signed-by=/etc/apt/keyrings/grafana.gpg] https://packages.grafana.com/oss/deb stable main" | **

cd : sudo tee /etc/apt/sources.list.d/grafana.list > /dev/null
sudo apt update

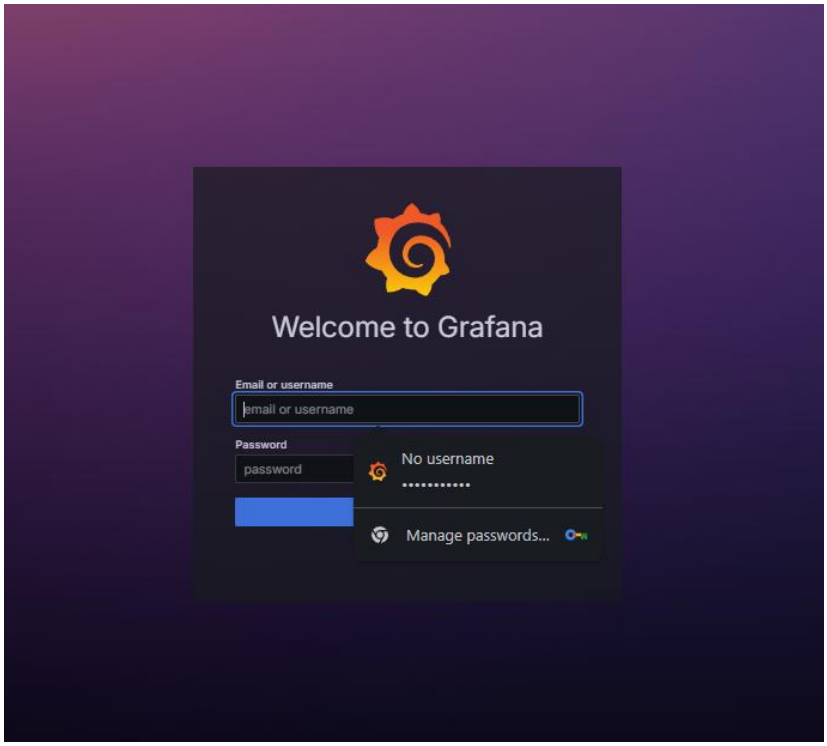
Installation de Grafana :
cd : sudo apt install grafana -y

Démarrage et activation du service :
cd : sudo systemctl enable --now grafana-server

Vérification :
cd : sudo systemctl status grafana-server

Accès via navigateur :
http://localhost:3000
Identifiants par défaut :

- **utilisateur : admin**
- **mot de passe : admin (à modifier au premier accès) fait on accéde alors aux paramètres**



Exploration des métrics :

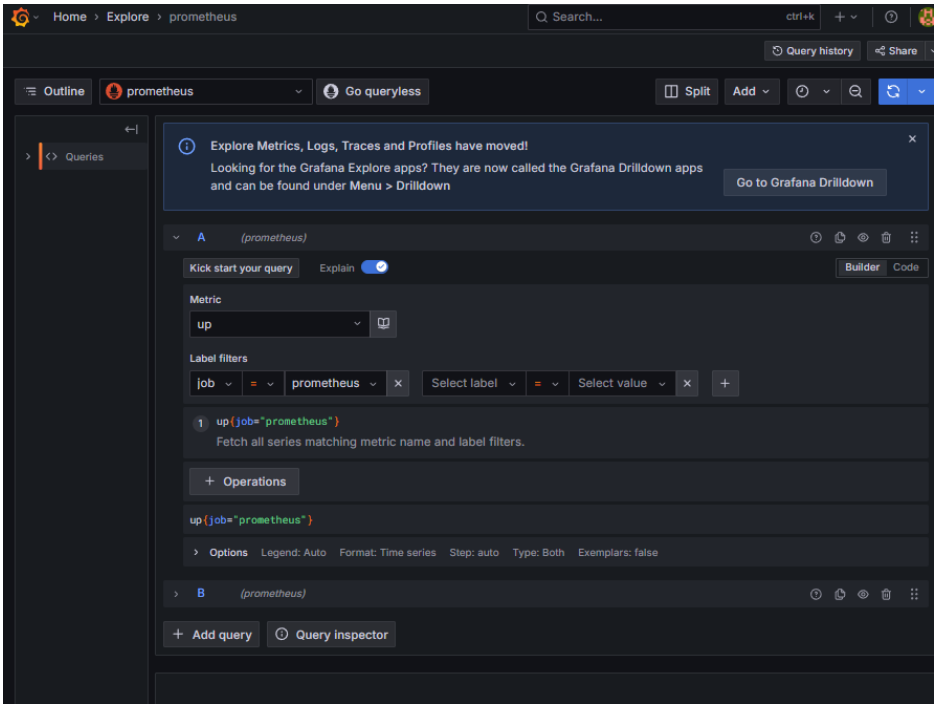


Diagramme et résultat : exemple

À retenir :

Élément	Détail
Port par défaut	**3000**
Service	**grafana-server**
Dashboards JSON	**Importables depuis /etc/grafana/provisioning/**
Sources de données	**Prometheus : http://localhost:9091 (par défaut)**

3.5 Telegraf

Objectif

Telegraf est un **agent léger de collecte de métriques et de logs**. Il fait partie de la suite TICK (Telegraf, InfluxDB, Chronograf, Kapacitor), mais peut fonctionner **indépendamment** et exporter vers Prometheus ou Grafana.

Il collecte, transforme et envoie des données.

- **Collecte** → via des *plugins d'entrée* (inputs)
- **Traitement / mise en forme** → *plugins de transformation*
- **Envoi** → via des *plugins de sortie* (outputs)

Ajout du dépôt officiel InfluxData

cd : curl -s https://repos.influxdata.com/influxdata-archive.key | sudo gpg --dearmor -o /usr/share/keyrings/influxdata-archive-keyring.gpg

cd : sudo apt update

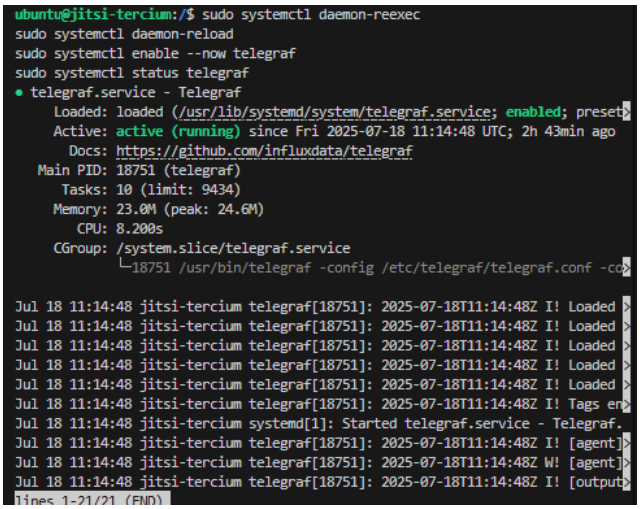
Installation de Telegraf
cd : sudo apt install telegraf -y

Préparation (optionnelle)
cd : sudo mkdir -p /etc/telegraf/

Démarrage et activation
cd : sudo systemctl enable --now telegraf

Vérification
cd : sudo systemctl status telegraf

Synthèse : Activation et lancement
cd : sudo systemctl daemon-reexec
sudo systemctl daemon-reload
sudo systemctl enable --now telegraf
sudo systemctl status telegraf



Exemple d'intégration vers Prometheus
Ajoutez dans /etc/telegraf/telegraf.conf :

[[outputs.prometheus_client]]

listen = ":9273"

Vérifiez avec :
cd : curl http://localhost:9273/metrics

Puis ajoutez dans prometheus.yml :

```
- job_name: 'telegraf'
  static_configs:
    - targets: ['localhost:9273']
```

echo "deb [signed-by=/usr/share/keyrings/influxdata-archive-keyring.gpg] https://repos.influxdata.com/ubuntu $(lsb_release -cs) stable" | \

sudo tee /etc/apt/sources.list.d/influxdata.list > /dev/null

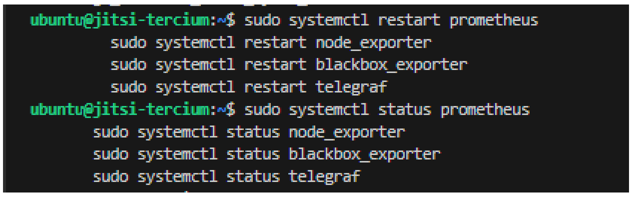
Vérification des démarrages des modules :

Relancer les services système via le terminal :

cd : **sudo systemctl restart prometheus**
sudo systemctl restart node_exporter
sudo systemctl restart blackbox_exporter
sudo systemctl restart telegraf

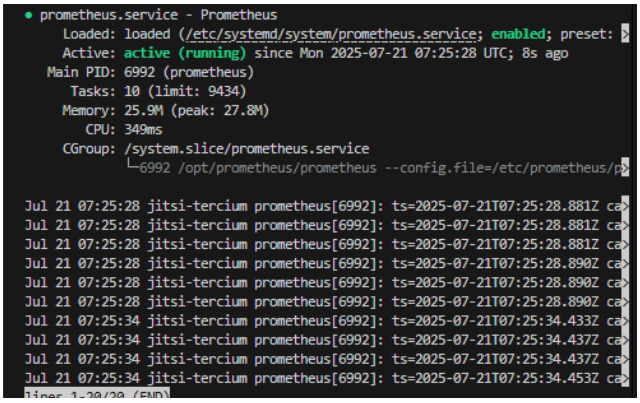
Pour vérifier qu'ils tournent :

cd : **sudo systemctl status prometheus**
sudo systemctl status node_exporter
sudo systemctl status blackbox_exporter
sudo systemctl status telegraf



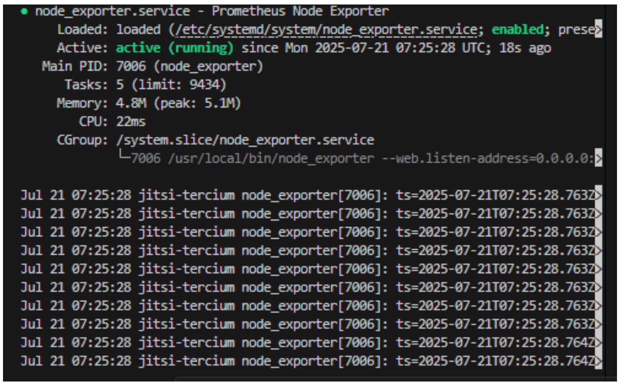
```
ubuntu@jitsi-tercium:~$ sudo systemctl restart prometheus
          sudo systemctl restart node_exporter
          sudo systemctl restart blackbox_exporter
          sudo systemctl restart telegraf
ubuntu@jitsi-tercium:~$ sudo systemctl status prometheus
        sudo systemctl status node_exporter
        sudo systemctl status blackbox_exporter
        sudo systemctl status telegraf
```

D'abord Prometheus:



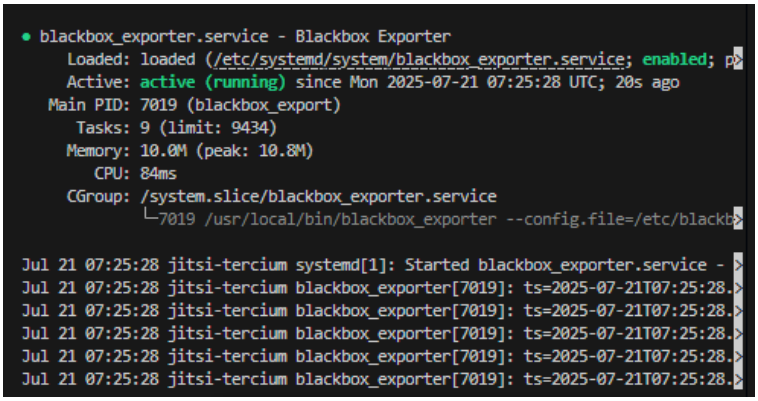
```
● prometheus.service - Prometheus
     Loaded: loaded (/etc/systemd/system/prometheus.service; enabled; preset: >
     Active: active (running) since Mon 2025-07-21 07:25:28 UTC; 8s ago
   Main PID: 6992 (prometheus)
      Tasks: 10 (limit: 9434)
     Memory: 25.9M (peak: 27.8M)
        CPU: 349ms
     CGroup: /system.slice/prometheus.service
             └─6992 /opt/prometheus/prometheus --config.file=/etc/prometheus/p>

Jul 21 07:25:28 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:28.881Z ca>
Jul 21 07:25:28 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:28.881Z ca>
Jul 21 07:25:28 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:28.881Z ca>
Jul 21 07:25:28 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:28.890Z ca>
Jul 21 07:25:28 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:28.890Z ca>
Jul 21 07:25:28 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:28.890Z ca>
Jul 21 07:25:34 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:34.433Z ca>
Jul 21 07:25:34 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:34.437Z ca>
Jul 21 07:25:34 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:34.437Z ca>
Jul 21 07:25:34 jitsi-tercium prometheus[6992]: ts=2025-07-21T07:25:34.453Z ca>
lines 1-20/20 (END)
```

Puis node_exporter :

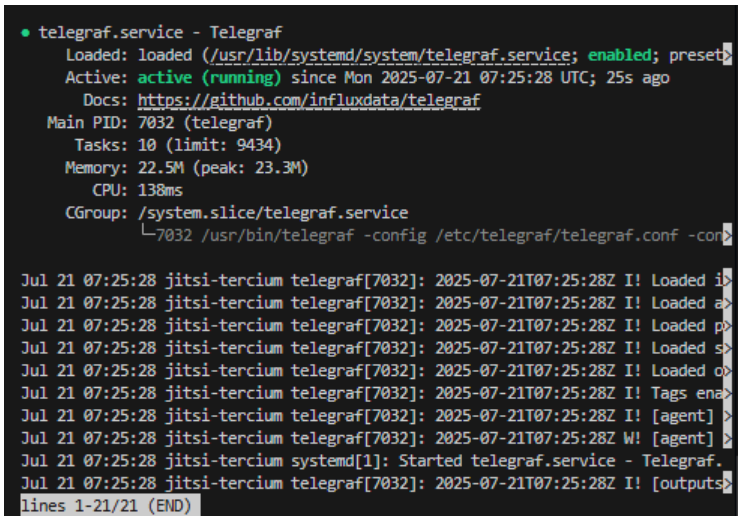


```
● node_exporter.service - Prometheus Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; prese>
     Active: active (running) since Mon 2025-07-21 07:25:28 UTC; 18s ago
   Main PID: 7006 (node_exporter)
      Tasks: 5 (limit: 9434)
     Memory: 4.8M (peak: 5.1M)
        CPU: 22ms
     CGroup: /system.slice/node_exporter.service
             └─7006 /usr/local/bin/node_exporter --web.listen-address=0.0.0.0:>

Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.763Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.764Z>
Jul 21 07:25:28 jitsi-tercium node_exporter[7006]: ts=2025-07-21T07:25:28.764Z>
```

Puis blackbox_exporter :



Puis Telegraf :



Accès aux interfaces Web :

Si tous les services sont actifs, l'on peut accéder aux interfaces suivantes :

Composant	URL locale	URL distante (publique)
Prometheus	**http://localhost:9090**	**http://37.156.46.238:9091** ***(si exposé)***
Node Exporter	**http://localhost:9100**	**http://37.156.46.238:9100** ***(si exposé)***
Blackbox Exporter	**http://localhost:9115**	**http://37.156.46.238:9115** ***(si exposé)***
Grafana	**http://localhost:3000**	**http://37.156.46.238:3000** ✅

4. Configuration consolidée

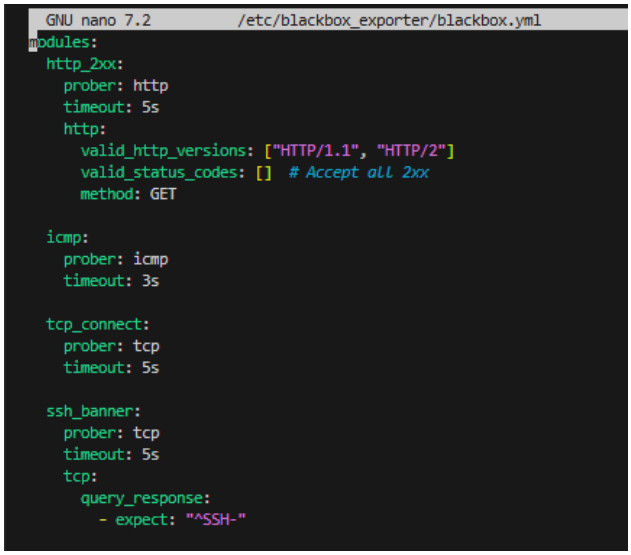
4.1 /etc/prometheus/prometheus.yml

Inclut les scrape_configs pour :

- Prometheus local/distant
- Node Exporter local
- Blackbox Exporter : modules HTTP, ICMP, TCP, SSH
- Règles d'alerte via alert.rules.yml


```yaml
# Objectif : monitorer à la fois l'instance locale et distante de Prometheus

global:
  scrape_interval: 15s     # Intervalle global d'interrogation des cibles
  evaluation_interval: 15s # Intervalle global d'évaluation des règles d'alerte
  external_labels:
    monitor: 'jitsi-cluster' # Label global utilisé pour identifier cette instance (utile pour Gr

# Configuration d'Alertmanager (désactivé ici mais prêt)
alerting:
  alertmanagers:
    - static_configs:
        - targets: ['localhost:9093'] # Port standard d'Alertmanager (non utilisé ici)

# Chargement de fichiers de règles (commenté ici par défaut)
# rule_files:
#   - "first_rules.yml"
#   - "second_rules.yml"

# ---- CIBLES SCRAPE CONFIGS ----

scrape_configs:

  # 🔵 Prometheus local : autosurveillance
  - job_name: 'prometheus_local'
    scrape_interval: 5s
    scrape_timeout: 5s
    metrics_path: /metrics
    static_configs:
      - targets: ['localhost:9090']
        labels:
          instance: 'local_prometheus'

  # 🔵 Prometheus distant : surveillance d'une autre instance
  - job_name: 'prometheus_distant'
    scrape_interval: 5s
    scrape_timeout: 5s
    metrics_path: /metrics
    static_configs:
      - targets: ['37.156.46.238:9091']
        labels:
          instance: 'distant_prometheus'

  # 🟢 Node Exporter local (CPU, RAM, disque…)
  - job_name: 'node_exporter'
    static_configs:
      - targets: ['localhost:9100']
        labels:
          instance: 'node_local'

# Fin du fichier
```

4.2 /etc/blackbox_exporter/blackbox.yml



```
GNU nano 7.2          /etc/blackbox_exporter/blackbox.yml
modules:
  http_2xx:
    prober: http
    timeout: 5s
    http:
      valid_http_versions: ["HTTP/1.1", "HTTP/2"]
      valid_status_codes: []  # Accept all 2xx
      method: GET

  icmp:
    prober: icmp
    timeout: 3s

  tcp_connect:
    prober: tcp
    timeout: 5s

  ssh_banner:
    prober: tcp
    timeout: 5s
    tcp:
      query_response:
        - expect: "^SSH-"
```

Modules activés : http_2xx, tcp_connect, icmp, ssh_banner, grpc, irc_banner, etc.

4.3 /etc/prometheus/alert.rules.yml

```
GNU nano 7.2          /etc/prometheus/alert.rules.yml
# Règles d'alerte Prometheus – modules Blackbox et Node Exporter

groups:
  - name: blackbox_alerts
    rules:

    # ALERTE : cible HTTP indisponible
    - alert: HTTP_Target_Down
      expr: probe_success{job="blackbox_exporter_http"} == 0
      for: 30s
      labels:
        severity: critical
      annotations:
        summary: "Cible HTTP indisponible ({{ $labels.instance }})"
        description: "La cible HTTP {{ $labels.instance }} est injoignable via

    # ALERTE : ICMP échoué (ping KO)
    - alert: ICMP_Down
      expr: probe_success{job="blackbox_exporter_icmp"} == 0
      for: 30s
      labels:
        severity: warning
      annotations:
        summary: "Perte ICMP (ping KO) : {{ $labels.instance }}"
        description: "Impossible de ping {{ $labels.instance }} via le module

    # ALERTE : port TCP injoignable
    - alert: TCP_Port_Down
      expr: probe_success{job="blackbox_exporter_tcp"} == 0
      for: 30s
      labels:
        severity: warning
      annotations:
        summary: "Connexion TCP échouée ({{ $labels.instance }})"
        description: "La connexion TCP à {{ $labels.instance }} via Blackbox e

    # ALERTE : HTTP trop lent (latence > 5s)
    - alert: HTTP_Response_Slow
      expr: probe_duration_seconds{job="blackbox_exporter_http"} > 5
      for: 1m
      labels:
        severity: warning
      annotations:
        summary: "Réponse HTTP lente : {{ $labels.instance }}"
        description: "Latence HTTP supérieure à 5 secondes pour {{ $labels.ins

  - name: node_exporter_alerts
    rules:
```

Exemples installés : HTTP_Target_Down / ICMP_Down / TCP_Port_Down / HTTP_Response_Slow

Ajouts possibles : High_CPU_Usage / Low_Available_Memory / Disk_Space_Low

4.3 /etc/telegraf/telegraf.conf

```toml
[[inputs.snmp]]
  agents = [ "IP_PFSENSE:161" ]
  version = 2
  community = "public" # ou ta chaîne SNMP
  name = "pfsense"

  [[inputs.snmp.field]]
    name = "uptime"
    oid = "1.3.6.1.2.1.1.3.0"
    is_tag = true

  [[inputs.snmp.table]]
    name = "interface"
    oid = "1.3.6.1.2.1.2.2"
```

5. Dashboards Grafana

- Import des dashboards JSON Blackbox :

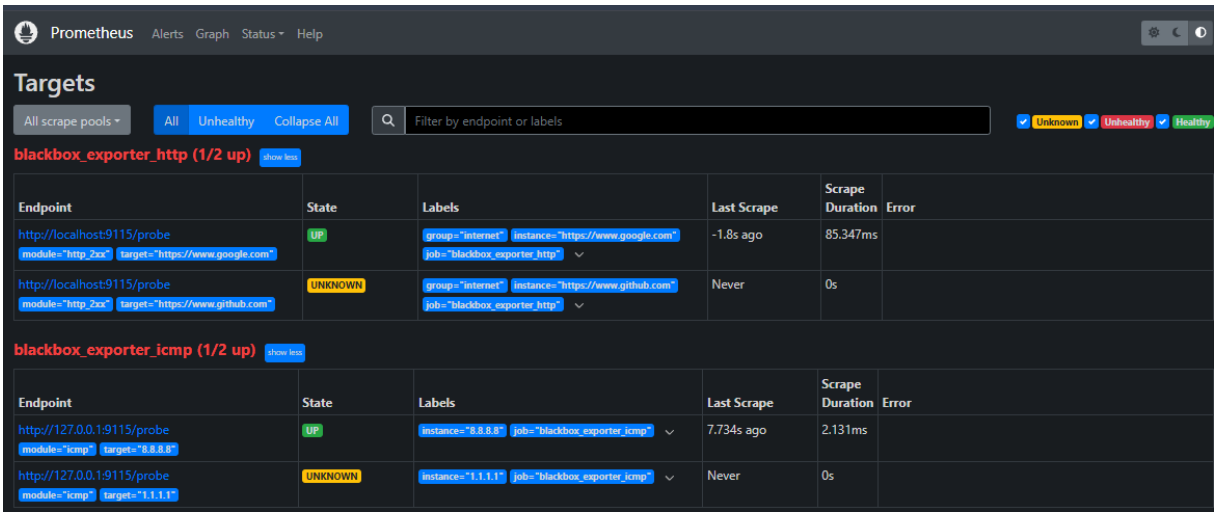
blackbox_exporter.json > ...

```json
{
  "id": null,
  "title": "Blackbox Exporter - HTTP Probes",
  "timezone": "browser",
  "schemaVersion": 26,
  "version": 1,
  "panels": [
    {
      "type": "graph",
      "title": "HTTP Probe Success",
      "datasource": "Prometheus",
      "targets": [
        {
          "expr": "probe_success{job=\"blackbox_http\"}",
          "legendFormat": "{{instance}}",
          "refId": "A"
        }
      ],
      "gridPos": {
        "x": 0,
        "y": 0,
        "w": 24,
        "h": 6
      }
    },
    {
      "type": "graph",
      "title": "HTTP Probe Duration",
      "datasource": "Prometheus",
      "targets": [
        {
          "expr": "probe_duration_seconds{job=\"blackbox_http\"}",
          "legendFormat": "{{instance}}",
          "refId": "B"
        }
      ],
      "gridPos": {
        "x": 0,
        "y": 7,
        "w": 24,
        "h": 6
      }
    }
  ]
}
```

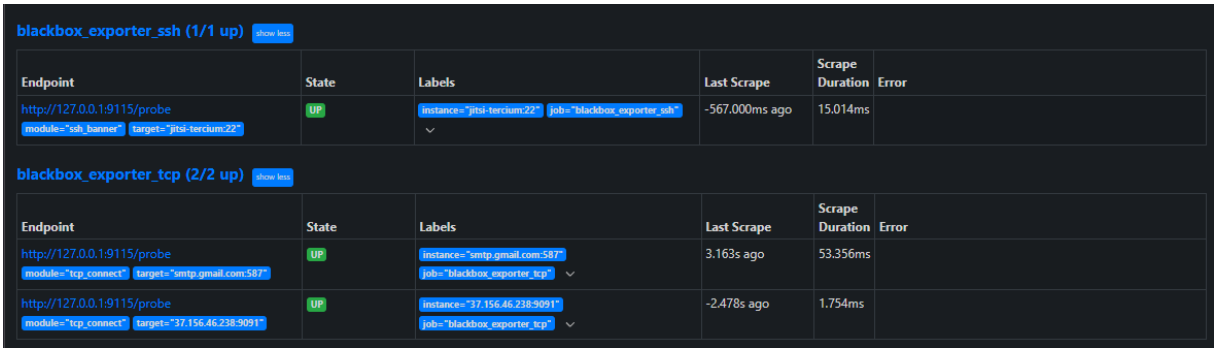
- Représentation des états HTTP/ICMP/TCP/SSH par couleur

Vérification avec Prometheus : http://37.156.46.238:9091

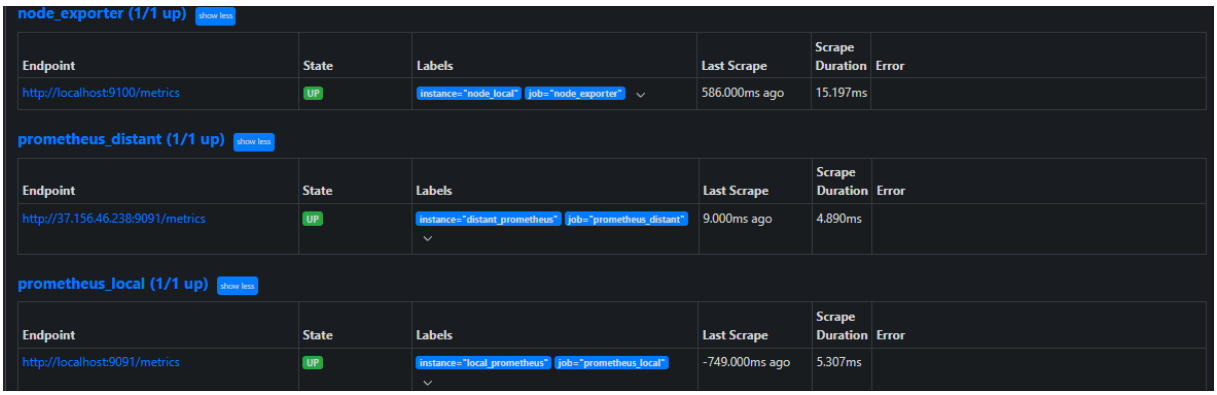
Balckbox_exporter : http & icmp



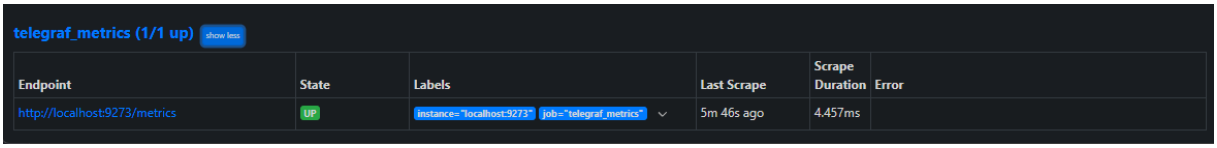
Balckbox_exporter : ssh & tcp



node_exporter & prometheus distant + local

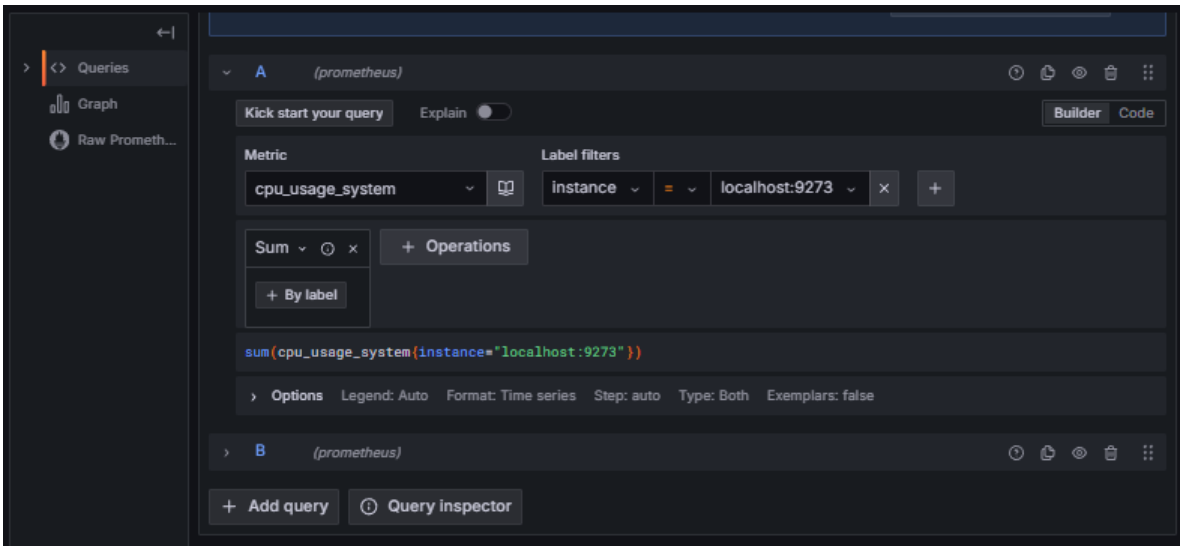


Telegraf :

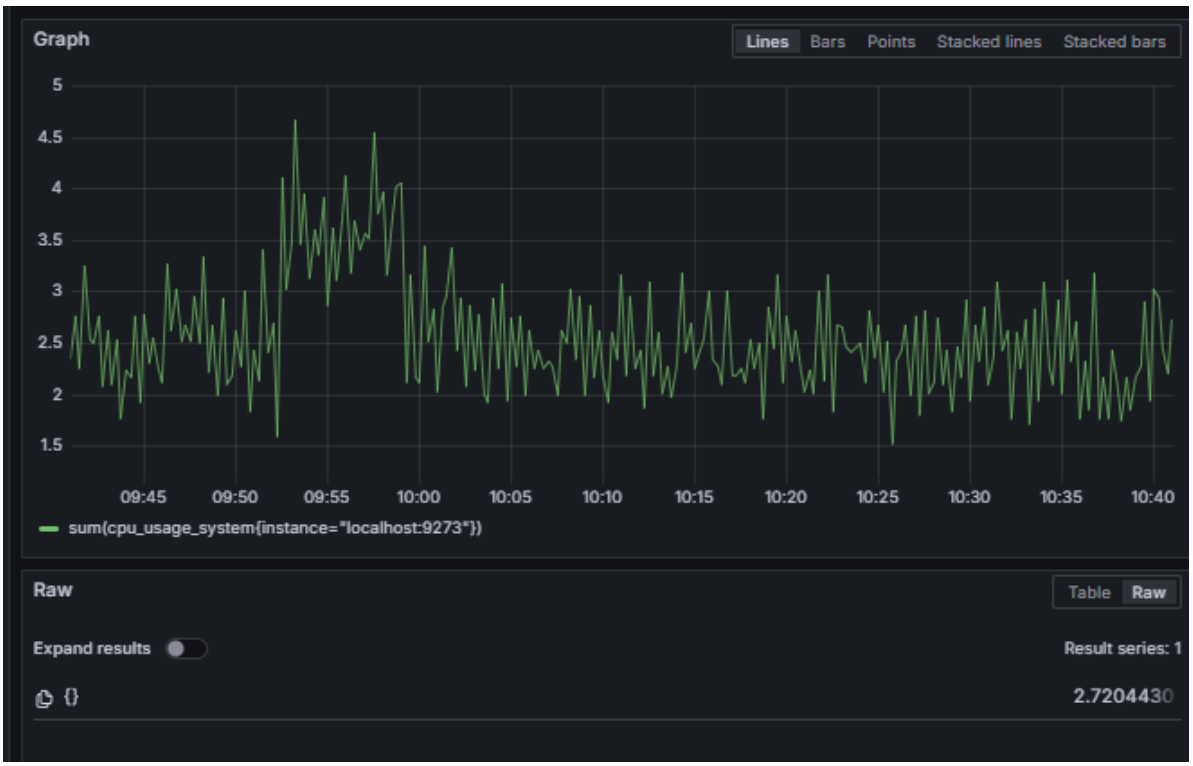


- **Analyse de la charge CPU, RAM, et I/O**

Paramétrage dans Grafana:



Résuultat :



6. Alerting : tests

- **Requête : curl -s http://localhost:9091/api/v1/alerts | jq**

Version synthétique :
cd : curl http://localhost:9091/metrics

Quelques **commandes alternatives filtrées** selon les besoins :

🔹 1. N'afficher que les 20 premières lignes (aperçu global)
cd : curl -s http://localhost:9091/metrics | head -n 20

🔹 2. Ne récupérer que les lignes contenant un mot-clé (ex. : cpu)
cd : curl -s http://localhost:9091/metrics | grep 'cpu'

🔹 3. Lister uniquement les noms des métriques (sans valeurs) **cd : curl -s http://localhost:9091/metrics | grep -v '^#' | cut -d' ' -f1 | sort -u**

🔹 4. Afficher les métriques contenant une valeur spécifique (ex. : node_memory ou up)
cd : curl -s http://localhost:9091/metrics | grep 'node_memory'
ou :
cd : curl -s http://localhost:9091/metrics | grep '^up'

🔹 5. Nombre total de métriques exposées
cd : curl -s http://localhost:9091/metrics | grep -v '^#' | wc -l

Ceux-sont des métriques Prometheus, majoritairement liées à l'environnement **Go Runtime** (métriques internes à Prometheus). Il faut des métriques plus pertinentes sur les **services supervisés** (comme **node_exporter, blackbox_exporter, etc.**).

Commandes ciblées :

🔹 1. Pour vérifier l'état de tes cibles (via up)
cd : curl -s http://localhost:9091/metrics | grep '^up'

- Résultat : up{job="node_exporter", instance="localhost:9100"} 1
- Signification : 1 = OK, 0 = HS

🔹 2. Pour extraire des métriques node_exporter (CPU, mémoire, disque)
cd : curl -s http://localhost:9091/metrics | grep 'node_cpu_seconds_total' | head -n 5
cd : curl -s http://localhost:9091/metrics | grep 'node_memory' | head -n 5

🔹 3. Pour tester les résultats d'un module Blackbox (HTTP par ex.)
cd : curl -s http://localhost:9091/metrics | grep 'probe_success'

- Cela te dira si les sondes (HTTP/ICMP/TCP) renvoient 1 (succès) ou 0 (échec).

◈ 4. Pour n'afficher que les noms des métriques disponibles
cd : curl -s http://localhost:9091/metrics | grep -v '^#' | awk '{print $1}' | sort -u

◈ 5. Version synthétique personnalisée dans un fichier **cd :** curl -s http://localhost:9091/metrics | grep -E 'up|probe_success|node_memory|node_cpu_seconds' > mini-metrics.txt

- **Affichage des alertes en console ou via Grafana (mail/SMS à activer)**

7. Résultats observés

- Toutes les sondes actives
- Réception des alertes sur latence ou coupure réseau
- Bonne visibilité des métriques dans Grafana

8. Problèmes rencontrés

- Conflit utilisateur prometheus non système → solution : suppression manuelle + purge dpkg : voir les deux manuels.

9. Outils CLI utilisés

curl http://localhost:9091/metrics
curl -s http://localhost:9091/api/v1/targets | jq
promtool check config /etc/prometheus/prometheus.yml

10. Bonnes pratiques

- Toujours vérifier les fichiers avec promtool
- Redémarrer avec systemctl daemon-reexec après modification de services
- Grouper les cibles par job_name explicite
- Annoter les dashboards pour suivi opérationnel

L'ensemble de ces documents devrait clore la section *Monitoring – Outils et installation*, car je considère que les tests par scripts (charge, alertes, sélection des bons KPI et rapports) relèvent désormais de la phase d'usage et de maintenance du système d'information.