Supervision d'infrastructure

Prometheus · Grafana · Vagrant

Mise en place d'une plateforme de monitoring sous machine virtuelle Debian, supervisant les équipements réseau du VLAN 50 (routeurs, switches, points d'accès).

Tiago Quenette

Mediaschool

Sommaire

1. Contexte et objectifs

Le projet vise à mettre en place une plateforme de supervision centralisée pour deux équipements réseau du VLAN 50, sur le **réseau privé 192.168.50.0/24** : **RT1-IRIS** (routeur Cisco 1900, 192.168.50.254 — assure aussi le DHCP du subnet) et **SW1-IRIS** (switch Cisco 2960, 192.168.50.253). L'ensemble communique en interne via le bridge *br-50* du serveur, sans exposition directe sur Internet. La supervision collecte via SNMP l'état et le trafic des interfaces, surveille la disponibilité par sondes ICMP, et présente le tout dans des tableaux de bord exploitables.

Le choix s'est porté sur le couple **Prometheus / Grafana**, standard de l'écosystème cloud-native : Prometheus pour la collecte et le stockage des métriques sous forme de séries temporelles, Grafana pour la visualisation et l'alerting.

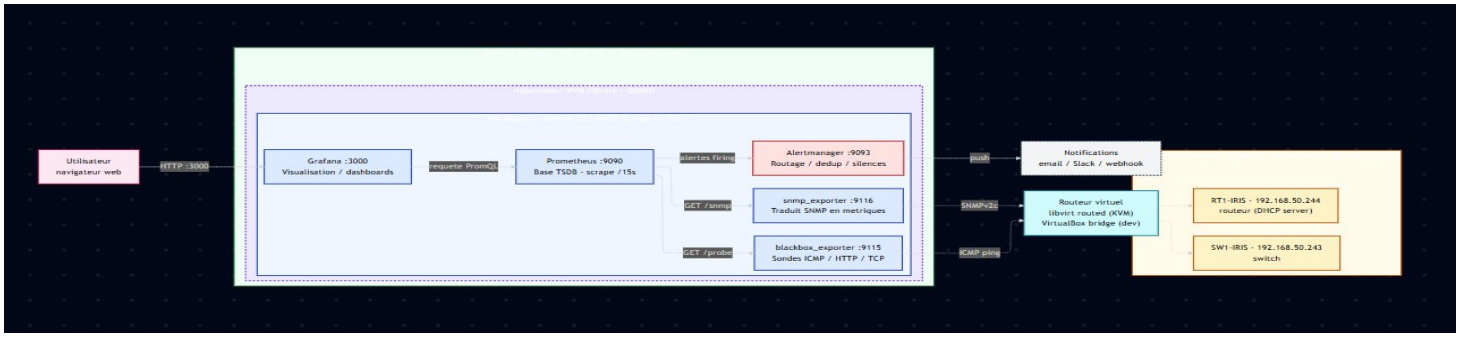
Pour garantir la portabilité et la reproductibilité du livrable, l'ensemble de la stack est encapsulée dans une **machine virtuelle Debian 12** provisionnée par **Vagrant** sous **VirtualBox**. La VM tourne en mode *headless* (sans interface graphique) — l'accès aux interfaces de supervision se fait depuis le navigateur de n'importe quel poste, via port forwarding en développement ou via l'IP publique du serveur en production. La VM est volontairement indépendante de la stack de monitoring déjà existante en conteneurs Docker sur le serveur, afin de fournir un livrable autonome et démontrable.

Objectifs techniques

- Interroger les deux équipements en SNMPv2c (interfaces, trafic, état admin/oper).
- Tester la disponibilité de chaque équipement par sonde ICMP (Blackbox exporter).
- Centraliser les alertes (Alertmanager) et les notifier sur le canal voulu.
- Exposer une interface Grafana sur le LAN pour la consultation des dashboards (avec dashboard auto-provisionné).
- Fournir un déploiement reproductible en une commande (vagrant up).

2. Architecture

Le schéma ci-dessous présente l'architecture globale et les flux de supervision. La VM Vagrant héberge l'ensemble de la stack de monitoring (Prometheus, Grafana, exporters). Elle est bridgée sur le VLAN 50 du serveur, ce qui lui permet d'atteindre directement les équipements réseau ainsi que les exporters du serveur sur leur IP LAN **192.168.50.10**.



Flux de supervision

Trait plein	Prometheus interroge en HTTP les exporters (snmp_exporter, blackbox_exporter) toutes les 15 secondes.

Trait pointillé	snmp_exporter dialogue en SNMPv2c avec les équipements réseau ; blackbox_exporter envoie des sondes ICMP pour mesurer la disponibilité.
Routeur virtuel	Toutes les sondes vers les équipements du LAN (SNMP, ICMP) transitent par la couche de routage virtuel — fournie par *libvirt routed network* côté KVM en production, et par le bridge VirtualBox côté développement. C'est ce qui permet à la VM d'avoir sa propre IP sur le LAN VLAN 50 tout en restant un objet virtualisé.
Utilisateur	Accède à Grafana en HTTP sur le port 3000 de la VM.

3. Composants de la stack

Composant	Port	Hôte	Rôle
Prometheus	9090	VM	Cœur du système. Base de données *time-series* qui interroge (« scrape ») les exporters à intervalle régulier (15 s) et stocke les valeurs localement. Interrogeable en PromQL. Évalue les règles d'alerting et envoie les alertes *firing* à Alertmanager.
Alertmanager	9093	VM	Gestion des alertes. Reçoit les alertes émises par Prometheus, les regroupe (*grouping*), supprime les doublons (*deduplication*), applique les règles d'inhibition et de silence, puis route vers les destinataires configurés (mail, Slack, webhook…).
Grafana	3000	VM	Interface web de visualisation. Lit les données dans Prometheus (datasource auto-provisionnée) et affiche tableaux de bord, graphes, alertes. Login : *admin* / *admin* au premier accès.
snmp_exporter	9116	VM	Traducteur SNMP → HTTP. Reçoit *GET /snmp?target=<ip>&module=if_mib*, interroge l'équipement en SNMPv2c et renvoie les compteurs au format Prometheus (trafic, état des interfaces, erreurs, uptime).
blackbox_exporter	9115	VM	Sondes externes. Teste la **disponibilité** sans agent installé : ping ICMP, GET HTTP, ouverture TCP, résolution DNS, validité TLS. Renvoie le succès / échec et la latence.

Différence snmp_exporter vs blackbox_exporter

Les deux exporters sont complémentaires. Le **snmp_exporter** remonte des métriques **détaillées** (octets in/out par interface, paquets en erreur, état administratif/opérationnel) à condition que l'équipement réponde en SNMP. Le **blackbox_exporter** ne remonte qu'une information binaire (*répond* / *ne répond pas*) et un temps de réponse, mais fonctionne sur n'importe quoi qui parle IP : il sert de filet de sécurité pour détecter une coupure même quand l'agent détaillé est inopérant.

4. Mise en place

4.1 Structure du projet

Le dossier *Prometheus/* contient l'ensemble du livrable. Il est versionnable et reproductible.

```
Prometheus/
■■■ Vagrantfile
■■■ provision/                (deploye dans la VM)
■   ■■■ bootstrap.sh      (script de provisioning)
■   ■■■ prometheus.yml    (configuration des scrapes)
■   ■■■ alert_rules.yml   (règles d'alerting)
■   ■■■ alertmanager.yml  (routes et receivers)
■   ■■■ grafana-datasource.yml
■   ■■■ grafana-dashboards.yml  (provisioning)
■   ■■■ grafana-dashboard.json  (dashboard IRIS)
■■■ serveur/                  (deploye sur l'hote)
■   ■■■ traefik-dynamic/
■       ■■■ grafanatiago.yml  (file provider Traefik)
■■■ test-local/               (test sur switch local)
■■■ docs/
    ■■■ documentation.pdf
    ■■■ tutoriel.md
```

4.2 Spécifications de la VM

Système	Debian 12 Bookworm (box *debian/bookworm64*)
Hyperviseur (dev)	VirtualBox sur le poste de travail Windows (mode **headless**, sans fenêtre)
Hyperviseur (prod)	**KVM** + *libvirt* + *QEMU* sur le serveur Debian (192.168.50.10)
CPU	2 vCPU
RAM	6 Go
Disque	VMDK dynamique (sparse) — taille allouée maximale issue de la box, espace réellement consommé sur le disque hôte : ~5 à 10 Go pour cette stack
Réseau	*public_network* (bridged) sur le bridge **br-50** du serveur (eno3.50). Tous les équipements communiquent dans le réseau privé **192.168.50.0/24** (VLAN 50). IP de la VM attribuée par **DHCP** du routeur RT1-IRIS
Interface	Aucune GUI dans la VM — accès uniquement via navigateur web depuis n'importe quel poste (port forwarding ou IP de la VM sur le LAN)
Provisioning	Script shell exécuté au premier *vagrant up*

4.3 Script de provisioning

Le script *provision/bootstrap.sh* est exécuté au premier boot de la VM. Il :

- met à jour APT et installe les outils de base ;

- étend automatiquement la partition racine si le disque a été agrandi ;
- ajoute le dépôt officiel Grafana (clé GPG signée) ;
- installe les paquets *prometheus*, *prometheus-alertmanager*, *prometheus-snmp-exporter*, *prometheus-blackbox-exporter* et *grafana* ;
- déploie les fichiers de configuration depuis le dossier monté */vagrant/provision/* ;
- active et démarre les services *systemd* (Prometheus, Alertmanager, exporters, Grafana).

4.4 Prérequis côté équipements réseau

Chaque équipement supervisé doit avoir **SNMP activé** avec une *community* en lecture seule. Sur Cisco IOS :

```
snmp-server community public RO
snmp-server contact <votre-contact>
snmp-server location <votre-localisation>
```

Le port UDP 161 doit être joignable depuis la VM. Si une ACL est en place sur l'équipement, ajouter l'IP de la VM (IP DHCP de la VM dans 192.168.50.0/24) dans les hôtes autorisés. La même logique s'applique aux APs (souvent via leur contrôleur) et au routeur.

4.5 Workflow de déploiement

Le projet est construit et validé sur le poste de travail **Windows**, puis transféré sur le serveur final :

- **Sur le poste Windows** : *vagrant up* dans le dossier *Prometheus/*, hyperviseur VirtualBox. À ce stade, seuls Prometheus et Grafana de la VM sont fonctionnels — les targets distants resteront *DOWN* tant que la VM n'est pas sur le bon LAN.
- **Export depuis VirtualBox** : une fois la VM validée, l'image est exportée au format OVA (*VBoxManage export prometheus-monitoring -o monitoring.ova*), puis le disque *.vmdk* contenu est converti en *.qcow2* compatible KVM avec *qemu-img convert -f vmdk -O qcow2 disk.vmdk monitoring.qcow2*.
- **Import sur le serveur** : import dans *libvirt* via *virt-manager* ou *virsh*, en branchant le disque sur l'interface réseau bridge *eno3.50* du serveur. La VM démarre alors directement sous KVM avec sa propre IP DHCP dans 192.168.50.0/24.

5. Utilisation et validation

5.1 Accès aux interfaces

La VM tournant en mode *headless*, l'accès aux interfaces se fait exclusivement par le navigateur web. Trois modes d'accès selon le contexte :

- **Depuis le poste hôte en dev (navigateur Windows)** — Vagrant expose les ports de la VM en *localhost* du PC via du port forwarding sur des ports élevés (13000, 19090, 19093). Aucune configuration réseau supplémentaire n'est nécessaire.
- **Depuis n'importe quel poste sur le LAN** — la VM est bridgée sur le VLAN 50 et obtient une IP par DHCP. Cette IP, affichée à la fin du provisioning, est joignable directement sur les ports standards (3000, 9090, 9093).
- **Depuis Internet (en production sur le serveur)** — Grafana est exposé via le reverse proxy **Traefik** déjà en place sur le serveur, sur le sous-domaine **grafanatiago.iris.a3n.fr** avec certificat Let's Encrypt automatique et redirection HTTP → HTTPS. Aucun port custom à mémoriser, juste un nom de domaine signé.

Identifiants Grafana au premier accès : **admin / admin** (changement de mot de passe demandé immédiatement).

Service	PC (port-fwd)	LAN (IP DHCP de la VM)	Internet (Traefik)
Grafana	localhost:13000	<IP-VM>:3000	grafanatiago.iris.a3n.fr
Prometheus	localhost:19090	<IP-VM>:9090	—
Alertmanager	localhost:19093	<IP-VM>:9093	—

Configuration Traefik côté serveur (production) — Grafana tournant dans une VM (et non dans un container Docker), Traefik ne peut pas la découvrir via les labels Docker. La déclaration se fait via le *file provider* de Traefik : un fichier YAML déposé dans son dossier *dynamic* (fourni dans *serveur/traefik-dynamic/grafanatiago.yml*).

```
http:
  routers:
    grafanatiago-http:
      rule: "Host(`grafanatiago.iris.a3n.fr`)"
      entryPoints: [web]
      middlewares: [grafanatiago-redirect]
    grafanatiago:
      rule: "Host(`grafanatiago.iris.a3n.fr`)"
      entryPoints: [websecure]
      tls: { certResolver: letsencrypt }
      service: grafanatiago
  middlewares:
    grafanatiago-redirect:
      redirectScheme: { scheme: https, permanent: true }
  services:
    grafanatiago:
      loadBalancer:
        servers: [{url: "http://<IP-VM>:3000"}]
        passHostHeader: true
```

La VM reçoit son IP par **DHCP** sur le LAN VLAN 50 (récupération avec *sudo virsh domifaddr monitoring*). Pour stabiliser l'IP, faire une **réservation DHCP** côté RT2-IRIS sur la MAC de la VM, sinon il faut adapter *<IP-VM>* dans le file provider Traefik à chaque changement d'IP. Côté Grafana, le *bootstrap.sh* configure automatiquement */etc/grafana/grafana.ini* avec le bon *domain* et *root_url*, avec *enforce_domain = false* pour conserver l'accès local en parallèle. Traefik recharge le file provider automatiquement, pas besoin de redémarrer.

5.2 Vérification des targets

La page */targets* de Prometheus liste l'ensemble des cibles scrapées avec leur état (*UP* ou *DOWN*). En fonctionnement nominal, on doit retrouver :

Job	Cible(s)	Attendu
prometheus	localhost:9090	UP
alertmanager	localhost:9093	UP
snmp-network	RT1-IRIS, SW1-IRIS	UP × 2
blackbox-icmp	RT1-IRIS, SW1-IRIS	UP × 2

5.3 Tableaux de bord Grafana

La datasource Prometheus et un dashboard **IRIS – Supervision réseau** sont pré-provisionnés au démarrage de la VM (fichiers *provision/grafana-datasource.yml* et *provision/grafana-dashboard.json*). Le dashboard est automatiquement présent dans Grafana au premier login, dans le dossier *IRIS*. Possibilité d'importer en plus des dashboards communautaires :

Source	Dashboard	Couvre
Provisionné	IRIS – Supervision réseau	RT1, SW1 — vue d'ensemble + détails
Grafana.com 14857	SNMP Interface Stats	Trafic générique
Grafana.com 7587	Blackbox Exporter	Disponibilité ICMP / HTTP

5.4 Règles d'alerting configurées

Les règles d'alerte sont définies dans *provision/alert_rules.yml* et organisées en trois groupes thématiques. Prometheus évalue chacune d'elles toutes les 30 secondes. Lorsqu'une condition est vraie pendant la durée *for* indiquée, l'alerte passe en état *firing* et est transmise à Alertmanager.

Catégorie	Alerte	Condition	Sévérité
Disponibilité	TargetDown	Une cible ne répond plus depuis 2 min	critical
Réseau	NetworkDeviceDown	Pas de réponse ICMP depuis 2 min	critical
Réseau	HighNetworkLatency	RTT ICMP > 100 ms pendant 5 min	warning
Réseau	SnmpInterfaceDown	Interface admin up mais oper down	warning
Réseau	SnmpHighInterfaceTraffic	Trafic > 800 Mb/s pendant 10 min	info

Routage des alertes

Le fichier *provision/alertmanager.yml* définit le routage. Le receiver par défaut est silencieux — les alertes restent consultables dans l'UI Alertmanager sur le port 9093 — et trois exemples de receivers (mail, Slack, webhook) sont fournis commentés, à activer selon le canal de notification souhaité. Une règle d'inhibition empêche les alertes *warning* de tomber en doublon quand une *critical* est déjà active sur le même hôte.

5.5 Limites et perspectives

- **Notifications** : Alertmanager est en place mais aucun receiver externe n'est branché par défaut. Pour la production, configurer SMTP ou un webhook (Slack, Discord, PagerDuty…) dans *alertmanager.yml*.
- **SNMPv3** : la configuration utilise SNMPv2c avec community *public* (en lecture). Migrer vers SNMPv3 (auth + chiffrement) renforcerait la sécurité.
- **Persistance** : les données Prometheus sont stockées dans la VM. Pour de la rétention longue, envisager *Thanos* ou *VictoriaMetrics*.
- **Reverse-proxy** : exposer Grafana derrière Nginx/Traefik avec TLS et authentification SSO.