Monitoring avec Prometheus - Thanos - Grafana

X-STRA

Présentation de l’avancée des projets dans le cadre de mon apprentissage à l’IPHC

Maître d’apprentissage : Sébastien GEIGER

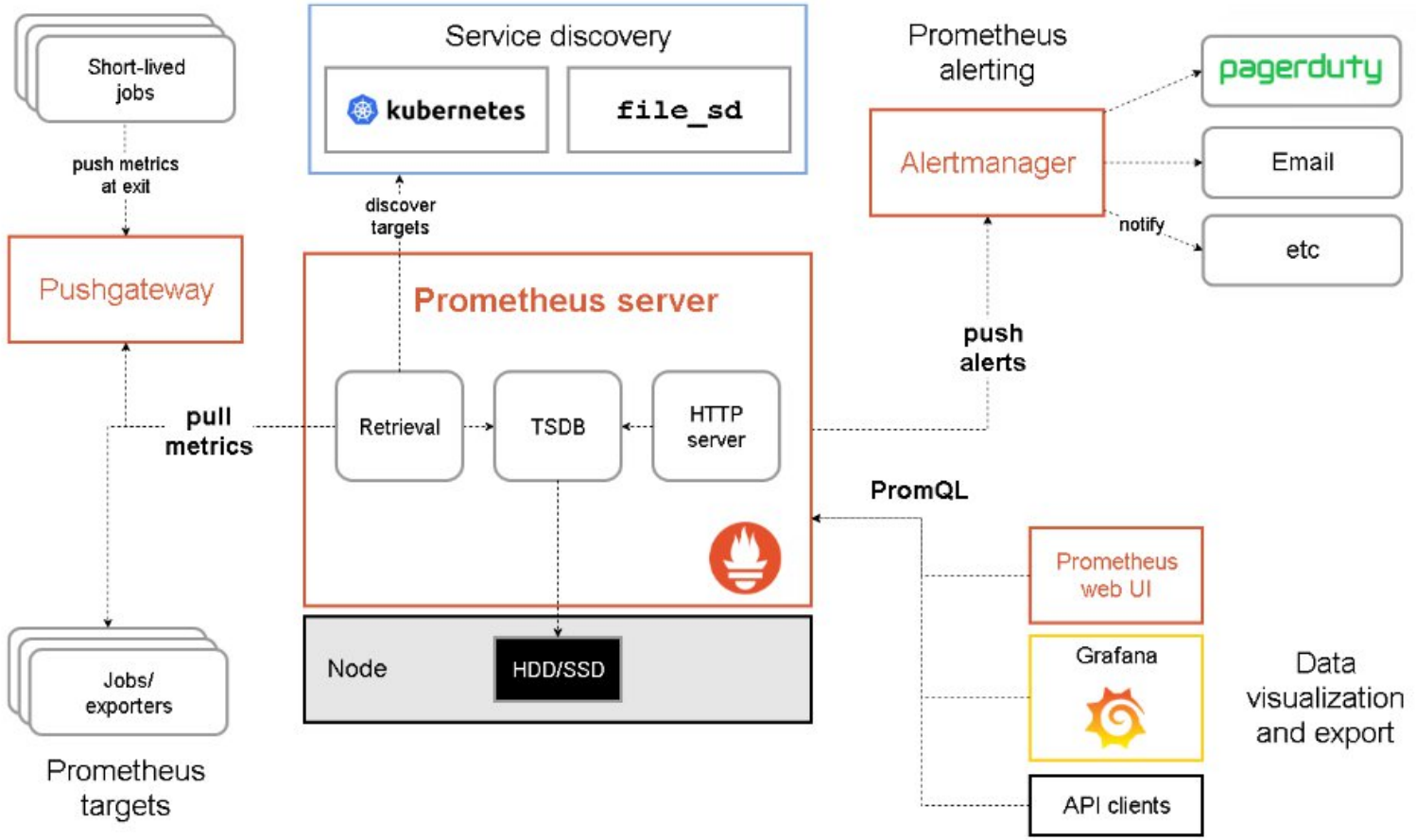
L’importance de la surveillance des serveurs et services.

- Détection et résolution préventive des problèmes
  - Identifier les problèmes, anomalies
- Optimisation des performances
  - Identifier les processus gourmands en ressources
  - Besoins futurs (capacité)
- Amélioration de la disponibilité du système
  - Minimise les temps d’arrêt
- Sécurité et conformité
  - Failles de sécurité, tentatives d’accès non autorisées

Prometheus - Grafana

- Prometheus
  - collecte de métriques
  - TSDB
  - node_exporter
- Grafana
  - visualisation de données, tableaux de bord
  - authentification ldap
  - plusieurs datasource : prometheus, thanos, elasticsearch...

Infrastructure : Prometheus – Grafana



Stockage des données

- Prometheus :
  - Stocke en moyenne 1-2 octets par échantillon
  - Par défaut :
    - 15 jours de rétention par défaut
    - Node exporter (~ 300 métriques, ~ 2064 séries de données)
    - Scraping toutes les 10 secondes

Stockage sur de longues périodes

- Intérêts :
  - Garder la saisonnalité des variations (mois,trimestres,années)
  - Détecter des tendances (augmentations, baisses)
  - Plus on s’éloigne dans le temps, moins on a besoin de « zoomer » dans les mesures
  - Pouvoir définir des temps de retentions en fonction de la résolution des datas

Stockage sur de longues périodes

- Solutions :
  - Compression :
    - Comprimer les séries de données avec lz4, zstd...
    - Delta encoding
  - Compaction :
    - Enregistrer les séries dans des blocs de 2h, puis de 1j, 1 semaine
    - Supprimer les blocs quand le délai de rétention est dépassé
  - Downsampling :
    - Réduire la fréquence d'échantillonnage d'un signal
    - Exemple sur une période d'1h:
      1*10s=360vals/h, 5m: 5*60=12val/h, 1h:1val/h
      soit un facteur de réduction de 30 sur intervalle de 5min
      et un facteur de 360 sur une intervalle d'une heure

Gestion de rétention

- Configuration des durées de rétention :
  - retention.resolution-raw=30j (données brutes gardées 30j)
  - retention.resolution-5m=90j (1 valeur toutes les 5m sur 90j)
  - retention.resolution-1h=1an (1 valeur/h sur 1 an)
- Comment sont calculés les valeurs :
  - Calcul de la moyenne
  - Conserve les valeurs min, max, nombre d’échantillons (count) et la somme (sum)
  - Cela permet de conserver la précision des requêtes de longue durée

Quel outil choisir ?

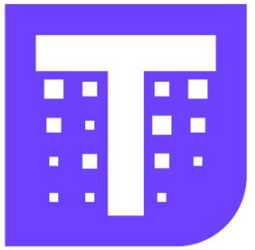
	Prometheus	Mimir	VictoriaMetrics	Thanos
Opensource	✔	✔	✘	✔
Stockage long terme	✘	✔	✔	✔
Downsampling	✘	✘	✘✔	✔
Compression	✘	✘	✔ (x7)	✘
Cache query	✘	✔	✔	✔
HA	✘	✔	✔	✔

Thanos => Métriques à différents intervalles : raw, 5m, 1h
=> On ne s'en sert pas pour compresser

Mimir => Pas de Downsampling, indiqué comme fonctionnel mais n'est pas implémenté

VictoriaMetrics => Sous double licence

Thanos



- Downsampling & Compaction
- Retention
- TSDB, stocke des métriques sur le long terme
- Compatible avec Prometheus (prometheus API)
- Mode Sidecar ou Mode Receive
- Maquette avec Docker Compose
- Installation et configuration de la version en production

Thanos Receive

- Utilise l’API de Prometheus Remote Write
- Dans le prometheus.yaml :
  - remote_write:
    - url: "http://thanos_receive:19291/api/v1/receive"

- Dans le compose.yaml :
  - command :
    - '--remote-write.address=thanos_receive:19291'

Thanos Receive (relabel.yml)

```
- action: keep
  regex: "node_filesystem_.*|node_network_receive_b.*|node_network_transmit_b.*|
node_disk_.*|node_memory_.*|node_load1|node_procs_running|node_cpu_seconds_total
|thanos_.*|rabbitmq_.*|instance_.*|up.*"
  source_labels: ["__name__"]

- action: drop
  regex: "node_filesystem_device_error|node_filesystem_f.*|node_filesystem_reado
nly"
  source_labels: ["__name__"]

- action: drop
  regex: "node_filesystem_size_bytes;(tmpfs|nfs|nfs4|vfat|fuse)"
  source_labels: ["__name__","fstype"]

- action: drop
  regex: "node_filesystem_size_bytes;(/home|/|/tmp/snap.*)"
  source_labels: ["__name__","mountpoint"]

- action: drop
  regex: "rabbitmq_a.*|rabbitmq_io_.*|rabbitmq_erlang_.*"
  source_labels: ["__name__"]
```

Thanos Architecture

- **Prometheus :** Collecte les métriques, puis les envoie au Thanos Receiver via l'API Remote Write.
- **Thanos Receiver :** Reçoit les métriques en temps réel, prend en compte le fichier de relabel et les écrit dans le stockage objet.
- **Storage :** Agit comme le stockage centralisé, contenant les blocs de données compressés.
- **Thanos Compactor :** Optimise et compacte les blocs de données dans le stockage pour améliorer les performances des requêtes et réduire l'espace utilisé.
- **Thanos Store Gateway :** Récupère les données historiques depuis le stockage et les rend accessibles pour les requêtes via Thanos Query.
- **Thanos Query :** L'interface permettant d'interroger à la fois les données en temps réel (via le Receiver) et les données historiques (via le Store Gateway).

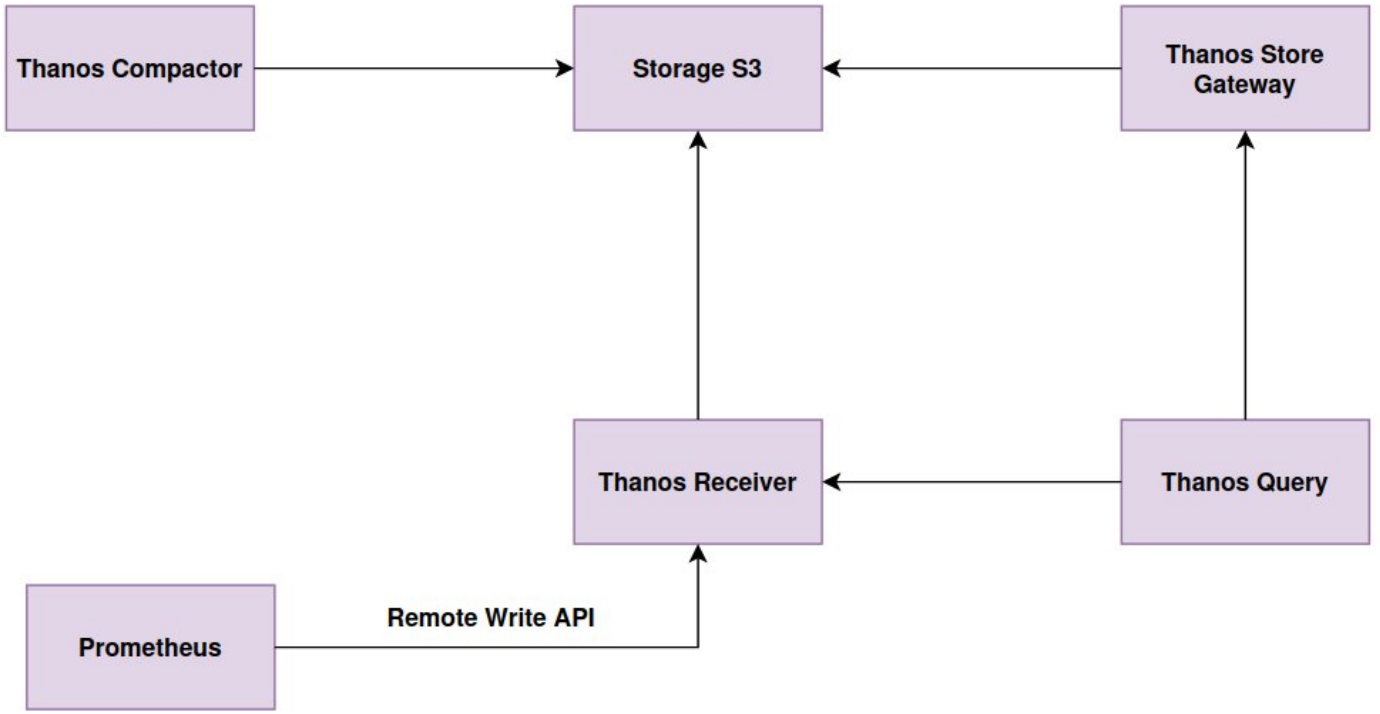


Tableau de bord Node

- Exporter : node_exporter
- Job : cephosd, li, ui, stockage, wn
- Panels :
  - charge moyenne
  - process actifs
  - trafic moyen reçu
  - quantité totale, libre, mise en cache, buffers
  - temps d’accès read/write

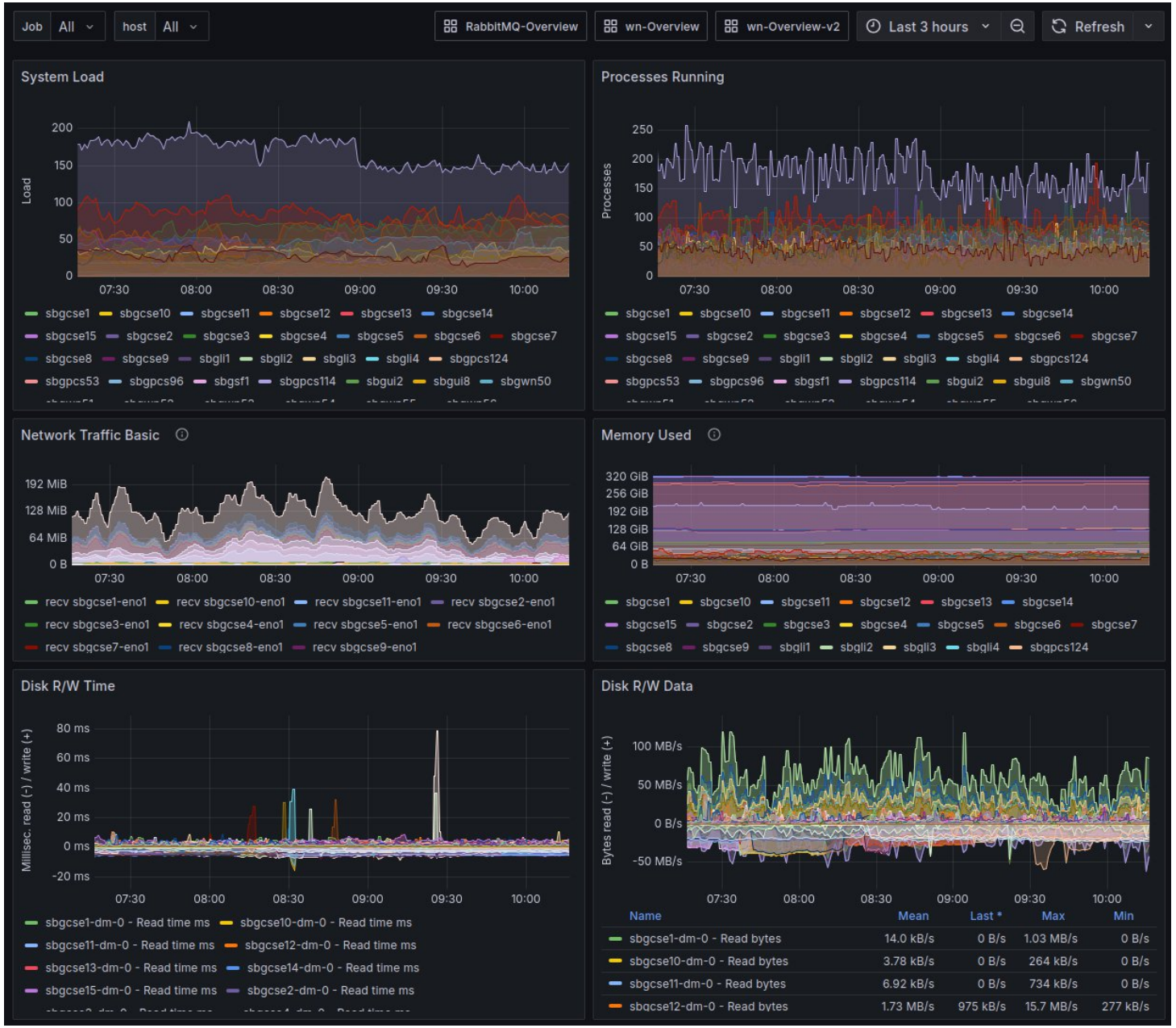


Tableau de bord CEPH

- Exporter : ceph_exporter
- Cluster : Etat, Capacity, Débit, opérations Read, Write
- OSD : Nombre UP et Down...

Tableau de bord OpenStack

- Exporter : prometheus-openstack-exporter

- volume cinder utilisé
- nombre de cpu...

Tableau de bord RabbitMQ

- Exporter : rabbitmq_exporter
- Nodes, Queuds Messages, Incoming Messages, Outgoing Messages, Queues, Channels, Connections

Tableau de bord Grille v2

- Exporter : node_exporter

- Minimum CPU, Moyenne CPU, Max CPU
- Mémoire Totale Libre
- Network Traffic

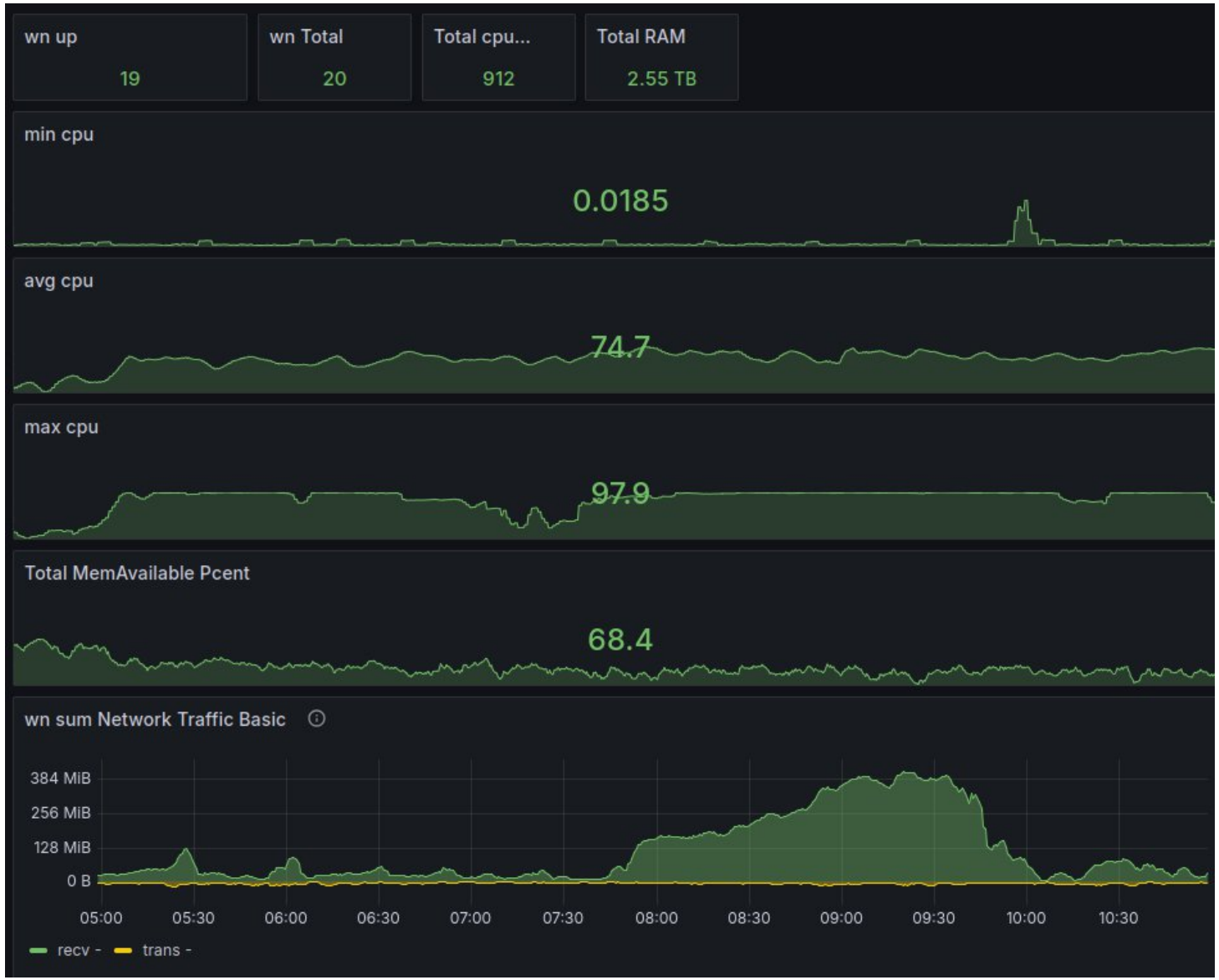


Tableau de bord Grille v2

- Utilisation des CPU pour chaque machine
- De 0 a 124 % (voir la surcharge)
- En fonction de l’utilisation des CPU les couleurs changent

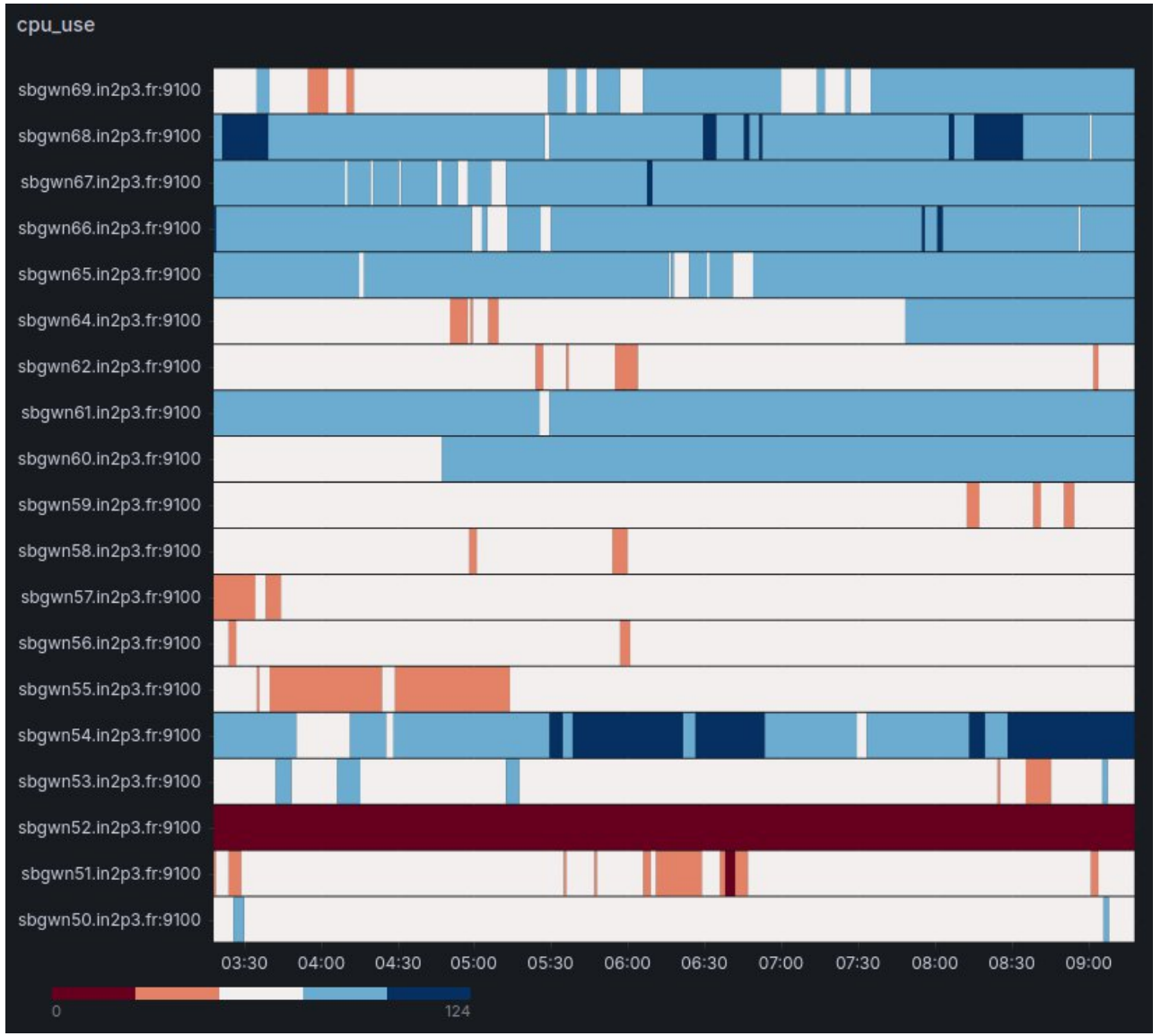


Tableau de bord Grille

- Last 2 days :
  Lundi 17 – Mardi 18

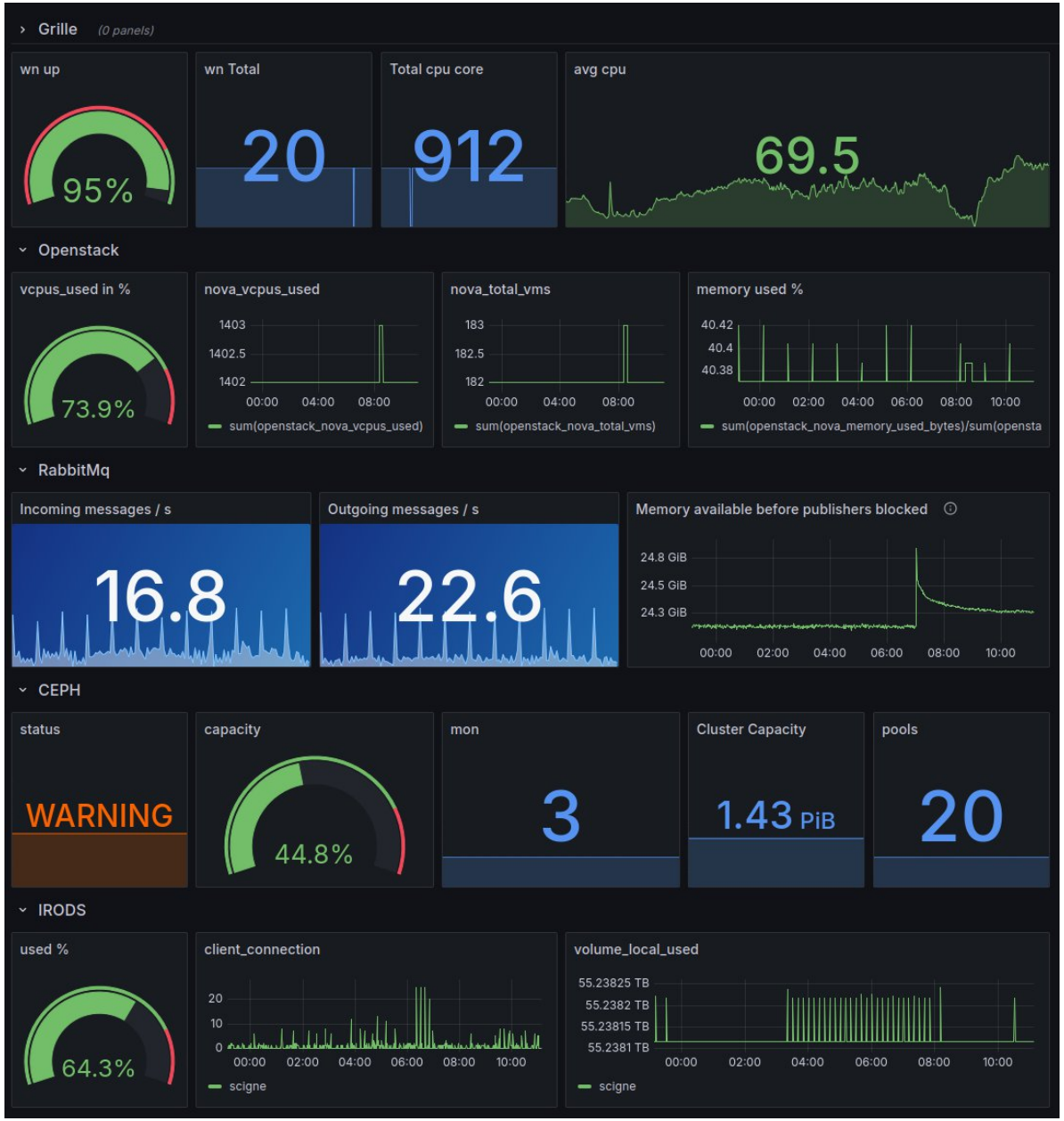
Tableau de bord Grille

- Last 24 hours :

Tableau de bord Irods

- Volume global used
- Volume local total/used
- Network in/out

Overview


Grille (0 panels)
wn up
95%
wn Total
20
Total cpu core
912
avg cpu
69.5
Openstack
vcpus_used in %
73.9%
nova_vcpus_used
1403
1402.5
1402
00:00 04:00 08:00
sum(openstack_nova_vcpus_used)
nova_total_vms
183
182.5
182
00:00 04:00 08:00
sum(openstack_nova_total_vms)
memory used %
40.42
40.4
40.38
00:00 02:00 04:00 06:00 08:00 10:00
sum(openstack_nova_memory_used_bytes)/sum(opensta
RabbitMq
Incoming messages / s
16.8
Outgoing messages / s
22.6
Memory available before publishers blocked
24.8 GiB
24.5 GiB
24.3 GiB
00:00 02:00 04:00 06:00 08:00 10:00
CEPH
status
WARNING
capacity
44.8%
mon
3
Cluster Capacity
1.43 PiB
pools
20
IRODS
used %
64.3%
client_connection
20
10
0
00:00 02:00 04:00 06:00 08:00 10:00
scigne
volume_local_used
55.23825 TB
55.2382 TB
55.23815 TB
55.2381 TB
00:00 02:00 04:00 06:00 08:00 10:00
scigne

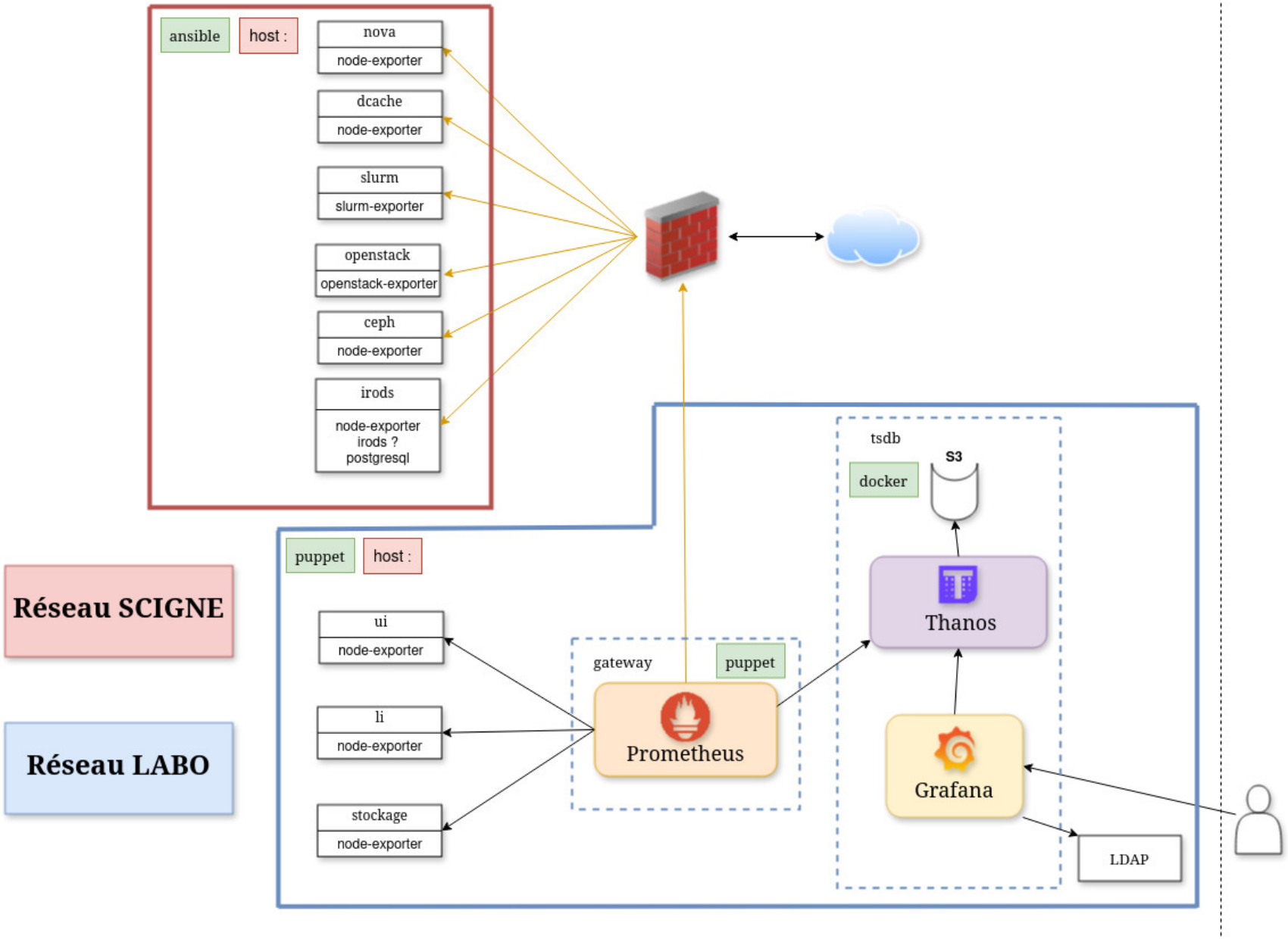
Autres tableaux de bord

- Tableaux de bord en cours :
  - Gestionnaire Slurm
  - Températures des serveurs et des disques
- D’autre tableaux de bord pour d’autres services :
  - Webserver : Apache, Nginx
  - SGDB : Mysql, postgres...
  - Métriques réseaux
  - Métriques environnementales :
    température, refroidissement passif (free cooling), éclairage, hydrométrie, polluant, acoustique
- Et encore plus : https://grafana.com/grafana/dashboards/

Alertes & Notifications

- Prometheus
  - Alert manager, règles promQL
  - Avertissement par mail ou etc
  - Géré par les admins prometheus
  - https://prometheus.io/docs/alerting/latest/alertmanager/
- Grafana
  - Règles grafana, mais même métriques
  - Notification : mail, slack, telegram
  - Géré par les Editor, créateur des tableaux de bords
  - https://grafana.com/docs/grafana/latest/alerting/notifications/

Schéma infrastructure



Automatisation des configurations

- Ansible
  - Rôle Nagios
    - NRPE
    - compléter les vérifications Hardware des services :
      Raid logiciel, raid matériel, IPMI, SNMP, firmware...
    - État du réseau
  - Rôle Prometheus
    - Synchroniser les serveurs depuis l'inventaire
    - Configurer les exporters en fonction des types de services
  - Rôle Syslog
    - Centraliser les logs syslog
- Puppet
  - Compléter les configurations des sondes et des exporters des serveurs