Network Système & Admin

T8-NSA-800_msc2024

MONITORING & RESILIENCE

Documentation du projet

Groupe 2

- Vincent Pokpa Sakouvogui
- Nassym Alassane
- Lewis Guillaume Gboyou
- Harold Da-Costa

Table des Matières

Contexte

Le monitoring ou la supervision informatique permet d'analyser en temps réel l'état du système informatique et l'état du réseau informatique à des fins préventives. Il permet d'alerter en cas de dysfonctionnement des systèmes d'information et de pouvoir ainsi agir le plus rapidement possible afin d'éviter une catastrophe.

Mission

Notre mission est de garantir une haute disponibilité des services en appliquant un système de surveillance et de gestion des incidents pour un client.

Objectifs

Pour ce faire, nous procédons par l'installation des services pour la surveillance des systèmes, ensuite l'établissement d'une politique de sauvegarde bien détaillée et spécifique au système. Puis nous terminons par l'établissement d'un plan de réponse aux incidents pour une résilience du système.

Méthodologie

Dans la réalisation de nos objectifs l'approche agile a été adopté. Agile est une approche de gestion de projet qui vise à offrir une réponse rapide et flexible aux changements dans un environnement de travail en constante évolution. L'approche Agile met l'accent sur la collaboration en équipe, la communication régulière avec les parties prenantes, l'itération rapide et l'amélioration continue.

Matrice des Compétences

SKILL MATRIX								
	Vincent Sakouvogui		Alassne Nassym		Lewis Gboyou		Harold Da-Costa	
Capabilities	Proficiency	Interest	Proficiency	Interest	Proficiency	Interest	Proficiency	Interest
Linux	0	0	1	1	2	1	0	0
Ansible	3	0	0	1	0	1	2	1
Docker	1	1	1	1	1	1	3	1
Grafana	3	1	0	1	0	1	2	1
Alert Manager	3	0	0	0	0	0	3	1
Prometheus	1	1	0	1	0	0	0	0
Database	0	1	1	1	2	1	0	0
JavaScript	0	1	2	1	2	1	2	1
PostGreSQL	1	1	2	0	2	1	3	1
Data Security	0	0	0	1	2	1	0	1
Injection SQL	0	0	1	1	2	1	0	1
Writing	2	0	0	1	2	1	3	1
Node.JS	2	1	2	1	0	1	2	1
Canva	1	0	2	0	1	1	2	1
Teams	2	1	1	1	2	1	1	1

Proficiency level

0 = No capability

1 = Basic level

2 = Intermediate level

3 = Advanced level

Interest

0 = Has no interest in applying this capability

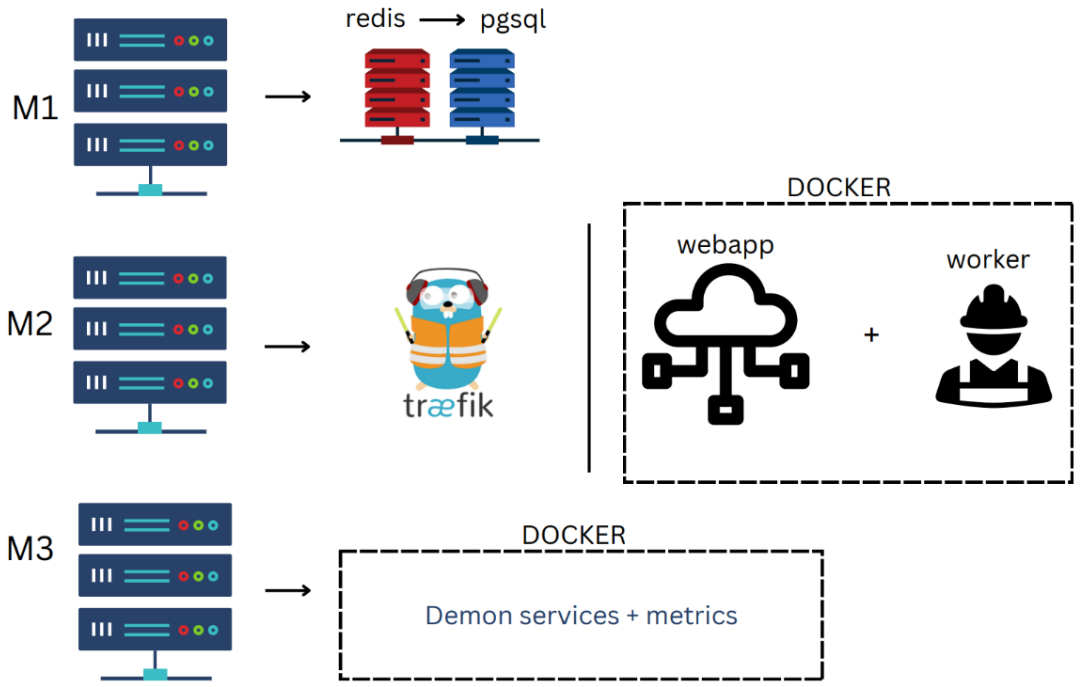
1 = Is interested in applying this capability

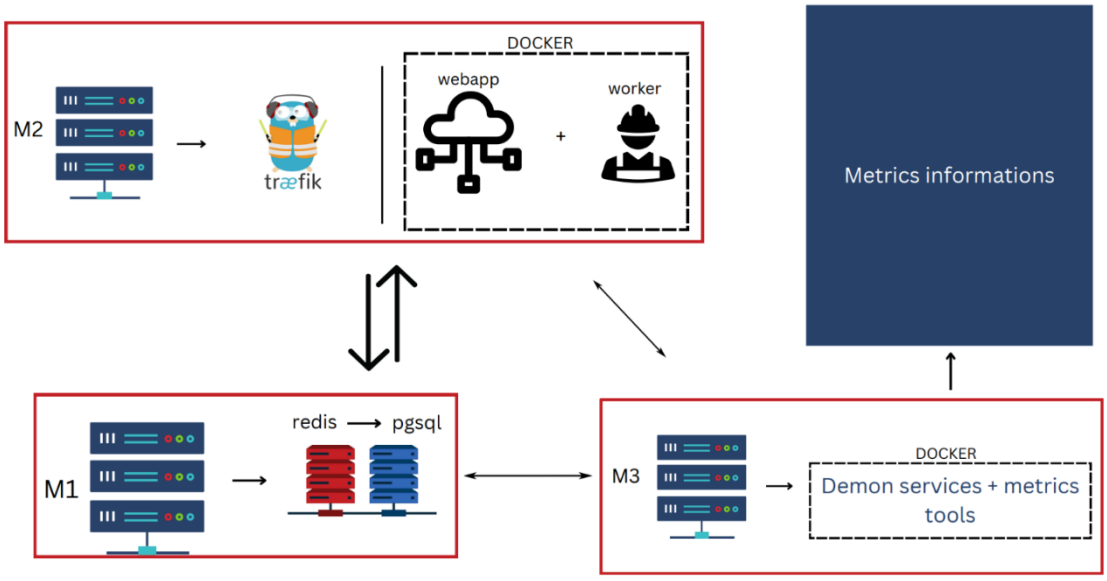
N°	Tâche	Sprint	Temps
1	Etudier et concevoir l'architecture du système entier	Etude Architecture	1H
2	Etudier et comprendre Prometheus	Etude Architecture	½ H
3	Etudier et comprendre Node Exporter	Etude Architecture	½ H
4	Etudier et comprendre Traefix	Etude Architecture	½ H

5	Etudier et comprendre Alertmanager	Etude Architecture	½ H
6	Etudier et comprendre LogRotate	Etude Architecture	½ H
7	Installater les services: node-exporter, postgres-exporter, Cadvisor, prometheus, grafana, alertmanager	Installation des services	3H
8	Configurer les services: prometheus, grafana, alertmanager	Configuration des services	3H
9	Sélectionner les métrics necessaires	Configuration des services	1H
10	Configurer les graphs	Configuration des services	2H
11	Implementer le BackUp	Mise en Place de la politique de Sauvegarde	2 H
12	Restorer le BackUp	Mise en Place de la politique de Sauvegarde	2 H
13	Gérer les Incidents	Gestion des Incidents	2 H

Etude Architecture

- ✓ Etude et apprehension de l'architecture du système entier





La première machine contient la base de données entière avec les services PostgreSQL et Redis. Les deux services fonctionnent sur cette même machine, ce qui permet de stocker et de gérer les données de manière efficace. La base

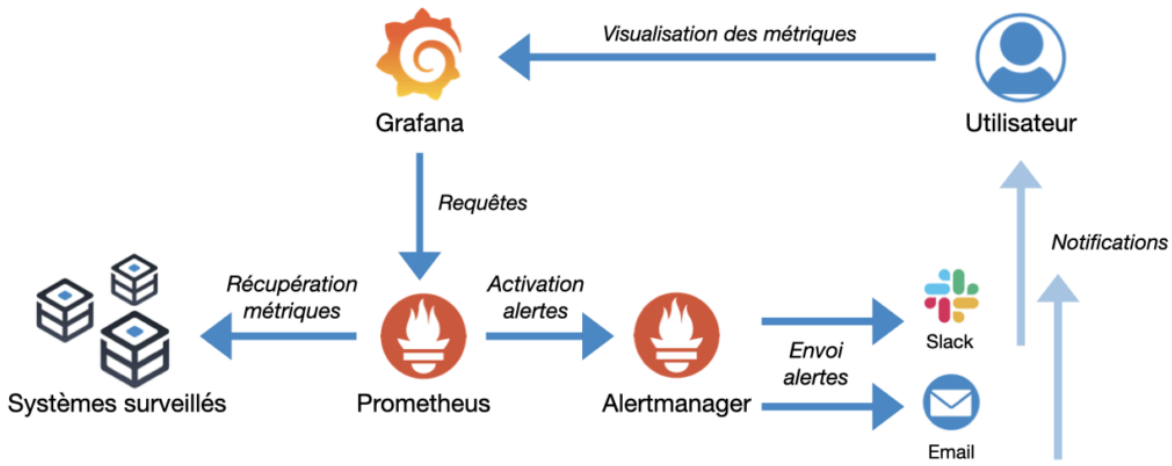
de données peut être utilisée par l'application pour stocker et récupérer les données nécessaires à son fonctionnement.

La deuxième machine contient l'application elle-même, avec une page de vote, une page de résultat et un worker exécuté sous forme de services Docker. La machine est également équipée d'un service Traefik, qui permet de router le trafic des utilisateurs vers les différents services de l'application. Le worker peut être utilisé pour effectuer des tâches de fond telles que la mise à jour de la base de données ou l'envoi de courriels.

La troisième machine ne contient qu'un démon Docker. Cette machine est utilisée pour installer des outils d'administration, de sauvegarde et de surveillance, tels que des outils de gestion des conteneurs, des outils de sauvegarde automatisés ou des outils de surveillance des performances. Ces outils peuvent être utilisés pour surveiller l'état des machines et des services dans l'ensemble de l'architecture, ainsi que pour effectuer des opérations de maintenance telles que des sauvegardes régulières de la base de données.

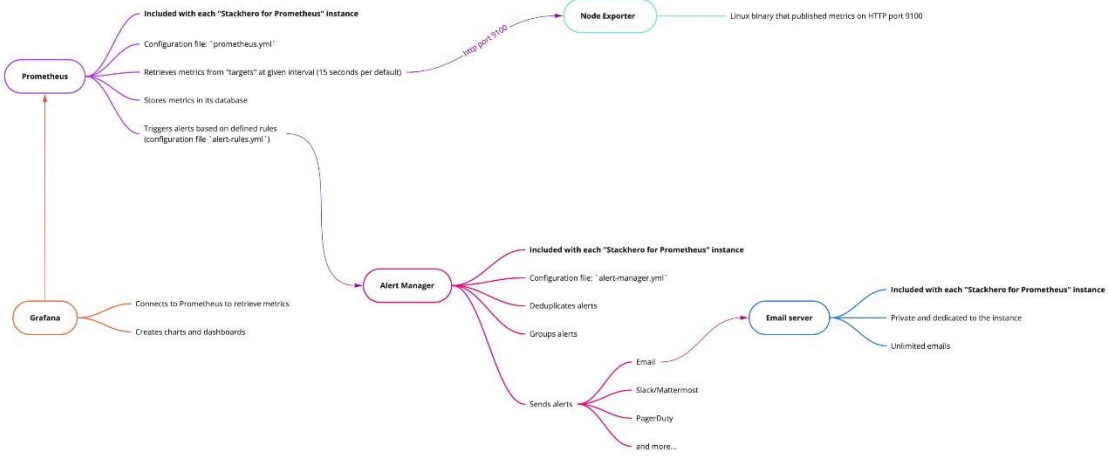
En résumé, cette architecture utilise trois machines distinctes pour séparer les différentes parties de l'application et les différents services, afin de les rendre plus modulaires, évolutifs et résilients. La première machine contient la base de données, la deuxième machine contient l'application et la troisième machine est utilisée pour installer des outils d'administration et de surveillance pour l'ensemble de l'architecture.

- ✓ Etude et apprehension de Prometheus



Le service Prometheus est un outil puissant et flexible pour la surveillance en temps réel des applications cloud natives. Il est facile à installer et à utiliser, avec une grande communauté d'utilisateurs et de développeurs qui offrent de nombreux plugins et intégrations pour étendre ses fonctionnalités. Si vous cherchez un outil de surveillance pour votre infrastructure, le service Prometheus est certainement un choix solide.

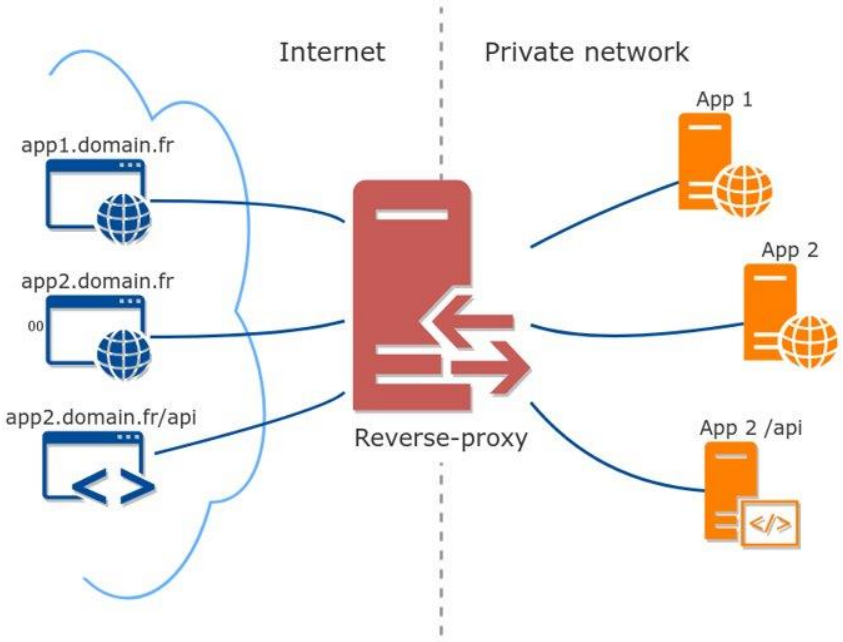
✓ Etude et apprehension de Node Exporter



Node Exporter est un outil de collecte de métriques système flexible et facile à utiliser, qui peut être utilisé pour surveiller les performances des nœuds

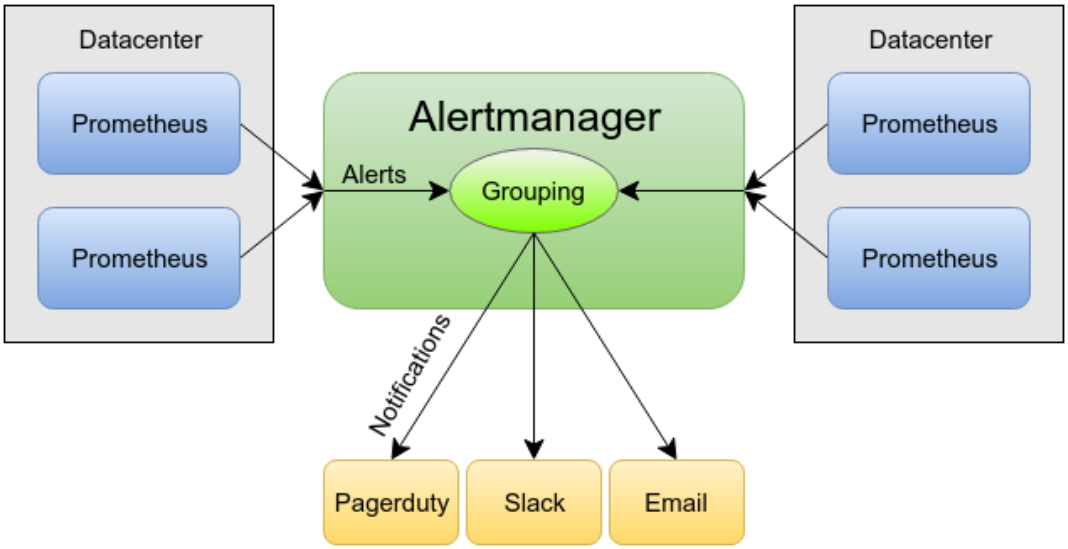
individuels dans un environnement de cluster. Avec sa compatibilité multiplateforme et sa haute configurabilité, il est devenu un outil populaire dans les environnements de cloud computing pour surveiller les performances de manière efficace et précise.

✓ Etude et apprehension de Traefix



Traefik est un outil puissant et polyvalent pour la gestion de trafic dans les environnements cloud natives. Il offre une configuration dynamique, des fonctionnalités de sécurité avancées et une compatibilité avec de nombreux fournisseurs de services cloud. Sa facilité d'utilisation et son interface utilisateur intuitive en font un choix populaire parmi les utilisateurs qui cherchent à gérer leur infrastructure de manière efficace et sécurisée.

✓ Etude et apprehension de Alert Manager



Alertmanager est un outil open-source qui permet de gérer les alertes générées par les systèmes de surveillance tels que Prometheus. Il est utilisé pour agréger et trier les alertes afin de les envoyer aux destinataires appropriés.

Alertmanager permet de configurer des règles de routage des alertes en fonction de divers critères tels que la gravité, la source et la durée. Il peut également supprimer les alertes en double et les combiner pour en faire des groupes.

L'outil permet également de définir les destinataires des alertes en fonction de différents canaux tels que Slack, e-mail, SMS ou PagerDuty. Les destinataires peuvent être organisés en groupes, ce qui permet une gestion plus efficace des alertes.

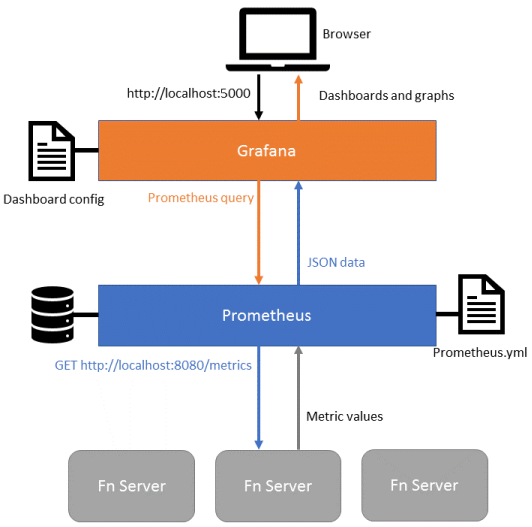
Alertmanager est hautement configurable et peut être utilisé avec différents systèmes de surveillance tels que Prometheus, Graphite et InfluxDB. Il est facile à installer et à utiliser, ce qui permet une configuration rapide et efficace.

Grâce à sa flexibilité et à sa facilité d'utilisation, Alertmanager est devenu un outil populaire pour la gestion des alertes dans les environnements de surveillance. Il

permet une gestion efficace et précise des alertes, ce qui permet aux administrateurs de surveiller leur infrastructure plus efficacement.

Alertmanager est un outil de gestion d'alertes hautement configurable et facile à utiliser qui permet une gestion efficace et précise des alertes générées par les systèmes de surveillance tels que Prometheus. Avec sa capacité à configurer des règles de routage, à définir les destinataires des alertes et à gérer les alertes en double, il est devenu un choix populaire pour la gestion des alertes dans les environnements de surveillance.

✓ Etude et apprehension de Grafana



Grafana est un outil de visualisation et d'analyse de données flexible et facile à utiliser qui permet de surveiller et d'analyser les performances de l'infrastructure en temps réel. Avec sa capacité à collecter et à visualiser des données à partir de différentes sources, ainsi que sa variété de fonctionnalités de visualisation et de configuration d'alertes, il est devenu un choix populaire pour la visualisation et l'analyse de données dans les environnements de cloud computing.

Connection aux Machines

Serveur Web

sudo ssh admusr@default-group3-web963538.francecentral.cloudapp.azure.com -i (chemin key)

Serveur Docker

sudo ssh admusr@default-group3-docker056103.francecentral.cloudapp.azure.com -i (chemin key)

Serveur Database

Sudo ssh admusr@default-group3-database333590.francecentral.cloudapp.azure.com -i (chemin key)

passphrase for key 'group-3_rsa': BU6MQrFYwop/BmlYlBW+nQ

Installation des services

✓ Installation de node_exporter

1- Dans un premier temps, nous allons télécharger le Node Exporter sur toutes les machines :

```
Wget https://github.com/prometheus/node_exporter/releases/download/v0.15.2/node_exporter-0.15.2.linux-amd64.tar.gz
```

2- Extraire l'archive téléchargée

```
tar -xf node_exporter-0.15.2.linux-amd64.tar.gz
```

3- Déplacez le binaire node_exporter vers /usr/local/bin :

```
sudo mv node_exporter-0.15.2.linux-amd64/node_exporter /usr/local/bin
```

4- Supprimez les fichiers résiduels avec :

```
rm -r node_exporter-0.15.2.linux-amd64*
```

5- Ensuite, nous allons créer des utilisateurs et des fichiers de service pour node_exporter.

Pour des raisons de sécurité, il est toujours recommandé d'exécuter tous les services/démons dans des comptes distincts. Ainsi, nous allons créer un compte utilisateur pour node_exporter. Nous avons utilisé l'indicateur -r pour indiquer qu'il s'agit d'un compte système et défini le shell par défaut sur **/bin/false** en utilisant **-s** pour empêcher les connexions.

```
sudo useradd -rs /bin/false node_exporter
```

6- Nous allons par la suite, créer un fichier d'unité systemd afin que node_exporter puisse être lancé au démarrage. sudo nano /etc/systemd/system/node_exporter.service

```
[Unit]
Description=Node Exporter
After=network.target

[Service]
User=node_exporter
Group=node_exporter
Type=simple
ExecStart=/usr/local/bin/node_exporter

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

7- Puisque nous avons créé un nouveau fichier unité, nous devons recharger le démon **systemd**, configurer le service pour qu'il s'exécute toujours au démarrage et le démarrer :

```
sudo systemctl daemon-reload
sudo systemctl enable node_exporter
sudo systemctl start node_exporter
```

✓ Installation de postgres_exporter

1- L'étape suivante consiste à télécharger et installer postgres-Exporter uniquement sur le serveur database.

```
sudo mkdir /opt/postgres_exporter

cd /opt/postgres_exporter

sudo wget https://github.com/wrouesnel/postgres_exporter/releases/download/v0.5.1/postgres_exporter_v0.5.1_linux-amd64.tar.gz

sudo tar -xzvf postgres_exporter_v0.5.1_linux-amd64.tar.gz

cd postgres_exporter_v0.5.1_linux-amd64

sudo cp postgres_exporter /usr/local/bin

cd /opt/postgres_exporter

sudo nano postgres_exporter.env

sudo useradd -rs /bin/false postgres

sudo nano /etc/systemd/system/postgres_exporter.service

sudo cat /etc/systemd/system/postgres_exporter.service

sudo systemctl daemon-reload

sudo systemctl start postgres_exporter

sudo systemctl enable postgres_exporter
```

✓ Installation de CAdvisor

1- L'étape suivante consiste à télécharger et installer CAdvisor uniquement sur le serveur web et docker.

```
sudo docker run \

--volume=/:/rootfs:ro \

--volume=/var/run:/var/run:rw \

--volume=/sys:/sys:ro \
```

```
--volume=/var/lib/docker/:/var/lib/docker:ro \
--publish=8080:8080 \
--detach=true \
--name=cadvisor \
google/cadvisor:latest
```

✓ Installation de Prometheus

1- L'étape suivante consiste à télécharger et installer Prometheus uniquement sur le serveur Docker.

```
wget https://github.com/prometheus/prometheus/releases/download/v2.1.0/prometheus-2.1.0.linux-amd64.tar.gz
```

2- Extraire l'archive Prometheus :

```
tar -xf prometheus-2.1.0.linux-amd64.tar.gz
```

3- Déplacez les binaires vers /usr/local/bin :

```
sudo mv prometheus-2.1.0.linux-amd64/prometheus prometheus-2.1.0.linux-amd64/promtool /usr/local/bin
```

4- Maintenant, nous devons créer des répertoires pour les fichiers de configuration et autres données prometheus.

```
sudo mkdir /etc/prometheus /var/lib/prometheus
```

5- Ensuite, nous déplaçons les fichiers de configuration dans le répertoire que nous avons créé précédemment :

```
sudo mv prometheus-2.1.0.linux-amd64/consoles prometheus-2.1.0.linux-amd64/console_libraries /etc/prometheus
```

7- Enfin, nous pouvons supprimer les fichiers restants car nous n'en avons plus besoin :

```
rm -r prometheus-2.1.0.linux-amd64*
```

✓ Configuration de Prometheus

Après avoir installé Prometheus, nous devons configurer Prometheus pour l'informer des points de terminaison HTTP qu'il doit surveiller. Prometheus utilise le format YAML pour sa configuration.

Allez dans **/etc/hosts** et ajoutez les lignes suivantes, remplacez x.x.x.x par l'adresse IP correspondante de la machine

x.x.x.x prometheus-target-1

x.x.x.x prometheus-target-2

```
127.0.0.1 localhost

20.19.174.118 databasemachine

20.199.116.255 webmachine
```

Nous **utiliserons /etc/prometheus/prometheus.yml** comme fichier de configuration

```
global:
  scrape_interval: 10s
rule_files:
  - alert.rules.yml
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      - 'localhost:9093'
scrape_configs:
  - job_name: 'prometheus_metrics'
    scrape_interval: 5s
```

```
    static_configs:
      - targets: ['20.19.173.248:9090']
  - job_name: 'node_exporter_metrics'
    scrape_interval: 5s
    static_configs:
      - targets: ['localhost:9100','default-group3-web963538.francecentral.cloudapp.azure.com','default-group3-web963538.francecentral.cloudapp.azure.com:9100','default-group3-database333590.francecentral.cloudapp.azure.com']
```

Dans ce fichier, nous avons défini un intervalle de grattage par défaut de 10 secondes. Nous avons également défini deux sources de métriques, nommées prometheus_metrics et node_exporter_metrics. Pour les deux, nous avons défini l'intervalle de grattage sur 5 secondes, remplaçant la valeur par défaut. Ensuite, nous avons précisé les emplacements où ces métriques seront disponibles. Prometheus utilise le port 9090 et node_exporter utilise le port 9100 pour fournir leurs métriques.

Enfin, nous modifierons également la propriété des fichiers que Prometheus utilisera :

```
sudo useradd -rs /bin/false prometheussudo chown -R prometheus: /etc/prometheus /var/lib/prometheus
```

Ensuite, créons un fichier d'unité systemd dans /etc/systemd/system/prometheus.service avec le contenu suivant :

```
[Unit]
Description=Prometheus
After=network.target

[Service]
User=prometheus
Group=prometheus
Type=simple
ExecStart=/usr/local/bin/prometheus \
  --config.file /etc/prometheus/prometheus.yml \
```

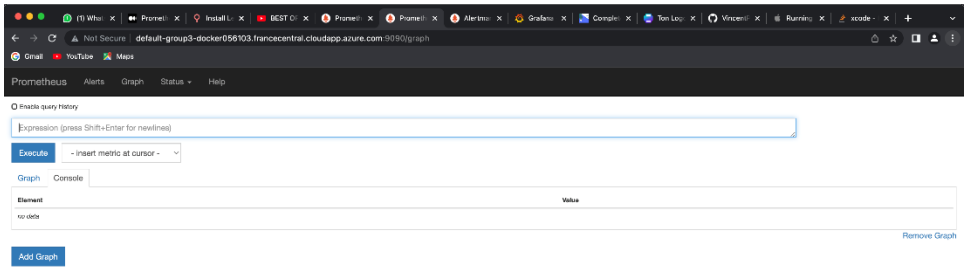
```
  --storage.tsdb.path /var/lib/prometheus/ \
  --web.console.templates=/etc/prometheus/consoles \
  --web.console.libraries=/etc/prometheus/console_libraries

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

Enfin, nous redémarrons le système :

```
sudo systemctl daemon-reload
sudo systemctl enable prometheus
sudo systemctl start prometheus
```

Prometheus fournit une interface utilisateur Web pour exécuter des requêtes de base situées à l'adresse http://default-group3-docker056103.francecentral.cloudapp.azure.com:9090/. Voici à quoi cela ressemble dans un navigateur Web :



✓ Configurer Grafana pour Prometheus

Tout d'abord, installez Grafana sur notre instance qui interroge notre serveur Prometheus.

wget https://s3-us-west-2.amazonaws.com/grafana-releases/release/grafana_5.0.4_amd64.deb

```
sudo apt-get install -y adduser libfontconfig
```

```
sudo dpkg -i grafana_5.0.4_amd64.deb
```

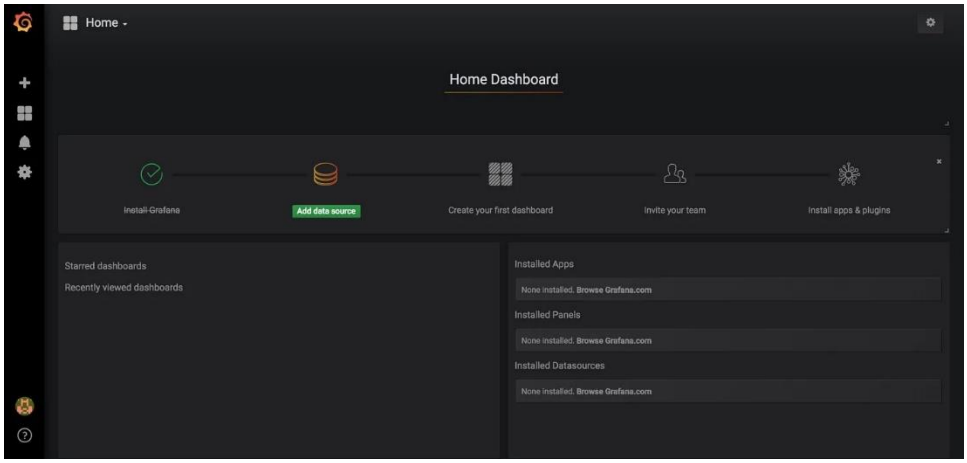
Ensuite, activez le démarrage automatique de Grafana par systemd :

sudo systemctl daemon-reload && sudo systemctl enable grafana-server && sudo systemctl start grafana-server.service

Installation avec docker:

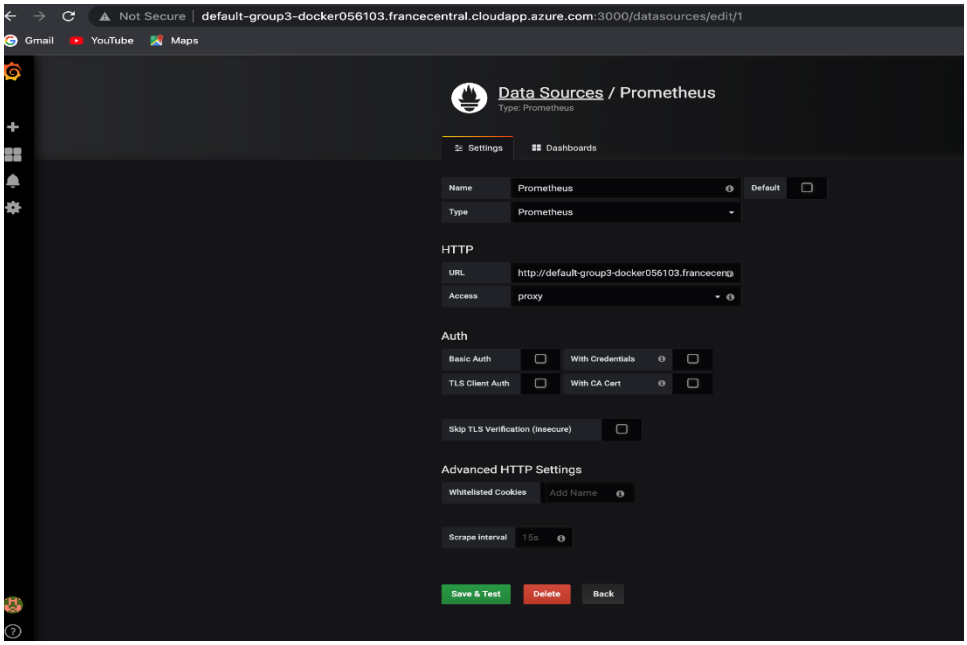
sudo docker run -d --name=grafana -p 3001:3000 grafana/grafana-enterprise:9.4.7-ubuntu

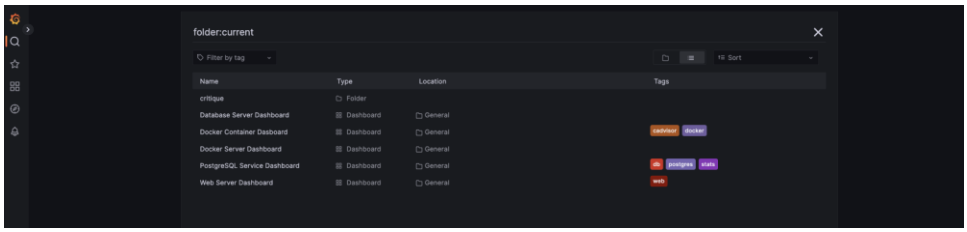
Grafana est en cours d'exécution et nous pouvons nous y connecter à l'adresse http://20.19.173.248:3001. L'utilisateur et le mot de passe par défaut sont admin / {Nsa800}.



Vous devez maintenant créer une source de données Prometheus :

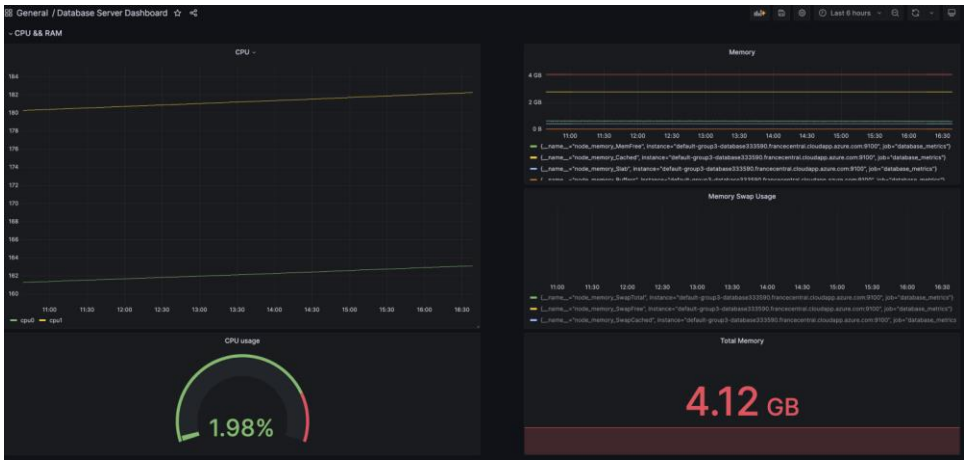
1- Cliquez sur le logo Grafana pour ouvrir la barre latérale.
2- Cliquez sur "Sources de données" dans la barre latérale.
3- Choisissez "Ajouter un nouveau".
4- Sélectionnez « Prometheus » comme source de données.
5- Définissez l'URL du serveur Prometheus (dans notre cas : http://default-group3-docker056103.francecentral.cloudapp.azure.com:9090)
6- Cliquez sur "Ajouter" pour tester la connexion et enregistrer la nouvelle source de données.





Interface grafana dévoilant les différentes dasboard disponile.

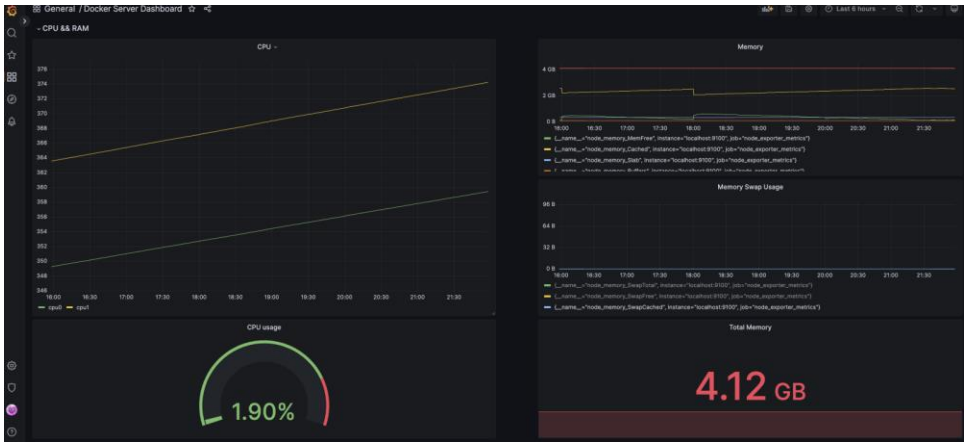
Grafana Database Server Dashboard



Graphes concernant l’usage du cpu et de la memoire

Graphes concernant l'usage du file système, de la base de données et du Traffic reseau.

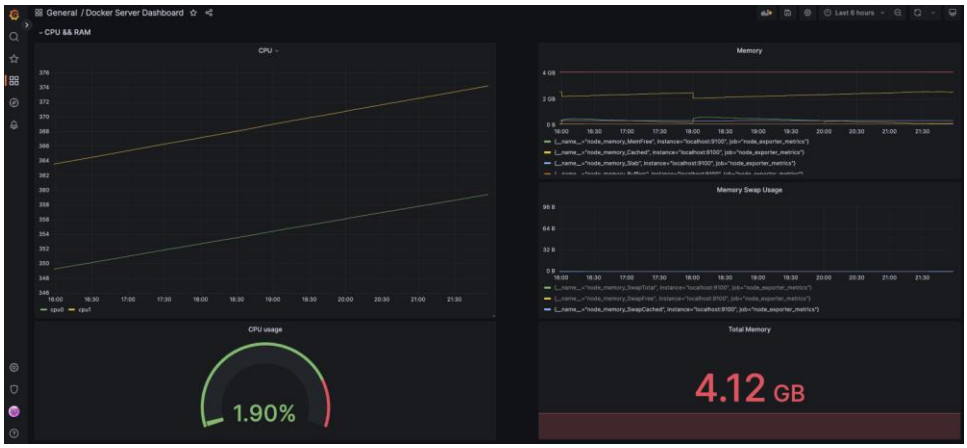
Grafana Docker Server Dashboard



Graphes concernant l'usage du cpu et de la memoire

Graphes concernant l'usage du file système et du Traffic reseau.

Grafana Web Server Dashboard

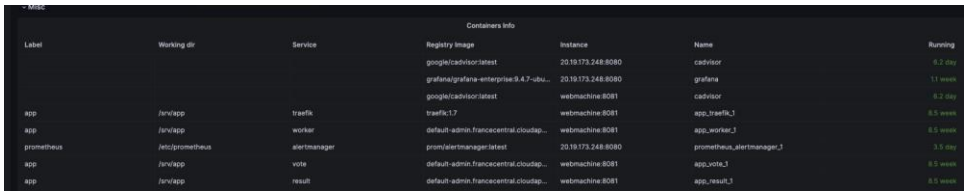


Graphes concernant l'usage du cpu et de la memoire

Graphes concernant l'usage du file système et du traffic reseau

Grafana Containers Dashboard

Graphes concernant l'usage du file cpu, de la memoire et du traffic reseau de chaque containers des différents serveurs (web et docker)



Information sur l'état de chaque containers des différents serveurs (web et docker)

Grafana Postgres Service Dashboard

General Counters, CPU, Memory and File Descriptor Stats

Graphes illustrant les statistiques de la base de données PostgreSQL

✓ Installation AlertManager

Tout d'abord, nous devons télécharger le dernier binaire d'AlertManager à partir d'ici.

```
sudo su,

cd /opt/

wget https://github.com/prometheus/alertmanager/releases/download/v0.11.0/alertmanager-0.11.0.linux-amd64.tar.gz

tar -xvzf alertmanager-0.11.0.linux-amd64.tar.gz

mv alertmanager-0.11.0.linux-amd64/alertmanager /usr/local/bin/
```

✓ Configuration d'AlertManager

L'AlertManager utilise un fichier de configuration nommé alertmanager.yml Ce fichier est contenu dans le répertoire extrait. Cependant, il n'est pas de notre utilité. C'est pourquoi nous devons créer notre propre alertmanager.yml

```
mkdir /etc/alertmanager/,

sudo nano /etc/alertmanager/alertmanager.yml
```

Mettez ensuite ce qui suit :

```
global:
  resolve_timeout: 1m
route:
  receiver: 'gmail-notifications'
receivers:
- name: 'gmail-notifications'
  email_configs:
  - to: vsakouvogui@gmail.com
    from: felicie1933@gmail.com
    smarthost: smtp.gmail.com:587
    auth_username: felicie1933@gmail.com
    auth_identity: felicie1933@gmail.com
    auth_password: vgqiyqlveauognbc
    send_resolved: true
```

Enfin, nous créons le service systemd AlertManager :

```
nano /etc/systemd/system/alertmanager.service

[Unit]

Description=AlertManager Server Service

Wants=network-online.target

After=network-online.target

[Service]

User=root

Group=root

Type=simple

ExecStart=/usr/local/bin/alertmanager --config.file /etc/alertmanager/alertmanager.yml -web.external-url=http://20.19.173.248:3000

[Install]

WantedBy=multi-user.target
```

L'utilisation de -web.external-url=http:// http://default-group3-docker056103.francecentral.cloudapp.azure.com:9093 permet de rediriger l'URL de notification vers l'interface Web prometheus AlertManager. 20.19.173.248 correspond à l'ip publique du serveur prometheus.

Rechargez ensuite le démon et lancez le service alertmanager :

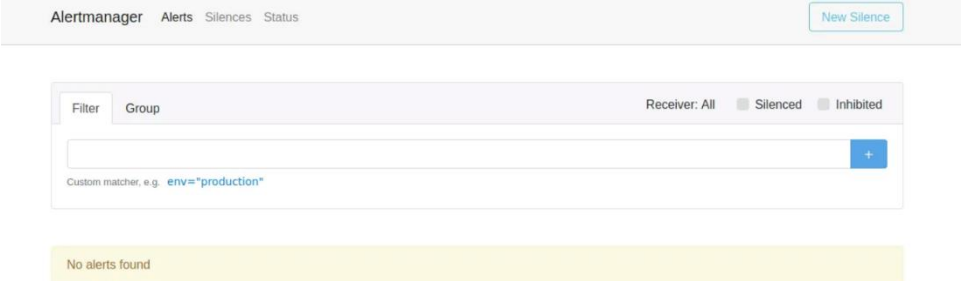
```
systemctl daemon-reload,

systemctl start alertmanager

systemctl enable alertmanager

systemctl status alertmanager
```

Vous pouvez maintenant vérifier : 20.19.173.248:9093 et vous devriez obtenir ceci :



Maintenant, nous devons configurer le serveur Prometheus pour qu'il puisse communiquer avec le service AlertManager. Nous allons mettre en place un fichier de règles d'alerte qui définit toutes les règles nécessaires pour déclencher une alerte.

Dans le fichier /etc/prometheus/prometheus.yml, ajoutez ce qui suit

```
rule_files:,
- alert.rules.yml
alerting:
        alertmanagers:
        - static_configs:
        - targets:
        - 'localhost:9093'
```

Ce qui nous amène à ce dernier fichier **etc/prometheus/prometheus.yml** :

```
global:
 scrape_interval: 10s


rule_files:
 - alert.rules.yml
alerting:
 alertmanagers:
 - static_configs:
   - targets: ['localhost:9093','20.19.173.248:9094']
```

```
scrape_configs:
  - job_name: 'prometheus_metrics'
    scrape_interval: 5s
    static_configs:
      - targets: ['20.19.173.248:9090','20.19.173.248:8080']
  - job_name: 'webmachine_metrics'
    scrape_interval: 5s
    static_configs:
      - targets: ['webmachine:9100','webmachine:8081']
  - job_name: 'database_metrics'
    scrape_interval: 5s
    static_configs:
          - targets: ['databasemachine:9100','default-group3-
database333590.francecentral.cloudapp.azure.com:9100','default-group3-
database333590.francecentral.cloudapp.azure.com:9187']
  - job_name: 'node_exporter_metrics'
    scrape_interval: 5s
    static_configs:
          - targets: ['localhost:9100']
```

Le serveur Prometheus va suivre les données de séries chronologiques entrantes, une fois que l'une des règles définies dans etc/prometheus/alert.rules.yml est satisfaite, une alerte est déclenchée vers le service AlertManager qui notifie le client sur Slack.

```
nano /etc/prometheus/alert.rules.yml
groups:
- name: alert.rules
```

```
  rules:
  - alert: InstanceDown
    expr: up == 0
    for: 1m
    labels:
      severity: "critical"
    annotations:
      summary: "Endpoint {{ $labels.instance }} down"
      description: "{{ $labels.instance }} of job {{ $labels.job }} has been down for more than 1
minutes."
  - alert: Trop_De_Load
    expr: node_load1 >= 0.6
    for: 10s
    labels:
      severity: critical
    annotations:
      summary: "{{ $labels.instance }} trop de load"
      description: "{{ $labels.instance }} of job {{ $labels.job }} fatigue le serveur."
  - alert: Space_Moitie_Plein
    expr: sum(node_filesystem_avail) >= sum(node_filesystem_size)*0.8
    for: 60m
    labels:
      severity: warning
    annotations:
      summary: "{{ $labels.instance }} A depasse la Moitie"
      description: "{{ $labels.instance }} of job {{ $labels.job }} utilise 80% de l'espace."
  - alert: Space_Presque_Plein
```

```
    expr: sum(node_filesystem_avail) >= sum(node_filesystem_size)*0.9
    for: 10m
    labels:
      severity: critique
    annotations:
      summary: "{{ $labels.instance }} A Moitie Plein"
      description: "{{ $labels.instance }} of job {{ $labels.job }} utilise 50% de l'espace."
  - alert: HostOutOfMemory
    expr: node_memory_MemAvailable / node_memory_MemTotal * 100 < 25
    for: 5m
    labels:
      severity: warning
    annotations:
      summary: "Host out of memory (instance {{ $labels.instance }})"
      description: "Node memory is filling up (< 25% left)\n  VALUE = {{ $value }}\n  LABELS: {{ $labels
}}"

  - alert: HostOutOfDiskSpace
    expr: (node_filesystem_avail{mountpoint="/"}  * 100) / node_filesystem_size{mountpoint="/"} <
50
    for: 1s
    labels:
      severity: warning
    annotations:
      summary: "Host out of disk space (instance {{ $labels.instance }})"
      description: "Disk is almost full (< 50% left)\n  VALUE = {{ $value }}\n  LABELS: {{ $labels }}"
  - alert: HostHighCpuLoad
    expr: (sum by (instance) (irate(node_cpu{job="node_exporter_metrics",mode="idle"}[5m]))) > 80
```

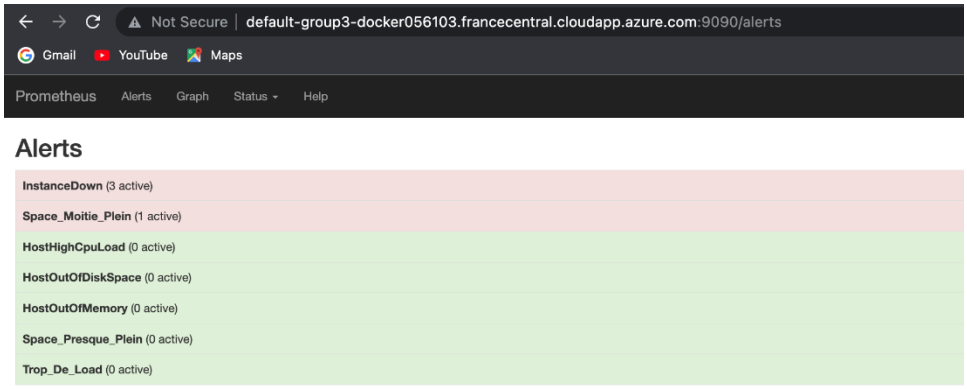
```
    for: 5m
    labels:
     severity: warning
    annotations:
     summary: "Host high CPU load (instance {{ $labels.instance }})"
     description: "CPU load is > 80%\n  VALUE = {{ $value }}\n  LABELS: {{ $labels }}"
- alert: Vote service Stopped
    expr: (time() - container_start_time_seconds{instance="webmachine:8081",
container_label_com_docker_compose_service="vote"})/86400 <0.1
    for: 1m
    labels:
     severity: critical
    annotations:
     summary: "Le container Vote s'est arrêté"
     description: "Redémarre le container pour reprendre service"
  - alert: Result service Stopped
    expr: (time() - container_start_time_seconds{instance="webmachine:8081",
container_label_com_docker_compose_service="result"})/86400 <0.1
    for: 1m
    labels:
     severity: critical
    annotations:
     summary: "Le container Result s'est arrêté"
     description: "Redémarre le container pour reprendre service"
```

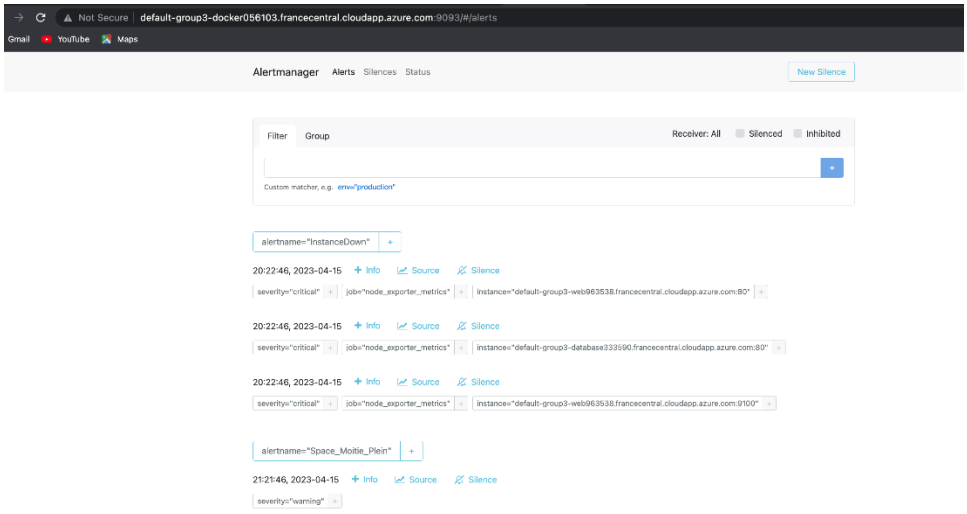
```
  - alert: Worker service Stopped
    expr: (time() - container_start_time_seconds{instance="webmachine:8081", container_label_com_docker_compose_service="worker"})/86400 <0.1
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "Le container Worker s’est arrêté"
      description: "Redémarre le container pour reprendre service"
  - alert: Traefik service Stopped
    expr: (time() - container_start_time_seconds{instance="webmachine:8081", container_label_com_docker_compose_service="traefik"})/86400 <0.1
    for: 1m
    labels:
      severity: critical
    annotations:
      summary: "Le container Traefik s’est arrêté"
      description: "Redémarre le container pour reprendre service"
```

Restart services:

```
sudo systemctl stop node_exporter &&,
sudo systemctl start node_exporter &&
sudo systemctl stop prometheus &&
sudo systemctl start prometheus &&
sudo systemctl stop alertmanager &&
```

sudo systemctl start alertmanager





Configuration du Backup de la DB

I- Postgres

Nous avons utilisé **pgbackrest** pour la mise en place du Backup.

- Installation
  1- Être en Sudo

2- sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'
3- wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | apt-key add -
4- apt update
5- apt install -y pgbackrest

- Mise en place du backup

Pour la première fois, on fait le full backup: sudo -u postgres pgbackrest --stanza=db-primary --type=full backup

On automatise ensuite les autres backups dans cron (crontab –u postgres –e) :

On fait respectivement une sauvegarde incrémentielle chaque Mercredi à minuit et une autre différentielle, chaque Samedi à minuit. On envoie ensuite ce backup avec une nomenclature “db-primary_pgsql_{DATE}.zip” vers le serveur docker.

```
# m h  dom mon dow   command
0 0 * * 3 /usr/bin/pgbackrest --stanza=db-primary --type=incr backup
0 0 * * 6 /usr/bin/pgbackrest --stanza=db-primary --type=diff backup
30 0 * * 6 /usr/bin/python3 /home/admusr/key/backup.py
```

Script python pour le remote backup

```python
import paramiko
import os
import datetime

# Paramètres de la connexion SSH pour le serveur de destination
destination_hostname = 'default-group3-docker056103.francecentral.cloudapp.azure.com'
destination_username = 'admusr'
destination_key_filename = '/home/admusr/key/group-3_rsa'  # chemin vers le fichier id_rsa destination
destination_key_password = 'BU6MQrFYwop/BmlYlBW+nQ'  # passphrase du fichier id_rsa destination (si nécessaire)

# Paramètres du transfert de fichiers
remote_path = '/tmp/'  # répertoire distant où les fichiers seront transférés
local_path = '/var/lib/pgbackrest/backup/db-primary'  # répertoire local contenant les fichiers à transférer

#céer un client SSH pour le serveur de destination
destination_client = paramiko.SSHClient()
destination_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
destination_client.connect(destination_hostname, username=destination_username, key_filename=destination_key_filename, password=destination_key_password)


#local_dir = '/var/lib/pgbackrest/backup/db-primary'

remote_dir = '/tmp/'

x = datetime.datetime.now()
DATE = x.strftime("%d-%m-%y")

zip_path = f'/var/lib/pgbackrest/backup/db-primary_pgsql_{DATE}.zip'

os.system(f'cd {local_path} && zip -r {zip_path} *')

# Transférer les fichiers du serveur source vers le serveur de destination
sftp = destination_client.open_sftp()

sftp.put(zip_path, remote_path + f'db-primary_pgsql_{DATE}.zip')
sftp.close()
print('Le fichier a été transféré avec succès.')

# Fermer les connexions SSH
#source_client.close()
destination_client.close()
```

Checking du backup

```
admusr@default-Group3-Database:~$ sudo -u postgres pgbackrest --stanza=db-primary info
stanza: db-primary
    status: ok
    cipher: none

    db (current)
        wal archive min/max (10): 00000001000000000000000B/00000001000000000000000E

        full backup: 20230418-202040F
            timestamp start/stop: 2023-04-18 20:20:40 / 2023-04-18 20:20:46
            wal start/stop: 00000001000000000000000B / 00000001000000000000000B
            database size: 22.5MB, database backup size: 22.5MB
            repo1: backup set size: 2.6MB, backup size: 2.6MB

        incr backup: 20230418-202040F_20230418-202218I
            timestamp start/stop: 2023-04-18 20:22:18 / 2023-04-18 20:22:20
            wal start/stop: 00000001000000000000000D / 00000001000000000000000D
            database size: 22.5MB, database backup size: 8.2KB
            repo1: backup set size: 2.6MB, backup size: 424B
            backup reference list: 20230418-202040F

        diff backup: 20230418-202040F_20230418-202241D
            timestamp start/stop: 2023-04-18 20:22:41 / 2023-04-18 20:22:42
            wal start/stop: 00000001000000000000000F / 00000001000000000000000F
            database size: 22.5MB, database backup size: 8.2KB
            repo1: backup set size: 2.6MB, backup size: 423B
            backup reference list: 20230418-202040F

        incr backup: 20230418-202040F_20230419-000001I
            timestamp start/stop: 2023-04-19 00:00:01 / 2023-04-19 00:00:03
            wal start/stop: 00000001000000000000011 / 00000001000000000000011
            database size: 22.5MB, database backup size: 8.2KB
            repo1: backup set size: 2.6MB, backup size: 428B
            backup reference list: 20230418-202040F, 20230418-202040F_20230418-202241D

        diff backup: 20230418-202040F_20230422-000002D
            timestamp start/stop: 2023-04-22 00:00:02 / 2023-04-22 00:00:03
            wal start/stop: 00000001000000000000013 / 00000001000000000000013
            database size: 22.5MB, database backup size: 8.2KB
            repo1: backup set size: 2.6MB, backup size: 429B
            backup reference list: 20230418-202040F
```

II- Redis

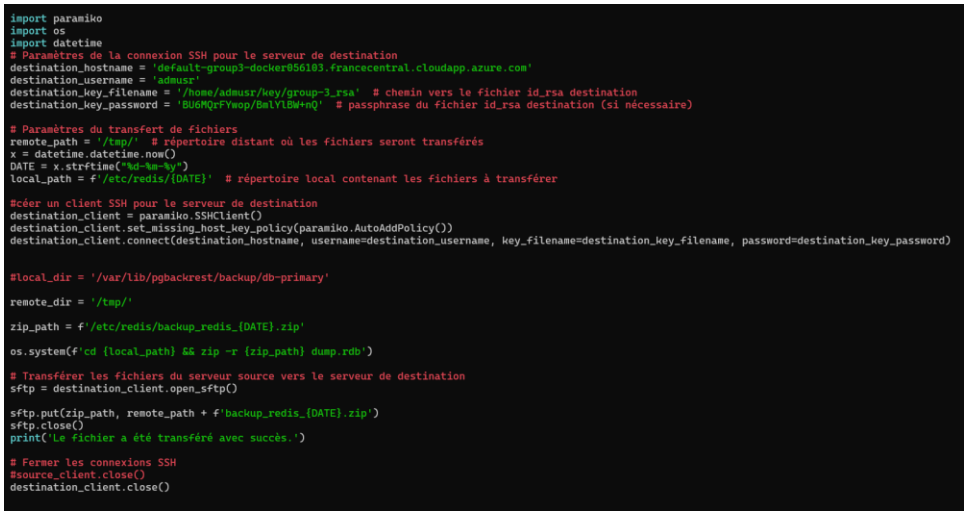
Nous devons d'abord faire un dump de la base de données de redis. Pour celà, nous écrivons ce script bash.

```
#!/bin/bash
rdb_file="/var/lib/redis/dump.rdb"
redis_cli="/usr/bin/redis-cli"

DIR=`date +%d-%m-%y`
DEST=/etc/redis/$DIR
sudo mkdir $DEST

echo save| $redis_cli
sudo cp $rdb_file $DEST
exit 1
```

Nous faisons ensuite le remote backup vers la machine docker.



```
import paramiko
import os
import datetime
# Paramètres de la connexion SSH pour le serveur de destination
destination_hostname = 'default-group3-docker056103.francecentral.cloudapp.azure.com'
destination_username = 'admusr'
destination_key_filename = '/home/admusr/key/group-3_rsa'  # chemin vers le fichier id_rsa destination
destination_key_password = 'BU6MQrFYwop/BmlYlBW+nQ'  # passphrase du fichier id_rsa destination (si nécessaire)

# Paramètres du transfert de fichiers
remote_path = '/tmp/'  # répertoire distant où les fichiers seront transférés
x = datetime.datetime.now()
DATE = x.strftime("%d-%m-%y")
local_path = f'/etc/redis/{DATE}'  # répertoire local contenant les fichiers à transférer

#céer un client SSH pour le serveur de destination
destination_client = paramiko.SSHClient()
destination_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
destination_client.connect(destination_hostname, username=destination_username, key_filename=destination_key_filename, password=destination_key_password)


#local_dir = '/var/lib/pgbackrest/backup/db-primary'

remote_dir = '/tmp/'

zip_path = f'/etc/redis/backup_redis_{DATE}.zip'

os.system(f'cd {local_path} && zip -r {zip_path} dump.rdb')

# Transférer les fichiers du serveur source vers le serveur de destination
sftp = destination_client.open_sftp()

sftp.put(zip_path, remote_path + f'backup_redis_{DATE}.zip')
sftp.close()
print('Le fichier a été transféré avec succès.')

# Fermer les connexions SSH
#source_client.close()
destination_client.close()
```

Une automatisation avec cron est ensuite faite (crontab –u redis –e). Cette sauvegarde est éffectuée tous les jours à 00h00min (minuit).

```
# m h  dom mon dow   command
0 0 * * * /home/admusr/key/redis_backup.sh
0 0 * * * /usr/bin/python3 /home/admusr/key/backup_redis.py
```

Checking des sauvegardes sur la machine docker

```
admusr@default-Group3-Docker:/tmp$ ls
_MEIMAGD1K                     systemd-private-ae8a4caa854a4d3892509aca9176a3ff-systemd-resolved.service-S
_MEIaBoJfT                     systemd-private-ae8a4caa854a4d3892509aca9176a3ff-systemd-timesyncd.service-
backup_redis_29-04-23.zip      tmp.L3g3wyPJnn
db-primary_pgsql_29-04-23.zip  ubuntu-advantage
snap-private-tmp
admusr@default-Group3-Docker:/tmp$
```

Références :

Backup pgsql : https://www.scaleway.com/en/docs/tutorials/backup-postgresql-pgbackrest-s3/

Backup redis : https://simplebackups.com/blog/the-complete-redis-backup-guide-with-examples/

https://medium.com/devops-dudes/install-prometheus-on-ubuntu-18-04-a51602c6256b

https://medium.com/devops-dudes/prometheus-alerting-with-alertmanager-e1bbba8e6a8e