



CLOUD T-CLO-901

ANSIBLE MONITORING & RESILIENCE

Documentation du projet

Groupe Cloud

- Vincent Pokpa Sakouvogui
- Nassym Alassane
- Damia Tobossi
- Serge Gnansounou

Table des Matières

Contexte	4
Mission.....	4
Objectifs.....	4
Méthodologie	4

Matrice des Compétences	5
Elaboration Des Sprints.....	5
Etude Architecture.....	6
✓ Etude et apprehension de l'architecture du système entier	6
✓ Etude et apprehension de Prometheus	8
✓ Etude et apprehension de Node Exporter	8
✓ Etude et apprehension de Traefix.....	10
✓ Etude et apprehension de Alert Manager	11
✓ Etude et apprehension de Grafana	12
Connection aux Machines.....	13
Installation des services	13
✓ Installation de node_exporter	13
✓ Installation de Prometheus.....	14
✓ Configuration de Prometheus.....	15
✓ Configurer Grafana pour Prometheus	18
Grafana Database Server Dashboard.....	20
Grafana Docker Server Dashboard	21
Grafana Web Server Dashboard	22
Grafana Containers Dashboard.....	22
Grafana Postgres Service Dashboard.....	23
✓ Installation AlertManager	23
✓ Configuration d'AlertManager	24
Configuration du Backup de la DB	32
Ansible File – Inventory.ini.....	39
Ansible File –playbook Deploy Prometheus, Grafana, and Node Exporter	39
Ansible File – node_exporter.service.....	46
Ansible File – prometheus.service	46
Ansible File – prometheus.yml.....	47
Ansible File – Grafana.service	48
Ansible File – Backup.py.yml.....	49
Backup des conteneurs : https://www.geeksforgeeks.org/backing-up-a-docker-container/	52

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

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19	SKILL MATRIX							
	Vincent Sakouvogui		Alasane Nassym		Serge Gnansounou		Damia Tobossi	
	Capabilities	Proficiency Interest	Proficiency Interest	Proficiency Interest	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

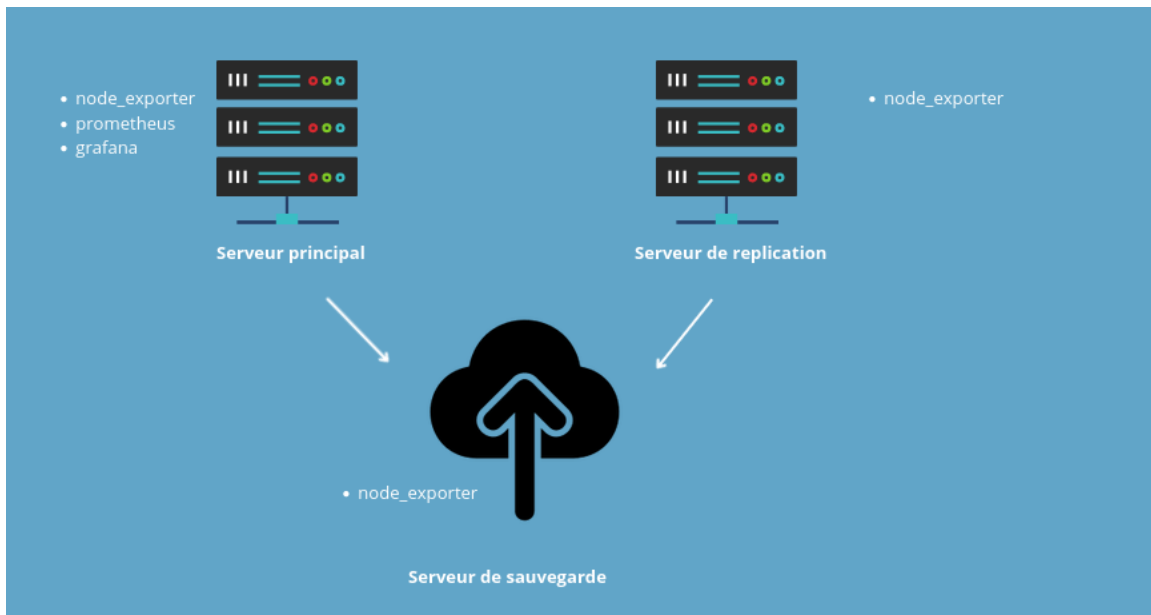
1 = Is interested in applying this capability

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 Ansible	Etude Architecture	½ H

5	Etudier et comprendre Alertmanager	Etude Architecture	½ H
6	Etudier et comprendre LogRotate	Etude Architecture	½ H
7	Installer 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 métriques nécessaires	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

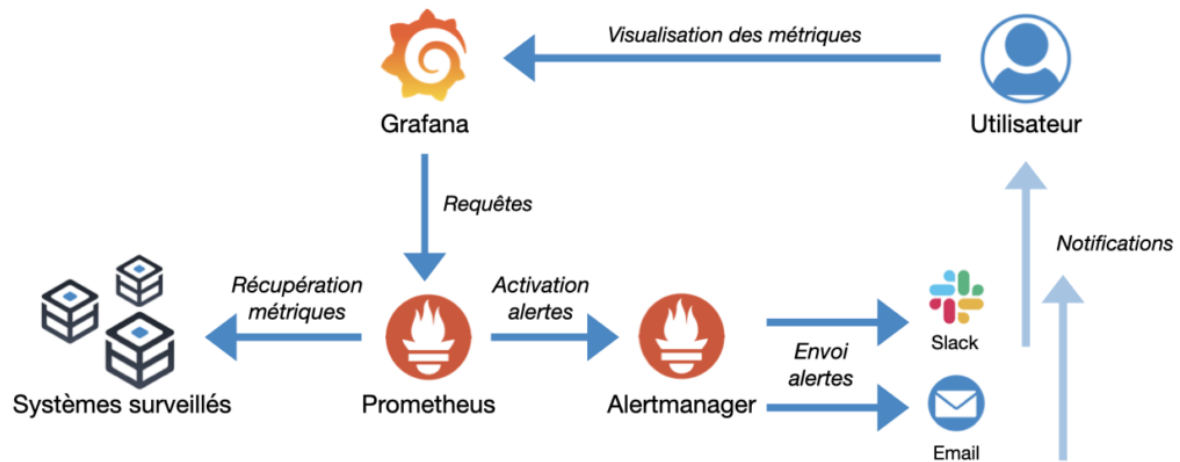


La première et deuxième machine contient la base de données entière avec les services Node exporter, l'application et tout. Les deux services fonctionnent sur cette même machine, ce qui permet de stocker et de gérer les données de manière efficace. La base de données peut être utilisée par l'application pour stocker et récupérer les données nécessaires à son fonctionnement.

La troisième machine ne contient grafana, prometheus et le backup. 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.

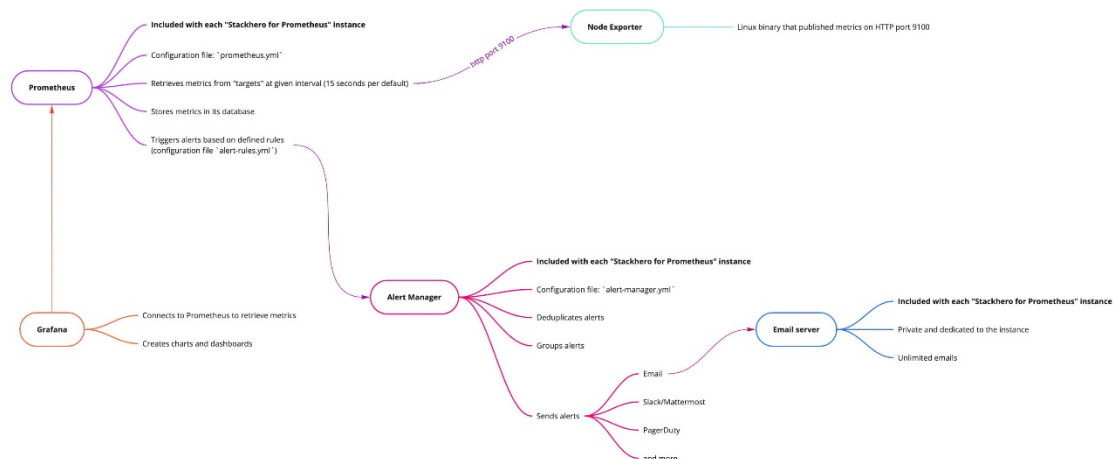
En résumé, cette architecture utilise trois machines distinctes pour séparer les différentes parties de l'application et les différents services, afin de les rendre plus modulaires, évolutifs et résilients. La première machine contient la base de données, la deuxième machine contient l'application et 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



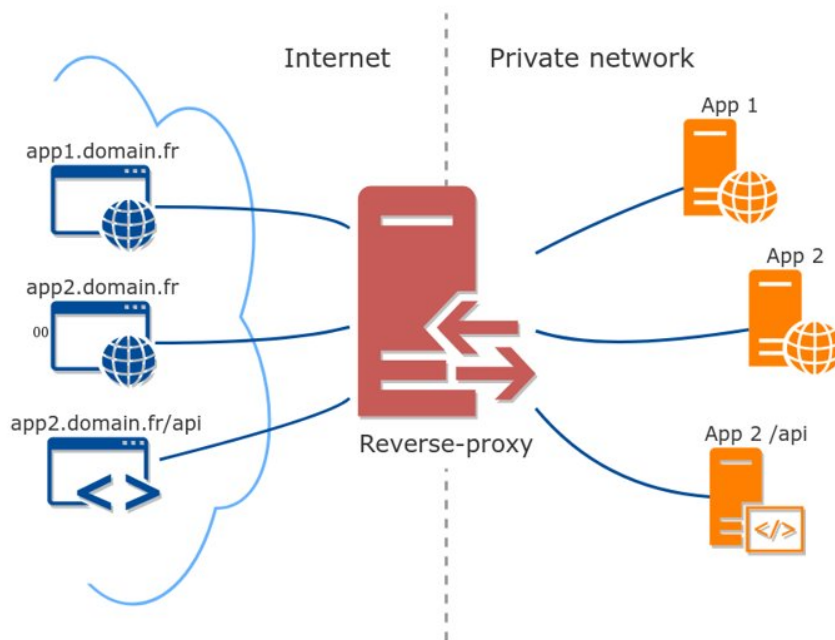
Le service Prometheus est un outil puissant et flexible pour la surveillance en temps réel des applications cloud natives. Il est facile à installer et à utiliser, avec une grande communauté d'utilisateurs et de développeurs qui offrent de nombreux plugins et intégrations pour étendre ses fonctionnalités.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

✓ Etude et apprehension de Node Exporter



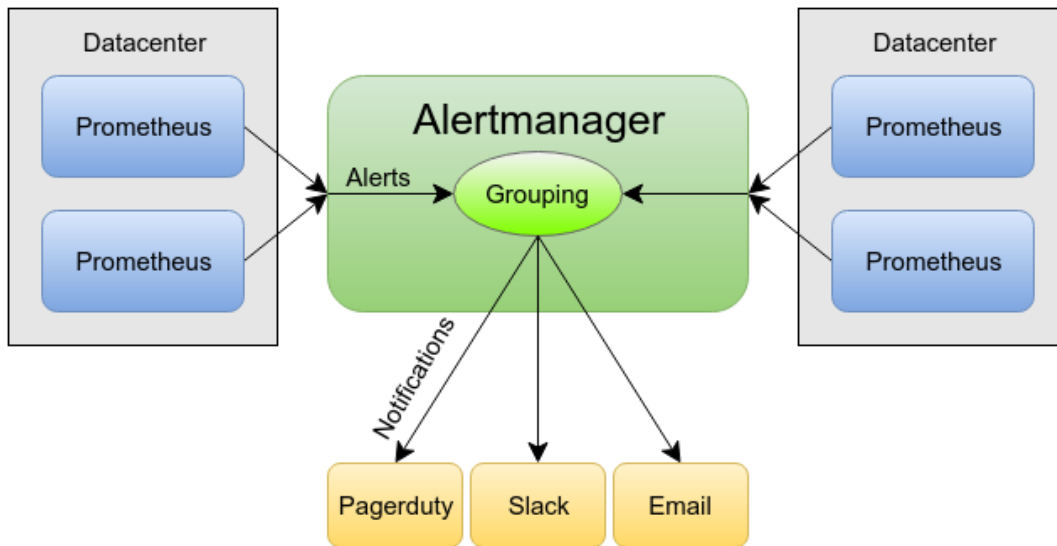
Node Exporter est un outil de collecte de métriques système flexible et facile à utiliser, qui peut être utilisé pour surveiller les performances des nœuds individuels dans un environnement de cluster. Avec sa compatibilité multiplateforme et sa haute configurabilité, il est devenu un outil populaire 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.

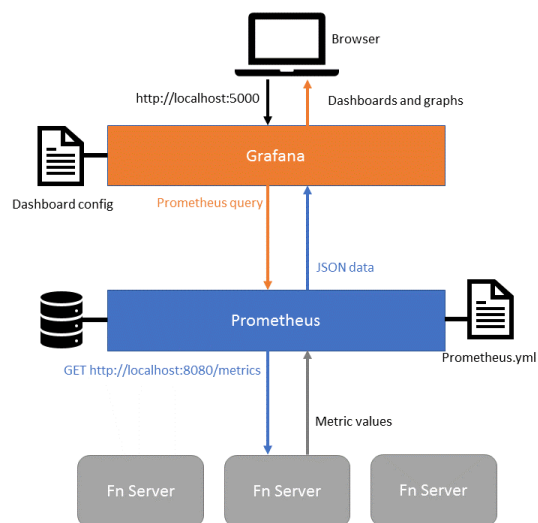
Alertmanager est hautement configurable et peut être utilisé avec différents systèmes de surveillance tels que Prometheus, Graphite et InfluxDB. Il est facile à installer et à utiliser, ce qui permet une configuration rapide et efficace.

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

```
ssh epitech@vm1335250.westeurope.cloudapp.azure.com -i vpscloud
```

```
ssh epitech@vmbackup.westeurope.cloudapp.azure.com -i vpscloud
```

```
ssh epitech@slave.westeurope.cloudapp.azure.com -i vpscloud
```

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 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: ['server1IP:9090']

- job_name: 'node_exporter_metrics'

  scrape_interval: 5s

  static_configs:

  - targets: ['localhost:9100', '{{ node_exporter_server1_ip }}:9100', '{{ node_exporter_server3_ip }}:9100', , '{{ prometheus_server2_ip }}:9100']
```

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
```

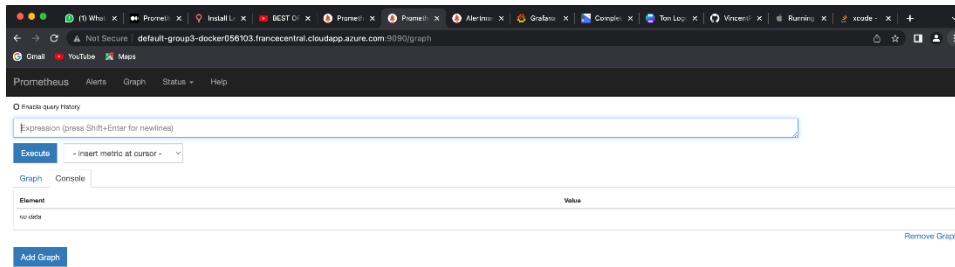
Ensuite, créons un fichier d'unité systemd dans
/etc/systemd/system/prometheus.service avec le contenu suivant :

```
[Unit]  
Description=Prometheus  
After=network.target  
  
[Service]  
User=prometheus  
Group=prometheus  
Type=simple  
ExecStart=/usr/local/bin/prometheus \\\n  --config.file /etc/prometheus/prometheus.yml \\\n  --storage.tsdb.path /var/lib/prometheus/ \\\n  --web.console.templates=/etc/prometheus/consoles \\\n  --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/](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
```

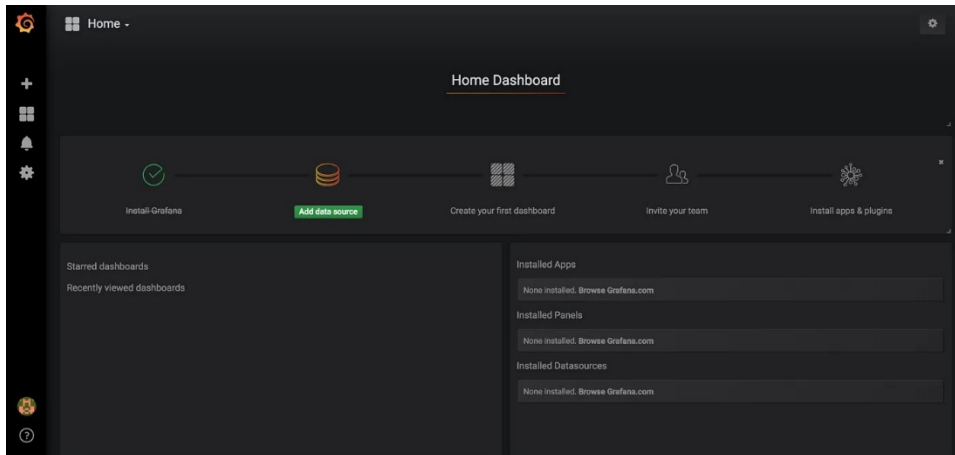
Ensuite, activez le démarrage automatique de Grafana par systemd :

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

Installation avec docker:

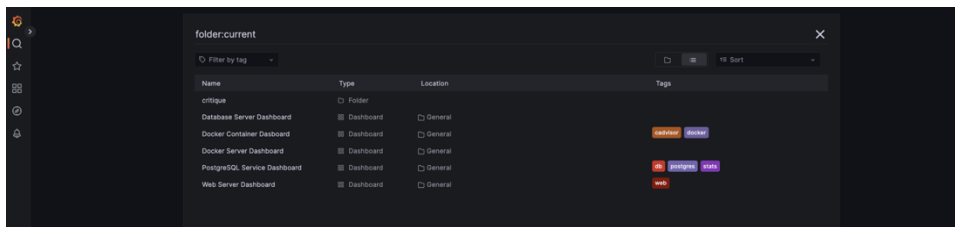
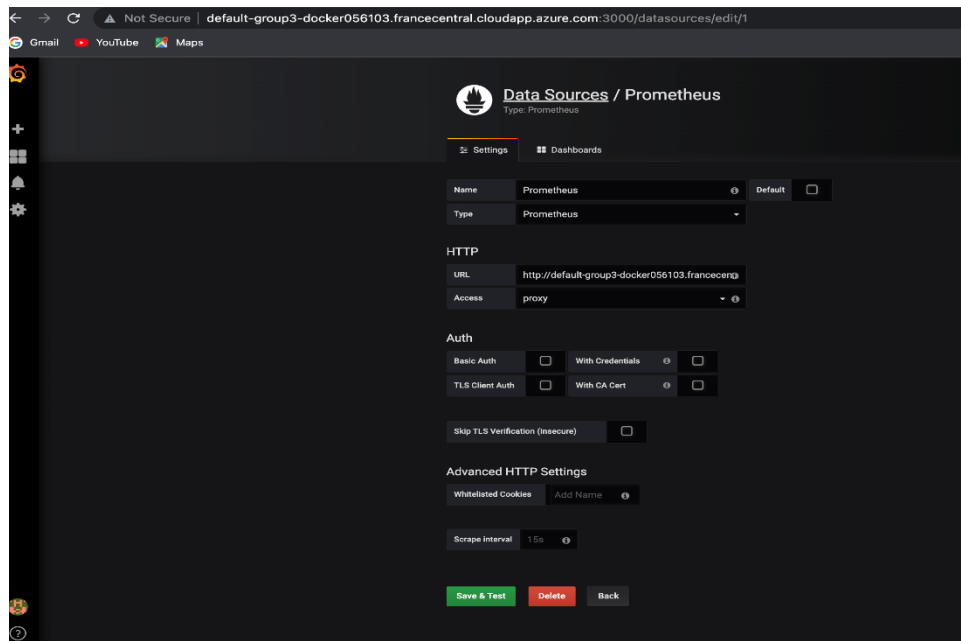
```
sudo docker run -d --name=grafana -p 3001:3000 grafana/grafana-enterprise:9.4.7-ubuntu
```

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}.



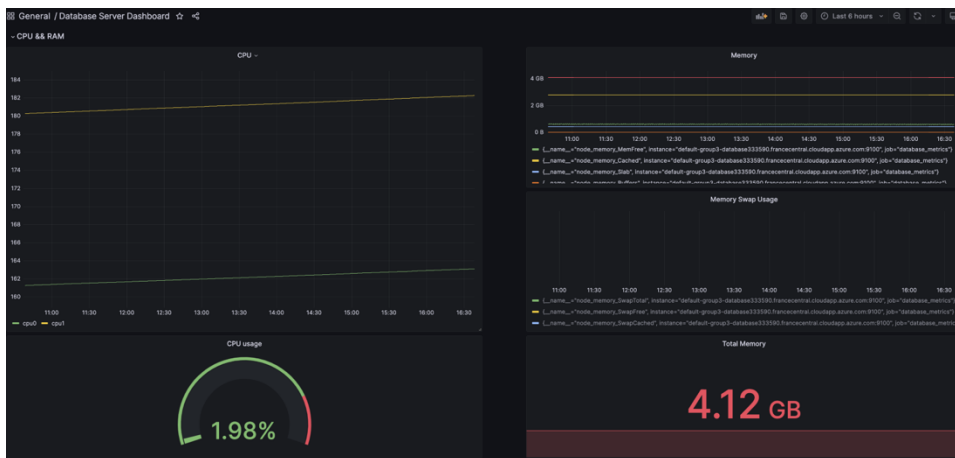
Vous devez maintenant créer une source de données Prometheus :

- 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.

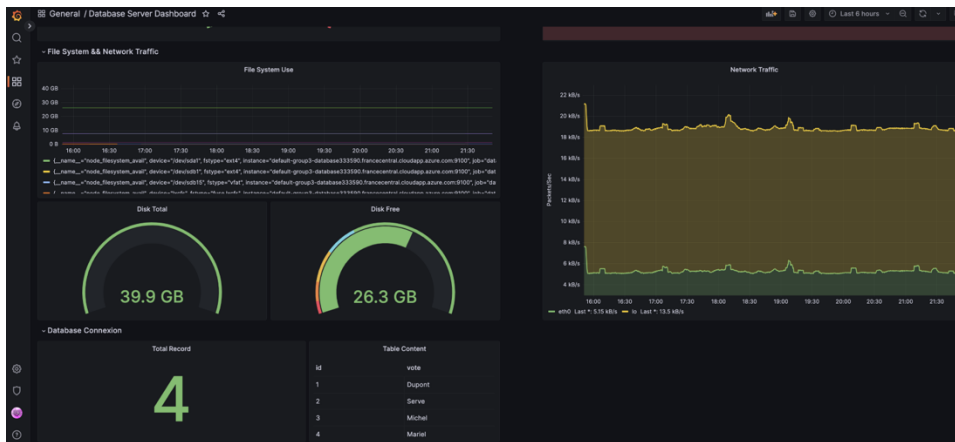


Interface grafana dévoilant les différentes dashboard disponible.

Grafana Database Server Dashboard

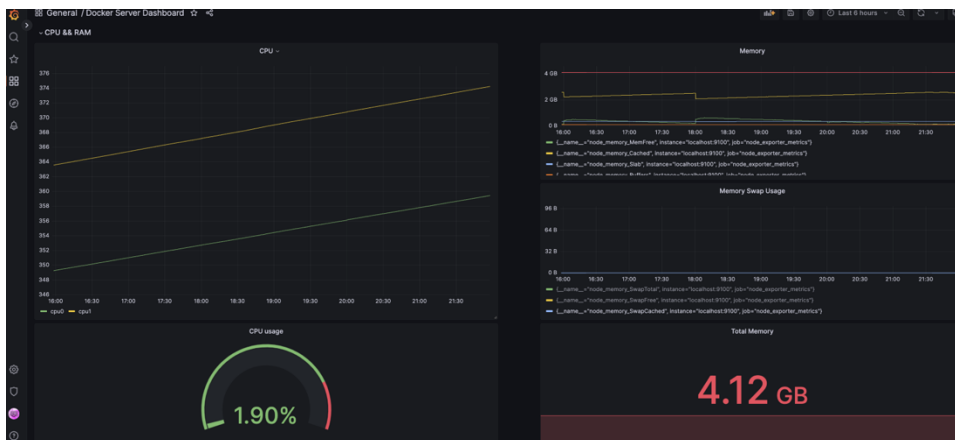


Graphes concernant l'usage du cpu et de la memoire

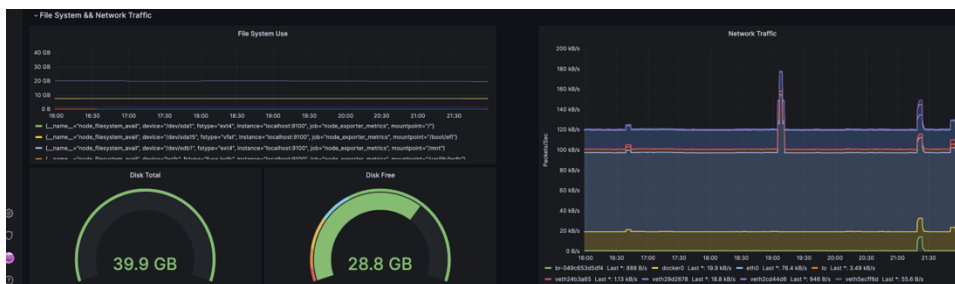


Graphes concernant l'usage du file système, de la base de données et du Traffic reseau.

Grafana Docker Server Dashboard



Graphes concernant l'usage du cpu et de la memoire

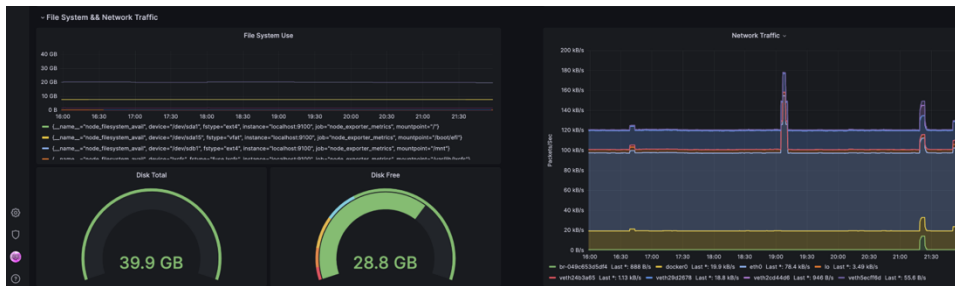


Graphes concernant l'usage du file système et du Traffic reseau.

Crafana Web Server Dashboard

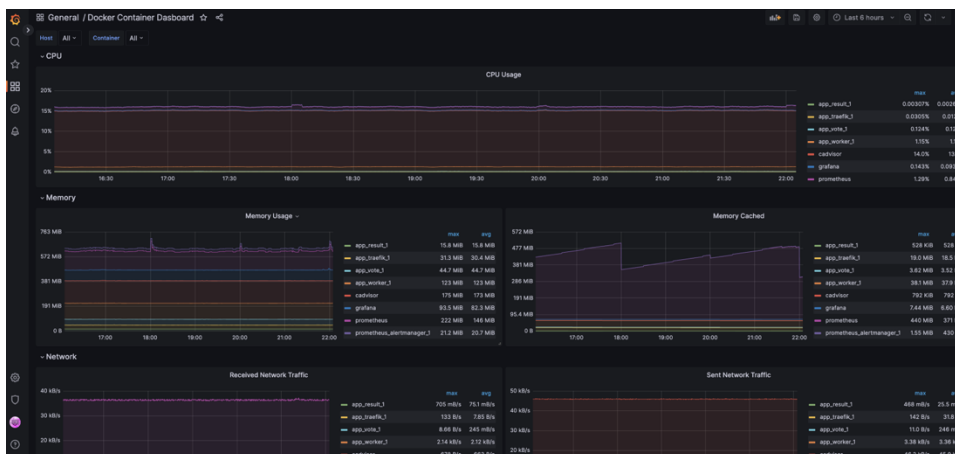


Graphes concernant l'usage du cpu et de la memoire



Graphes concernant l'usage du file système et du trafic réseau

Crafana Containers Dashboard

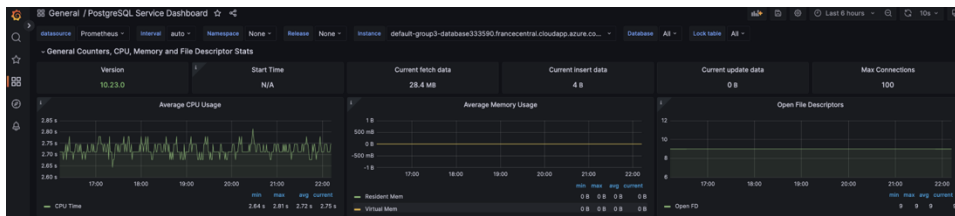


Graphes concernant l'usage du file cpu, de la memoire et du trafic réseau de chaque containers des différents serveurs (web et docker)

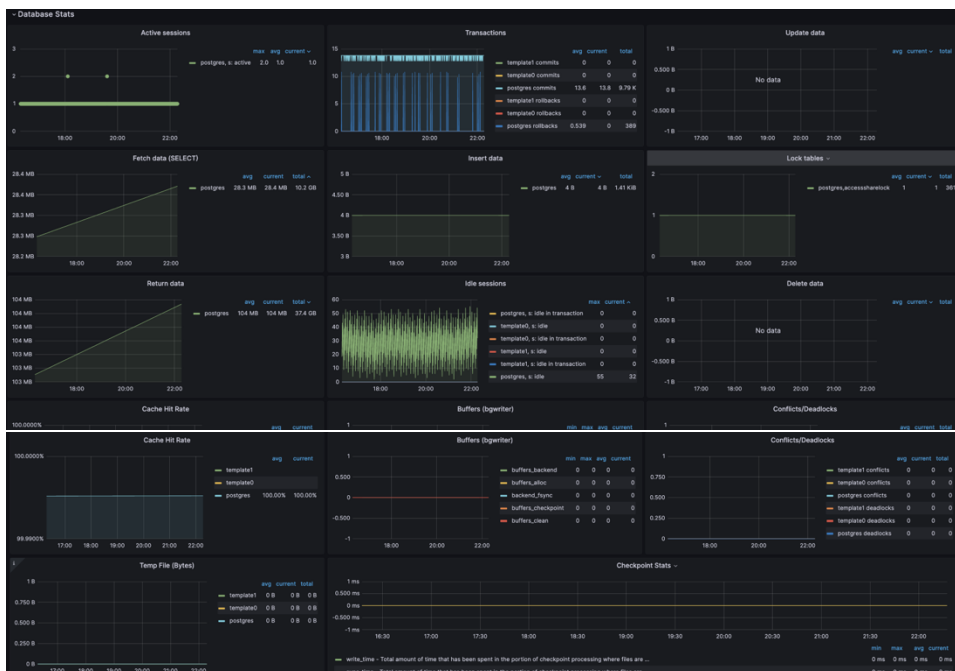
Containers Info						
Label	Working dir	Service	Registry Image	Instance	Name	Running
			google/cadvisor:latest	2016173.248.8080	cadvisor	6.2 day
			grafana/grafana-enterprise:9.4.7-ubuntu	2016173.248.8080	grafana	1.1 week
			google/cadvisor:latest	webmachine:8081	cadvisor	6.2 day
app	/usr/app	traefik	traefik:1.7	webmachine:8081	app_traefik_1	6.5 week
app	/usr/app	worker	default-admin:francecentral.cloudapp...	webmachine:8081	app_worker_1	6.5 week
prometheus	/etc/prometheus	alertmanager	prom/alertmanager:latest	2016173.248.8080	prometheus_alertmanager_1	3.3 day
app	/usr/app	note	default-admin:francecentral.cloudapp...	webmachine:8081	app_note_1	6.5 week
app	/usr/app	result	default-admin:francecentral.cloudapp...	webmachine:8081	app_result_1	6.5 week

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 &&

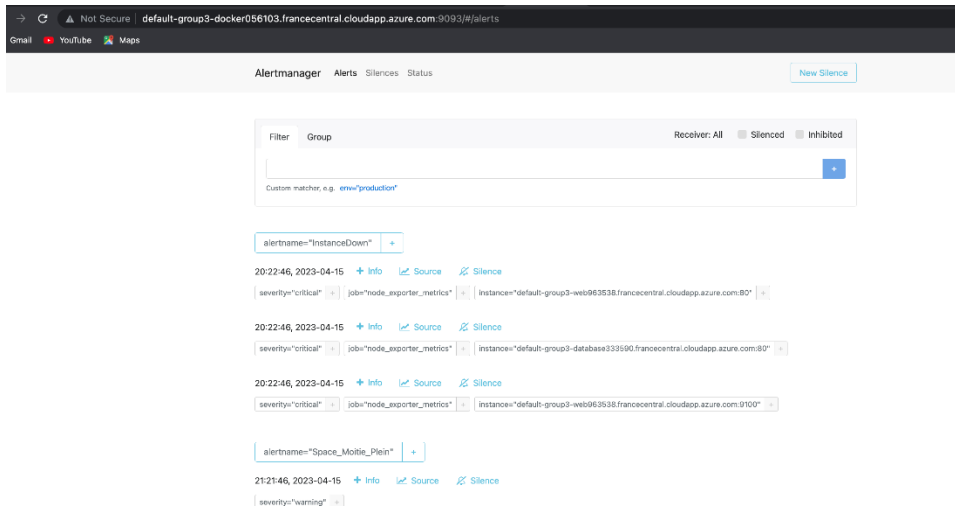
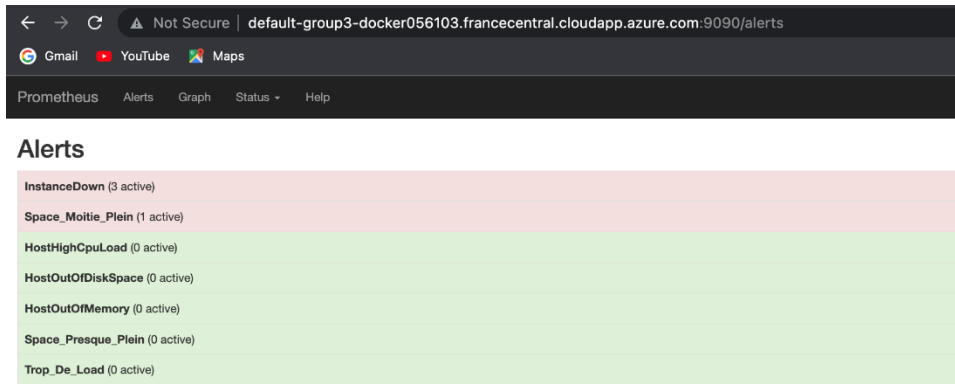
sudo systemctl start node_exporter &&

sudo systemctl stop prometheus &&

sudo systemctl start prometheus &&

sudo systemctl stop alertmanager &&

sudo systemctl start alertmanager



Configuration du Backup de la DB

I- Mysql

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

```
import paramiko
```

```
import os
```

```
import datetime
```



```
import sys

import subprocess

php_script_path = '/home/epitech/backup.php'

# Run the PHP script using subprocess

result = subprocess.run(['php', php_script_path], capture_output=True,
text=True)

# Print the output and error if any

print('Output:', result.stdout)

print('Error:', result.stderr)

# Check the return code to see if the command was successful

if result.returncode == 0:

    print('PHP script executed successfully.')

else:

    print(f'Error executing PHP script. Return code: {result.returncode}')

    sys.exit()

# Paramètres de la connexion SSH pour le serveur de destination

destination_hostname = 'vmbackup.westeurope.cloudapp.azure.com'

destination_username = 'epitech'
```

```
destination_key_filename = '/home/epitech/vpscloud' # chemin vers le fichier  
id_rsa destination
```

```
# Paramètres du transfert de fichiers
```

```
remote_path = '/tmp/' # répertoire distant où les fichiers seront transférés
```

```
local_path = '/tmp/db-backups' # répertoire local contenant les fichiers à  
transférer
```

```
# Créer 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)
```

```
# Local_dir = '/var/lib/pgbackrest/backup/db-primary'
```

```
remote_dir = '/tmp/'
```

```
x = datetime.datetime.now()
```

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

```
zip_path = f'/tmp/db-backups/db-mysql_{DATE}.sql.zip'
```

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

```
php_script_path = '/home/epitech/backup.php'
```

```
# Transférer les fichiers du serveur source vers le serveur de destination
```

```
sftp = destination_client.open_sftp()

sftp.put(zip_path, remote_path + f'db-mysql_{DATE}.sql.zip')

sftp.close()

print('Le fichier a été transféré avec succès.')
```

```
# Fermer les connexions SSH
```

```
# source_client.close()
```

```
destination_client.close()
```

- o 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 journalière à minuit. On envoie ensuite ce backup avec une nomenclature "db-mysql_{DATE}.sql.zip" vers le serveur vmbackup.

```
# 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](#)

```

import paramiko
import os
import datetime

# Paramètres de la connexion SSH pour le serveur de destination
destination_hostname = 'default-group3-docker@56103.francecentral.cloudapp.azure.com'
destination_username = 'admsur'
destination_key_filename = '/home/admsur/.ssh/group-3_rsa' # chemin vers le fichier id_rsa destination
destination_key_password = 'BUEMQrFYwop/BmLYlEw+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

# Créer 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_psql_{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_psql_{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

```

admsur@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): 000000010000000000000000B/000000010000000000000014

  full backup: 20230418-202040F
    timestamp start/stop: 2023-04-18 20:20:40 / 2023-04-18 20:20:46
    wal start/stop: 000000010000000000000000B / 00000001000000000000000B
    database size: 22.5MB, database backup size: 22.5MB
    repl1: 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: 000000010000000000000000D / 00000001000000000000000D
    database size: 22.5MB, database backup size: 8.2KB
    repl1: 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: 000000010000000000000000F / 00000001000000000000000F
    database size: 22.5MB, database backup size: 8.2KB
    repl1: 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: 0000000100000000000000011 / 000000010000000000000011
    database size: 22.5MB, database backup size: 8.2KB
    repl1: backup set size: 2.6MB, backup size: 428B
    backup reference list: 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: 0000000100000000000000013 / 000000010000000000000013
    database size: 22.5MB, database backup size: 8.2KB
    repl1: backup set size: 2.6MB, backup size: 429B
    backup reference list: 20230418-202040F

```

Nous devons d'abord faire un dump de la base de données de redis. Pour cela, 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-docker956183.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 = 'BUC6QrFwop/belVlBW+Hq' # 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

# Créer 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 effectuée tous les jours à 23h10min et est envoyé sur le serveur distant à 23h30min.

```
# m h dom mon dow command
10 23 * * * /home/admusr/key/redis_backup.sh
30 23 * * * /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$

```

III - Backup de la base de donnée du server web

Pour le backup des conteneurs nous avons procédé à une automatisation avec cron(sudo crontab -e -u root). Cette sauvegarde est effectuée tous les jours à 1h00min du matin.

```

# at 3 a.m every week with.
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 1 * * *    ~/backup.sh

```

Voici la dénomination des backups

- catvisor-backup-\$(date +%Y%m%d).sql.zip
- db-mysql_\$(date +%Y%m%d).sql.zip
- db-mysql_\$(date +%Y%m%d).sql.zip

Les backups sont sauvegardés dans le dossier */tmp/*

```

dpkg.status.1.gz
dpkg.status.0
catvisor-backup-20230501.tar
app_traefik_1-backup-20230501.tar
app_result_1-backup-20230501.tar
app_worker_1-backup-20230501.tar
app_vote_1-backup-20230501.tar

```

Ansible File – Inventory.ini

```
[servers]
```

```
server1 ansible_host=vm1335250.westeurope.cloudapp.azure.com ansible_user=epitech
```

```
server2 ansible_host=vmbackup.westeurope.cloudapp.azure.com ansible_user=epitech
```

```
server3 ansible_host=slave.westeurope.cloudapp.azure.com ansible_user=epitech
```

Ansible File –playbook Deploy Prometheus, Grafana, and Node Exporter

```
- name: Deploy Prometheus, Grafana, and Node Exporter
```

```
hosts: servers
```

```
become: true # To run tasks with sudo
```

```
vars:
```

```
prometheus_server2_ip: "vmbackup.westeurope.cloudapp.azure.com"
```

```
node_exporter_server1_ip: "vm1335250.westeurope.cloudapp.azure.com"
```

```
node_exporter_server3_ip: "slave.westeurope.cloudapp.azure.com"
```

```
tasks:
```

```
- name: Update apt cache (for Debian/Ubuntu)
```

```
apt:
```

```
update_cache: yes
```

```
when: ansible_os_family == "Debian"
```

```
- name: Install necessary dependencies
```

```
package:
```

```
name: "{{ item }}"
```

state: present

loop:

- curl

- software-properties-common # for adding repositories

- apt-transport-https # for using HTTPS with APT

- name: Download and extract Node Exporter on server1

become: true

shell: |

wget

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

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

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

rm -r node_exporter-0.15.2.linux-amd64*

sudo useradd -rs /bin/false node_exporter

sudo systemctl daemon-reload

args:

executable: /bin/bash

when: "'server1' in inventory_hostname or 'server3' in inventory_hostname"

- name: Download and extract Node Exporter on server2

become: true

shell: |

wget

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

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

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

rm -r node_exporter-0.15.2.linux-amd64*


```
sudo apt-get install gnupg -y  
sudo useradd -rs /bin/false node_exporter  
sudo systemctl daemon-reload
```

args:

executable: /bin/bash

when: "'server2' in inventory_hostname"

- name: Create systemd unit file for Node Exporter on server1

become: yes

template:

src: node_exporter.service

dest: /etc/systemd/system/node_exporter.service

notify: restart node_exporter

when: "'server1' in inventory_hostname or 'server3' in inventory_hostname"

- name: Enable and start Node Exporter service on server1

become: yes

systemd:

name: node_exporter

enabled: yes

state: started

when: "'server1' in inventory_hostname or 'server3' in inventory_hostname"

- name: Create systemd unit file for Node Exporter on server2

become: yes

template:

src: node_exporter.service

dest: /etc/systemd/system/node_exporter.service

notify: restart node_exporter

when: "'server2' in inventory_hostname"

- name: Enable and start Node Exporter service on server2

become: yes

systemd:

name: node_exporter

enabled: yes

state: started

when: "'server2' in inventory_hostname"

- name: Add Prometheus APT key

apt_key:

url: <https://packages.grafana.com/gpg.key>

when: "'server2' in inventory_hostname"

- name: Add Prometheus APT repository

apt_repository:

repo: deb <https://packages.grafana.com/oss/deb> stable main

when: "'server2' in inventory_hostname"

- name: Download and extract Prometheus on server2

become: true

shell: |

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

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

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

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

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

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

```
args:
```

```
executable: /bin/bash
```

```
when: "'server2' in inventory_hostname"
```

```
- name: Add lines to /etc/hosts on server2
```

```
lineinfile:
```

```
path: /etc/hosts
```

```
line:
```

```
- '127.0.0.1 localhost'
```

```
- '{{ prometheus_server2_ip }} backupmachine'
```

```
- '{{ node_exporter_server1_ip }} vmprimaire'
```

```
state: present
```

```
when: "'server2' in inventory_hostname"
```

```
- name: Create Prometheus configuration file
```

```
become: yes
```

```
template:
```

```
src: prometheus.yml.j2
```

```
dest: /etc/prometheus/prometheus.yml
```

```
notify: restart prometheus
```

```
when: "'server2' in inventory_hostname"
```

```
- name: Create systemd unit file for Prometheus
```

```
become: yes
```

```
template:
```

```
src: prometheus.service
```

```
dest: /etc/systemd/system/prometheus.service
```

```
notify: restart prometheus
```

```
when: "'server2' in inventory_hostname"

- name: Ensure Prometheus directories ownership
  file:
    path: "{{ item }}"
    state: directory
    owner: prometheus
    group: prometheus
    mode: '0755'
  loop:
    - /etc/prometheus
    - /var/lib/prometheus
  when: "'server2' in inventory_hostname"

- name: Restart Prometheus on server2
  become: true
  shell: |
    sudo systemctl daemon-reload
    sudo systemctl enable prometheus
    sudo systemctl start prometheus
  args:
    executable: /bin/bash
  when: "'server2' in inventory_hostname"

- name: Install Grafana on server2
  apt:
    name: grafana
    state: present
  when: "'server2' in inventory_hostname"
```

- name: Enable and start Grafana service

systemd:

name: grafana-server

enabled: yes

state: started

when: "'server2' in inventory_hostname"

- name: Restart Grafana on server2

become: yes

shell: |

sudo systemctl daemon-reload

args:

executable: /bin/bash

when: "'server2' in inventory_hostname"

- name: Install Prometheus, AlertManager, and Grafana on server2

package:

name: "{{ item }}"

state: present

loop:

- prometheus

- grafana

when: "'server2' in inventory_hostname"

handlers:

- name: restart node_exporter

systemd:

```
name: node_exporter
state: restarted

- name: restart prometheus

systemd:
name: prometheus
state: restarted

- name: restart grafana

systemd:
name: grafana
state: restarted
```

Ansible File – 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

Ansible File – prometheus.service

[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

[Ansible File – prometheus.yml](#)

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: ['{{ prometheus_server2_ip }}:9090']

- job_name: 'node_exporter_metrics'

scrape_interval: 5s

static_configs:

- targets: ['localhost:9100', '{{ node_exporter_server1_ip }}:9100', '{{ node_exporter_server3_ip }}:9100', , '{{ prometheus_server2_ip }}:9100']

[Ansible File – Grafana.service](#)

[Unit]

Description=Grafana

After=network.target

[Service]

User=grafana

Group=grafana

ExecStart=/usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --

pidfile=/var/run/grafana/grafana-server.pid

Restart=always

[Install]

WantedBy=multi-user.target

[Ansible File – Backup.py.yml](#)

```
import paramiko
```

```
import os
```

```
import datetime
```

```
import sys
```

```
import subprocess
```

```
php_script_path = '/home/epitech/backup.php'
```

```
# Run the PHP script using subprocess
```

```
result = subprocess.run(['php', php_script_path], capture_output=True, text=True)
```

```
# Print the output and error if any
```

```
print('Output:', result.stdout)
```

```
print('Error:', result.stderr)
```

```
# Check the return code to see if the command was successful
```

```
if result.returncode == 0:
```

```
    print('PHP script executed successfully.')
```

```
else:
```

```
    print(f'Error executing PHP script. Return code: {result.returncode}')
```

```
sys.exit()
```

```
# Paramètres de la connexion SSH pour le serveur de destination
```

```
destination_hostname = 'vmbackup.westeurope.cloudapp.azure.com'
```

```
destination_username = 'epitech'
```

```
destination_key_filename = '/home/epitech/vpscloud' # chemin vers le fichier id_rsa
```

```
destination
```

```
# Paramètres du transfert de fichiers
```

```
remote_path = '/tmp/' # répertoire distant où les fichiers seront transférés
```

```
local_path = '/tmp/db-backups' # répertoire local contenant les fichiers à transférer
```

```
# Créer 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)
```

```
# Local_dir = '/var/lib/pgbackrest/backup/db-primary'
```

```
remote_dir = '/tmp/'
```

```
x = datetime.datetime.now()
```

```
DATE = x.strftime("%d-%m-%Y")
```

```
zip_path = f'/tmp/db-backups/db-mysql_{DATE}.sql.zip'
```

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

```
php_script_path = '/home/epitech/backup.php'
```

```
# Transférer les fichiers du serveur source vers le serveur de destination
```

```
sftp = destination_client.open_sftp()
```

```
sftp.put(zip_path, remote_path + f'db-mysql_{DATE}.sql.zip')
```

```
sftp.close()
```

```
print('Le fichier a été transféré avec succès.')
```

```
# Fermer les connexions SSH
```

```
# source_client.close()
```

```
destination_client.close()
```

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>

Backup des conteneurs : <https://www.geeksforgeeks.org/backing-up-a-docker-container/>