PROMETHEUS

Alves Lobo Michael

- https://www.linkedin.com/in/michael-alves-lobo
- https://github.com/kairel-4a0057b3/
- devops, dev, réseau et administration système
- En poste chez Click2buy
- https://github.com/kairel/learning/blob/main/devops_prometheus.pdf
- Automatisation
  - ansible
  - puppet
  - terraform
  - docker

SOMMAIRE

Prometheus

- 1.Pr
- 2.Pe
-

Prometheus - Présentation

Comment on va procéder ?

- Pas bcp de théorie , bcp de pratique
- On va déployer une plateforme complète
- On va d'abord essayer de faire des choses qui fonctionnent , et on le modifiera pour suivre les bonnes pratiques et que ce soit réutilisable et compréhensible

Prometheus - Présentation
1.1 définition

Prometheus est avant tout un logiciel de supervision

Il permet donc de superviser et de récolter des métriques sur des ressources (des serveurs, des applicatifs, des firewall etc ..)

Prometheus - Présentation

1.1 définition

Prometheus est un système de surveillance open source basé sur des métriques. Il collecte les données des services et des hôtes en scrappant les metriques sur des endpoints.

Il stocke ensuite les résultats dans une base de données chronologique et le rend disponible pour analyse et alerte.

Prometheus - Présentation
1.1 définition

Prometheus est fait en GO, en réalité prometheus et un ensemble de plusieurs logiciels

- une base de données orientée time-series qui permet de stocker les métriques
- une interface graphique qui permet de gérer et de visualiser ces métriques
- un logiciel d'alerting : alertmanager
- un logiciel d'observabilité : grafana (dashboards)

Prometheus - Présentation
1.2 Installation

- On peut installer Prometheus de plein de façons différentes, on va utiliser l'installation via docker & docker-compose, c'est la plus simple et si on se trompe ça n'affectera que le container que l'on aura installer

Prometheus - Présentation
1.2 Installation

- installation docker avec le user prometheus

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key add
add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/debian $(lsb_release
-cs) stable"

sudo   apt update
sudo apt install docker-ce
```

Prometheus - Présentation
1.2 Installation

- On ajoute ensuite l’utilisateur prometheus au groupe docker

sudo usermod -aG docker prometheus

Prometheus - Présentation
1.2 Installation

- docker-compose

dans le répertoire /tmp

```
wget https://github.com/docker/compose/releases/download/v2.5.1/docker-compose-linux-x86_64
mv /tmp/docker-compose-linux-x86_64 /usr/local/bin/docker-compose
chmod +x /usr/local/bin/docker-compose
```

Prometheus - Présentation
1.2 Installation

Pour vérifier que tout est ok on va lancer ces 2 commandes

docker container ls

docker-compose -v

Prometheus - Présentation
2.1 Configuration

Prometheus est configuré via la ligne de commande et un fichier de configuration. Alors que les indicateurs de ligne de commande configurent des paramètres système immuables (tels que les emplacements de stockage, la quantité de données à conserver sur le disque et en mémoire, etc.), le fichier de configuration définit tout ce qui concerne les tâches de scraping et leurs instances, ainsi que les fichiers de règles charger.

Prometheus

MAL - 23/09/2022

Pourquoi Prometheus

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

- Solution plus souple permettant de facilement intégrer de nouvelles sondes
- Opensource
- standard actuel
- mode scraping
- interrogation des métriques en direct
- intégration native avec grafana
- orienté alerting (facilite la mise en place d’alerte)

Prometheus chez C2b

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

1- les briques choisis

On utilise 3 briques de bases

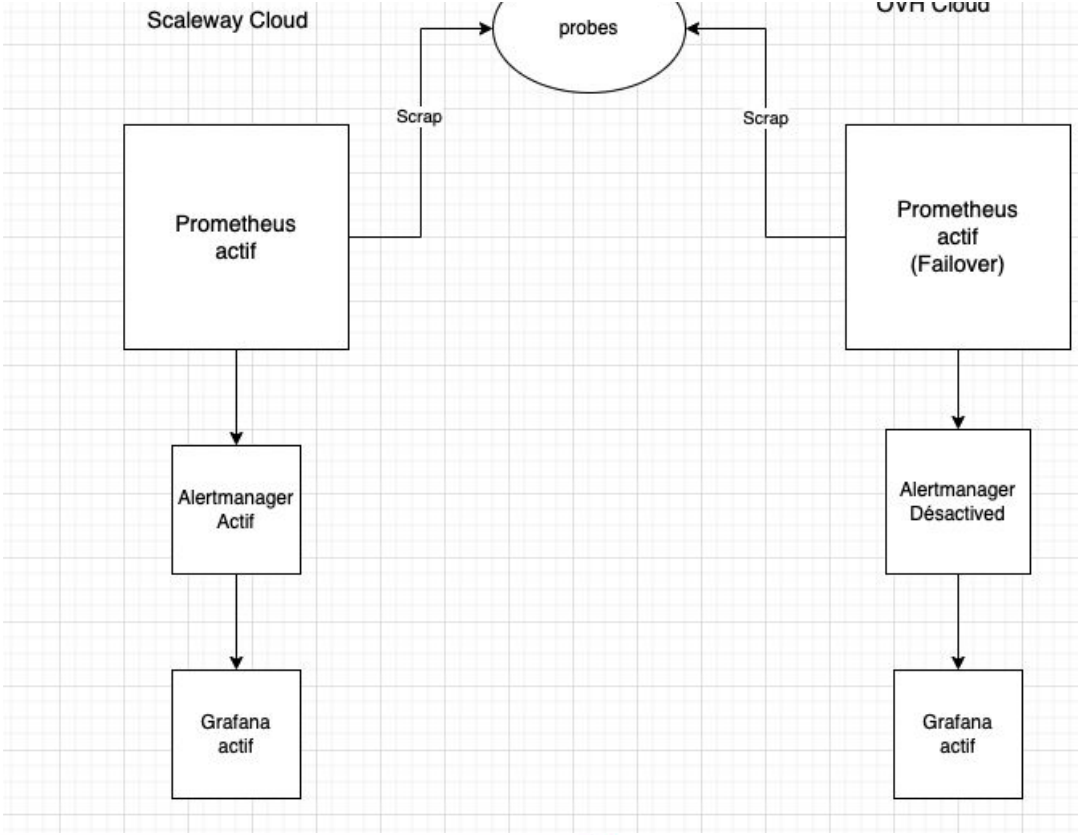
- prometheus -> (base de donnéés + interface web)
- alertmanager -> (routage des alerts)
- grafana -> (dashboard)

à ces briques on ajoute de l'automatisation avec

- ansible (déploiement)
- docker (infra)

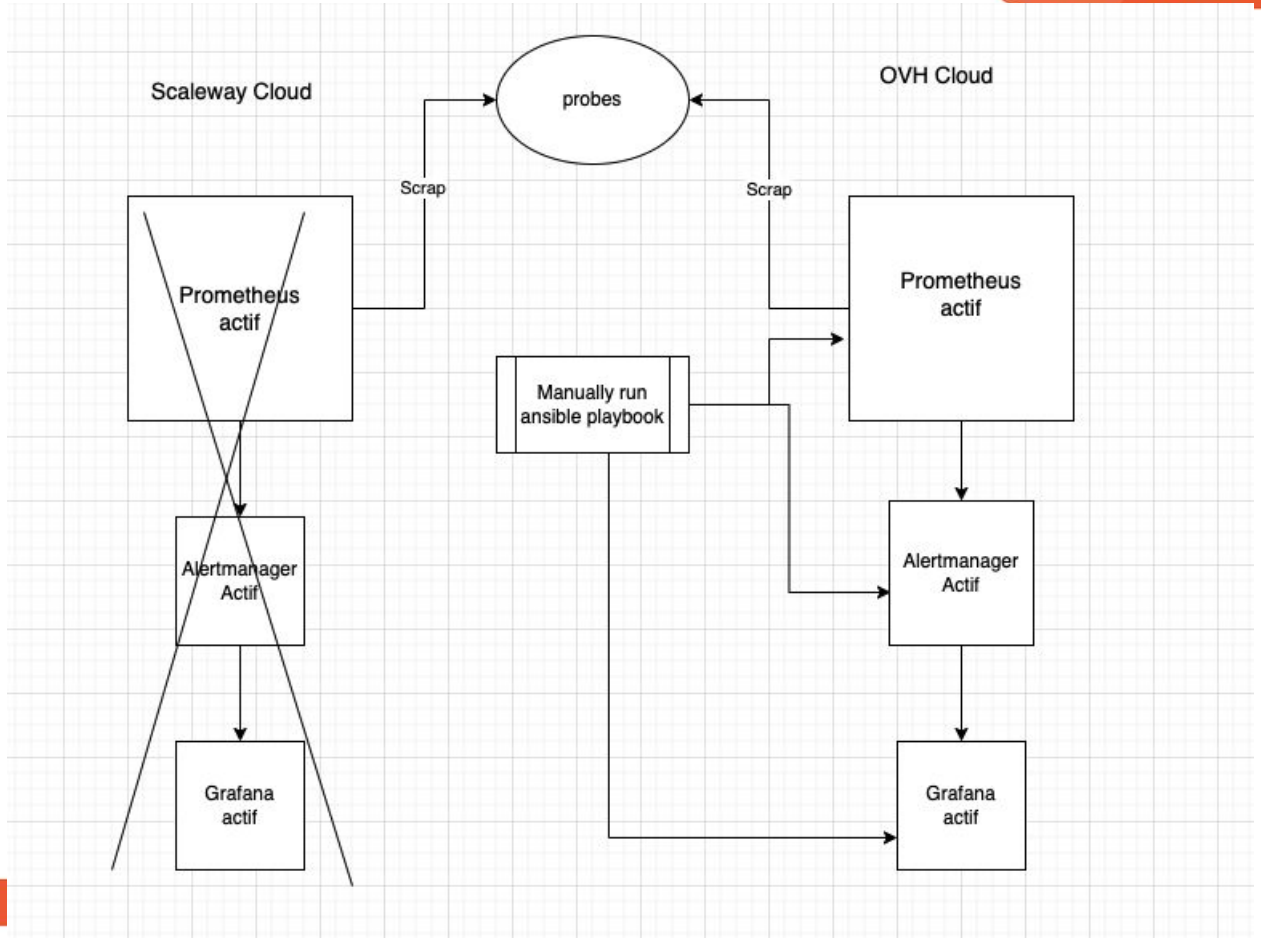
Prometheus chez C2b

Infra en place
- production



Prometheus chez C2b

Infra en place
- Failover



Prometheus les sondes

Pour Prometheus une sonde est un exporter on a ainsi une liste d’exporter à installer

- node_exporter (serveur)
- psql_exporter (postgresql)

chaque exporter exposent ces métriques via un port avec une url du style

http://probe.click2buy.com:9145/metrics

Prometheus les sondes

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

Les métriques ont un format bien spécifique:

```
# HELP activemq_queues_consumer_count Telegraf collected metric
# TYPE activemq_queues_consumer_count untyped
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsagent2Hbase.VirtualTopic.bor",port="8161",source="192.168.0.120
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsagent2Hbase.VirtualTopic.scraper",port="8161",source="192.168.0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsagent2Hive.VirtualTopic.scraper",port="8161",source="192.168.0.
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmslogagent2Hbase.VirtualTopic.scraper",port="8161",source="192.16
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsstatsagent2Hbase.VirtualTopic.bor",port="8161",source="192.168.
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="DLQ.Consumer.FlumeJmsagent2Hbase.VirtualTopic.scraper",port="8161",source="192.1
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="DLQ.Consumer.FlumeJmsagent2Hive.VirtualTopic.scraper",port="8161",source="192.16
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="bor",port="8161",source="192.168.0.120"} 2
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="crawlhigh",port="8161",source="192.168.0.120"} 0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="crawllow",port="8161",source="192.168.0.120"} 0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="daily",port="8161",source="192.168.0.120"} 0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="mx_crawlhigh",port="8161",source="192.168.0.120"} 0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="mx_crawllow",port="8161",source="192.168.0.120"} 0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="ru_crawlhigh",port="8161",source="192.168.0.120"} 0
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="ru_crawllow",port="8161",source="192.168.0.120"} 1
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="uk_crawlhigh",port="8161",source="192.168.0.120"} 1
activemq_queues_consumer_count{host="prod-hadoop-extra-1.clic2buy.com",name="uk_crawllow",port="8161",source="192.168.0.120"} 0
# HELP activemq_queues_dequeue_count Telegraf collected metric
# TYPE activemq_queues_dequeue_count untyped
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsagent2Hbase.VirtualTopic.bor",port="8161",source="192.168.0.120"
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsagent2Hbase.VirtualTopic.scraper",port="8161",source="192.168.0.
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsagent2Hive.VirtualTopic.scraper",port="8161",source="192.168.0.1
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmslogagent2Hbase.VirtualTopic.scraper",port="8161",source="192.168
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="Consumer.FlumeJmsstatsagent2Hbase.VirtualTopic.bor",port="8161",source="192.168.0
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="DLQ.Consumer.FlumeJmsagent2Hbase.VirtualTopic.scraper",port="8161",source="192.16
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="DLQ.Consumer.FlumeJmsagent2Hive.VirtualTopic.scraper",port="8161",source="192.168
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="bor",port="8161",source="192.168.0.120"} 81965
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="crawlhigh",port="8161",source="192.168.0.120"} 1.046736e+06
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="crawllow",port="8161",source="192.168.0.120"} 874333
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="daily",port="8161",source="192.168.0.120"} 277454
activemq_queues_dequeue_count{host="prod-hadoop-extra-1.clic2buy.com",name="mx_crawlhigh",port="8161",source="192.168.0.120"} 393987
```

Prometheus les sondes

Une fois les sondes configurées , il faut demande à prometheus de venir les “scraper” , en gros de parser et stocker les metriques.

Dans Prometheus on doit donc:

- définir le lien ou scraper ces métriques (targets)
- check que les données sont bonnes et valides
- optionnellement créer des régles d’alerting

Prometheus les sondes

- 1 : les targets

les targets sont les cibles sur lesquelles prometheus doit aller chercher l'information , un exemple de définition de target dans prometheus

```
- job_name: node_high_cpu
  static_configs:
    - targets: ['prod-bhs3-scrapping.server.clic2buy.com:7694', 'prod-gra5-scrapping-1.server.clic2buy.com:7694', 'prod
- job_name: flume
  static_configs:
    - targets: ['prod-hadoop-extra-1.server.clic2buy.com:9273']
- job_name: rundeck
  static_configs:
    - targets: ['51.77.207.100:9620']
- job_name: alertmanager
  static_configs:
    - targets: ['alertmanager:9093']
  basic_auth:
    username: 'admin'
    password: 'JDkdo24Ld4LM0'
- job_name: 'blackbox'
```

Prometheus les targets

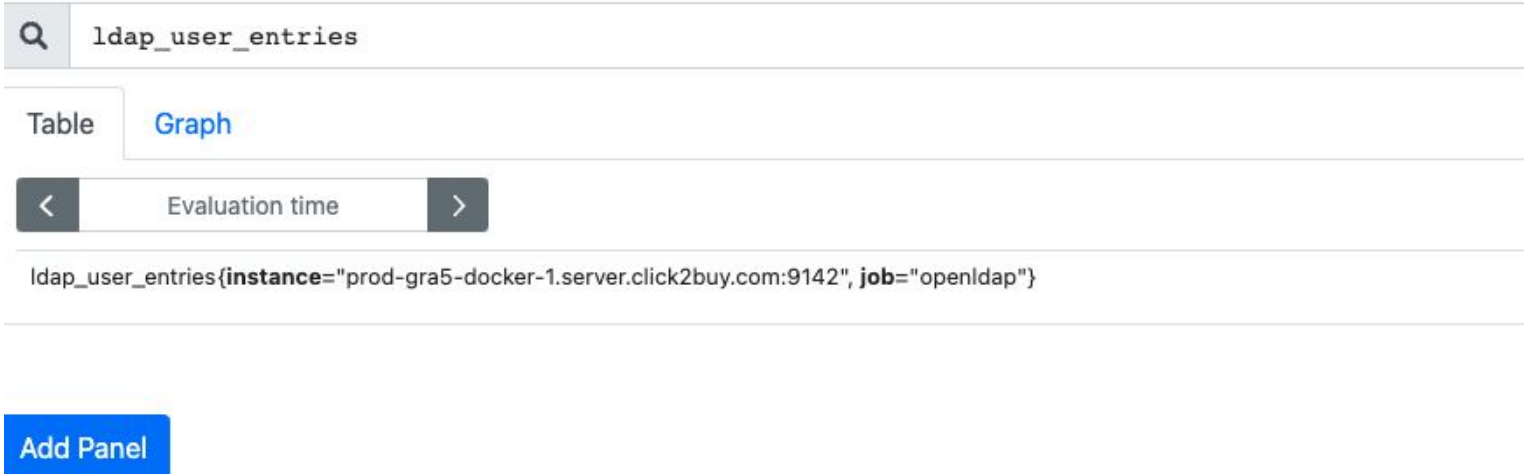
- 1 : les targets
  - job_name: une façon de grouper les métriques par catégories
  - static_config: permet de spécifier les url des exporters
  - basic_auth: si l'exporter à une auth basic (marche aussi avec token)

```
- job_name: node_high_cpu
  static_configs:
    - targets: ['prod-bhs3-scrapping.server.clic2buy.com:7694', 'prod-gra5-scrapping-1.server.clic2buy.com:7694', 'prod
- job_name: flume
  static_configs:
    - targets: ['prod-hadoop-extra-1.server.clic2buy.com:9273']
- job_name: rundeck
  static_configs:
    - targets: ['51.77.207.100:9620']
- job_name: alertmanager
  static_configs:
    - targets: ['alertmanager:9093']
  basic_auth:
    username: 'admin'
    password: 'JDkdo24Ld4LM0'
- job_name: 'blackbox'
```

Prometheus les requêtes

- 2 : les requêtes

  Une fois les métriques stockées dans la base de données de prometheus (ingluxdb ou quelque chose du genre) , on peut les intérroger via un langage (promql) directement depuis l'interface graphique



Prometheus les requêtes

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

- 2 : les requêtes

  Le langage de requête est très riche et permet de faire des requêtes complexe

  - req normales
  - req d'aggrégation
  - req dans le temps (passé)
  - req prédictive

l'interface graphique permet l'auto-complétion des requêtes , on peut afficher des graphs (avec les offsets) -> présenter en direct sur l'outil

Prometheus les alertes

- 3 : les alerts

Une fois que l'on connait le langage de requête on peut donc passer à l'étape suivante les régles

Avec Prometheus on peut définir des régles qui permettront de générer les alertes

```
groups:
- name: Openldap
  rules:
  - alert: NotEnoughGroup
    expr: ldap_group_entries < 4
    for: 5m
    labels:
      severity: warning
    annotations:
      summary: Not Enough group in ldap
      description: "Ldap must have most than 4 groups (instance {{ $labels.instance }})"

  - alert: NotEnoughUser
    expr: ldap_user_entries < 10
    for: 5m
    labels:
      severity: warning
    annotations:
      summary: Not Enough user in ldap
      description: "Ldap must have most than 9 users (instance {{ $labels.instance }})"
```

Prometheus les alertes

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

- 3 : les alerts

  on les classes par groupes

  - chaque alerte à un nom
  - chaque alerte à une desription
  - chaque alerte à une expression
  - chaque alerte a un label(severity, message d’erreur)

Prometheus les alertes

- 3 : les alerts

L'expression est en fait le calcul de l'alerte : un exemple

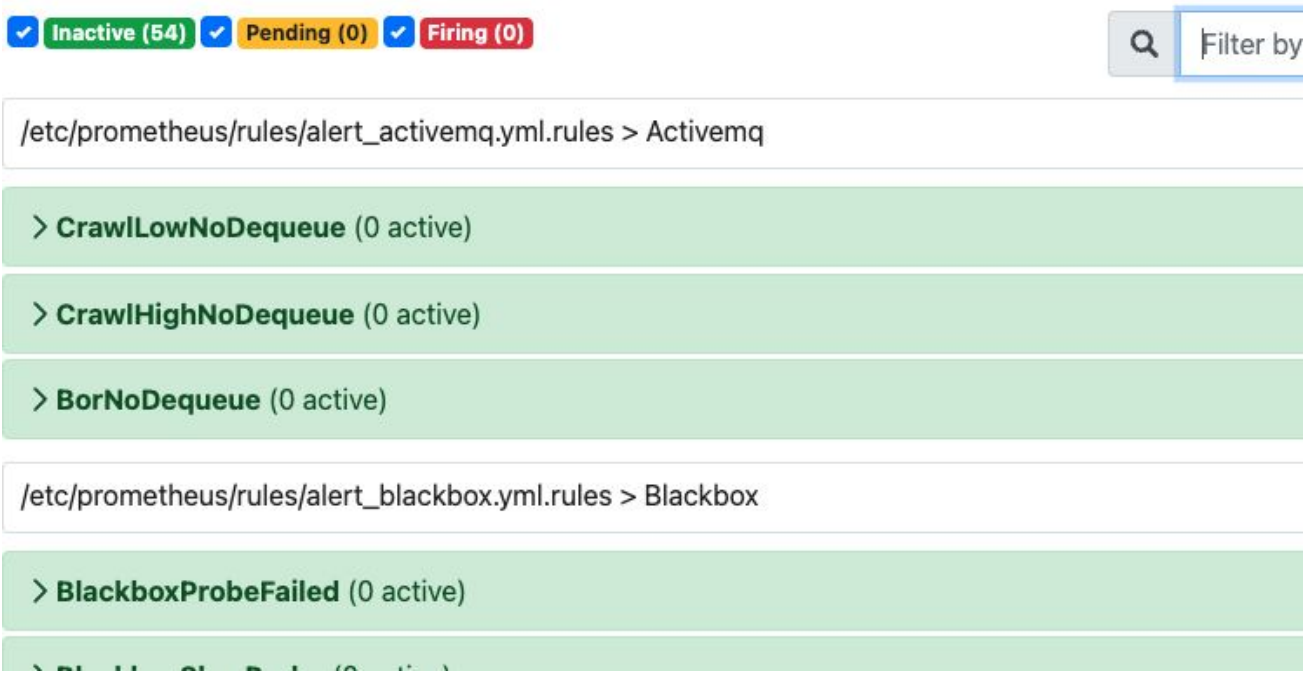
```
- alert: NotEnoughGroup
  expr: ldap_group_entries < 4
  for: 5m
```

- expr: le nombre de groupe est < 4 pendant 5m génère une alerte

Prometheus les alertes

- 3 : les alerts

interface



AlertManager

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

Une fois les alertes configurées , on peut décider de vouloir envoyer des mails , sms , notifications slack etc ..

Pour cela on utilise AlertManager qui est une application nativement liée à prometheus, pour cela il suffit de dire a prometheus de l'utiliser

```
alerting:
  alertmanagers:
  - scheme: http
    static_configs:
    - targets: ["alertmanager:9093"]
    basic_auth:
      username: admin
      password: JDkdo24Ld4LM0
```

AlertManager

CLICK2BUY
YOU COMMUNICATE. WE CONNECT.

AlertManager est conçu comme un systéme de routage qui oriente les alertes vers les receivers spécifiés

```
global:
  resolve_timeout: 2m

route:
  group_by: ['instance', 'severity']
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 3h
  # If an alert isn't caught by a route, send it slack.
  receiver: slack_prometheus

receivers:
- name: slack_prometheus
  slack_configs:
  - api_url: https://hooks.slack.com/services/T02C8ENJW/B017HEAU8L9/V
    channel: '#prometheus'
    icon_emoji: ":rotating_light:"
    send_resolved: true
    title: |-
     [{{ .Status | toUpper }}{{ if eq .Status "firing" }}:{{ .Alerts.
     {{- if gt (len .CommonLabels) (len .GroupLabels) -}}
       {{" "}}(
       {{- with .CommonLabels.Remove .GroupLabels.Names }}
```

Un exemple concret

Comment ça se passe concrètement pour superviser un serveur

il faut d’abord deployer la sonde : ça se fait via ansible

```
ansible-playbook ansible/playbooks/prometheus/probes/node_manager_deploy.yml --extra-vars

"templates_path=/Users/malveslobo/c2b_devops/templates"
```

Un exemple concret

Une fois la sonde déployée (et autorisée avec iptables), on ajoute le serveur dans la liste des métriques à scrapper ..

on va donc dans le fichier prometheus.yml.j2 sur c2b_devops et on ajoute notre serveur

```
scrape_configs:
  - job_name: node
    static_configs:
    - targets: ['prod-hadoop-yarn-1.server.clic2buy.com:7694', 'prod-hadoop-yarn-2.server.clic2buy.com:7694',
'prod-hadoop-yarn-3.server.clic2buy.com:7694', 'prod-hadoop-yarn-4.server.clic2buy.com:7694']
    - targets: ['prod-rbx8-docker-solr.server.clic2buy.com:7694']
```

Un exemple concret

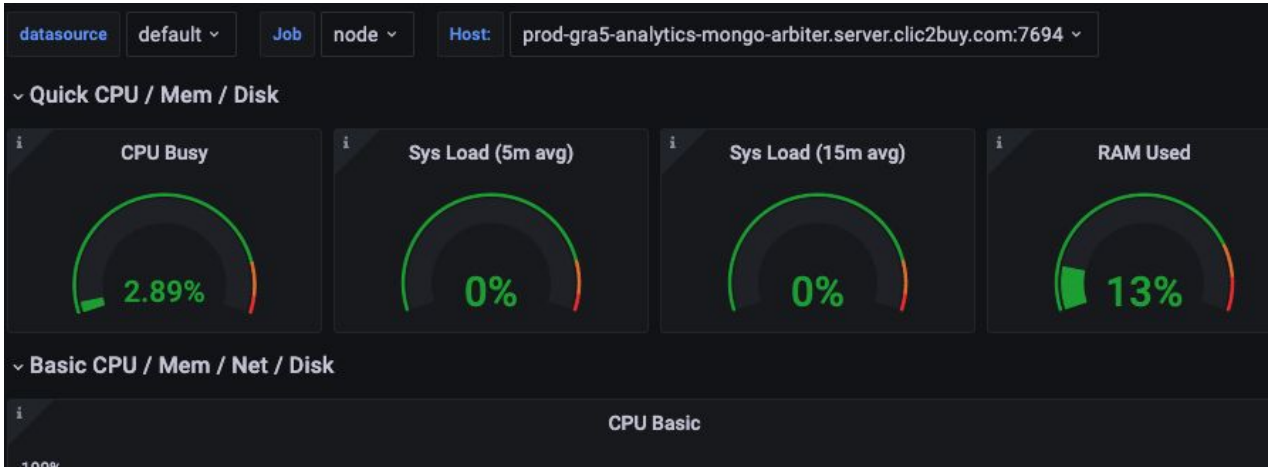
on deploie la nouvelle conf avec ansible

```
ansible-playbook ansible/playbooks/prometheus/update.yml --extra-vars
"templates_path=/Users/malveslobo/c2b_devops/templates"
```

Un exemple concret

et c’est tout le serveur est configuré et présent dans notre supervision

il apparaîtra également dans Grafana



Prometheus - Le projet

- Le but de ce projet qui va durer de quelques heures à quelques jours va être de:
    - Déployer automatiquement depuis ansible 2 VM(s) debian 11 de 50Go , 2CPUs et 4Go de RAM (pour ceux qui veulent aller plus vite vous pouvez les creer manuellement)
    - sur ces 2 vms on va créer
        - un utilisateur prometheus et ansible , tous les deux dans le groupe sudo
        - on utilisera l'utilisateur ansible pour lancer les playbooks ansible
        - on va installer docker et docker compose (je vous donnerais les binaires nécessaire pour le faire)
    - sur une seule des vms qui sera donc la vm ou l'on va installer prometheus
        - on va créer les repertoires:
            - /opt/prometheus
            - /opt/prometheus/conf
            - /opt/prometheus/probes
            - /opt/prometheus/rules
            - /opt/prometheus/targets

Prometheus - Le projet

    - copier les fichiers de conf que je vais vous donner
    - faire tourner prometheus
  - sur la 1ére Vm on va installer
    - node_exporter (sonde systéme)
- on va donc installer prometheus sur une des vm et superviser l'autre avec prometheus