Prometheus

Prometheus est un système de monitoring et d'alerte open source conçu pour la fiabilité et l'efficacité, particulièrement adapté pour les environnements de cloud computing et les architectures de services distribués.

Il collecte et stocke ses métriques sous forme de séries temporelles, permettant aux utilisateurs de surveiller l'état de leurs systèmes et services en temps réel.

- Présentation
  - Un peu d'histoire
  - Caractéristiques
- Prometheus docker-compose
- Installation & Configuration

Présentation

Un peu d'histoire

L'histoire de Prometheus commence en 2012 chez SoundCloud, une plateforme de partage musical. Confrontés à la nécessité de surveiller une infrastructure de plus en plus complexe et à l'insatisfaction vis-à-vis des solutions de monitoring existantes, Matt T. Proud et Julius Volz, alors ingénieurs chez SoundCloud, ont commencé à développer Prometheus.

Leur objectif était de créer un système de surveillance et d'alerte qui soit à la fois puissant, fiable et adapté aux architectures modernes basées sur les conteneurs et les microservices.

Genèse et Philosophie

2012-2013 : Le développement de Prometheus débute. Inspiré par le système de monitoring Borgmon utilisé chez Google, où Matt et Julius avaient précédemment effectué des stages, Prometheus est conçu pour collecter et stocker des métriques sous forme de séries temporelles. Sa conception repose sur des principes tels que la simplicité, l'efficacité et une forte intégration avec l'écosystème des conteneurs.

Caractéristiques Principales

Modèle de données multidimensionnel : Prometheus utilise un modèle de données puissant qui permet aux utilisateurs de définir et de requêter des métriques avec une grande flexibilité.

Collecte de métriques via le scraping HTTP : Prometheus collecte des métriques en "scrapant" des endpoints HTTP exposés par les applications ou les services surveillés, ce qui facilite l'intégration avec une variété de sources.

Support des alertes : Prometheus inclut un composant d'alerte, Alertmanager, qui gère les notifications et les agrégations d'alertes, permettant aux équipes d'être rapidement informées des problèmes potentiels.

Croissance et Adoption

2015 : Prometheus est rendu public et open source, ce qui marque le début de sa croissance rapide au sein de la communauté des développeurs et des professionnels de l'IT. Sa capacité à répondre aux défis posés par les architectures distribuées et à grande échelle lui vaut une adoption rapide.

2016 : Prometheus rejoint la Cloud Native Computing Foundation (CNCF) en tant que deuxième projet hébergé, après Kubernetes. Cette intégration à la CNCF contribue à accélérer son adoption et à valider son importance dans l'écosystème des technologies cloud natives.

Évolution et Fonctionnalités

Depuis son lancement, Prometheus a continué à évoluer, ajoutant de nouvelles fonctionnalités, améliorant ses performances et étendant son écosystème d'intégrations. La communauté autour de Prometheus a également grandi, contribuant à son développement et à sa documentation.

Impact et Utilisation

Devenu un standard de facto : Prometheus est devenu un standard de facto pour le monitoring dans les environnements cloud natifs, largement utilisé pour surveiller des applications, des infrastructures et des services à grande échelle.

Large écosystème : L'écosystème de Prometheus s'est élargi pour inclure des intégrations avec de nombreux autres outils et systèmes, des dashboards personnalisables avec Grafana, à l'intégration avec des systèmes d'orchestration comme Kubernetes.

Aujourd'hui et l'Avenir

Continuité de l'innovation : Prometheus continue d'innover et de s'adapter aux besoins changeants des architectures modernes, avec une communauté active qui pousse constamment le projet vers de nouvelles frontières.

Importance croissante de l'observabilité : Dans un contexte où l'observabilité devient de plus en plus cruciale pour la fiabilité des systèmes informatiques, Prometheus joue un rôle clé en fournissant des données précieuses pour l'analyse des performances et la détection des anomalies.

L'histoire de Prometheus est celle d'une solution née d'un besoin interne et qui a évolué pour devenir un pilier de l'observabilité dans les environnements cloud natifs.

Sa conception innovante, son modèle de données puissant et sa communauté active en font une composante essentielle de l'écosystème des technologies open source modernes.

Caractéristiques

Collecte de Données Multidimensionnelle

- **Modèle de données puissant** : Prometheus stocke les données sous forme de séries temporelles, identifiées par leur nom de métrique et leurs paires clé/valeur (étiquettes), permettant une collecte de données multidimensionnelle.
- **Pull model** : Contrairement à de nombreux systèmes de monitoring qui utilisent un modèle push, Prometheus récupère (pull) activement les métriques depuis les cibles configurées à des intervalles définis.

Langage de Requête Flexible

- **PromQL** : Prometheus introduit PromQL, un langage de requête puissant qui permet aux utilisateurs de sélectionner et d'agréger des données de séries temporelles. Cela facilite l'analyse complexe et la visualisation des métriques.

Support des Alertes

- **Gestion des alertes** : Prometheus permet de définir des règles d'alerte basées sur les données de séries temporelles. Lorsque les conditions d'alerte sont remplies, les notifications peuvent être envoyées via l'Alertmanager, un composant de Prometheus dédié à la gestion des alertes.
- **Alertmanager** : Gère les notifications et les agrégations d'alertes, supportant plusieurs canaux de notification comme l'email, Slack, et PagerDuty.

Haute Disponibilité et Fiabilité

- **Stockage efficace** : Utilise un modèle de stockage local hautement efficace pour stocker les données de séries temporelles, optimisé pour un accès rapide et une utilisation efficace de l'espace disque.
- **Support de la haute disponibilité** : Bien que Prometheus lui-même ne soit pas conçu pour être exécuté en mode cluster, il peut être configuré en instances multiples pour assurer la redondance et la haute disponibilité.

Découverte de Services

- **Découverte dynamique** : Supporte la découverte de services dans divers environnements, y compris Kubernetes, AWS, et d'autres, permettant une configuration automatique des cibles de monitoring à mesure que les systèmes évoluent.

Intégrations et Ecosystème

- **Exportateurs** : Un large éventail d'exportateurs est disponible pour Prometheus, permettant de collecter des métriques depuis divers systèmes et services qui ne supportent pas nativement Prometheus.
- **Intégration avec Grafana** : Prometheus est souvent utilisé en tandem avec Grafana pour la visualisation avancée des données de monitoring, offrant une solution complète pour le monitoring et l'alerte.

Communauté et Support

- **Communauté active** : Bénéficie d'une large communauté d'utilisateurs et de contributeurs qui développent constamment de nouvelles fonctionnalités, des améliorations, et des corrections.
- **Documentation riche** : Fournit une documentation complète et des tutoriels pour aider les utilisateurs à démarrer, à configurer le monitoring, et à utiliser PromQL pour l'analyse des données.

En résumé, Prometheus est une solution de monitoring puissante et flexible, idéale pour les environnements dynamiques et à grande échelle. Sa capacité à collecter, stocker, et analyser des métriques de manière efficace, combinée à un système d'alerte robuste et à une large intégration avec d'autres outils, en fait un choix privilégié pour les équipes de développement et d'opérations cherchant à surveiller la santé et les performances de leurs systèmes.

Prometheus docker-compose

docker compose prometheus

```
version: '3.3'
services:
  prometheus:
   ports:
    - '9093:9090'
   image: prom/prometheus
   volumes:
    - /mnt/koa-02/prometheus:/etc/prometheus
   deploy:
     mode: global
     placement:
       constraints: [node.hostname == EVA-02]
```

Les dernières images sont disponibles sur le site officiel Docker Hub, accessibles via ce lien :

Prometheus

Fichier de scraping **prometheus.yml**

```
# my global config
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093

# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
```

```
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
    - targets: ['localhost:9090']

################################################################################
################################

  - job_name: 'docker-eva-00-netdata-scrape'

    metrics_path: '/api/v1/allmetrics'
    params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    honor_labels: true
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:19999']

################################################################################
################################

  - job_name: 'docker_eva-01-netdata-scrape'
```

```
    metrics_path: '/api/v1/allmetrics'
    params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    honor_labels: true
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:19999']

##############################################################################
################################

  - job_name: 'docker_eva-02-netdata-scrape'

    metrics_path: '/api/v1/allmetrics'
    params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    honor_labels: true
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:19999']
```

```
#####################################################################
################################

  - job_name: 'nextcloud-netdata-scrape'

    metrics_path: '/api/v1/allmetrics'
    params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    honor_labels: true
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:19999']

#####################################################################
################################

  - job_name: 'Storage-koa-01-netdata-scrape'

    metrics_path: '/api/v1/allmetrics'
    params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
```

```
    # for Netdata to uniquely identify it
    #server: ['prometheus1']
  honor_labels: true
  static_configs:
    - targets: ['XXX.XXX.XXX.XXX:19999']

##############################################################################
################################

  - job_name: 'Storage-koa-02-netdata-scrape'

    metrics_path: '/api/v1/allmetrics'
    params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    honor_labels: true
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:19999']

##############################################################################
################################

  - job_name: 'cadvisor-docker-eva-02-scrape'

    metrics_path: '/metrics'
    scheme: http
    #params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      #format: [prometheus]
```

```
    #
    # source: as-collected | raw | average | sum | volume
    # default is: average
    #source: [as-collected]
    #
    # server name for this prometheus - the default is the client IP
    # for Netdata to uniquely identify it
    #server: ['prometheus1']
  #honor_labels: true
  static_configs:
    - targets: ['XXX.XXX.XXX.XXX:8489']
  relabel_configs:
    - separator: ;
      regex: (.*)
      target_label: instance
      replacement: cadvisor-docker-eva-02
      action: replace


################################################################################
################################

  - job_name: 'cadvisor-docker-eva-01-scrape'

    metrics_path: '/metrics'
    scheme: http
    #params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
      #format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    #honor_labels: true
    static_configs:
```

```
    - targets: ['XXX.XXX.XXX.XXX:8488']
  relabel_configs:
    - separator: ;
      regex: (.*)
      target_label: instance
      replacement: cadvisor-docker-eva-01
      action: replace


#####################################################################
################################

 - job_name: 'cadvisor-docker-eva-00-scrape'

  metrics_path: '/metrics'
  scheme: http
  #params:
    # format: prometheus | prometheus_all_hosts
    # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname
instead of IP
    #format: [prometheus]
    #
    # source: as-collected | raw | average | sum | volume
    # default is: average
    #source: [as-collected]
    #
    # server name for this prometheus - the default is the client IP
    # for Netdata to uniquely identify it
    #server: ['prometheus1']
  #honor_labels: true
  static_configs:
    - targets: ['XXX.XXX.XXX.XXX:8487']
  relabel_configs:
    - separator: ;
      regex: (.*)
      target_label: instance
      replacement: cadvisor-docker-eva-00
      action: replace


#####################################################################
```

```
################################

  - job_name: 'OPNSense'

    metrics_path: '/metrics'
    scheme: http
    #params:
      # format: prometheus | prometheus_all_hosts
      # You can use `prometheus_all_hosts` if you want Prometheus to set the `instance` to your hostname instead of IP
      #format: [prometheus]
      #
      # source: as-collected | raw | average | sum | volume
      # default is: average
      #source: [as-collected]
      #
      # server name for this prometheus - the default is the client IP
      # for Netdata to uniquely identify it
      #server: ['prometheus1']
    #honor_labels: true
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:9100']
    #relabel_configs:
    #  - separator: ;
    #    regex: (.*)
    #    target_label: instance
    #    replacement: cadvisor-docker-minion-4

#################################################################################################################################

  - job_name: 'uptime'
    scrape_interval: 30s
    scheme: http
    metrics_path: '/metrics'
    static_configs:
      - targets: ['XXX.XXX.XXX.XXX:3001']
    basic_auth: # Only needed if authentication is enabled (default)
      username: XXXXXXXX
      password: XXXXXXXX
```

```
###################################################################
##############################
```

Installation & Configuration