Rules avec NGOT

Démo Point de synchro du 02/06/23

Plan

- Rappels (4 slides)
- NGOT : changements (1 slide)
- NGOT : création et déploiement (5 slides)
- NGOT : labels dans les expressions (4 slides)
- Blackbox-Exporter sur NGOT (1 slide)
- Bonnes pratiques (2 slides)
- Conclusion (1 slide)

Rappels (1/4)

C’est quoi une “rule” ?

Une rule évalue une expression (au format PromQL)

- une “recording rule” génère une nouvelle métrique avec cette expression
- une “alerting rule” génère une alerte et l’envoie à Alertmanager

Une rule est évaluée par Prometheus. Une alerte est donc créée sur Prometheus (avant d’être envoyée à Alertmanager).

Rappels (2/4)

Les rules sur Caascad

Nous avons distingué 2 types de rules :

- un jeu de rules issues de sources upstream, mises en place et gérées par l'équipe Monitoring
- des rules dites "custom",
  - dans le repo https://git.corp.caascad.com/caascad/applications/caascad-prometheus-rules
  - dans le sous-répertoire `rules/rules/`

Rappels (3/4)

Le format des rules sur Caascad

```
rulesets: [RulesetName= "infra" | "cloud-caascad"]: apps: "thanos": groups: {
        ThanosCompactBucketOperationFailure: {
                rules: [
                        {
                                alert: "ThanosCompactBucketOperationFailure"
                                expr:  "sum(rate(thanos_objstore_bucket_operation_failu
                                for:   "5m"
                                labels: severity: "warning"
                                annotations: {
                                        message: "Thanos Compact has failing storage op
                                }
                        },
                ]
        }
}
```

Rappels (4/4)

Le déploiement des rules

```
trackbone apply -c prometheus-rules -z <zone infra|cloud>
```

Les rules sont toutes déployés dans le namespace `prometheusrules`

(sauf les rules “upstream” : `prometheusrules-dashboards-upstream`)

NGOT : changements (1/1)

Qu’est-ce qui change avec NGOT ?

Réponse : rien ne change ! Il n’y a que des ajouts !

1. On a juste ajouté 2 nouveaux noms de rulesets pour NGOT :
    - `cluster-ngot`
    - `service-ngot`
2. Pour les `service-ngot`,
    - le nom de l’`apps` prend de l’importance…
    - il faut ajouter la configuration `prometheus-rules` au service

NGOT : création et déploiement (1/5)

Le format des rules sur Caascad et NGOT

```
rulesets: [RulesetName= "infra" | "cloud-caascad" | "service-ngot"]: apps: "thanos": g
        ThanosCompactBucketOperationFailure: {
                rules: [
                        {
                                alert: "ThanosCompactBucketOperationFailure"
                                expr:  "sum(rate(thanos_objstore_bucket_operation_failu
                                for:   "5m"
                                labels: severity: "warning"
                                annotations: {
                                        message: "Thanos Compact has failing storage op
                                }
                        },
                ]
        }
}
```

Noter l’ajout de `service-ngot`

NGOT : création et déploiement (2/5)

service-ngot

Les services sont un concept nouveau car ils ne sont pas déployés sur tous les clusters. C'est pareil pour les rules.

Il faut donc ajouter la configuration `prometheus-rules` à la définition du service avec la liste des `apps` dont il faut déployer les rules.

Principes :

- les rules sont déployées sur les clusters NGOT, mais seulement ceux concernés par le service indiqué via l'`apps`
- les rules sont évaluées par les Prometheus des clusters
- les métriques concernées sont celles du service indiqué via l'`apps`

NGOT : création et déploiement (3/5)

service-ngot

Exemple : contexts/ngot/envs.cue

```
#NgotServiceZones: "grafana": {
        zone:           _
        infra_zone:     _
        admin_zone:     _
        configurations: close({
                .....
                "prometheus-rules": _ & {_rule_apps: [
                                        "grafana",
                                        "thanos-query",
                ]}
                .....
        })
}
```

NGOT : création et déploiement (4/5)

cluster-ngot

Les rules `cluster-ngot` sont très similaires aux rules `infra-caascad` ou `cloud-caascad` : elles concernent les applicatifs propres aux clusters.

Principes :

- les rules sont déployées sur tous les clusters NGOT
- les rules sont évaluées par les Prometheus des clusters
- les métriques concernées sont celles qu'on retrouve sur tous les clusters

C'est simple.

Note : les Prometheus centraux évaluent peu de rules : ce sont des cas particuliers gérés par l'équipe Monitoring

NGOT : création et déploiement (5/5)

Le déploiement des rules

```
trackbone apply -c prometheus-rules -z <zone>
```

Les rules

- sont déployés dans le namespace `prometheusrules`
  - (sauf les rules “upstream” : `prometheusrules-dashboards-upstream` mais ce namespace devrait changer dans quelques semaines)
- sont évaluées par le Prometheus du cluster
- alertent via les 2 Alertmanagers centraux

NGOT : labels dans les expressions (1/4)

Labels nécessaires

Les labels nécessaires sont :

- `obs_client` : nom du client.
- `cluster` : nom du cluster sur lequel la rule est évaluée
- `namespace` : namespace du composant en erreur

Les labels suivants sont nécessaires pour les services seulement :

- `ngot_contract` : nom du contrat
- `ngot_service` : nom du service

NGOT : labels dans les expressions (2/4)

Comment poser simplement les labels nécessaires

Normalement, les labels nécessaires doivent déjà être posés sur les métriques. Il n’y a donc rien à faire au niveau des rules.

Pour poser les labels sur les métriques, la documentation mentionne :

- pour les labels Prometheus :

```
_relabelings: #ServiceMonitorNgotRelabelings[zone.type] & {"\(zone.name)
relabelings:  _relabelings[zone.name]
```

- pour les labels K8S du PrometheusRules :

```
labels:       #ServiceMonitorNgotLabels
```

NGOT : labels dans les expressions (3/4)

Cas du label `cluster`

Le label `cluster` n'existe pas sur les Prometheus des clusters. Il est posé en tant qu'*external label*.

Conséquence:

- il est inutile de l'utiliser dans les expressions
- il ne faut pas l'utiliser avec `absent()` (sinon la fonction retourne systématiquement 1)
- bien qu'absent, il sera posé sur les rules lorsqu'elles sont envoyées à Alertmanager.

NGOT : labels dans les expressions (4/4)

Cas des rules `absent()`

La fonction `absent()` retourne 1 lorsque la métrique (avec ses labels) n'existe pas. Cette fonction sert beaucoup pour alerter lorsqu'un exporter dysfonctionne ou n'est pas déployé.

Les labels de la métrique indiquée à la fonction `absent()` doivent être spécifiés explicitement (sauf le label `cluster`).

La création de ces rules est plus délicate que les autres. Mais il existe des exemples. N'hésitez pas à demander de l'aide à l'équipe Monitoring.

Blackbox-Exporter sur NGOT (1/1)

Rules

Pré-requis : les métriques Blackbox-Exporter doivent être présentes (une présentation aura lieu ultérieurement sur ce sujet).

L'alerte `BlackboxHTTPDown` fonctionne pour toutes les métriques, il n'y a rien à configurer.

Il est conseillé d'ajouter une vérification de vos métriques. On peut s'inspirer de celles qui existent déjà dans le fichier rules/rules/blackbox-exporter-absent.cue (NGOT seulement).

Bonnes pratiques (1/2)

La documentation

- Opérations/Alerting : page d’accueil
- Opérations/Alerting : Workflow for writing Prometheus Rules
- Opérations/Alerting : Writing Prometheus Rules
- NGOT/pf-ngot/Architecture/Nomenclature : Labels Monitoring
- NGOT/pf-ngot/Applications/Prometheus : Collect metrics from new applications

Bonnes pratiques (2/2)

Ne bloquez pas vos collègues !

Les créations et modifications de rules se passent sur le repo caascad-prometheus-rules.

Lorsqu'une modification y est *mergée*, elle bloque le déploiement de toute modification ultérieure tant qu'elle n'a pas été déployée (ce qui est problématique en cas d'incident de Production).

Ne bloquez pas vos collègues !

- faites valider votre MR, ne la *mergez* **pas tout de suite** !
- préparez votre MR envs-ng et votre CAASCHR/stg
- *mergez* votre MR et réalisez votre CAASCHR/stg **le même jour** si possible
- essayez de réaliser votre CAASCHR/prd dès **le lendemain**

Conclusion

Ce qu’il faut retenir

Caascad :

- pas de changement

NGOT :

- métriques “cluster” : `cluster-ngot` et c’est tout
- métriques “services” : `service-ngot` et ajouter `prometheus-rules` et le nom de l’`apps` dans `envs.cue`

Merci pour votre attention.