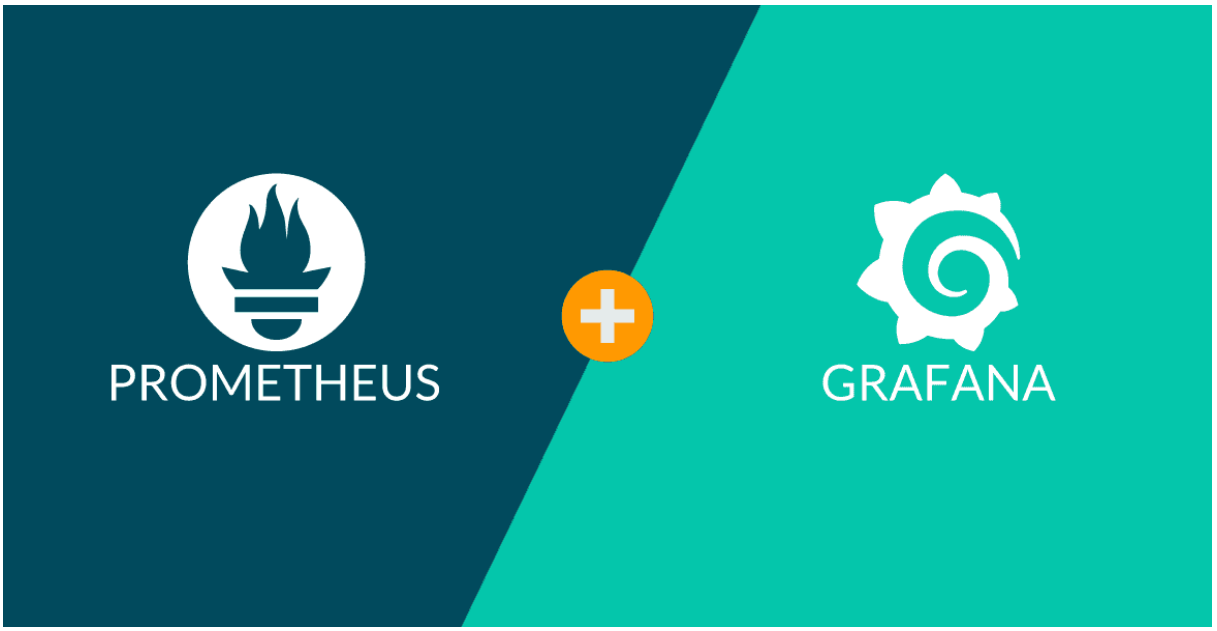

kubernetes

Manual de instalación y configuración de Prometheus y Grafana

Grupo 24

Prometheus

Antes de empezar, debemos saber los requisitos para poder instalar Prometheus y Grafana en un clúster de Kubernetes, estos son los siguientes:

- Un clúster de Kubernetes >=1.16.0 con control de acceso basado en roles (RBAC) habilitado.
- Una cuenta de Grafana Cloud.
- Una clave de API de Grafana Cloud con el rol MetricsPublisher.
- La herramienta kubectl de línea de comandos instalada en su máquina local y configurada para conectarse a su clúster.

Teniendo esto en cuenta, ya podremos iniciar instalando Prometheus Operator, para realizar esto instalaremos todas las definiciones de recursos personalizados (CRD) para Kubernetes de Prometheus Operator, estas definen las abstracciones de Prometheus, AlertManager y ServiceMonitor, las cuales son utilizadas para configurar la pila de monitoreo de Prometheus.

Para iniciar con esto, debemos instalar el Operador de Prometheus utilizando el bundle.yaml que se encuentra en el repositorio de Prometheus Operator, esto lo realizamos con el siguiente comando:

```
$ kubectl create -f https://raw.githubusercontent.com/prometheus-operator/prometheus-operator/master/bundle.yaml
```

Lo cual debería devolver el siguiente resultado:

```
customresourcedefinition.apiextensions.k8s.io/alertmanagerconfigs.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/alertmanagers.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/podmonitors.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/probes.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/prometheuses.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/prometheusrules.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/servicemonitors.monitoring.coreos.com created
customresourcedefinition.apiextensions.k8s.io/thanosrulers.monitoring.coreos.com created
clusterrolebinding.rbac.authorization.k8s.io/prometheus-operator created
clusterrole.rbac.authorization.k8s.io/prometheus-operator created
deployment.apps/prometheus-operator created
serviceaccount/prometheus-operator created
service/prometheus-operator created
```

Este proceso instalará el CRD para objetos de Prometheus, su controlador y el servicio de operador de Prometheus.

Ya teniendo esto instalado, debemos configurar los permisos RBAC de Prometheus mediante un ClusterRole y vincularemos este ClusterRole a una cuenta de servicio mediante un objeto ClusterRoleBinding.

Primero, se crea un directorio en el que se almacenarán los manifiestos de K8 utilizados:

```
$ mkdir operator_k8s
$ cd operator_k8s
```

Luego, creamos el archivo prom_rbac.yaml que debe tener la siguiente configuración:

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: prometheus
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: prometheus
rules:
- apiGroups: [""]
  resources:
  - nodes
  - nodes/metrics
  - services
  - endpoints
  - pods
  verbs: ["get", "list", "watch"]
- apiGroups: [""]
  resources:
  - configmaps
  verbs: ["get"]
- apiGroups:
  - networking.k8s.io
  resources:
  - ingresses
  verbs: ["get", "list", "watch"]
- nonResourceURLs: ["/metrics"]
  verbs: ["get"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: prometheus
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: prometheus
subjects:
- kind: ServiceAccount
  name: prometheus
  namespace: default
```

Esto crea una cuenta de servicio llamada prometheus y la vincula a prometheus ClusterRole. El manifiesto otorga los privilegios de API get de ClusterRole list y watchK8s.

Ejecutando el comando *kubectl apply -f* creará los objetos y Prometheus tendrá acceso a la API de K8, que puede ser implementado en el clúster.

Ahora crearemos el archivo prometheus.yaml para implementar Prometheus, el cual debe tener la siguiente configuración:

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
  labels:
    app: prometheus
spec:
  image: quay.io/prometheus/prometheus:v2.22.1
  nodeSelector:
    kubernetes.io/os: linux
  replicas: 2
  resources:
    requests:
      memory: 400Mi
  securityContext:
    fsGroup: 2000
    runAsNonRoot: true
    runAsUser: 1000
  serviceAccountName: prometheus
  version: v2.22.1
  serviceMonitorSelector: {}
```

De igual manera que con el archivo anterior, lo desplegamos en su clúster utilizando el comando *kubectl apply -f*, lo que implementará Prometheus para que pueda ser utilizado por el clúster y ya lo podremos exponer mediante nuestro servicio.

Ahora creamos el archivo prom_svc.yaml para poder crear un servicio de Prometheus. Este debe tener la siguiente estructura:

```
apiVersion: v1
kind: Service
metadata:
  name: prometheus
  labels:
    app: prometheus
spec:
  ports:
  - name: web
    port: 9090
    targetPort: web
  selector:
    app: prometheus
  sessionAffinity: ClientIP
~
```

También ejecutamos el comando *kubectl apply -f* para crear el servicio de Prometheus.

Realizado todo esto, ya podremos ver la siguiente pantalla en el puerto 9090:

Teniendo esto, debemos crear el archivo prometheus_servicemonitor.yaml para crear un ServiceMonitor de Prometheus, este debe tener la siguiente estructura:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: prometheus-self
  labels:
    app: prometheus
spec:
  endpoints:
  - interval: 30s
    port: web
  selector:
    matchLabels:
      app: prometheus
```

De igual manera, debemos implementarlo usando *kubectl apply -f*, además, con el comando *kubectl port-forward svc/prometheus 9090* podemos verificar los cambios y configuración reenviando un puerto al servidor Prometheus.

Grafana

Debemos crear un Kubernetes Secret para almacenar las credenciales de Grafana Cloud:

```
$ kubectl create secret generic kubepromsecret \
>    --from-literal=username=<your_grafana_cloud_prometheus_username>\
>    --from-literal=password='<your_grafana_cloud_API_key>'
```

Habilitamos el remote_write en prometheus.yaml, debe quedar con la siguiente estructura:

```
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
  labels:
    app: prometheus
spec:
  image: quay.io/prometheus/prometheus:v2.22.1
  nodeSelector:
    kubernetes.io/os: linux
  replicas: 2
  resources:
    requests:
      memory: 400Mi
  securityContext:
    fsGroup: 2000
    runAsNonRoot: true
    runAsUser: 1000
  serviceAccountName: prometheus
  version: v2.22.1
  serviceMonitorSelector: {}
remoteWrite:
  - url: "<Your Metrics instance remote_write endpoint>"
    basicAuth:
      username:
        name: kubepromsecret
        key: username
      password:
        name: kubepromsecret
        key: password
  replicaExternalLabelName: "__replica__"
  externalLabels:
    cluster: "<choose_a_prom_cluster_name>"
```

Se despliegan los cambios utilizando el comando *kubectl apply -f*.

Si accedemos a la url, podremos ver la siguiente pantalla: