Déploiement d'un DaemonSet fluentd

1. But de cet exercice

Vous allez maintenant déployer des agents *fluentd* sur l'ensemble des machines du cluster. Ces agents seront configurés pour envoyer les logs dans la stack Elastic mise en place dans l'exercice précédent.

Note: assurez-vous que la stack Elastic est toujours active, vous devrez la relancer si ce n'est pas le cas.

2. Définition des droits d'accès

Vous allez commencer par définir les ressources nécessaires pour la spécification des droits d'accès dont aura besoin l'application *fluentd* qui sera lancée dans la suite.

La spécification suivante définie 3 ressources:

- ServiceAccount: le compte que les Pods fluentd utiliseront pour effectuer des opérations sur le cluster
- ClusterRole: définit un ensemble des règles pour la lecture des Pods et des namespaces du cluster
- ClusterRoleBinding: associe le ClusterRole et le ServiceAccount et donne ainsi le droit au service account d'effectuer les actions définies par le ClusterRole

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluentd
  namespace: kube-system

---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRole
metadata:
  name: fluentd
  namespace: kube-system
rules:
- apiGroups:
  - ""
  resources:
  - pods
```

```
  - namespaces
  verbs:
  - get
  - list
  - watch

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1beta1
metadata:
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: fluentd
  namespace: kube-system
---
```

Copiez le contenu de cette spécification dans un fichier *rbac.yaml* et créez les ressources avec la commande suivante.

```
kubectl create -f rbac.yaml
serviceaccount "fluentd" created
clusterrole "fluentd" created
clusterrolebinding "fluentd" created
```

Le service account, que l'on a nommé *fluentd*, sera utilisé dans la spécification du DaemonSet comme on va le voir dans la suite.

3. Spécification du DaemonSet utilisé pour le lancement de fluentd

La spécification suivante définit un DaemonSet utilisé pour assurer qu'un Pod contenant un container *fluentd* est lancé sur chaque node du cluster. Ce container est configuré pour envoyer les entrées de logs dans l'instance *elasticsearch* qui tourne sur le cluster.

Note: *fluentd* est similaire à *logstash* en terme de fonctionnalités, il permet notamment d'ingérer des logs, de les parser, de les enrichir et d'envoyer le résultat sur des solutions tierces.

```
apiVersion: extensions/v1beta1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
    version: v1
    kubernetes.io/cluster-service: "true"
spec:
  template:
    metadata:
      labels:
        k8s-app: fluentd-logging
        version: v1
        kubernetes.io/cluster-service: "true"
    spec:
      serviceAccount: fluentd
      serviceAccountName: fluentd
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
      - name: fluentd
        image: fluent/fluentd-kubernetes-daemonset:v1.3.3-debian-elasticsearch-
1.1
        env:
          - name: FLUENT_UID
            value: "0"
          - name:  FLUENT_ELASTICSEARCH_HOST
            value: "elasticsearch.default.svc.cluster.local"
          - name:  FLUENT_ELASTICSEARCH_PORT
            value: "9200"
          - name: FLUENT_ELASTICSEARCH_SCHEME
            value: "http"
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
```

```
            path: /var/lib/docker/containers
```

Il y a plusieurs choses à noter ici:

- 2 volumes sont définis:
    - le premier a accès aux fichiers de log se trouvant dans le répertoire */var/log*
    - le second à ceux se trouvant dans */var/lib/docker/containers*
      Ces volumes étant montés dans le container *fluentd*, celui-ci aura respectivement accès aux logs systèmes et aux logs applicatifs
- la stack Elastic tourne dans le namespace *default* mais le DaemonSet est lancé dans le namespace *kube-system*, il est donc nécessaire de renseigner le FQDN de *elasticsearch* dans le cluster, *elasticsearch.default.svc.cluster.local*
- une toleration est utilisée afin de s'assurer qu'un Pod fluentd sera également schédulé sur les nodes Master. Sans cette tolération, seuls les nodes non Master auraient reçu une instance du Pod.
- la spécification du Pod contient le serviceAccount (*fluentd*) à utiliser pour le lancement

Copiez cette spécification dans le fichier *daemonset-fluentd.yaml* et utilisez la commande suivante pour lancer le DaemonSet:

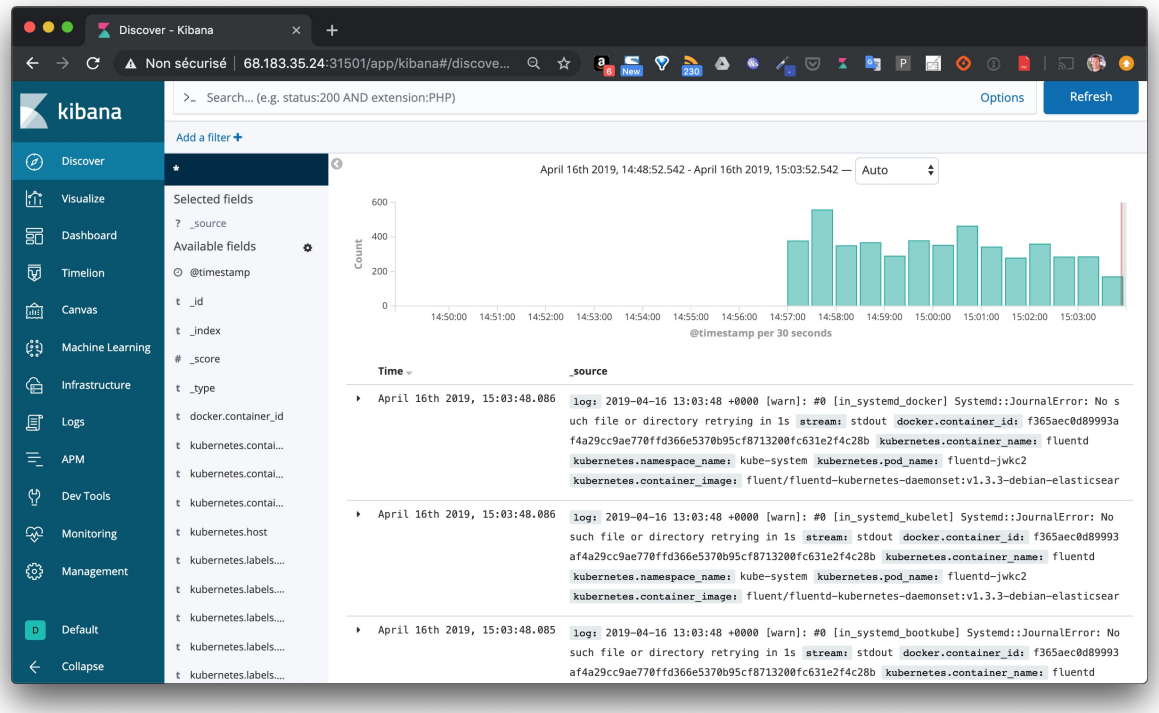
```
$ kubectl create -f daemonset-fluentd.yaml
daemonset.extensions "fluentd" created
```

Nous pouvons alors vérifier qu'un Pod a été lancé sur chaque node du cluster.

```
$ kubectl get po -n kube-system | grep "fluentd"
NAME              READY     STATUS    RESTARTS   AGE
fluentd-cfgbn     1/1       Running   0          32s
```

4. Visualisation des logs

Depuis l'interface de Kibana, vérifiez que de nouvelles entrées de logs sont reçues en permanence. Vous pourrez pour cela mettre en place le refresh automatique depuis l'interface.


Discover - Kibana
Non sécurisé | 68.183.35.24:31501/app/kibana#/discove...
kibana
Search... (e.g. status:200 AND extension:PHP)
Options
Refresh
Add a filter +
Discover
Visualize
Dashboard
Timelion
Canvas
Machine Learning
Infrastructure
Logs
APM
Dev Tools
Monitoring
Management
Default
Collapse
*
Selected fields
? _source
Available fields
@timestamp
t _id
t _index
_score
t _type
t docker.container_id
t kubernetes.contai...
t kubernetes.contai...
t kubernetes.contai...
t kubernetes.host
t kubernetes.labels....
t kubernetes.labels....
t kubernetes.labels....
t kubernetes.labels....
t kubernetes.labels....
April 16th 2019, 14:48:52.542 - April 16th 2019, 15:03:52.542 — Auto
Count
600
400
200
0
14:50:00 14:51:00 14:52:00 14:53:00 14:54:00 14:55:00 14:56:00 14:57:00 14:58:00 14:59:00 15:00:00 15:01:00 15:02:00 15:03:00
@timestamp per 30 seconds
Time
_source
April 16th 2019, 15:03:48.086
log: 2019-04-16 13:03:48 +0000 [warn]: #0 [in_systemd_docker] Systemd::JournalError: No such file or directory retrying in 1s stream: stdout docker.container_id: f365aec0d89993af4a29cc9ae770ffd366e5370b95cf8713200fc631e2f4c28b kubernetes.container_name: fluentd kubernetes.namespace_name: kube-system kubernetes.pod_name: fluentd-jwkc2 kubernetes.container_image: fluent/fluentd-kubernetes-daemonset:v1.3.3-debian-elasticsear
April 16th 2019, 15:03:48.086
log: 2019-04-16 13:03:48 +0000 [warn]: #0 [in_systemd_kubelet] Systemd::JournalError: No such file or directory retrying in 1s stream: stdout docker.container_id: f365aec0d89993af4a29cc9ae770ffd366e5370b95cf8713200fc631e2f4c28b kubernetes.container_name: fluentd kubernetes.namespace_name: kube-system kubernetes.pod_name: fluentd-jwkc2 kubernetes.container_image: fluent/fluentd-kubernetes-daemonset:v1.3.3-debian-elasticsear
April 16th 2019, 15:03:48.085
log: 2019-04-16 13:03:48 +0000 [warn]: #0 [in_systemd_bootkube] Systemd::JournalError: No such file or directory retrying in 1s stream: stdout docker.container_id: f365aec0d89993af4a29cc9ae770ffd366e5370b95cf8713200fc631e2f4c28b kubernetes.container_name: fluentd