Logging & Monitoring

» Kubernetes Monitoring

Overview

First, define a deployment and service for influxdb.

```
cat > monitoring-influxdb.yaml <<EOF
apiVersion: v1
kind: Service
metadata:
  labels:
    task: monitoring
    kubernetes.io/name: monitoring-influxdb
  name: monitoring-influxdb
  namespace: kube-system
spec:
  ports:
  - port: 8086
    targetPort: 8086
  selector:
    k8s-app: influxdb
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: monitoring-influxdb
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: influxdb
    spec:
      containers:
      - name: influxdb
        image: gcr.io/google_containers/heapster-influxdb-amd64:v1.1.1
        volumeMounts:
        - mountPath: /data
          name: influxdb-storage
      volumes:
      - name: influxdb-storage
        emptyDir: {}
EOF
```

Next, define a deployment and service for heapster.

```
cat > monitoring-heapster.yaml <<EOF
apiVersion: v1
kind: Service
metadata:
  labels:
    task: monitoring
    kubernetes.io/name: Heapster
  name: monitoring-heapster
  namespace: kube-system
spec:
  ports:
  - port: 80
    targetPort: 8082
  selector:
    k8s-app: heapster
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: monitoring-heapster
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: heapster
    spec:
      containers:
      - name: heapster
        image: gcr.io/google_containers/heapster-amd64:v1.3.0-beta.1
        imagePullPolicy: IfNotPresent
        command:
        - /heapster
        - --source=kubernetes:https://kubernetes.default
        - --sink=influxdb:http://monitoring-influxdb:8086
EOF
```

Next, define a deployment and service for grafana.

```
cat > monitoring-grafana.yaml <<EOF
apiVersion: v1
kind: Service
metadata:
  labels:
    kubernetes.io/name: monitoring-grafana
  name: monitoring-grafana
  namespace: kube-system
spec:
  # In a production setup, we recommend accessing Grafana through an external Loadbalancer
  # or through a public IP.
  # type: LoadBalancer
  # You could also use NodePort to expose the service at a randomly-generated port
  # type: NodePort
  ports:
  - port: 80
    targetPort: 3000
  selector:
    k8s-app: grafana
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: monitoring-grafana
  namespace: kube-system
spec:
  replicas: 1
  template:
    metadata:
      labels:
        task: monitoring
        k8s-app: grafana
    spec:
      containers:
      - name: grafana
        image: gcr.io/google_containers/heapster-grafana-amd64:v4.0.2
        ports:
          - containerPort: 3000
            protocol: TCP
        volumeMounts:
        - mountPath: /var
          name: grafana-storage
        env:
        - name: INFLUXDB_HOST
```

```
            name: INFLUXDB_HOST
            value: monitoring-influxdb
          - name: GRAFANA_PORT
            value: "3000"
            # The following env variables are required to make Grafana accessible via
            # the kubernetes api-server proxy. On production clusters, we recommend
            # removing these env variables, setup auth for grafana, and expose the grafana
            # service using a LoadBalancer or a public IP.
          - name: GF_AUTH_BASIC_ENABLED
            value: "false"
          - name: GF_AUTH_ANONYMOUS_ENABLED
            value: "true"
          - name: GF_AUTH_ANONYMOUS_ORG_ROLE
            value: Admin
          - name: GF_SERVER_ROOT_URL
            # If you're only using the API Server proxy, set this value instead:
            # value: /api/v1/proxy/namespaces/kube-system/services/monitoring-grafana/
            value: /
        volumes:
        - name: grafana-storage
          emptyDir: {}
EOF
```

Create all of the monitoring resources.

```
kubectl create -f monitoring-influxdb.yaml
kubectl create -f monitoring-heapster.yaml
kubectl create -f monitoring-grafana.yaml
```

Save the hostname for the lab cluster into a shell variable for easy use. Remember to replace `labXX` with your actual lab number.

```
export CLUSTER=labXX.coreostrain.me
echo $CLUSTER
```

Obtain one of the IP addresses for the lab cluster load balancer.

```
host $CLUSTER
```

Save an IP address returned into a shell variable.

```
export ELB_IP=`host $CLUSTER | awk 'NR==1{print $4}'`
echo $ELB_IP
```

Define an ingress resource to the grafana service.

```
cat >> monitoring-grafana-ingress.yaml <<EOF
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  namespace: kube-system
  name: monitoring-grafana-ingress
  annotations:
    kubernetes.io/ingress.class: "tectonic"
spec:
  rules:
    - host: grafana.$ELB_IP.xip.io
      http:
        paths:
          - path: /
            backend:
              serviceName: monitoring-grafana
              servicePort: 80
EOF
```

Create the ingress object.

```
kubectl create -f monitoring-grafana-ingress.yaml
```

The Grafana dashboard should be available at http://grafana.ELB_IP.xip.io (http://grafana.ELB_IP.xip.io).