

Sesión 16 prometheus on kubernetes

- https://linuxhint.com/kubernetes_operator/

google cloud

- `gcloud config set compute/zone europe-west1-b` ←
(<https://cloud.google.com/compute/docs/regions-zones/>)
- `gcloud container clusters create bootcamp --num-nodes 3 --scopes «https://www.googleapis.com/auth/projecthosting,storage-rw»`
- `gcloud container clusters list`
- en local, con las utilidades G instaladas: `gcloud container clusters get-credentials bootcamp --zone europe-west1-b --project <project>`
 - <https://cloud.google.com/sdk/docs/quickstart-linux>
- y para eliminar después el cluster: `gcloud container clusters delete bootcamp`

helm

- <https://medium.com/google-cloud/helm-on-gke-cluster-quick-hands-on-guide-ecffad94b0>
- `k apply -f create-helm-service-account.yaml`

[create-helm-service-account.yaml](#)

```
# This is an extract from here:
http://jayunit100.blogspot.fi/2017/07/helm-on.html
apiVersion: v1
kind: ServiceAccount
metadata:
  name: helm
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: helm
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: helm
  namespace: kube-system
```

- obtener password: `gcloud container clusters describe bootcamp --zone europe-west1-b --<project_id>`
 - igual no necesario
- inicializamos helm `helm init --service-account helm`
- `k get deploy,svc tiller-deploy -n kube-system`
 - tiller: parte servidor de helm, con la que interactuamos

- creamos para deployar: `helm create samplechart`
- deployamos: `helm install --name helm-test ./samplechart --set service.type=LoadBalancer`
- vemos el deploy: `k get svc`
- comprobación de nuestro NGINX desplegado: `curl <ip_publica>`

prometheus

- <https://itnext.io/kubernetes-monitoring-with-prometheus-in-15-minutes-8e54d1de2e13>
- instalación a través de helm: `helm install stable/prometheus-operator --name prometheus-operator --namespace monitoring`
- verificamos: `kubectl get pods -n monitoring`
- prometheus: `kubectl port-forward -n monitoring prometheus-prometheus-operator-prometheus-0 9090 → http://localhost:9090`
- grafana: `kubectl port-forward $(kubectl get pods --selector=app=grafana -n monitoring --output=jsonpath={.items..metadata.name}) -n monitoring 3000 → http://localhost:3000` (admin:prom-operator)
- alertmanager: `kubectl port-forward -n monitoring alertmanager-prometheus-operator-alertmanager-0 9093 → http://localhost:9093`

deploy traefik

para usar con el deploy del WHO (necesitamos la IP asignada por el uso de **nip.io**)

[traefik.yaml](#)

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
  labels:
    k8s-app: traefik-ingress-lb
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: traefik-ingress-lb
  template:
    metadata:
      labels:
        k8s-app: traefik-ingress-lb
        name: traefik-ingress-lb
    spec:
      serviceAccountName: traefik-ingress-controller
      terminationGracePeriodSeconds: 60
```

```

    containers:
      - image: traefik
        name: traefik-ingress-lb
        ports:
          - name: http
            containerPort: 80
          - name: admin
            containerPort: 8080
        args:
          - --api
          - --kubernetes
          - --logLevel=INFO
          - --metrics
          - --metrics.prometheus
          - --web
          - --web.metrics
          - --web.metrics.prometheus
    ---
kind: Service
apiVersion: v1
metadata:
  name: traefik-ingress-service
  namespace: kube-system
spec:
  selector:
    k8s-app: traefik-ingress-lb
  ports:
    - protocol: TCP
      port: 80
      name: web
    - protocol: TCP
      port: 8080
      name: admin
  type: LoadBalancer

```

rbac.yaml

```

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

```

```

rules:
  - apiGroups:
      - ""
    resources:
      - services
      - endpoints
      - secrets
    verbs:
      - get
      - list
      - watch
  - apiGroups:
      - extensions
    resources:
      - ingresses
    verbs:
      - get
      - list
      - watch

```

- nos asignamos admin para poder hacer el rbac: `kubectl create clusterrolebinding cluster-role-user --clusterrole=cluster-admin --user=<user_email_gcloud>`
- `k deploy -f rbac.yaml -n kube-system`
- `k deploy -f traefik.yaml -n kube-system`
- `k get svc -n kube-system` : obtener IP pública → <http://<ip>:8080> (panel admin)
 - <http://<ip>:8080/metrics>

deploy WHO

- `k create namespace who`
- `k apply -f whoami-deployment.yaml -n who`
- `k apply -f whoami-ingress.yaml -n who`
- `k apply -f whoami-service.yaml -n who`
- `k apply -f whoareyou.yaml -n who`
- `k get svc,deply -n who`

otro

[otro-who.yaml](#)

```

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: whoami-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: whoami
  template:
    metadata:
      labels:

```

```

    app: whoami
  spec:
    containers:
      - name: whoami-container
        image: containous/whoami
    ---
  apiVersion: v1
  kind: Service
  metadata:
    name: whoami-service
  spec:
    ports:
      - name: http
        targetPort: 80
        port: 80
    selector:
      app: whoami
    ---
  apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    name: whoami-ingress
    annotations:
      kubernetes.io/ingress.class: traefik
  spec:
    rules:
      - host: whoami.localhost
        http:
          paths:
            - path: /
              backend:
                serviceName: whoami-service
                servicePort: http

```

añadir ingress para grafana de prometheus

grafana.yaml

```

  apiVersion: extensions/v1beta1
  kind: Ingress
  metadata:
    name: prometheus-operator-grafana
    namespace: monitoring
    annotations:
      kubernetes.io/ingress.class: traefik
  spec:
    rules:
      - host: grafana-35-187-100-153.nip.io
        http:
          paths:
            - path: /
              backend:
                serviceName: prometheus-operator-grafana

```

`servicePort: 80`

- `k apply -f grafana.yaml [-n monitoring]`

otros

- cloudflare : anti-DDOS + CDN
- confluence (10\$ por 10 users on-premise)
- docker system df

From:

<https://matebcn.eu/> - miguel angel torres egea

Permanent link:

<https://matebcn.eu/info:cursos:pue:devops:sesion16>

Last update: **07/04/2019 08:18**

