Sesión 12 : k8s

k8s

lab (traefik)

- traefik como ingress controller:

traefik-rbac.yaml

```yaml
---
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
---
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
```

traekik-daemonset.yaml

```
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
---
kind: DaemonSet
apiVersion: extensions/v1beta1
metadata:
  name: traefik-ingress-controller
  namespace: kube-system
  labels:
    k8s-app: traefik-ingress-lb
spec:
  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
          hostPort: 80
        - name: admin
          containerPort: 8080
          hostPort: 8080
        securityContext:
          capabilities:
            drop:
            - ALL
            add:
            - NET_BIND_SERVICE
        args:
        - --api
        - --kubernetes
        - --logLevel=INFO
---
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
```

esta versión no se ha desplegado correctamente VS la anterior

traefik-deployment.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
---
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: NodePort
```

lab (who)

https://medium.com/@geraldcroes/kubernetes-traefik-101-when-simplicity-matters-957eeede2cf8

whoami-deployment.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
```

whoami-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: whoami-service
spec:
  ports:
  - name: http
    targetPort: 80
    port: 80
  selector:
    app: whoami
```

whoami-ingress.yaml

```
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
```

si queremos NO tocar el localhost para que resuelva **whoami.localhost**, podemos lanzar este comando: `curl -H 'Host: whoami.localhost' 192.168.99.100`

también podemos usar http://nip.io:

whoami-ingress.yaml

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: whoami-ingress
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: whoami-192-168-99-100.nip.io
    http:
      paths:
      - path: /
        backend:
          serviceName: whoami-service
          servicePort: http
```

whoareyou-deployment.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: whoareyou-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: whoareyou
  template:
    metadata:
      labels:
        app: whoareyou
```

```
    spec:
      containers:
      - name: whoareyou-container
        image: containous/whoami
---
apiVersion: v1
kind: Service
metadata:
  name: whoareyou-service
spec:
  ports:
  - name: http
    targetPort: 80
    port: 80
  selector:
    app: whoareyou
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: whoareyou-ingress
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: whoareyou-192-168-99-100.nip.io
    http:
      paths:
      - path: /
        backend:
          serviceName: whoareyou-service
          servicePort: http
```

lab (flocker)

- `minukube dashboard` : lanza proceso dashboard web

instalar (sin flocker) un WP y MYSQL

- mysql:

  mysql-secrets.yaml

  ```
  apiVersion: v1
  kind: Secret
  metadata:
    name: mysql-secret
  type: Opaque
  data:
    password: cm9vdA==
  ```

mysql.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  ports:
    - port: 3306
  selector:
    app: wordpress
    tier: mysql
  clusterIP: None
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress-mysql
  labels:
    app: wordpress
spec:
  selector:
    matchLabels:
      app: wordpress
      tier: mysql
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: mysql
    spec:
      containers:
      - image: mysql:5.6
        name: mysql
        env:
        - name: MYSQL_ROOT_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret
```

```
              key: password
        ports:
        - containerPort: 3306
          name: mysql
        volumeMounts:
        - name: mysql-persistent-storage
          mountPath: /var/lib/mysql
      volumes:
      - name: mysql-persistent-storage
        persistentVolumeClaim:
          claimName: mysql-pv-claim
```

wordpress.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: wordpress-service
  labels:
    app: wordpress
spec:
  ports:
  - name: http
    targetPort: 80
    port: 80
# - port: 80
  selector:
    app: wordpress
    tier: frontend
#  type: LoadBalancer
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wordpress
  labels:
    app: wordpress
spec:
#  replicas: 1
  selector:
    matchLabels:
```

```
      app: wordpress
      tier: frontend
#  strategy:
#    type: Recreate
  template:
    metadata:
      labels:
        app: wordpress
        tier: frontend
    spec:
      containers:
      - image: wordpress:4.8-apache
        name: wordpress
        env:
        - name: WORDPRESS_DB_HOST
          value: wordpress-mysql
        - name: WORDPRESS_DB_PASSWORD
          valueFrom:
            secretKeyRef:
              name: mysql-secret
              key: password
        ports:
        - containerPort: 80
          name: wordpress
        volumeMounts:
        - name: wordpress-persistent-storage
          mountPath: /var/www/html
      volumes:
      - name: wordpress-persistent-storage
        persistentVolumeClaim:
          claimName: wp-pv-claim
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: wordpress-ingress
  annotations:
    kubernetes.io/ingress.class: traefik
spec:
  rules:
  - host: wordpress-192-168-99-100.nip.io
    http:
      paths:
      - path: /
        backend:
          serviceName: wordpress-service
          servicePort: http
```

- kubectl edit <objeto>
- kubectl scale –replicas=3 deployment/<pod>
- kubectl autscale –min=3 –max=10 –cpu-percent=2 deplyment/<pod>
- kubectl get all –all-namespaces
- kubetel get all –namespace=<namespace>
    - lo que no se muestra:
        - kubectl get pv

        - `kubectl get pvc`
        - `kubectl get ingresses`
- `kubectl api-resources`

flocker

- instalación: https://flocker.readthedocs.io/en/latest/kubernetes-integration/ ← fails!
- https://github.com/linux-on-ibm-z/docs/wiki/Building-Flocker

objetos kubernetes

- replicationcontroller: https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/
- replicaset: https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/
- deployment: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
    - rollout, concepto de log de revisiones de deployment
    - sección deployment, sección replicaset, sección pod
- daemonset: https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/
- statefulsets: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/
    - storageclassname : servicio de gestión de volúmenes
- persistentvolumes: https://kubernetes.io/docs/concepts/storage/persistent-volumes/
    - modos:
        - readWriteOnce : solo un POD puede usar ese volumen para lectura/escritura
        - readOnlyMany
        - readWriteMany
    - persistentVolumeProvisioning: https://github.com/kubernetes/examples/blob/master/staging/persistent-volume-provisioning/README.md
    - storage classes: https://kubernetes.io/docs/concepts/storage/storage-classes/
- secrets: https://kubernetes.io/docs/concepts/configuration/secret/
- labels: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/

otros

- `helm fetch <>` : descarga