Kubernetes ++

Prof. Lic. R. Alejandro Mansilla
Prof. Ing. Sebastián Cardello

Licenciatura en Ciencias de la Computación
Facultad de Ingeniería
Universidad Nacional de Cuyo

Kubernetes: más allá de los PODs

- **Deployments**
  - arreando pods ...
- **Services**
  - cómo llego a mis pods ?
- **ConfigMaps**
  - cómo configuro mis containers (dentro de los pods) ?
- **Secrets**
  - cómo proveo configuración de secretos
- **StatefulSets**, **DaemonSets**
  - deployments + volúmenes persistentes
- **Ingress**
  - cómo llego a mis pods via *http virtualhost* ?

Kubernetes object: `Deployment`

- Objeto de más "alto" nivel, controla *Pods* via *ReplicaSets*
- Tiene estrategia de rollout:

```
kubectl get deployment web -oyaml | grep -A3 strategy:
```

  - `maxSurge`: cuánto se puede "inflar" (en pods) el deploy durante rollout
  - `maxUnavailable`: cuántos pods pueden estar *down* durante rollout
- Objecto intermedio: ReplicaSet
  - observar el *output* del último `kubectl get deploy,rs,pod`
- Exploremos un poco:

```
kubectl get pod
kubectl get pod web-xxxxxxxx-yyyyy
kubectl logs web-xxxxxxxx-yyyyy
kubectl exec -it web-xxxxxxxx-yyyyy -- /bin/bash
```

- Ahora ... ¿ Cómo lo accedemos ?

```
kubectl get pod
kubectl port-forward web-xxxxxxxx-yyyyy 8080:8080
curl http://localhost:8080/ # o via browser
```

  - Todo ésto ^^^ **no sirve** para production, ¿ Por qué ?

Kubernetes object: `Service`

- Son efectivamente *loadbalancers internos* de Kubernetes
  - provistos por la plataforma, via CNI

- Típicamente a cargo del `kube-proxy` que corre en cada nodo
- Hay varios tipos, de acuerdo a su *reachability*:
  - **`ClusterIP`** (default)
    - accesible desde **dentro** del cluster (nodos y pods)
      - Típicamente usado para conectar micro-services internos, o necesario para →
  - **`NodePort`**
    - ClusterIP + **ports dedicados en cada nodo worker**
      - Por ejemplo para apuntar un *loadbalancer* externo a todos los nodos, mismo port
  - **`LoadBalancer`**
    - NodePort + creación automática de LB provisto por el Cloud Provider

Kubernetes object: `ConfigMap`

- Es un objeto que contiene un conjunto de key: value, en su campo data
- Fundamentalmente usado para configurar los parámetros runtime de las aplicaciones
- Se puede "consumir" como:
  - Archivos mapeados en en container (key: filename, value: contenido)
  - Environmental variables
  - Command line arguments $(ENV_VAR)

Kubernetes object: `Secret`

- **Objetivo: Arrancar un motor de base de datos** [usamos https://hub.docker.com/_/mysql/]

```
git clone git@github.com:FING-Sistemas-Distribuidos-2022/k8s-mas-mas.git

cd k8s-mas-mas/

kubectl apply -f 01-db-simple.deploy.yaml
```

- ¿ Qué ocurrió ?

```
kubectl get pod -owide
kubectl logs db-simple-xxxxxxxxxx-yyyyy
```

Un secreto es un objeto que contiene una pequeña cantidad de datos confidenciales, como una contraseña, un token o una clave. [**12f++**: no debe quedar información sensitiva dentro de las imágenes/repos] *Solución: "volúmenes" tipo secreto.*

```
kubectl create secret generic db-secret --from-literal=db-username=root
--from-literal=db-password=secretisimo --from-literal=db-name=whois -o yaml --dry-run >
02-db-prod.secret.yaml


kubectl apply -f 02-db-prod.secret.yaml
kubectl apply -f 03-db-prod.deploy.yaml
```

- ¿ Qué ocurrió ?

```
kubectl get deploy,secret,pod -owide
kubectl exec -it db-prod-xxxxxxxxxx-yyyyy -- /bin/bash
```

p04- Kubernetes object: `Secret & ConfigMap`

- **Objetivo: Arrancar un motor de base de datos con su SCHEMA ya creado.**

  Creamos un "secreto" con el código sql para crear el SCHEMA de la DB y luego Deployamos

```
kubectl create cm db-sql --from-file=schema.sql -o yaml --dry-run > 04-db-prod-schema.cm.yaml

kubectl apply -f 04-db-prod-schema.cm.yaml
kubectl apply -f 05-db-prod-schema.deploy.yaml
```

  * Este deploy usa los secrets de `02-db-prod.secret.yaml`

- ¿ Qué ocurrió ?

```
kubectl get deploy,secret,pod -owide
kubectl exec -it db-prod-schema-xxxxxxxxxx-yyyyy -- /bin/bash
```

- ¿ Le metemos data ?

```
cat data.sql | kubectl exec db-prod-schema-xxxxxxxxxx-yyyyy -it -- mysql -u root --password=secretisimo whois
```

- ¿ Hay data ?

```
echo 'select cidr,json_extract(whois,"$.network.name") from whois;' | kubectl exec db-prod-schema-xxxxxxxx-yyyyy -it -- mysql -r -s -u root --password=secretisimo whois
```