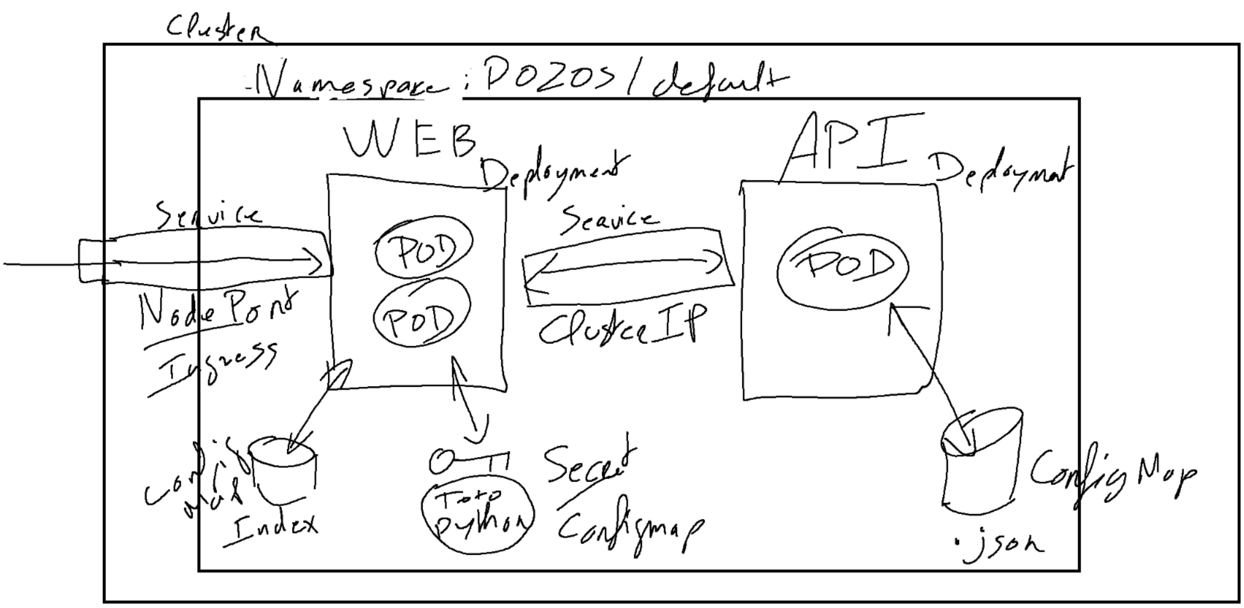
Project - Student List Scale

Introduction

Faisons un état des lieux des ressources nécessaires:

Besoin	Ressource Kubernetes
API	Deployment
Website	Deployment / Replica 2
Connection Website <-> Api	Service : ClusterIP
Connection au Website depuis l'extérieur	Service : **NodePort** ou Ingress
Api -> Fichier student.json	ConfigMap
Website -> Fichier index.php	ConfigMap
Website credentials (User/Pass)	**Secret** ou ConfigMap



by Grp2 DAZ

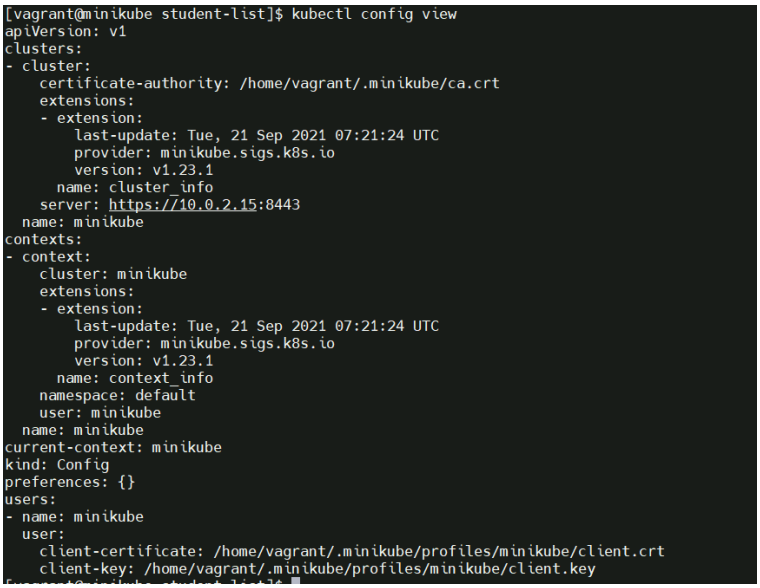
Namespace

Pour plus d’isolation, nous avons décidé de créer un nouveau namespace “pozos” pour accueillir notre application:

```
apiVersion: v1
kind: Namespace
metadata:
  name: pozos
```

Pour entrer dans ce namespace, il faut changer le contexte d'exécution. Pour cela, affichons déjà la configuration actuelle.

```
kubectl config view
```



Le contexte est donc dans le namespace ”default”.

Pour changer:

```
kubectl config set-context --current --namespace=pozos
```

```
[vagrant@minikube student-list]$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/vagrant/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Tue, 21 Sep 2021 07:21:24 UTC
        provider: minikube.sigs.k8s.io
        version: v1.23.1
      name: cluster_info
    server: https://10.0.2.15:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Tue, 21 Sep 2021 07:21:24 UTC
        provider: minikube.sigs.k8s.io
        version: v1.23.1
      name: context_info
    namespace: pozos
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/vagrant/.minikube/profiles/minikube/client.crt
    client-key: /home/vagrant/.minikube/profiles/minikube/client.key
```

Configuration

Nous avons besoin de 2 déploiements: un pour L'API, et un pour le website. Mais ceux-ci auront besoin de fichier de configuration, ou de crédential.
Pour cela, nous allons faire appel à deux objets: **ConfigMap**, et **Secret**

ConfigMap

Nous avons besoin de deux configmap.
Pour cela, nous devons "peupler" les configmaps depuis nos fichiers.

Pour la configmap de l'API pour le fichier student_age.json:

```
kubectl create configmap student-config --from-file=./simple_api/student_age.json
```

Pour avoir le résultat:

```
kubectl get configmaps student-config -oyaml
```

```
[vagrant@minikube student-list]$ kubectl get configmaps student-config -oyaml
apiVersion: v1
data:
  student_age.json: |
    {
        "bob": "13",
        "alice": "12"
    }
kind: ConfigMap
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"student_age.json":"{\n    \"bob\": \"13\",\n    \"al
ent-config","namespace":"pozos"}}
  creationTimestamp: "2021-09-21T12:07:04Z"
  name: student-config
  namespace: pozos
  resourceVersion: "12819"
  uid: a155ce02-953e-4d53-8fa1-7e3f838f7fd1
```

Pour celle du website pour l'index.php

```
kubectl create configmap index-php --from-file=./website/index.php
```

Pour avoir le résultat:

```
kubectl get configmaps index-php -oyaml
```

Ensuite, nous pouvons récupérer ces configuration, enlever les informations “superflues” et ça nous donne:

```
apiVersion: v1
data:
  student_age.json: |
    {
        "bob": "13",
        "alice": "12"
    }
kind: ConfigMap
metadata:
  name: student-config
  namespace: pozos
---
apiVersion: v1
data:
  index.php: |2

    <html>
        <head>
            <title>POZOS</title>
        </head>

        <body>
            <h1>Student Checking App</h1>
            <ul>
                <form action="" method="POST">
                <!--<label>Enter student name:</label><br />
                <input type="text" name="" placeholder="Student Name" required/>
                <br /><br />-->
                <button type="submit" name="submit">List Student</button>
                </form>

                <?php
                  if($_SERVER['REQUEST_METHOD'] == "POST" and isset($_POST['submit']))
                  {
                  $username = getenv('USERNAME');
                  $password = getenv('PASSWORD');
                  if ( empty($username) ) $username = 'fake_username';
                  if ( empty($password) ) $password = 'fake_password';
                  $context = stream_context_create(array(
                    "http" => array(
                    "header" => "Authorization: Basic " . base64_encode("$username:$password"),
                  )));

                  $url = 'http://api:5000/pozos/api/v1.0/get_student_ages';
                  $list = json_decode(file_get_contents($url, false, $context), true);
                  echo "<p style='color:red;; font-size: 20px;'>This is the list of the student with age</p>";
                  foreach($list["student_ages"] as $key => $value) {
                      echo "- $key are $value years old <br>";
                  }
                 }
                ?>
            </ul>
        </body>
    </html>
kind: ConfigMap
metadata:
  name: index-php
  namespace: pozos
```

Secret

Un secret un objet similaire au configmap, mais destiné au donnée sensible. Pour nos USERNAME et PASSWORD, il va donc de soit que c'est la meilleur option.
Pour créer notre secret:

```
kubectl create secret generic student-creds --from-literal=USERNAME='toto' --from-literal=PASSWORD='python'
```

Pour avoir le résultat:

```
kubectl get secret student-creds -oyaml
```

```
[vagrant@minikube student-list]$ kubectl get secret student-creds -oyaml
apiVersion: v1
data:
  PASSWORD: cHl0aG9u
  USERNAME: dG90bw==
kind: Secret
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","data":{"PASSWORD":"cHl0aG9u","USERNAME":"dG90bw=="},"
"type":"Opaque"}
  creationTimestamp: "2021-09-21T12:07:04Z"
  name: student-creds
  namespace: pozos
  resourceVersion: "12832"
  uid: 9ed2055b-ea51-431c-9b5a-be4b1c9defc5
type: Opaque
```

Nous voyons bien que les données ne sont pas en clair. Mais attention, elles ne sont pas pour autant "ultra-sécurisées". En effet, un simple decode en base 64 suffit à dévoiler le pot aux roses:

```
echo 'cHl0aG9u' | base64 -d
```

```
[vagrant@minikube student-list]$ echo "cHl0aG9u" | base64 -d
python[vagrant@minikube student-list]$
```

En nettoyant un peu la sortie du get secret, nous avons donc:

```
apiVersion: v1
data:
  PASSWORD: cHl0aG9u
  USERNAME: dG90bw==
kind: Secret
metadata:
  name: student-creds
  namespace: pozos
type: Opaque
```

Deployments

Pour nos déploiement, il nous faut donc préparer deux objets, poser les bon labels pour être exploité après par les services, et conforme au spécificité de l'application (ports, nombres de réplicats etc).
Ensuite, nous avons besoin de monter les configmaps en temps que volumes, et le secret en temps que variable d'environnement:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: api
  name: api
  namespace: pozos
spec:
  selector:
    matchLabels:
      app: api
  template:
    metadata:
      labels:
        app: api
    spec:
      containers:
      - image: localhost:8080/student_age:latest
        name: api
        ports:
        - containerPort: 5000
        volumeMounts:
        - mountPath: /data
          name: student-list
      volumes:
      - configMap:
          name: student-config
        name: student-list
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: website
  namespace: pozos
  labels:
    app: website
spec:
  replicas: 2
  selector:
    matchLabels:
      app: website
  template:
    metadata:
      labels:
        app: website
    spec:
      containers:
      - name: website
        image: php:apache
        envFrom:
        - secretRef:
            name: student-creds
        ports:
        - containerPort: 80
        volumeMounts:
          - name: index-file
            mountPath: /var/www/html
      volumes:
      - name: index-file
        configMap:
          name: index-php
```

```
[vagrant@minikube student-list]$ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
api       1/1     1            1           23m
website   2/2     2            2           23m
```

Service

Nous avons maintenant nos déploiements qui font tourner nos applications dans des pods, mais ils ne communiquent pas encore entre eux, et nous ne pouvons pas encore y accéder depuis l'extérieur. Pour cela, nous devons configurer les Services:

- Pour l'API, nous aurons juste besoin de connexion interne, un ClusterIP suffira
- Pour le website, nous avons besoin d'une connexion extérieure. Il faudra donc choisir un type adéquat, dans notre cas, un NodePort

```
apiVersion: v1
kind: Service
metadata:
  name: api
  namespace: pozos
spec:
  type: ClusterIP
  selector:
    app: api
  ports:
    - protocol: TCP
      port: 5000
      targetPort: 5000
---
apiVersion: v1
kind: Service
metadata:
  name: website
  namespace: pozos
spec:
  type: NodePort
  selector:
    app: website
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

```
[vagrant@minikube student-list]$ kubectl get service
NAME      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
api       ClusterIP   10.107.148.220   <none>        5000/TCP       37s
website   NodePort    10.110.199.180   <none>        80:30402/TCP   37s
```

Conclusion

Une fois le manifeste construit avec tous ces objets, il nous suffit de l'appliquer:

```
kubectl apply -f manifest.yaml
```

```
[vagrant@minikube student-list]$ kubectl apply -f manifest.yaml
namespace/pozos created
configmap/student-config created
configmap/index-php created
secret/student-creds created
deployment.apps/api created
deployment.apps/website created
service/api created
service/website created
```

```
api       ClusterIP   10.107.148.220   <none>        5000/TCP       26s
website   NodePort    10.110.199.180   <none>        80:30402/TCP   26s
[vagrant@minikube student-list]$ ^C
[vagrant@minikube student-list]$ kubectl get service
NAME      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
api       ClusterIP   10.107.148.220   <none>        5000/TCP       37s
website   NodePort    10.110.199.180   <none>        80:30402/TCP   37s
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
[vagrant@minikube student-list]$
```