1. Микросервисная архитектура и контейнеризация

1.1. Урок 5. Сетевые абстракции Kubernetes

https://github.com/adterskov/geekbrains-conteinerization/tree/master/homework/5.kubernetes-network

Создаем пространство имён:

```
kubectl create ns gb-devops-conteiner-hw05
kubectl config set-context --current --namespace=gb-devops-conteiner-hw05
```

Создаем PVC для базы `PersistentVolumeClaim-DB-HW04.yaml`:

```yaml
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: postgresql-storage
  namespace: gb-devops-conteiner-hw05
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: "csi-ceph-ssd-gz1"
```

И разворачиваем командой `kubectl apply -f PersistentVolumeClaim-DB-HW04.yaml`

Создаем секрет для хранения пароля от базы:

```
kubectl create secret generic postgresql-secret --from-literal=PASS=mysecretpass
```

Подготавлдиваем деплой и разворачиваем базу:

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgresql-db
  namespace: gb-devops-conteiner-hw05
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgresql-db
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: postgresql-db
    spec:
      initContainers:
      - image: busybox
        name: mount-permissions-fix
        command: ["sh", "-c", "chmod 777 /var/lib/postgresql/data"]
        volumeMounts:
        - name: data
          mountPath: /var/lib/postgresql/data
      containers:
      - image: postgres:10.13
        name: postgres
        env:
        - name: POSTGRES_USER
          value: "db_user"
        - name: POSTGRES_DB
          value: "db_database"
        - name: PGDATA
          value: "/var/lib/postgresql/data/pgdata"
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: postgresql-secret
              key: PASS
        ports:
        - containerPort: 5432
          protocol: TCP
        resources:
          requests:
            cpu: 100m
            memory: 1Gi
          limits:
```

```
            cpu: 100m
            memory: 1Gi
        volumeMounts:
        - name: data
          mountPath: /var/lib/postgresql/data
      volumes:
      - name: data
        persistentVolumeClaim:
          claimName: postgresql-storage
```

```
kubectl apply -f Deployment-DB-HW04.yaml
```

POD поднялся:

```
PS C:\Repos\GB-DevOps-Conteinerization\HW05> kubectl apply -f Deployment-DB-HW04.yaml
deployment.apps/postgresql-db created
PS C:\Repos\GB-DevOps-Conteinerization\HW05> kubectl get po
NAME                             READY   STATUS    RESTARTS   AGE
postgresql-db-5d7b5575df-vf26k   1/1     Running   0          17s
PS C:\Repos\GB-DevOps-Conteinerization\HW05>
```

Создаем и разворачиваем сервис для доступа к базе из других контейнеров:

```
---
apiVersion: v1
kind: Service
metadata:
  name: postgresql-service
  namespace: gb-devops-conteiner-hw05
spec:
  ports:
  - port: 5432
    targetPort: 5432
  selector:
    app: postgresql-db
  type: ClusterIP
```

```
kubectl apply -f Service-Postgre.yaml
```

Создаем секрет для хранения пароля от Redmine:

```
kubectl create secret generic redmine-secret --from-literal=KEY=myverysecretkey
```

Подготавлдиваем деплой и разворачиваем Redmine:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: redmine-app
  namespace: gb-devops-conteiner-hw05
spec:
  replicas: 1
  selector:
    matchLabels:
      app: redmine-app
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: redmine-app
    spec:
      containers:
      - image: redmine:4.1.1
        name: redmine
        env:
          - name: REDMINE_DB_POSTGRES
            value: "postgresql-service"
          - name: REDMINE_DB_USERNAME
            value: "db_user"
          - name: REDMINE_DB_DATABASE
            value: "db_database"
          - name: REDMINE_DB_PASSWORD
            valueFrom:
              secretKeyRef:
                name: postgresql-secret
                key: PASS
          - name: REDMINE_SECRET_KEY_BASE
            valueFrom:
              secretKeyRef:
                name: redmine-secret
                key: KEY
        ports:
        - containerPort: 3000
```

```
kubectl apply -f Deployment-Redmine.yaml
```

Создаем и разворачиваем сервис для доступа к Redmine из других контейнеров:

```
---
apiVersion: v1
kind: Service
metadata:
  name: redmine-service
  namespace: gb-devops-conteiner-hw05
spec:
  ports:
  - port: 80
    targetPort: 3000
  selector:
    app: redmine-app
  type: ClusterIP
```

```
kubectl apply -f Service-Redmine.yamlПодготавлдиваем и разворачиваем конфигурацию
Ingress:
```

```
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: redmine-ingress
  namespace: gb-devops-conteiner-hw05
  annotations:
    kubernetes.io/ingress.class: nginx
spec:
  rules:
  - http:
      paths:
      - path: "/"
        pathType: Prefix
        backend:
          service:
            name: redmine-service
            port:
              number: 80vicePort: 80
```

```
kubectl apply -f Ingress.yaml
```

Смотрим статус PODов, все ли поднялось:

```
kubectl get po
kubectl get svc -o wide
```

```
PS C:\Repos\GB-DevOps-Conteinerization\HW05> kubectl get po
NAME                          READY   STATUS    RESTARTS   AGE
postgresql-db-5d7b5575df-vf26k   1/1     Running   0          23m
redmine-app-5757557b9f-n4rp6     1/1     Running   0          18m
PS C:\Repos\GB-DevOps-Conteinerization\HW05> kubectl get svc -o wide
NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE   SELECTOR
postgresql-service   ClusterIP   10.254.98.229    <none>        5432/TCP   21m   app=postgresql-db
redmine-service      ClusterIP   10.254.243.165   <none>        80/TCP     17m   app=redmine-app
```

Находимм внешний IP:

```
PS C:\Repos\GB-DevOps-Conteinerization\HW05> kubectl get svc -A
NAMESPACE                NAME                                 TYPE           CLUSTER-IP       EXTERNAL-IP
default                  kubernetes                           ClusterIP      10.254.0.1       <none>
gb-devops-conteiner-hw05 postgresql-service                   ClusterIP      10.254.98.229    <none>
gb-devops-conteiner-hw05 redmine-service                      ClusterIP      10.254.243.165   <none>
ingress-nginx            ingress-nginx-controller             LoadBalancer   10.254.58.63     94.139.246.246.nip.io
ingress-nginx            ingress-nginx-controller-admission   ClusterIP      10.254.95.232    <none>
ingress-nginx            ingress-nginx-controller-metrics     ClusterIP      10.254.141.94    <none>
ingress-nginx            ingress-nginx-default-backend        ClusterIP      10.254.65.107    <none>
kube-system              calico-typha                         ClusterIP      10.254.48.204    <none>
kube-system              csi-cinder-controller-service        ClusterIP      10.254.113.153   <none>
kube-system              kube-dns                             ClusterIP      10.254.0.10      <none>
kube-system              metrics-server                       ClusterIP      10.254.100.64    <none>
kubernetes-dashboard     dashboard-metrics-scraper            ClusterIP      10.254.127.245   <none>
kubernetes-dashboard     kubernetes-dashboard                 ClusterIP      10.254.204.26    <none>
opa-gatekeeper           gatekeeper-webhook-service           ClusterIP      10.254.110.63    <none>
PS C:\Repos\GB-DevOps-Conteinerization\HW05>
```

Открываем в браузере - все работает: