Команды:

Просмотр всех установленных компонентов:

Kubectl get all

Обновление конфигурации:

Kubectl apply -f <имя файла конфигурации.yaml> <- для применения конкретного файла

Kubectl apply -f <путь к папке с файлами \*.yaml> <- для применения всех файлов \*.yaml в указанной папке

Развертывания:

kubectl delete --all deployments <- удалить все развертывания

kubectl get deployments <- просмотреть все развертывания

Сервисы:

kubectl delete --all services <- удалить все сервисы

kubectl get servives <- просмотреть все сервисы

kubectl describe svc <- просмотреть определения сервисов

kubectl get endpointslices <- просмотреть

Поды:

kubectl delete --all pods <- удалить все поды

kubectl get pods <- просмотреть все поды

Сеты:

kubectl delete statefulsets –all <- удалить все сеты

kubectl get statefulsets <- просмотреть все сеты

Локальные тома и локальные тома клаймсы:

kubectl get pv <- просмотреть локальные тома

kubectl get pvс <- просмотреть тома-клаймсы

kubectl delete –all pv <- удалить локальные тома

kubectl delete –all pvc <- удалить локальные тома-клаймсы

kubectl delete pv <наименование тома>

kubectl delete pvc <наименование тома-клаймса>

Секреты:

kubectl create secret generic <наименование секрета> --from-literal <наименование ключа>=<значение> [--from literal ...] <- создать секрет

kubectl get secrets <- просмотреть секреты

kubectl delete secrets –all <- удалить секреты

Просмотреть лог нужного пода:

kubectl logs <наименование пода>

Подключиться к поду postgres:

kubectl exec -it <наименование пода> -- psql -h localhost -U postgres --password -p 5432 postgres

Просмотр установок кластера:

kubectl cluster-info

Установка балансера запросов:

kubectl expose deployments <наименование деплоя> --type=LoadBalancer

Восстанавливаем данные в БД Postgres в кластере

1. Ставим кластер

- загружаем образ postgres в докер

- делаем файл подключения тома между кластером и файловой системой postgres-volume.yaml:

apiVersion: v1

kind: PersistentVolume

metadata:

name: task-pv-volume

labels:

type: local

spec:

storageClassName: manual

capacity:

storage: 1Gi

accessModes:

- ReadWriteOnce

hostPath:

path: "/run/desktop/mnt/host/D/Docker/Volumes/Postgres"

type: DirectoryOrCreate

---

apiVersion: v1

kind: PersistentVolumeClaim

metadata:

name: postgres-pvc

spec:

storageClassName: manual

accessModes:

- ReadWriteOnce

resources:

requests:

storage: 1Gi

* Делаем файл разворачивания экземпляра докера в кластере postgres-deployment.yaml:

apiVersion: apps/v1

kind: StatefulSet

metadata:

name: postgre

labels:

app: postgre

spec:

replicas: 1

selector:

matchLabels:

app: postgre

template:

metadata:

labels:

app: postgre

spec:

containers:

- name: postgre

image: postgres

envFrom:

- configMapRef:

name: postgre-config

- secretRef:

name: postgre-secrets

ports:

- containerPort: 5432

name: postgre

volumeMounts:

- mountPath: "/var/lib/postgresql/data"

name: postgredb

volumes:

- name: postgredb

persistentVolumeClaim:

claimName: postgres-pvc

---

apiVersion: v1

kind: ConfigMap

metadata:

name: postgre-config

labels:

app: postgre

data:

POSTGRES\_DB: "activities"

---

apiVersion: v1

kind: Secret

metadata:

name: postgre-secrets

labels:

app: postgre

type: Opaque

data:

POSTGRES\_USER: cG9zdGdyZXM= #postgres

POSTGRES\_PASSWORD: YWRtaW4= #admin

---

apiVersion: v1

kind: Service

metadata:

name: postgre

labels:

app: postgre

spec:

selector:

app: postgre

type: NodePort

ports:

- port: 5432

nodePort: 30432

* Разворачиваем сначала том:

kubectl apply -f postgres-volume.yaml

* Далее разворачиваем экземпляр сервера postgres:

kubectl apply -f postgres-deployment.yaml

* Проверяем:

kubectl get pv <-это должен быть смонтирован том Volume, должен быть Bounded

kubectl get pvc <- это должен быть смонтирован том VolumeClaim, должен быть Bounded

kubectl get pods <- должен быть запущен экземпляр postgre-0, статус Running

1. Восстанавливаем БД

- должен быть бекап БД в формате tar

- открываем каталог - [\\wsl.localhost\docker-desktop\mnt\host\D\Docker\Volumes\Postgres](file:///\\wsl.localhost\docker-desktop\mnt\host\D\Docker\Volumes\Postgres), там должны быть данные – здесь спроецирован каталог с данными работающего экземпляра сервера postgres. Копируем в этот каталог наш бекап.

- открываем в DockerDesctop в запущенных контейнерах контейнер с именем k8s\_postgre\_postgre-0\_....

Это наш экземпляр.

Или его можно открыть из консоли:

kubectl exec –stdin –tty postgre-0 -- /bin/bash

Далее устанавливаем файловый менеджер:

apt update

apt install mc

mc

переходим -> /var/lib/postgresql/data

там должен находится наш бекап

выходим из mc -> exit

также можно было переходить командами cd (переход) и ls (просмотр).

**Восстанавливаем нужные БД из бекапов.**

- подключаемся к БД и создаем нужную БД:

psql -U postgres -d postgres

create database <имя БД>;

выходим в командную консоль – exit

восстанавливаем БД:

pg\_restore -h localhost -p 5432 -U postgres -d <имя БД> -v <имя файла бекапа БД>

должно все восстановится. Проверяем:

psql -U postgres -d <имя БД>

select \* from “public”.”<наименование любой не пустой таблицы БД>”;

должны отобразится данные, важно – наименование таблиц – casesensitive! Так что нужно указывать точно.

1. **Восстанавливаем pgAdmin в кластере.**

- нужный файл конфигурации:

apiVersion: v1

kind: Secret

metadata:

name: pgadmin-secret

type: Opaque

data:

pgadmin-default-password: YWRtaW4= #admin

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: pgadmin

spec:

selector:

matchLabels:

app: pgadmin

replicas: 1

template:

metadata:

labels:

app: pgadmin

spec:

containers:

- name: pgadmin4

image: dpage/pgadmin4

env:

- name: PGADMIN\_DEFAULT\_EMAIL

value: "admin@admin.com"

- name: PGADMIN\_DEFAULT\_PASSWORD

valueFrom:

secretKeyRef:

name: pgadmin-secret

key: pgadmin-default-password

- name: PGADMIN\_PORT

value: "80"

ports:

- containerPort: 80

name: pgadminport

---

apiVersion: v1

kind: Service

metadata:

name: pgadmin

labels:

app: pgadmin

spec:

selector:

app: pgadmin

type: NodePort

ports:

- port: 80

nodePort: 30200

Сохраняем скрипт в файл – pgadmin.yaml.

* Скачиваем образ pgadmin -> docker pull dpage/pgadmin4
* Разворачиваем под в кластере -> kubectl apply -f pgadmin.yaml
* Проверяем что все работает – должен быть сервис и deployment -> kubectl get all
* Подключаемся к pgadmin, он работает через браузер по порту 30200 – это видно в определении сервиса:



* Во входных данных логин – то, что указали в файле конфигурации pgadmin -> [admin@admin.com](mailto:admin@admin.com)

Пароль - admin, должно все открыться.

* Подключаемся к серверу БД ->

Адрес сервера – свой локальный IP-адрес, смотрим по команде ipconfig

Порт – то что указали в файле конфигурации определения сервера БД -> 30432

Логин -> стандартный админ для сервера Postgres -> postgres

Пароль -> то, что указали в конфигурации сервера БД -> admin

Теперь должно подключиться и отобразиться ранее восстановленная БД.