УДК 004

Создание модуля MangoDB в Kubernetes

Еровлева Регина Викторовна
Приамурский государственный университет имени Шолом-Алейхема
Студент

Еровлев Павел Андреевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье будут рассмотрены работы по добавления модуля базы данный MangoDB в Kubernetes. Будет произведена настройка компонентов необходимых для разворачивания модуля. Итоговым результатом будет являться настроенное приложение с подкаченной базой данных.
Ключевые слова: Kubernetes, Java, База данный

Creation of the Mangodb module in Kubernetes

Erovleva Regina Viktorovna
Sholom-Aleichem Priamursky State University
Student

Erovlev Pavel Andreevich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article will consider work on adding a database module given Mangodb in Kubernetes. The components of the components necessary for deploying the module will be configured. The final result will be a configured application with a pumped database.
Keywords: Kubernetes, Java, Database

База данных – это необходимый сервис для работы любого существующего проекта. Она должна быть хорошо защищена, иметь достаточный уровень шифрования и обработки данных, а также должна быть отказоустойчивой, для полноценной работы.

MangoDB – это отличная база данных имеющая удобный сетевой интерфейс. Но не имеющая возможность просматривать нагрузку во время работы.

Kubernetes – это открытое ПО для автоматического развертывания контейнеризированных приложений.

Цель данной статьи – настроить Kubernetes для шифрования данных.

М.С. Ивченко, В.Г. Тарасов рассмотрели Kubernetes для построения облачной платформы для запуска удаленных учебных сервисов [1]. О.О. Сергеева и А.Р. Белозерова провели сравнительный анализ потребительский сведений об аппаратной виртуализации на основе Kubernetes и Docker [2]. В своей работе А.А. Артамонова провела сравнительный анализ систем для автоматической контейнеризации приложений: Kubernetes и Docker [3]. Так же Д.П. Белов и И.А. Пелевин разработали методику для обеспечения отказоустойчивости микросервисных архитектур на базе Kubernetes и Docker [4]. С.П. Хиков разработал модель эффективного выбора средств сканирования изображений в Kubernetes [5].

Сперва создадим файл службы MongoDB (рис.1).

```
apiVersion: v1
kind: Service
metadata:
  labels:
    name: mongo
  name: mongo
spec:
  ports:
    - port: 27017
      targetPort: 27017
  selector:
    name: mongo
```

Рисунок 1 – service.yaml

Теперь создадим сервис командой «kubectl apply -f service.yaml» и после успешного создания получим в ответ «service/mongo created».

Немного разберем файл «service.yaml».

- port: 2701 – это порт, на котором размещена служба;
- targetPort: 27017 – это порт, на который отображается входящий порт; по умолчанию «targetPort» имеет то же значение, что и «port».

Тип службы по умолчанию — «ClusterIp». Чтобы получить доступ к сервису из-за пределов кластера, то необходимо сменить его тип на «LoadBalancer».

Проверим создание сервиса командой «kubectl get services» (рис.2).

```
NAME         TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP
31m
mongo        ClusterIP   10.107.164.120   <none>        27017/TCP
12s
```

Рисунок 2 – Созданный сервис

Теперь создадим файл для развертывания базы данных (рис.3).

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongo
  labels:
    app: mongo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongo
  template:
    metadata:
      labels:
        app: mongo
    spec:
      containers:
      - name: mongo
        image: mongo
        ports:
          - containerPort: 27017
        volumeMounts:
            - mountPath: /tmp/mongo/db
              name: mongo-storage
      volumes:
        - emptyDir: {}
          name: mongo-storage
```

Рисунок 3 – deployment.yaml

- spec -> template -> metadata -> labels: это ярлыки модуля, эти метки используются развертыванием и службой для выбора модулей для управления.
- spec -> template -> spec -> containers -> volumeMounts: это точка монтирования тома.
- mountPath: /tmp/mongo/db: определяет путь к монтированию базы данных.

После выполнения операции по запуску пода проверим его наличие (рис.4).

```
total 184
-rw------- 1 mongodb mongodb    50 Mar  2 13:00 WiredTiger
-rw------- 1 mongodb mongodb    21 Mar  2 13:00 WiredTiger.lock
-rw------- 1 mongodb mongodb  1466 Mar  2 13:01 WiredTiger.turtle
-rw------- 1 mongodb mongodb 36864 Mar  2 13:01 WiredTiger.wt
-rw------- 1 mongodb mongodb  4096 Mar  2 13:00 WiredTigerHS.wt
-rw------- 1 mongodb mongodb 20480 Mar  2 13:01 _mdb_catalog.wt
-rw------- 1 mongodb mongodb 20480 Mar  2 13:01 collection-0-
-7227758633283240453.wt
-rw------- 1 mongodb mongodb 20480 Mar  2 13:01 collection-2-
-7227758633283240453.wt
-rw------- 1 mongodb mongodb  4096 Mar  2 13:00 collection-4-
-7227758633283240453.wt
drwx------ 2 mongodb mongodb  4096 Mar  2 13:01 diagnostic.data
-rw------- 1 mongodb mongodb 20480 Mar  2 13:01 index-1-
-7227758633283240453.wt
-rw------- 1 mongodb mongodb 20480 Mar  2 13:01 index-3-
-7227758633283240453.wt
-rw------- 1 mongodb mongodb  4096 Mar  2 13:00 index-5-
-7227758633283240453.wt
-rw------- 1 mongodb mongodb  4096 Mar  2 13:00 index-6-
-7227758633283240453.wt
drwx------ 2 mongodb mongodb  4096 Mar  2 13:00 journal
-rw------- 1 mongodb mongodb     2 Mar  2 13:00 mongod.lock
-rw------- 1 mongodb mongodb  4096 Mar  2 13:00 sizeStorer.wt
-rw------- 1 mongodb mongodb   114 Mar  2 13:00 storage.bson
```

Рисунок 4 – Наличие пода

Каталог «/data/db» специфичен для каждого контейнера Docker, он содержит данные из MongoDB. Каталог «/data/db» содержит данные, которые являются общими для всех контейнеров Docker.

Осталось проверить, что сервис MongoDB запущен, для этого проверяем лог (рис.5).

```
{"minWireVersion":0,"maxWireVersion":13},"incomingInternalClient":
{"minWireVersion":13,"maxWireVersion":13},"outgoing":
{"minWireVersion":13,"maxWireVersion":13},"isInternalClient":true}}}
{"t":{"$date":"2022-03-02T13:00:16.261+00:00"},"s":"I",
"c":"STORAGE",  "id":5071100, "ctx":"initandlisten","msg":"Clearing
temp directory"}
{"t":{"$date":"2022-03-02T13:00:16.261+00:00"},"s":"I",
"c":"CONTROL",  "id":20536,   "ctx":"initandlisten","msg":"Flow
Control is enabled on this deployment"}
{"t":{"$date":"2022-03-02T13:00:16.262+00:00"},"s":"I",  "c":"FTDC",
"id":20625,   "ctx":"initandlisten","msg":"Initializing full-time
diagnostic data capture","attr":
{"dataDirectory":"/data/db/diagnostic.data"}}
{"t":{"$date":"2022-03-02T13:00:16.263+00:00"},"s":"I",
"c":"STORAGE",  "id":20320,
"ctx":"initandlisten","msg":"createCollection","attr":
{"namespace":"local.startup_log","uuidDisposition":"generated","uuid":
{"uuid":{"$uuid":"50fd8baf-0466-4a7f-93e0-0e5966f2b829"}},"options":
{"capped":true,"size":10485760}}}
{"t":{"$date":"2022-03-02T13:00:16.276+00:00"},"s":"I",  "c":"INDEX",
"id":20345,   "ctx":"initandlisten","msg":"Index build: done
building","attr":
{"buildUUID":null,"namespace":"local.startup_log","index":"_id_","comm
itTimestamp":null}}
{"t":{"$date":"2022-03-02T13:00:16.277+00:00"},"s":"I",  "c":"REPL",
"id":6015317, "ctx":"initandlisten","msg":"Setting new configuration
state","attr":
{"newState":"ConfigReplicationDisabled","oldState":"ConfigPreStart"}}
{"t":{"$date":"2022-03-02T13:00:16.279+00:00"},"s":"I",
"c":"STORAGE",  "id":20320,
"ctx":"LogicalSessionCacheRefresh","msg":"createCollection","attr":
{"namespace":"config.system.sessions","uuidDisposition":"generated","u
uid":{"uuid":{"$uuid":"5d7ca7b9-2ee7-48a7-848c-
4718764a34c1"}},"options":{}}}
{"t":{"$date":"2022-03-02T13:00:16.279+00:00"},"s":"I",
"c":"CONTROL",  "id":20712,
"ctx":"LogicalSessionCacheReap","msg":"Sessions collection is not set
up; waiting until next sessions reap interval","attr":
{"error":"NamespaceNotFound: config.system.sessions does not exist"}}
{"t":{"$date":"2022-03-02T13:00:16.280+00:00"},"s":"I",
"c":"NETWORK",  "id":23015,   "ctx":"listener","msg":"Listening
on","attr":{"address":"/tmp/mongodb-27017.sock"}}
{"t":{"$date":"2022-03-02T13:00:16.280+00:00"},"s":"I",
"c":"NETWORK",  "id":23015,   "ctx":"listener","msg":"Listening
on","attr":{"address":"0.0.0.0"}}
{"t":{"$date":"2022-03-02T13:00:16.280+00:00"},"s":"I",
"c":"NETWORK",  "id":23016,   "ctx":"listener","msg":"Waiting for
connections","attr":{"port":27017,"ssl":"off"}}
```


Рисунок 5 – Проверка логов

В логах видна запись «Waiting for connections», это означает, что сервис ожидает подключения. Перезапустим его с увеличением масштабирования (рис.6.).

```
kubectl scale deploy mongo --replicas=4
```

Рисунок 6 - Перезапуск

После перезапуска проверяем сервис и видим, что он запущен (рис.7).

```
NAME                     READY   STATUS    RESTARTS   AGE
mongo-598bb4bfc-5ln4j    1/1     Running   0          4s
mongo-598bb4bfc-n42kc    1/1     Running   0          4s
mongo-598bb4bfc-qrnh2    1/1     Running   0          3m40s
mongo-598bb4bfc-srnjw    1/1     Running   0          4s
```

Рисунок 7 – Сервис запущен

В данной статье был рассмотрен процесс подключения базы данных MongoDB к Kubernetes.

Библиографический список

1. Ивченко М.С., Тарасов В.Г. Использование kubernetes для построения облачной платформы для удаленного запуска учебных сервисов // Информационные технологии в науке, промышленности и образовании. 2020. С.107-109.
2. Сергеева О.О., Белозерова А.Р. Kubernetes с docker на локальной машине // Вестник димитровградского инженерно-технологического института. 2020. №1(21). С. 44-53.
3. Артамонова А.А. Сравнительный анализ двух систем управления контейнерами: docker swarm и kubernetes // Синергия наук. 2018. №23. С. 1173-1182.
4. Белов Д.П., Пелевин И.А. Реализация отказоустойчивости в системе оркестрации микросервисной архитектуры kubernetes // Вестник УРФО. безопасность в информационной сфере. 2019. №2(32). С. 5-11.
5. Хиков С.П. Модель выбора эффективного средства сканирования изображений контейнеров в инфраструктуре kubernetes // Инновационные технологии: теория, инструменты, практика. 2019. №1. С. 249-253.