KUB 28: Вертикальное масштабирование

Описание:

Вертикальное масштабирование — немного более большая задача. Для того чтобы сделать вертикальное масштабирование, необходимо установить vertical autoscaler, который доступен по ссылке. Давайте попробуем это сделать:

```
$ git clone https://github.com/kubernetes/autoscaler/
Cloning into 'autoscaler'...
remote: Enumerating objects: 1, done.
remote: Counting objects: 100% (1/1), done.
remote: Total 117801 (delta 0), reused 0 (delta 0), pack-reused
117800
Receiving objects: 100% (117801/117801), 108.23 MiB | 988.00
KiB/s, done.
Resolving deltas: 100% (75786/75786), done.
Updating files: 100% (18425/18425), done.
$ cd autoscaler/
# Нужно переключиться на релиз 0.8, если у вас openssl < 1.1.0,
иначе установщику не удастся сгенерировать сертификат
$ git checkout vpa-release-0.8
Updating files: 100% (11639/11639), done.
Branch 'vpa-release-0.8' set up to track remote branch
'vpa-release-0.8' from 'origin'.
Switched to a new branch 'vpa-release-0.8'
$ ./vertical-pod-autoscaler/hack/vpa-up.sh
customresourcedefinition.apiextensions.k8s.io/verticalpodautosca
lers.autoscaling.k8s.io created
customresourcedefinition.apiextensions.k8s.io/verticalpodautosca
lercheckpoints.autoscaling.k8s.io created
clusterrole.rbac.authorization.k8s.io/system:metrics-reader
created
clusterrole.rbac.authorization.k8s.io/system:vpa-actor created
clusterrole.rbac.authorization.k8s.io/system:vpa-checkpoint-acto
r created
clusterrole.rbac.authorization.k8s.io/system:evictioner created
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-read
er created
```

```
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-actor
created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-checkpoi
nt-actor created
clusterrole.rbac.authorization.k8s.io/system:vpa-target-reader
created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-target-r
eader-binding created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-eviction
ter-binding created
serviceaccount/vpa-admission-controller created
clusterrole.rbac.authorization.k8s.io/system:vpa-admission-contr
oller created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-admissio
n-controller created
clusterrole.rbac.authorization.k8s.io/system:vpa-status-reader
created
clusterrolebinding.rbac.authorization.k8s.io/system:vpa-status-r
eader-binding created
serviceaccount/vpa-updater created
deployment.apps/vpa-updater created
serviceaccount/vpa-recommender created
deployment.apps/vpa-recommender created
Generating certs for the VPA Admission Controller in
/tmp/vpa-certs.
Generating RSA private key, 2048 bit long modulus
.....................................................+++
.........................+++
e is 65537 (0x10001)
Generating RSA private key, 2048 bit long modulus
.........................................+++
..+++
e is 65537 (0x10001)
Signature ok
subject=/CN=vpa-webhook.kube-system.svc
Getting CA Private Key
Uploading certs to the cluster.
secret/vpa-tls-certs created
Deleting /tmp/vpa-certs.
deployment.apps/vpa-admission-controller created
service/vpa-webhook created
```

Здорово, теперь у нас все настроено для теста! Давайте создадим deployment с alpine на борту:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: alpine
spec:
  replicas: 2
  selector:
    matchLabels:
      app: alpine
  template:
    metadata:
      labels:
        app: alpine
    spec:
      containers:
        - name: alpine
          image: alpine
          command: ["ping"]
          args: ["8.8.8.8"]
          resources:
            requests:
              cpu: 50m
              memory: 50Mi
            limits:
              cpu: 50m
              memory: 50Mi
```

Мы создаем простой alpine, который будет всегда пинговать 8.8.8.8. Также мы ограничили контейнер по памяти — 50 Мб — и по cpu — 50 millicores.
Давайте теперь попробуем создать VerticalPodAutoscaler, который для начала будет наблюдать за нашим деплойментом:

```yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: alpine-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
```

```
    kind:       Deployment
    name:       alpine
  updatePolicy:
    updateMode: "Off"
```

В этом случае мы отключили автоматическое обновление подов (updateMode: off). Вообще можно выставить следующие значения:

- "Initial" — выставлять значения pod только при создании и не изменять их в будущем;
- "Recreate" — пересоздавать под, если новая рекомендация по ресурсам сильно отличается от текущих значений;
- "Auto" — сейчас это то же самое, что и Recreate. Как только обновлять ресурсы можно будет на лету — Auto будет работать по этому методу.

Теперь мы можем посмотреть рекомендации, которые выдает VPA:

```
$ kubectl get vpa  alpine-vpa -o  yaml
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler

... skipped ...

status:
  conditions:
  - lastTransitionTime: "2020-12-04T15:13:22Z"
    status: "True"
    type: RecommendationProvided
  recommendation:
    containerRecommendations:
    - containerName: alpine
      lowerBound:
        cpu: 25m
        memory: 262144k
      target:
        cpu: 25m
        memory: 262144k
      uncappedTarget:
        cpu: 25m
        memory: 262144k
      upperBound:
        cpu: 2295m
        memory: "2399961538"
```

Как видите, рекомендации — увеличить память и уменьшить cpu. Давайте попробуем изменить VPA и загрузить контейнер:

```
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: alpine-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind:       Deployment
    name:       alpine
  updatePolicy:
    updateMode: "Recreate"
```

Мы выставили политику пересоздания на основе рекомендаций VerticalPodAutoscaler. Давайте посмотрим, что произойдет дальше. Запускаем команду, которая будет кушать достаточно cpu (на генерации случайных значений):

```
$ kubectl exec -it alpine-5476d9677d-kk92m -- dd if=/dev/urandom
of=/dev/zero
```

И смотрим логи VPA-контроллера:

```
$ kubectl -n kube-system logs -f vpa-updater-7b668b79d8-5rxsn
I1204 15:24:24.699632       1 updater.go:193] evicting pod
alpine-5476d9677d-kk92m
I1204 15:24:24.729511       1 event.go:281]
Event(v1.ObjectReference{Kind:"Pod", Namespace:"default",
Name:"alpine-5476d9677d-kk92m",
UID:"36ca0d38-20a2-440a-a701-cf24ec66e205", APIVersion:"v1",
ResourceVersion:"1135275", FieldPath:""}): type: 'Normal'
reason: 'EvictedByVPA' Pod was evicted by VPA Updater to apply
resource recommendation.
```

Как видите — наш VPA-контроллер решил пересоздать под согласно новым рекомендациям. Если посмотреть на новый под, то можно увидеть, что у него были выставлены новые параметры requests / limits:

```
   resources:
      limits:
        cpu: 63m
        memory: 262144k
      requests:
        cpu: 63m
        memory: 262144k
```

Задание:

1. Установите metrics server в Kubernetes кластер.
2. Установите vertical autoscaler в ваш Kubernetes кластер
3. Создайте дейплоймент alpine-dp в namespace default, который внутри будет запускать команду dd для копирования файлов из /dev/zero в /dev/null.
4. Задайте деплойменту ограничение ресурсов 200m cpu и 128mb памяти так, чтобы класс QoS у контейнеров получился Guaranteed
5. Создайте VerticalPodAutoscaler vpa-alpine в namespace default с выключенным обновлением деплоймента (updateMode: Off). Обратите внимание, иногда для полноценного сбора статистики нужно подождать до 20 минут.
6. Отправьте задание на проверку.