МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра инструментального и прикладного программного обеспечения

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №7

по дисциплине

«Технологии виртуализации клиент-серверных приложений»

Выполнил студент группы ИКБО-01-22 Прокопчук Р.О.

Принял преподаватель кафедры ИиППО Волков М.Ю.

Практические работы выполнены «__»_____2025 г.

«Зачтено» «__»_____2025 г.

Москва
2025

Теоретическое введение

Основные типы ресурсов в системе — процессорное время и оперативная память. В манифестах Kubernetes эти типы ресурсов измеряются в следующих единицах:

— CPU — в ядрах;

— RAM — в байтах.

Для каждого ресурса есть возможность задавать два типа требований — requests и limits. Requests — описывает минимальные требования к свободным ресурсам ноды для запуска контейнера, в то время как limits устанавливает жесткое ограничение ресурсов, доступных контейнеру.

Важно понимать, что в манифесте не обязательно явно определять оба типа, при этом поведение будет следующим:

Если явно задан только limits ресурса, то requests для этого ресурса автоматически принимает значение, равное limits. Т.е. фактически работа контейнера будет ограничена таким же количеством ресурсов, которое он требует для своего запуска.

Если для ресурса явно задан только requests, то никаких ограничений сверху на этот ресурс не задается — т.е. контейнер ограничен только ресурсами самой ноды.

Также существует возможность настроить управление ресурсами не только на уровне конкретного контейнера, но и на уровне namespace при помощи следующих сущностей:

— LimitRange — описывает политику ограничения на уровне контейнера/пода, нужна для того, чтобы описать ограничения по умолчанию на контейнер/под, а также предотвращать создание объектов, требующих много ресурсов, ограничивать их количество и определять возможную разницу значений в limits и requests.

— ResourceQuotas — описывают политику ограничения по всем контейнерам в ns и используется, как правило, для разграничения ресурсов по окружениям.

Каждому поду присваивается один из 3 QoS-классов:

— guaranuted — назначается тогда, как для каждого контейнера в поде для memory и cpu задан request и limit, причем эти значения должны совпадать

— burstable — хотя бы один контейнер в поде имеет request и limit, при этом request < limit

— best effort — когда ни один контейнер в поде не ограничен по ресурсам

При этом, когда на ноде наблюдается нехватка ресурсов (диска, памяти), kubelet начинает ранжировать и выселять под’ы по определенному алгоритму, который учитывает приоритет пода и его QoS-класс.

Т.е. при одинаковом приоритете, kubelet в первую очередь будет выселять с узла поды с QoS-классом best effort.

Когда стоит задача автоматически увеличивать и уменьшать количество pod в зависимости от использования ресурсов в её решении может помочь такая сущность k8s как HPA (Horizontal Pod Autoscaler), алгоритм которого заключается в следующем:

— Определяются текущие показания наблюдаемого ресурса (currentMetricValue).

— Определяются желаемые значения для ресурса (desiredMetricValue), которые для системных ресурсов задаются при помощи request.

— Определяется текущее количество реплик (currentReplicas).

По следующей формуле рассчитывается желаемое количество реплик

desiredReplicas = [currentReplicas * (currentMetricValue / desiredMetricValue)]

При этом масштабирования не произойдет, когда коэффициент (currentMetricValue / desiredMetricValue) близок к 1 (при этом допустимую погрешность мы можем задавать сами, по умолчанию она равна 0.1).

Постановка задачи

Для выполнения практической работы понадобится minikube. Необходимо выполнить все команды, создать необходимые конфигурации и отобразить их в отчете.

Запустим сервер метрик в minikube:

Нужно проверить работу сервера меток:

Если API ресурсов метрики доступно, в выводе команды будет содержаться ссылка на metrics.k8s.io

Нужно создать пространство имён, чтобы ресурсы, которыми будем пользоваться в данном упражнении, были изолированы от остального кластера

Для установки запроса памяти контейнеру используется поле resources:requests

Для ограничений по памяти используется resources:limits

Раздел args конфигурационного файла содержит аргументы для контейнера в момент старта. Аргументы "--vm-bytes", "150M" указывают контейнеру попытаться занять 150 Мб памяти.

Далее нужно создать конфигурационный файл memory-request-limit.yaml:

Далее необходимо создать под:

Нужно проверить статус Pod'a:

Необходимо просмотреть подробную информацию о Pod'e:

Вывод команды должен показать что для контейнера в Pod'e зарезервировано 100 Мб памяти и выставлено 200 Мб ограничения.

Запустите kubectl top, чтобы получить метрики Pod'a:

Нужно произвести удаление Pod:

Далее создан Pod, который попытается занять больше памяти, чем для него ограничено. Ниже представлен конфигурационный файл для Pod'a с одним контейнером, имеющим 50 Мб на запрос памяти и 100 Мб лимита памяти:

Далее необходимо создать Pod:

Нужно просмотреть подробную информацию о Pod'e:

В этот момент контейнер уже либо запущен, либо убит. Повторяем предыдущую команду, пока контейнер не окажется убитым:

Требуется изучить ещё более подробный вид статуса контейнера:

В выводе показано, что контейнер был убит по причине недостатка памяти (OOM):

Так как контейнер может быть перезапущен, kubelet стартует его. Требуется выполнить команду несколько раз, чтобы увидеть, как контейнер убивается и запускается заново:

Вывод показывает, что контейнер убит, перезапущен, снова убит.:

Требуется просмотреть подробную информацию об истории Pod'a:

Вывод показывает, что контейнер постоянно запускается и падает: Посмотрим детальную информацию о нодах на кластере: В выводе содержится запись о том, что контейнер убивается по причине нехватки памяти:

Далее необходимо удалить Pod:

После удаления нужно создать Pod, чей запрос памяти будет превышать ёмкость любой ноды в кластере. Ниже представлен конфигурационный файл для Pod'a с одним контейнером, имеющим запрос памяти в 1000 Гб (что наверняка превышает ёмкость любой имеющейся ноды):

Создадим Pod:

Проверим статус Pod'a:

Вывод показывает, что Pod имеет статус PENDING. Это значит, что он не запланирован ни на одной ноде, и такой статус будет сохраняться всё время:

Посмотрим подробную информацию о Pod'e, включающую события:

Вывод показывает невозможность запуска контейнера из-за нехватки памяти на нодах:

Удалим Pod:

Удалим пространство имён. Эта операция удалит все Pod'ы, созданные в рамках данного упражнения:

Далее необходимо создать Pod, имеющий один контейнер. Требуется задать для контейнера запрос в 0.5 CPU и лимит в 1 CPU. Конфигурационный файл для такого Pod'a:

Раздел args конфигурационного файла содержит аргументы для контейнера в момент старта. Аргумент -cpus "2" говорит контейнеру попытаться использовать 2 CPU.

Нужно создать Pod и удостовериться, что он запущен:

Далее нужно просмотреть детальную информацию о Pod'e:

В выводе видно, что Pod имеет один контейнер с запросом в 500 миллиCPU и с ограничением в 1 CPU.

Требуется запустить kubectl top, чтобы получить метрики Pod'a:

В этом варианте вывода Pod'ом использовано 974 милли-CPU, что лишь чуть меньше заданного в конфигурации Pod'a ограничения в 1 CPU.

Далее можно удалить Pod:

Далее нужно создать Pod с запросом CPU, превышающим мощности любой ноды в вашем кластере. Ниже представлен конфигурационный файл для Pod'a с одним контейнером. Контейнер запрашивает 100 CPU, что почти наверняка превышает имеющиеся мощности любой ноды в кластере.

Далее нужно создать Pod:

Требуется проверить статус Pod'a:

Вывод показывает Pending статус у Pod'a. То есть Pod не запланирован к запуску ни на одной ноде и будет оставаться в статусе Pending постоянно:

Нужно просмотреть подробную информацию о Pod'e, включающую в себя события:

В выводе отражено, что контейнер не может быть запланирован из-за нехватки ресурсов CPU на нодах:

Далее можно удалить Pod:

Ход работы

Ход работы представлен на рисукнках 1-16.

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> minikube addons enable metrics-server
* metrics-server is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Используется образ registry.k8s.io/metrics-server/metrics-server:v0.8.0
* The 'metrics-server' addon is enabled
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl get apiservices
NAME                                SERVICE                      AVAILABLE   AGE
v1.                                 Local                        True        4m39s
v1.admissionregistration.k8s.io     Local                        True        4m39s
v1.apiextensions.k8s.io             Local                        True        4m39s
v1.apps                             Local                        True        4m39s
v1.authentication.k8s.io            Local                        True        4m39s
v1.authorization.k8s.io             Local                        True        4m39s
v1.autoscaling                      Local                        True        4m39s
v1.batch                            Local                        True        4m39s
v1.certificates.k8s.io              Local                        True        4m39s
v1.coordination.k8s.io              Local                        True        4m39s
v1.discovery.k8s.io                 Local                        True        4m39s
v1.events.k8s.io                    Local                        True        4m39s
v1.flowcontrol.apiserver.k8s.io     Local                        True        4m39s
v1.networking.k8s.io                Local                        True        4m39s
v1.node.k8s.io                      Local                        True        4m39s
v1.policy                           Local                        True        4m39s
v1.rbac.authorization.k8s.io        Local                        True        4m39s
v1.resource.k8s.io                  Local                        True        4m39s
v1.scheduling.k8s.io                Local                        True        4m39s
v1.storage.k8s.io                   Local                        True        4m39s
v1beta1.metrics.k8s.io              kube-system/metrics-server   True        77s
v2.autoscaling                      Local                        True        4m39s
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl create namespace roman-prokopchuk-ikbo-01-22
namespace/roman-prokopchuk-ikbo-01-22 created
```

Рисунок 1 – Подготовка окружения

memory-request-limit.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo
  namespace: roman-prokopchuk-ikbo-01-22
spec:
  containers:
    - name: memory-demo-ctr
      image: polinux/stress
      resources:
        limits:
          memory: "200Mi"
        requests:
          memory: "100Mi"
      command: ["stress"]
      args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

Рисунок 2 – Конфигурация pod-а memory-demo

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 apply -f memory-reque
st-limit.yaml
pod/memory-demo created
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 get pod memory-demo
NAME          READY   STATUS    RESTARTS   AGE
memory-demo   1/1     Running   0          41s
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 get pod memory-demo -
-output=yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"memory-demo","namespace":"roman-prokopchuk
-ikbo-01-22"},"spec":{"containers":[{"args":["--vm","1","--vm-bytes","150M","--vm-hang","1"],"command":["stress"],"im
age":"polinux/stress","name":"memory-demo-ctr","resources":{"limits":{"memory":"200Mi"},"requests":{"memory":"100Mi"}
}}]}}
  creationTimestamp: "2025-12-09T19:13:21Z"
  generation: 1
  name: memory-demo
  namespace: roman-prokopchuk-ikbo-01-22
  resourceVersion: "902"
  uid: 66ec8543-3efc-4328-b871-482315e668aa
spec:
  containers:
  - args:
    - --vm
    - "1"
    - --vm-bytes
    - 150M
    - --vm-hang
    - "1"
    command:
    - stress
    image: polinux/stress
    imagePullPolicy: Always
    name: memory-demo-ctr
    resources:
      limits:
        memory: 200Mi
      requests:
        memory: 100Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-5cpmj
      readOnly: true
```

Рисунок 3 – Применение конфигурации memory-demo и просмотр

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 top pod memory-demo
NAME          CPU(cores)   MEMORY(bytes)
memory-demo   9m           150Mi
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 delete pod memory-demo
pod "memory-demo" deleted
```

Рисунок 4 – Вывод информации о ресурсах и удаление memory-demo

memory-request-limit-2.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-2
  namespace: roman-prokopchuk-ikbo-01-22
spec:
  containers:
    - name: memory-demo-2-ctr
      image: polinux/stress
      resources:
        requests:
          memory: "50Mi"
        limits:
          memory: "100Mi"
      command: ["stress"]
      args: ["--vm", "1", "--vm-bytes", "250M", "--vm-hang", "1"]
```

Рисунок 5 – Конфигурация pod-а memory-demo-2

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 apply -f memory-request-limit-
2.yaml
pod/memory-demo-2 created
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 get pod memory-demo-2

NAME            READY   STATUS      RESTARTS      AGE
memory-demo-2   0/1     OOMKilled   3 (38s ago)   64s
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 get pod memory-demo-2
 --output=yaml
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      {"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"name":"memory-demo-2","namespace":"roman-prokopch
uk-ikbo-01-22"},"spec":{"containers":[{"args":["--vm","1","--vm-bytes","250M","--vm-hang","1"],"command":["stress"],"
image":"polinux/stress","name":"memory-demo-2-ctr","resources":{"limits":{"memory":"100Mi"},"requests":{"memory":"50M
i"}}}]}}
  creationTimestamp: "2025-12-09T19:19:31Z"
  generation: 1
  name: memory-demo-2
  namespace: roman-prokopchuk-ikbo-01-22
  resourceVersion: "1285"
  uid: e0fc4cea-bfb3-40a7-a7a4-f4b453f99e20
spec:
  containers:
  - args:
    - --vm
    - "1"
    - --vm-bytes
    - 250M
    - --vm-hang
    - "1"
    command:
    - stress
    image: polinux/stress
    imagePullPolicy: Always
    name: memory-demo-2-ctr
    resources:
      limits:
        memory: 100Mi
      requests:
        memory: 50Mi
    terminationMessagePath: /dev/termination-log
    terminationMessagePolicy: File
    volumeMounts:
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
      name: kube-api-access-wr9bt
      readOnly: true
```

Рисунок 6 – Применение конфигурации memory-demo-2 и просмотр

```
containerStatuses:
- allocatedResources:
    memory: 50Mi
  containerID: docker://4809e40686f5f4f6b2c598dc8fb88cc1b6798abe24fc9e51d63328ee79f2d804
  image: polinux/stress:latest
  imageID: docker-pullable://polinux/stress@sha256:b6144f84f9c15dac80deb48d3a646b55c7043ab1d83ea0a697c09097aaad21aa
  lastState:
    terminated:
      containerID: docker://4809e40686f5f4f6b2c598dc8fb88cc1b6798abe24fc9e51d63328ee79f2d804
      exitCode: 1
      finishedAt: "2025-12-09T19:20:25Z"
      reason: OOMKilled
      startedAt: "2025-12-09T19:20:25Z"
```

Рисунок 7 – Информации о lastState memory-demo-2

```
Events:
  Type     Reason     Age                 From               Message
  ----     ------     ----                ----               -------
  Normal   Scheduled  3m45s               default-scheduler  Successfully assigned roman-prokopchuk-ikbo-01-22/memor
y-demo-2 to minikube
  Normal   Pulled     3m42s               kubelet            Successfully pulled image "polinux/stress" in 2.77s (2.
77s including waiting). Image size: 9744175 bytes.
  Normal   Pulled     3m39s               kubelet            Successfully pulled image "polinux/stress" in 3.146s (3
.146s including waiting). Image size: 9744175 bytes.
  Normal   Pulled     3m19s               kubelet            Successfully pulled image "polinux/stress" in 2.363s (2
.363s including waiting). Image size: 9744175 bytes.
  Normal   Pulled     2m51s               kubelet            Successfully pulled image "polinux/stress" in 2.636s (2
.636s including waiting). Image size: 9744175 bytes.
  Normal   Pulled     2m7s                kubelet            Successfully pulled image "polinux/stress" in 2.346s (2
.346s including waiting). Image size: 9744175 bytes.
  Normal   Pulling    36s (x6 over 3m45s)  kubelet            Pulling image "polinux/stress"
  Normal   Created    33s (x6 over 3m42s)  kubelet            Created container: memory-demo-2-ctr
  Normal   Started    33s (x6 over 3m42s)  kubelet            Started container memory-demo-2-ctr
  Normal   Pulled     33s                 kubelet            Successfully pulled image "polinux/stress" in 2.428s (2
.428s including waiting). Image size: 9744175 bytes.
  Warning  BackOff    8s (x18 over 3m38s)  kubelet            Back-off restarting failed container memory-demo-2-ctr
in pod memory-demo-2_roman-prokopchuk-ikbo-01-22(e0fc4cea-bfb3-40a7-a7a4-f4b453f99e20)
```

Рисунок 8 – Информация о прекращении перезапуска memory-demo-2

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 delete pod memory-de
o-2
pod "memory-demo-2" deleted
```

Рисунок 9 – Удаление memory-demo-2

memory-request-limit-3.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memory-demo-3
  namespace: roman-prokopchuk-ikbo-01-22
spec:
  containers:
    - name: memory-demo-3-ctr
      image: polinux/stress
      resources:
        limits:
          memory: "1000Gi"
        requests:
          memory: "1000Gi"
      command: ["stress"]
      args: ["--vm", "1", "--vm-bytes", "150M", "--vm-hang", "1"]
```

Рисунок 10 – Конфигурация pod-а memory-demo-3

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 apply -f memory-reque
st-limit-3.yaml
pod/memory-demo-3 created
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 get pod memory-demo-3

NAME            READY   STATUS    RESTARTS   AGE
memory-demo-3   0/1     Pending   0          16s
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 describe pod memory-d
emo-3
Name:             memory-demo-3
Namespace:        roman-prokopchuk-ikbo-01-22
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  memory-demo-3-ctr:
    Image:      polinux/stress
    Port:       <none>
    Host Port:  <none>
    Command:
      stress
    Args:
      --vm
      1
      --vm-bytes
      150M
      --vm-hang
      1
    Limits:
      memory:  10006i
    Requests:
      memory:     10006i
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-bjdp4 (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-bjdp4:
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    ConfigMapOptional:        <nil>
    DownwardAPI:              true
QoS Class:                    Burstable
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  41s   default-scheduler  0/1 nodes are available: 1 Insufficient memory. no new claims t
o deallocate, preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
```

Рисунок 11 – Применение конфигурации memory-demo-3 и просмотр

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl --namespace roman-prokopchuk-ikbo-01-22 delete pod memory-dem
o-3
pod "memory-demo-3" deleted
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl delete namespace roman-prokopchuk-ikbo-01-22
namespace "roman-prokopchuk-ikbo-01-22" deleted
```

Рисунок 12 – Удаление namespace и memory-demo-3

cpu-request-limit.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: cpu-demo
  namespace: roman-prokopchuk-ikbo-01-22
spec:
  containers:
    - name: cpu-demo-ctr
      image: vish/stress
      resources:
        limits:
          cpu: "1"
        requests:
          cpu: "0.5"
      args:
        - -cpus
        - "2"
```

Рисунок 13 – Конфигурация pod-а cpu-demo

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 apply -f cpu-request-limit.yaml
pod/cpu-demo configured
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 get pod cpu-demo
NAME       READY   STATUS    RESTARTS   AGE
cpu-demo   1/1     Running   0          100s
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 top pod cpu-demo
NAME       CPU(cores)   MEMORY(bytes)
cpu-demo   990m         1Mi
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 delete pod cpu-demo
pod "cpu-demo" deleted
```

Рисунок 14 – Применение конфигурации cpu-demo, просмотр информации о нем и его удаление

cpu-request-limit-2.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: cpu-demo-2
  namespace: roman-prokopchuk-ikbo-01-22
spec:
  containers:
    - name: cpu-demo-ctr-2
      image: vish/stress
      resources:
        limits:
          cpu: "100"
        requests:
          cpu: "100"
      args:
        - -cpus
        - "2"
```

Рисунок 15 – Конфигурация pod-а cpu-demo-2

```
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 apply -f cpu-request-limit-2.y
aml
pod/cpu-demo-2 created
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 get pod cpu-demo-2
NAME         READY   STATUS    RESTARTS   AGE
cpu-demo-2   0/1     Pending   0          15s
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 describe pod cpu-demo-2
Name:             cpu-demo-2
Namespace:        roman-prokopchuk-ikbo-01-22
Priority:         0
Service Account:  default
Node:             <none>
Labels:           <none>
Annotations:      <none>
Status:           Pending
IP:
IPs:              <none>
Containers:
  cpu-demo-ctr-2:
    Image:      vish/stress
    Port:       <none>
    Host Port:  <none>
    Args:
      -cpus
      2
    Limits:
      cpu:  100
    Requests:
      cpu:        100
    Environment:  <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-hm97z (ro)
Conditions:
  Type           Status
  PodScheduled   False
Volumes:
  kube-api-access-hm97z:
    Type:                     Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:   3607
    ConfigMapName:            kube-root-ca.crt
    ConfigMapOptional:        <nil>
    DownwardAPI:              true
QoS Class:                    Burstable
Node-Selectors:               <none>
Tolerations:                  node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                              node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  29s   default-scheduler  0/1 nodes are available: 1 Insufficient cpu. no new claims to d
eallocate, preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl -n roman-prokopchuk-ikbo-01-22 delete pod cpu-demo-2
pod "cpu-demo-2" deleted
PS C:\Users\Remsely\dev\mirea\csavt\practice-7> kubectl delete namespace roman-prokopchuk-ikbo-01-22
namespace "roman-prokopchuk-ikbo-01-22" deleted
```

Рисунок 16 – Применение конфигурации cpu-demo-2, просмотр информации о нем и его удаление

Вывод

В результате выполнения данной практической было выполнено ознакомление с конфигурацией лимитов pod-ов в Kubernetes.

Ответы на вопросы к практической работе

1. Назовите 3 QoS-класса:

Guaranteed (requests = limits для всех ресурсов), Burstable (requests < limits или не все поля заданы), Best Effort (ресурсы не заданы вообще).

2. Основные ресурсы и единицы измерения:

CPU (в ядрах или милли-ядрах, например, 500m), RAM (в байтах, например, Mi, Gi).

3. Для чего нужен HPA (Horizontal Pod Autoscaler)?

Для автоматического увеличения или уменьшения количества реплик (подов) в зависимости от текущего потребления ресурсов (нагрузки).

4. Для чего необходимо устанавливать ограничения (limits)?

Чтобы один контейнер не забрал все ресурсы ноды, вызвав сбои в других процессах, и для предотвращения создания слишком "тяжелых" объектов.

5. Что будет с узлом при превышении ограничений?

Если заканчивается память (RAM), Kubelet начнет "выселять" (evict) поды, начиная с класса Best Effort. Если превышается CPU, процессы будут просто замедляться (throttling), так как CPU — сжимаемый ресурс.

Список источников информации

1. Kubernetes: лучшие практики. — СПб.: Питер, 2021. — 288 с.: ил. — (Серия «Для профессионалов»).

2. K8S для начинающих. Первая часть — Текст: электронный [сайт]. — URL: https://habr.com/ru/post/589415/

3. Kubernetes или с чего начать, чтобы понять что это и зачем он нужен — Текст: электронный [сайт]. — URL: https://habr.com/ru/company/otus/blog/537162/

4. Основы Kubernetes — Текст: электронный [сайт]. — URL: https://habr.com/ru/post/258443/