Задание 1: настройка Azure Cloud Shell

Задание 2: Скачивание стартовых файлов

Задание 3: Создание ресурсной группы

Задание 4: Создание SSH ключа

Задание 5: Развертывание ARM шаблона

Задание 6: Создание GitHub репозитория

Задание 7: Подключение к build агенту

Задание 8: [Настройка](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/Before%20the%20HOL%20-%20Cloud-native%20applications.md#task-8-complete-the-build-agent-setup) build агента

Задание 9: Клонирование репозитория на Build агент

[Задание 10: Запуск](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/Before%20the%20HOL%20-%20Cloud-native%20applications.md#task-10-test-the-starter-application) стартового приложения

Задание 11: Сборка Docker контейнеров

Задание 12: Запуск docker контейнеров

Задание 13: Конфигурация переменных окружения

Задание 14: Отправка образов в Azure Container Registry

Задание 15: Настройка CI Pipeline для отправки образов

* Задание 1. Миграция MongoDB в Cosmos DB с помощью службы миграции базы данных Azure.

Задача 1. Включение поставщика ресурсов Microsoft.DataMigration

Задача 2: предоставление службы миграции базы данных Azure

Задача 3. Перенести данные в Azure Cosmos DB.

* Задание 2. Разверните решение в службе Azure Kubernetes.

Задача 1. Туннель в кластер службы Azure Kubernetes.

Задача 2. Разверните службу с помощью портала Azure.

Задача 3. Разверните сервис с помощью kubectl

Задача 4. Обзор Azure Monitor для контейнеров.

* Задание 3: масштабируйте приложение и тестируйте высокую доступность

Задача 1. Увеличение количества экземпляров службы с портала Azure.

Задача 2: устранение сбоев при инициализации реплик.

Задача 3: перезапустить контейнеры и протестировать HA

Задача 4. Настройка автомасштабирования Cosmos DB.

Задача 5. Тестирование автомасштабирования Cosmos DB.

* Задание 4: Работа со службами и маршрутизация трафика приложений.

Задача 1. Обновите внешнюю службу для поддержки динамического обнаружения с помощью балансировщика нагрузки.

Задача 2. Отрегулируйте ограничения ЦП, чтобы улучшить масштаб

Задача 3: выполнить скользящее обновление

Задача 4: настроить Kubernetes Ingress

**Аннотация**

Эта практическая лабораторная работа предназначена для того, чтобы провести вас через процесс создания и развертывания образов Docker на платформе Kubernetes, размещенной на Azure Kubernetes Services (AKS), а также научиться работать с динамическим обнаружением сервисов, масштабированием сервисов и высокой доступность.

По завершении этой лабораторной работы вы сможете создавать и развертывать контейнерные приложения в службе Azure Kubernetes и выполнять стандартные процедуры DevOps.

**Обзор**

Fabrikam Medical Conferences (FabMedical) предоставляет услуги веб-сайта конференций, адаптированные для медицинского сообщества. Они реорганизуют код своего приложения на основе node.js, чтобы его можно было запускать как приложение Docker, и хотят реализовать POC, который поможет им ознакомиться с процессом разработки, жизненным циклом развертывания и критическими аспектами хостинга. среда. Они будут развертывать свои приложения в службе Azure Kubernetes и хотят узнать, как развертывать контейнеры с динамической балансировкой нагрузки, обнаруживать контейнеры и масштабировать их по запросу.

В этой практической лабораторной работе вы поможете заполнить этот POC с подмножеством кодовой базы приложения. Вы создадите агент сборки на основе Linux и кластер службы Azure Kubernetes для запуска развернутых приложений. Вы поможете им завершить настройку Docker для своего приложения, протестировать локально, отправить в репозиторий образов, развернуть в кластере и протестировать балансировку нагрузки и масштабирование.

Важно: для большинства ресурсов Azure требуются уникальные имена. На протяжении этих шагов вы будете видеть слово «СУФФИКС» как часть имен ресурсов. Вы должны заменить это уникальным дескриптором (например, префиксом электронной почты вашей учетной записи Microsoft), чтобы обеспечить уникальные имена для ресурсов..

**Архитектура решения**

Ниже представлена схема архитектуры решения, которую вы создадите. Пожалуйста, изучите это внимательно, чтобы понять решение в целом, работая над различными компонентами.

Решение будет использовать службу Azure Kubernetes (AKS), что означает, что топология кластера контейнеров подготавливается в соответствии с количеством запрошенных узлов. Предлагаемые контейнеры, развернутые в кластере, показаны ниже с Cosmos DB в качестве управляемой службы:

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/solution-topology.png)

Each tenant will have the following containers:

* **Conference Web site**: Приложение SPA, которое будет использовать параметры конфигурации для обработки пользовательских стилей для клиента.
* **Admin Web site**: Приложение SPA, которое владельцы конференций используют для управления деталями конфигурации конференции, управления регистрацией участников, управления кампаниями и общения с участниками.
* **Registration service**: API, который обрабатывает все действия по регистрации, создавая новые регистрации на конференции с соответствующим выбором пакетов и соответствующей стоимостью.
* **Email service**: API, который обрабатывает уведомления по электронной почте для участников конференции во время регистрации или когда владельцы конференции решают привлечь участников через свой сайт администратора.
* **Config service**: API-интерфейс, который обрабатывает такие параметры конфигурации конференции, как даты, места, таблицы цен, специальные предложения, обратный отсчет и т. д.
* **Content service**: API, который обрабатывает контент для конференции, такой как докладчики, сессии, семинары и спонсоры.

**Задание 1: Миграция MongoDB в Cosmos DB с помощью службы миграции базы данных Azure**

На этом этапе у вас есть веб-приложения и приложения API, запущенные в экземпляре Docker (VM - Build Agent). Следующим шагом является перенос данных базы данных MongoDB в Azure Cosmos DB. В этом упражнении будет использоваться служба миграции базы данных Azure для переноса данных из базы данных MongoDB в Azure Cosmos DB.

**Задача 1. Включение поставщика ресурсов Microsoft.DataMigration**

В этой задаче вы включите использование службы миграции базы данных Azure в своей подписке Azure, зарегистрировав поставщика ресурсов Microsoft.DataMigration.

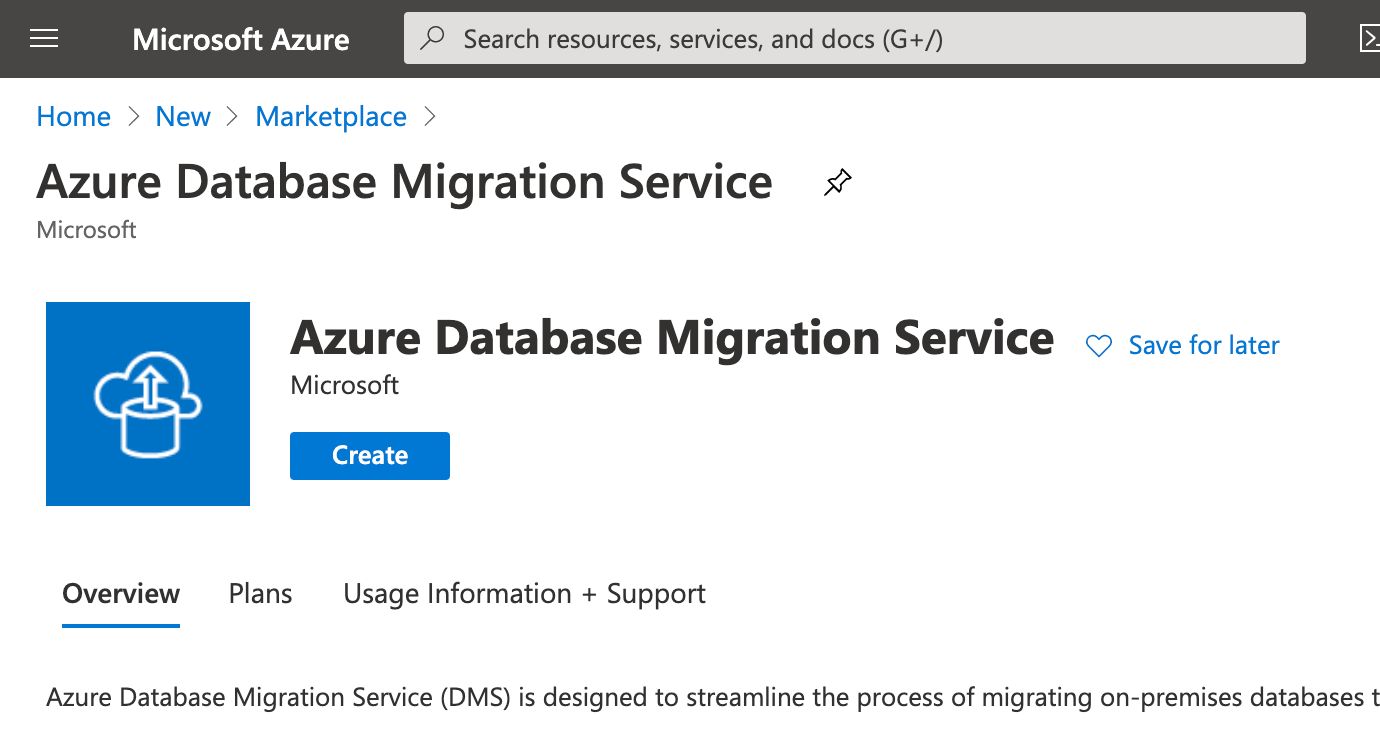
1. Откройте Azure Cloud Shell.
2. Выполните следующую команду Azure CLI, чтобы зарегистрировать поставщика ресурсов Microsoft.DataMigration в своей подписке Azure:

az provider register --namespace Microsoft.DataMigration

**Задача 2. Подготовка службы миграции базы данных Azure**

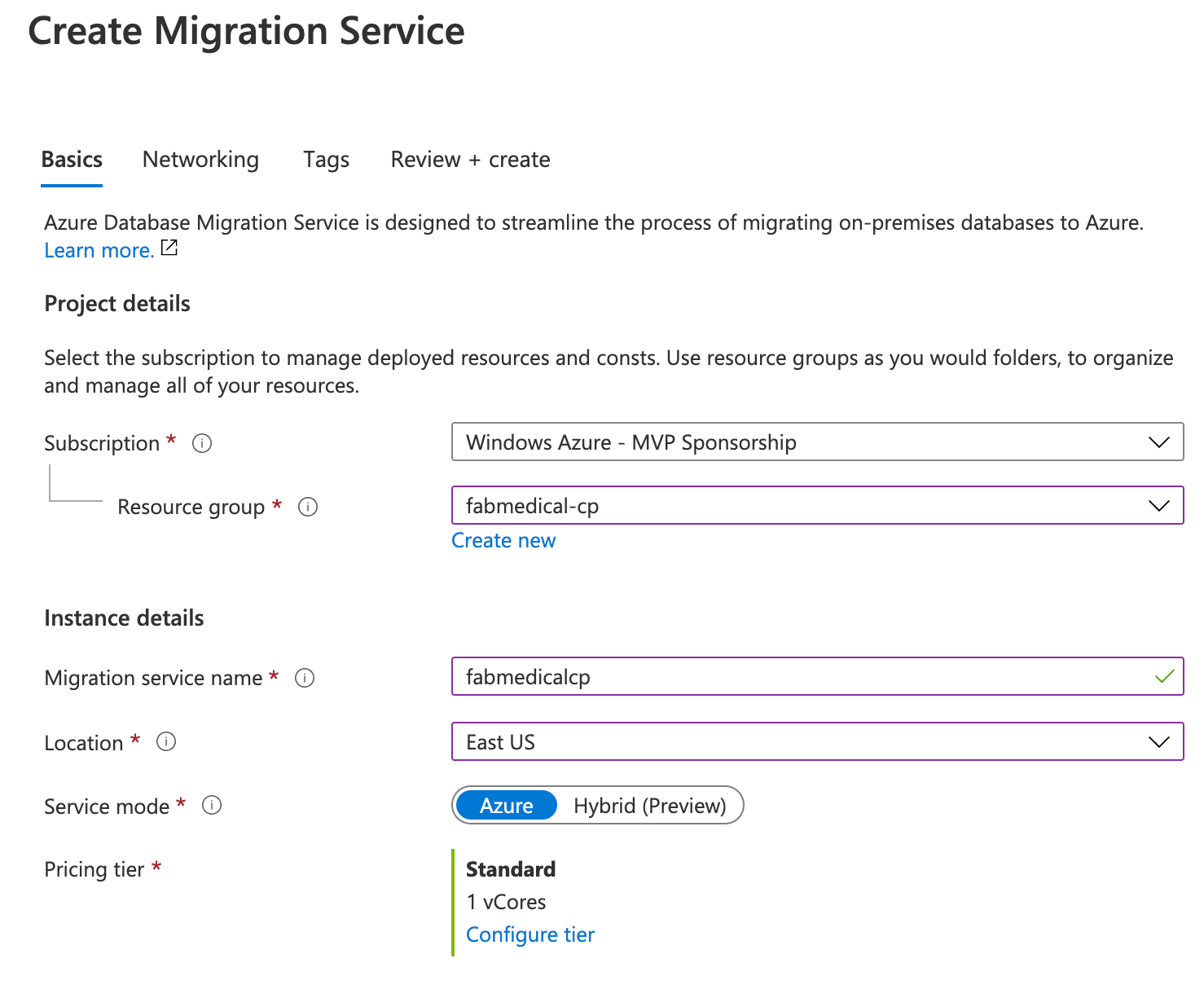
В этой задаче вы развернете экземпляр службы миграции базы данных Azure, который будет использоваться для миграции данных из MongoDB в Cosmos DB.

1. На портале Azure выберите + Создать ресурс.
2. Найдите на торговой площадке Службу миграции базы данных Azure и выберите ее.
3. Выберите "Создать".

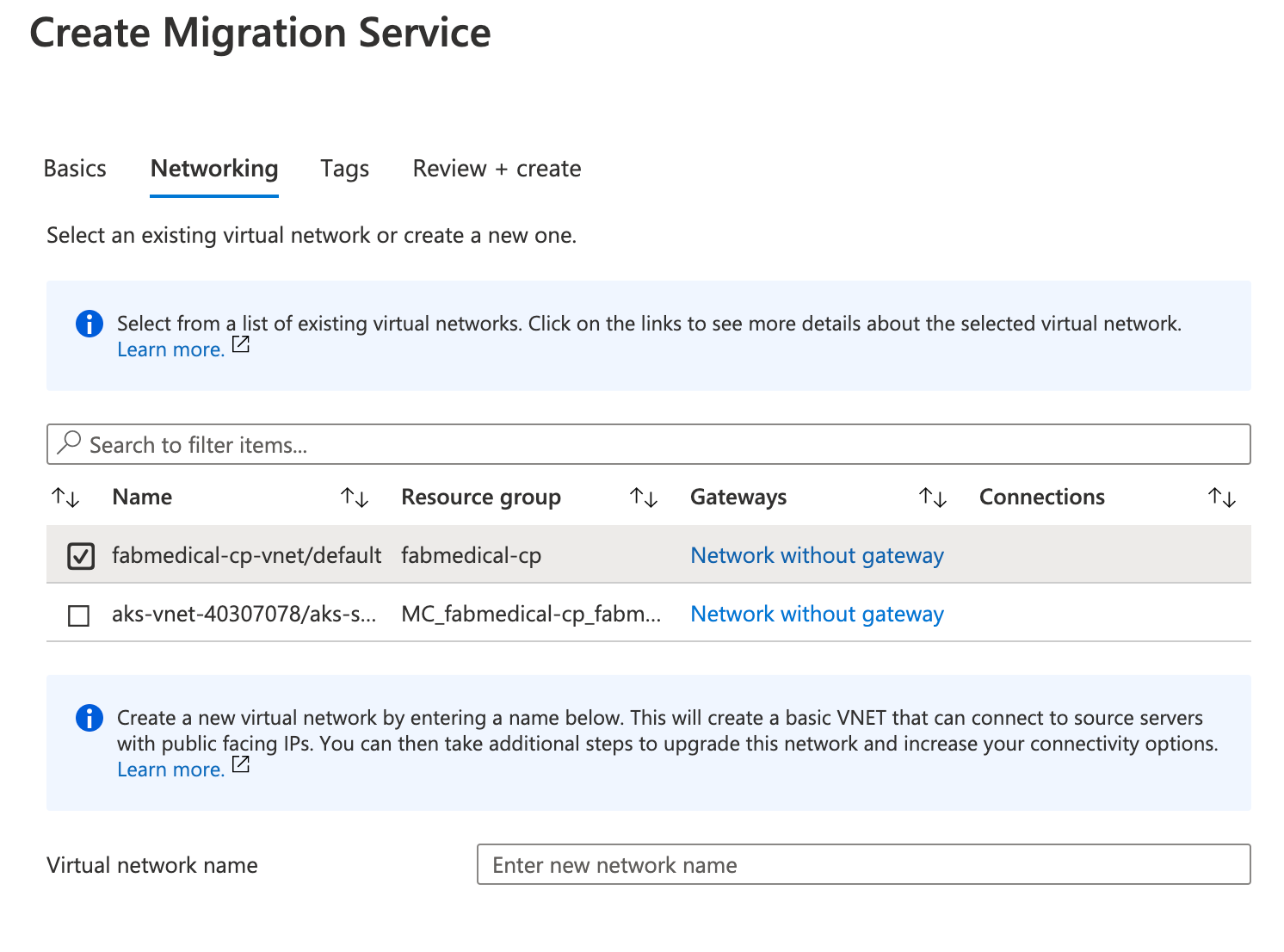
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-marketplace-create.png)

На вкладке «Основные» панели «Создание службы миграции» введите следующие значения:

* + Группа ресурсов: выберите группу ресурсов, созданную с помощью этой лабораторной работы.
  + Название службы миграции: введите имя, например fabmedical [SUFFIX].
  + Расположение: выберите регион Azure, используемый для группы ресурсов.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-create-basics.png)

1. Выберите Далее: Сеть >>.
2. На вкладке «Сеть» выберите виртуальную сеть в группе ресурсов fabmedical- [SUFFIX]..

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-create-networking.png)

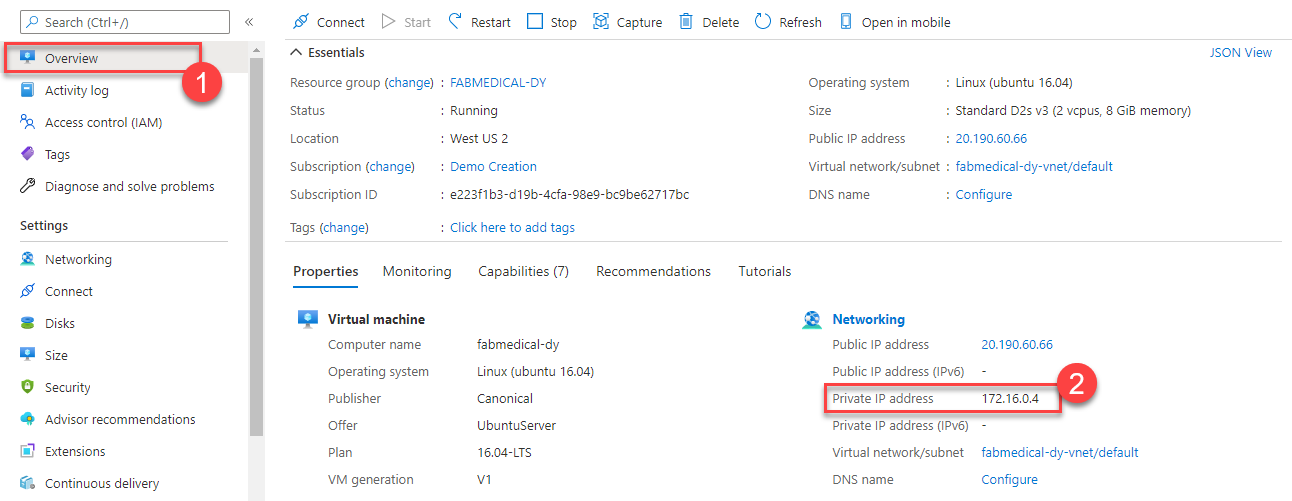
1. 6. Выберите Обзор + создать.
2. 7. Выберите «Создать», чтобы создать экземпляр службы миграции базы данных Azure.

Предоставление услуги может занять 5-10 минут.

**Задача 3. Перенести данные в Azure Cosmos DB.**

В этой задаче вы создадите проект миграции в службе миграции базы данных Azure, а затем перенесете данные из MongoDB в Azure Cosmos DB.

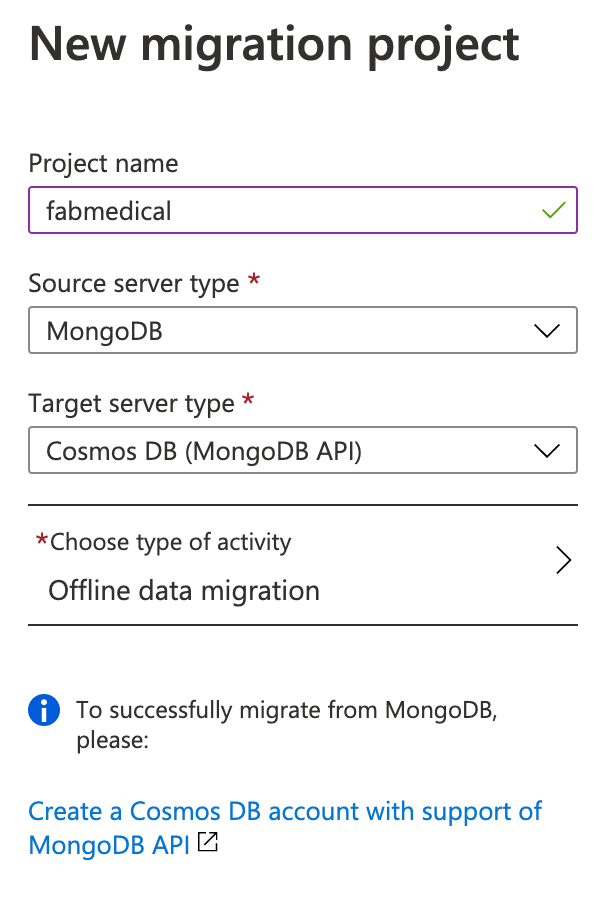
1. На портале Azure перейдите к виртуальной машине агента сборки и скопируйте частный IP-адрес (2). Вставьте содержимое в текстовый редактор по вашему выбору (например, Блокнот в Windows, пользователи macOS могут использовать TextEdit) для использования в будущем.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/agent-vm-private-ip-address.png)

2. На портале Azure перейдите к ранее подготовленной службе миграции базы данных Azure.

3. В колонке «Служба миграции базы данных Azure» выберите + Новый проект миграции на панели «Обзор».

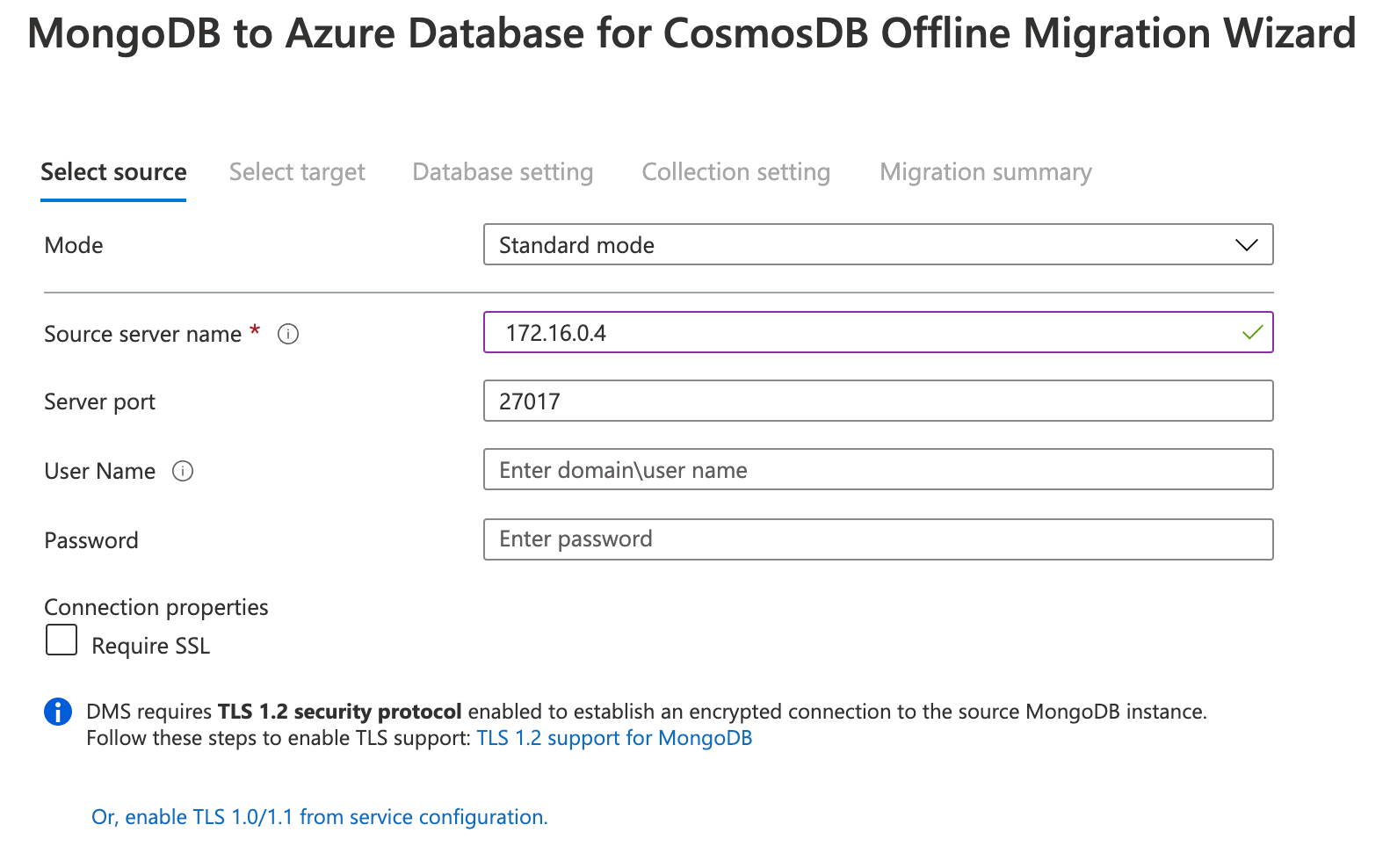
4. На панели «Новый проект миграции» введите следующие значения, затем выберите «Создать и запустить действие»:  
- Название проекта: fabmedical  
- Тип исходного сервера: MongoDB  
- Тип целевого сервера: CosmosDB (MongoDB API)  
- Выберите тип деятельности: offline data migration.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-new-migration-project.png)

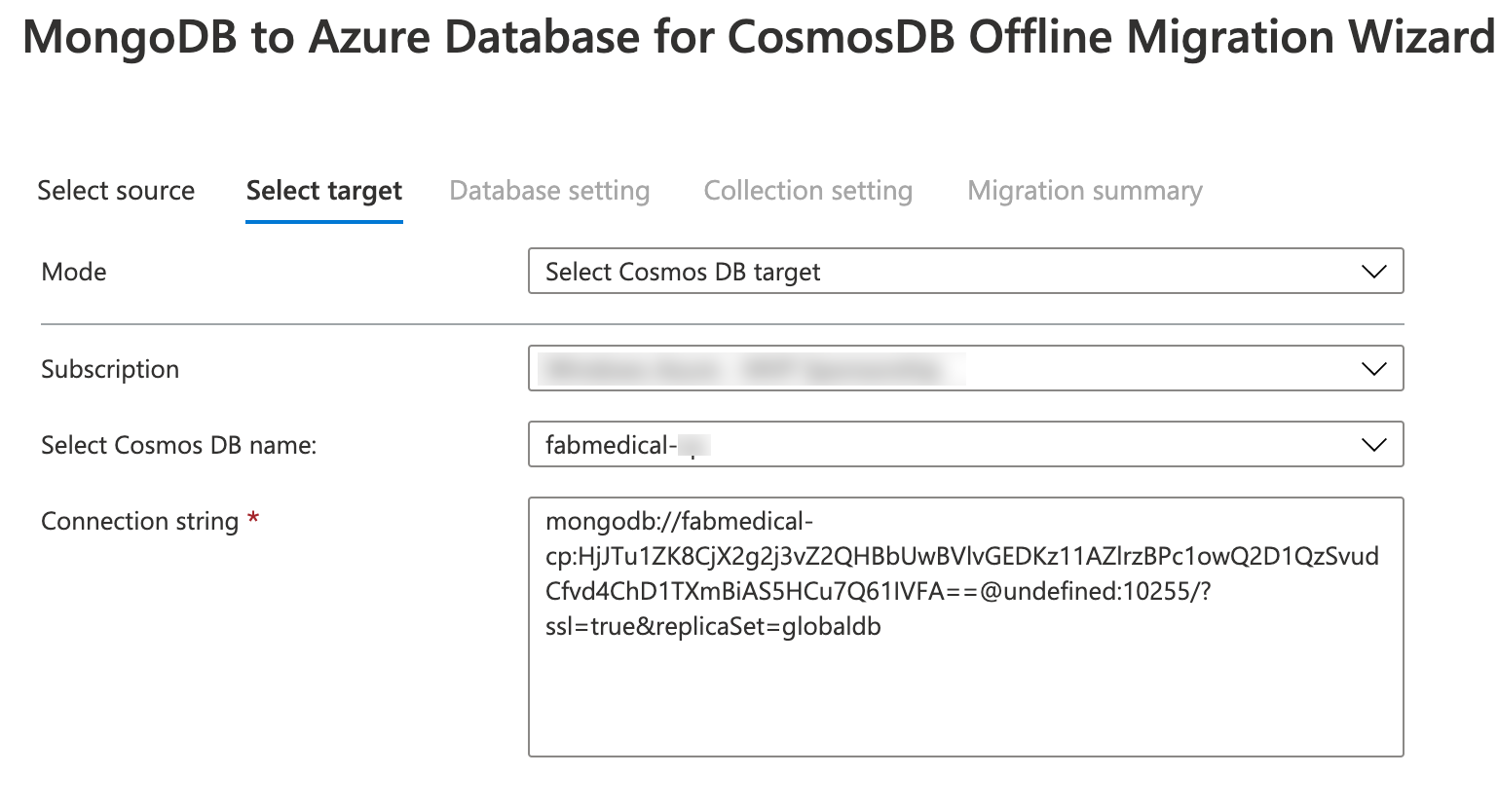
Примечание. Выбран тип действия «Автономная миграция данных», поскольку вы будете выполнять однократную миграцию с MongoDB на Cosmos DB. Кроме того, данные в базе данных не будут обновляться во время миграции. В производственном сценарии вам нужно будет выбрать тип деятельности проекта миграции, который наилучшим образом соответствует требованиям вашего решения.

1. На панели мастера автономной миграции MongoDB в базу данных Azure для CosmosDB введите следующие значения для вкладки Выбор источника.:
   * Mode: **Standard mode**
   * Source server name: Enter the Private IP Address of the Build Agent VM used in this lab.
   * Server port: 27017
   * Require SSL: Unchecked

Примечание. Оставьте поля «Имя пользователя» и «Пароль» пустыми, поскольку для экземпляра MongoDB на виртуальной машине с агентом сборки для этой лабораторной работы не включена проверка подлинности. Служба миграции базы данных Azure подключена к той же виртуальной сети, что и виртуальная машина агента сборки, поэтому она может обмениваться данными в виртуальной сети напрямую с виртуальной машиной, не открывая службу MongoDB в Интернете. В производственных сценариях вы всегда должны включать аутентификацию на MongoDB.

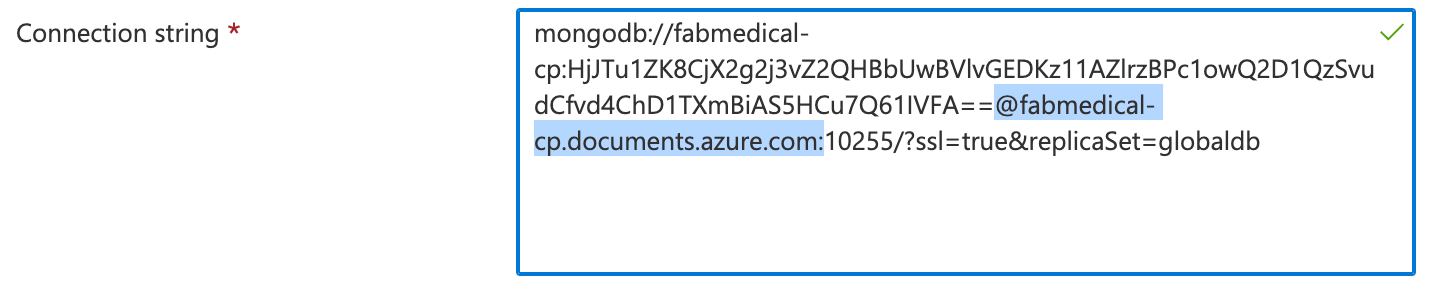
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-select-source.png)

1. Выберите Далее: Выбрать цель >>.
2. На панели «Выбрать цель» выберите следующие значения.:
   * Mode: **Select Cosmos DB target**
   * Subscription: Select the Azure subscription you're using for this lab.
   * Выберите имя Cosmos DB: выберите экземпляр fabmedical- [SUFFIX] Cosmos DB..

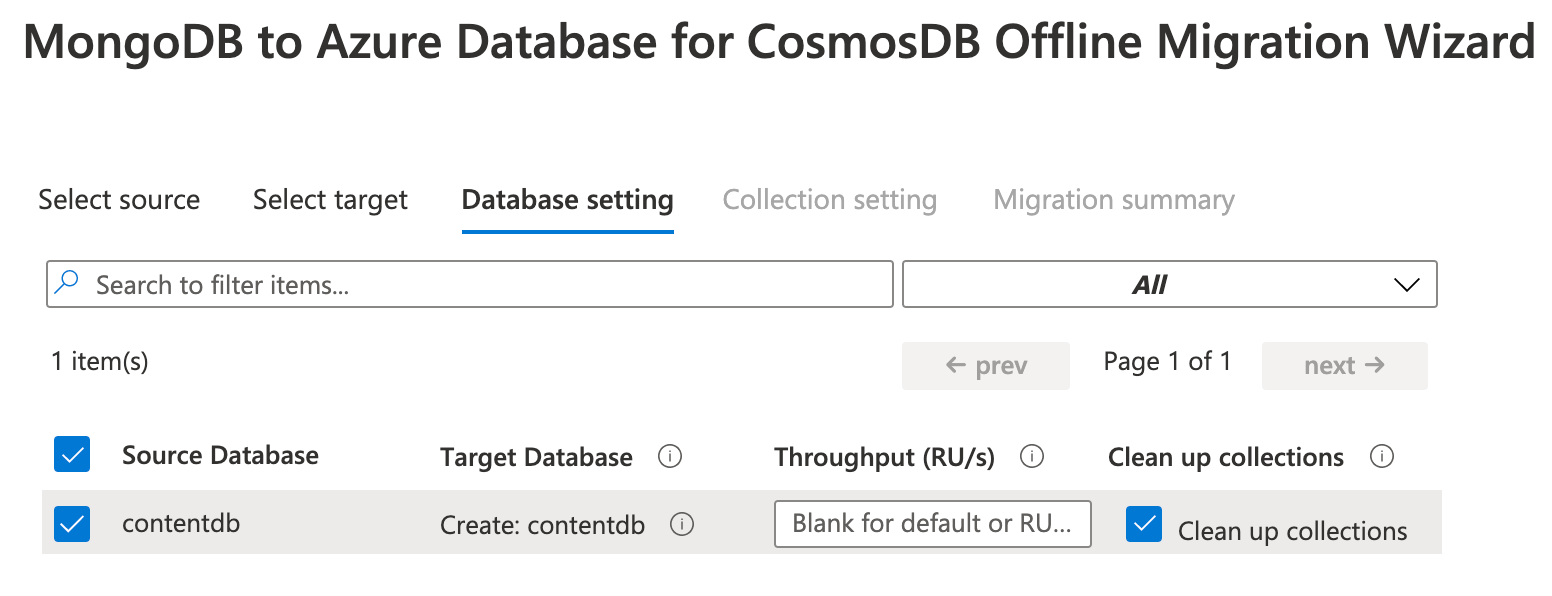
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-select-target.png)

Обратите внимание, что строка подключения автоматически заполнится ключом для вашего экземпляра Azure Cosmos DB.

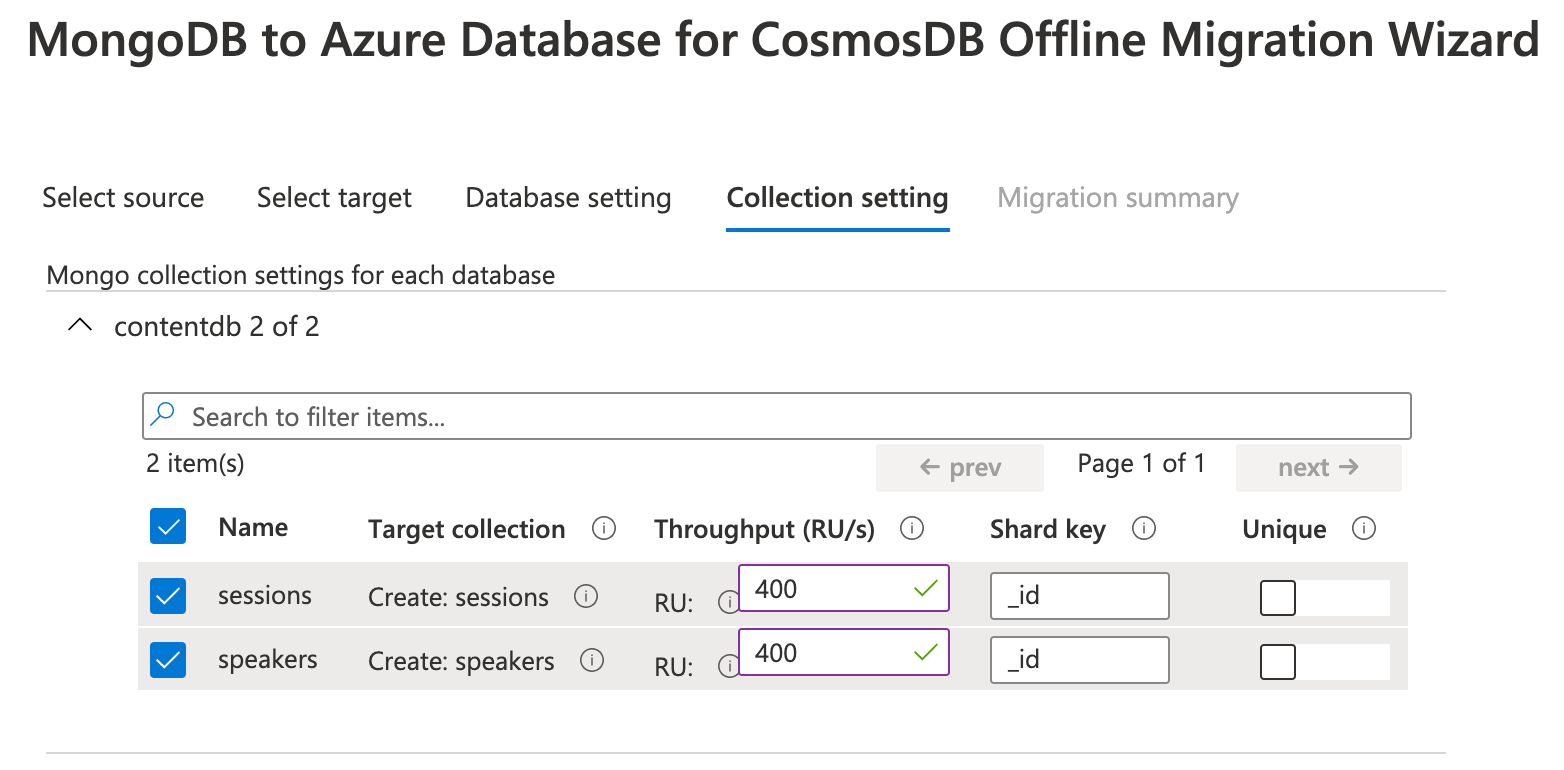
1. Измените строку подключения, заменив @undefined: на @ fabmedical- [SUFFIX] .documents.azure.com: так, чтобы имя DNS соответствовало экземпляру Azure Cosmos DB. Обязательно замените [СУФФИКС].

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-select-target-connection-string.png)

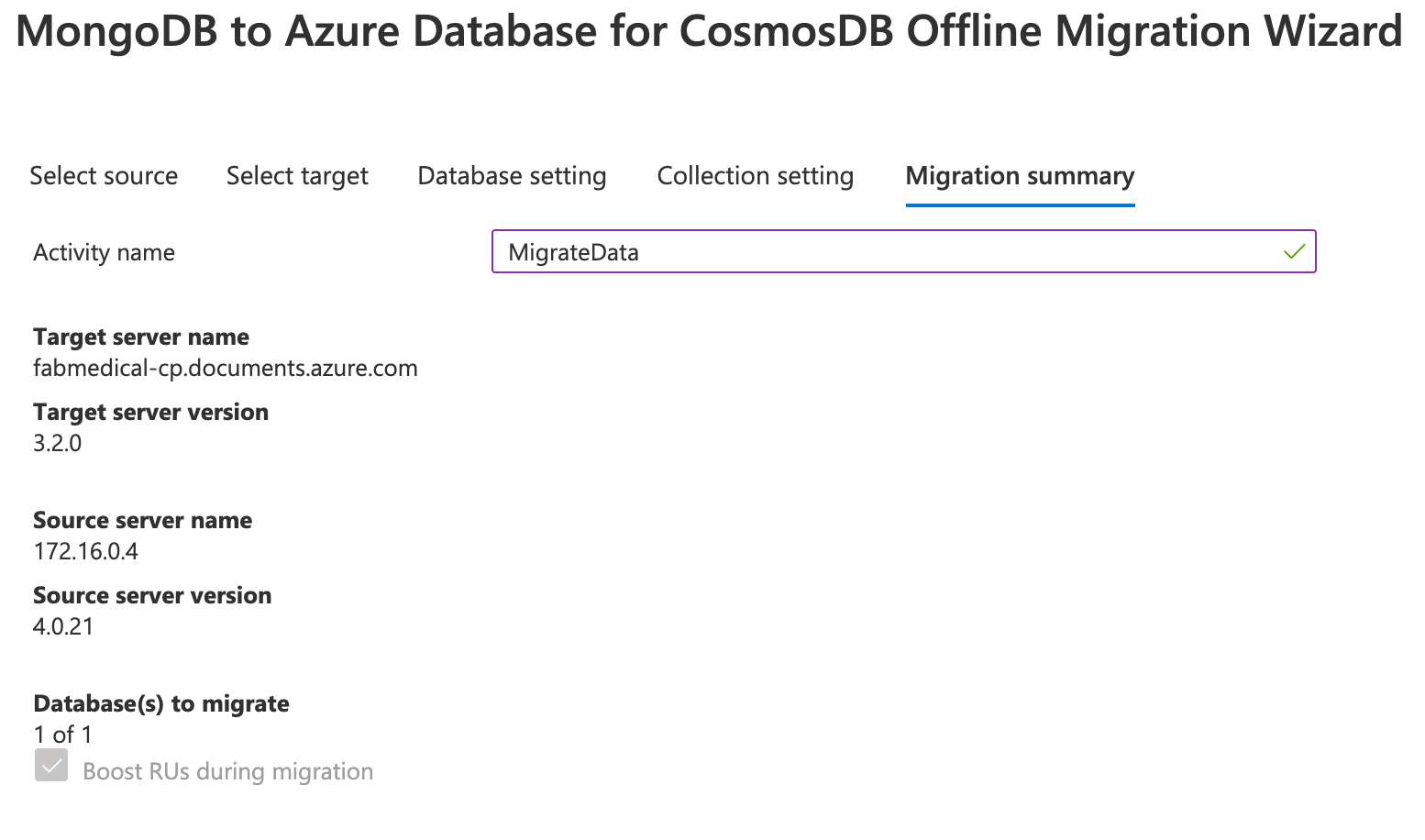
1. Выберите **Next: Database setting >>**.
2. На вкладке «Параметры базы данных» выберите исходную базу данных contentdb, чтобы эта база данных из MongoDB была перенесена в Azure Cosmos DB..

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-database-setting.png)

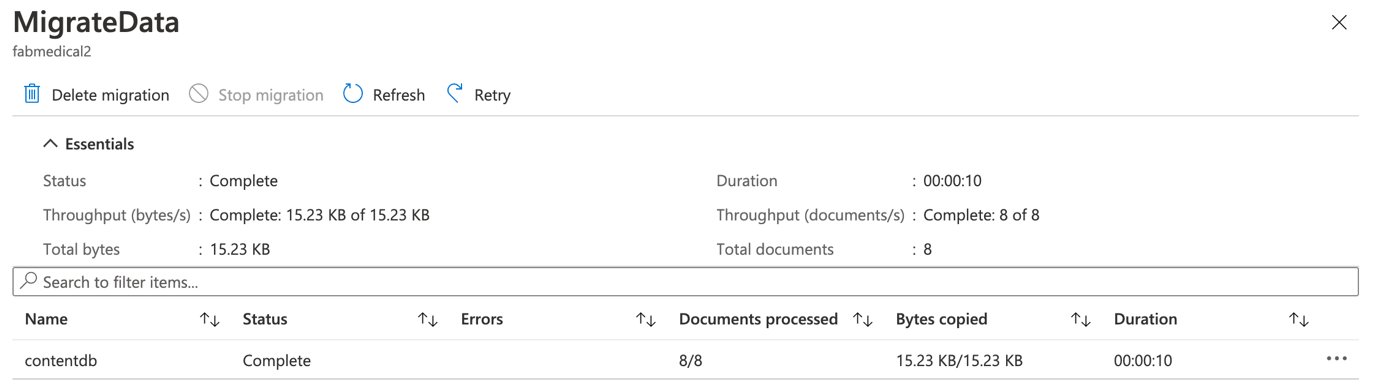
1. Выберите Далее: Настройка коллекции >>.
2. На вкладке «Параметры коллекции» разверните базу данных contentdb и убедитесь, что для миграции выбраны коллекции сеансов и выступающих. Кроме того, обновите пропускную способность (RU / s) до 400 для обеих коллекций.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-collection-setting.png)

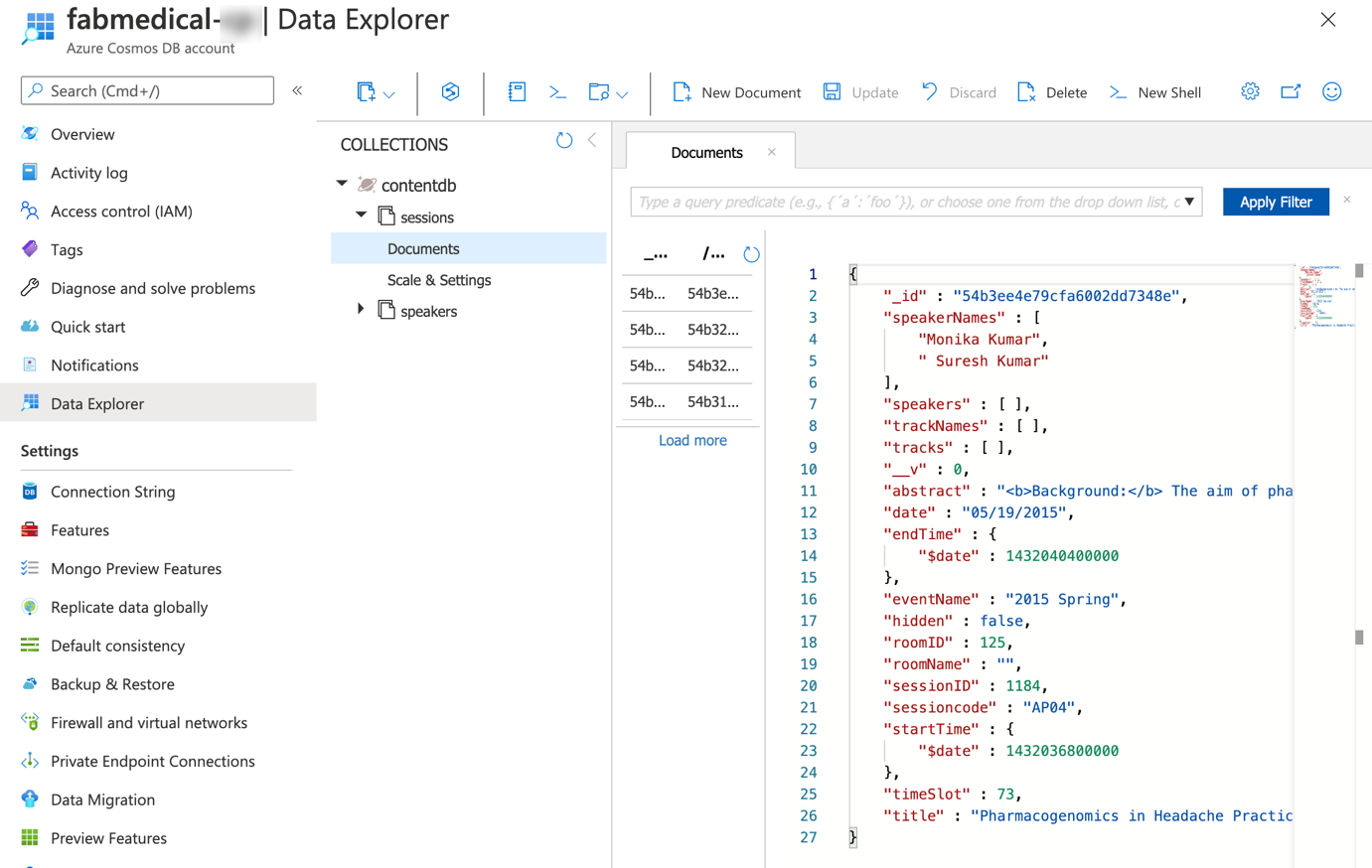
1. Выберите Далее: Сводка миграции >>.
2. На вкладке «Сводка миграции» введите MigrateData в поле «Имя действия», затем выберите «Начать миграцию», чтобы начать миграцию данных MongoDB в Azure Cosmos DB..

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-migration-summary.png)

1. Будет показан статус миграционной активности. Перенос займет всего несколько секунд. Выберите «Обновить», чтобы перезагрузить статус и убедиться, что он показывает статус «Завершено».

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-migrate-complete.png)

1. Чтобы убедиться, что данные были перенесены, перейдите к учетной записи Cosmos DB на портале Azure, затем выберите проводник данных. Вы увидите коллекции докладчиков и сессий, перечисленные в базе данных contentdb, и сможете изучить документы внутри.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/dms-confirm-data-in-cosmosdb.png)

**Задание 2. Развертывание решения в службе Azure Kubernetes.**

В этом задании вы подключитесь к кластеру службы Azure Kubernetes, который вы создали перед практической лабораторной работой, и развернете приложение Docker в кластере с помощью Kubernetes.

**Задача 1: туннель в кластер службы Azure Kubernetes**

В этой задаче вы соберете необходимую информацию о кластере службы Azure Kubernetes для подключения к кластеру и выполните команды для подключения к панели управления Kubernetes из облачной оболочки.

Примечание. Следующие задачи должны выполняться в облачной оболочке, а не на машине сборки, поэтому отключитесь от машины сборки, если она все еще подключена.Verify that you are connected to the correct subscription with the following command to show your default subscription:

az account show

* + Если вы не подключены к правильной подписке, укажите свои подписки, а затем установите подписку по ее идентификатору с помощью следующих команд (аналогично тому, что вы делали в облачной оболочке перед лабораторной работой):

az account list

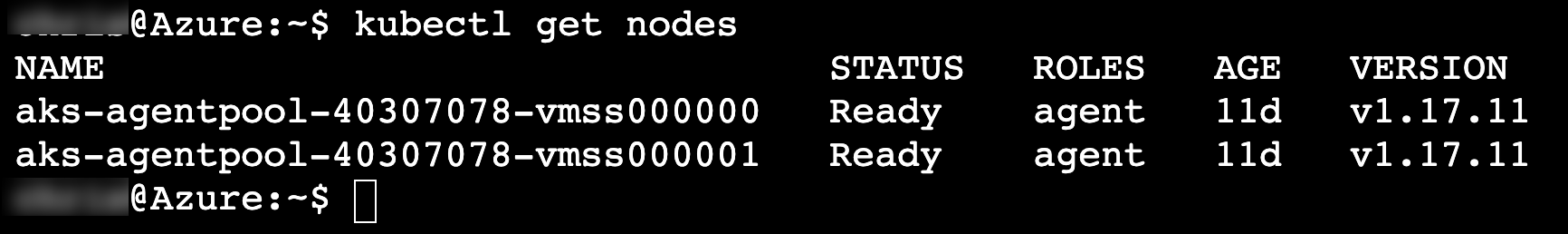
az account set --subscription {id}

1. Настройте kubectl для подключения к кластеру Kubernetes:

az aks get-credentials -a --name fabmedical-SUFFIX --resource-group fabmedical-SUFFIX

1. Проверьте правильность конфигурации, выполнив простую команду kubectl для создания списка узлов.:

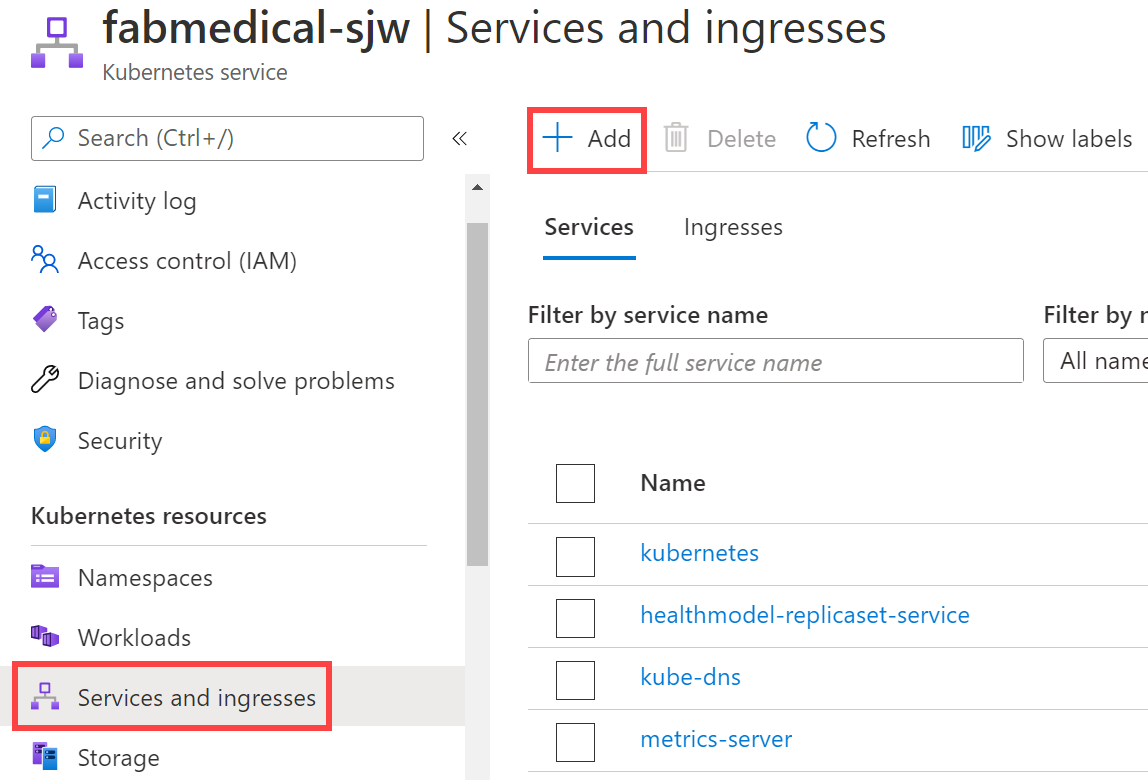
kubectl get nodes

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/image75.png)

**Задача 2. Развертывание службы с помощью портала Azure.**

В этой задаче вы развернете приложение API в кластере службы Azure Kubernetes с помощью портала Azure.

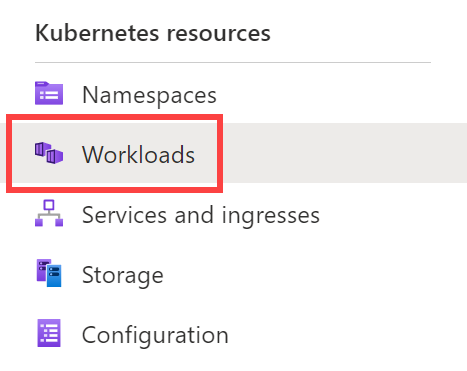
1. Сначала нам нужно определить Сервис для нашего API, чтобы приложение было доступно в кластере. В колонке AKS на портале Azure выберите Services and ingresses, а на вкладке Службы выберите + Добавить.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-04-04.png)

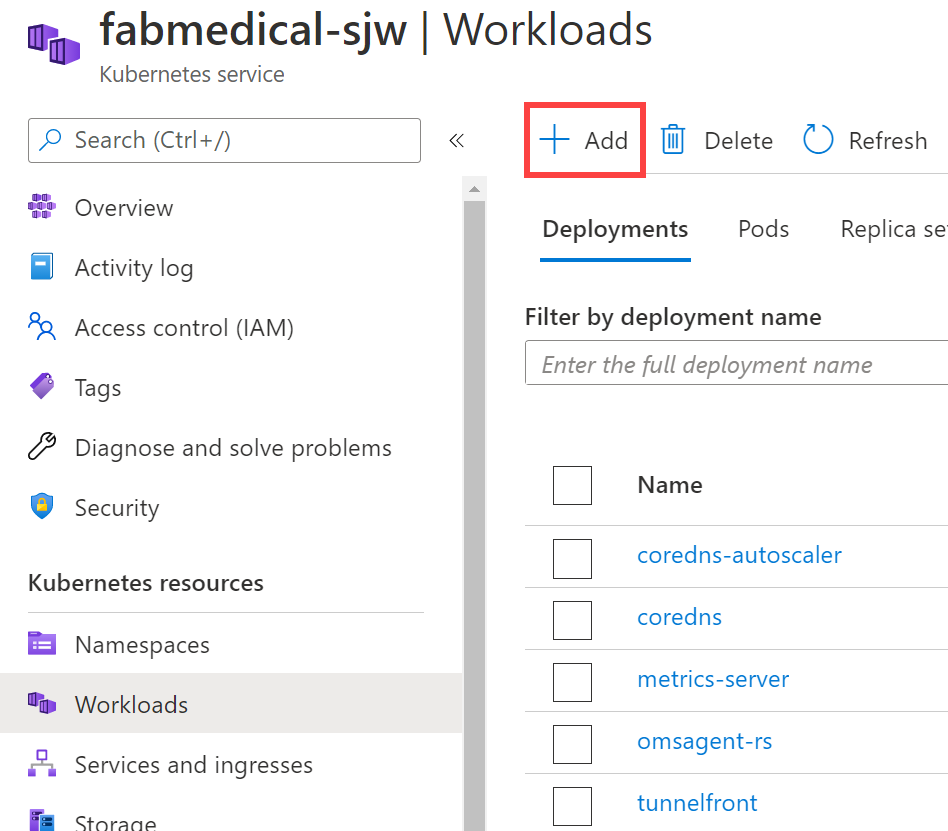
1. На экране «Добавить с YAML» вставьте следующий YAML и выберите «Добавить».
2. apiVersion: v1
3. kind: Service
4. metadata:
5. labels:
6. app: api
7. name: api
8. spec:
9. ports:
10. - name: api-traffic
11. port: 3001
12. protocol: TCP
13. targetPort: 3001
14. selector:
15. app: api
16. sessionAffinity: None

type: ClusterIP

1. Теперь выберите Workloads в разделе ресурсов Kubernetes на левой панели навигации.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-04-35.png)

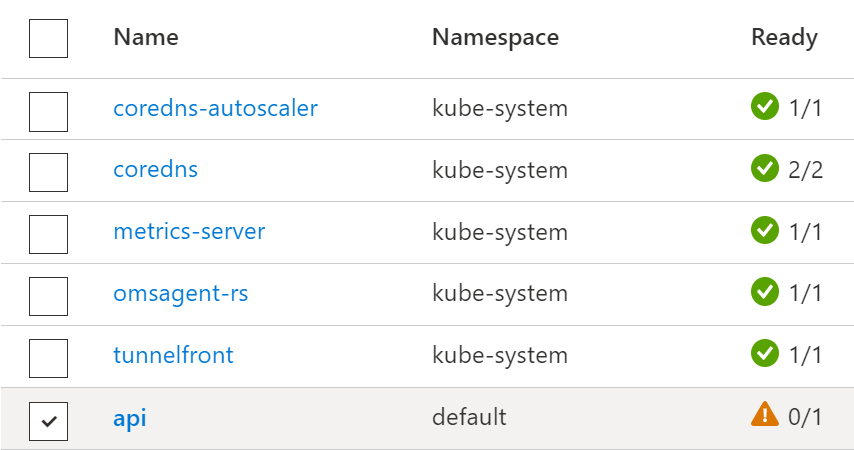
1. В представлении «Рабочие нагрузки», выбрав «Развертывания» (по умолчанию), нажмите «+ Добавить».

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-05-05.png)

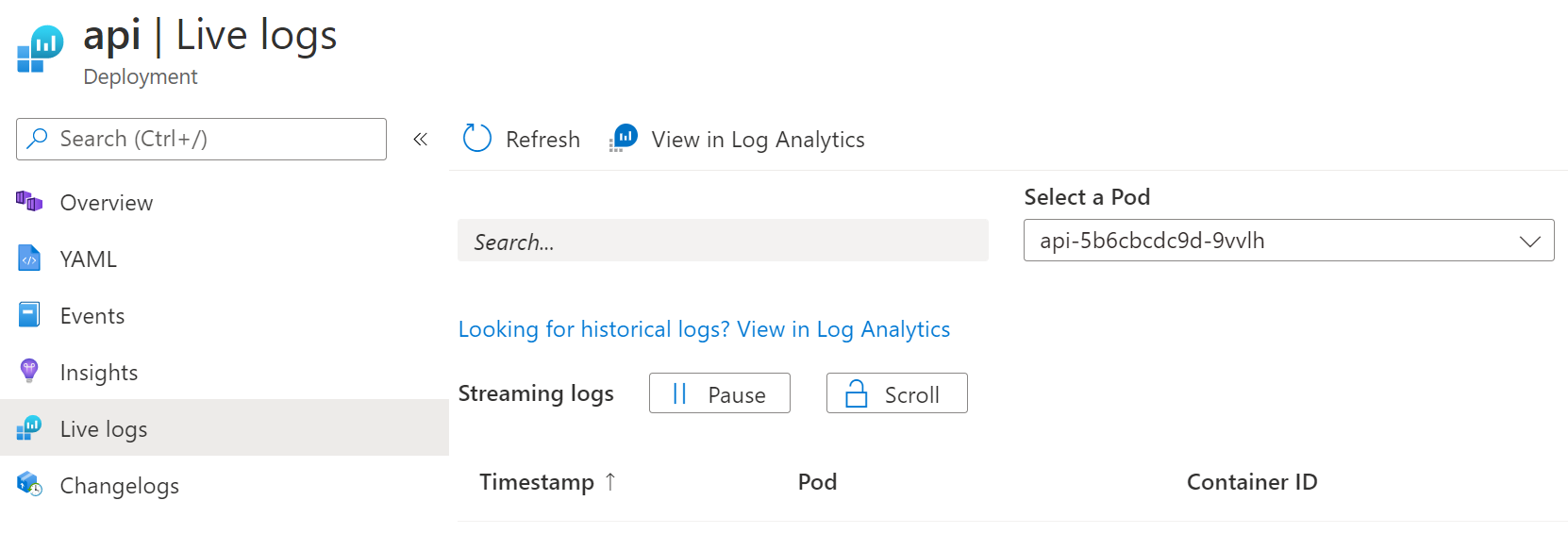
1. На загружающемся экране «Добавить с YAML» вставьте следующий YAML и обновите заполнитель [LOGINSERVER], указав имя экземпляра ACR.
2. apiVersion: apps/v1
3. kind: Deployment
4. metadata:
5. labels:
6. app: api
7. name: api
8. spec:
9. replicas: 1
10. selector:
11. matchLabels:
12. app: api
13. strategy:
14. rollingUpdate:
15. maxSurge: 1
16. maxUnavailable: 1
17. type: RollingUpdate
18. template:
19. metadata:
20. labels:
21. app: api
22. name: api
23. spec:
24. containers:
25. - name: api
26. image: [LOGINSERVER].azurecr.io/content-api
27. imagePullPolicy: Always
28. livenessProbe:
29. httpGet:
30. path: /
31. port: 3001
32. initialDelaySeconds: 30
33. periodSeconds: 20
34. timeoutSeconds: 10
35. failureThreshold: 3
36. ports:
37. - containerPort: 3001
38. hostPort: 3001
39. protocol: TCP
40. resources:
41. requests:
42. cpu: 1
43. memory: 128Mi
44. securityContext:
45. privileged: false
46. terminationMessagePath: /dev/termination-log
47. terminationMessagePolicy: File
48. dnsPolicy: ClusterFirst
49. restartPolicy: Always
50. schedulerName: default-scheduler
51. securityContext: {}

terminationGracePeriodSeconds: 30

1. Выберите Добавить, чтобы начать развертывание. Это может занять несколько минут, после чего вы увидите список развертывания.

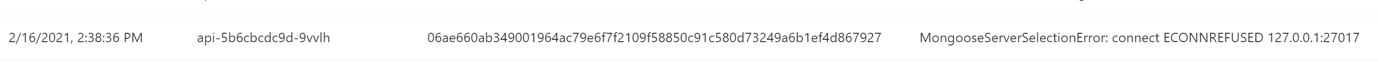
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-05-36.png)

1. Выберите развертывание api, чтобы открыть Deployment, выберите Live logs, а затем Pod из раскрывающегося списка. Через несколько секунд должны появиться живые журналы.

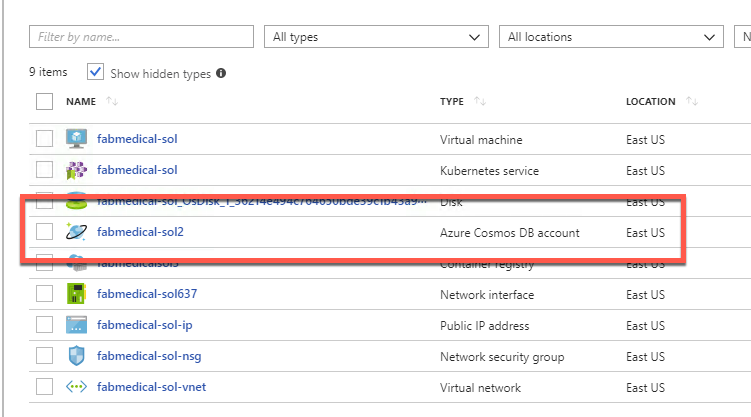
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-06-09.png)

Примечание: если журналы не отображаются, возможно, модуля больше не существует. Вы можете использовать представление в Log Analytics для просмотра журналов истории независимо от модуля.

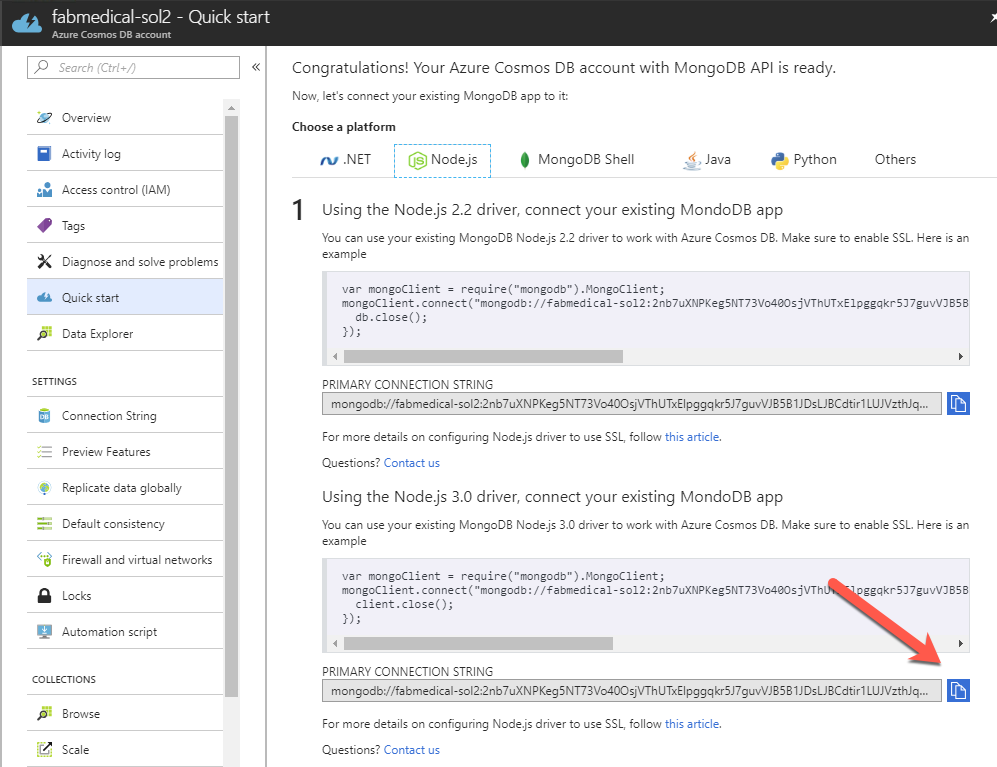
1. Если вы прокрутите журнал, вы увидите, что это указывает на то, что приложение content-api снова дает сбой, потому что не может найти MongoDB api для связи. Вы решите эту проблему, подключившись к Cosmos DB.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-07-13.png)

1. На портале Azure перейдите к своей группе ресурсов и найдите свою Cosmos DB. Выберите ресурс Cosmos DB, чтобы просмотреть подробности.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task1.9.png)

1. В разделе «Быстрый запуск» выберите вкладку Node.js и скопируйте строку подключения к Node.js 3.0.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task1.10.png)

1. Измените скопированную строку подключения, добавив базу данных contentdb к URL-адресу вместе с replicaSet globaldb. Результирующая строка подключения должна выглядеть как в примере ниже. Обратите внимание, что вам может потребоваться изменить URL-адрес конечной точки.

Примечание. Имя пользователя и пароль отредактированы для краткости.

mongodb://<USERNAME>:<PASSWORD>@fabmedical-<SUFFIX>.documents.azure.com:10255/contentdb?ssl=true&replicaSet=globaldb

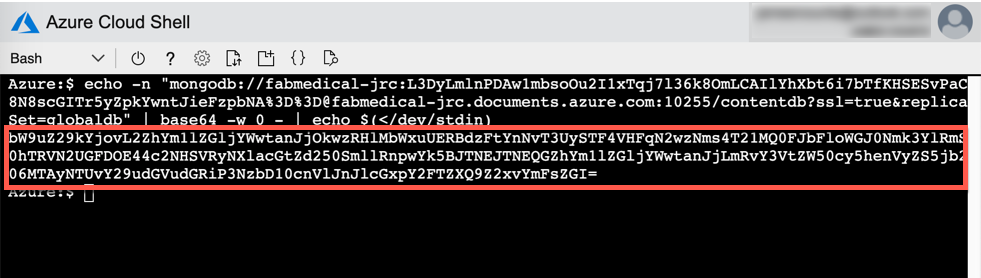
Примечание. В этой лабораторной работе использовалась версия API Cosmos DB 3.2. Однако, если ваша скопированная строка подключения имеет суффикс конечной точки .mongo.cosmos.azure.com, не стесняйтесь использовать его, поскольку он будет ссылаться на API 3.6. Лаборатория совместима с обеими версиями. Вот как строка подключения, показанная выше, будет отображаться с API версии 3.6:

mongodb://<USERNAME>:<PASSWORD>@fabmedical-<SUFFIX>.mongo.cosmos.azure.com:10255/contentdb?ssl=true&replicaSet=globaldb

1. Вы настроите секрет Kubernetes для хранения строки подключения и настроите приложение content-api для доступа к секрету. Во-первых, вы должны base64 закодировать секретное значение. Откройте окно Azure Cloud Shell и используйте следующую команду для кодирования строки подключения, а затем скопируйте выходные данные.

Примечание: двойные кавычки, окружающие строку подключения, необходимы для успешного создания требуемого вывода.

echo -n "[CONNECTION STRING VALUE]" | base64 -w 0 - | echo $(</dev/stdin)

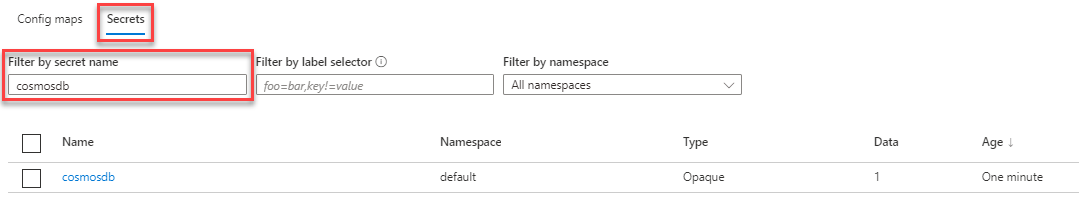
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/hol-2019-10-18_07-12-13.png)

1. Вернитесь в колонку AKS на портале Azure и выберите «Конфигурация» в разделе ресурсов Kubernetes. Выберите Секреты и нажмите + Добавить.
2. На экране «Добавить с YAML» вставьте следующий YAML и замените заполнитель на закодированную строку подключения из буфера обмена и выберите «Добавить». Обратите внимание, что YAML чувствителен к положению, поэтому вы должны убедиться, что отступы правильные при вводе или вставке.
3. apiVersion: v1
4. kind: Secret
5. metadata:
6. name: cosmosdb
7. type: Opaque
8. data:

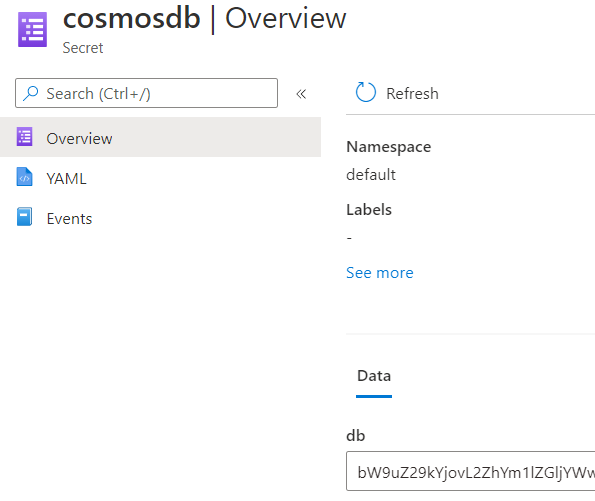
db: <base64 encoded value>

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-08-06.png)

1. Отсортируйте список секретов по имени, и теперь вы должны увидеть свой новый секрет.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/find-cosmosdb-secret.png)

1. Просмотрите сведения о секрете cosmosdb, выбрав его в списке.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-08-54.png)

1. Затем загрузите конфигурацию api, используя следующую команду в окне Azure Cloud Shell.:

kubectl get -o=yaml deployment api > api.deployment.yml

1. Отредактируйте загруженный файл с помощью редактора кода облачной оболочки:

code api.deployment.yml

Добавьте следующую конфигурацию среды в спецификацию контейнера под свойством изображения:

env:

- name: MONGODB\_CONNECTION

valueFrom:

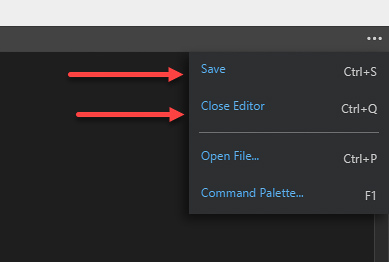
secretKeyRef:

name: cosmosdb

key: db

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task1.17.png)

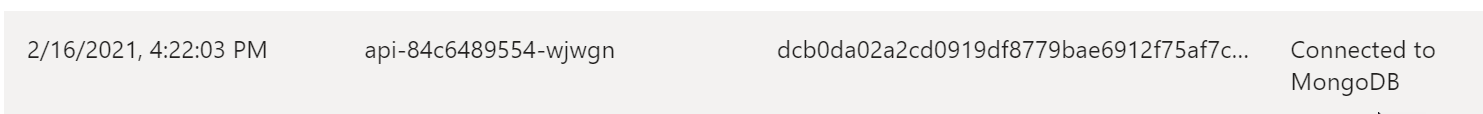
1. Сохраните изменения и закройте редактор.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task1.17.1.png)

1. Обновите развертывание API, используя kubectl для развертывания API.
2. kubectl delete deployment api

kubectl create -f api.deployment.yml

1. На портале Azure вернитесь к живым журналам (см. Шаг 5). Последний журнал должен отображаться как подключенный к MongoDB.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-25-17-09-24.png)

**Задача 3. Разверните сервис с помощью kubectl**

В этой задаче разверните веб-службу с помощью kubectl.

1. Откройте новую консоль Azure Cloud Shell.
2. Создайте текстовый файл с именем web.deployment.yml с помощью редактора Azure Cloud Shell..

code web.deployment.yml

1. Скопируйте и вставьте следующий текст в редактор:

apiVersion: apps/v1

kind: Deployment

metadata:

labels:

app: web

name: web

spec:

replicas: 1

selector:

matchLabels:

app: web

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

type: RollingUpdate

template:

metadata:

labels:

app: web

name: web

spec:

containers:

- image: [LOGINSERVER].azurecr.io/content-web

env:

- name: CONTENT\_API\_URL

value: http://api:3001

livenessProbe:

httpGet:

path: /

port: 3000

initialDelaySeconds: 30

periodSeconds: 20

timeoutSeconds: 10

failureThreshold: 3

imagePullPolicy: Always

name: web

ports:

- containerPort: 3000

hostPort: 80

protocol: TCP

resources:

requests:

cpu: 1000m

memory: 128Mi

securityContext:

privileged: false

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

dnsPolicy: ClusterFirst

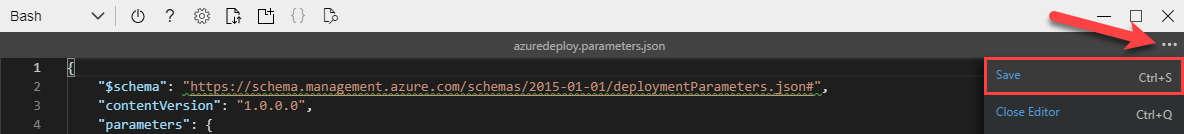
restartPolicy: Always

schedulerName: default-scheduler

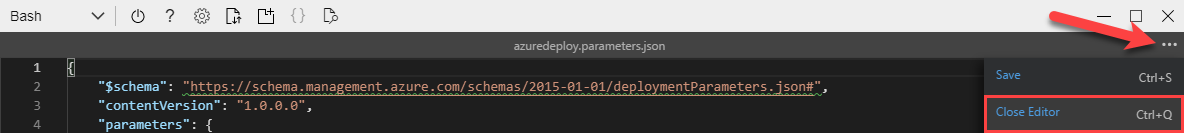
securityContext: {}

terminationGracePeriodSeconds: 30

1. Обновите запись [LOGINSERVER], чтобы она соответствовала имени вашего сервера входа в ACR.
2. Нажмите кнопку ... и выберите Сохранить..

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/b4-image62.png)

1. Снова нажмите кнопку… и выберите «Закрыть редактор»..

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/b4-image63.png)

1. Создайте текстовый файл с именем web.service.yml с помощью редактора Azure Cloud Shell.

code web.service.yml

1. Скопируйте и вставьте следующий текст в редактор:

apiVersion: v1

kind: Service

metadata:

labels:

app: web

name: web

spec:

ports:

- name: web-traffic

port: 80

protocol: TCP

targetPort: 3000

selector:

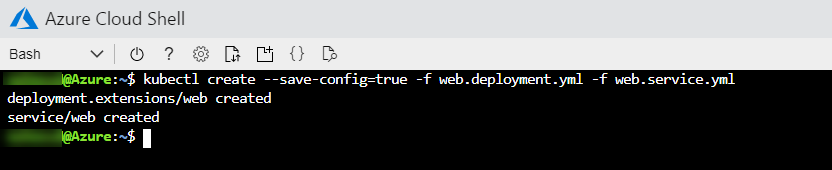
app: web

sessionAffinity: None

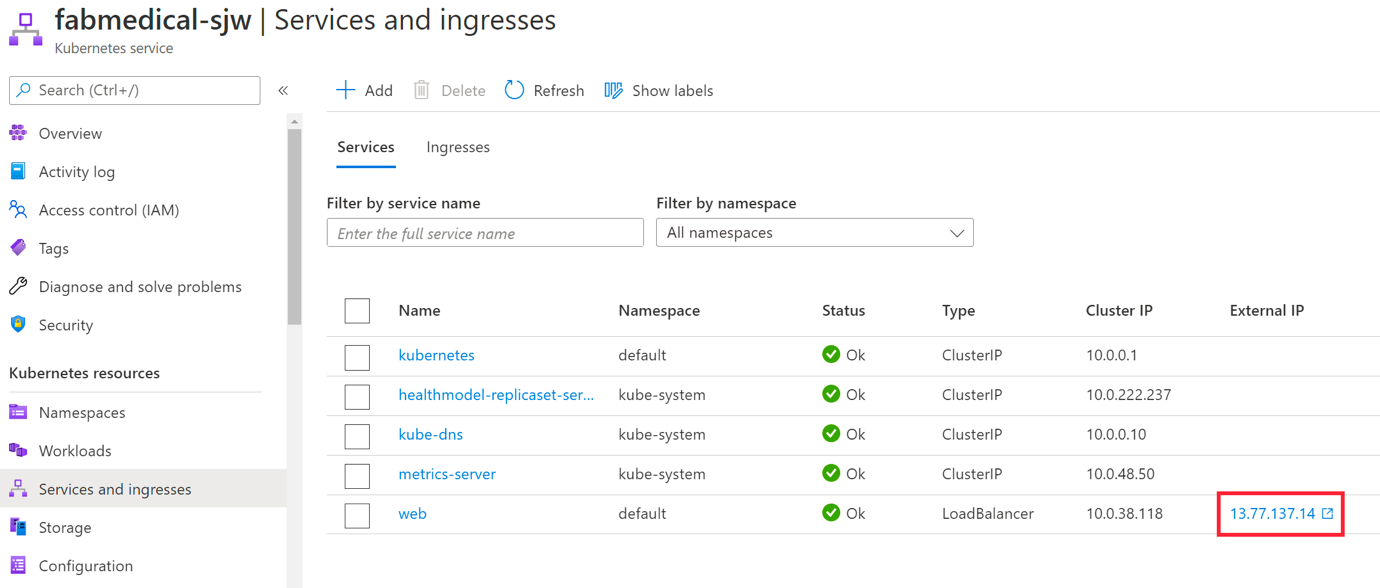
type: LoadBalancer

1. Сохраните изменения и закройте редактор.
2. Введите следующую команду, чтобы развернуть приложение, описанное в файлах YAML. Вы получите сообщение о том, что kubectl создал веб-развертывание и веб-службу.

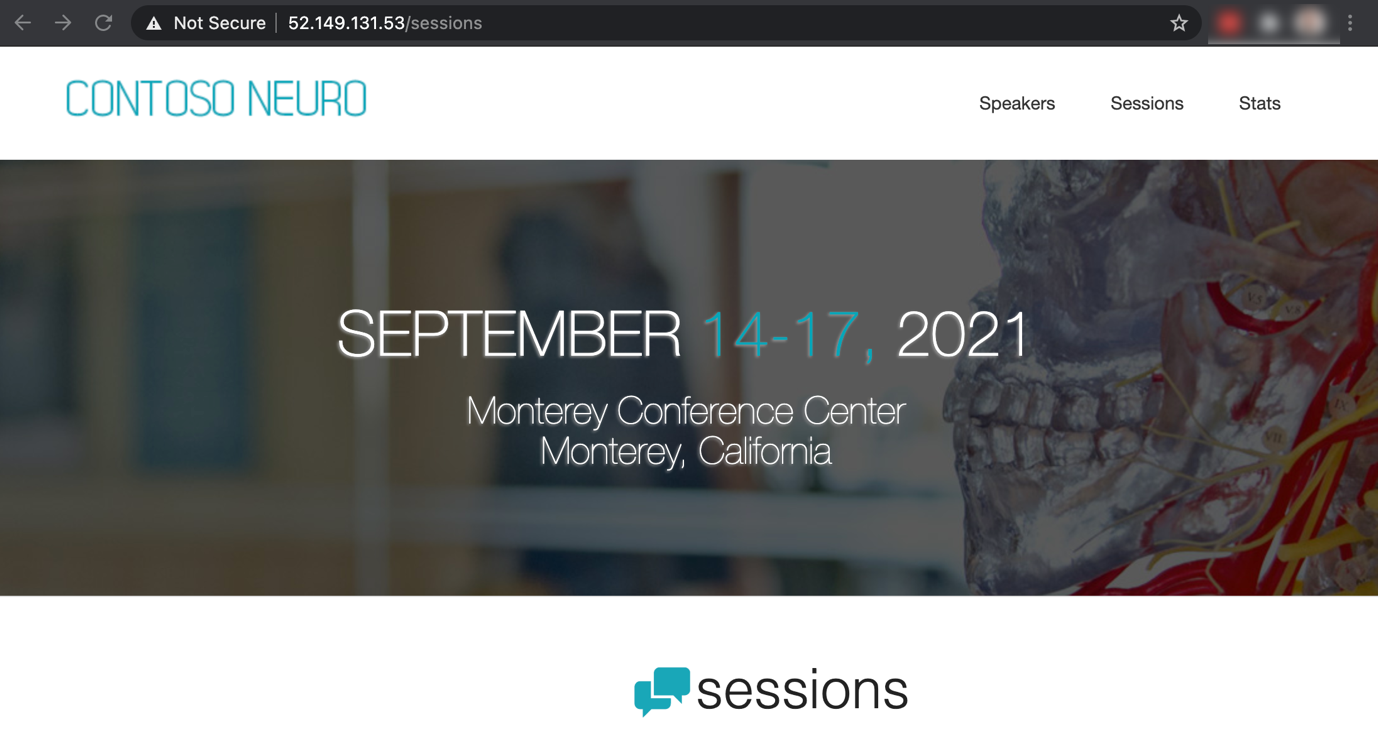
kubectl create --save-config=true -f web.deployment.yml -f web.service.yml

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/image93.png)

1. Вернитесь к колонке AKS на портале Azure. В меню навигации в разделе «Ресурсы Kubernetes» выберите представление «Службы и входящие данные». У вас должна быть возможность получить доступ к веб-сайту через внешнюю конечную точку.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/aks-resources-services-ingresses-view.png)

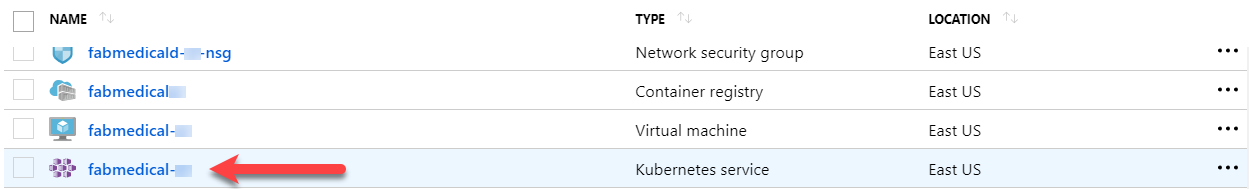
1. В верхней части навигации выберите ссылки на докладчиков и сеансы.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task3.11.png)

**Задача 4. Обзор Azure Monitor для контейнеров**

В этой задаче вы получите доступ и просмотрите различные журналы и панели мониторинга, доступные в Azure Monitor для контейнеров.

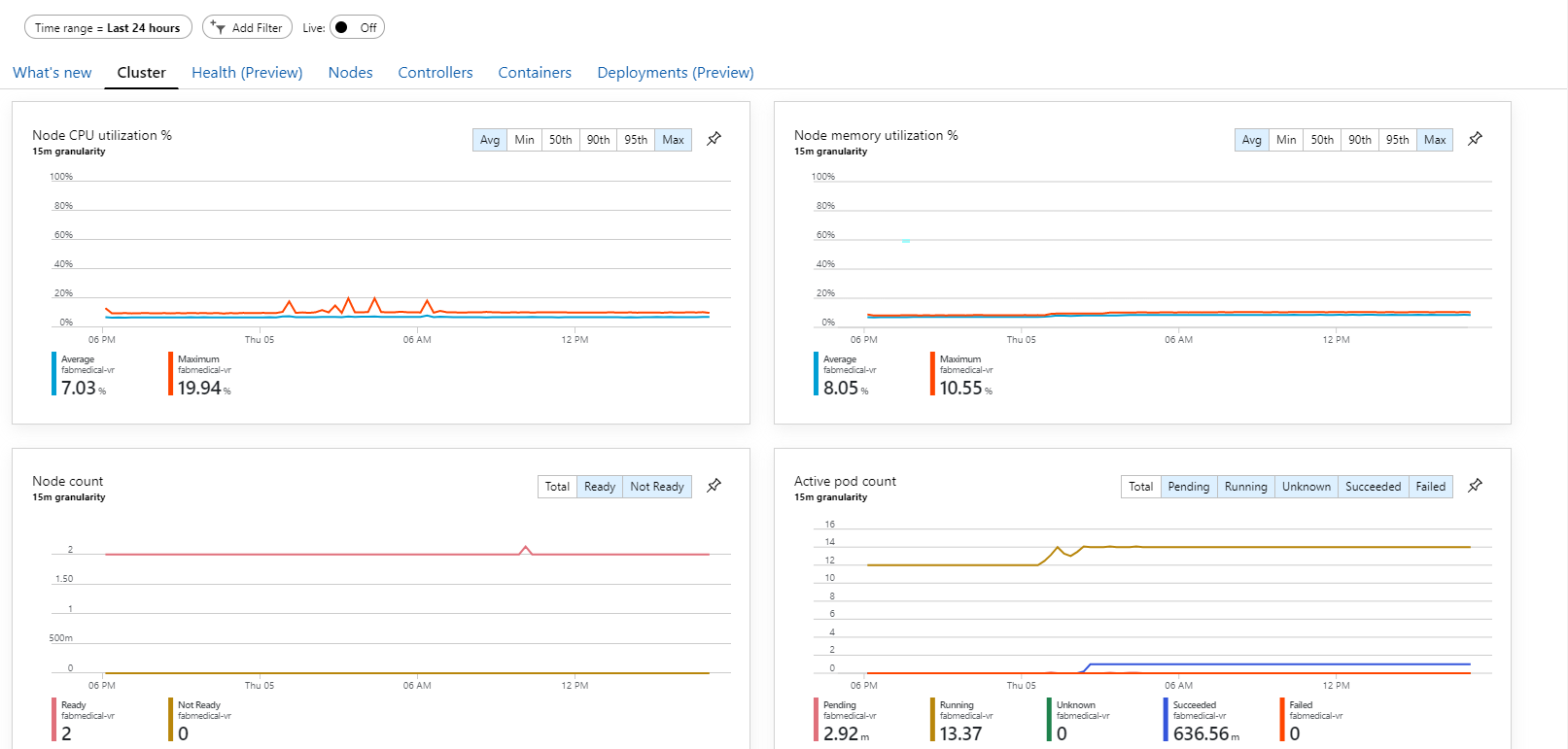
1. На портале Azure выберите созданную вами группу ресурсов с именем fabmedical-SUFFIX, а затем выберите ресурс Azure службы Kubernetes.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task8.1.png)

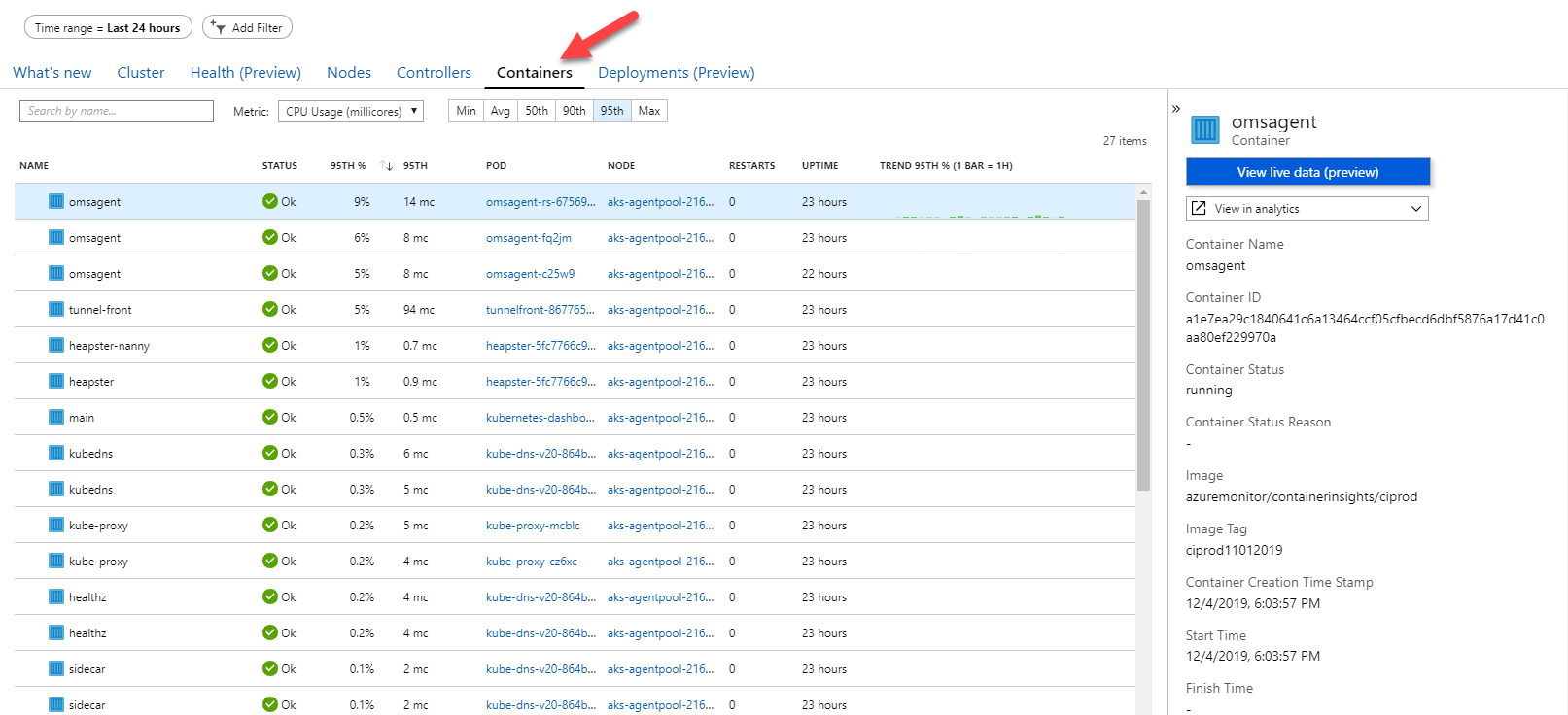
1. В колонке "Мониторинг" выберите Insights.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task8.2.png)

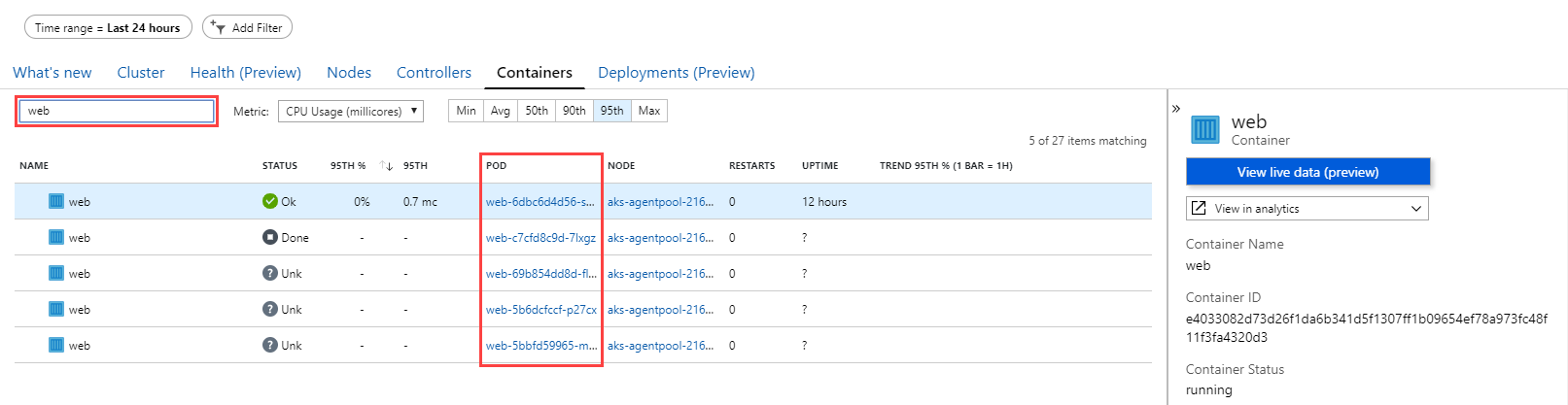
1. Просмотрите различные доступные информационные панели и более подробно изучите различные метрики и журналы, доступные для кластера, узлов, контроллеров и развернутых контейнеров.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex2-Task8.3.png)

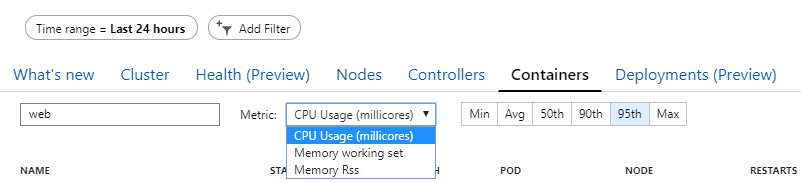
1. Чтобы просмотреть информационные панели «Контейнеры» и просмотреть более подробную информацию о каждом контейнере, выберите вкладку «Контейнеры».

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_1.png)

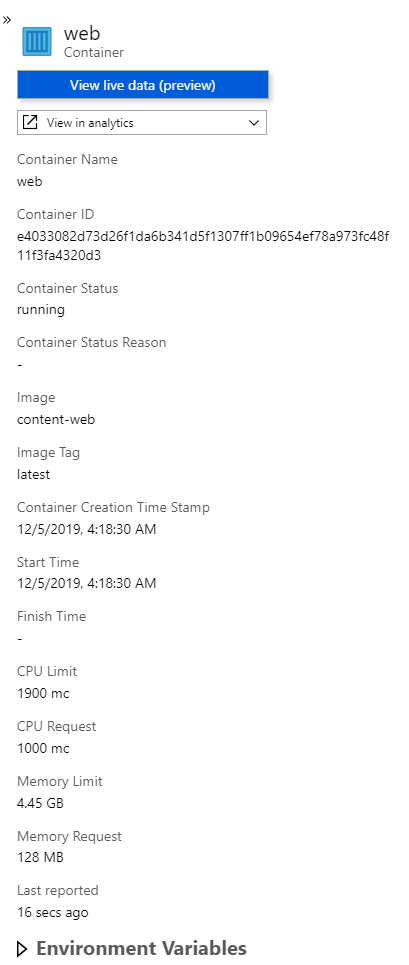
1. Теперь выполните фильтрацию по имени контейнера и найдите веб-контейнеры. Вы увидите все контейнеры, созданные в кластере Kubernetes, с именами модулей.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_3.png)

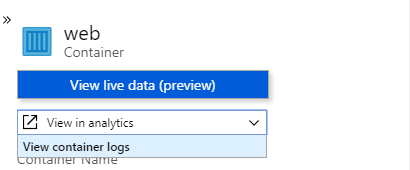
1. По умолчанию будет выбрана метрика использования ЦП, отображающая всю информацию о ЦП для выбранного контейнера. Чтобы переключиться на другую метрику, откройте раскрывающийся список метрик и выберите другую метрику.

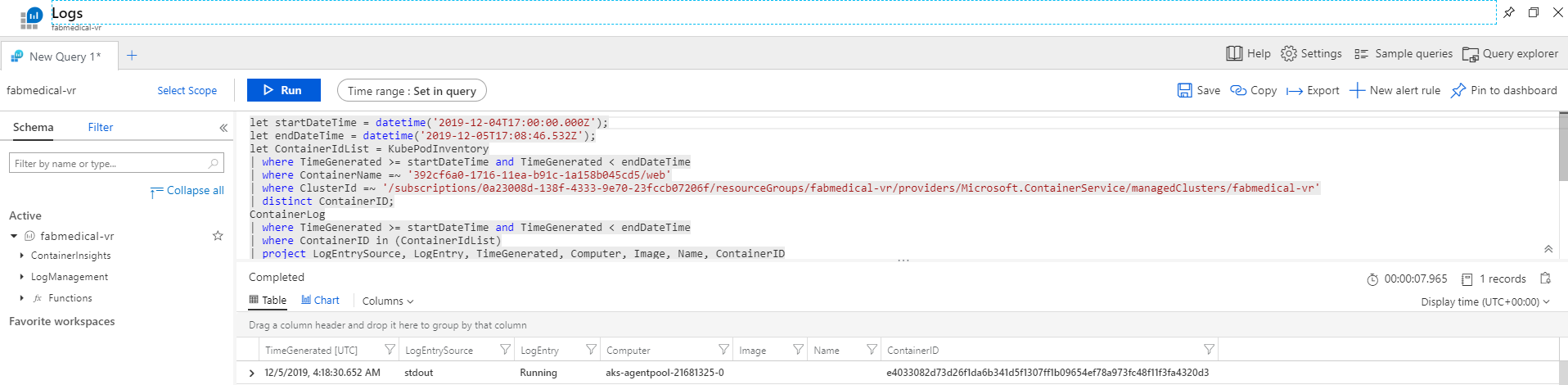
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_2.png)

1. После выбора любого модуля вся информация, относящаяся к выбранной метрике, будет отображаться на правой панели, и это будет иметь место при выборе любой другой метрики, детали будут отображаться на правой панели для выбранного пода.

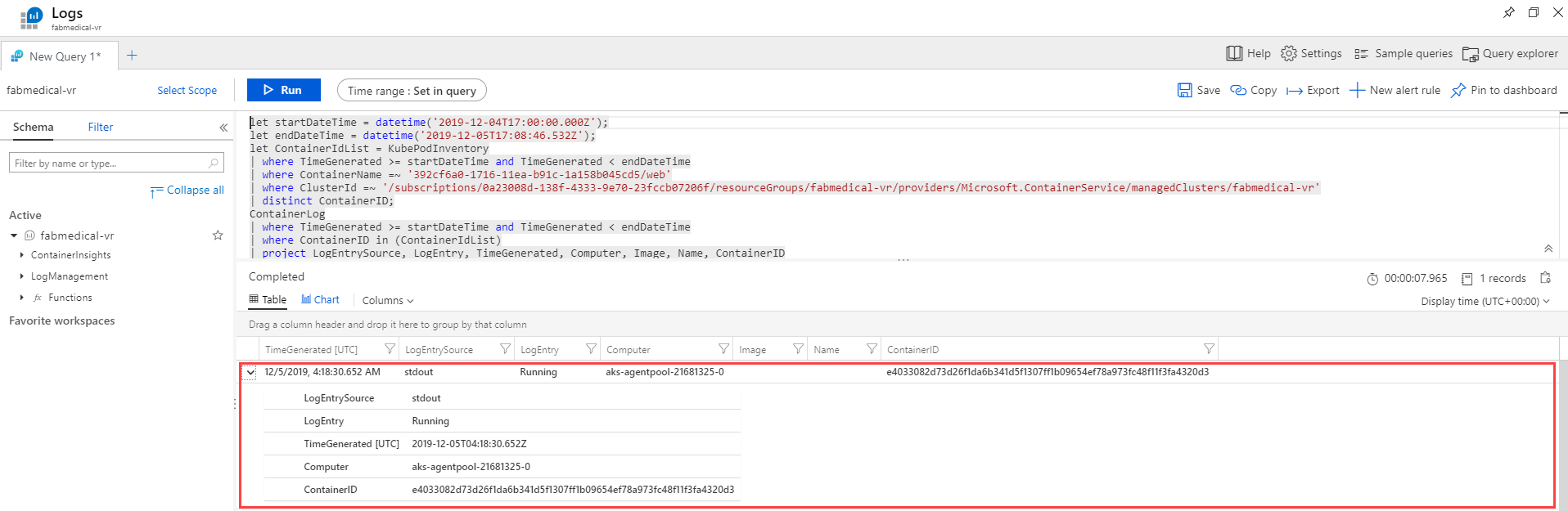
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_4.png)

1. Чтобы отобразить журналы для любого контейнера, просто выберите его и просмотрите правую панель, и вы найдете опцию «Просмотр журналов контейнера», в которой будут перечислены все журналы для этого конкретного контейнера.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_5.png)

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_6.png)

1. Для каждой записи журнала вы можете отобразить дополнительную информацию, развернув запись в журнале, чтобы просмотреть приведенные ниже сведения.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/monitor_7.png)

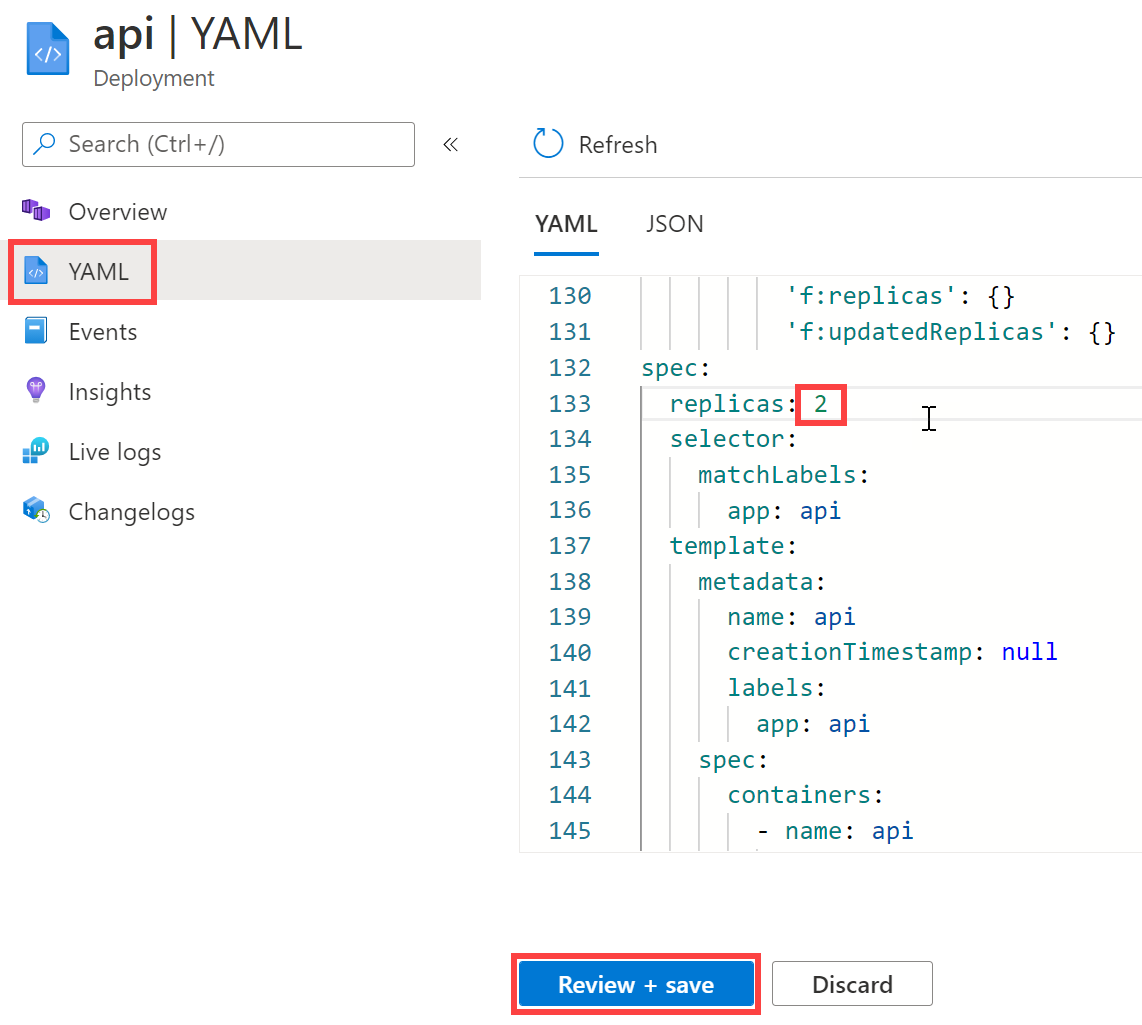
**Упражнение 3: масштабируйте приложение и тестируйте высокую доступность**

На этом этапе вы развернули один экземпляр контейнеров веб-служб и служб API. В этом упражнении вы увеличите количество экземпляров контейнера для веб-службы и масштабируете интерфейсную часть существующего кластера.

**Задача 1. Увеличение количества экземпляров службы на портале Azure.**

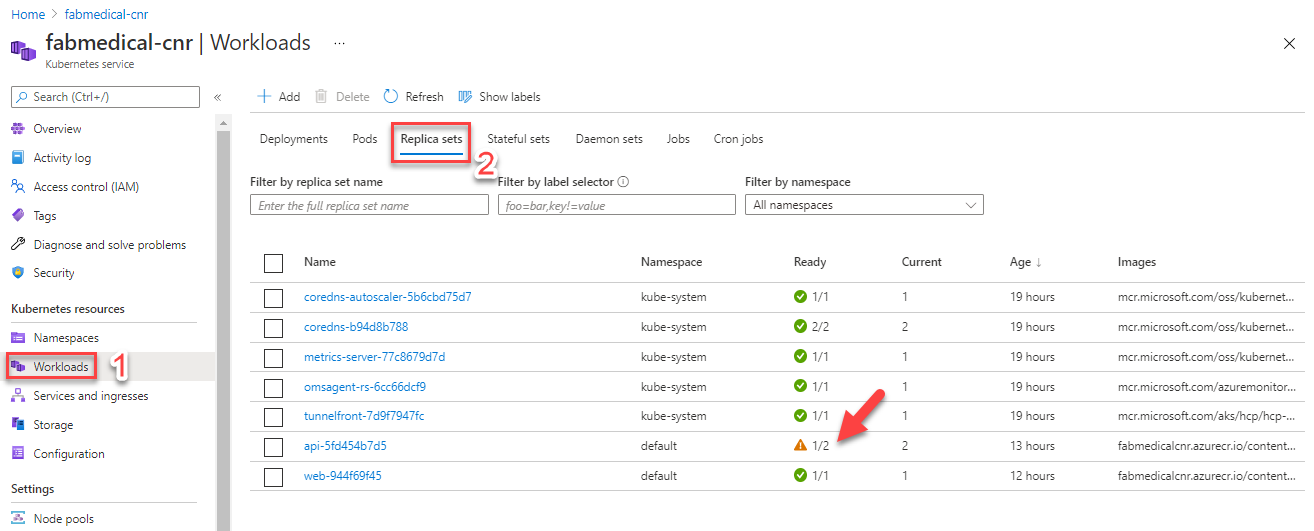
В этой задаче вы увеличите количество экземпляров для развертывания API в колонке AKS Azure Portal. Во время развертывания вы увидите изменение статуса.

1. В колонке AKS на портале Azure выберите «Рабочие нагрузки», а затем выберите развертывание API.
2. Выберите YAML в загружающемся окне и прокрутите вниз, пока не найдете реплики. Измените количество реплик на 2, а затем выберите Просмотр + сохранение. При появлении запроса установите флажок Подтвердить изменение манифеста и выберите Сохранить.

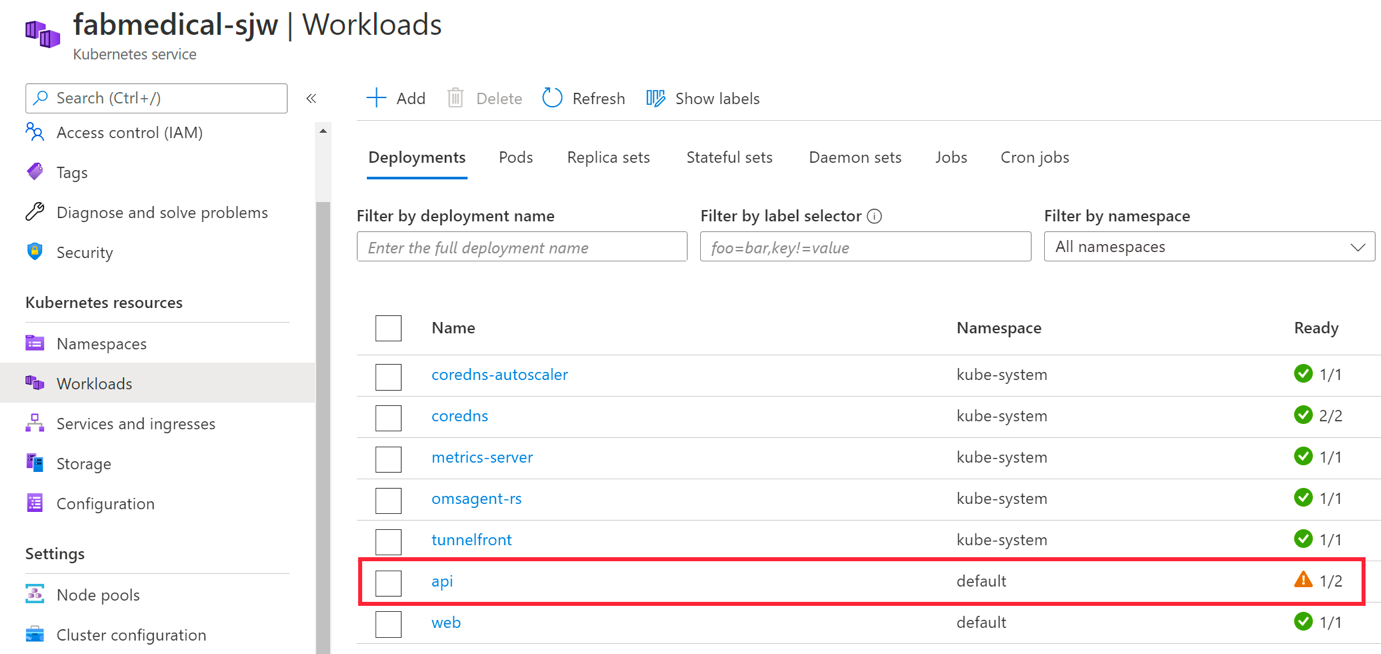
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-16-49-32.png)

Примечание: Если развертывание завершается быстро, вы можете не увидеть состояния ожидания развертывания на портале, как описано в следующих шагах.

1. В представлении «Набор реплик» для API вы увидите, что он развертывается, и что есть один работоспособный и один ожидающий экземпляры.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/api-replica-set.png)

1. В меню навигации выберите Рабочие нагрузки. Обратите внимание, что при развертывании api отображается предупреждение и отображается количество модулей 1 из 2 (показано как 1/2).

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-16-50-38.png)

Примечание. Если вы получаете сообщение об ошибке, указывающее, что ЦП недостаточно, это нормально. Мы увидим, как с этим справиться, в следующей задаче (подсказка: вы можете использовать параметр Insights на портале AKS Azure, чтобы просмотреть статус узла и просмотреть журналы событий Kubernetes).

На этом этапе мы представляем обзор состояния окружающей среды:

* + Одно развертывание и один набор реплик являются работоспособными для веб-службы.
  + Развертывание api и набор реплик находятся в состоянии предупреждения.
  + Два модуля исправны в пространстве имен "по умолчанию".

1. Откройте веб-приложение Contoso Neuro Conference. Приложение должно работать без ошибок при переходе на страницы докладчиков и сеансов.
   * Перейдите на страницу /stats. Вы увидите информацию о среде хостинга, включая:
     + webTaskId: идентификатор задачи для экземпляра веб-службы.
     + taskId: идентификатор задачи для экземпляра службы API.
     + hostName: идентификатор имени хоста для экземпляра службы API.
     + pid: идентификатор процесса для экземпляра службы API.
     + mem: некоторые индикаторы памяти, возвращаемые экземпляром службы API.
     + counters: счетчики для самой службы, возвращенные экземпляром службы API.
     + uptime: время безотказной работы службы API.

**Задача 2: устранение сбоев при инициализации реплик**

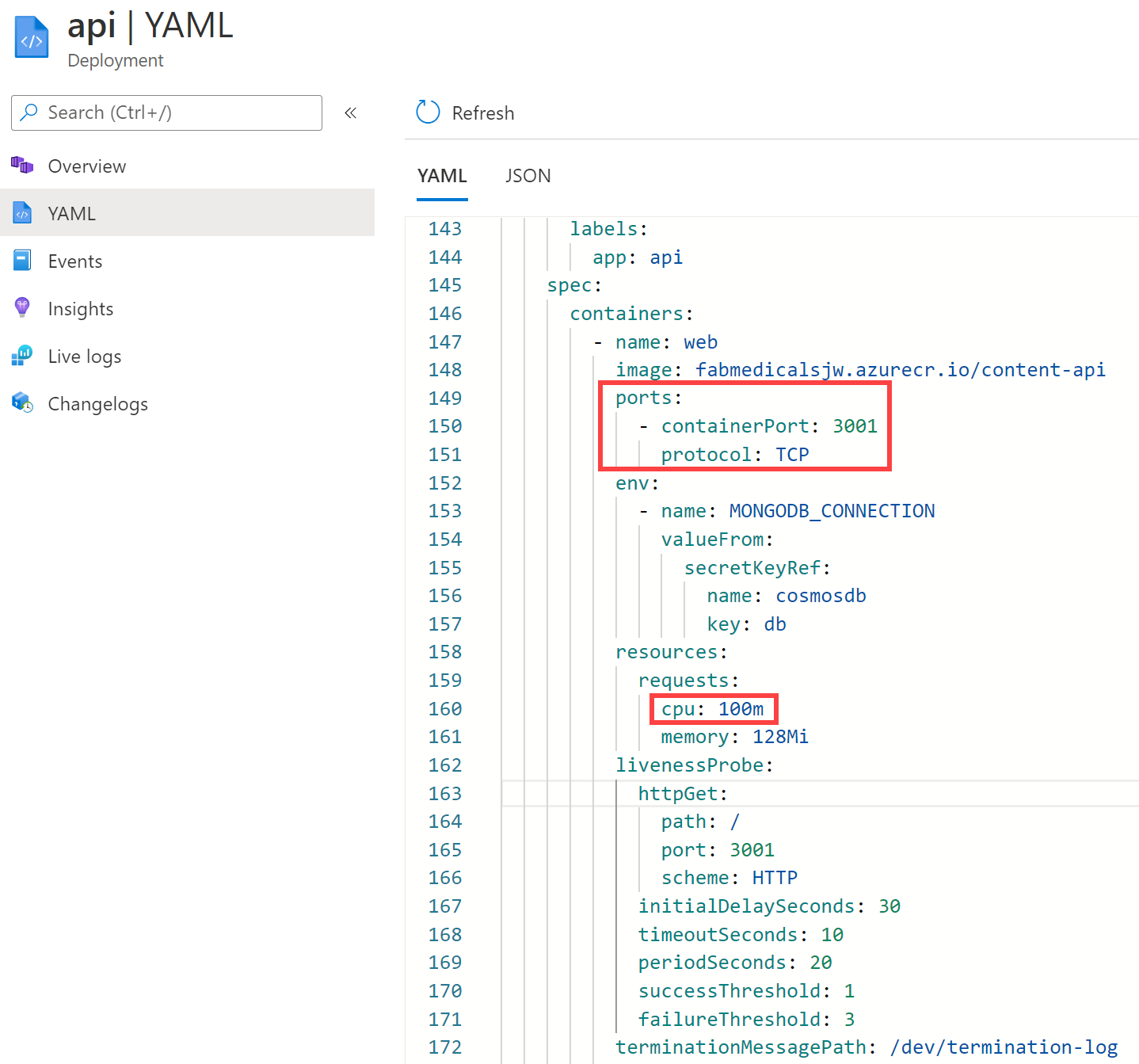
В этой задаче вы устраните неисправные реплики API. Эти сбои происходят из-за неспособности кластеров удовлетворить запрошенные ресурсы.

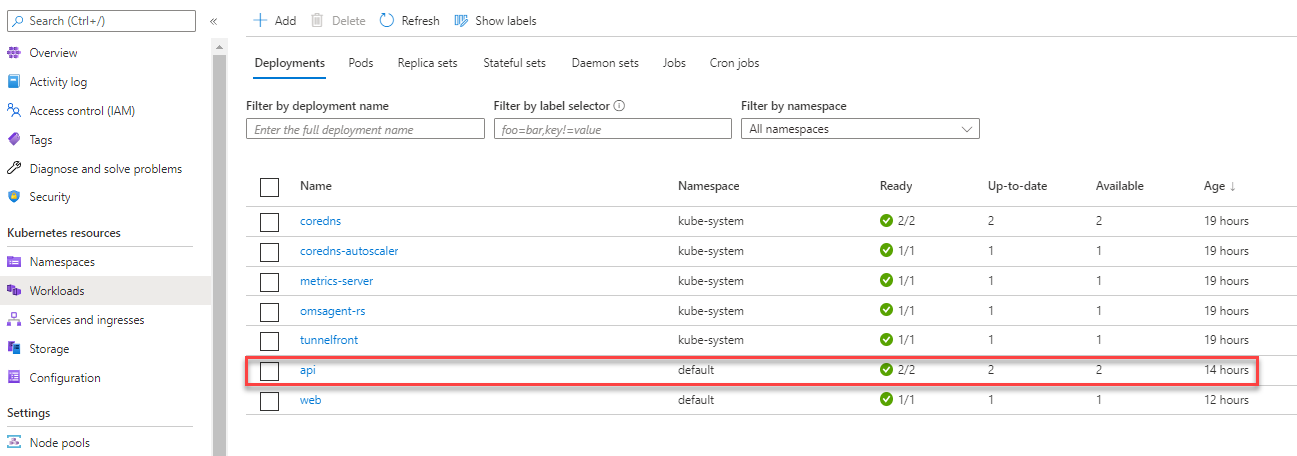
1. В колонке AKS на портале Azure выберите «Рабочие нагрузки», а затем выберите развертывание API. Выберите элемент навигации YAML.
2. Прокрутите экран YAML вниз и измените следующие элементы:
   * Измените порты и удалите hostPort. Два модуля не могут подключаться к одному и тому же порту хоста.
   * ports:
   * - containerPort: 3001

protocol: TCP

* + Измените ЦП и установите его на 100 м. ЦП разделен между всеми модулями на узле.
  + resources:
  + requests:
  + cpu: 100m

memory: 128Mi

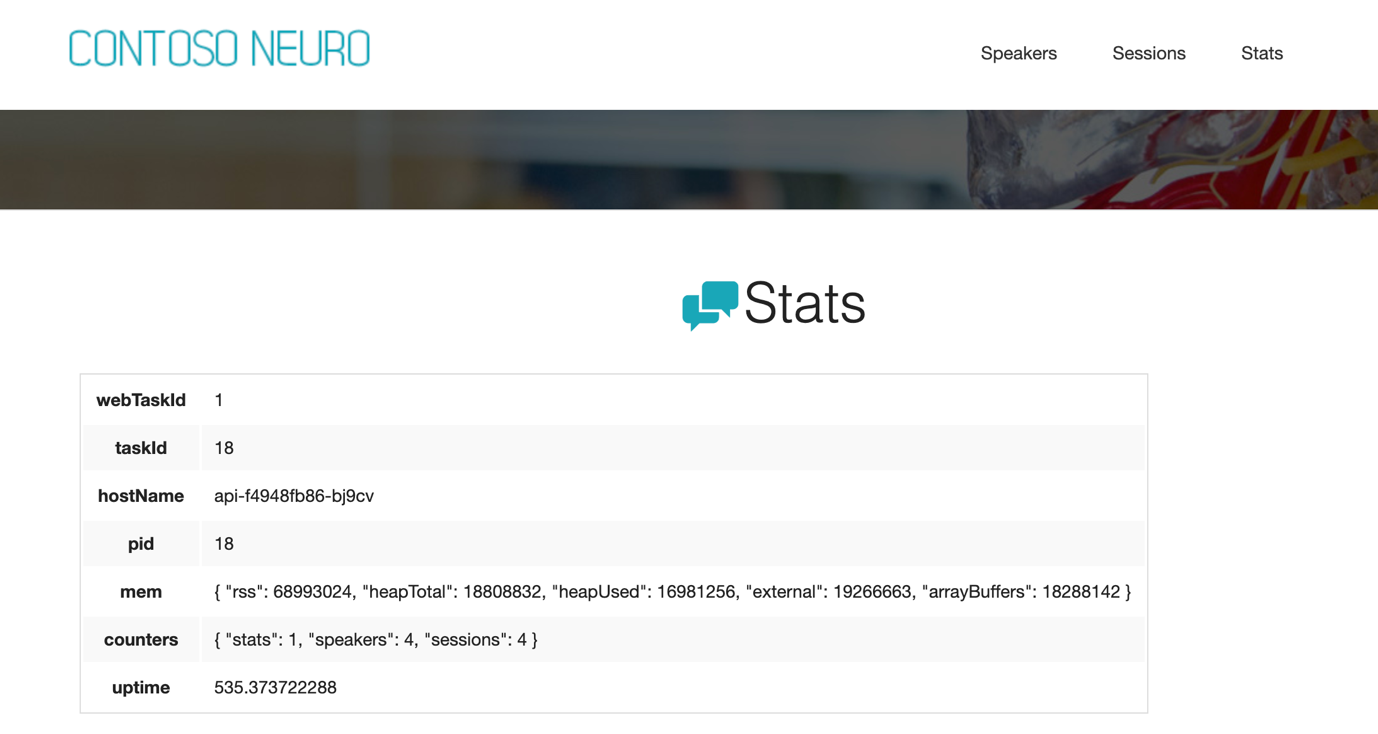
1. Выберите «Обзор + сохранение», при появлении запроса подтвердите изменения и нажмите «Сохранить».
2. [](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-16-56-28.png)
3. Вернитесь к главному представлению рабочих нагрузок на портале AKS Azure, и теперь вы увидите, что развертывание исправно с двумя работающими модулями.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/healthy-deployment.png)

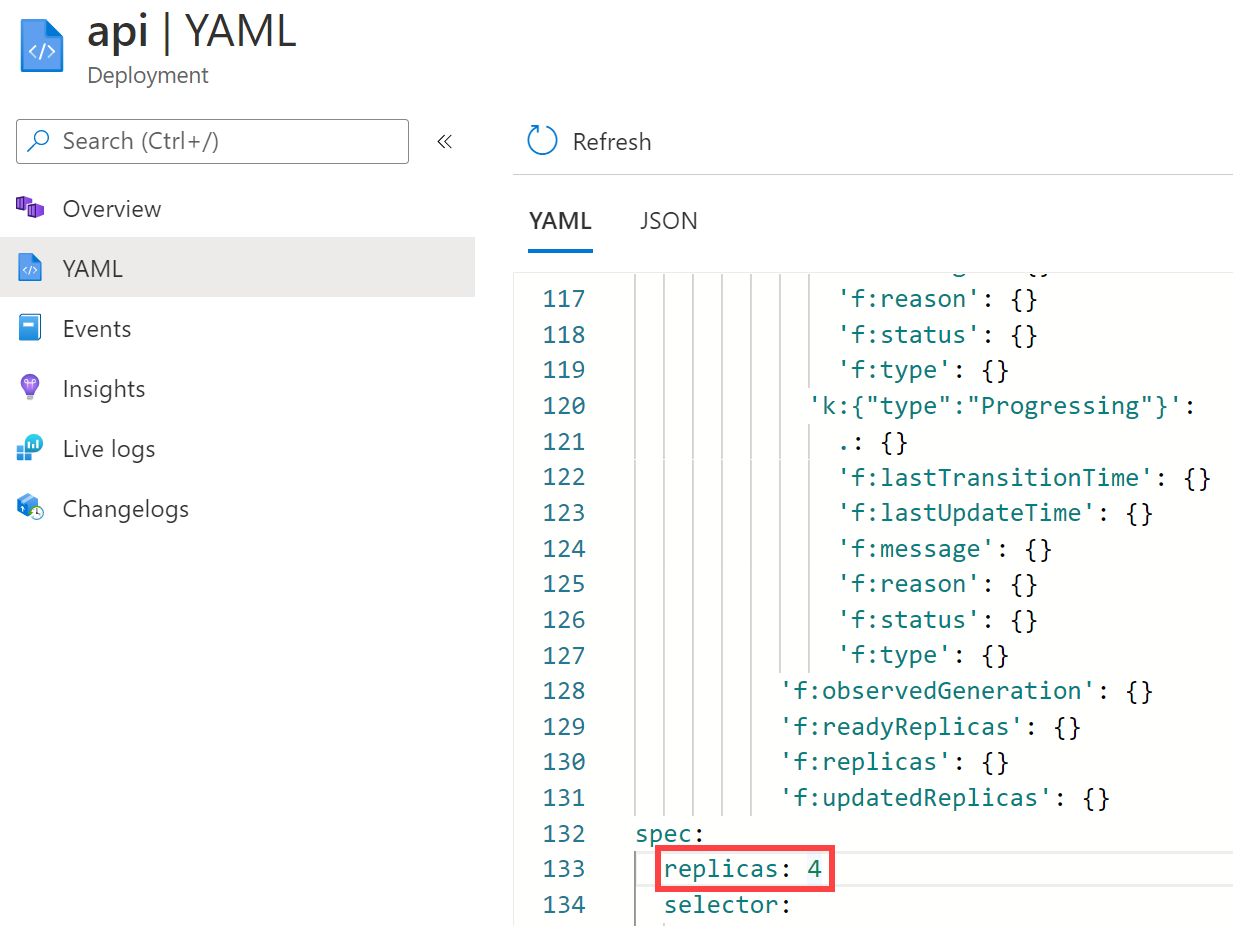
**Задача 3: перезапустить контейнеры и протестировать HA**

В этой задаче вы перезапустите контейнеры и убедитесь, что перезапуск не влияет на работающую службу.

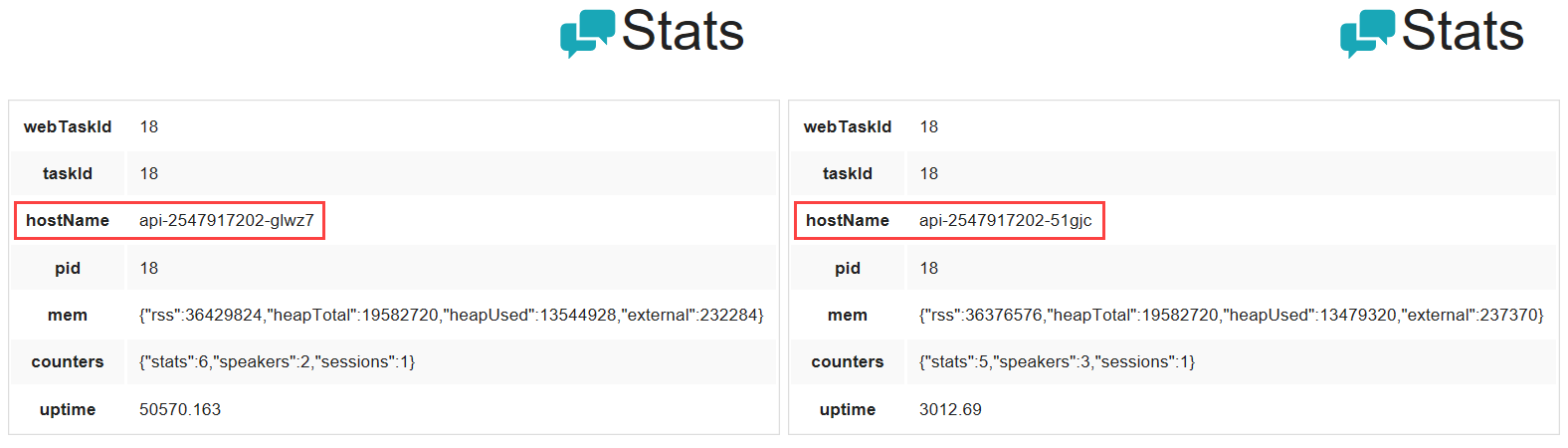
1. Откройте образец веб-приложения и перейдите на страницу «Статистика», как показано.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/image123.png)

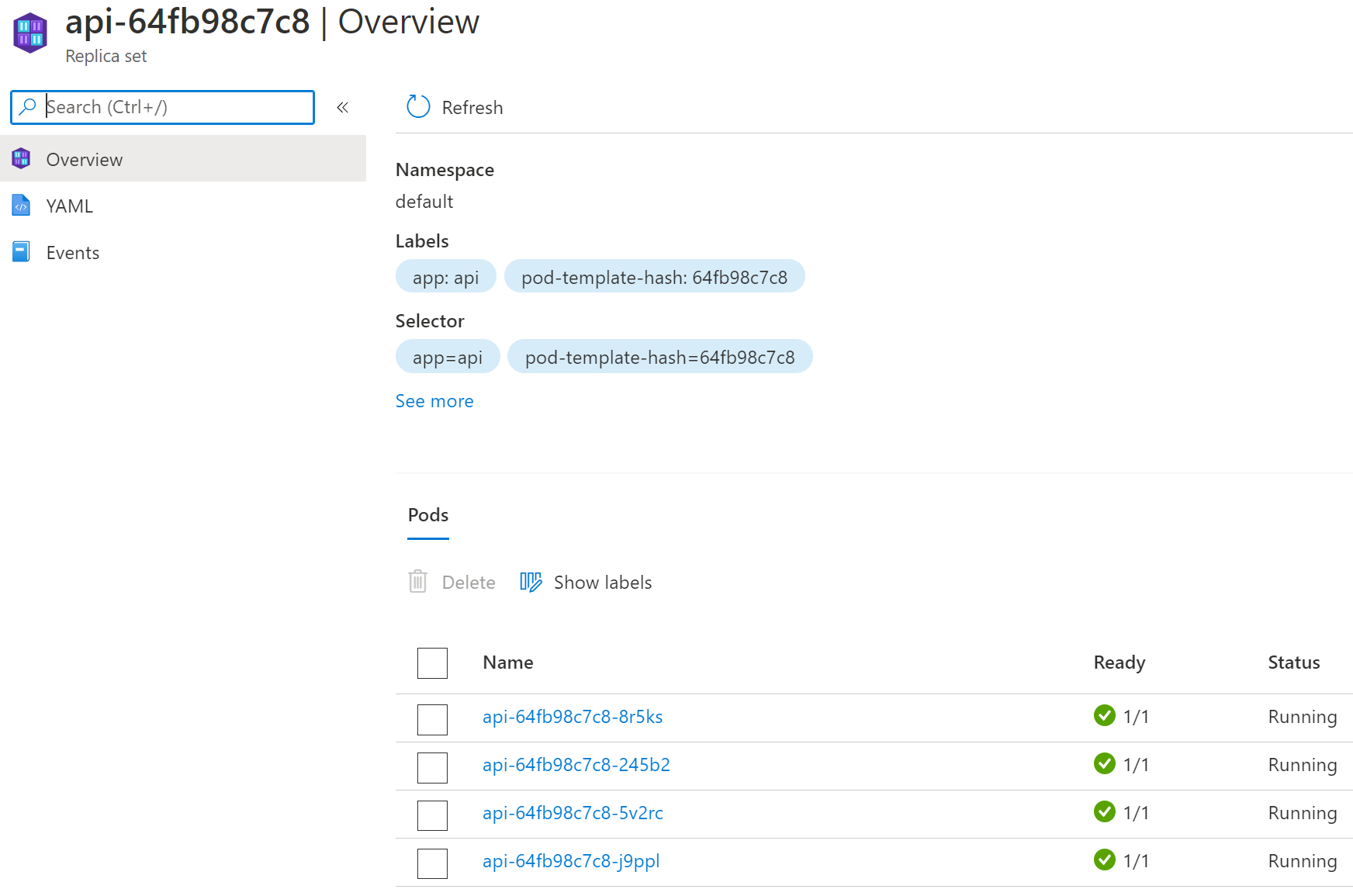
1. В колонке AKS на портале Azure откройте развертывание api и увеличьте необходимое количество реплик до 4.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-17-30-28.png)

1. Через несколько секунд вы обнаружите, что развертывание API теперь успешно выполняет 4 реплики.
2. Вернитесь на вкладку браузера с загруженной страницей статистики веб-приложения. Обновляйте страницу снова и снова. Вы не увидите никаких ошибок, но вы будете видеть, что имя хоста api периодически меняется между четырьмя экземплярами api pod. Идентификатор задачи и pid также могут меняться между четырьмя экземплярами api pod.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/image126.png)

1. После обновления достаточного количества раз, чтобы увидеть, что значение hostName изменяется, а служба остается работоспособной, вы можете открыть представление Replica Sets для API на портале Azure.
2. В этом представлении вы можете увидеть, что значение hostName, показанное на странице статистики веб-приложения, соответствует именам модулей для запущенных модулей.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-17-31-02.png)

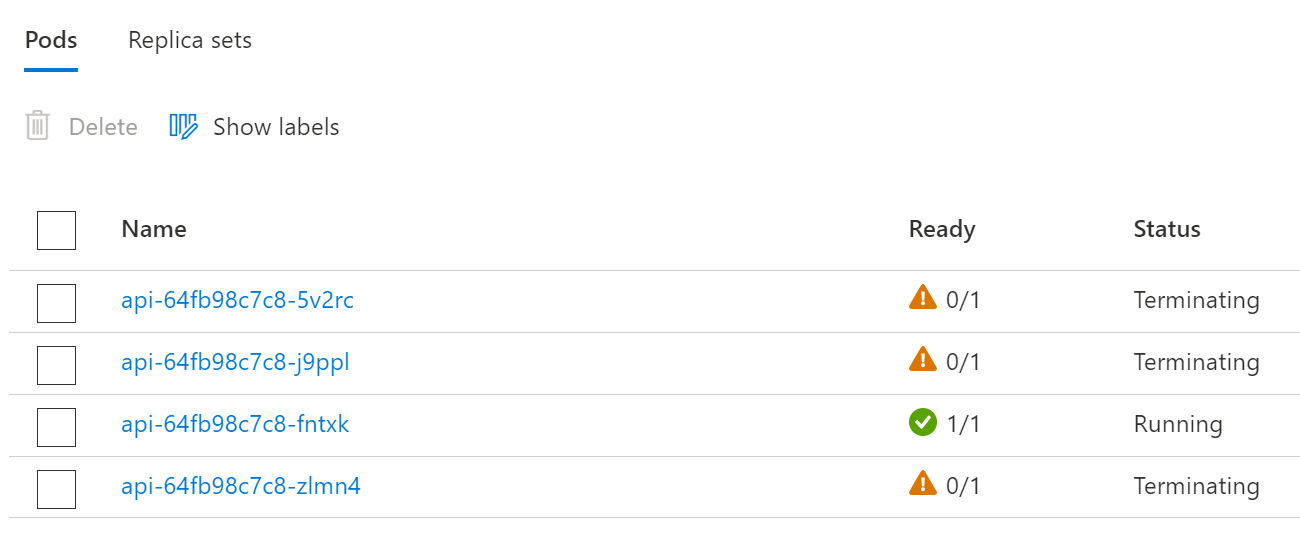
1. Выберите случайным образом два модуля и нажмите «Удалить». Выберите Подтвердить удаление и снова нажмите Удалить.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-17-31-31.png)

1. Kubernetes запустит новые поды, чтобы обеспечить необходимое количество реплик. В зависимости от вашего представления вы можете увидеть, что старые экземпляры завершаются, а новые экземпляры создаются.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-17-31-54.png)

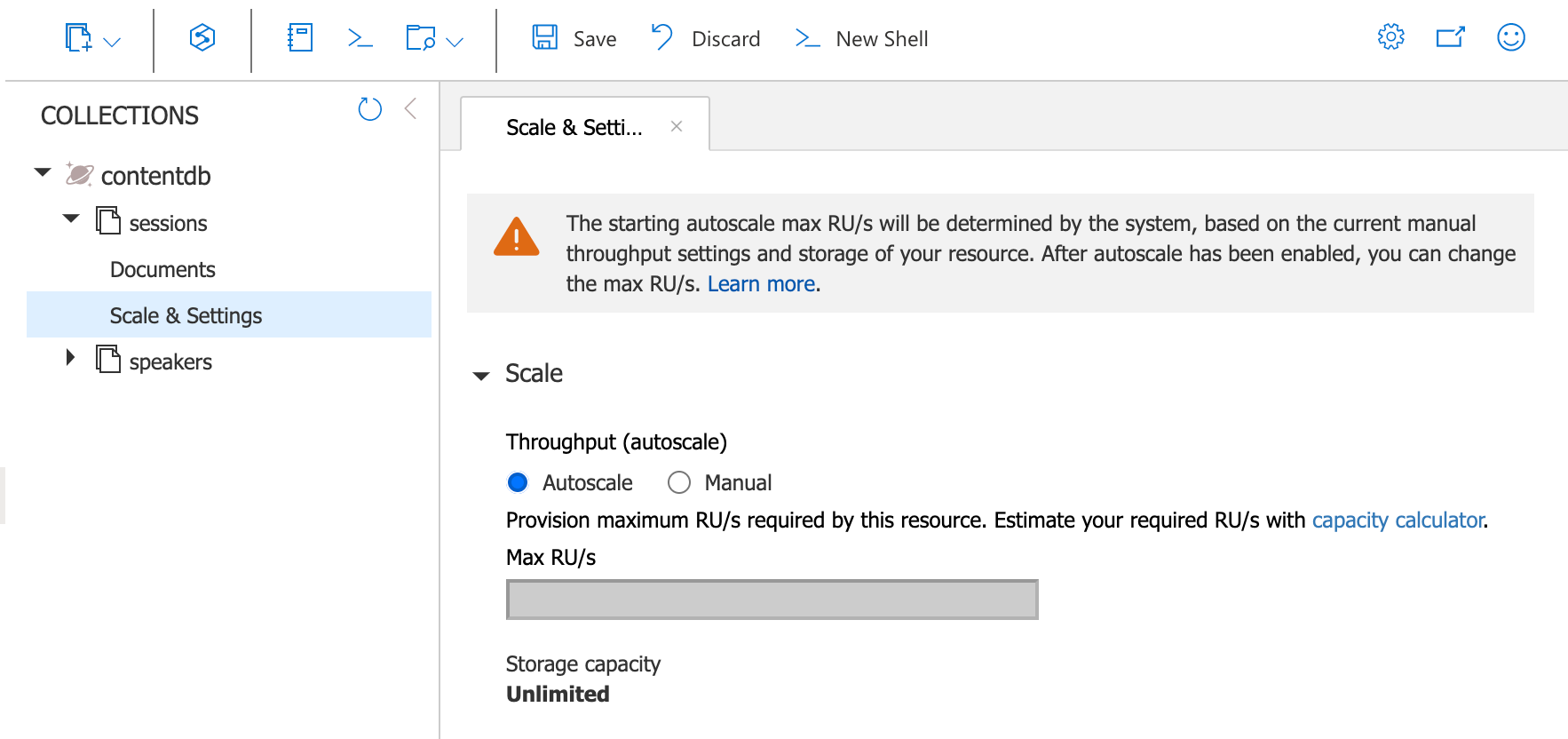
1. Вернитесь к развертыванию API и уменьшите масштаб до 1 реплики. См. Шаг 2 выше, чтобы узнать, как это сделать, если вы не уверены.
2. Вернитесь на страницу статистики примера веб-сайта в браузере и обновите, пока Kubernetes сокращает количество подов. Вы заметите, что отображается только одно имя хоста API, хотя вы все еще можете видеть несколько запущенных модулей в представлении набора реплик API. Несмотря на то, что запущено несколько модулей, Kubernetes больше не будет отправлять трафик в модули, которые он выбрал для завершения. Через несколько секунд в представлении набора реплик API отобразится только один модуль.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-17-32-24.png)

**Задача 4. Настройка автомасштабирования Cosmos DB.**

В этой задаче вы настроите автомасштабирование в Azure Cosmos DB.

1. 1. На портале Azure перейдите к учетной записи fabmedical- [SUFFIX] Azure Cosmos DB.
2. Выберите **Data Explorer**.
3. В проводнике данных разверните базу данных contentdb, затем разверните коллекцию сеансов.
4. В коллекции сеансов выберите Scale & Settings.
5. В разделе Scale & Settings выберите Autoscale для параметра в разделе.

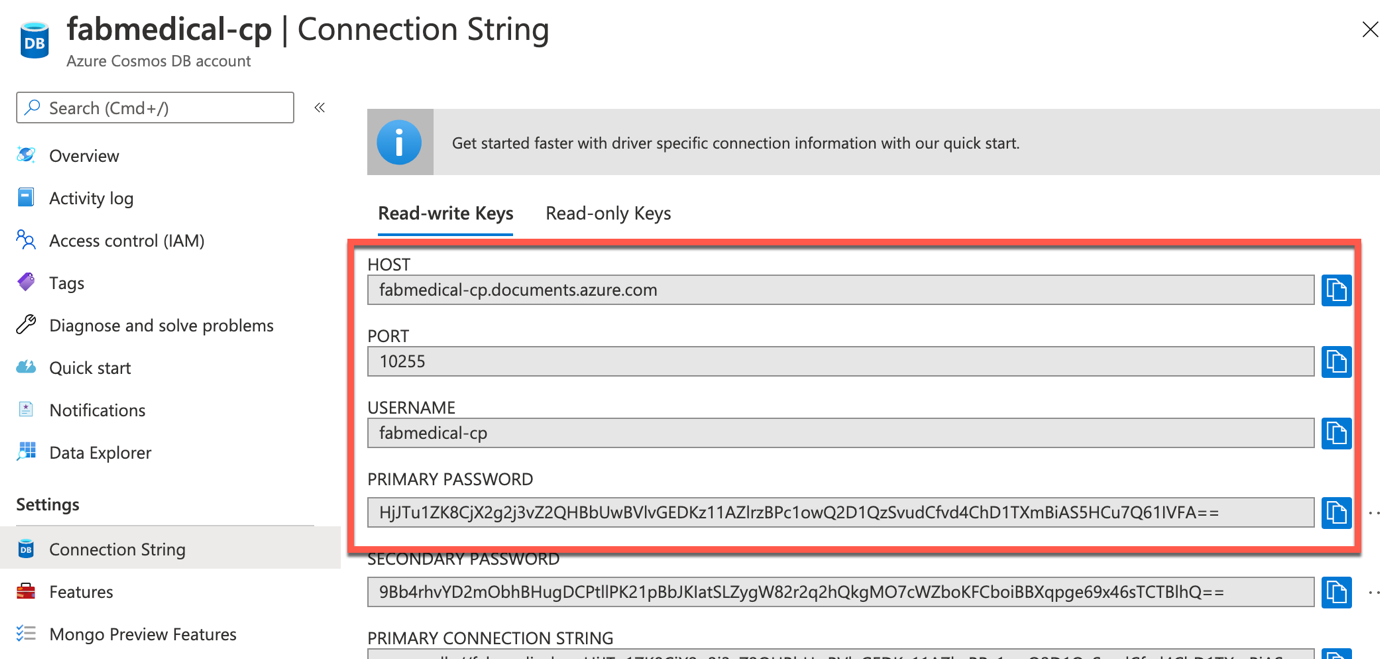
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/cosmosdb-autoscale.png)

1. Выполните ту же задачу, чтобы включить автоматическое масштабирование пропускной способности для коллекции speakers.

**Задача 5. Тестирование автомасштабирования Cosmos DB**

В этой задаче вы запустите сценарий тестирования производительности, который проверит функцию автомасштабирования в Azure Cosmos DB, чтобы вы могли увидеть, что теперь масштабирование будет превышать 400 RU/s.

1. На портале Azure перейдите к учетной записи fabmedical- [SUFFIX] Cosmos DB.
2. В разделе «Настройки» выберите «Строка подключения».
3. На панели строки подключения скопируйте значения HOST, USERNAME и PRIMARY PASSWORD. Сохраните их для использования позже.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/cosmos-connection-string-pane.png)

Примечание. В своей учетной записи Cosmos DB вы можете увидеть, что конечная точка хоста использует .mongo.cosmos.azure.com, который предназначен для Mongo DB версии 3.6. Показанная здесь конечная точка - .documents.azure.com, которая предназначена для Mongo DB версии 3.2. Вы можете использовать любую конечную точку для целей этой Задачи.

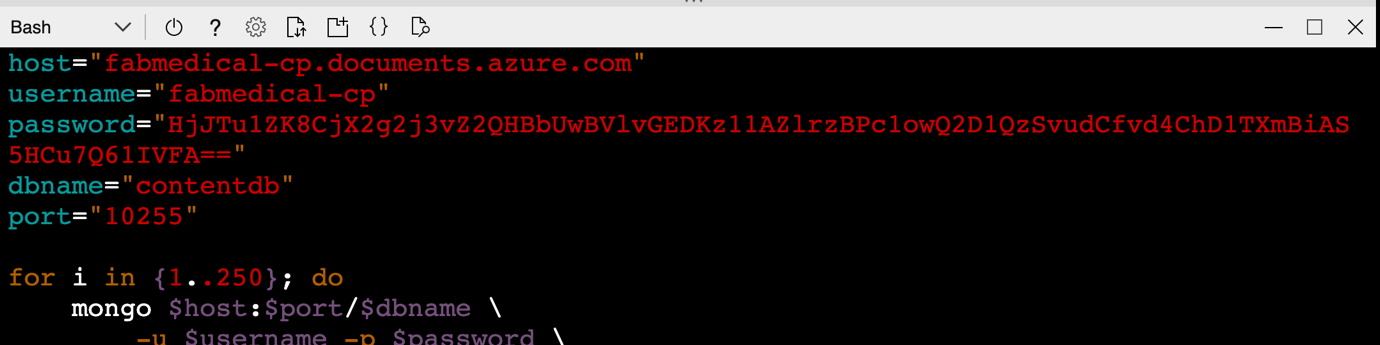
1. Откройте Azure Cloud Shell и подключитесь по SSH к виртуальной машине агента сборки.
2. На виртуальной машине агента сборки перейдите в каталог ~/Fabmedical.

cd ~/Fabmedical

1. Выполните следующую команду, чтобы открыть сценарий perftest.sh для редактирования в Vim.

vi perftest.sh

1. В верхней части скрипта perftest.sh объявлено несколько переменных. Измените переменные хоста, имени пользователя и пароля, установив их значения в соответствующие значения строки подключения Cosmos DB, которые были скопированы ранее.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/cosmos-perf-test-variables.png)

1. Сохраните файл и выйдите из Vim.
2. Выполните следующую команду, чтобы выполнить сценарий perftest.sh для запуска небольшого нагрузочного теста для Cosmos DB. Этот скрипт будет использовать RU в Cosmos DB, вставляя множество документов в контейнер Sessions.

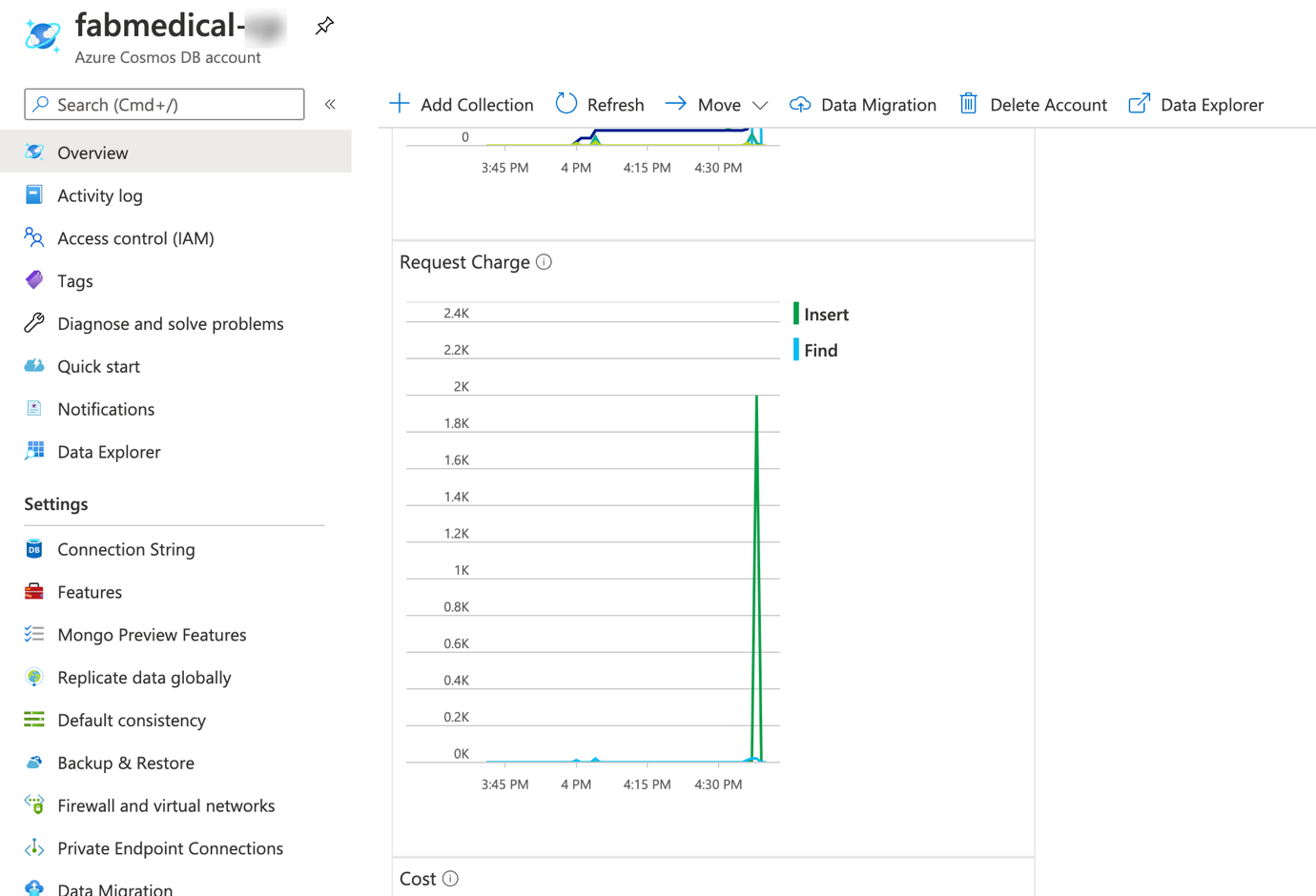
bash ./perftest.sh

Примечание. Скрипт завершится через минуту.

1. После завершения сценария вернитесь к учетной записи Cosmos DB на портале Azure.
2. После завершения сценария вернитесь к учетной записи Cosmos DB на портале Azure.

Примечание. Для отображения активности в коллекции Cosmos DB в журнале активности может потребоваться 2–5 минут. Подождите пару минут, а затем обновите панель, если недавняя плата за запрос не отображается прямо сейчас.

1. Обратите внимание, что плата за запрос теперь показывает, что в учетной записи Cosmos DB была активность, которая превысила предел в 400 единиц в секунду, который был ранее установлен до включения автомасштабирования.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/cosmos-request-charge.png)

**Упражнение 4: Работа со службами и маршрутизация трафика приложений**

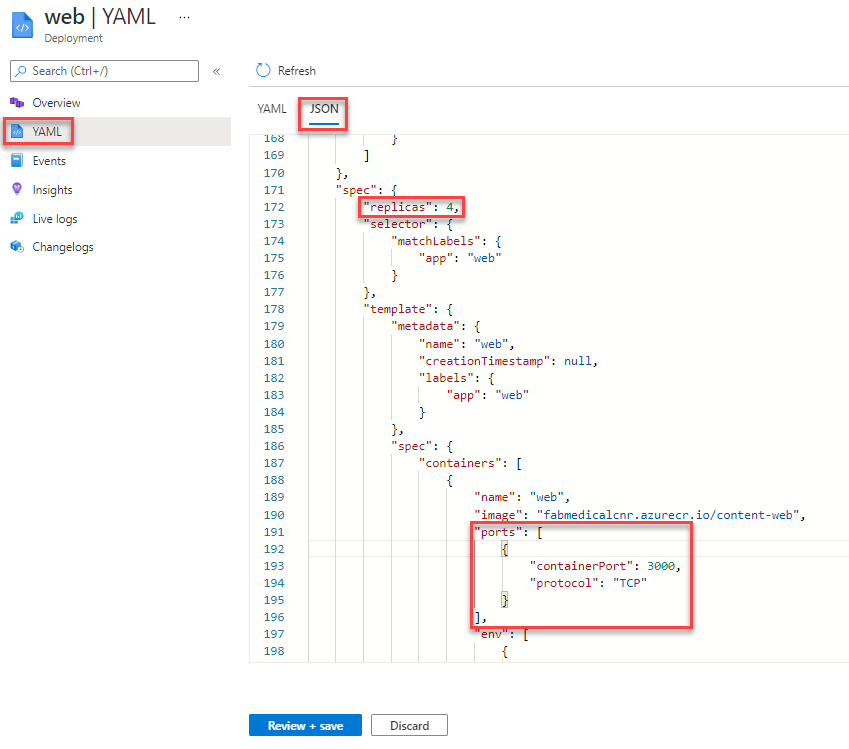
В предыдущем задании мы ввели ограничение на свойства масштабирования службы. В этом упражнении вы настроите развертывания api для создания модулей, которые используют динамическое сопоставление портов, чтобы устранить ограничение ресурсов порта во время масштабных операций.

Службы Kubernetes могут обнаруживать порты, назначенные каждому модулю, что позволяет запускать несколько экземпляров модуля на одном узле агента - что невозможно при настройке определенного статического порта (например, 3001 для службы API).

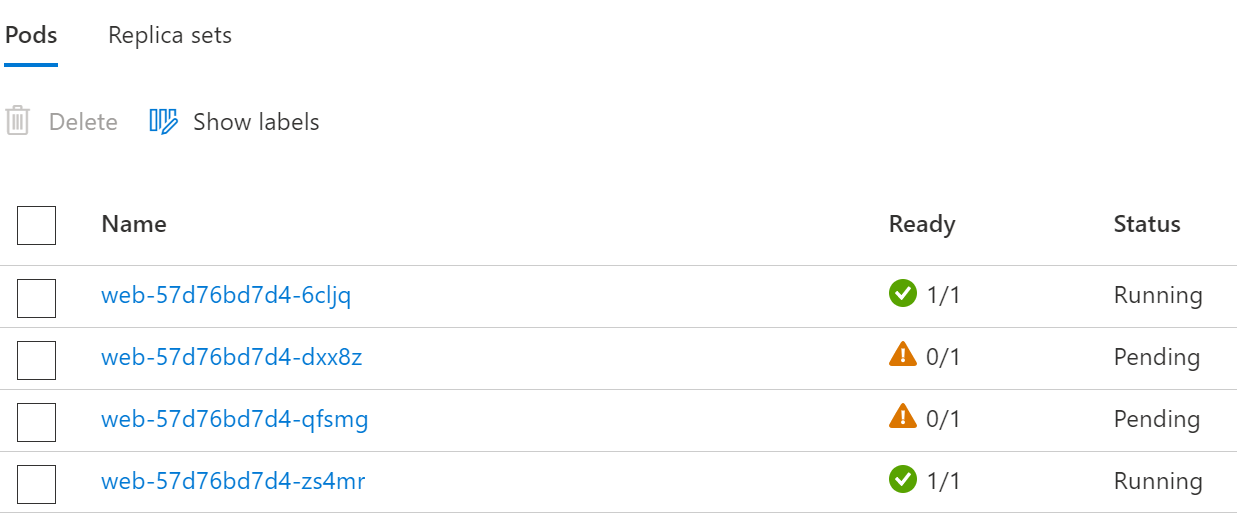
**Задача 1. Обновите внешнюю службу для поддержки динамического обнаружения с помощью балансировщика нагрузки.**

В этой задаче вы обновите веб-службу, чтобы она поддерживала динамическое обнаружение через балансировщик нагрузки Azure.

1. В меню ресурсов AKS Kubernetes выберите «Развертывания» в разделе «Рабочие нагрузки». Из списка выберите веб-развертывание.
2. Выберите YAML, затем перейдите на вкладку JSON.
3. Сначала найдите узел реплик и обновите необходимое количество до 4.
4. Затем перейдите к спецификации веб-контейнеров, как показано на снимке экрана. Удалите запись hostPort для сопоставления портов веб-контейнера.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/update-web-deployment.png)

1. Выберите «Обзор + сохранение», затем подтвердите изменение и нажмите «Сохранить».
2. Проверьте состояние горизонтального масштабирования, обновив представление веб-развертывания. В меню навигации выберите «Модули» в разделе «Рабочие нагрузки». Выберите веб-модули. В этом представлении вы должны увидеть ошибку, подобную показанной на следующем снимке экрана.

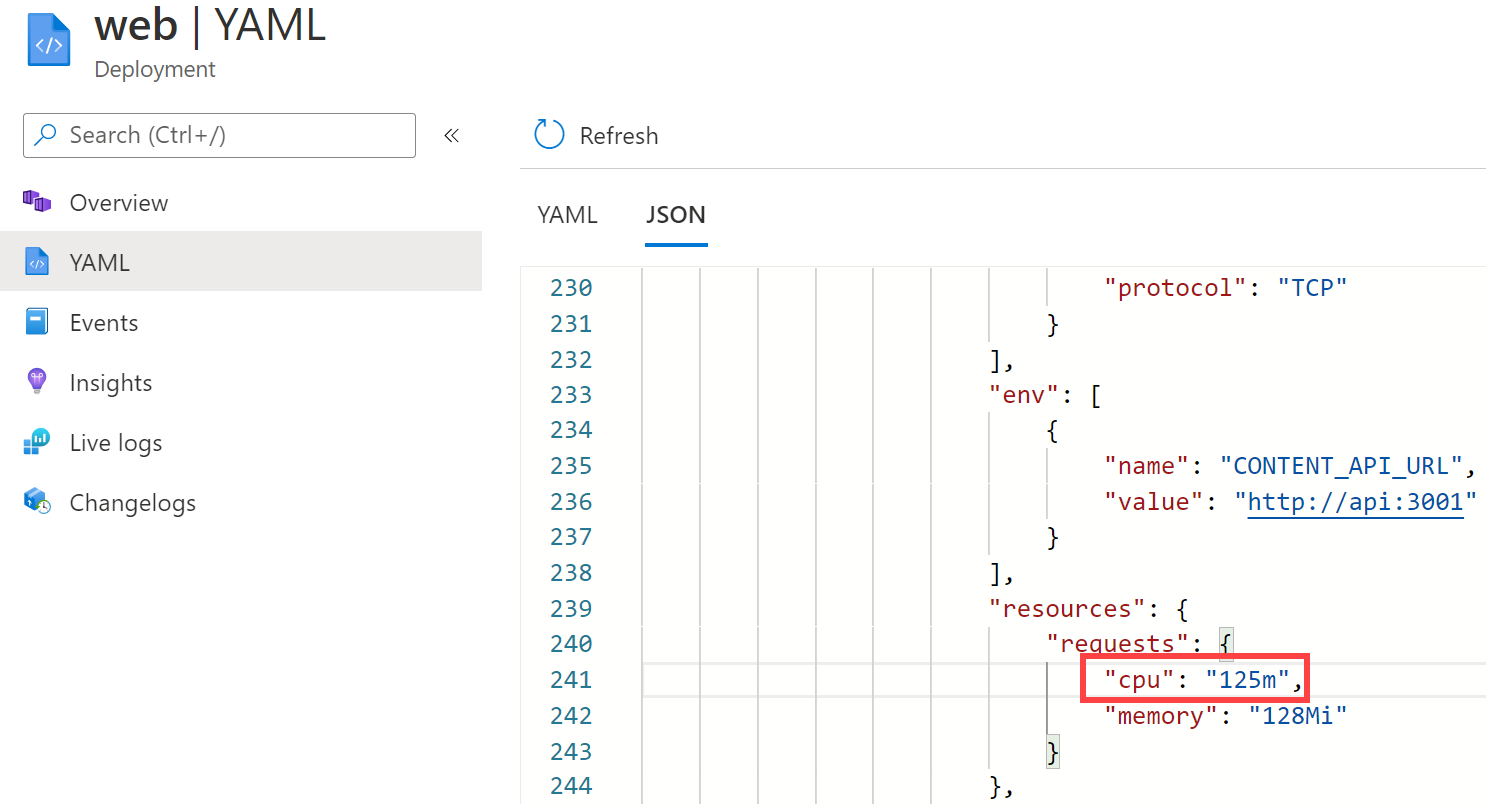
[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-18-23-38.png)

Подобно развертыванию API, веб-развертывание использовало фиксированный hostPort, и ваши возможности масштабирования были ограничены количеством доступных узлов агента. Однако после решения этой проблемы для веб-службы путем удаления параметра hostPort веб-развертывание по-прежнему не может масштабироваться за пределы двух модулей из-за ограничений ЦП. Развертывание требует больше ЦП, чем требуется веб-приложению, поэтому вы исправите это ограничение в следующей задаче.

**Задача 2. Отрегулируйте ограничения ЦП, чтобы улучшить масштабирование**

В этой задаче вы измените требования к ЦП для веб-службы, чтобы ее можно было масштабировать до большего количества экземпляров.

1. Снова откройте представление JSON для веб-развертывания, а затем найдите требования к ресурсам ЦП для веб-контейнера. Измените это значение на 125 м.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-18-24-06.png)

1. Выберите «Обзор + сохранение», подтвердите изменение, а затем выберите «Сохранить», чтобы обновить развертывание.
2. В меню навигации выберите «Наборы реплик» в разделе «Рабочие нагрузки». В списке Replica Sets представления выберите набор веб-реплик.
3. По завершении обновления развертывания четыре веб-модуля должны отображаться в рабочем состоянии.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-18-24-35.png)

**Задача 3. Добавьте поддержку Application Insights**

В этой задаче вы отредактируете исходный код веб-приложения, чтобы добавить Application Insights и обновить образ Docker, используемый при развертывании. Затем вы выполните скользящее обновление, чтобы продемонстрировать, как развернуть изменение кода.

1. 1. Выполните эту команду в Azure Cloud Shell, чтобы получить ключ инструментария для ресурса Content-Web Application Insights:

az resource show -g fabmedical-[SUFFIX] -n content-web --resource-type "Microsoft.Insights/components" --query properties.InstrumentationKey -o tsv

Скопируйте это значение. Вы будете использовать это позже.

Примечание: если у вас пустой результат, убедитесь, что введенная вами команда относится к нужному ресурсу.

1. На виртуальной машине обновите файлы репозитория fabmedical, извлекая последние изменения из репозитория git:
2. cd ~/fabmedical/content-web

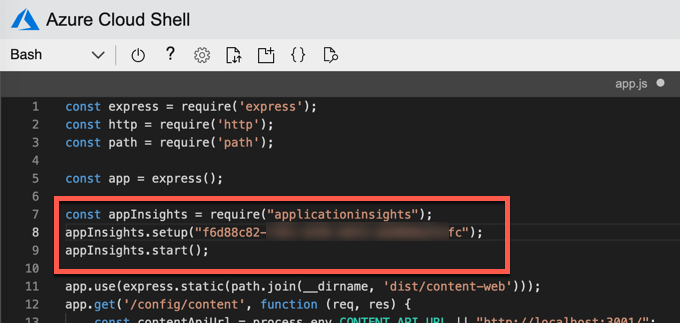
git pull

1. Установите поддержку Application Insights.

npm install applicationinsights --save

1. Отредактируйте файл app.js с помощью Vim или Visual Studio Code и добавьте следующие строки сразу после создания экземпляра express в строке 6:
2. const appInsights = require("applicationinsights");
3. appInsights.setup("[YOUR APPINSIGHTS KEY]");

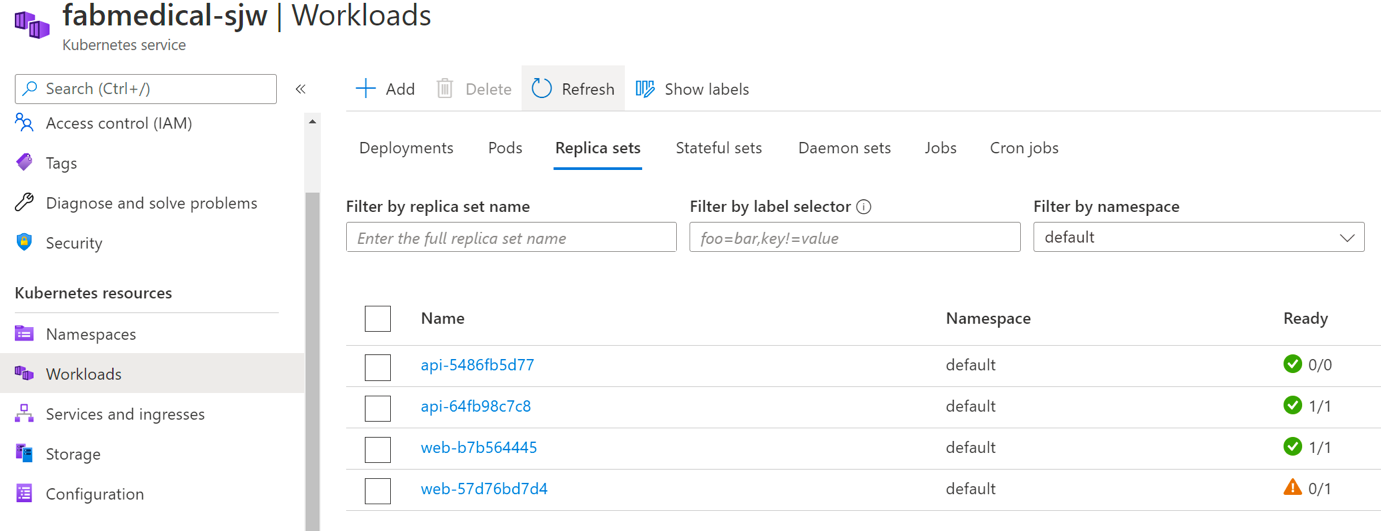
appInsights.start();

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/hol-2019-10-02_12-33-29.png)

1. Сохраните изменения и закройте редактор.
2. Отправьте эти изменения в свой репозиторий, чтобы GitHub Actions CI построил и развернул новый образ контейнера.
3. git add .
4. git commit -m "Added Application Insights"

git push

1. Посетите действие content-web для вашего репозитория GitHub Fabmedical и посмотрите, как новый образ развертывается в вашем кластере Kubernetes.
2. Пока выполняется это обновление, верните портал Azure в браузере.
3. В меню навигации выберите «Наборы реплик» в разделе «Рабочие нагрузки». В этом представлении вы увидите новый набор реплик для Интернета, который может все еще находиться в процессе развертывания (как показано ниже) или уже полностью развернут.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-18-25-30.png)

1. Во время развертывания вы можете перейти к веб-приложению и посетить страницу статистики по адресу /stats. Обновляйте страницу по мере выполнения непрерывного обновления. Обратите внимание, что служба работает нормально, а задачи продолжают балансировать нагрузку.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/image145.png)

**Задача 4: настроить Kubernetes Ingress**

В этой задаче вы настроите Kubernetes Ingress с использованием прокси-сервера nginx, чтобы воспользоваться преимуществами маршрутизации на основе адресов и работой с TLS.

1. В Azure Cloud Shell выполните следующую команду, чтобы добавить стабильный репозиторий Helm nginx:

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx

1. Обновите список пакетов Helm.

helm repo update

Примечание. Если вы получили сообщение «Репозитории не найдены». ошибка, затем выполните следующую команду. Это вернет официальный "стабильный" репозиторий Helm.

helm repo add stable https://charts.helm.sh/stable

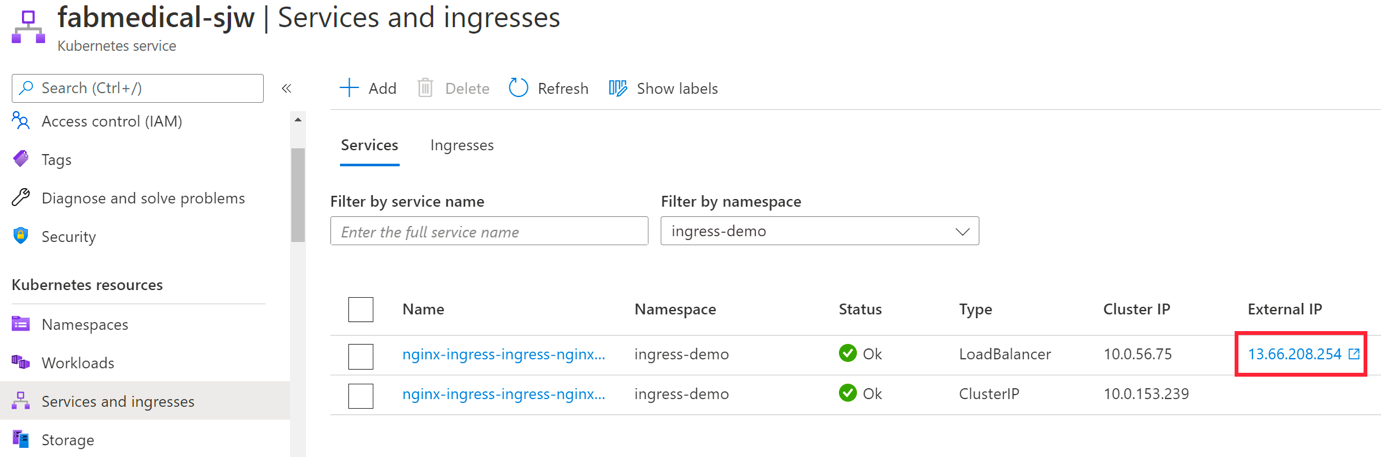
1. Создайте пространство имен в Kubernetes для установки ресурсов Ingress.

kubectl create namespace ingress-demo

1. Установите ресурс контроллера Ingress для обработки входящих запросов по мере их поступления. Контроллер Ingress получит собственный общедоступный IP-адрес на балансировщике нагрузки Azure и сможет обрабатывать запросы для нескольких служб через порты 80 и 443.
2. helm install nginx-ingress ingress-nginx/ingress-nginx \
3. --namespace ingress-demo \
4. --set controller.replicaCount=2 \
5. --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \
6. --set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux \

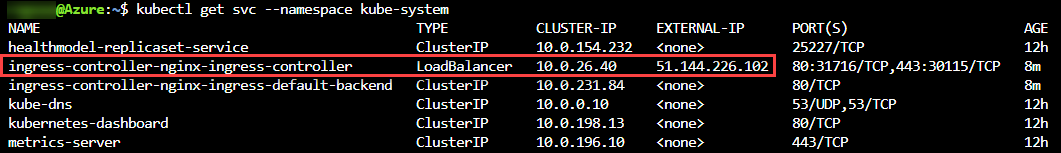
--set controller.admissionWebhooks.patch.nodeSelector."beta\.kubernetes\.io/os"=linux

1. На портале Azure в разделе «Службы и входящие данные» скопируйте IP-адрес для внешнего IP-адреса службы nginx-ingress-RANDOM-nginx-ingress.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/2021-03-26-18-26-13.png)

Примечание. Обновление может занять несколько минут. Кроме того, вы можете найти IP-адрес с помощью следующей команды в Azure Cloud Shell.

kubectl get svc --namespace ingress-demo

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex4-Task5.5a.png)

1. Откройте колонку «Группы ресурсов портала Azure» и найдите группу ресурсов, которая была автоматически создана для размещения пулов узлов для AKS. Он будет иметь формат именования MC\_fabmedical-[SUFFIX]\_fabmedical-[SUFFIX]\_[REGION].
2. В Azure Cloud Shell создайте сценарий для обновления общедоступного DNS-имени для входящего внешнего IP-адреса.

code update-ip.sh

Вставьте следующее как содержимое. Обязательно замените в скрипте следующие заполнители:

* + [INGRESS PUBLIC IP]: Замените это IP-адресом, скопированным с шага 5.
  + [AKS NODEPOOL RESOURCE GROUP]: Замените именем группы ресурсов, скопированным с шага 6.
  + [SUFFIX]: Замените его тем же значением SUFFIX, которое использовалось ранее в этой лабораторной работе.

#!/bin/bash

# Public IP address

IP="[INGRESS PUBLIC IP]"

# Resource Group that contains AKS Node Pool

KUBERNETES\_NODE\_RG="[AKS NODEPOOL RESOURCE GROUP]"

# Name to associate with public IP address

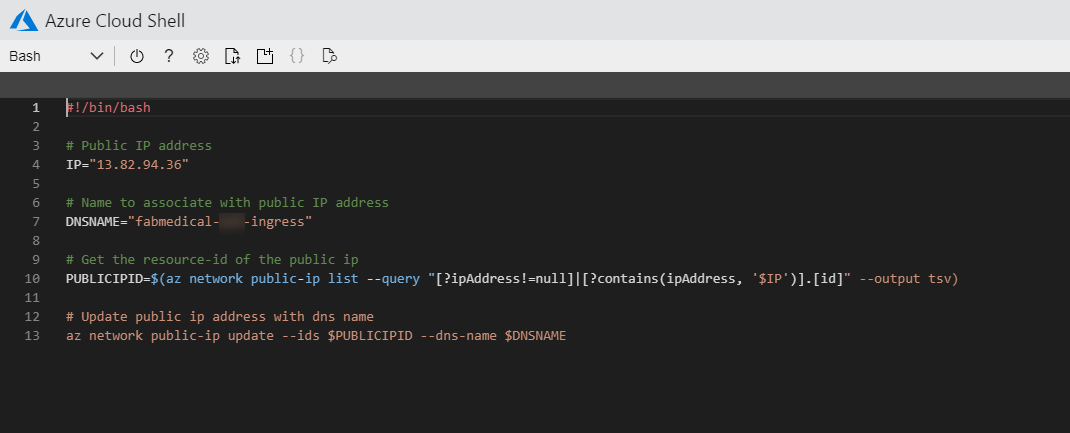
DNSNAME="fabmedical-[SUFFIX]-ingress"

# Get the resource-id of the public ip

PUBLICIPID=$(az network public-ip list --resource-group $KUBERNETES\_NODE\_RG --query "[?ipAddress!=null]|[?contains(ipAddress, '$IP')].[id]" --output tsv)

# Update public ip address with dns name

az network public-ip update --ids $PUBLICIPID --dns-name $DNSNAME

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex4-Task5.6.png)

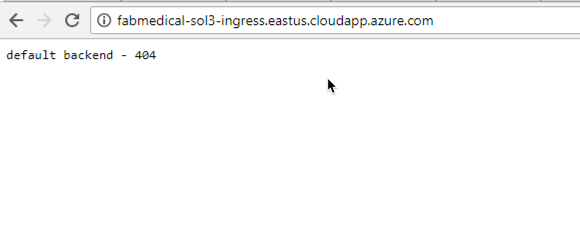
1. Сохраните изменения и закройте редактор.
2. Запускаем скрипт обновления.

bash ./update-ip.sh

1. Проверьте обновление IP, посетив URL-адрес в своем браузере.

Примечание. В настоящее время получение сообщения 404 является нормальным.

http://fabmedical-[SUFFIX]-ingress.[AZURE-REGION].cloudapp.azure.com/

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex4-Task5.9.png)

1. Используйте helm для установки cert-manager, инструмента, который может автоматически предоставлять сертификаты SSL с letsencrypt.org.
2. kubectl create namespace cert-manager
3. kubectl label namespace cert-manager cert-manager.io/disable-validation=true

kubectl apply --validate=false -f https://github.com/jetstack/cert-manager/releases/download/v1.0.1/cert-manager.yaml

1. Диспетчеру сертификатов потребуется специальный ресурс ClusterIssuer для обработки запросов SSL-сертификатов.

code clusterissuer.yml

Следующая конфигурация ресурсов должна работать как есть:

apiVersion: cert-manager.io/v1

kind: ClusterIssuer

metadata:

name: letsencrypt-prod

spec:

acme:

# The ACME server URL

server: https://acme-v02.api.letsencrypt.org/directory

# Email address used for ACME registration

email: user@fabmedical.com

# Name of a secret used to store the ACME account private key

privateKeySecretRef:

name: letsencrypt-prod

# Enable HTTP01 validations

solvers:

- http01:

ingress:

class: nginx

1. Сохраните изменения и закройте редактор.
2. Создайте эмитент с помощью kubectl.

kubectl create --save-config=true -f clusterissuer.yml

1. Теперь вы можете создать объект сертификата.

Примечание:

Cert-manager, возможно, уже создал для вас объект сертификата с помощью ingress-shim.

Чтобы убедиться, что сертификат был успешно создан, используйте команду kubectl describe certificate tls-secret.

Если сертификат уже доступен, перейдите к шагу 16.

code certificate.yml

Используйте следующее в качестве содержимого и обновите [SUFFIX] и [AZURE-REGION], чтобы они соответствовали вашему входному DNS-имени.

apiVersion: cert-manager.io/v1

kind: Certificate

metadata:

name: tls-secret

spec:

secretName: tls-secret

dnsNames:

- fabmedical-[SUFFIX]-ingress.[AZURE-REGION].cloudapp.azure.com

issuerRef:

name: letsencrypt-prod

kind: ClusterIssuer

1. Сохраните изменения и закройте редактор.
2. Создайте сертификат с помощью kubectl.

kubectl create --save-config=true -f certificate.yml

Примечание. Чтобы проверить статус выдачи сертификата, используйте команду kubectl describe certificate tls-secret и найдите выходные данные событий, аналогичные приведенным ниже:

Type Reason Age From Message

---- ------ ---- ---- -------

Normal Generated 38s cert-manager Generated new private key

Normal GenerateSelfSigned 38s cert-manager Generated temporary self signed certificate

Normal OrderCreated 38s cert-manager Created Order resource "tls-secret-3254248695"

Normal OrderComplete 12s cert-manager Order "tls-secret-3254248695" completed successfully

Normal CertIssued 12s cert-manager Certificate issued successfully

Прежде чем tls-secret станет доступным, может пройти от 5 до 30 минут. Это связано с задержкой, связанной с предоставлением сертификата TLS от letsencrypt.

1. Теперь вы можете создать входящий ресурс для контент-приложений.

code content.ingress.yml

Используйте следующее в качестве содержимого и обновите [SUFFIX] и [AZURE-REGION], чтобы они соответствовали вашему входному DNS-имени:

apiVersion: networking.k8s.io/v1beta1

kind: Ingress

metadata:

name: content-ingress

annotations:

kubernetes.io/ingress.class: nginx

nginx.ingress.kubernetes.io/rewrite-target: /$1

nginx.ingress.kubernetes.io/use-regex: "true"

nginx.ingress.kubernetes.io/ssl-redirect: "false"

cert-manager.io/cluster-issuer: letsencrypt-prod

spec:

tls:

- hosts:

- fabmedical-[SUFFIX]-ingress.[AZURE-REGION].cloudapp.azure.com

secretName: tls-secret

rules:

- host: fabmedical-[SUFFIX]-ingress.[AZURE-REGION].cloudapp.azure.com

http:

paths:

- path: /(.\*)

backend:

serviceName: web

servicePort: 80

- path: /content-api/(.\*)

backend:

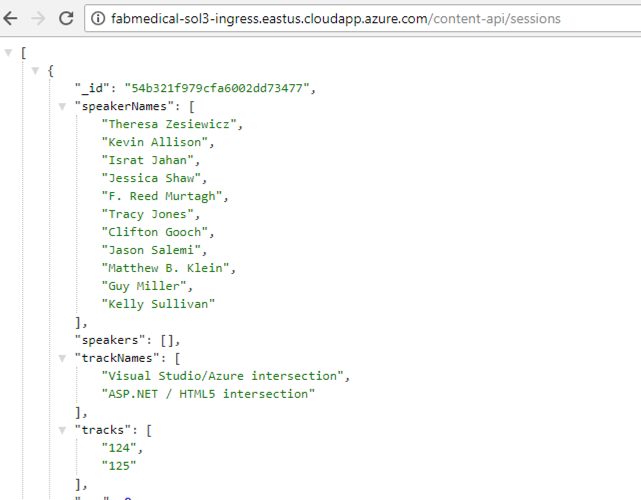
serviceName: api

servicePort: 3001

1. Сохраните изменения и закройте редактор.
2. Создайте вход с помощью kubectl.

kubectl create --save-config=true -f content.ingress.yml

1. Обновите конечную точку входа в браузере. Вы должны иметь возможность посещать страницы докладчиков и сессий и видеть весь контент.
2. Посетите API напрямую, перейдя в /content-api/sessions на входящей конечной точке.

[](https://github.com/microsoft/MCW-Cloud-native-applications/blob/main/Hands-on%20lab/media/Ex4-Task5.19.png)

1. Протестируйте завершение TLS, снова посетив обе службы, используя https.

Примечание. Прежде чем сайт SSL станет доступным, может пройти от 5 до 30 минут. Это связано с задержкой, связанной с предоставлением сертификата TLS от letsencrypt.