

Назначение

DigitalSkills Web Application предоставляет собой веб-сайт для организации веб-конференций. В течение двух конкурсных дней вы преобразуете существующее локальное приложение в приложение на основе контейнера. Этот веб-приложение предоставит решение для размещения мультитенантных веб-приложений с использованием службы Azure Kubernetes (AKS), контейнеров Docker на узлах Linux с миграцией из MongoDB на CosmosDB.

- Службы Azure и сопутствующие продукты
- Служба Azure Kubernetes (AKS)
- Реестр контейнеров Azure
- GitHub/Azure DevOps
- Docker
- Cosmos DB (включая MongoDB API)

Обзор

Перед практической работой вам нужно будет подготовить среду, развернув базу данных и приложение локально на виртуальной машине с помощью Docker и MongoDB. Вам также нужно будет соединить репозиторий GitHub, содержащий лабораторию, с вашей собственной учетной записью GitHub, чтобы иметь возможность настроить конвейер CI/CD.

Требования

1. Подписка Microsoft Azure должна быть оплачиваемой по факту использования или MSDN.
2. Пробные подписки не работают.

Чтобы завершить эту лабораторную настройку, убедитесь, что в вашей учетной записи есть следующее:

- Имеет встроенную роль владельца (Owner) для используемой вами подписки.
- Является ли пользователь-участник в используемом вами клиенте Azure AD. (У гостевых пользователей не будет необходимых разрешений.)
- В вашей подписке должно быть достаточно ядер, чтобы создать агент сборки и кластер службы Azure Kubernetes в Задаче 5: Развертывание шаблона ARM. Вам понадобится восемь ядер, если вы будете следовать точным инструкциям в лабораторной работе, и больше, если вы выберете дополнительных агентов или виртуальные машины большего размера. Выполните шаги, необходимые до начала лабораторной работы, чтобы узнать, нужно ли запрашивать дополнительные ядра в вашей подписке.
- Учетная запись в Microsoft GitHub.
- Локальная или виртуальная машина, настроенная с:
- Браузер, предпочтительно Chrome для согласованности с лабораторными тестами реализации.

- Вам будет предложено установить другие инструменты во время выполнения упражнений.

Before the hands-on lab

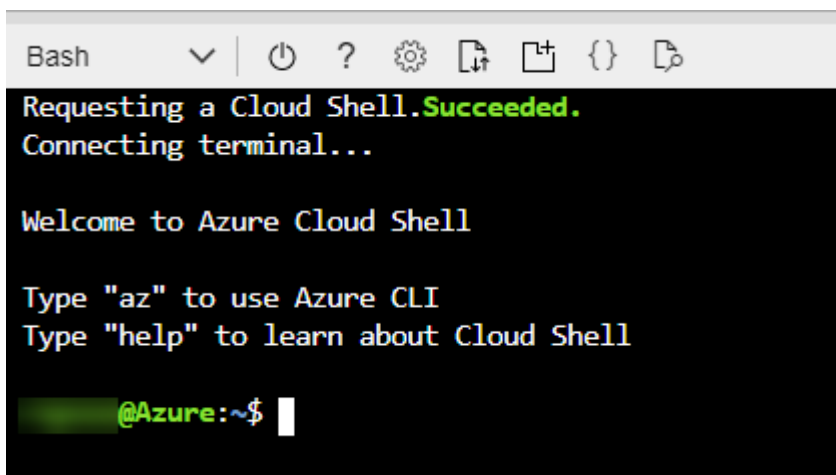
Длительность: 60 минут

Задание 1: Установка Azure Cloud Shell

1. Откройте Azure Cloud Shell, выбрав значок в строке меню..

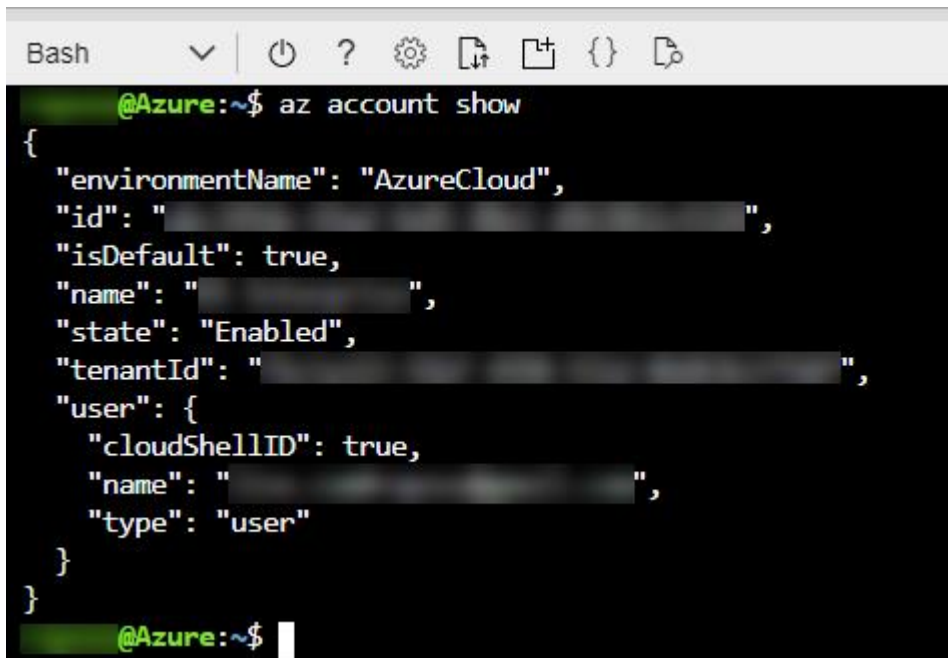


2. Azure Cloud Shell открывается в окне браузера. Выберите Bash, если будет предложено или используйте раскрывающееся меню слева в строке меню оболочки, чтобы выбрать Bash из раскрывающегося списка. При появлении запроса выберите Подтвердить.



3. Убедитесь, что подписка по умолчанию установлена правильно. Чтобы просмотреть текущий тип подписки:

```
az account show
```

A terminal window titled 'Bash' with a standard Linux prompt. The user has entered the command 'az account show'. The output is a JSON object representing the current Azure account configuration. The fields include 'environmentName' (AzureCloud), 'id' (a GUID), 'isDefault' (true), 'name' (a name), 'state' (Enabled), 'tenantId' (a GUID), and 'user' (an object with 'cloudShellID' (true), 'name' (a name), and 'type' (user)).

```
Bash
@Azure:~$ az account show
{
  "environmentName": "AzureCloud",
  "id": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
  "isDefault": true,
  "name": "XXXXXXXX",
  "state": "Enabled",
  "tenantId": "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX",
  "user": {
    "cloudShellID": true,
    "name": "XXXXXXXX",
    "type": "user"
  }
}
```

4. Чтобы установить подписку по умолчанию, отличную от текущего выбора, введите следующее, заменив {id} на желаемое значение идентификатора подписки:

```
az account set --subscription {id}
```

Заметка: чтобы вывести список всех ваших подписок, введите:

```
az account list
```

```
Bash  ▾ | 🔌 ? ⚙️
[~]$ az account list
[
  {
    "cloudName": "AzureCloud",
    "id": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "isDefault": false,
    "name": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "state": "Enabled",
    "tenantId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "user": {
      "name": "XXXXXXXXXXXXXXXXXXXX",
      "type": "user"
    }
  },
  {
    "cloudName": "AzureCloud",
    "id": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "isDefault": true,
    "name": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "state": "Enabled",
    "tenantId": "XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "user": {
      "name": "XXXXXXXXXXXXXXXXXXXX",
      "type": "user"
    }
  }
]
```

Задание 2: Получение необходимых файлов

В этой задаче вы используете git, чтобы скопировать лабораторный контент в cloud shell.

```
git clone https://github.com/masyan/Cloud-applications.git
```

```
Bash  ▾ | 🔌 ? ⚙️ 📄 📁 {} 🗨️
Azure:$ git clone https://github.com/microsoft/MCW-Containers-and-DevOps.git
Cloning into 'MCW-Containers-and-DevOps'...
remote: Enumerating objects: 14, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (12/12), done.
remote: Total 1562 (delta 5), reused 5 (delta 2), pack-reused 1548
Receiving objects: 100% (1562/1562), 46.06 MiB | 42.34 MiB/s, done.
Resolving deltas: 100% (860/860), done.
Checking connectivity... done.
Azure:$
```

Задание 3: Ресурсные группы

Создайте группу ресурсов Azure для хранения ресурсов, которые вы создаете в этой практической лабораторной работе. Такой подход упрощает последующую очистку.

1. В окне cloud shell введите команду, аналогичную следующей, обязательно замените переменные:

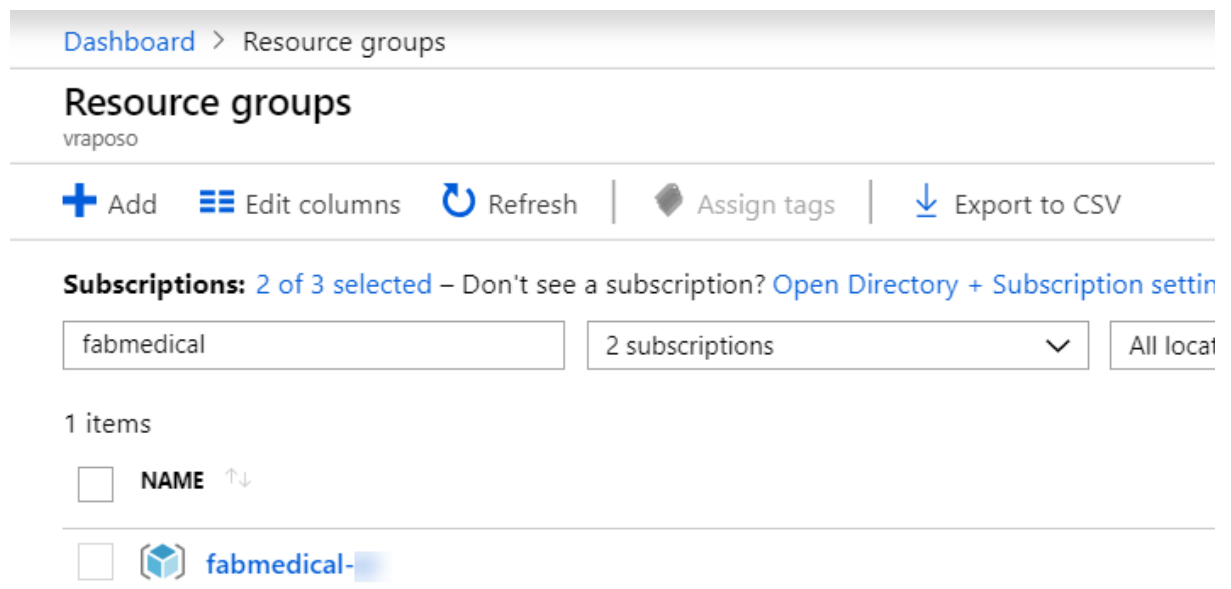
```
az group create -l '[LOCATION]' -n 'fabmedical-[SUFFIX]'
```

- **Suffix:** На протяжении всех заданий следует использовать суффикс, чтобы сделать ресурсы уникальными, например префикс электронной почты или ваше имя и фамилию..
- **Location:** Выберите регион для Azure Container Registry: Canada Central, Canada East, North Central US, Central US, South Central US, East US, East US 2, West US, West US 2, West Central US, France Central, UK South, UK West, North Europe, West Europe, Australia East, Australia Southeast, Brazil South, Central India, South India, Japan East, Japan West, Korea Central, Southeast Asia, East Asia и запомните это для будущих шагов, чтобы все ресурсы, которые вы создаете в Azure, хранились в одном регионе.

Например:

```
az group create -l 'west us' -n 'fabmedical-sol'
```

2. Когда этот шаг будет завершен, то на портале Azure отобразится ваша группа ресурсов.



Задание 4: Создание SSH ключа

Вы создадите виртуальные машины во время следующего задания. В этом разделе вы создадите ключ SSH для безопасного доступа к виртуальным машинам.

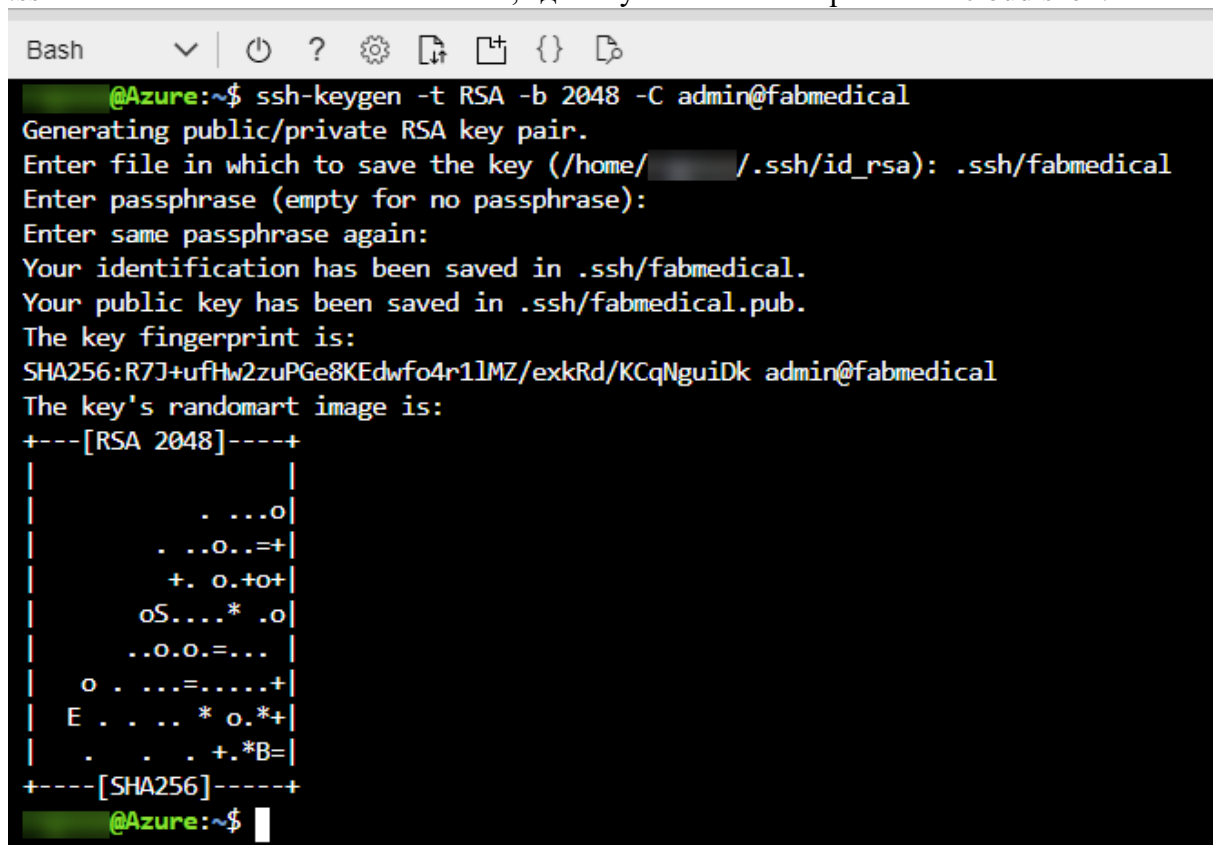
В командной строке cloud shell введите следующую команду, чтобы убедиться, что каталог для ключей SSH существует. Вы можете игнорировать любые ошибки, которые вы видите в выводе.

```
mkdir .ssh
```

1. В командной строке cloud shell введите следующую команду, чтобы сгенерировать пару ключей SSH. Вы можете заменить admin на ваше предпочтительное имя или дескриптор.

```
ssh-keygen -t RSA -b 2048 -C admin@fabmedical
```

2. Когда вас попросят сохранить сгенерированный ключ в файл, введите в качестве имени .ssh/fabmedical
3. Введите парольную фразу и **не забудьте ее!**
4. Поскольку вы ввели .ssh/fabmedical, то ssh-keygen генерирует файл в папке .ssh в вашей пользовательской папке, где по умолчанию открывается cloud shell.



```
Bash
@Azure:~$ ssh-keygen -t RSA -b 2048 -C admin@fabmedical
Generating public/private RSA key pair.
Enter file in which to save the key (/home/ / .ssh/id_rsa): .ssh/fabmedical
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in .ssh/fabmedical.
Your public key has been saved in .ssh/fabmedical.pub.
The key fingerprint is:
SHA256:R7J+ufHw2zuPGe8KEdwfo4r1lMZ/exkRd/KCqNguiDk admin@fabmedical
The key's randomart image is:
+----[RSA 2048]-----+
|
|      . ...o|
|      . ..o..=+|
|      +. o.+o+|
|      oS....* .o|
|      ..o.o.=...|
|      o . ...=.....+|
|      E . . . . * o.*+|
|      . . . . +.*B=|
+----[SHA256]-----+
@Azure:~$
```

5. В командной строке введите следующую команду для вывода содержимого открытого ключа. Скопируйте эту информацию, чтобы использовать позже.

```
cat .ssh/fabmedical.pub
```

6. Не закрывайте cloud shell и оставайтесь в каталоге по умолчанию. Вы будете использовать командную строку в более поздних задачах.

```
Bash
@Azure:~$ cat .ssh/fabmedical.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDUB2N7y68UTg6TbkD9uTd1K3BIQh3Xwa+TYieu3MI2I/1cN0LI8+DzSoUA0
LkoVFNZsbsfYbVw
EVmJB0cHyefB0P9
ZQYETD admin@fabmedical
abmedical
@Azure:~$
```

Задача 5: Развертывание ARM шаблона

В этом разделе вы настраиваете и выполняете шаблон ARM, который создает все ресурсы, необходимые для выполнения задания.

1. Перейдите в каталог с ARM шаблоном:

```
cd MCW-Cloud-native-applications/Hands-on\ lab/arm/
```

2. Откройте файл `azuredeploy.parameters.json` для редактирования.

```
code azuredeploy.parameters.json
```

```
1 {
2   "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentParameters.json#",
3   "contentVersion": "1.0.0.0",
4   "parameters": {
5     "Suffix": {
6       "value": "SUF"
7     },
8     "VirtualMachineAdminUsernameLinux": {
9       "value": "adminfabmedical"
10    },
11    "VirtualMachineAdminPublicKeyLinux": {
12      "value": "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDC57qEjeLUoLT1SEfmPlxir3W0riC//yBsBSJapLYyIDev
13    },
14    "CosmosLocation": {
15      "value": "location"
16    },
17    "CosmosLocationName": {
18      "value": "Location Name"
19    },
20    "CosmosPairedLocation": {
21      "value": "pairedlocation"
22    },
23    "CosmosPairedLocationName": {
24      "value": "Paired Location Name"
25    }
26  }
27 }
```

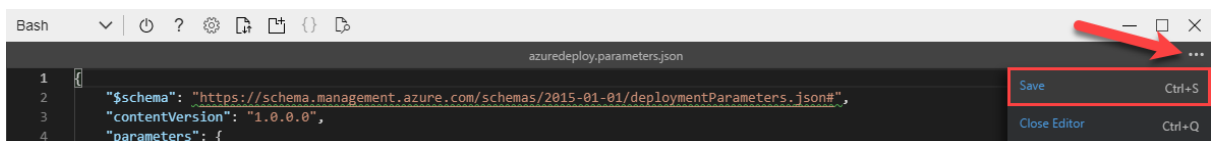
3. Обновите значения для ключей, чтобы они соответствовали вашей среде.:

- **Suffix:** введите сокращенную версию суффикса, содержащую не более 3 символов.
- **VirtualMachineAdminUsernameLinux:** имя пользователя администратора виртуальной машины Linux Build Agent (пример: «adminfabmedical»).

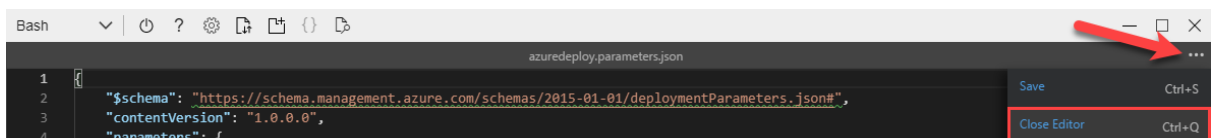
- **VirtualMachineAdminPublicKeyLinux**: открытый ключ ssh администратора виртуальной машины Linux Build Agent. Вы найдете это значение в файле `.ssh / fabmedical.pub`, созданном ранее (пример: «ssh-rsa AAAAB3N (...) vPiybQV admin @ fabmedical»).
- **CosmosLocation**: основное расположение Azure Cosmos DB. Используйте то же расположение, что и ранее созданная группа ресурсов (пример: "eastus").
- **CosmosLocationName**: имя основного расположения Azure Cosmos DB. Используйте имя того же местоположения, что и ранее созданная группа ресурсов (пример: «eastus»).
- **CosmosPairedLocation**: дополнительное расположение Azure Cosmos DB. Приведенную ниже ссылку можно использовать для поиска пары регионов Azure для вашего основного местоположения. (пример: "westus").
- **CosmosPairedLocationName**: имя дополнительного расположения Azure Cosmos DB. Используйте имя местоположения, которое соответствует второстепенному местоположению, определенному в предыдущем ключе (пример: «westus»).

Заметка: Список регионов Azure: <https://docs.microsoft.com/en-us/azure/best-practices-availability-paired-regions#azure-regional-pairs>.

4. Выберите ... и нажмите Сохранить.



5. Закройте редактор.



6. Создайте необходимые ресурсы с помощью ARM шаблона, заменив {resourceGroup} именем ресурсной группы, которую вы создали ранее:

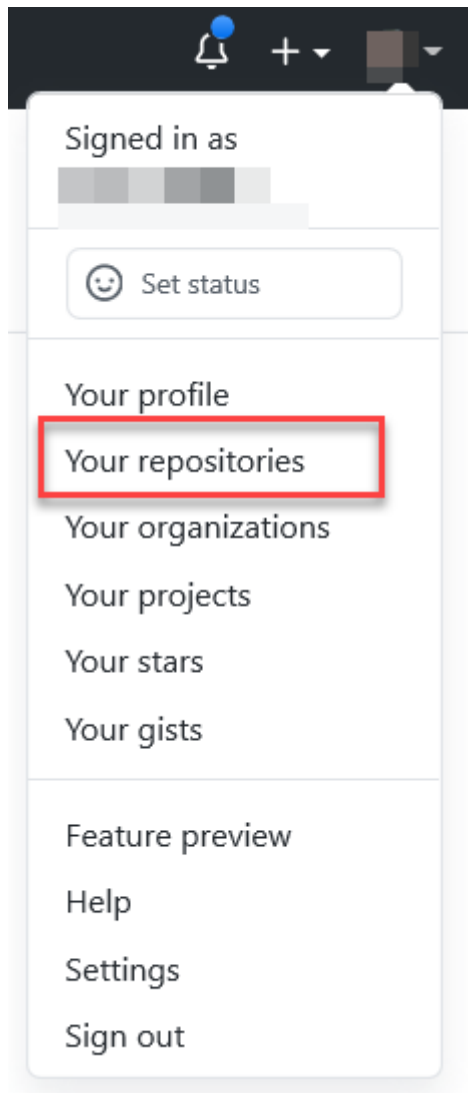
```
az deployment group create --resource-group {resourceGroup} --
template-file azuredeploy.json --parameters
azuredeploy.parameters.json
```

Эта команда занимает от 30 до 60 минут для развертывания всех ресурсов. Вы можете перейти к следующей задаче, чтобы настроить GitHub во время развертывания.

Задание 6: Создание GitHub репозитория

DigitalSkills предоставила вам стартовые файлы. Это копия веб-сайта, который вам предстоит развернуть.

1. Откройте веб-браузер и перейдите на <https://www.github.com>. Войдите в систему, используя данные своей учетной записи GitHub.
2. В правом верхнем углу разверните раскрывающееся меню пользователя и выберите Your repositories.



3. Рядом с поиском найдите и нажмите кнопку «Создать».



4. На экране «Create a new repository» назовите репозиторий ***** и нажмите кнопку «Create repository».

Create a new repository

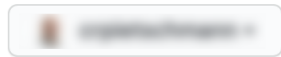
A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Repository template

Start your repository with a template repository's contents.

No template ▾

Owner *



Repository name *

Fabmedical ✓

Great repository names are short and memorable. Need inspiration? How about **solid-fortnight**?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README

This will let you immediately clone the repository to your computer.

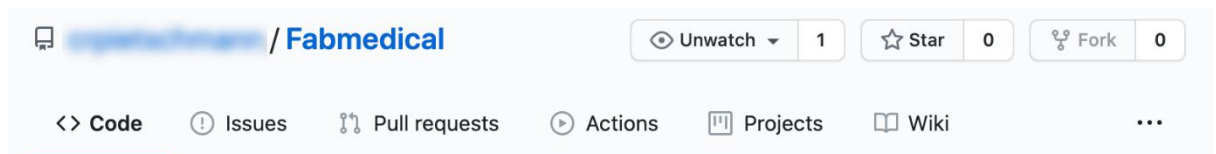
Add .gitignore: None ▾

Add a license: None ▾



Create repository

5. На экране быстрой настройки скопируйте URL-адрес HTTPS GitHub для вашего нового репозитория, вставьте его в блокнот для использования в будущем.



Quick setup — if you've done this kind of thing before



Set up in Desktop

or

HTTPS

SSH

[https://github.com/\[username\]/Fabmedical.git](https://github.com/[username]/Fabmedical.git)



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

6. Откройте новую консоль Azure Cloud Shell. Вы можете сделать это, нажав кнопку «Open new session» на первой консоли или перейдя на <https://shell.azure.com> и войдя в систему с теми же учетными данными лаборатории.
7. Перейдите в папку с исходным кодом FabMedical и перечислите ее содержимое.
8.

```
cd ~/MCW-Cloud-native-applications/Hands-on\ lab/lab-files/developer/  
ls
```
9. Вы увидите, что список включает три папки: одну для веб-сайта, другую для API контента и одну для инициализации данных API:
10.

```
content-api/  
content-init/  
content-web/
```
12. Задайте свое имя пользователя и адрес электронной почты, которые git использует для коммитов.
13.

```
git config --global user.email "you@example.com"  
git config --global user.name "Your Name"
```
14. Используя Cloud Shell, инициализируйте новый репозиторий git:
15.

```
git init  
git add .  
git commit -m "Initial Commit"
```
17. Установите для удаленного источника URL-адрес GitHub, введя следующую команду:
- ```
git remote add origin <your GitHub URL>
```
18. Настройте git CLI для кэширования ваших учетных данных, чтобы вам не приходилось повторно вводить их.
19. 

```
git config --global --unset credential.helper
git config --global credential.helper store
```
20. Перейдите в основную ветку, введя следующую команд:
21. 

```
git branch -m master main
git push -u origin main
```
22. Обновите репозиторий GitHub, теперь вы должны увидеть опубликованный код.

## Задание 7: Подключение к build агенту

В этом разделе мы проверим, что сможем подключиться к новой виртуальной машине с агентом для сборки.

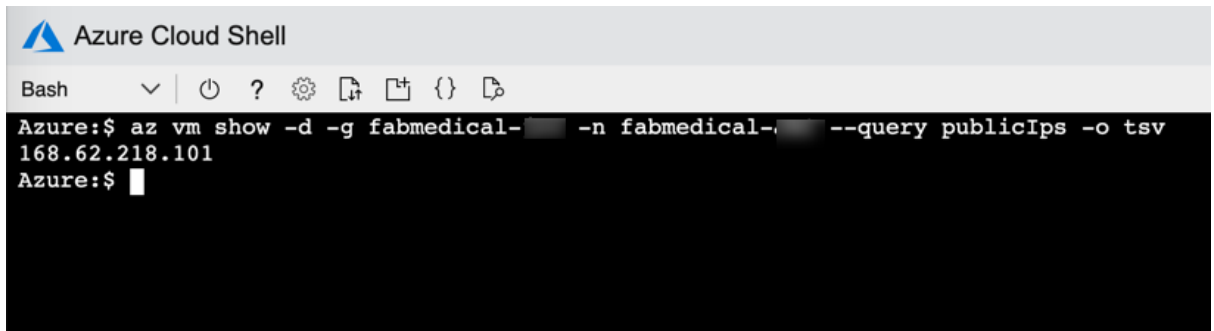
1. Откройте новую консоль Azure Cloud Shell и выполните следующую команду, чтобы найти IP-адрес виртуальной машины агента сборки, подготовленной при запуске развертывания ARM.:

```
az vm show -d -g fabmedical-[SUFFIX] -n fabmedical-[SHORT_SUFFIX] --
query publicIps -o tsv
```

Например:

```
az vm show -d -g fabmedical-sol -n fabmedical-SOL --query publicIps -o tsv
```

2. В ответе обратите внимание на общедоступный IP-адрес виртуальной машины..



```
Azure Cloud Shell
Bash
Azure:$ az vm show -d -g fabmedical-sol -n fabmedical-SOL --query publicIps -o tsv
168.62.218.101
Azure:$
```

3. Подключитесь к новой виртуальной машине, которую вы создали, введя следующую команду:

```
ssh -i [PRIVATEKEYNAME] [BUILDAGENTUSERNAME]@[BUILDAGENTIP]

ssh -i .ssh/fabmedical adminfabmedical@52.174.141.11
```

4. Когда вас попросят подтвердить, хотите ли вы подключиться, поскольку подлинность подключения не может быть подтверждена, введите «да».
5. Когда вас попросят ввести кодовую фразу для ранее созданного закрытого ключа, введите это значение.
6. SSH подключается к виртуальной машине и отображает следующую командную строку. Оставьте это окно облачной оболочки открытым для следующего шага:

```
adminfabmedical@fabmedical-SUFFIX:~$
```

```
Azure Cloud Shell
Bash
@Azure:~$ ssh -i .ssh/fabmedical adminfabmedical@23.97.213.98
The authenticity of host '23.97.213.98 (23.97.213.98)' can't be established.
ECDSA key fingerprint is SHA256:6ptcz3vihXphBPoqcZzc/qJEGdBm9VpfWGHMZn85Ck8.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '23.97.213.98' (ECDSA) to the list of known hosts.
Enter passphrase for key '.ssh/fabmedical':
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1063-azure x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

adminfabmedical@fabmedical:~$
```

## Задание 8: Завершите конфигурацию build агента

В этой задаче вы обновляете пакеты и устанавливаете Docker.

1. Перейдите в окно Cloud Shell, в котором открыто SSH-соединение с виртуальной машиной агента сборки.
2. Обновите пакеты Ubuntu и установите curl и поддержку репозитория через HTTPS за один шаг, набрав следующую команду в одной строке. Ответьте, набрав Y и нажав клавишу ВВОД, если вас спросят, хотите ли вы продолжить.

```
sudo apt-get update && sudo apt install apt-transport-https ca-
certificates curl software-properties-common
```

3. Добавьте официальный ключ GPG Docker, введя в однострочную команду следующее::

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-
key add -
```

4. Добавьте стабильный репозиторий Docker в список пакетов Ubuntu, набрав в однострочную команду следующую команду:

```
sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable"
```

5. Добавьте NodeJs PPA, чтобы использовать выпуск NodeJS LTS, обновить пакеты Ubuntu и установить механизм Docker, node.js и npm, введя следующие команды, каждую в отдельной строке. Если вас спросят, хотите ли вы продолжить, ответьте, набрав Y и нажав Enter.

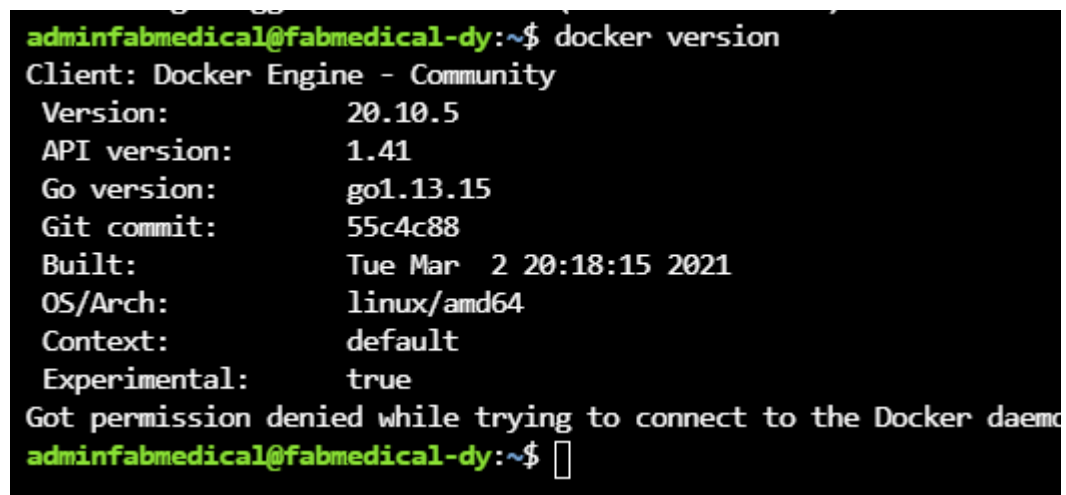
```
6. sudo apt-get install curl python-software-properties -y
7.
8. curl -sL https://deb.nodesource.com/setup_12.x | sudo -E bash -
9.
sudo apt-get update && sudo apt-get install -y docker-ce nodejs
mongodb-clients
```

10. Теперь обновите пакеты Ubuntu до последней версии, введя в однострочную команду следующее:.

```
sudo apt-get upgrade -y
```

11. Когда команда будет завершена, проверьте установленную версию Docker, выполнив эту команду. Результат может выглядеть примерно так, как показано на следующем снимке экрана. Обратите внимание, что версия сервера еще не отображается, потому что вы не запускали команду с повышенными привилегиями (мы рассмотрим это в ближайшее время).

```
docker version
```

A terminal window showing the output of the 'docker version' command. The prompt is 'adminfabmedical@fabmedical-dy:~\$'. The output lists Docker Engine - Community version 20.10.5, API version 1.41, Go version go1.13.15, Git commit 55c4c88, built on Tue Mar 2 20:18:15 2021, OS/Arch linux/amd64, context default, and experimental true. At the bottom, it says 'Got permission denied while trying to connect to the Docker daemon' and the prompt returns to 'adminfabmedical@fabmedical-dy:~\$' with a cursor.

```
adminfabmedical@fabmedical-dy:~$ docker version
Client: Docker Engine - Community
Version: 20.10.5
API version: 1.41
Go version: go1.13.15
Git commit: 55c4c88
Built: Tue Mar 2 20:18:15 2021
OS/Arch: linux/amd64
Context: default
Experimental: true
Got permission denied while trying to connect to the Docker daemon socket
adminfabmedical@fabmedical-dy:~$
```

12. Вы также можете проверить версии node.js и npm, просто для информации, используя эти команды:

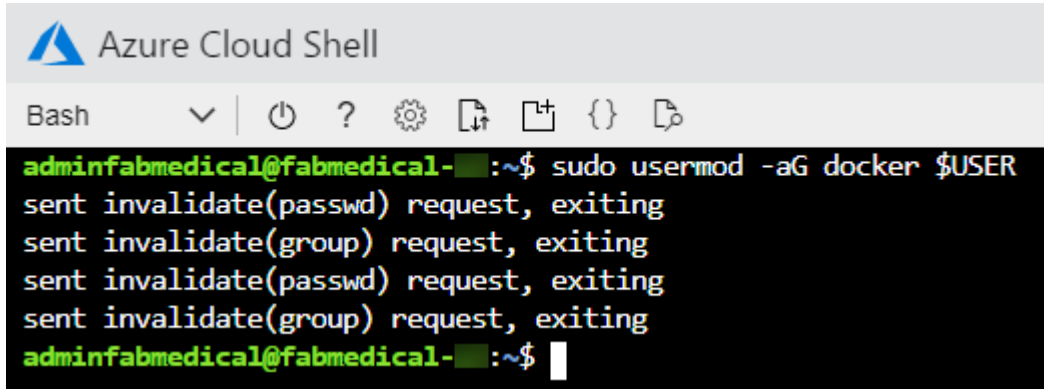
```
13. nodejs --version
14.
npm -version
```

15. Установите Angular CLI.

```
sudo npm install -g @angular/cli
```

16. Чтобы удалить требование использовать `sudo`, добавьте своего пользователя в группу Docker. Вы можете игнорировать любые ошибки, которые видите в выводе.

```
sudo usermod -aG docker $USER
```



```
Azure Cloud Shell
Bash
adminfabmedical@fabmedical-:~$ sudo usermod -aG docker $USER
sent invalidate(passwd) request, exiting
sent invalidate(group) request, exiting
sent invalidate(passwd) request, exiting
sent invalidate(group) request, exiting
adminfabmedical@fabmedical-:~$
```

17. Чтобы изменения прав пользователя вступили в силу, выйдите из сеанса SSH, набрав `exit`, затем нажмите <Enter>. Повторно подключитесь к виртуальной машине агента сборки с помощью SSH, как вы это делали в предыдущей задаче.

```
ssh -i .ssh/fabmedical adminfabmedical@52.174.141.11
```

18. Повторите команду `docker version` и обратите внимание, что в выводе теперь также отображается версия сервера.

```
adminfabmedical@fabmedical-dy:~$ docker version
Client: Docker Engine - Community
Version: 20.10.5
API version: 1.41
Go version: go1.13.15
Git commit: 55c4c88
Built: Tue Mar 2 20:18:15 2021
OS/Arch: linux/amd64
Context: default
Experimental: true

Server: Docker Engine - Community
Engine:
Version: 20.10.5
API version: 1.41 (minimum version 1.12)
Go version: go1.13.15
Git commit: 363e9a8
Built: Tue Mar 2 20:16:12 2021
OS/Arch: linux/amd64
Experimental: false
containerd:
Version: 1.4.4
GitCommit: 05f951a3781f4f2c1911b05e61c160e9c30eaa8e
runc:
Version: 1.0.0-rc93
GitCommit: 12644e614e25b05da6fd08a38ffa0cfe1903fdec
docker-init:
Version: 0.19.0
GitCommit: de40ad0
```

19. Выполните несколько команд Docker:

```
docker container ls
docker container ls -a
```

20. В обоих случаях у вас есть пустой список, но нет ошибок при выполнении команды. Ваш агент сборки готов, и движок Docker работает правильно.

```
Azure Cloud Shell
Bash
adminfabmedical@fabmedical-:~$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
adminfabmedical@fabmedical-:~$ docker container ls -a
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
adminfabmedical@fabmedical-:~$
```

## Задание 9: Склонируйте репозиторий на Build агента

В этой задаче вы клонируете свои репозитории из GitHub, чтобы вы могли работать с ними в агенте сборки.



1. Как и ранее в облачной оболочке, укажите имя пользователя и адрес электронной почты, которые используются для коммитов git.
2. 

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
```

**Note:** В некоторых случаях пользователь root владеет папкой .config вашего пользователя. Если это произойдет, выполните следующую команду, чтобы вернуть право собственности на adminfabmedical, а затем попробуйте команду git еще раз.:

```
sudo chown -R $USER:$(id -gn $USER)
/home/adminfabmedical/.config
```

3. Настройте git CLI для кеширования учетных данных, чтобы вам не приходилось повторно вводить их.

```
git config --global credential.helper cache
```

4. Используйте URL-адрес GitHub, чтобы клонировать код репозитория на свой компьютер с агентом сборки.

```
git clone <GITHUB_REPOSITORY_URL>
```

## Задание 10: Запустите стартовое приложение

В этой задаче вы возьмете стартовые файлы и запустите приложение node.js как приложение Docker. Вы создадите образы Docker из существующих файлов и запустите контейнеры для тестирования и выполнения приложения.

1. Из Azure Cloud Shell подключитесь к своему агенту сборки, если вы еще не подключены.
2. Введите следующую команду, чтобы создать сеть Docker с именем fabmedical:

```
docker network create fabmedical
```

3. Запустите экземпляр mongodb, чтобы использовать его для локального тестирования.

```
docker container run --name mongo --net fabmedical -p 27017:27017 -d
mongo:4.0
```

**Note:** С существующим исходным кодом, написанным для MongoDB, его можно указать на конечную точку API MongoDB в Azure Cosmos DB. Эмулятор Azure Cosmos DB можно использовать для локальной разработки в Windows; однако эмулятор Cosmos DB не поддерживает Linux. В результате при использовании Linux для разработки MongoDB по-прежнему требуется для локальных сред разработки; с использованием Azure Cosmos DB для хранения данных в облаке. Это позволяет легко перенести существующий исходный код, написанный для хранилища MongoDB, на серверную часть Azure Cosmos DB.

4. Убедитесь, что контейнер mongo запущен и готов.
5. 

```
docker container list
```

```
docker container logs mongo
```

[illegible]

6. Подключитесь к экземпляру `mongo` с помощью оболочки `mongo` и проверьте некоторые основные команды:

```
mongo
show dbs
quit()
```

```
Bash | ? | ? | ? | ? | ? | ?
```

```
adminfabmedical@fabmedical:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
http://docs.mongodb.org/
Questions? Try the support group
http://groups.google.com/group/mongodb-user

Server has startup warnings:
2019-10-04T12:45:09.692+0000 I STORAGE [initandlisten]
2019-10-04T12:45:09.692+0000 I STORAGE [initandlisten] ** WARNING: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine
See http://dochub.mongodb.org/core/prodnotes-filesystem
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten] ** WARNING: Access control is not enabled for the database.
Read and write access to data and configuration is unrestricted.
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten] ** WARNING: /sys/kernel/mm/transparent_hugepage/enabled is 'always'.
We suggest setting it to 'never'
2019-10-04T12:45:10.630+0000 I CONTROL [initandlisten]

> show dbs
admin 0.000GB
config 0.000GB
local 0.000GB
> quit()

adminfabmedical@fabmedical:~$
```

7. Чтобы инициализировать локальную базу данных тестовым содержимым, сначала перейдите в каталог `content-init` и запустите `npm install`.
8. 

```
cd ~/Fabmedical/content-init
npm install
```
9. Инициализируйте базу данных.

```
nodejs server.js
```

```
Bash
```

```
adminfabmedical@fabmedical-:~$ cd content-init/
adminfabmedical@fabmedical-:~/content-init$ npm install
npm WARN content-init@1.0.0 No description
npm WARN content-init@1.0.0 No repository field.

added 28 packages from 17 contributors and audited 35 packages in 1.61s
found 0 vulnerabilities

adminfabmedical@fabmedical:~/content-init$ nodejs server.js
Clean Sessions table
(node:9955) DeprecationWarning: collection.remove is deprecated. Use deleteOne, deleteMany, or bulkWrite instead.
Connected to MongoDB
All Sessions deleted
Load sessions from JSON file
Session saved successfully
Session saved successfully
Session saved successfully
Session saved successfully
Clean Speakers table
All Speakers deleted
Load Speakers from JSON file
Speaker saved successfully
Speaker saved successfully
Speaker saved successfully
Speaker saved successfully
adminfabmedical@fabmedical-:~/content-init$
```

10. Убедитесь, что база данных теперь содержит тестовые данные.

```
mongo
show dbs
use contentdb
show collections
db.speakers.find()
db.sessions.find()
quit()
```

Это должно привести к выводу, подобному следующему:

[illegible]

11. Теперь перейдите в каталог `content-api` и запустите `npm install`.

```
12. cd ../content-api
 npm install
```

### 13. Запустите API в фоновом режиме.

```
nodejs ./server.js &
```





```
Azure Cloud Shell
Bash
es/karma/node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.9 (node_modules/@angular/compiler-cli/node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.9: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.0.7 (node_modules/fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.0.7: wanted {"os":"darwin","arch":"any"} (current: {"os":"linux","arch":"x64"})

audited 1459 packages in 8.957s
found 780 vulnerabilities (757 low, 5 moderate, 18 high)
 run `npm audit fix` to fix them, or `npm audit` for details
adminfabmedical@fabmedical-cp4:~/Fabmedical/content-web$ ng build
Your global Angular CLI version (10.0.7) is greater than your local version (8.3.4). The local Angular CLI version is used.

To disable this warning use "ng config -g cli.warnings.versionMismatch false".

chunk {main} main.js, main.js.map (main) 54.3 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 282 kB [initial] [rendered]
chunk {polyfills-es5} polyfills-es5.js, polyfills-es5.js.map (polyfills-es5) 602 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 9.72 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 3.87 MB [initial] [rendered]
Date: 2020-08-24T02:16:08.748Z - Hash: ba53090244b74f736cb1 - Time: 11137ms
adminfabmedical@fabmedical-cp4:~/Fabmedical/content-web$
```

19. В Azure Cloud Shell выполните следующую команду, чтобы найти IP-адрес виртуальной машины агента сборки, подготовленной при запуске развертывания ARM.

```
az vm show -d -g fabmedical-[SUFFIX] -n fabmedical-[SHORT_SUFFIX] --query publicIps -o tsv
```

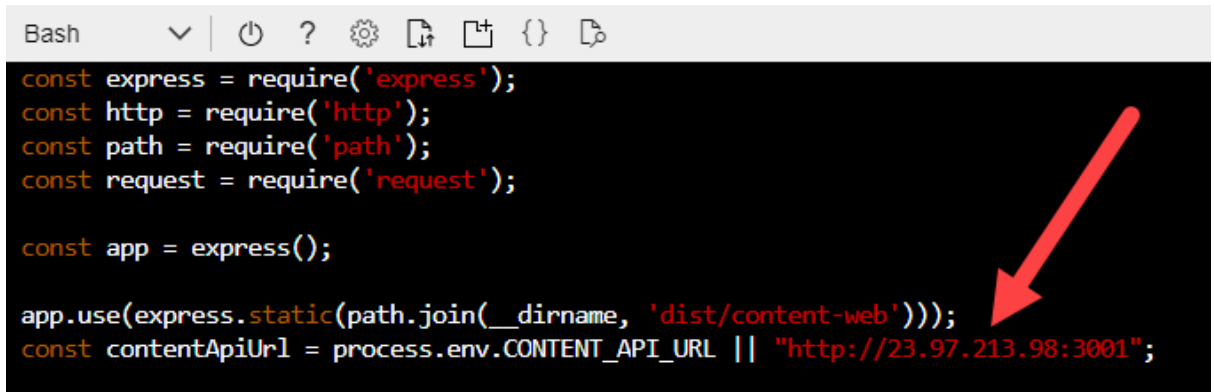
Например:

```
az vm show -d -g fabmedical-sol -n fabmedical-SOL --query publicIps -o tsv
```

20. Из облачной оболочки на машине сборки отредактируйте файл app.js с помощью vim.

```
vim app.js
```

Затем нажмите `i`, чтобы перейти в режим редактирования, после чего замените `localhost` на IP-адрес машины сборки.



```
Bash
const express = require('express');
const http = require('http');
const path = require('path');
const request = require('request');

const app = express();

app.use(express.static(path.join(__dirname, 'dist/content-web')));
const contentApiUrl = process.env.CONTENT_API_URL || "http://23.97.213.98:3001";
```

Затем нажмите `ESC`, напишите: `wq`, чтобы сохранить изменения и закрыть файл.

21. Теперь запустите контент-веб-приложение в фоновом режиме.









```
node ./app.js &
```

Снова нажмите клавишу `ВВОД`, чтобы получить командную строку для следующего шага.

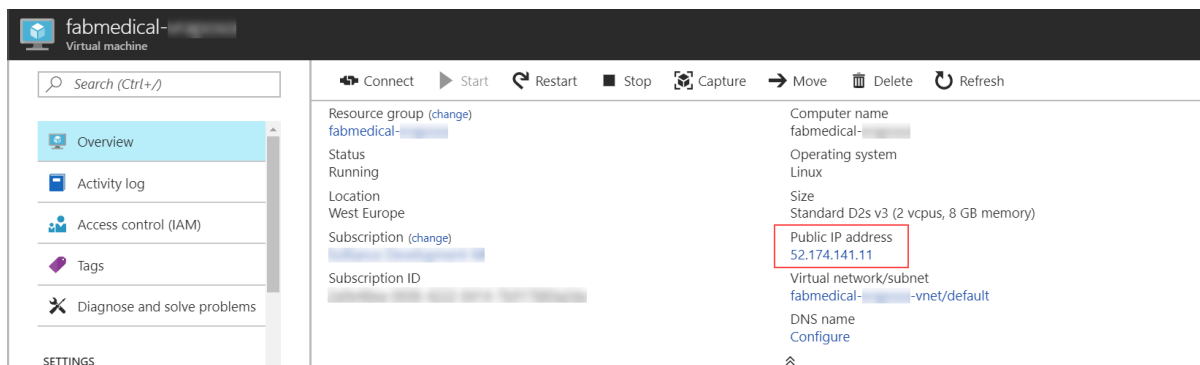
22. Протестируйте веб-приложение с помощью `curl`. Вы увидите, что вывод HTML возвращается без ошибок.

```
curl http://localhost:3000
```

23. Оставьте приложение запущенным для выполнения следующей задачи.
24. Если вы получили ответ JSON на запрос содержимого / динамиков и ответ HTML от веб-приложения, ваша среда работает должным образом.
25. На портале Azure выберите созданную вами группу ресурсов с именем `fabmedical-SUFFIX`.
26. Выберите виртуальную машину агента сборки с именем `fabmedical-SUFFIX` из списка доступных ресурсов.

| NAME ▾                                                                                                   | TYPE ▾                  | LOCATION ▾ |     |
|----------------------------------------------------------------------------------------------------------|-------------------------|------------|-----|
|  fabmedical-soll        | Virtual machine         | East US 2  | ... |
|  LinuxAsm               | Microsoft.Compute/vi... | East US 2  | ... |
|  fabmedical-soll261     | Network interface       | East US 2  | ... |
|  fabmedicalsolldiag273  | Storage account         | East US 2  | ... |
|  fabmedicalsolldisks565 | Storage account         | East US 2  | ... |
|  fabmedical-soll-ip     | Public IP address       | East US 2  | ... |
|  fabmedical-soll-nsg    | Network security group  | East US 2  | ... |
|  fabmedical-soll-vnet   | Virtual network         | East US 2  | ... |

22. В обзоре виртуальной машины найдите IP-адрес виртуальной машины.



The screenshot shows the 'Overview' tab for a virtual machine. The 'Public IP address' field is highlighted with a red box, displaying the IP address 52.174.141.11. Other details visible include the resource group 'fabmedical-', status 'Running', location 'West Europe', and subscription ID.

23. Протестируйте веб-приложение в браузере. Перейдите к веб-приложению, используя IP-адрес вашего агента сборки на порту 3000.

`http://[BUILDAGENTIP]:3000`

EXAMPLE: `http://13.68.113.176:3000`

24. Выберите ссылки «Спикеры» и «Сеансы» в заголовке. Вы увидите, что страницы отображают HTML-версию содержимого JSON, которое вы скручивали ранее.

25. Убедившись, что приложение доступно через браузер, перейдите в окно облачной оболочки и остановите запущенные процессы узла.

```
killall nodejs
killall node
```

## Задание 11: Сборка Docker образов

В этой задаче вы создадите образы Docker для приложения - один для приложения API, а другой - для веб-приложения. Каждый образ будет создан с помощью команд Docker, которые полагаются на Dockerfile.

1. В Cloud Shell, подключенном к виртуальной машине агента сборки, введите следующую команду, чтобы просмотреть любые образы Docker на виртуальной машине. В списке будет только образ mongodb, загруженный ранее.

```
docker image ls
```

2. В папке content-api (cd ~ / Fabmedical / content-api), содержащей файлы приложения API и файл Dockerfile, введите следующую команду, чтобы создать образ Docker для приложения API. Эта команда выполняет следующие действия:
  - Выполняет команду сборки Docker для создания образа
  - Помечает полученное изображение с именем content-api (-t)
  - Последняя точка (.) указывает на использование Dockerfile в контексте текущего каталога. По умолчанию ожидается, что этот файл будет иметь имя Dockerfile (с учетом регистра).

```
docker image build -t content-api .
```

3. После успешного создания образа снова запустите команду Docker images list. Вы увидите несколько новых изображений: изображения узлов и изображение вашего контейнера.

```
docker image ls
```

Обратите внимание на немаркированное изображение. Это этап сборки, который содержит все промежуточные файлы, которые не нужны в вашем окончательном образе.

```
adminfabmedical@fabmedical- :~/Fabmedical/content-api$ docker image ls
```

| REPOSITORY  | TAG    | IMAGE ID     | CREATED            | SIZE  |
|-------------|--------|--------------|--------------------|-------|
| content-api | latest | 798d89b8f0aa | About a minute ago | 129MB |
| <none>      | <none> | 8a86289adce3 | About a minute ago | 674MB |
| mongo       | latest | 409c3f937574 | 4 days ago         | 493MB |
| node        | alpine | 0f2c18cef5d3 | 11 days ago        | 117MB |
| node        | argon  | ef4b194d8fcf | 2 years ago        | 653MB |

```
adminfabmedical@fabmedical- :~/Fabmedical/content-api$
```

4. Снова перейдите в папку content-web и выведите список файлов. Обратите внимание, что в этой папке также есть Dockerfile.

5. cd ../content-web  
ll

6. Просмотрите содержимое Dockerfile. Введите следующую команду:

```
cat Dockerfile
```

Обратите внимание, что этап сборки файла Dockerfile с веб-контентом включает в себя дополнительные инструменты для интерфейсного приложения Angular в дополнение к установке пакетов npm.



7. Введите следующую команду, чтобы создать образ Docker для веб-приложения.

```
docker image build -t content-web .
```

8. Снова перейдите в папку content-init и перечислите файлы. Обратите внимание, что в этой папке уже есть Dockerfile.

```
9. cd ../content-init
 ll
```

10. Просмотрите содержимое Dockerfile. Введите следующую команду:

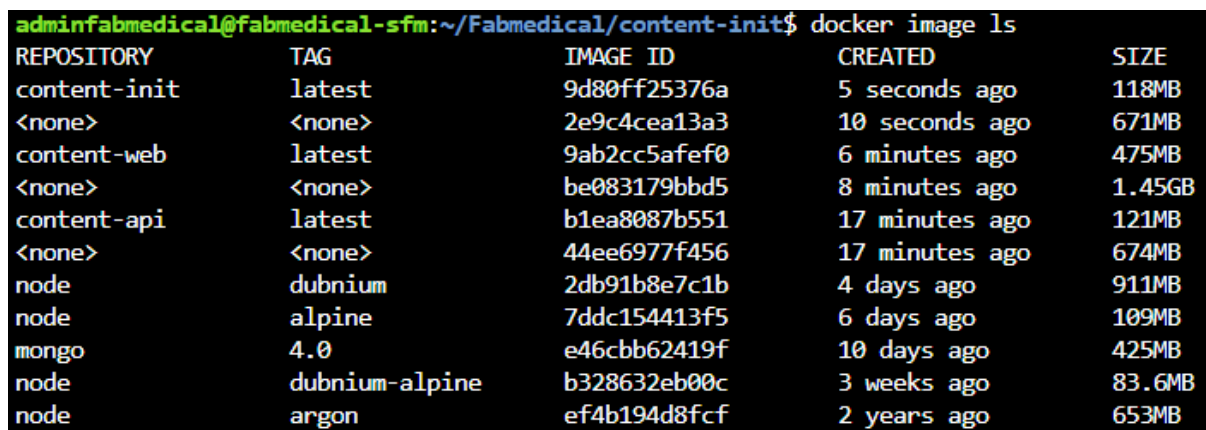
```
cat Dockerfile
```

11. Введите следующую команду, чтобы создать образ Docker для приложения инициализации.

```
docker image build -t content-init .
```

12. По завершении вы увидите восемь образов, которые теперь существуют, когда вы запустите команду Docker images.

```
docker image ls
```

A screenshot of a terminal window showing the output of the 'docker image ls' command. The terminal has a dark background with green and blue text for the prompt and command. The output is a table with five columns: REPOSITORY, TAG, IMAGE ID, CREATED, and SIZE. It lists eight Docker images, including 'content-init', 'content-web', 'content-api', and various base images like 'node', 'mongo', and 'argon'.

| REPOSITORY   | TAG            | IMAGE ID     | CREATED        | SIZE   |
|--------------|----------------|--------------|----------------|--------|
| content-init | latest         | 9d80ff25376a | 5 seconds ago  | 118MB  |
| <none>       | <none>         | 2e9c4cea13a3 | 10 seconds ago | 671MB  |
| content-web  | latest         | 9ab2cc5afef0 | 6 minutes ago  | 475MB  |
| <none>       | <none>         | be083179bbd5 | 8 minutes ago  | 1.45GB |
| content-api  | latest         | b1ea8087b551 | 17 minutes ago | 121MB  |
| <none>       | <none>         | 44ee6977f456 | 17 minutes ago | 674MB  |
| node         | dubnium        | 2db91b8e7c1b | 4 days ago     | 911MB  |
| node         | alpine         | 7ddc154413f5 | 6 days ago     | 109MB  |
| mongo        | 4.0            | e46cbb62419f | 10 days ago    | 425MB  |
| node         | dubnium-alpine | b328632eb00c | 3 weeks ago    | 83.6MB |
| node         | argon          | ef4b194d8fcf | 2 years ago    | 653MB  |

## Задание 12: Запуск Docker контейнеров

Контейнер веб-приложения будет вызывать конечные точки, предоставляемые контейнером приложения API, а контейнер приложения API будет взаимодействовать с mongodb. В этом упражнении вы запустите созданные вами образы как контейнеры в той же сети моста, которую вы создали при запуске mongodb.

1. Создайте и запустите контейнер приложения API с помощью следующей команды. Команда делает следующее:
  - Называет контейнер api для дальнейшего использования с командами Docker.
  - Дает указание движку Docker использовать сеть fabmedical.
  - Указывает подсистеме Docker использовать порт 3001 и сопоставить его с внутренним портом контейнера 3001.

- Создает контейнер из указанного изображения по его тегу, например content-api.

```
docker container run --name api --net fabmedical -p 3001:3001
content-api
```

2. Команда запуска контейнера докеров не удалась, поскольку она настроена для подключения к mongodb с использованием URL-адреса localhost. Однако теперь, когда content-api изолирован в отдельном контейнере, он не может получить доступ к mongodb через localhost даже при работе на том же хосте докеров. Вместо этого API должен использовать мостовую сеть для подключения к mongodb.

```
3. > content-api@0.0.0 start
4. > node ./server.js
5.
6. Listening on port 3001
7. Could not connect to MongoDB!
8. MongooseServerSelectionError: connect ECONNREFUSED 127.0.0.1:27017
9. npm notice
10. npm notice New patch version of npm available! 7.0.8 -> 7.0.13
11. npm notice Changelog:
 <https://github.com/npm/cli/releases/tag/v7.0.13>
12. npm notice Run `npm install -g npm@7.0.13` to update!
13. npm notice
14. npm ERR! code 255
15. npm ERR! path /usr/src/app
16. npm ERR! command failed
17. npm ERR! command sh -c node ./server.js
18.
19. npm ERR! A complete log of this run can be found in:
20. npm ERR! /root/.npm/_logs/2020-11-23T03_04_12_948Z-debug.log
```

21. Приложение content-api позволяет переменной среды настраивать строку подключения mongodb. Удалите существующий контейнер, а затем проинструктируйте движок докеров установить переменную среды, добавив переключатель -e в команду запуска контейнера докеров. Кроме того, используйте переключатель -d для запуска api как демона.

```
22. docker container rm api
docker container run --name api --net fabmedical -p 3001:3001 -e
MONGODB_CONNECTION=mongodb://mongo:27017/contentdb -d content-api
```

23. Введите команду, чтобы показать запущенные контейнеры. Вы заметите, что контейнер api находится в списке. Используйте команду docker logs, чтобы увидеть, что приложение API подключилось к mongodb.

```
24. docker container ls
docker container logs api
```

```
adminfabmedical@fabmedical-cp4:~/Fabmedical/content-web$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
122437f5a5c4 content-api "docker-entrypoint.s..." 27 seconds ago Up 26 seconds 0.0.0.0:3001->3001/tcp api
ef8d527320a1 mongo "docker-entrypoint.s..." 58 minutes ago Up 58 minutes 0.0.0.0:27017->27017/tcp mongo
adminfabmedical@fabmedical-cp4:~/Fabmedical/content-web$ docker container logs api
> content-api@0.0.0 start /usr/src/app
> node ./server.js

Listening on port 3001
Connected to MongoDB
adminfabmedical@fabmedical-cp4:~/Fabmedical/content-web$
```

25. Протестируйте API, свернув URL-адрес. Вы увидите вывод JSON, как и при предыдущем тестировании.

```
curl http://localhost:3001/speakers
```

26. Создайте и запустите контейнер веб-приложения с помощью аналогичной команды запуска контейнера докера - укажите движку докера использовать любой порт с помощью команды -P.

```
docker container run --name web --net fabmedical -P -d content-web
```

27. Введите команду, чтобы снова показать запущенные контейнеры, и вы увидите, что в списке есть и API, и веб-контейнеры. Веб-контейнер показывает динамически назначаемое сопоставление порта с внутренним портом контейнера 3000.

```
docker container ls
```

```
adminfabmedical@fabmedical:~$ docker container ls
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
aa76f2d69b38 content-web "docker-entrypoint.s..." 18 seconds ago Up 17 seconds 0.0.0.0:32768->3000/tcp web
122437f5a5c4 content-api "docker-entrypoint.s..." 2 minutes ago Up 2 minutes 0.0.0.0:3001->3001/tcp api
ef8d527320a1 mongo "docker-entrypoint.s..." About an hour ago Up About an hour 0.0.0.0:27017->27017/tcp mongo
```

28. Протестируйте веб-приложение, получив URL-адрес с помощью curl. Для порта используйте динамически назначаемый порт, который вы можете найти в выходных данных предыдущей команды. Вы увидите вывод HTML, как и при предыдущем тестировании.

```
curl http://localhost:[PORT]/speakers.html
```

### Задание 13: Конфигурация переменных окружения

В этой задаче вы настроите веб-контейнер для связи с контейнером API с помощью переменной среды, аналогично тому, как строка подключения mongodb предоставляется API-интерфейсу.

1. В Cloud Shell, подключенном к виртуальной машине агента сборки, остановите и удалите веб-контейнер, используя следующие команды.
2. 

```
docker container stop web
docker container rm web
```
3. Убедитесь, что веб-контейнер больше не работает или не присутствует, используя флаг -a, как показано в этой команде. Вы увидите, что веб-контейнера больше нет в списке.

```
docker container ls -a
```

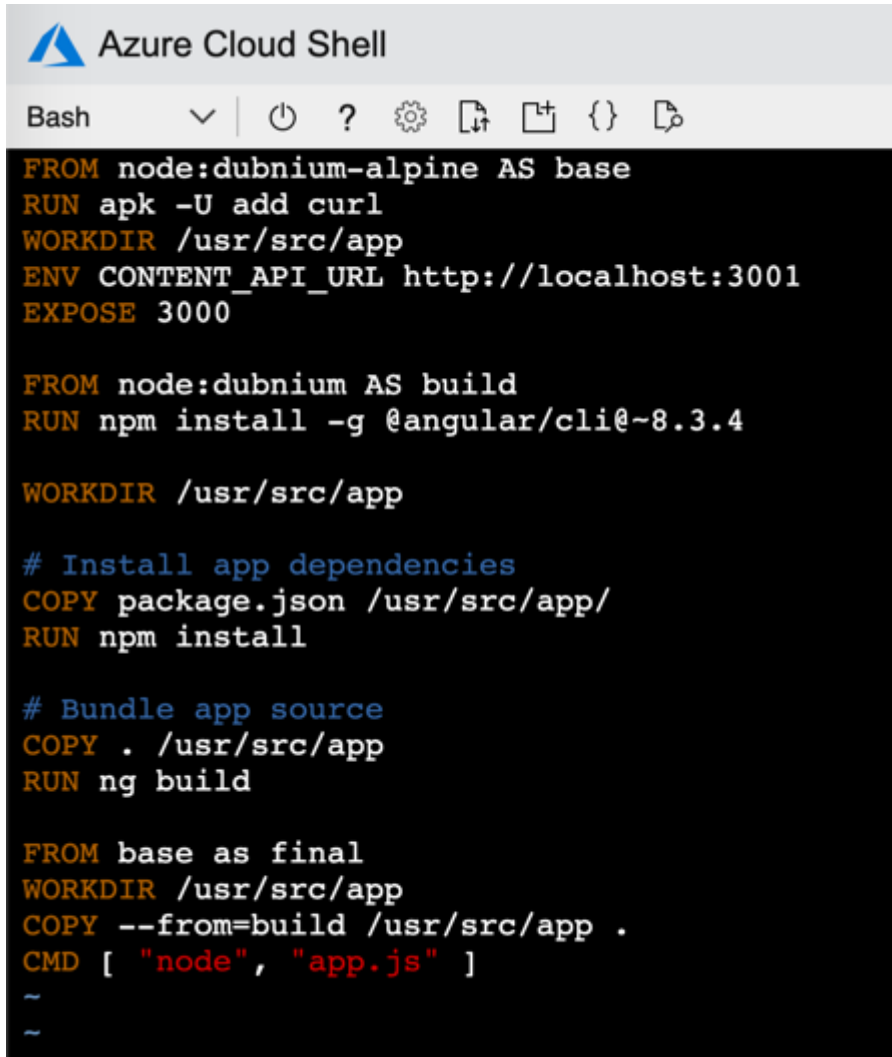
4. Проверьте app.js
5. 

```
cd ../content-web
cat app.js
```
6. Обратите внимание, что переменная contentApiUrl может быть установлена с помощью переменной среды.

```
const contentApiUrl = process.env.CONTENT_API_URL || "http://[VM IP]:3001";
```

7. Откройте Dockerfile для редактирования с помощью Vim и нажмите клавишу i, чтобы перейти в режим редактирования.
8. vi Dockerfile  
<i>
9. Найдите строку EXPOSE, показанную ниже, и добавьте строку над ней, которая устанавливает значение по умолчанию для переменной среды, как показано на снимке экрана.

```
ENV CONTENT_API_URL http://localhost:3001
```

The image shows a screenshot of the Azure Cloud Shell interface. At the top, there's a header with the Azure logo and the text "Azure Cloud Shell". Below the header is a toolbar with icons for "Bash", a dropdown menu, a power button, a help icon, a settings gear, a file upload icon, a file download icon, a code editor icon, and a terminal icon. The main area of the shell is a dark terminal window displaying the content of a Dockerfile. The Dockerfile instructions are as follows:

```
FROM node:dubnium-alpine AS base
RUN apk -U add curl
WORKDIR /usr/src/app
ENV CONTENT_API_URL http://localhost:3001
EXPOSE 3000

FROM node:dubnium AS build
RUN npm install -g @angular/cli@~8.3.4

WORKDIR /usr/src/app

Install app dependencies
COPY package.json /usr/src/app/
RUN npm install

Bundle app source
COPY . /usr/src/app
RUN ng build

FROM base as final
WORKDIR /usr/src/app
COPY --from=build /usr/src/app .
CMD ["node", "app.js"]
```

10. Восстановите образ Docker веб-приложения, используя ту же команду, что и раньше.

```
docker image build -t content-web .
```

11. Создайте и запустите образ, передав правильный URI в контейнер API в качестве переменной среды. Эта переменная будет обращаться к приложению API, используя его имя контейнера в созданной вами сети Docker. После запуска контейнера убедитесь, что он запущен, и обратите внимание на динамическое назначение порта для следующего шага.

```
12. docker container run --name web --net fabmedical -P -d -e
 CONTENT_API_URL=http://api:3001 content-web
 docker container ls
```

13. Снова выполните команду `curl`, используя порт, назначенный веб-контейнеру. Опять же, вы увидите возвращенный HTML, но поскольку `curl` не обрабатывает javascript, вы не можете определить, взаимодействует ли веб-приложение с приложением `api`. Вы должны проверить это соединение в браузере.

```
curl http://localhost:[PORT]/speakers.html
```

14. Вы не сможете просматривать веб-приложение на временном порте, потому что виртуальная машина предоставляет только ограниченный диапазон портов. Теперь вы остановите веб-контейнер и перезапустите его, используя порт 3000 для тестирования в браузере. Введите следующие команды, чтобы остановить контейнер, удалить его и снова запустить, используя явные настройки для порта.

```
15. docker container stop web
16. docker container rm web
 docker container run --name web --net fabmedical -p 3000:3000 -d -e
 CONTENT_API_URL=http://api:3001 content-web
```

17. Используйте `Curl` снова на порту 3000. Вы увидите тот же HTML-код.

```
curl http://localhost:3000/speakers.html
```

18. Теперь вы можете использовать веб-браузер для перехода на веб-сайт и успешного просмотра приложения через порт 3000. Замените `[BUILDAGENTIP]` на IP-адрес, который вы использовали ранее.

```
19. http://[BUILDAGENTIP]:3000
```

```
20.
 EXAMPLE: http://13.68.113.176:3000
```

21. Зафиксируйте свои изменения и отправьте в репозиторий.

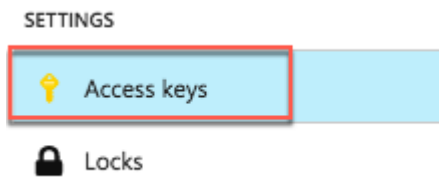
```
22. cd ~/Fabmedical
23. git add .
24. git commit -m "Setup Environment Variables"
 git push
```

## Задание 14: Отправка образов в Azure Container Registry

Для запуска контейнеров в удаленной среде вы обычно отправляете образы в реестр Docker, где вы можете хранить и распространять образы. У каждой службы будет репозиторий, который можно отправлять и извлекать с помощью команд Docker. Реестр контейнеров Azure (ACR) - это управляемая частная служба реестра Docker, основанная на реестре Docker v2.

В этой задаче вы отправите образы в свою учетную запись ACR, образы версий с тегами и настроите непрерывную интеграцию (CI) для создания будущих версий ваших контейнеров и автоматически отправите их в ACR.

1. На портале Azure перейдите к ACR, который вы создали перед практической лабораторной работой.
2. Выберите **Access keys** в разделе **Settings**.

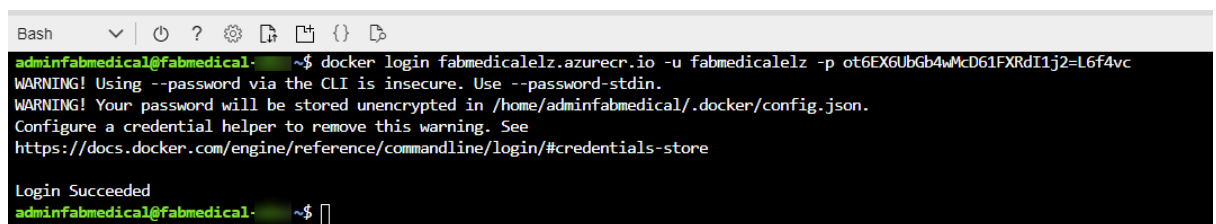


3. В сеансе облачной оболочки, подключенном к вашей виртуальной машине сборки, войдите в свою учетную запись ACR, введя следующую команду. Следуйте инструкциям для завершения входа в систему.

```
docker login [LOGINSERVER] -u [USERNAME] -p [PASSWORD]
```

Например:

```
docker login fabmedicalsoll.azurecr.io -u fabmedicalsoll -p
+W/j=1+Fcze=n07SchxvGS1vsLRh/7ga
```



```
Bash
adminfabmedical@fabmedical- ~$ docker login fabmedicalelz.azurecr.io -u fabmedicalelz -p ot6EX6UbGb4wMcD61FXRdI1j2=L6f4vc
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /home/adminfabmedical/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
adminfabmedical@fabmedical- ~$
```

4. Выполните следующие команды, чтобы правильно пометить изображения, чтобы они соответствовали имени вашей учетной записи ACR.
5. 

```
docker image tag content-web [LOGINSERVER]/content-web
```

```
docker image tag content-api [LOGINSERVER]/content-api
```

**Note:** Be sure to replace the [LOGINSERVER] of your ACR instance.

6. Перечислите свои образы докеров и посмотрите репозиторий и тег. Обратите внимание, что к репозиторию добавляется префикс вашего имени сервера входа в ACR, как в примере, показанном на снимке экрана ниже.

```
docker image ls
```

```
adminfabmedical@fabmedical-cnr:~/Fabmedical/content-web$ docker image ls
```

| REPOSITORY                           | TAG            | IMAGE ID     | CREATED        | SIZE   |
|--------------------------------------|----------------|--------------|----------------|--------|
| content-web                          | latest         | 2bbdad714d6  | 20 minutes ago | 474MB  |
| fabmedicalcnr.azurecr.io/content-web | latest         | 2bbdad714d6  | 20 minutes ago | 474MB  |
| <none>                               | <none>         | 3b3492c394a3 | 21 minutes ago | 1.46GB |
| content-init                         | latest         | 934535501206 | 29 minutes ago | 122MB  |
| <none>                               | <none>         | 64b029c7be16 | 29 minutes ago | 678MB  |
| <none>                               | <none>         | 2bcdaa0e5f59 | 30 minutes ago | 474MB  |
| <none>                               | <none>         | 4f1bd3b306d6 | 31 minutes ago | 1.46GB |
| content-api                          | latest         | 343bcea66675 | 58 minutes ago | 125MB  |
| fabmedicalcnr.azurecr.io/content-api | latest         | 343bcea66675 | 58 minutes ago | 125MB  |
| <none>                               | <none>         | 7b6782c45805 | 58 minutes ago | 681MB  |
| node                                 | alpine         | 3d9a0ea05233 | 2 days ago     | 112MB  |
| node                                 | dubnium-alpine | 9bfaa3adc4e1 | 3 days ago     | 82.7MB |
| node                                 | dubnium        | b15b7f33f1e6 | 4 days ago     | 910MB  |
| mongo                                | 4.0            | b0df4f866657 | 9 days ago     | 428MB  |
| node                                 | argon          | ef4b194d8fcf | 2 years ago    | 653MB  |

```
adminfabmedical@fabmedical-cnr:~/Fabmedical/content-web$
```

7. Отправьте образы в свою учетную запись ACR с помощью следующей команды:

8. `docker image push [LOGINSERVER]/content-web`  
`docker image push [LOGINSERVER]/content-api`

```
Bash
```

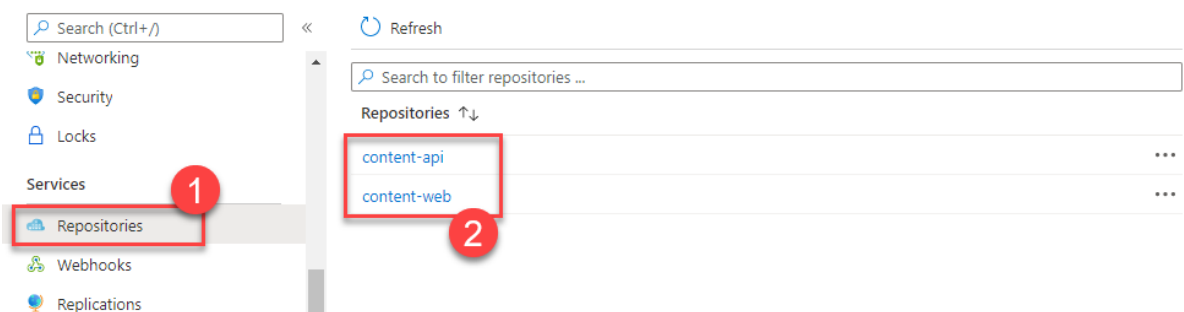
```
adminfabmedical@fabmedical-:~$ docker image push fabmedical.azurecr.io/content-web
```

The push refers to repository [fabmedical.azurecr.io/content-web]

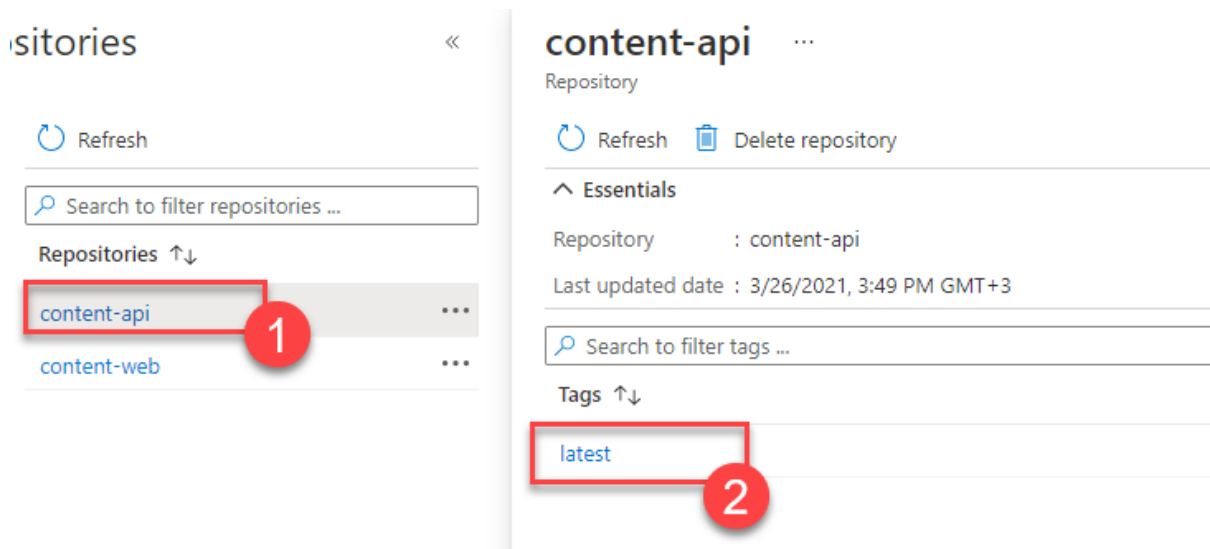
```
ac4f3c73d0c1: Pushed
d265965e417c: Pushed
e2087b61ba4d: Pushed
d684f266314f: Pushed
49c743f4257f: Pushed
e02756f841b2: Pushed
77cae8ab23bf: Pushed
latest: digest: sha256:99825a8bbc535a6cd0675f9559014ea83ad77827a7470b8ae17a5f681037c531 size: 1788
```

```
adminfabmedical@fabmedical-:~$
```

9. На портале Azure перейдите к своей учетной записи ACR и выберите «Репозитории» в разделе «Службы» в меню слева. Теперь вы увидите два контейнера (2), по одному для каждого изображения.



10. Выберите content-api (1). Вы увидите, что последний тег (2) назначен.



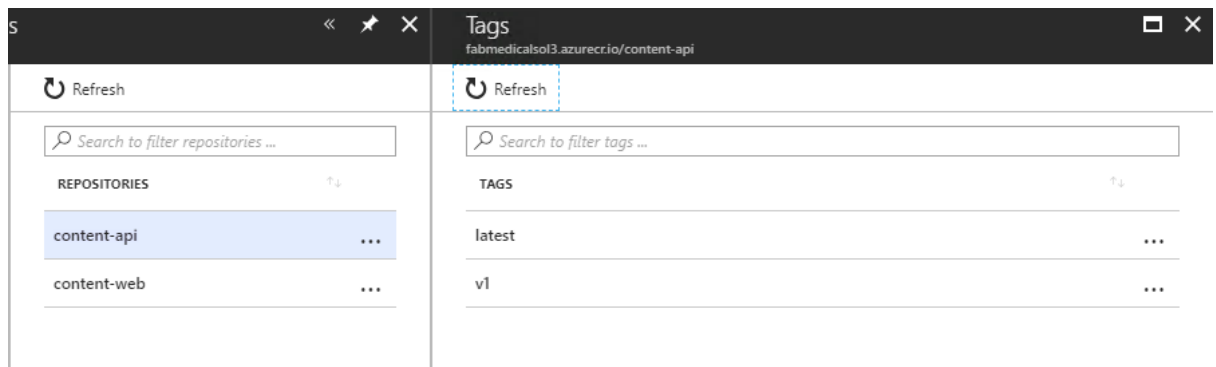
11. В сеансе облачной оболочки, подключенном к виртуальной машине, назначьте тег v1 каждому изображению с помощью следующих команд. Затем перечислите образы Docker, чтобы отметить, что теперь для каждого образа есть две записи: с указанием последнего тега latest и тега v1. Также обратите внимание, что ID изображения одинаков для двух записей, так как имеется только одна копия изображения.
12. `docker image tag [LOGINSERVER]/content-web:latest [LOGINSERVER]/content-web:v1`
13. `docker image tag [LOGINSERVER]/content-api:latest [LOGINSERVER]/content-api:v1`  
`docker image ls`

```
adminfabmedical@fabmedical-cnr:~/Fabmedical/content-web$ docker image ls
REPOSITORY TAG IMAGE ID CREATED SIZE
content-web latest 2bbdadc714d6 25 minutes ago 474MB
fabmedicalcnr.azurecr.io/content-web latest 2bbdadc714d6 25 minutes ago 474MB
fabmedicalcnr.azurecr.io/content-web v1 2bbdadc714d6 25 minutes ago 474MB
<none> <none> 3b3492c394a3 26 minutes ago 1.46GB
content-init latest 934535501206 33 minutes ago 122MB
<none> <none> 64b029c7be16 33 minutes ago 678MB
<none> <none> 2bcdaa0e5f59 34 minutes ago 474MB
<none> <none> 4f1bd3b306d6 36 minutes ago 1.46GB
fabmedicalcnr.azurecr.io/content-api latest 343bcea66675 About an hour ago 125MB
fabmedicalcnr.azurecr.io/content-api v1 343bcea66675 About an hour ago 125MB
content-api latest 343bcea66675 About an hour ago 125MB
<none> <none> 7b6782c45805 About an hour ago 681MB
node alpine 3d9a0ea05233 2 days ago 112MB
node dubnium-alpine 9bfaa3adc4e1 3 days ago 82.7MB
node dubnium b15b7f33f1e6 4 days ago 910MB
mongo 4.0 b0df4f866657 9 days ago 428MB
node argon ef4b194d8fcf 2 years ago 653MB
adminfabmedical@fabmedical-cnr:~/Fabmedical/content-web$
```

14. Отправьте образы в свою учетную запись ACR с помощью следующей команды:
15. `docker image push [LOGINSERVER]/content-web:v1`  
`docker image push [LOGINSERVER]/content-api:v1`



16. Обновите один из репозиториях, чтобы теперь появились две версии изображения.



17. Выполните следующие команды, чтобы получить изображение из репозитория. Обратите внимание, что по умолчанию извлекаются изображения с тегами latest. Вы можете получить конкретную версию с помощью тега версии. Также обратите внимание, что, поскольку изображения уже существуют в агенте сборки, ничего не загружается.
18. 

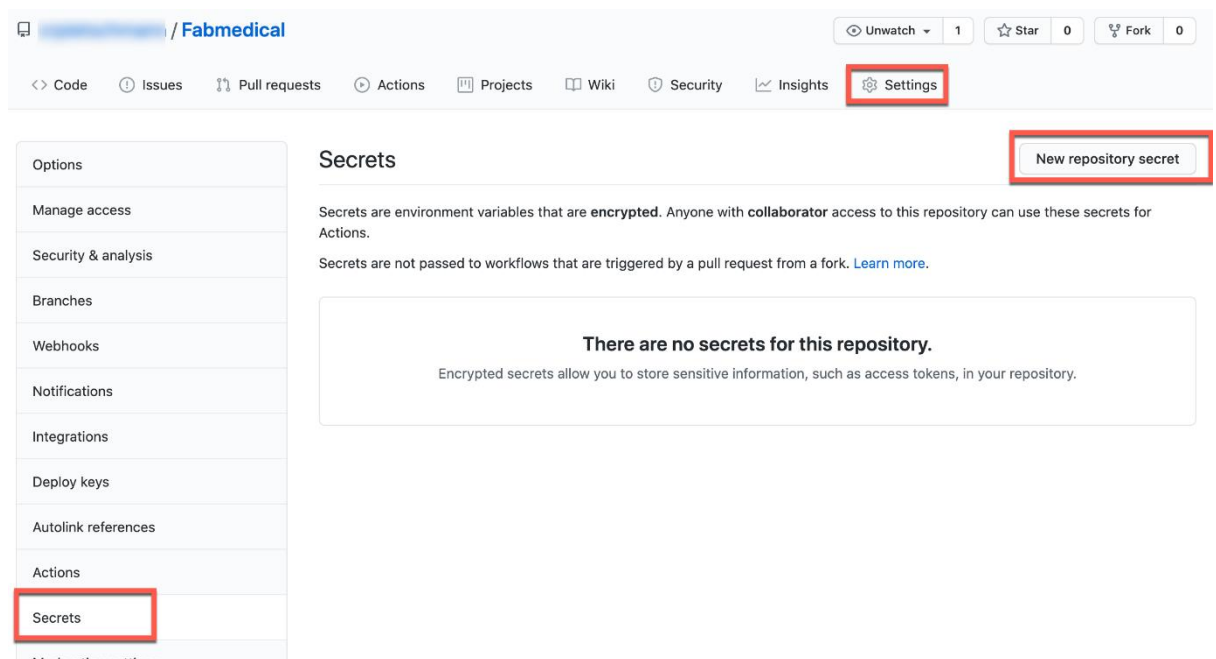
```
docker image pull [LOGINSERVER]/content-web
```

```
docker image pull [LOGINSERVER]/content-web:v1
```

## Задание 15: Настройка CI Pipeline для отправки образов

В этой задаче вы будете использовать YAML для определения рабочего процесса GitHub Actions, который создает ваш образ Docker и автоматически отправляет его в ваш экземпляр ACR.

1. В GitHub вернитесь на экран репозитория и выберите вкладку «Настройки»..
2. В меню слева выберите "Секреты".
3. Нажмите кнопку «Новый секрет репозитория».



4. В форме «Новый секрет» введите имя ACR\_USERNAME и в качестве значения вставьте имя пользователя реестра контейнеров Azure, которое было скопировано ранее. Выберите Добавить секрет.

## Secrets / New secret

Name

ACR\_USERNAME

Value

fabmedical

Add secret



5. Добавьте еще один секрет, введя имя ACR\_PASSWORD и вставив в качестве значения пароль реестра контейнеров Azure, который был скопирован ранее.

## Secrets

New secret

Secrets are environment variables that are **encrypted** and only exposed to selected actions. Anyone with **collaborator** access to this repository can use these secrets in a workflow.

Secrets are not passed to workflows that are triggered by a pull request from a fork. [Learn more](#).

|                                                                                                  |                        |                         |                         |
|--------------------------------------------------------------------------------------------------|------------------------|-------------------------|-------------------------|
|  ACR_PASSWORD | Updated now            | <button>Update</button> | <button>Remove</button> |
|  ACR_USERNAME | Updated 23 seconds ago | <button>Update</button> | <button>Remove</button> |

6. В сеансе Azure Cloud Shell, подключенном к виртуальной машине агента сборки, перейдите в каталог ~/Fabmedical:

```
cd ~/Fabmedical
```

7. Перед настройкой рабочих процессов GitHub Actions необходимо создать каталог .github/workflows, если он еще не существует. Сделайте это, выполнив следующие команды:
8. 

```
mkdir ~/Fabmedical/.github
```

```
mkdir ~/Fabmedical/.github/workflows
```

9. Перейдите в каталог `.github / workflows`:

```
cd ~/Fabmedical/.github/workflows
```

10. Затем создайте YAML-файл рабочего процесса.

11. `vi content-web.yml`

Добавьте следующее в качестве содержимого. Обязательно замените следующие заполнители:

- замените `[SHORT_SUFFIX]` на ваш короткий суффикс, например `SOL`.

```
name: content-web
```

```
This workflow is triggered on push to the 'content-web' directory
of the main branch of the repository
on:
```

```
 push:
 branches:
 - main
 paths:
 - 'content-web/**'
```

```
 # Configure workflow to also support triggering manually
 workflow_dispatch:
```

```
Environment variables are defined so that they can be used
throughout the job definitions.
```

```
env:
 imageRepository: 'content-web'
 resourceGroupName: 'fabmedical-[SHORT_SUFFIX]'
 containerRegistryName: 'fabmedical[SHORT_SUFFIX]'
 containerRegistry: 'fabmedical[SHORT_SUFFIX].azurecr.io'
 dockerfilePath: './content-web'
 tag: '${{ github.run_id }}'
```

```
Jobs define the actions that take place when code is pushed to the
main branch
```

```
jobs:
 build-and-publish-docker-image:
 name: Build and Push Docker Image
 runs-on: ubuntu-latest
 steps:
 # Checkout the repo
 - uses: actions/checkout@master

 - name: Set up Docker Buildx
 uses: docker/setup-buildx-action@v1

 - name: Login to ACR
 uses: docker/login-action@v1
 with:
 registry: ${ env.containerRegistry }
 username: ${ secrets.ACR_USERNAME }
 password: ${ secrets.ACR_PASSWORD }

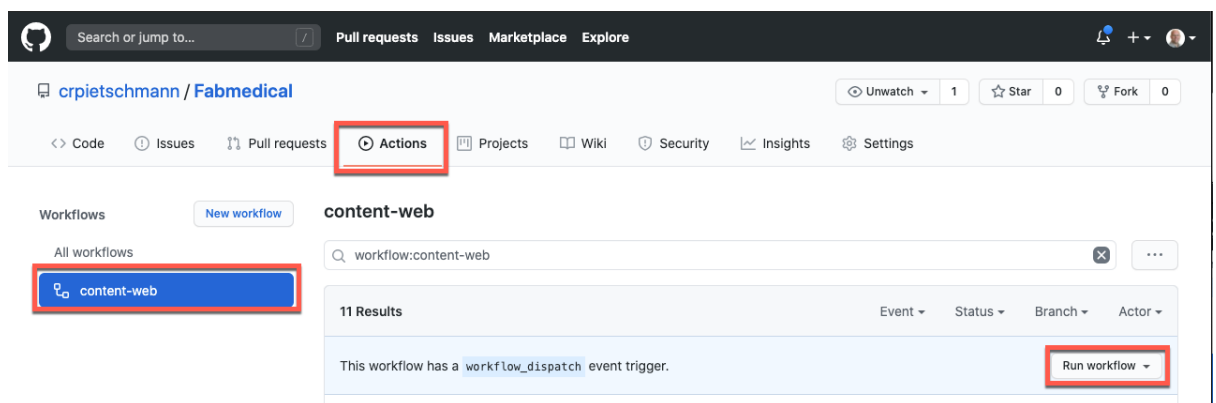
 - name: Build and push an image to container registry
 uses: docker/build-push-action@v2
 with:
```

```

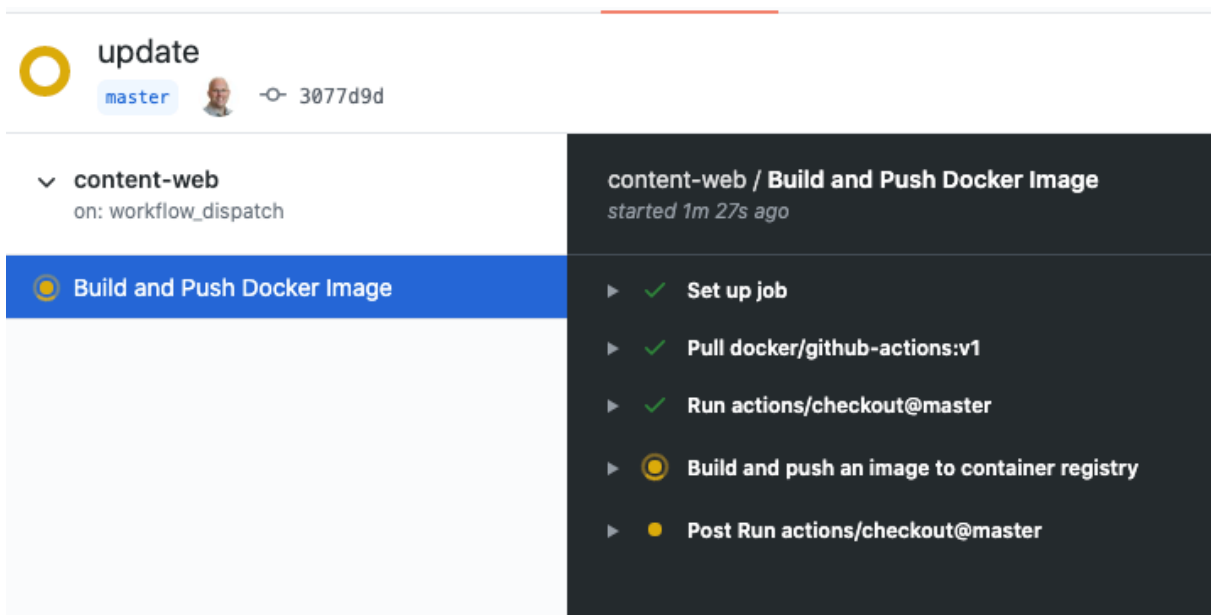
context: ${ env.dockerfilePath }}
file: "${ env.dockerfilePath }}/Dockerfile"
pull: true
push: true
tags: |
 ${ env.containerRegistry }}/${ env.imageRepository }}:${ env.tag }}
 ${ env.containerRegistry }}/${ env.imageRepository }}:latest

```

12. Сохраните файл и выйдите.
13. Сохраните конвейер YAML, затем выполните фиксацию и отправьте его в репозиторий Git:
14. `git add .`
15. `git commit -m "Added workflow YAML"`  
`git push`
16. В GitHub вернитесь на экран репозитория и выберите вкладку Действия.
17. На странице "Действия" выберите рабочий процесс контент-веб.
18. В рабочем процессе контент-веб выберите Запустить рабочий процесс и вручную запустите рабочий процесс для выполнения.



19. Через секунду в списке отобразится только что запущенное выполнение рабочего процесса. Выберите новое исполнение Content-Web, чтобы просмотреть его статус.
20. При выборе задания Build and Push Docker Image рабочего процесса будет отображаться состояние его выполнения.



21. Затем настройте рабочий процесс content-api. Этот репозиторий уже включает content-api.yml, расположенный в каталоге .github/workflows. Откройте файл .github/workflows/content-api.yml для редактирования.
22. Измените значения среды resourceGroupName и containerRegistry, заменив [SHORT\_SUFFIX] вашим собственным трехбуквенным суффиксом, чтобы он соответствовал имени вашего реестра контейнеров и группе ресурсов.

```
required: true
default: 'warning'

Environment variables are defined so that they can be used throughout the job definitions.
env:
 imageRepository: 'content-api'
 resourceGroupName: 'Fabmedical-[SHORT-SUFFIX]'
 containerRegistry: 'fabmedical[SHORT_SUFFIX].azurecr.io'
 dockerfilePath: './content-api'
 tag: '${{ github.run_id }}'

Jobs define the actions that take place when code is pushed to the master branch
jobs:
```

23. Зафиксируйте и отправьте изменения в репозиторий Git:
24. `git add .`
25. `git commit -m "Updated workflow YAML"`  
`git push`
26. Сохраните файл, затем перейдите к репозиториям в GitHub, выберите Действия, а затем вручную запустите рабочий процесс content-api.