

## Lab 2.

# Введение

Red Hat® OpenShift® Source-to-Image (S2I) — это платформа, упрощающая создание образов, принимающих исходный код приложения в качестве входных данных и создающих новый образ, запускающий собранное приложение в качестве выходных данных. Основным преимуществом использования S2I для создания воспроизводимых образов контейнеров является простота использования для разработчиков.

В этом руководстве вы узнаете, как использовать функцию OpenShift S2I. Мы будем использовать пример приложения `pyFlask` (популярная веб-инфраструктура на основе Python), размещенного на GitHub. Кроме того, вы узнаете, как настроить веб-перехватчик GitHub для уведомления OpenShift о новых событиях отправки/фиксации кода в GitHub, чтобы OpenShift выполнял автоматическую пересборку и повторное развертывание приложения с использованием последних изменений кода в вашем репозитории GitHub.

## Предварительные требования

Чтобы включить непрерывное развертывание с помощью функции OpenShift S2I, убедитесь, что выполнены следующие предварительные условия:

Доступ к кластеру OpenShift

Знакомство с основными операциями GitHub (клонирование/создание репозитория, создание и обновление файлов в репозитории и т. д.)

### Шаги

Создайте проект OpenShift для развертывания приложения.

Клонировать репозиторий GitHub

Создайте новое развертывание из репозитория GitHub.

Убедитесь, что приложение `pyFlask` успешно работает.

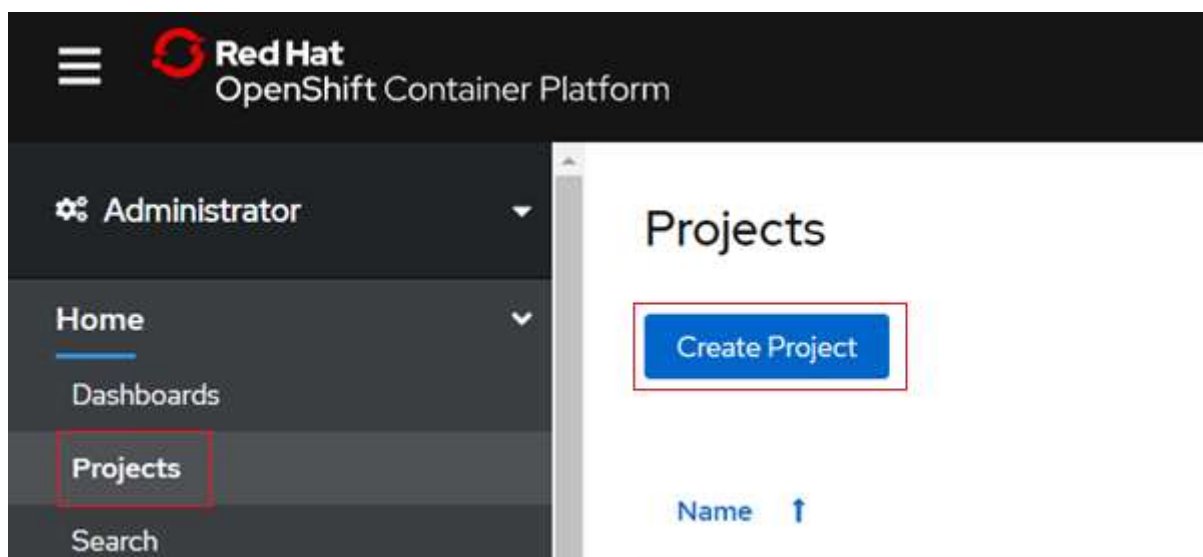
Настройте веб-перехватчик GitHub для уведомления OpenShift о событиях GitHub и настройте автоматическое развертывание.

Измените исходный код в GitHub и убедитесь, что модуль развертывается автоматически.

## Building Kubernetes Applications

**Упражнение 1. Создайте проект OpenShift для развертывания приложения.**

Перейдите на вкладку «Проекты» и на странице «Проекты» нажмите «Создать проект».



В диалоговом окне «Создать проект» введите сведения о проекте, такие как имя, отображаемое имя и описание, и нажмите «Создать».

### Create Project

Name \*

pyflask-demo

Display Name

pyflask s2i demo

Description

Cancel

Create

## Building Kubernetes Applications

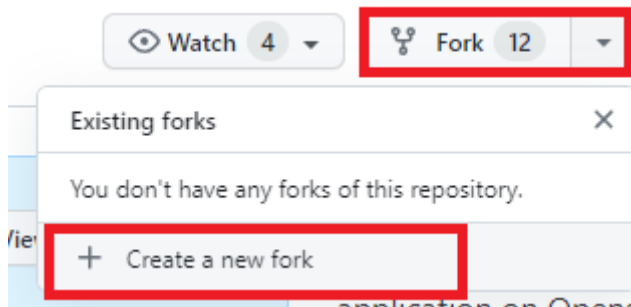
На странице Проект убедитесь, что проект создан. Можно использовать фильтры для быстрого поиска проекта.



### Упражнение 2. Клонирование репозитория GitHub

Приложение pyFlask с тремя конечными точками API (root, version и test) доступно по ссылке: <https://github.com/ocp-power-demos/s2i-pyflask-demo>

Нужно сделать форк этого репозитория, потому что мы будем вносить изменения в код GitHub, а для этого потребуются привилегии владельца.



### Упражнение 3. Создайте развертывание из репозитория GitHub

В графическом интерфейсе OpenShift переключитесь в перспективу разработчика и нажмите +Добавить , чтобы создать новое развертывание.

Затем нажмите « Из Git» . Убедитесь, что вы выбрали проект pyflask-demo, который мы создали ранее.

Building Kubernetes Applications

Developer

+Add

Topology

Monitoring

Search

Builds

Helm

Project

You are logged in as a temporary administrative user. Update the [cluster OAuth configuration](#)

Project: **pyflask-demo** Application: all applications

Add

Select a way to create an application, component or service from one of the options.

Quick Starts

Setting up Serverless

Installing the Pipelines Operator

Getting started with a sample

See all Quick Starts →

Samples

Create an application from a code sample

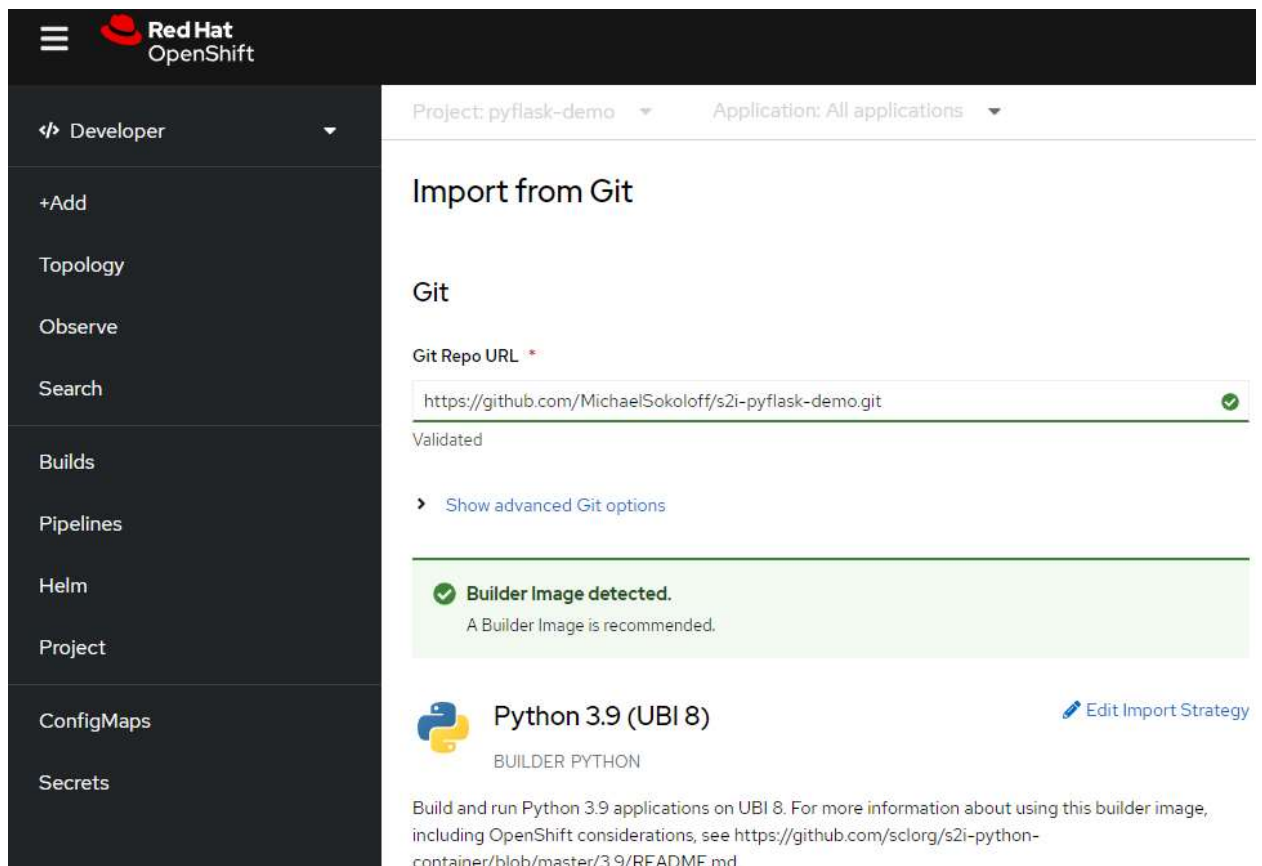
From Git

Import code from your Git repository to be built and deployed

## Building Kubernetes Applications

Заполните форму Import from Git необходимыми данными.

Укажите путь к репозиторию GitHub в поле Git Repo URL.



The screenshot shows the Red Hat OpenShift Developer console interface. On the left is a dark sidebar with navigation links: Developer, +Add, Topology, Observe, Search, Builds, Pipelines, Helm, Project, ConfigMaps, and Secrets. The main area is titled 'Import from Git' and shows the 'Git Repo URL' field with the value 'https://github.com/MichaelSokoloff/s2i-pyflask-demo.git'. Below this is a green notification box stating 'Builder Image detected. A Builder Image is recommended.' and a section for 'Python 3.9 (UBI 8)' with a link to 'Edit Import Strategy'.

Project: pyflask-demo Application: All applications

### Import from Git

#### Git

Git Repo URL \*

https://github.com/MichaelSokoloff/s2i-pyflask-demo.git

Validated

Show advanced Git options

Builder Image detected.  
A Builder Image is recommended.

Python 3.9 (UBI 8) [Edit Import Strategy](#)

BUILDER PYTHON

Build and run Python 3.9 applications on UBI 8. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/s2i-python-container/blob/master/3.9/README.md>.

Если появится сообщение «Невозможно обнаружить образ компоновщика», игнорируйте его.

## Building Kubernetes Applications

Введите имя, которое OpenShift будет использовать для приложения и ресурсов, которые будут созданы в ходе этого развертывания.

### General

#### Application name

s2i-pyflask

A unique name given to the application grouping to label your resources.

#### Name \*

s2i-pyflask

A unique name given to the component that will be used to name associated resources.

#### Resource type

Deployment

Resource type to generate. The default can be set in [User Preferences](#).

OpenShift по умолчанию поддерживает только безопасные (HTTPS) маршруты. Поэтому потребуется внести некоторые изменения в конфигурацию.

Нажмите «Маршрутизация» и в форме «Маршрутизация» установите флажок «Безопасный маршрут», чтобы включить TLS edge termination.

### Advanced Options

☒ Create a route to the application

Exposes your application at a public URL.

Click on the names to access advanced options for [routing](#), [Health Checks](#), [Build Configuration](#), [Deployment](#), [Scaling](#), [Resource Limits](#) and [Labels](#).

## Routing

### Hostname

Public hostname for the route. If not specified, a hostname is generated.

### Path

Path that the router watches to route traffic to the service.

### Target Port

Target port for traffic.

### Security

☒ Secure Route

Routes can be secured using several TLS termination types for serving certificates.

### TLS Termination

Edge ▼

### Insecure Traffic

None ▼


Policy for traffic on insecure schemes like HTTP.

Сохраните значения по умолчанию для остальных полей и нажмите **«Создать»**.

В представлении «Топология» выберите свое приложение. В представлении «Развертывание» дождитесь завершения сборки и перехода модуля в состояние **«Выполняется»**.

Building Kubernetes Applications

На этом шаге OpenShift создает образ контейнера (используя образ компоновщика Python и исходный код из репозитория GitHub), создает образ Docker, загружает образ во внутренний реестр образов OpenShift и создает модуль, используя этот образ Docker.



D s2i-pyflask

A s2i-pyflask

D s2i-pyflask

Actions

Health Checks

Container s2i-pyflask does not have health checks to ensure your application is running correctly. Add Health Checks

Details

Resources

Monitoring

Pods

P s2i-pyflask-6c8c69fc5c-7chwl

Running

View logs

Builds

BC s2i-pyflask

Start Build

Build #1 is complete (2 hours ago)

View logs

Project: pyflask-demo

Builds

Build details

B s2i-pyflask-1

Complete

Details

Metrics

YAML

Environment

Logs

Events

Log stream ended.

Search

77 lines

64

78c433309d32b7dc2d20b4d0790e2724d5ee86fad79bbf6b07c738ddeca05965

65

66

Pushing image image-registry.openshift-image-registry.svc:5000/pyflask-demo/s2i-pyflask:latest ...

67

Getting image source signatures

68

Copying blob sha256:da2461c9cf2ff635cde3fd1f6c1bdfacc7c9b3b62ed88464af670a1933a2d31e

69

Copying blob sha256:87b6121ef647e82c2efa8e6489d94c7668d88af38c138236592c6675acdf055a

70

Copying blob sha256:a6ed27b32c9775a3ea5c5d08d851dbfb051227dba59c0c70db3dbcccb64b1445

71

Copying blob sha256:6208c5a2e205726f3a2cd42a392c5e4f05256850d13197a711000c4021ede87b

72

Copying blob sha256:fe0156f1ce368b6a2b7e85dbab87b5a5ec42556f1d31d2b62eff09f03c095852

73

Copying config sha256:78c433309d32b7dc2d20b4d0790e2724d5ee86fad79bbf6b07c738ddeca05965

74

Writing manifest to image destination

75

Storing signatures

76

Successfully pushed image-registry.openshift-image-registry.svc:5000/pyflask-demo/s2i-pyflask@sha256:

77

Push successful



## Building Kubernetes Applications

### Упражнение 4. Проверка результата.

Убедитесь, что приложение pyFlask успешно работает. В представлении «Топология» щелкните приложение s2i-pyflask, прокрутите вниз страницу ресурсов и щелкните URL-адрес маршрутов.

Topology view showing the application s2i-pyflask. The application is highlighted with a red box. The right sidebar shows the Builds, Services, and Routes sections. The Route section shows the URL <https://s2i-pyflask-pyflask-demo.apps.test-ocp-ce4a.161.156.153.93.xip.io>, which is also highlighted with a red box.

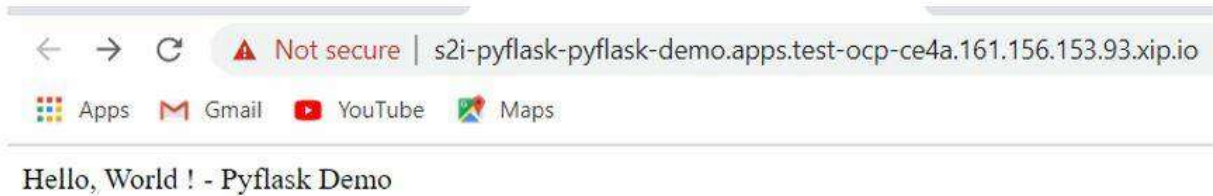
Browser address bar showing the URL <https://s2i-pyflask-pyflask-demo.apps-crc.testing>. The browser toolbar shows various icons and text.

# Hello, World ! - Pyflask Demo

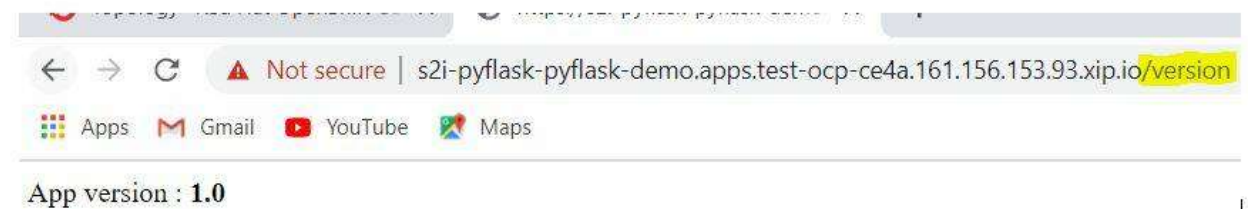
## Building Kubernetes Applications

URL-адрес открывается в вашем браузере и отображает «Hello World...». сообщение, как указано в исходном коде GitHub приложения pyFlask (см. функцию app.py's hello\_world(), которая сопоставляется с корневой конечной точкой).

**Примечание.** Если при первом открытии этого URL-адреса вы увидите предупреждение системы безопасности браузера, нажмите «**Дополнительно**» и продолжите открытие URL-адреса.



Аналогичным образом откройте конечные точки /version и /test и обратите внимание, что отображается соответствующее сообщение, закодированное в исходном коде приложения GitHub.



Приложение pyFlask, развернутое из источника GitHub, работает, вы успешно развернули модуль прямо из репозитория GitHub.

### Упражнение 5. Настройте веб-хук GitHub для уведомления OpenShift о событиях GitHub и настройте автоматическое развертывание.

Веб-хуки Github позволяют уведомлять внешние службы, когда происходят определенные события GitHub.

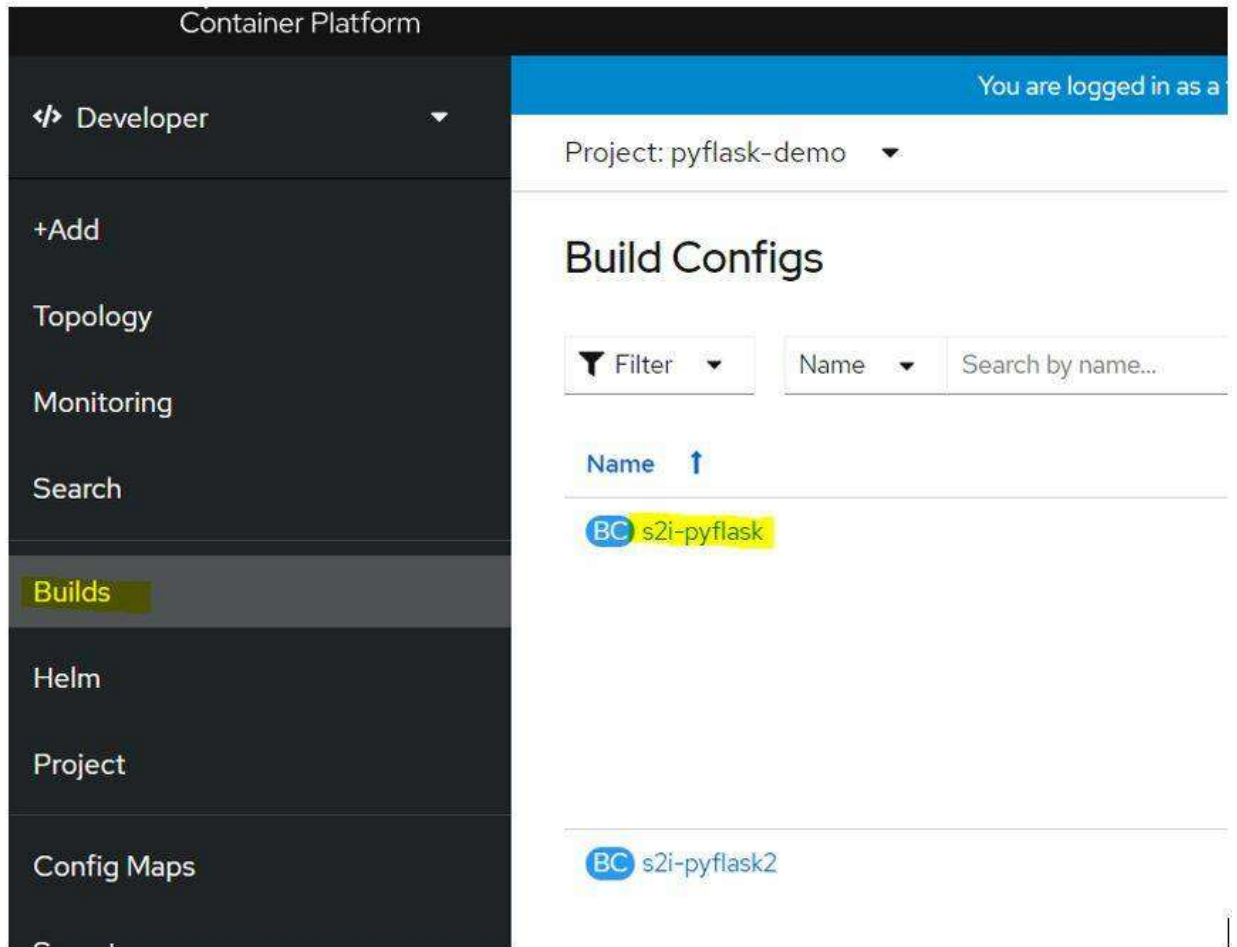
Непрерывное развертывание означает, что при обновления исходного кода в GitHub должны автоматически развертываться новые модули.

Чтобы это работало, нужно настроить наш репозиторий GitHub на использование веб-хука, который OpenShift предоставляет как часть атрибута BuildConfig.

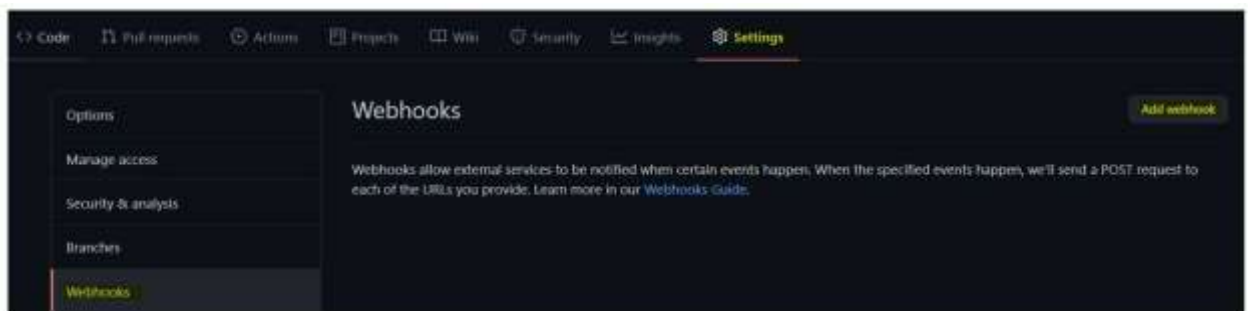
Это нагляднее можно сделать с помощью графического интерфейса.

## Building Kubernetes Applications

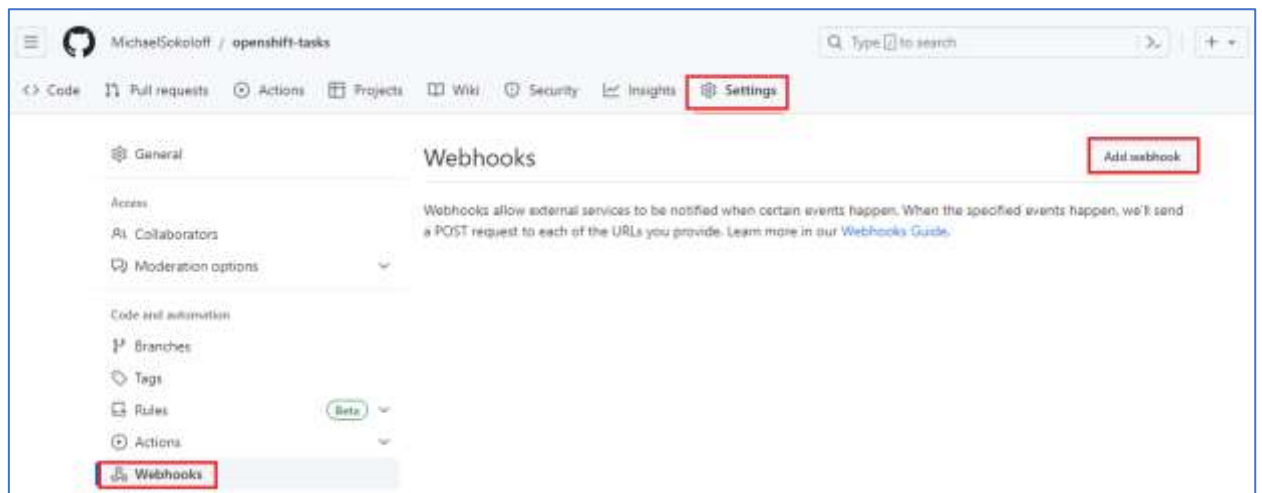
На странице BuildConfigs щелкните запись buildconfig и прокрутите вниз, чтобы найти универсальный веб-перехватчик, и нажмите «Копировать URL-адрес с секретом».



Перейдите в репозиторий GitHub и в разделе «Настройки» щелкните вкладку «Webhooks» и добавьте новый веб-перехватчик, используя скопированный URL-адрес из предыдущего шага.



## Building Kubernetes Applications



Введите скопированный URL-адрес в поле **URL-адрес полезной нагрузки** , выберите параметр **application/json** в списке **Тип контента** и нажмите **Отключить (не рекомендуется)** для

проверки SSL. Сохраните значения по умолчанию для остальных полей и нажмите **Add webhook**.

**Webhooks / Add webhook**

We'll send a POST request to the URL below with details of any subscribed (JSON, x-www-form-urlencoded, etc). More information can be found in [our](#)

---

**Payload URL \***

https://api.test-ocp-ce4a.161.156.153.93.xip.io:6443/apis/build.oper


**Content type**

application/json

**Secret**

---

**SSL verification**

 By default, we verify SSL certificates when delivering payloads.

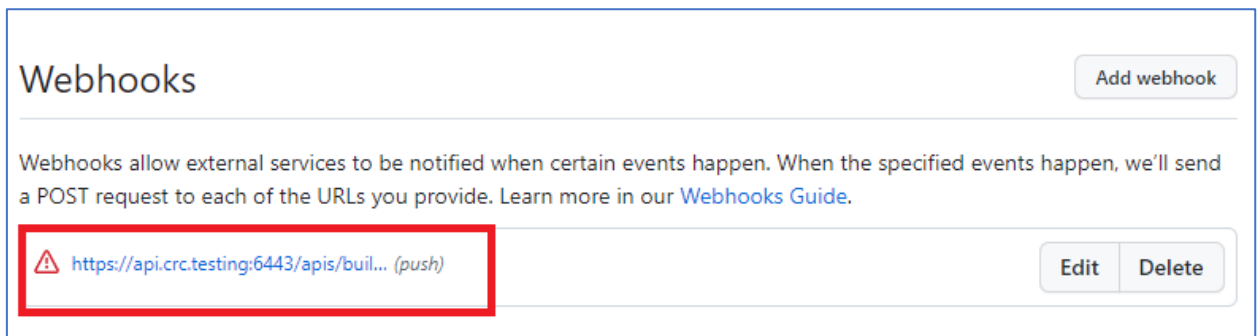
☐ Enable SSL verification ☒ **Disable (not recommended)**

---

**Which events would you like to trigger this webhook?**

☒ Just the push event.

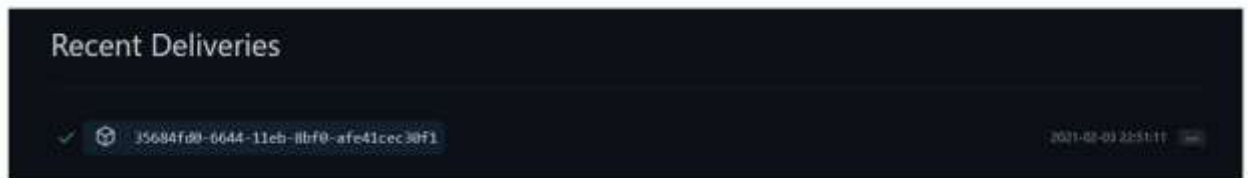
В случае успеха обратите внимание, что ваш недавно добавленный веб-хук отображается на странице веб-хуков.



Нажмите на недавно добавленный веб-хук, чтобы просмотреть подробности. GitHub обычно выполняет простой ping-тест как часть добавления нового веб-хука.

Перейдите на вкладку «Recent Deliveries»

Прокрутите вниз до раздела «Последние доставки» и обратите внимание, что отображается запись с префиксом «галочка» (указывающая, что проверка связи выполнена успешно). Щелкните запись, чтобы найти дополнительные сведения о вызове REST API и связанном ответе для проверки связи.



Успешный пинг-тест (ответ 200) будет означать, что GitHub может подключиться к вашему кластеру OpenShift.



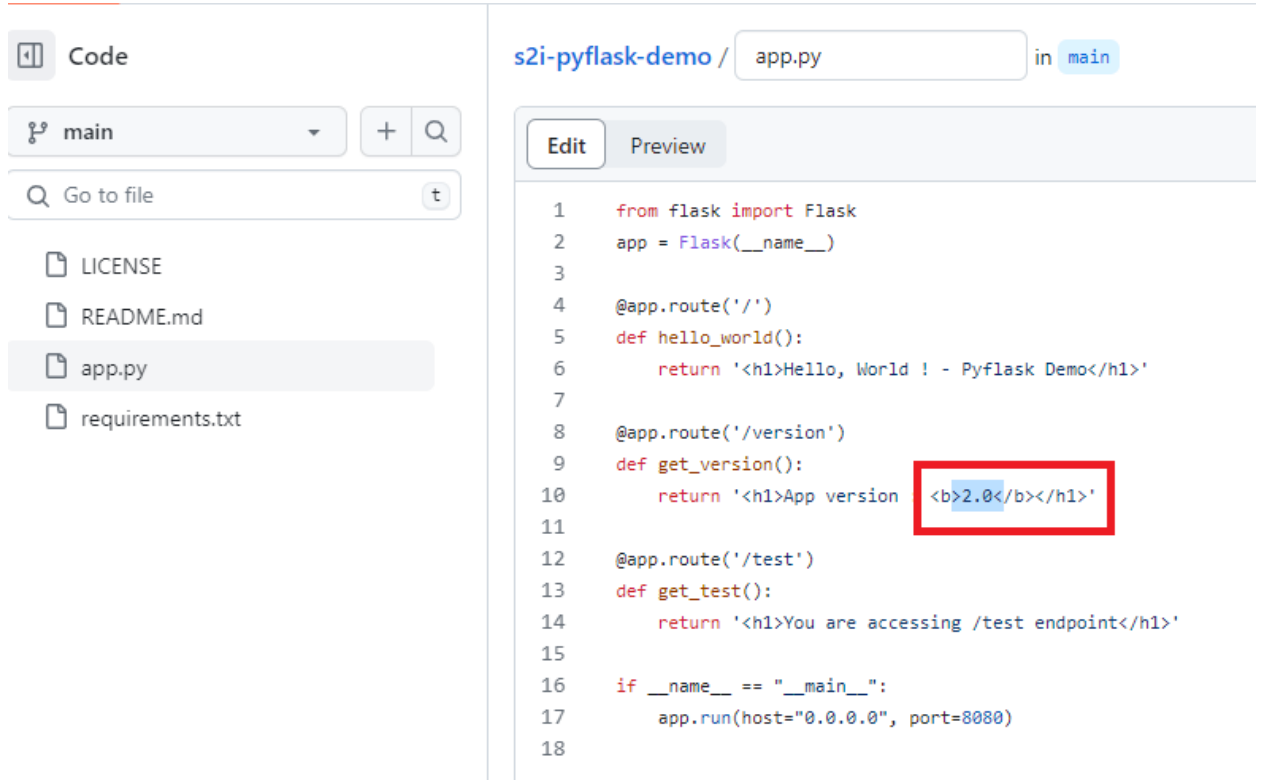
Теперь вы успешно настроили веб-хук GitHub.

## Building Kubernetes Applications

**Упражнение 6. Измените исходный код на GitHub и убедитесь, что модуль разворачивается автоматически.**

Внесите изменение в исходный код, изменив версию на 2.0 в app.py. Это должно вызвать событие push из GitHub в OpenShift и, в свою очередь, заставить OpenShift перестроить образ Docker и повторно развернуть наш модуль, используя только что созданный образ Docker.

Откройте файл app.py и в функции get\_version() измените версию с 1.0 на 2.0.



```
Code

main + 🔍

Go to file t

LICENSE
README.md
app.py
requirements.txt

s2i-pyflask-demo / app.py in main

Edit Preview

1 from flask import Flask
2 app = Flask(__name__)
3
4 @app.route('/')
5 def hello_world():
6     return '<h1>Hello, World ! - Pyflask Demo</h1>'
7
8 @app.route('/version')
9 def get_version():
10     return '<h1>App version <b>2.0</b></h1>'
11
12 @app.route('/test')
13 def get_test():
14     return '<h1>You are accessing /test endpoint</h1>'
15
16 if __name__ == "__main__":
17     app.run(host="0.0.0.0", port=8080)
18
```

Прокрутите вниз, добавьте сообщение о фиксации и нажмите «**Commit changes**».

Commit changes

Commit message

Update app.py

Extended description

Update version

Commit Email

msokoloff@outlook.com

☒ Commit directly to the main branch

☐ Create a new branch for this commit and start a pull request

[Learn more about pull requests](#)

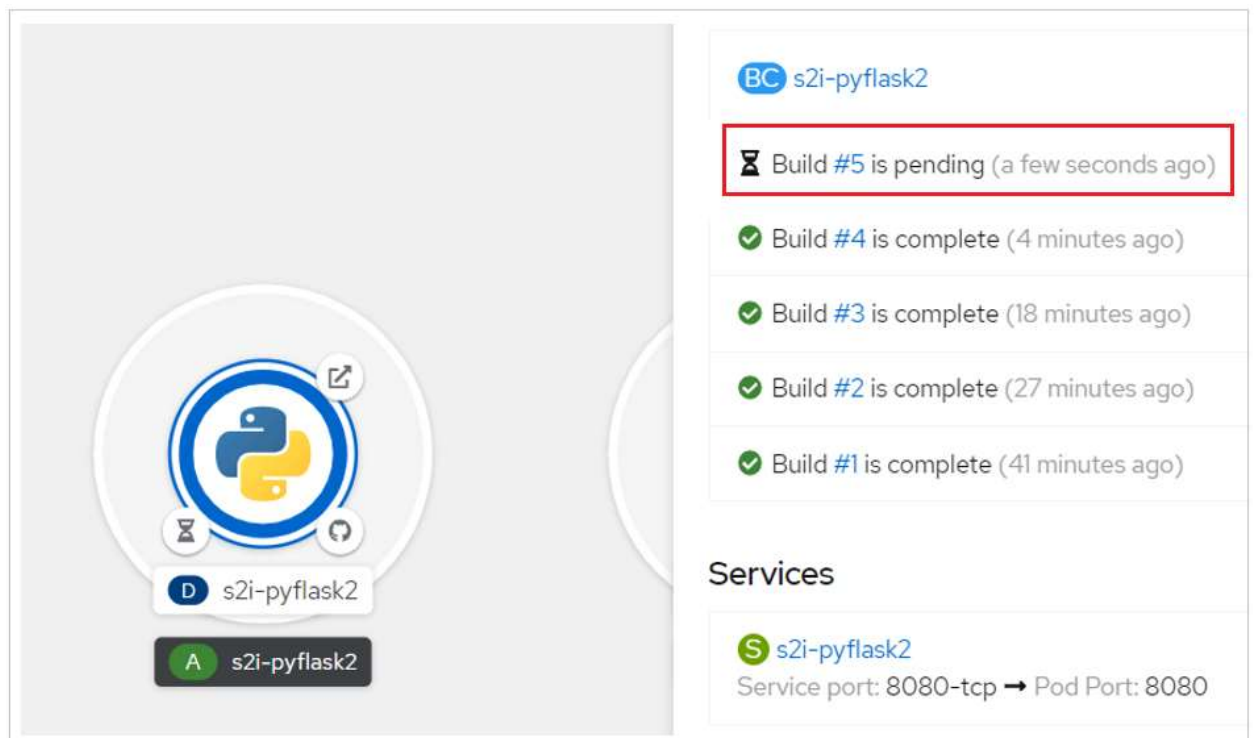
Cancel

Commit changes

Перейдите в графический интерфейс OpenShift, переключитесь на перспективу разработчика, щелкните приложение s2i-pyflask и наблюдайте, как новая сборка запускается автоматически.



## Building Kubernetes Applications



The screenshot displays the Kubernetes dashboard interface. On the left, a large circular icon features the Python logo, with a build configuration icon (BC) and a deployment icon (D) overlaid. Below this icon, the text 's2i-pyflask2' is visible. On the right, a panel titled 's2i-pyflask2' shows the build history. The first entry, 'Build #5 is pending (a few seconds ago)', is highlighted with a red border. Below it, four other builds are listed as complete, with their respective completion times. At the bottom of the right panel, a 'Services' section shows the 's2i-pyflask2' service with the port mapping 'Service port: 8080-tcp → Pod Port: 8080'.

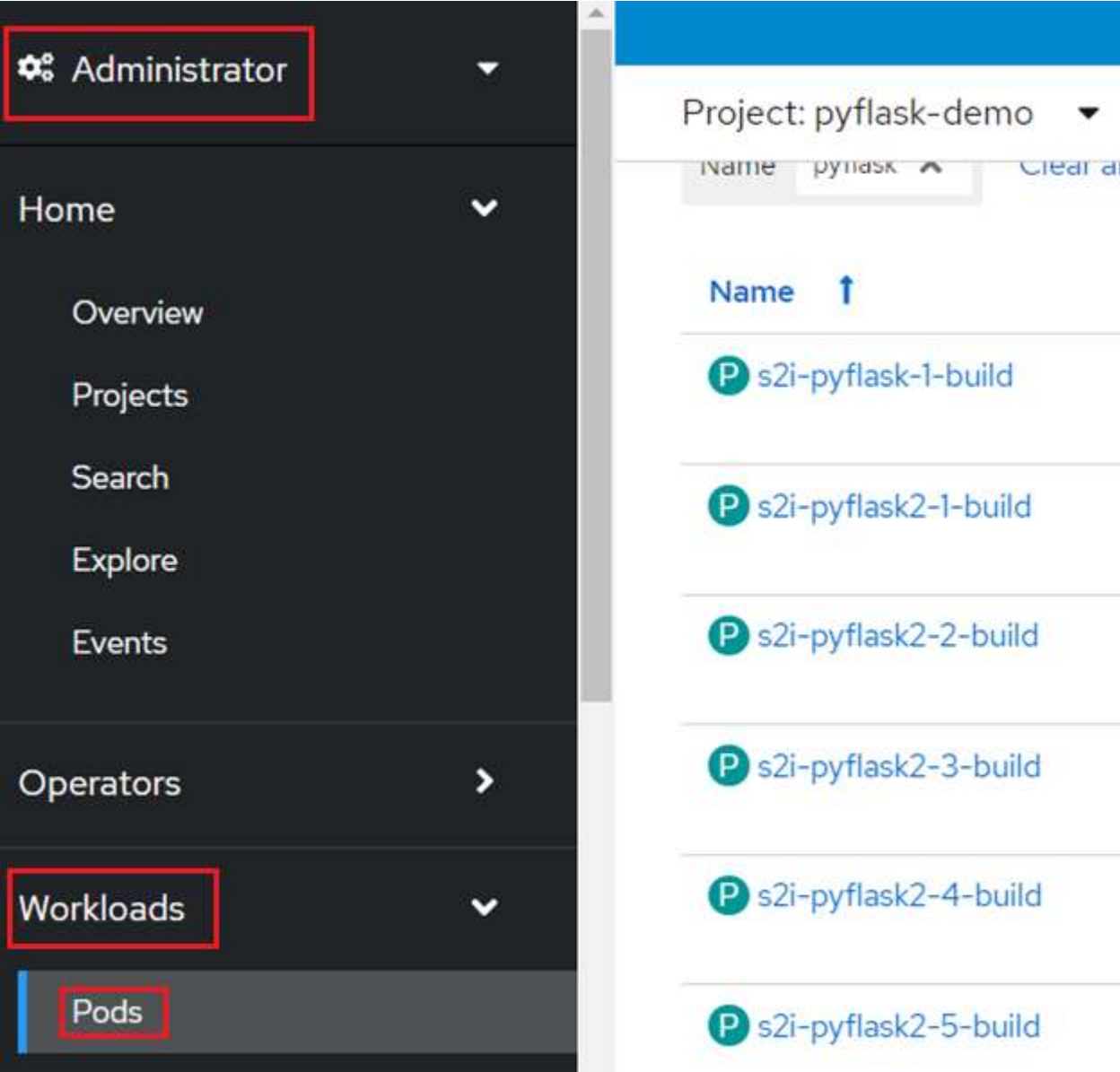
Build #	Status	Time
Build #5	pending	(a few seconds ago)
Build #4	complete	(4 minutes ago)
Build #3	complete	(18 minutes ago)
Build #2	complete	(27 minutes ago)
Build #1	complete	(41 minutes ago)

**Services**

Service	Port Mapping
s2i-pyflask2	Service port: 8080-tcp → Pod Port: 8080

После завершения сборки новый Pod будет автоматически развернут с использованием нового, созданного из этой сборки образа Docker.

Переключитесь на перспективу «Администратор», нажмите «Workloads» и нажмите «Pods».

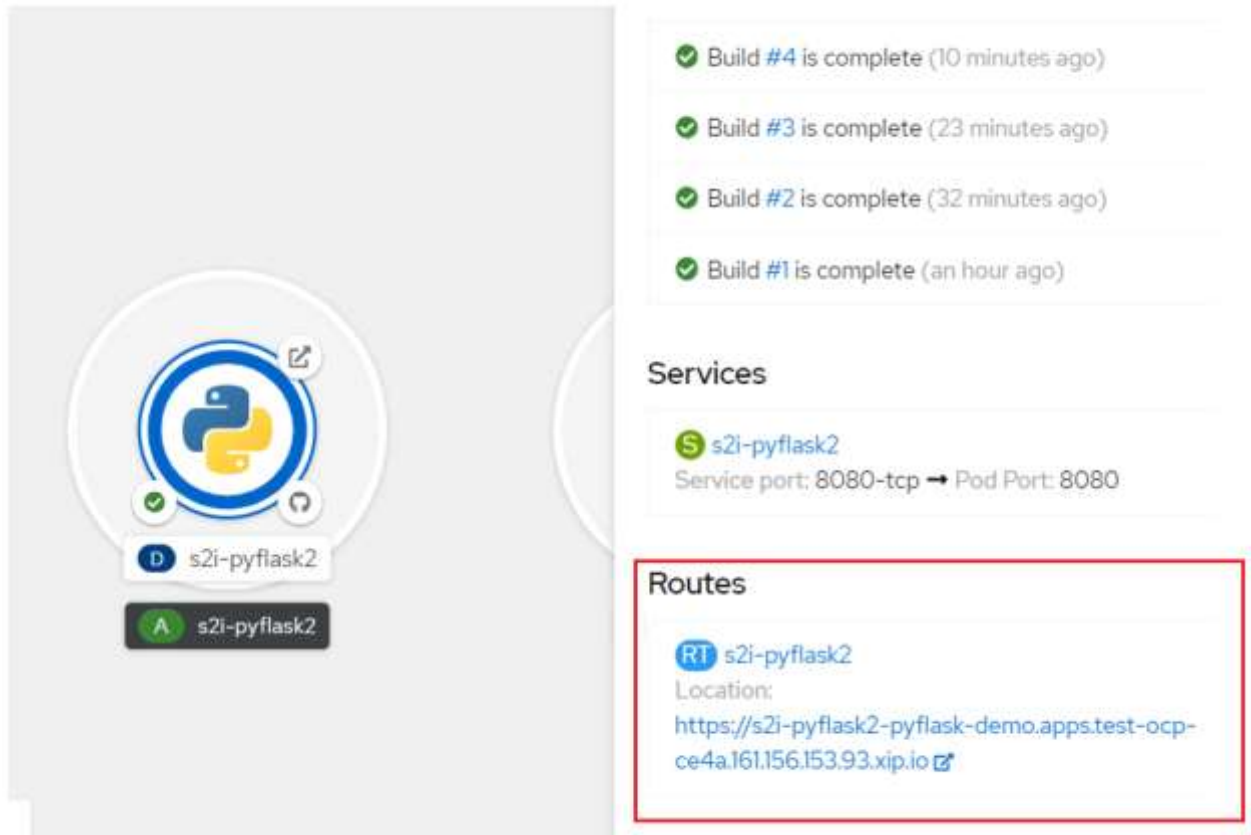


Обратите внимание, что старый pod завершается и создается новый pod.

 s2i-pyflask2-64f5c96f95-tqp9m	 Running	1/1	0	 RS s2i-pyflask2-64f5c96f95	96.9 MiB	-	 less than a minute ago
 s2i-pyflask2-776c71457c-2hwrb	 Terminating	0/1	0	 RS s2i-pyflask2-776c71457c	96.5 MiB	0.000 cores	 6 minutes ago

## Building Kubernetes Applications

Вернитесь в перспективу «Разработчик», нажмите **«Топология»**, выберите приложение **s2i-pyflask**, чтобы открыть окно ресурсов, и прокрутите вниз до раздела **«Маршруты»**.



Build #4 is complete (10 minutes ago)

Build #3 is complete (23 minutes ago)

Build #2 is complete (32 minutes ago)

Build #1 is complete (an hour ago)

**Services**

s2i-pyflask2  
Service port: 8080-tcp → Pod Port: 8080

**Routes**

RT s2i-pyflask2  
Location:  
<https://s2i-pyflask2-pyflask-demo.apps.test-ocp-ce4a.161.156.153.93.xip.io>

Щелкните URL-адрес маршрута и получите доступ к конечной точке `/version`.

Обратите внимание, что URL-адрес маршрутизатора не изменился, когда мы перешли с версии 1 на версию 2 нашего приложения. Если у вас уже открыт URL-адрес конечной точки `/version` (как часть предыдущего шага), вы можете просто обновить его.

Обратите внимание, что в контейнере действительно работает более новая версия приложения:



Not secure | s2i-pyflask2-pyflask-demo.apps.test-ocp-ce4a.161.156.153.93.xip.io/version

App version : 2.0

## Building Kubernetes Applications

Если вы перейдете к представлению веб-хуков GitHub, вы увидите новую запись в разделе «Последние доставки», которая соответствует недавнему уведомлению, доставленному GitHub в OpenShift в ответ на операцию изменения кода (push/commit) (изменение версии), которую мы выполнили.



Щелкните недавнюю доставку и обратите внимание на сообщение Response 200, указывающее на успешное восстановление и повторное развертывание модуля в нашем кластере OpenShift.

