Министерство Науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»



Инженерная школа информационных технологий и робототехники

Отделение информационных технологий

Направления подготовки: 09.04.01 «Информатика и вычислительная техника»

**Лабораторная работа №2**

**Проектирование и разработка приложений с использованием микросервисной архитектуры и бессерверных технологий**

по дисциплине:

**Облачные технологии**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исполнитель:** |  | | | | |
| студент группы | 8ВМ22 |  | Ямкин Н.Н. |  |  |
|  |  |  |  |  |  |
| **Руководитель:** | . | | | | |
| к.т.н доцент ОИТ, ИШИТР |  |  | Ботыгин И.А. |  |  |
|  |  |  |  |  |  |

Томск – 2023

**Теоретическая часть**

Бессерверные вычисления — это метод предоставления серверных услуг на основе фактического использования сервисов. Бессерверный провайдер позволяет пользователям писать и развёртывать код, не беспокоясь о базовой инфраструктуре. Компания, которая получает бэкенд-услуги от бессерверного поставщика, платит по факту за используемые ресурсы и не должна резервировать и оплачивать фиксированную пропускную способность или количество серверов, поскольку услуга автоматически масштабируется. Конечно, для предоставления клиенту бессерверных вычислений используются физические серверы, но разработчикам нет необходимости думать об их конфигурации, производительности, ядрах, памяти и прочем.

На заре интернета любой, кто хотел создать веб-приложение, должен был физически владеть оборудованием, необходимым для запуска сервера. Это было дорого и неудобно, потому что оборудование требовало много места.

Затем пришли облачные вычисления, когда нужное количество серверов или часть серверного пространства можно было арендовать в облаке. Разработчики и компании, которые арендуют эти ресурсы, обычно приобретают мощности с некоторым запасом, чтобы гарантировать, что всплеск трафика или активности пользователей не превысит их месячные лимиты в облачной инфраструктуре и не выведет их приложение из строя. Это означает, что некоторая часть оплачиваемого серверного пространства простаивает и не используется. Для решения этой проблемы облачные провайдеры предлагают модели автоматического масштабирования, но даже при такой модели выделения ресурсов нежелательный всплеск активности, вроде DDoS-атаки, может оказаться очень дорогостоящим.

Бессерверные вычисления позволяют разработчикам приобретать бэкенд-сервисы с оплатой по мере использования, что означает, что разработчикам нужно платить только за те услуги, которые они используют. Это похоже на переход с тарифного плана мобильного оператора с фиксированным ежемесячным лимитом на тариф, где плата взимается только за каждый фактически использованный байт данных.

Термин «бессерверный» несколько вводит в заблуждение, поскольку все же существуют серверы, предоставляющие эти внутренние сервисы. Но все проблемы, связанные с серверным пространством и инфраструктурой, решаются поставщиком. Бессерверный режим означает, что разработчики могут выполнять свою работу, вообще не беспокоясь о серверах, а также имеют следующие преимущества:

* Снижение затрат — бессерверные вычисления, как правило, выгодны, поскольку у многих крупных провайдеров облачных серверных услуг пользователь платит за неиспользуемое пространство или время простоя процессора.
* Упрощённая масштабируемость — разработчикам, использующим бессерверную архитектуру, не нужно беспокоиться о политиках для масштабирования своего кода. Бессерверный поставщик выполняет все масштабирование по запросу.
* Упрощённый внутренний код — с помощью FaaS разработчики могут создавать простые функции, которые независимо друг от друга выполняют одну задачу, например, выполнение вызова API.
* Более быстрый оборот — бессерверная архитектура может значительно сократить время выхода на рынок. Вместо того, чтобы требовать сложного процесса развёртывания для исправления ошибок и новых функций, разработчики могут добавлять и изменять код по частям.

В сравнении с другими моделями облачного сервиса.

Есть ещё пара технологий, которые часто путают с бессерверными вычислениями — это Backend-as-a-Service и Platform-as-a-Service. Хотя у них есть общие черты, эти модели не обязательно удовлетворяют требованиям бессерверности.

Backend-as-a-service (BaaS) — это модель обслуживания, в которой поставщик облачных услуг предлагает серверные службы (например, хранение данных), чтобы разработчики могли сосредоточиться на написании кода фронтенда. Но хотя бессерверные приложения управляются событиями и работают на периферии, приложения BaaS могут не соответствовать ни одному из этих требований.

Платформа как услуга (PaaS) — это модель, в которой разработчики, по сути, арендуют все необходимые инструменты для разработки и развёртывания приложений у облачного провайдера, включая такие вещи, как операционные системы и промежуточное ПО. Однако приложения PaaS не так легко масштабируются, как бессерверные приложения. PaaS также не обязательно работает на периферии и часто имеет заметную задержку запуска, которой нет в бессерверных приложениях.

Инфраструктура как услуга (IaaS) — это общий термин для поставщиков облачных услуг, размещающих инфраструктуру от имени своих клиентов. Поставщики IaaS могут предлагать бессерверные функции, но эти термины не являются синонимами.

**Практическая часть**

Cloud Functions позволяет запускать ваш код в обслуживаемой среде в виде [функции](https://cloud.yandex.ru/docs/functions/concepts/function). Сервис выделяет ресурсы, необходимые для выполнения функции. При увеличении количества вызовов функции происходит автоматическое масштабирование — сервис создает дополнительные экземпляры вашей функции.

Функция запускается в [среде выполнения](https://cloud.yandex.ru/docs/functions/concepts/runtime/), которая соответствует ее языку программирования. Среды выполнения доступны для Node.js, Python, Go, PHP, Bash, Java, R и C#.

Создадим функцию в облаке Yandex Cloud Functions и обратимся к ней через HTTP-запрос.

Заходим на главную страницу, подключаемся к платежному аккаунту и выбираем Сервисы => Cloud Functions => Создать функцию.

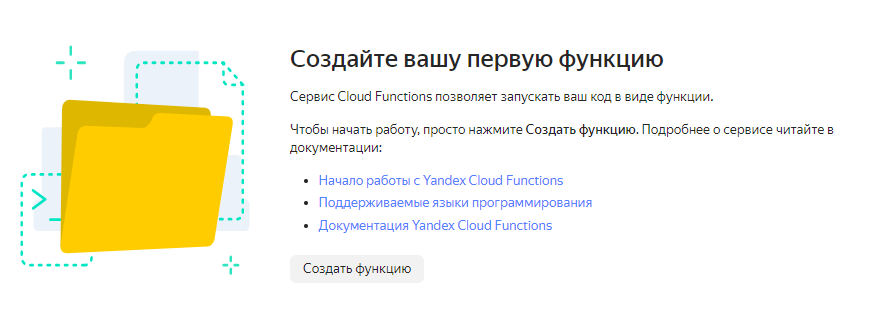


Рисунок 1 – Создание функции

Функция позволяет вам размещать код в Yandex Cloud, который можно запускать по запросу или триггеру.

Сразу после создания функция содержит только метаинформацию о себе: имя, описание, уникальный идентификатор и т. д. Для начала работы с функцией необходимо [создать версию функции](https://cloud.yandex.ru/docs/functions/operations/function/version-manage). Выполнить функцию можно с помощью HTTPS API или CLI.

Вводим имя функции и её описание (необязательно).

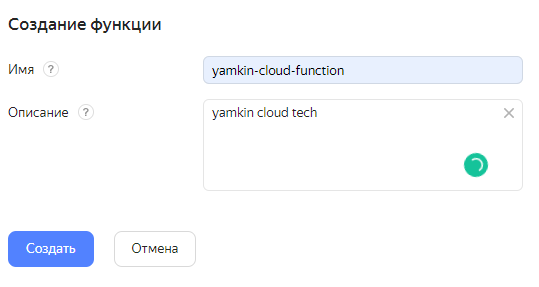


Рисунок 2 – Имя функции и её описание

Далее, в редакторе выбираем язык программирования, который будет использоваться при написании программы.

Выберем Python 3.11 и добавим файлы с примерами кода.

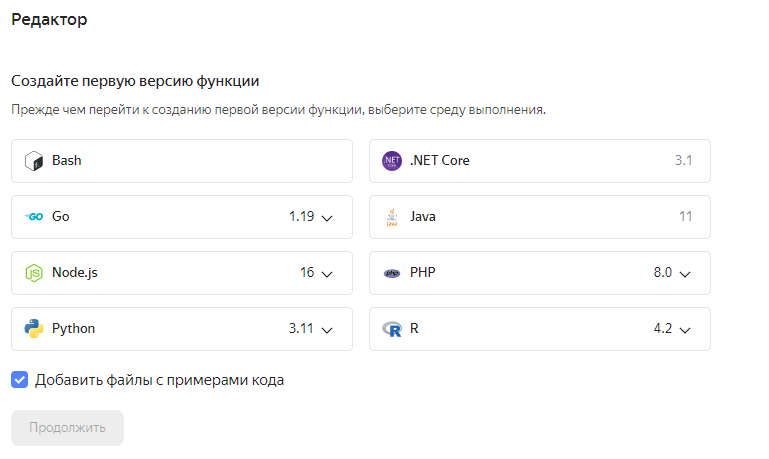


Рисунок 3 – Редактор

После этого открывается редактор, в котором уже написана исходная функция-обработчик запроса handler.

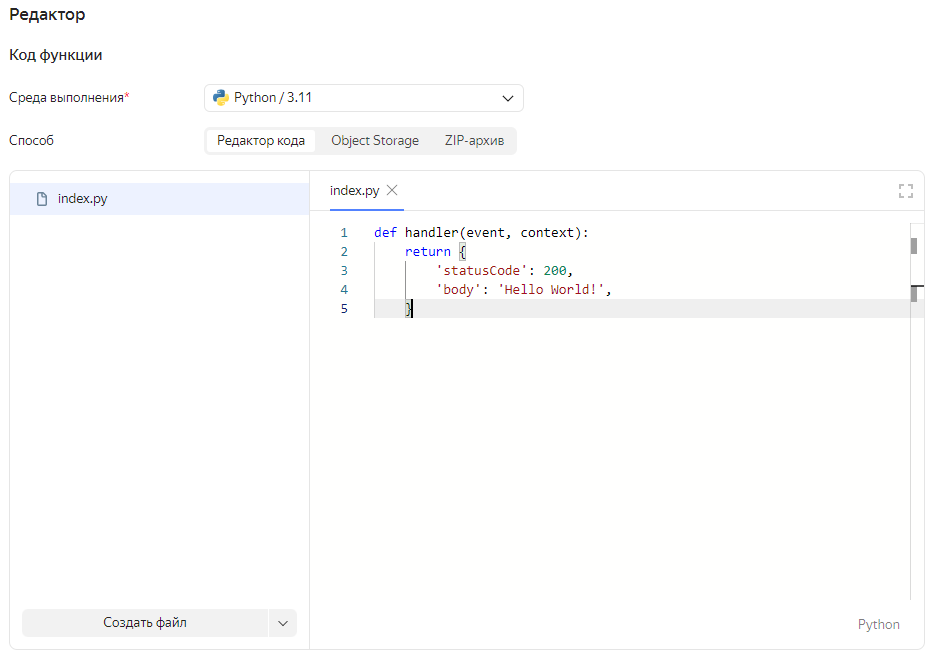


Рисунок 4 – Исходная функция-обработчик handler

Если функция вызывается для обработки HTTPS-запроса, то она получает первым аргументом данные о запросе в формате JSON: название HTTP-метода, заголовки, аргументы и другие параметры запроса.

JSON-структура запроса:

{

"httpMethod": "<название HTTP-метода>",

"headers": <словарь со строковыми значениями HTTP-заголовков>,

"multiValueHeaders": <словарь со списками значений HTTP-заголовков>,

"queryStringParameters": <словарь queryString-параметров>,

"multiValueQueryStringParameters": <словарь списков значений queryString-параметров>,

"requestContext": <словарь с контекстом запроса>,

"body": "<содержимое запроса>",

"isBase64Encoded": <true или false>

}

Результат, который возвращает функция, также должен представлять из себя JSON-документ, содержащий код ответа HTTP, заголовки ответа и содержимое ответа. Cloud Functions автоматически обработает этот JSON, и пользователь получит данные в виде стандартного HTTPS-ответа.

JSON-структура ответа:

{

"statusCode": <HTTP код ответа>,

"headers": <словарь со строковыми значениями HTTP-заголовков>,

"multiValueHeaders": <словарь со списками значений HTTP-заголовков>,

"body": "<содержимое ответа>",

"isBase64Encoded": <true или false>

}

В области справа (см. рисунок 4) можем написать собственный код или же загрузить его.

В области слева (см. рисунок 4) находится дерево файлов, в котором мы можем создавать, удалять и редактировать любые файлы.

На экране редактора снизу необходимо задать точку входа – имя функции, которая будет вызываться в качестве обработчика (указывается в формате <имя файла с функцией>.<имя обработчика>, например, index.myFunction), таймаут ответа и объём памяти, выделяемой для выполнения функции. Остальные параметры выставляются по желанию.

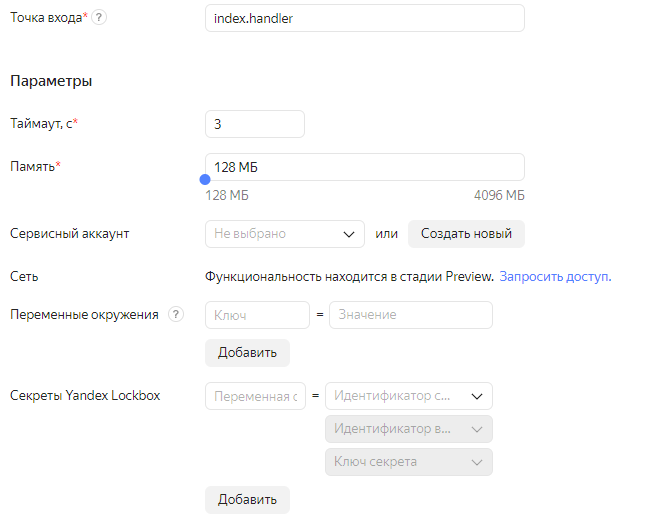


Рисунок 5 – Настройки вызова функции

Запустим исходную функцию handler. Нажимаем «Создать версию» и попадаем на экран Обзор. Далее переходим по ссылке для вызова функции.

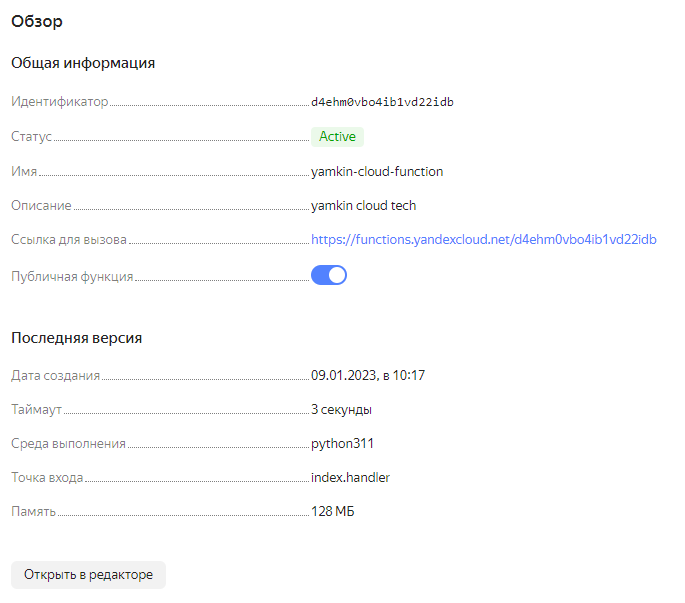


Рисунок 6 – Обзор функции

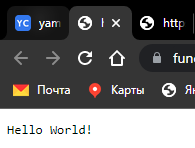


Рисунок 7 – Результат работы функции handler

Среда выполнения автоматически загружает ваш код и вызывает указанный вами [обработчик запросов](https://cloud.yandex.ru/docs/functions/lang/python/handler). В качестве аргументов он получает входящий запрос и [контекст вызова](https://cloud.yandex.ru/docs/functions/lang/python/context), который содержит дополнительную информацию о параметрах функции.

В Yandex Cloud Functions доступны все ранее созданные версии функции. Благодаря этому, пользователь может отредактировать любую функцию.

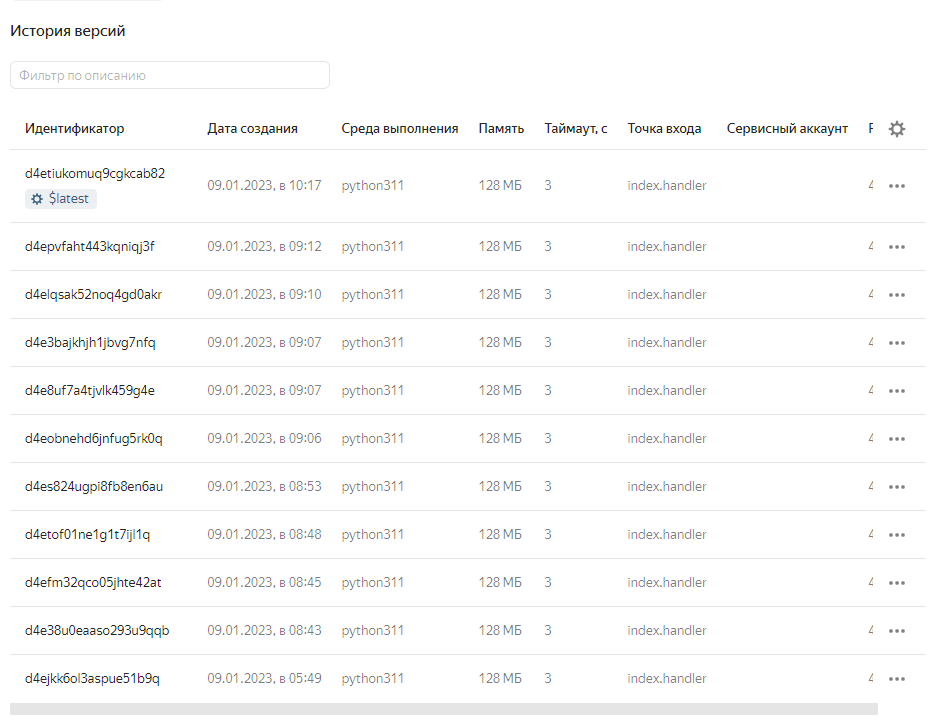


Рисунок 7 – История версий

Теперь добавим немного смысловой нагрузки в функцию. Напишем функцию расчета числа Фибоначчи по его номеру и вызовем её из функции-обработчика запросов handler по HTTP.

|  |
| --- |
| import json  # 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987...  # 1  2  3  4  5  6  7   8   9  10  11  12   13   14   15   16   17...  def Fibbonacci(n):    if n == 1 or n == 2:      return 1    return Fibbonacci(n-1) + Fibbonacci(n-2)  def handler(event, context):      fib = Fibbonacci(15)      return {          'statusCode': 200,          'body': fib,      } |

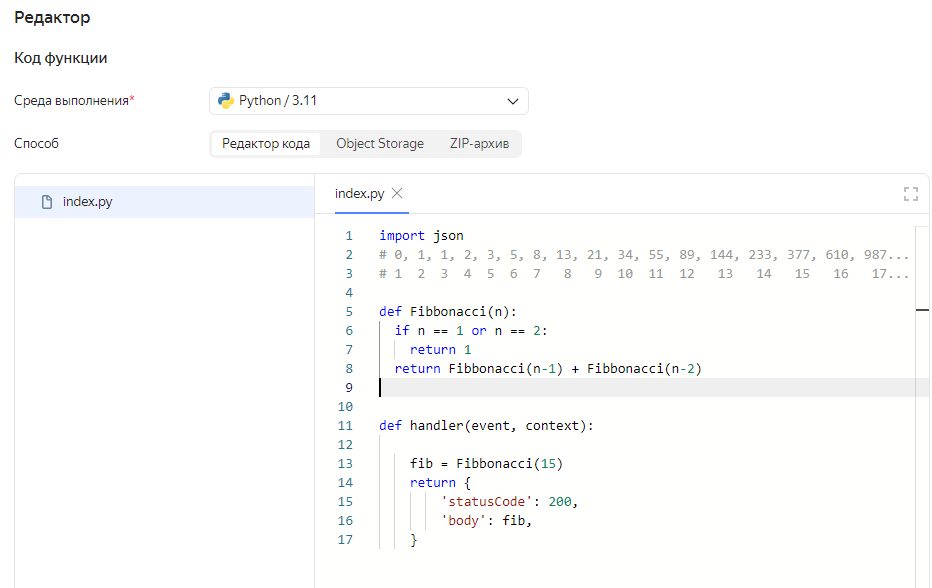


Рисунок 8 – Редактор кода

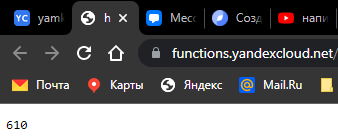


Рисунок 9 – Результат работы функции

**Заключение**

В ходе лабораторной работы была проведена ознакомительная работа с бессерверными вычислениями.

В качестве практической реализации была создана функция в облачной платформе Yandex Cloud Functions, которая возвращала по номеру само число Фибоначчи.

**Список использованных источников**

1. Yandex Cloud Functions. Пошаговые инструкции // Yandex Cloud Functions.URL:<https://cloud.yandex.ru/docs/functions/operations/> (дата обращения: 09.01.2023).