

Dokumentation: Modul 346

„Fizz-Buzz“

Inhaltsverzeichnis

1. Einleitung
2. Ziel der Arbeit
3. Technologien und Tools
4. Implementierung der FizzBuzz-Funktion
 - 4.1. C#
 - 4.2. Java
 - 4.3. JavaScript
 - 4.4. Python
5. Deployment in Azure
6. GitHub
7. Fazit

1. Einleitung

Die Cloud-Technologie hat sich in den letzten Jahren zu einem der wichtigsten Werkzeuge der Softwareentwicklung entwickelt. Azure Functions, eine serverlose Plattform von Microsoft, erlaubt mir, ereignisgesteuerte Anwendungen zu erstellen. In dieser Arbeit wird das klassische Problem "FizzBuzz" mithilfe von Azure Functions in vier Programmiersprachen implementiert: C#, Java, JavaScript und Python. Ziel ist es, die Logik zu entwickeln und zu dokumentieren.

2. Ziel der Arbeit

Ziel dieser Arbeit ist die Implementierung einer Azure Function, die die bekannte FizzBuzz-Logik umsetzt. Der Fokus liegt auf:

1. Der Implementierung der FizzBuzz-Logik in C#, Java, JavaScript und Python.
Bsp.: Python. (Simple Anzeige in Azure)

Name	Trigger	Status
pyfizzbuzz	HTTP	✓ Aktiviert

2. Der lokalen Entwicklung und dem Testen der Funktion.
 3. Der Bereitstellung (Deployment) der Funktion auf Microsoft Azure.
 4. Der Dokumentation und Analyse der Ergebnisse.
-



3. Technologien und Tools

3.1. Programmiersprachen

- **C#**: Die Hauptsprache der Azure Functions-Umgebung und gut geeignet für serverlose Anwendungen.

 csfizzbuzz.cs 



- **Java**: Eine universelle, plattformübergreifende Sprache, die in vielen Unternehmen verwendet wird.

 javafizzbuzz.java 

- **JavaScript**: Die Sprache des Webs, die auch für Backend-Entwicklung mit Node.js populär ist.

 jsfizzbuzz.js 

- **Python**: Eine flexible Sprache, bekannt für einfache Syntax und umfangreiche Bibliotheken.

 fizzbuzz.py 

3.2. Tools

- **Azure Functions Core Tools:** Ermöglicht die lokale Entwicklung und das Testen von Azure Functions.
- **Visual Studio Code/IDE:** Für die Entwicklung in den verschiedenen Sprachen.
- **Azure CLI:** Zum Verwalten der Azure-Ressourcen.
- **Postman/Curl:** Zum Testen der HTTP-Endpoints.

```
function_app.py > ...
1 import azure.functions as func
2 import logging
3
4 app = func.FunctionApp(http_auth_level=func.AuthLevel.Anonymous)
5
6 @app.route(route="pyfizzbuzz")
7 def pyfizzbuzz(req: func.HttpRequest) -> func.HttpResponse:
8     logging.info('Python HTTP trigger function processed a request.')
9
10     name = req.params.get('name')
11     if not name:
12         try:
13             req_body = req.get_json()
14         except ValueError:
15             pass
16         else:
17             name = req_body.get('name')
18
19     if name:
20         return func.HttpResponse(f"Hello, {name}. This HTTP triggered function executed successfully.", status_code=200)
21     else:
22         return func.HttpResponse("This HTTP triggered function executed successfully.", status_code=200)
```

Azure Functions: HTTP Trigger in Python

HTTP Trigger

The HTTP trigger lets you invoke a function with an HTTP request. You can use an HTTP trigger to build serverless APIs and respond to webhooks.

Using the Template

Following is an example code snippet for HTTP Trigger using the [Python programming model V2](#) (currently in Preview).

```
import azure.functions as func
import logging
```

```
{ host.json > ...
1 {
2     "version": "2.0",
3     "logging": {
4         "applicationInsights": {
5             "samplingSettings": {
6                 "isEnabled": true,
7                 "excludedTypes": "Request"
8             }
9         }
10    },
11    "extensionBundle": {
12        "id": "Microsoft.Azure.Functions.ExtensionBundle",
13        "version": "[4.*, 5.0.0)"
14    }
15 }
```

4. Implementierung der FizzBuzz-Funktion

Die FizzBuzz-Funktion nimmt eine Zahl als Input und gibt eine der folgenden Ausgaben zurück:

- **"FizzBuzz"**: Wenn die Zahl durch 3 und 5 teilbar ist.
- **"Fizz"**: Wenn die Zahl nur durch 3 teilbar ist.
- **"Buzz"**: Wenn die Zahl nur durch 5 teilbar ist.
- **Die Zahl selbst**: Wenn keine der Bedingungen zutrifft.

```
... if number:
...     try:
...         number = int(number) # Convert to integer
...         if number % 3 == 0 and number % 5 == 0:
...             return func.HttpResponse("FizzBuzz")
...         elif number % 3 == 0:
...             return func.HttpResponse("Fizz")
...         elif number % 5 == 0:
...             return func.HttpResponse("Buzz")
...         else:
...             return func.HttpResponse(str(number))
...     except ValueError:
```

Den Code habe ich in im Web geschrieben und hinzugefügt.

- ➔ Es ist wichtig zu sagen, dass Azure nicht immer meine Codes aus dem Visual Studio registriert hat. Wir haben uns dazu beschlossen einfach im Web-Designer den Code hineinzukopieren. *siehe 5*

4.1 C#

```
public static class Cfizzbuzz
{
    [FunctionName("cfizzbuzz")]
    public static IActionResult Run(
        [HttpTrigger(AuthorizationLevel.Anonymous, "get", "post", Route = "fizzbuzz")] HttpRequest req,
        ILogger log)
    {
        log.LogInformation("Processing FizzBuzz function in C#.");

        string numberQuery = req.Query["number"];
        int number;

        if (string.IsNullOrEmpty(numberQuery))
        {
            // Try to read the number from the request body if it's not in the query parameters
            string requestBody = new StreamReader(req.Body).ReadToEnd();
            dynamic data = JsonConvert.DeserializeObject(requestBody);
            numberQuery = data?.number;
        }

        if (!string.IsNullOrEmpty(numberQuery) && int.TryParse(numberQuery, out number))
        {
            // Implement the FizzBuzz logic
            if (number % 3 == 0 && number % 5 == 0)
            {
                return new OkObjectResult("FizzBuzz");
            }
            else if (number % 3 == 0)
            {
                return new OkObjectResult("Fizz");
            }
            else if (number % 5 == 0)
            {
                return new OkObjectResult("Buzz");
            }
            else
            {
                return new OkObjectResult(number.ToString());
            }
        }
        else
        {
            return new BadRequestObjectResult("Invalid input. Please provide a valid integer as a query parameter or in the request body.");
        }
    }
}
```

4.2 Java

```
public class javafizzbuzz {

    @FunctionName("javafizzbuzz")
    public HttpResponseMessage run(
        @HttpTrigger(name = "req",
            methods = {HttpMethod.GET, HttpMethod.POST},
            authLevel = AuthorizationLevel.ANONYMOUS,
            route = "fizzbuzz") HttpRequestMessage<Optional<String>> request,
        final ExecutionContext context) {

        context.getLogger().info("Processing FizzBuzz function in Java.");

        // Get the 'number' query parameter
        String query = request.getQueryParameters().get("number");
        String body = request.getBody().orElse(null);

        String numberStr = query != null ? query : body;
        if (numberStr == null) {
            return request.createResponseBuilder(HttpStatus.BAD_REQUEST)
                .body("Please provide a number as a query parameter or in the request body.")
                .build();
        }

        try {
            int number = Integer.parseInt(numberStr);

            // FizzBuzz logic
            if (number % 3 == 0 && number % 5 == 0) {
                return request.createResponseBuilder(HttpStatus.OK).body("FizzBuzz").build();
            } else if (number % 3 == 0) {
                return request.createResponseBuilder(HttpStatus.OK).body("Fizz").build();
            } else if (number % 5 == 0) {
                return request.createResponseBuilder(HttpStatus.OK).body("Buzz").build();
            } else {
                return request.createResponseBuilder(HttpStatus.OK).body(String.valueOf(number)).build();
            }
        } catch (NumberFormatException e) {
            return request.createResponseBuilder(HttpStatus.BAD_REQUEST)
                .body("Invalid input. Please provide a valid integer.")
                .build();
        }
    }
}
```

4.3 JS

```
module.exports = async function (context, req) {
  context.log('Processing FizzBuzz function in JavaScript.');
```



```
  let number = req.query.number || (req.body && req.body.number);
```



```
  if (!number) {
    context.res = {
      status: 400,
      body: "Please provide a number as a query parameter or in the request body."
    };
    return;
  }
```



```
  try {
    number = parseInt(number);
    if (isNaN(number)) throw new Error();
```



```
    if (number % 3 === 0 && number % 5 === 0) {
      context.res = { status: 200, body: "FizzBuzz" };
    } else if (number % 3 === 0) {
      context.res = { status: 200, body: "Fizz" };
    } else if (number % 5 === 0) {
      context.res = { status: 200, body: "Buzz" };
    } else {
      context.res = { status: 200, body: number.toString() };
    }
  } catch (error) {
    context.res = {
      status: 400,
      body: "Invalid input. Please provide a valid integer."
    };
  }
};
```

4.3 Python

```
def main(req: func.HttpRequest) -> func.HttpResponse:
    logging.info('Processing FizzBuzz function.')

    number = req.params.get('number')
    if not number:
        try:
            req_body = req.get_json()
        except ValueError:
            pass
        else:
            number = req_body.get('number')

    if number:
        try:
            number = int(number)
            if number % 3 == 0 and number % 5 == 0:
                return func.HttpResponse("FizzBuzz")
            elif number % 3 == 0:
                return func.HttpResponse("Fizz")
            elif number % 5 == 0:
                return func.HttpResponse("Buzz")
            else:
                return func.HttpResponse(str(number))
        except ValueError:
            return func.HttpResponse(
                "Invalid input. Please provide a valid integer.",
                status_code=400
            )
    else:
        return func.HttpResponse(
            "Please provide a number as a query parameter or in the request body.",
            status_code=400
        )
```


5. Deployment in Azure

Code-ansicht in Azure:
pyfizzbuzz | Programmieren und testen ...

lgfizzbuzz

Programmieren und testen

Integration

Funktionsschlüssel

Aufrufe

Protokolle

Metriken

Speichern

Verwerfen

Aktualisieren

Testen/Ausführen

Funktions-URL abrufen

Deaktivieren

Löschen

Hochladen

lgfizzbuzz / function_app.py

```
1 import azure.functions as func
2 import logging
3
4 app = func.FunctionApp(http_auth_level=func.AuthLevel.ANONYMOUS)
5
6 @app.route(route="pyfizzbuzz")
7 def pyfizzbuzz(req: func.HttpRequest) -> func.HttpResponse:
8     logging.info('Processing FizzBuzz function.')
9
10     number = req.params.get('number')
11     if not number:
12         try:
13             req_body = req.get_json()
```

Protokolle

App Insights-Protokolle

Protokollebene

Verbunden! Sie zeigen jetzt Protokolle der Funktionsausführungen im aktuellen Bereich „Code + Test“ an. Um alle Protokolle für „Protokolle“.

2024-12-18T07:43:16Z [Verbose] AuthenticationScheme: WebJobsAuthLevel was successfully authenticated.

2024-12-18T07:43:16Z [Verbose] AuthenticationScheme: Bearer was not authenticated.

2024-12-18T07:43:16Z [Verbose] Authorization was successful.

2024-12-18T07:43:16Z [Information] Executing 'Functions.pyfizzbuzz' (Reason='This function was programmatically called via...

2024-12-18T07:43:16Z [Verbose] Sending invocation id: 'ff5f6d41-78cc-4c11-806e-5e0c1fe08529

2024-12-18T07:43:16Z [Verbose] Posting invocation id:ff5f6d41-78cc-4c11-806e-5e0c1fe08529 on workerId:1130063c-0e32-436f-bc...

2024-12-18T07:43:16Z [Information] Processing FizzBuzz function.

2024-12-18T07:43:16Z [Information] Executed 'Functions.pyfizzbuzz' (Succeeded, Id=ff5f6d41-78cc-4c11-806e-5e0c1fe08529, Dur...

Eingabe

Ausgabe

HTTP-Antwortcode

200 OK

Inhalt der HTTP-Antwort

Fizz

Ausführen

Schließen

Ansicht Bsp.: Pyfizzbuzz in der Ressource.

Name	Trigger	Status
pyfizzbuzz	HTTP	<div>✓ Aktiviert</div>

Ansicht der Ressource – HTTP is up and running:

^ Zusammenfassung

Ressourcengruppe (...): [FaaS](#)

Status: Wird ausgeführt

Speicherort ([verschieben](#)): Switzerland North

Abonnement (...): [Azure for Students](#)

Abonnement-ID: 2fabf96-af90-42d3-abe1-e231aff6b1af

Tags ([bearbeiten](#)): [Tags hinzufügen](#)

Standarddomäne: [web1g346.azurewebsites.net](#)

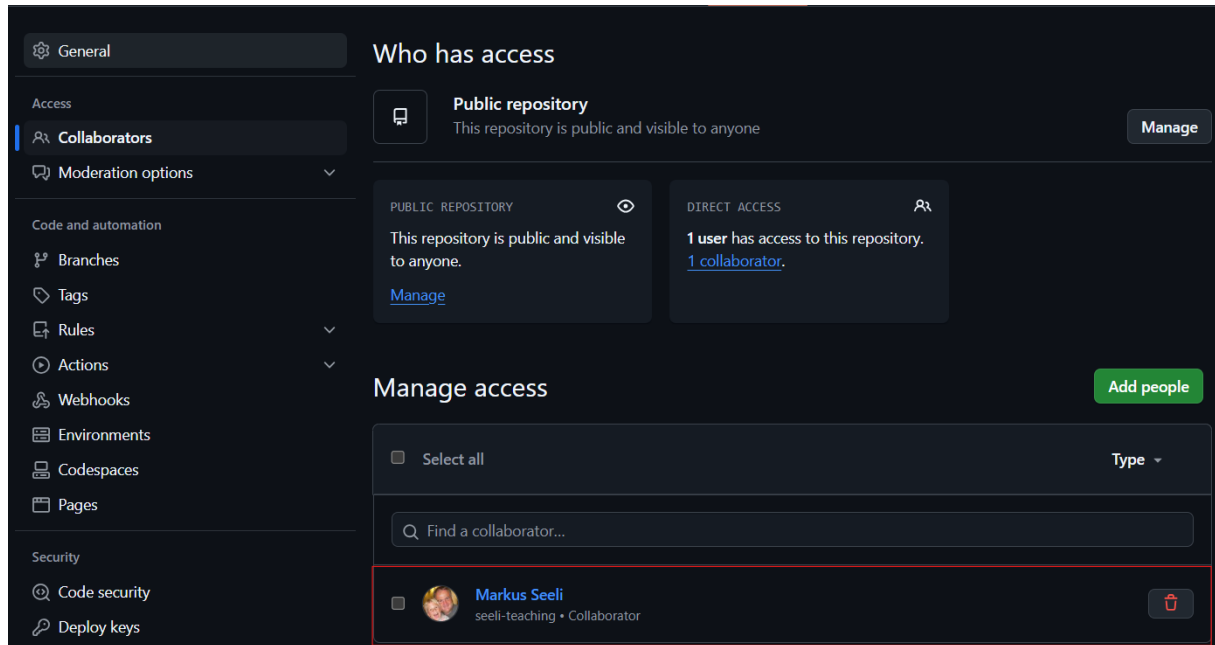
Betriebssystem: Linux

App Services-Plan: [ASP-FaaS-b6c3 \(Y1: 0\)](#)

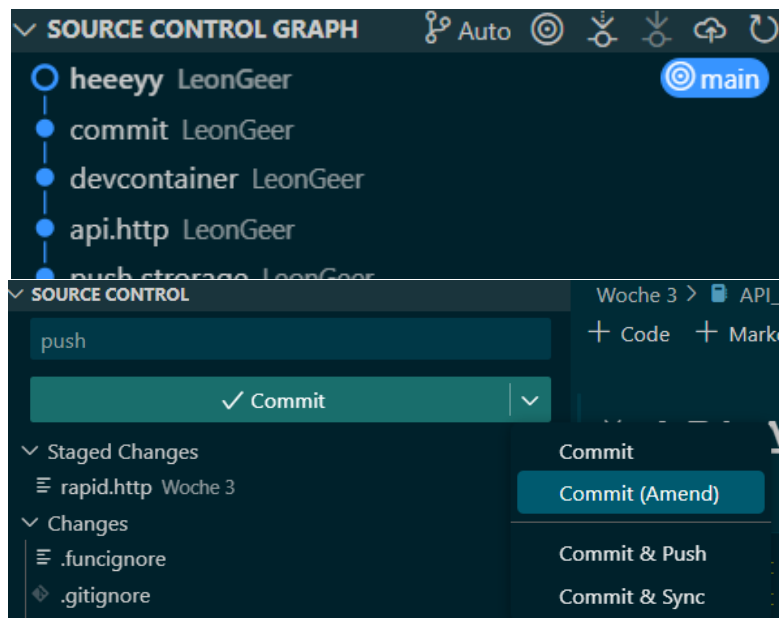
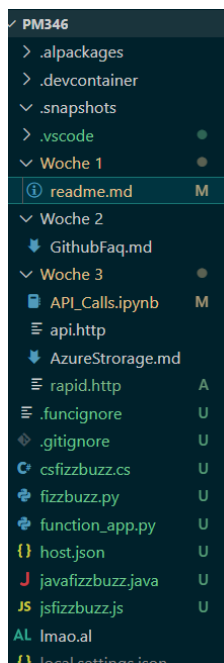
Laufzeitversion: 4.1036.2.2

6. GitHub

1. GitHub Account erstellt.
2. Git Repo erstellt.
3. Git Repo geteilt.



5. Git mit Visual Studio (und Azure) verbinden, Repo clonen und verbinden



7. Fazit

Mein Fazit der Arbeit hebt die Implementierung und Bereitstellung der FizzBuzz-Funktion in vier verschiedenen Programmiersprachen hervor. Serverlose Architekturen wie Azure Functions zeigt die Flexibilität und plattformunabhängige Entwicklung. Die Wahl von C#, Java, JavaScript und Python verdeutlicht die Vielseitigkeit der Plattform. Abschließend bietet die Arbeit eine Grundlage für weiterführende Projekte im Bereich serverlose Architekturen und deren Anwendung in komplexeren Szenarien. Hin und wieder hatte ich einige Schwierigkeiten, welche aber relativ schnell gelöst wurden. Gleichzeitig hat mir dieses Modul doch Spass gemacht und eine Tür für ein Thema geöffnet, wo ich noch nichts darüber kannte. Mit dem Unterricht und den Aufgaben kam ich relativ zügig voran.