06/19/2022

CI/CD & Performance

Kwetter-Case

Samir Zalmay
FONTYS HOGESCHOLEN

INHOUD

1. INTRODUCTION

This document is made to prove that I have enough knowledge in the learning outcome DevOps. A separate document is made for Monitoring. In chapter 2 you can find how I implemented CI/CD and in chapter 3 the performance tests executed by Jmeter.

2. CI/CD

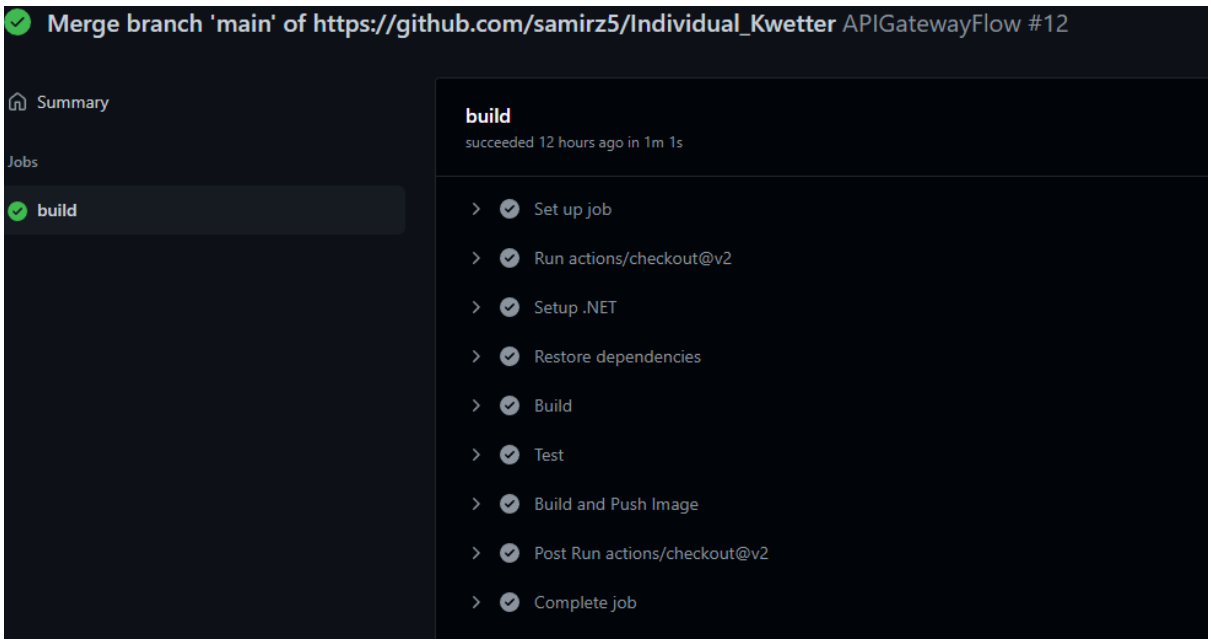
Continuous integration and deployment is of outmost important to have for your project. Not only does this make production deadlines faster, but also, once it is set up, there is no more headache, with manually pushing and deploying your yaml and docker files.

I looked into a few different CI/CD processes that may be utilized. First, I investigated Azure Cloud's pipelines. I chose this since Fontys provided us with a free account. I wanted to use other Azure services, which is why I wanted to construct the pipeline there as well, so that all of the services we use are on the same Cloud platform. I was unable to use this pipeline due to technical difficulties. This was due to a mistake reporting that parallelism had not been requested. Because free accounts have been abusing the pipelines, Azure has taken action.

```yaml
name: APIGatewayFlow

on:
  push:
    branches: [ main ]
    paths:
      - Backend/APIGateway/**
      - .github/workflows/ApiGateway.yml
  pull_request:
    branches: [ main ]
    paths:
      - APIGateway/**
      - .github/workflows/ApiGateway.yml

jobs:
  build:

    runs-on: ubuntu-latest
    steps:
    - uses: actions/checkout@v2
    - name: Setup .NET
      uses: actions/setup-dotnet@v1
      with:
        dotnet-version: 5.0.x
    - name: Restore dependencies
      run: dotnet restore
      working-directory: Backend/APIGateway
    - name: Build
      run: dotnet build --no-restore
      working-directory: Backend/APIGateway
    - name: Test
      run: dotnet test --no-build --verbosity normal
      working-directory: Backend/APIGateway
    - name: Build and Push Image
      uses: mr-smithers-excellent/docker-build-push@v5
      with:
        image: samirz5/apigateway
        tags: latest
        registry: docker.io
        dockerfile: ./Backend/APIGateway/Dockerfile
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}
```

I also looked at Gitlab from Fhict, but I quickly realized that setting up a CI/CD pipeline on Gitlab requires a lot more work. Runners, which are required for the pipeline, must be installed for Gitlab. With GitHub Actions, GitHub manages the pipeline and this is why I used GitHub Actions to build the process.. In order to configure a CI/CD pipeline in GitHub Actions, a workflow.yaml file must be produced for each service. On the picture above, you can find a workflow.yaml file for my ApiGateWay.

This workflow runs the following steps in the picture below. As you can see the pipeline has succeeded. However, there were some issues regarding naming and spacing, as I reorganized my repository, in which the Dockerfile was not too happy about. There is still one thing that is missing from the pipeline, and that is adding deployment. Since I did not manage to deploy my cluster in any Cloud-Service, I left it open for the next time.



3. PERFORMANCE TEST

Performance testing is a good way to test whether my non-functional requirements are holding up. This includes the following:

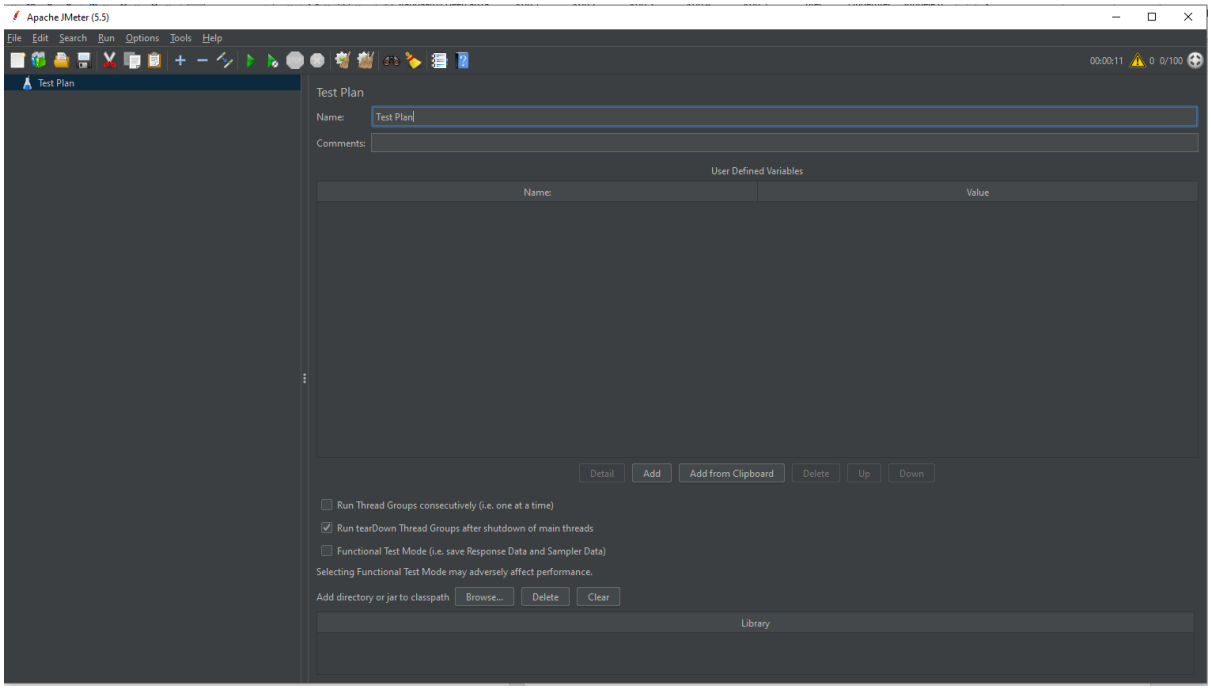
- Scalable (horizontal or vertical)
- Almost zero down-time
- Responsiveness while overused.

To performance test my project I made use of Jmeter.

Setting up Jmeter is annoying as the explanation is long and annoying to go through. So I decided to make a small tutorial her for myself in the future.

Download
First download install Jmeter binaries and unpack the zip file. You can make use of Jmeter by going to the bin folder and open the jmeter batch file.

This will open the following screen

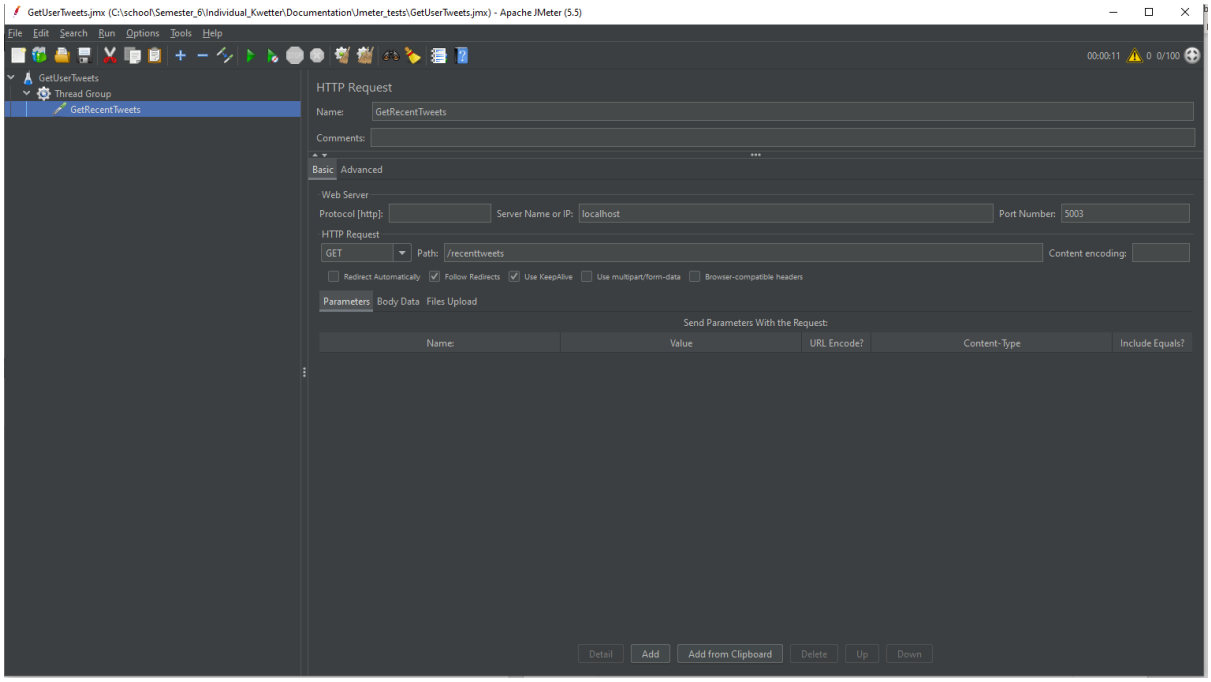


Here you can rename the Test plan to your liking and save it. By right clicking the name on the left side, we can add threads and other stuff:

For threads to remember:

- Number of threads: number of users Jmeter will generate for now I use 100;
- Ramp-up Period: time it takes before starting the thread over.
- Loop count: number of times test will be executed

For testing my endpoints a simple HTTP-request can be made by right clicking again -> sampler -> http request. This will speak for it self.



In order to view the results of your tests we need to add some listeners by right clicking again and add -> listener

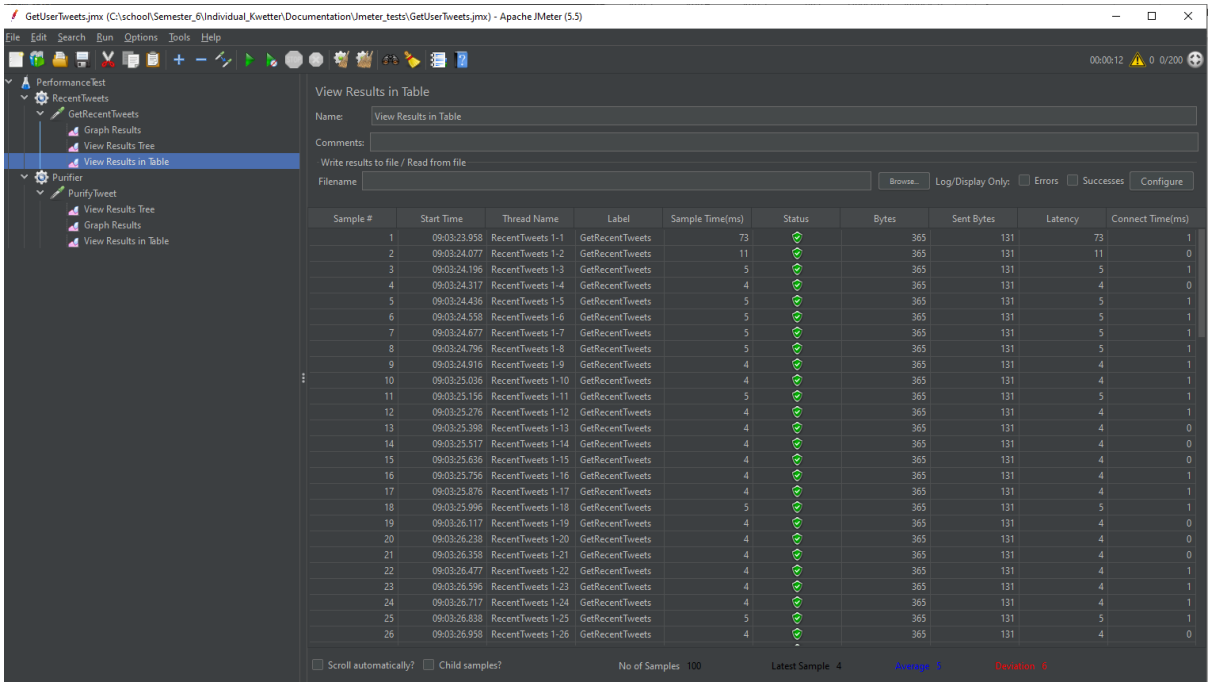
For now I have chosen to see the Result Tree and , Graph and results in table.

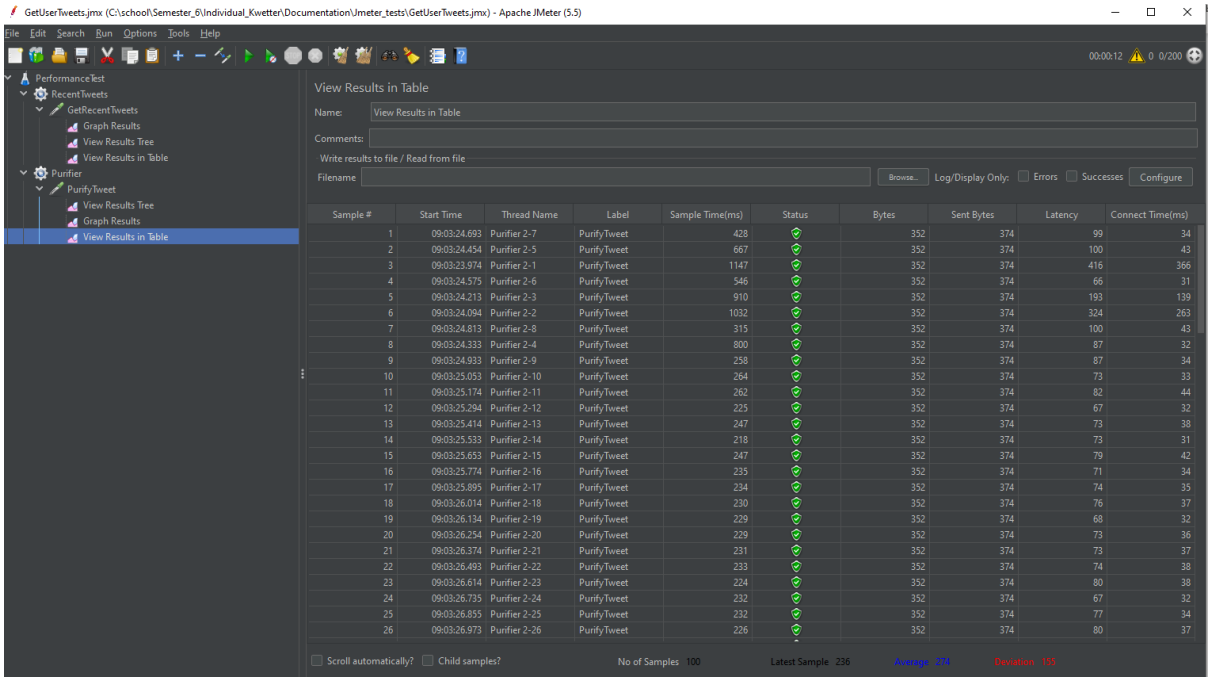
After setting up you can run the tests by clicking on the green button.

RESULTS

As I have not been able to deploy my project completely, I can not performance test on the cloud regarding my project. However, I added FaaS and added performance test for this. Unfortunately, I did not find the time to completely performance test both locally and my FaaS, to see the differences locally and the scaling in the cloud. Both performance test worked perfectly and the only thing I have to do is to add more users/threads for extra load.

For now I only used this document to proove that I can setup performance testing and I have the knowledge for it.


GetUserTweets.jmx (C:\school\Semester_6\Individual_Kwetter\Documentation\Jmeter_tests\GetUserTweets.jmx) - Apache JMeter (5.5)
File Edit Search Run Options Tools Help
00:00:12 0 0/200
Performance Test
RecentTweets
GetRecentTweets
Graph Results
View Results Tree
View Results in Table
Purifier
PurifyTweet
View Results Tree
Graph Results
View Results in Table
View Results in Table
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename
Browse... Log/Display Only: Errors Successes Configure
Sample # Start Time Thread Name Label Sample Time(ms) Status Bytes Sent Bytes Latency Connect Time(ms)
1 09:03:23.958 RecentTweets 1-1 GetRecentTweets 73 365 131 73 1
2 09:03:24.077 RecentTweets 1-2 GetRecentTweets 11 365 131 11 0
3 09:03:24.196 RecentTweets 1-3 GetRecentTweets 5 365 131 5 1
4 09:03:24.317 RecentTweets 1-4 GetRecentTweets 4 365 131 4 0
5 09:03:24.436 RecentTweets 1-5 GetRecentTweets 5 365 131 5 1
6 09:03:24.558 RecentTweets 1-6 GetRecentTweets 5 365 131 5 1
7 09:03:24.677 RecentTweets 1-7 GetRecentTweets 5 365 131 5 1
8 09:03:24.796 RecentTweets 1-8 GetRecentTweets 5 365 131 5 1
9 09:03:24.916 RecentTweets 1-9 GetRecentTweets 4 365 131 4 1
10 09:03:25.036 RecentTweets 1-10 GetRecentTweets 4 365 131 4 1
11 09:03:25.156 RecentTweets 1-11 GetRecentTweets 5 365 131 5 1
12 09:03:25.276 RecentTweets 1-12 GetRecentTweets 4 365 131 4 1
13 09:03:25.398 RecentTweets 1-13 GetRecentTweets 4 365 131 4 0
14 09:03:25.517 RecentTweets 1-14 GetRecentTweets 4 365 131 4 0
15 09:03:25.636 RecentTweets 1-15 GetRecentTweets 4 365 131 4 0
16 09:03:25.756 RecentTweets 1-16 GetRecentTweets 4 365 131 4 1
17 09:03:25.876 RecentTweets 1-17 GetRecentTweets 4 365 131 4 1
18 09:03:25.996 RecentTweets 1-18 GetRecentTweets 5 365 131 5 1
19 09:03:26.117 RecentTweets 1-19 GetRecentTweets 4 365 131 4 0
20 09:03:26.238 RecentTweets 1-20 GetRecentTweets 4 365 131 4 0
21 09:03:26.358 RecentTweets 1-21 GetRecentTweets 4 365 131 4 0
22 09:03:26.477 RecentTweets 1-22 GetRecentTweets 4 365 131 4 1
23 09:03:26.596 RecentTweets 1-23 GetRecentTweets 4 365 131 4 1
24 09:03:26.717 RecentTweets 1-24 GetRecentTweets 4 365 131 4 1
25 09:03:26.838 RecentTweets 1-25 GetRecentTweets 5 365 131 5 1
26 09:03:26.958 RecentTweets 1-26 GetRecentTweets 4 365 131 4 0
Scroll automatically? Child samples? No of Samples 100 Latest Sample 4 Average 5 Deviation 6


GetUserTweets.jmx (C:\school\Semester_6\Individual_Kwetter\Documentation\Jmeter_tests\GetUserTweets.jmx) - Apache JMeter (5.5)
File Edit Search Run Options Tools Help
00:00:12 0 0/200
Performance Test
RecentTweets
GetRecentTweets
Graph Results
View Results Tree
View Results in Table
Purifier
PurifyTweet
View Results Tree
Graph Results
View Results in Table
View Results in Table
Name: View Results in Table
Comments:
Write results to file / Read from file
Filename
Browse... Log/Display Only: Errors Successes Configure
Sample # Start Time Thread Name Label Sample Time(ms) Status Bytes Sent Bytes Latency Connect Time(ms)
1 09:03:24.693 Purifier 2-7 PurifyTweet 428 352 374 99 34
2 09:03:24.454 Purifier 2-5 PurifyTweet 667 352 374 100 43
3 09:03:23.974 Purifier 2-1 PurifyTweet 1147 352 374 416 366
4 09:03:24.575 Purifier 2-6 PurifyTweet 546 352 374 66 31
5 09:03:24.213 Purifier 2-3 PurifyTweet 910 352 374 193 139
6 09:03:24.094 Purifier 2-2 PurifyTweet 1032 352 374 324 263
7 09:03:24.813 Purifier 2-8 PurifyTweet 315 352 374 100 43
8 09:03:24.333 Purifier 2-4 PurifyTweet 800 352 374 87 32
9 09:03:24.933 Purifier 2-9 PurifyTweet 258 352 374 87 34
10 09:03:25.053 Purifier 2-10 PurifyTweet 264 352 374 73 33
11 09:03:25.174 Purifier 2-11 PurifyTweet 262 352 374 82 44
12 09:03:25.294 Purifier 2-12 PurifyTweet 225 352 374 67 32
13 09:03:25.414 Purifier 2-13 PurifyTweet 247 352 374 73 38
14 09:03:25.533 Purifier 2-14 PurifyTweet 218 352 374 73 31
15 09:03:25.653 Purifier 2-15 PurifyTweet 247 352 374 79 42
16 09:03:25.774 Purifier 2-16 PurifyTweet 235 352 374 71 34
17 09:03:25.895 Purifier 2-17 PurifyTweet 234 352 374 74 35
18 09:03:26.014 Purifier 2-18 PurifyTweet 230 352 374 76 37
19 09:03:26.134 Purifier 2-19 PurifyTweet 229 352 374 68 32
20 09:03:26.254 Purifier 2-20 PurifyTweet 229 352 374 73 36
21 09:03:26.374 Purifier 2-21 PurifyTweet 231 352 374 73 37
22 09:03:26.493 Purifier 2-22 PurifyTweet 233 352 374 74 38
23 09:03:26.614 Purifier 2-23 PurifyTweet 224 352 374 80 38
24 09:03:26.735 Purifier 2-24 PurifyTweet 232 352 374 67 32
25 09:03:26.855 Purifier 2-25 PurifyTweet 232 352 374 77 34
26 09:03:26.973 Purifier 2-26 PurifyTweet 226 352 374 80 37
Scroll automatically? Child samples? No of Samples 100 Latest Sample 236 Average 274 Deviation 155