In this exercise we will find a simple performance problem. We will see how it is reflected in a performance load tool, in profiler output , and after fixing it - what changed in the profiler and the small load that we will do.

* Clone the repository <https://github.com/tamartwe/basic_performance_workshop>
* Install autocannon - a nice tool to generate load on your system , by doing **npm install autocannon -g**
* The repository contains 2 scripts - **crypto\_server.js** and **crypto\_server\_with\_profiler.js**
* Run the script **crypto\_server.js** by doing in your command line

**node crypto\_server.js**

* Now, let’s do our first measurement with autocannon. While the crypto server is up , run the command

**autocannon -c 10 localhost:3000/api**

Small explanation about autocannon - Autocannon is a tool that simulate load on you http server . This tool has multiple abilities and I recommend reading about it and exploring it. We are loading as many requests that we can with 10 concurrent connections (simulated by the -c option)

* + What is the latency that you got ?
  + What is the 99% percentile of the latency ?
  + How many requests can you serve per second ? Look at the 50% percentile and at the standard deviation.
* It is time to open the profiler (-:
  + Now, Let’s look at the code of **crypto\_server\_with\_profiler.js** a little bit .
  + In the code we are using a v8 profiler library called **v8\_profiler\_next**
  + In line 10 you can see that we set the generated type of the output to a type that visual studio code can recognise.
  + You can look at the function “stopProfiling” to see how the profiler generates the input.
* Now, run the script **crypto\_server\_with\_profiler.js** and send one API request to the server .
  + send an HTTP Get request of **localhost:3000/api .** No headers or authentication is needed . You can use postman, curl, browser or any other way.
  + After sending one request , you can take the server down.
* A file called **sync-profiler-file-name.cpuprofile** is created under the root folder. You can open it in Visual Studio Code.
* Can you identify the performance problem in the trace ?
* Try to solve the performance problem in 2 ways. You can use another API type for the problematic api or you can use worker threads .
* After you rewrite the **crypto\_server.js** and fix the performance problem , rerun the load with autocannon

**autocannon -c 10 localhost:3000/api**

and answer the questions again :

* + What is the latency that you got ?
  + What is the 99% percentile of the latency ?
  + How many requests can you serve per second ? Look at the 50% percentile and at the standard deviation.