Testing Azure Kubernetes deployment



Table of Contents

[Testing with Postman 3](#_Toc156207283)

[Testing with frontend 3](#_Toc156207284)

[Load testing with K6 4](#_Toc156207285)

[What is k6? 4](#_Toc156207286)

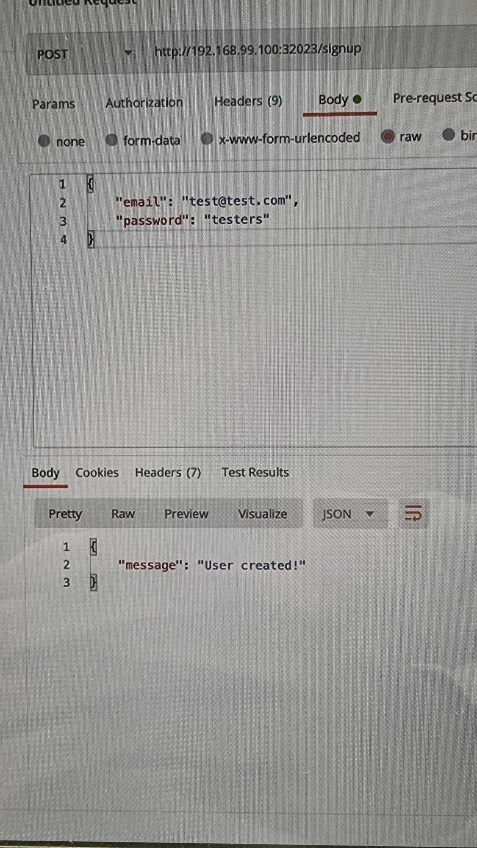
[How did I use k6? 4](#_Toc156207287)

[Stress testing 6](#_Toc156207288)

[Spike testing 7](#_Toc156207289)

# Testing with Postman

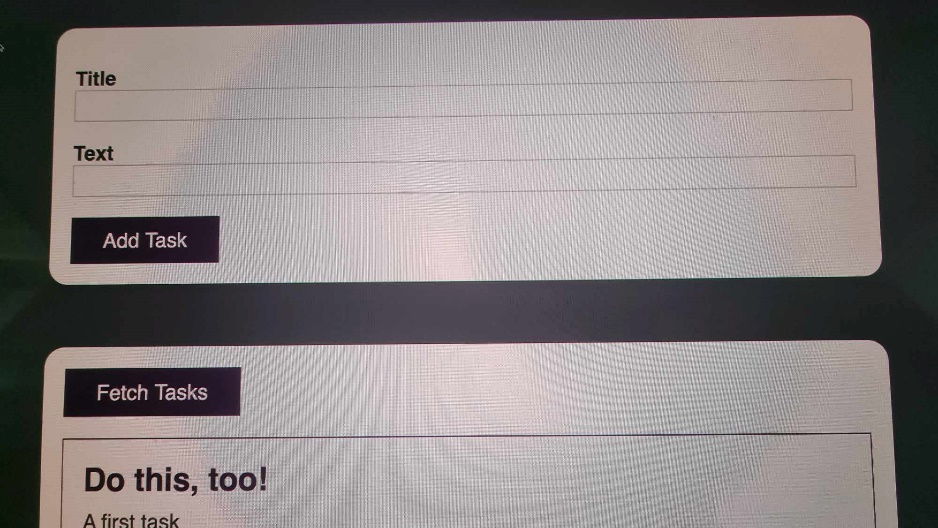
This is just making sure that the backend works properly.

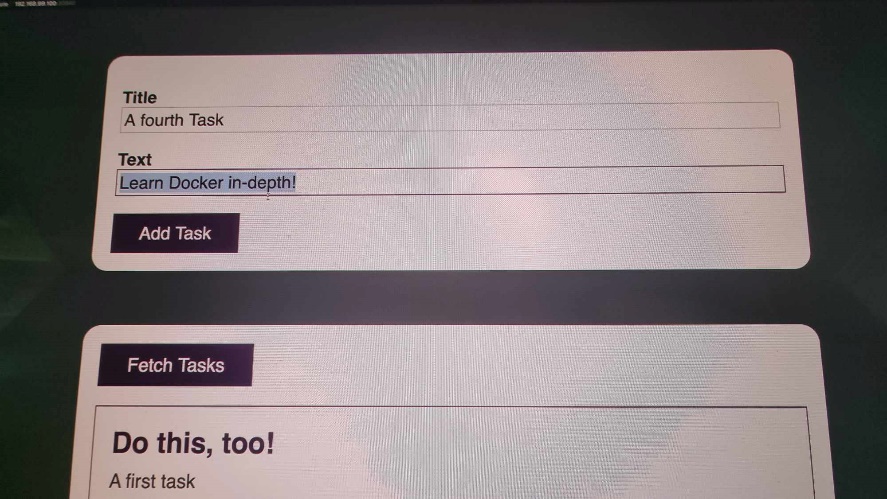


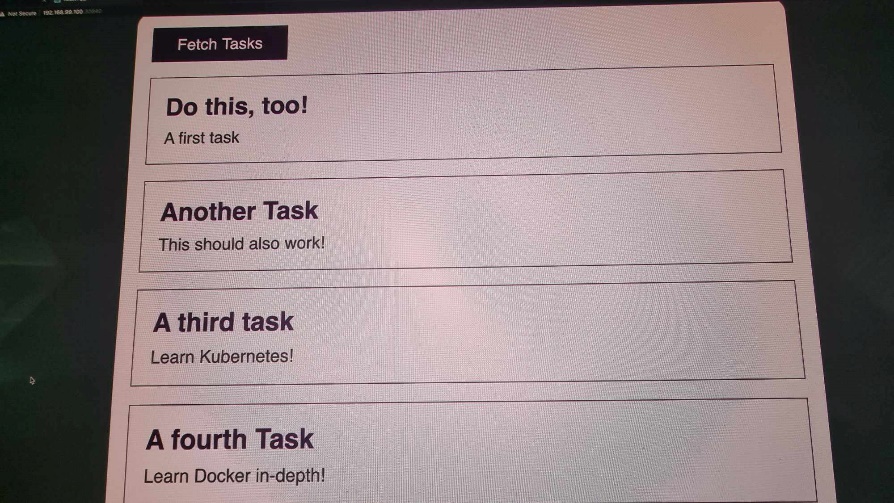
We can see through this that the application backend works as intended.

# Testing with frontend

This is just making sure that the frontend and backend work well together.







And we can see that the application is doing exactly as it should be doing. It is fetching tasks and writing them.

# Load testing with K6

## What is k6?

Grafana k6 is an open-source load testing tool that makes performance testing easy and productive for engineering teams. k6 is free, developer-centric, and extensible.

Using k6, you can test the reliability and performance of your systems and catch performance regressions and problems earlier. k6 will help you to build resilient and performant applications that scale.

k6 is developed by [Grafana Labs](https://grafana.com/) and the community.

## How did I use k6?

I used it to first create a small 10 second test and

A computer screen shot of a program

Description automatically generated

After that it did not run into any problems, but this is a basic test.  
The tests are performed before the production, because it might cause problems to the users.

For the second test we want to slowly ramp up the target of requests and have it work for around 20 mins.

What we are hoping to see is 99% of requests should come back in under 100ms:

A screen shot of a computer

Description automatically generated

After setting that requirement and running the load test for a bit we see A computer screen with numbers and letters

Description automatically generated

That all of the requests have returned and in the necessary duration as well, so everything is working properly.

# Stress testing

We want to see how the application will handle an increased load- requests are slowly ramped up from 200-1000 requests per second:

At 200 it is stable:

A screenshot of a computer

Description automatically generated

At 800 it is starting to show a bit of strain:

A screenshot of a computer

Description automatically generated

And at a 1000 it is still holding up, so it should be more than enough to handle an increased load of traffic: A screenshot of a computer

Description automatically generated

(The average cpu load is a bit over 50%)

# Spike testing

Very quick ramp up of traffic in which we will run 2000 requests per second and remove the 100ms deadline: A screenshot of a computer

Description automatically generated

And as expected the cluster is struggling a bit with 2000 requests per second, but in the tests at this point of time we are only using one node through a simulated-dummy cluster on our local machine, so this should be an amount of traffic that the cloud providers on which we are deploying can handle it.