Test report

ROOSH CHALLENGE 5

ROOSH.

Date	**:**	**123/01/2024**
Version	**:**	**1.0**
Status	**:**	**Open**
Author	**:**	**Group 1**

Table of Contents

Testing HPA using basicHPA.yml & advancedHPA.yml

To test the the functionalities of the two demo files we have created we deployed a busybox that is executing a command every 0.01 minutes. The command keeps spamming the health route of the pod in order to increase the load on the CPU and memory. The complete command can be seen in figure 1.

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\jelle> kubectl run -i --tty load-generator1 --rm --image=busybox:1.28 --restart=Never -- /bin/sh -c "while sleep 0.01; do wget -q -O- http://website/health; done"
```

Figure 1

In figure 2 you can see that we have used kubectl get HPA website-autoscaler –watch. This gives a live overview of what happeneds on the autoscaller and shoes the replicas that have been created. This figure shows that the HPA configuration is working.

```
PS C:\Users\jelle\OneDrive\Documenten\School\Fontys\Sem 3 - Infra\MDP> kubectl get hpa website-autoscaler
NAME                REFERENCE              TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
website-autoscaler   Deployment/website   1%/50%    1         10        1          40m
PS C:\Users\jelle\OneDrive\Documenten\School\Fontys\Sem 3 - Infra\MDP>
 kubectl get hpa website-autoscaler --watch
NAME                REFERENCE              TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
website-autoscaler   Deployment/website   1%/50%    1         10        1          40m
website-autoscaler   Deployment/website   69%/50%   1         10        1          41m
website-autoscaler   Deployment/website   69%/50%   1         10        2          41m
website-autoscaler   Deployment/website   44%/50%   1         10        2          42m
website-autoscaler   Deployment/website   42%/50%   1         10        2          43m
website-autoscaler   Deployment/website   56%/50%   1         10        2          44m
website-autoscaler   Deployment/website   56%/50%   1         10        3          44m
website-autoscaler   Deployment/website   47%/50%   1         10        3          45m
website-autoscaler   Deployment/website   48%/50%   1         10        3          47m
website-autoscaler   Deployment/website   10%/50%   1         10        3          48m
website-autoscaler   Deployment/website   1%/50%    1         10        3          49m
website-autoscaler   Deployment/website   1%/50%    1         10        3          53m
website-autoscaler   Deployment/website   1%/50%    1         10        1          53m
```

Figure 2

Testing Monitoring functionality using Prometheus Grafana

We tested Prometheus and Grafana the same way as we tested HPA in general. We used 2 busybox instances that execute a request command every 0.01 minutes until they scaled. We could monitor this from Grafana as seen in figure 3. In this figure you can see the apache-daan instance which I used for testing with the load generators. You can also see when it scaled and how many replicas there currently are.

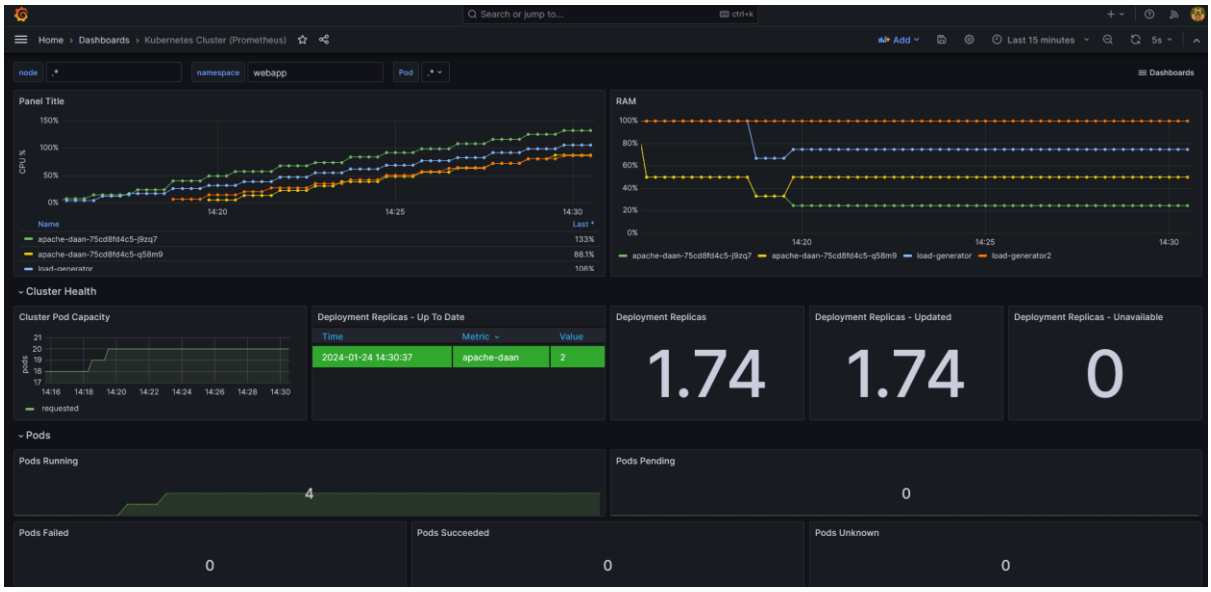

Figure 3

Figure 4 shows the monitoring from the last 5 minutes.

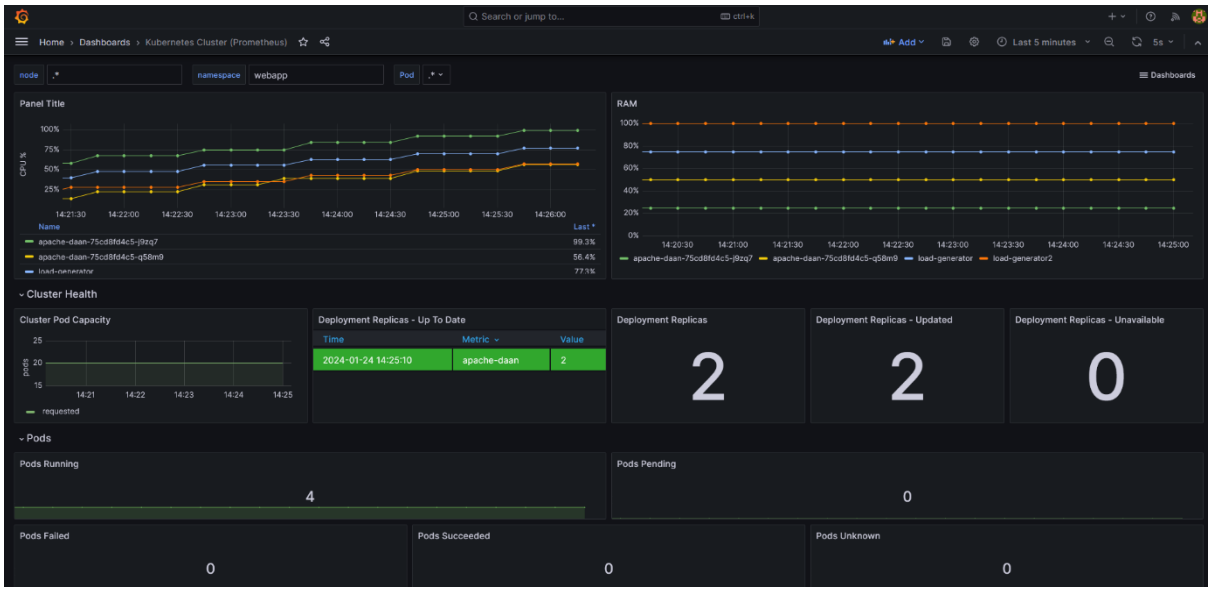

Figure 4