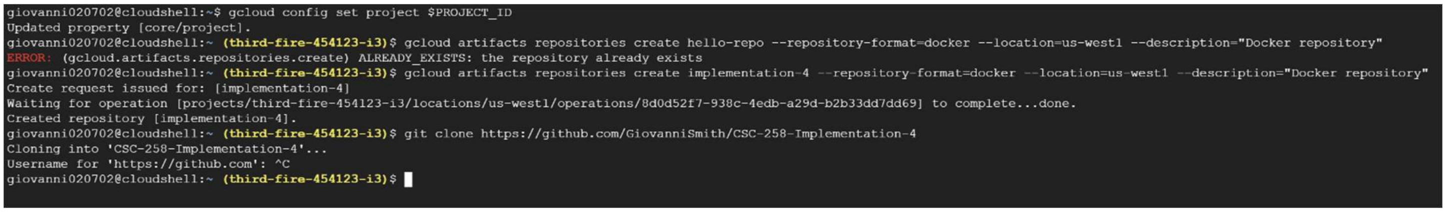
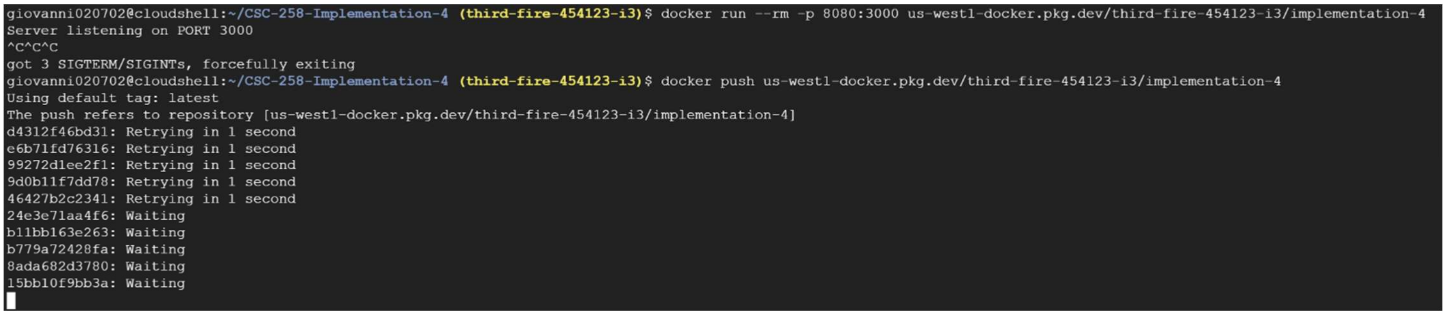
Giovanni Smith CSC-258 Implementation 4

I created a simple web application and docker image before the deadline (March 16), as can be seen in the GitHub commits (https://github.com/GiovanniSmith/Implementation4). That was the easy part. However, the hard part was pushing the docker image to the cloud. There were many small details and implicit instructions in Google's directions that stumped me for days. Because of that, I was not able to submit this assignment on time, and troubleshooting the instructions resulted in one of the most frustrating experiences in many semesters. I created and deleted about 8 total projects before I was finally able to upload the docker image to the cloud.



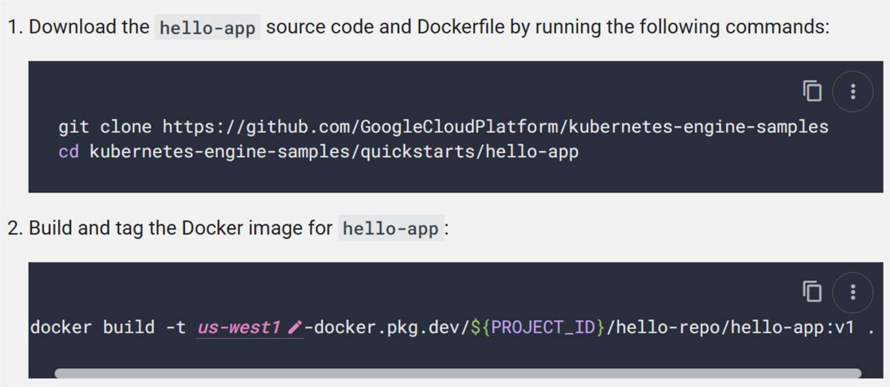
```
giovanni020702@cloudshell:~$ gcloud config set project $PROJECT_ID
Updated property [core/project].
giovanni020702@cloudshell:~ (third-fire-454123-i3)$ gcloud artifacts repositories create hello-repo --repository-format=docker --location=us-west1 --description="Docker repository"
ERROR: (gcloud.artifacts.repositories.create) ALREADY_EXISTS: the repository already exists
giovanni020702@cloudshell:~ (third-fire-454123-i3)$ gcloud artifacts repositories create implementation-4 --repository-format=docker --location=us-west1 --description="Docker repository"
Create request issued for: [implementation-4]
Waiting for operation [projects/third-fire-454123-i3/locations/us-west1/operations/8d0d52f7-938c-4edb-a29d-b2b33dd7dd69] to complete...done.
Created repository [implementation-4].
giovanni020702@cloudshell:~ (third-fire-454123-i3)$ git clone https://github.com/GiovanniSmith/CSC-258-Implementation-4
Cloning into 'CSC-258-Implementation-4'...
Username for 'https://github.com': ^C
giovanni020702@cloudshell:~ (third-fire-454123-i3)$
```

As seen in the screenshot above, I initially had trouble cloning my GitHub project. Simply replacing the repository link in the instructions with my own had the system requesting my username and password in the terminal. Even when I did that, it still did not work because apparently something had been deprecated. I then realized the problem – it was asking for my username and password because my repository was private by default. The repository that Google used in the example was public. When I made my repository public, I was finally able to make progress, however there came more issues.



```
giovanni020702@cloudshell:~/CSC-258-Implementation-4 (third-fire-454123-i3)$ docker run --rm -p 8080:3000 us-west1-docker.pkg.dev/third-fire-454123-i3/implementation-4
Server listening on PORT 3000
^C^C^C
got 3 SIGTERM/SIGINTs, forcefully exiting
giovanni020702@cloudshell:~/CSC-258-Implementation-4 (third-fire-454123-i3)$ docker push us-west1-docker.pkg.dev/third-fire-454123-i3/implementation-4
Using default tag: latest
The push refers to repository [us-west1-docker.pkg.dev/third-fire-454123-i3/implementation-4]
d4312f46bd31: Retrying in 1 second
e6b71fd76316: Retrying in 1 second
99272d1ee2f1: Retrying in 1 second
9d0b11f7dd78: Retrying in 1 second
46427b2c2341: Retrying in 1 second
24e3e71aa4f6: Waiting
b11bb163e263: Waiting
b779a72428fa: Waiting
8ada682d3780: Waiting
15bb10f9bb3a: Waiting
```

As seen in the screenshot above, I was able to run the repository within the Google shell – the "Server listening on PORT 3000" being proof. However, when I attempted to push it, I got messages saying "retrying in x seconds" and "waiting." After 30 seconds or so, that response exited, saying that my original message was in the wrong format. I then discovered the truly baffling culprit.



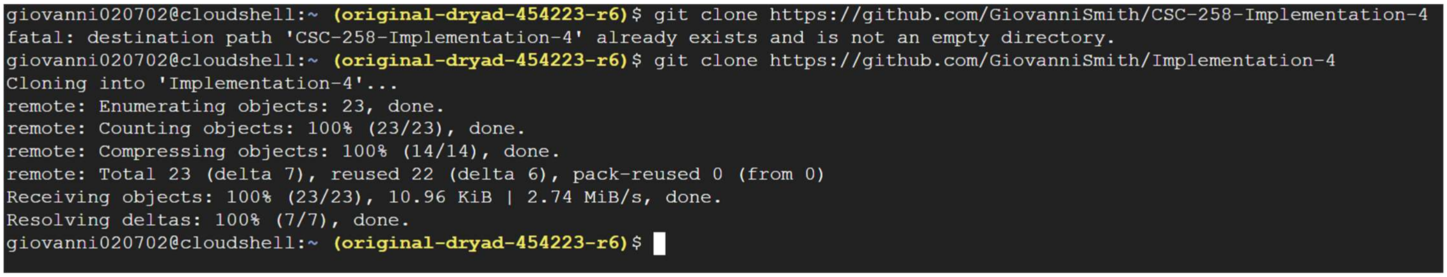
1. Download the `hello-app` source code and Dockerfile by running the following commands:

```
git clone https://github.com/GoogleCloudPlatform/kubernetes-engine-samples
cd kubernetes-engine-samples/quickstarts/hello-app
```

2. Build and tag the Docker image for `hello-app`:

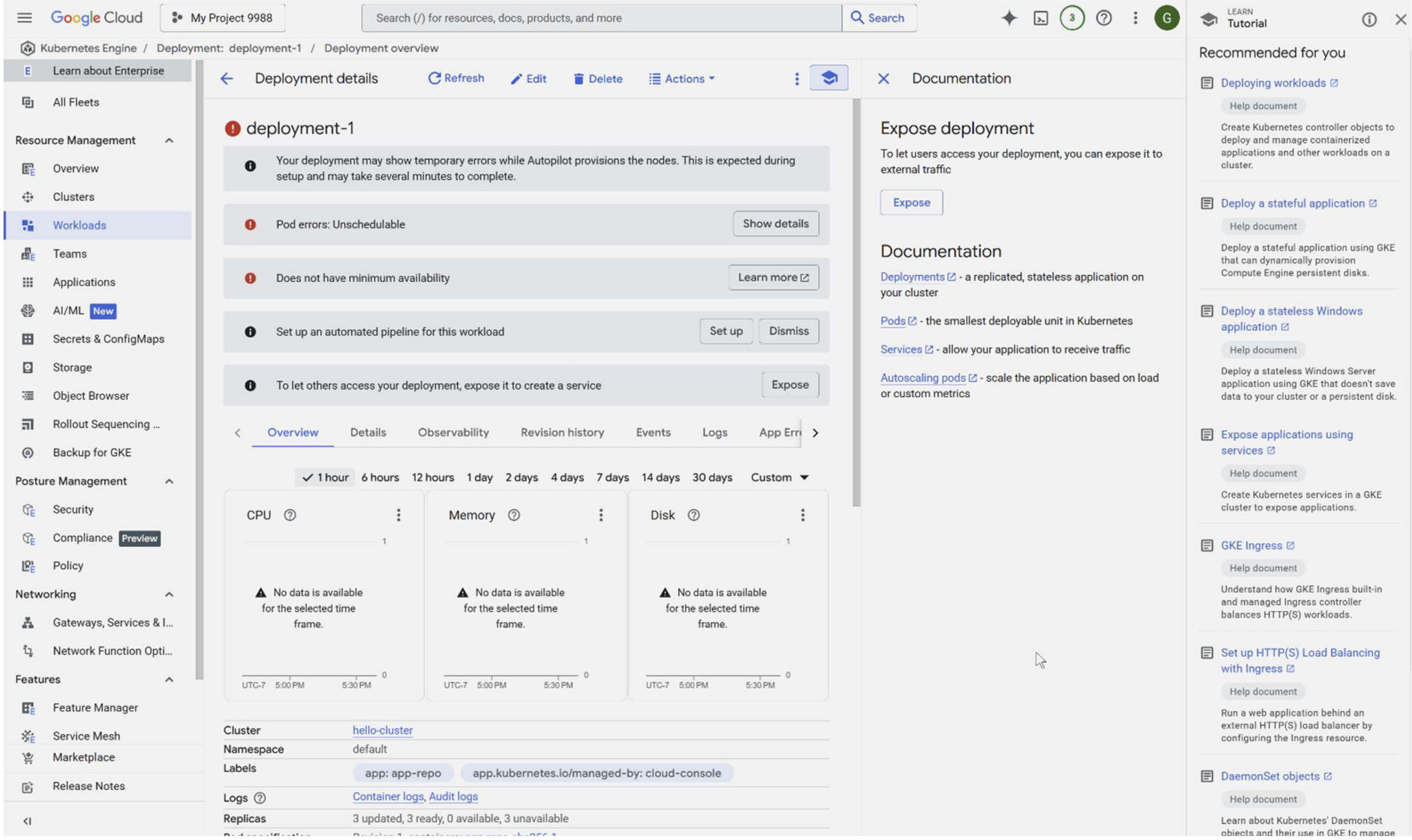
```
docker build -t us-west1-docker.pkg.dev/${PROJECT_ID}/hello-repo/hello-app:v1 .
```

As seen in the screenshot above, the GitHub repository's dockerfile is buried in a subfolder called "hello-app." My GitHub repository had all of the files (including the dockerfile) in the main root folder. On my computer, I placed all of the files (except for .gitignore) inside a main folder called "app-repo." This way, I would have to navigate to a subfolder just like in the example. Notice how in my line, there are only two forward slashes in the right part, while in the example there are three forward slashes. That worked, but there was another issue that confused me for a while.



```
giovanni020702@cloudshell:~ (original-dryad-454223-r6)$ git clone https://github.com/GiovanniSmith/CSC-258-Implementation-4
fatal: destination path 'CSC-258-Implementation-4' already exists and is not an empty directory.
giovanni020702@cloudshell:~ (original-dryad-454223-r6)$ git clone https://github.com/GiovanniSmith/Implementation-4
Cloning into 'Implementation-4'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 23 (delta 7), reused 22 (delta 6), pack-reused 0 (from 0)
Receiving objects: 100% (23/23), 10.96 KiB | 2.74 MiB/s, done.
Resolving deltas: 100% (7/7), done.
giovanni020702@cloudshell:~ (original-dryad-454223-r6)$
```

As seen in the screenshot above, I changed the name of my repository to a different name. I had been creating and deleting multiple projects in order to start fresh every time I got an error, but for some reason, the name of my GitHub repository had still been remembered by the system, even when I deleted an entire project. Because of this, I had to change the repository name to something different. One final small thing was that the default exposed port in Google Cloud was 8080, however the port in my files was 3000. I changed the port number in the YAML file online to 3000 so that it would match with the port number in the cloud.



deployment-1

Your deployment may show temporary errors while Autopilot provisions the nodes. This is expected during setup and may take several minutes to complete.

Pod errors: Unschedulable

Does not have minimum availability

Set up an automated pipeline for this workload

To let others access your deployment, expose it to create a service

No data is available for the selected time frame.

Cluster	hello-cluster
Namespace	default
Labels	app: app-repo app.kubernetes.io/managed-by: cloud-console
Logs	Container logs, Audit logs
Replicas	3 updated, 3 ready, 0 available, 3 unavailable

Expose deployment

To let users access your deployment, you can expose it to external traffic

Documentation

Deployments - a replicated, stateless application on your cluster

Pods - the smallest deployable unit in Kubernetes

Services - allow your application to receive traffic

Autoscaling pods - scale the application based on load or custom metrics

Deploying workloads

Create Kubernetes controller objects to deploy and manage containerized applications and other workloads on a cluster.

Deploy a stateful application

Deploy a stateful application using GKE that can dynamically provision Compute Engine persistent disks.

Deploy a stateless Windows application

Deploy a stateless Windows Server application using GKE that doesn't save data to your cluster or a persistent disk.

Expose applications using services

Create Kubernetes services in a GKE cluster to expose applications.

GKE Ingress

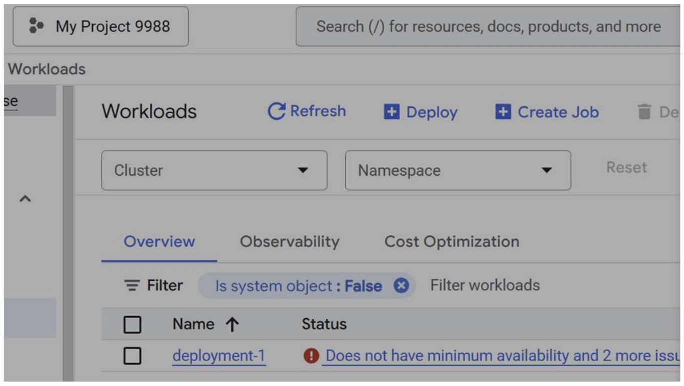
Understand how GKE Ingress built-in and managed Ingress controller balances HTTP(S) workloads.

Set up HTTP(S) Load Balancing with Ingress

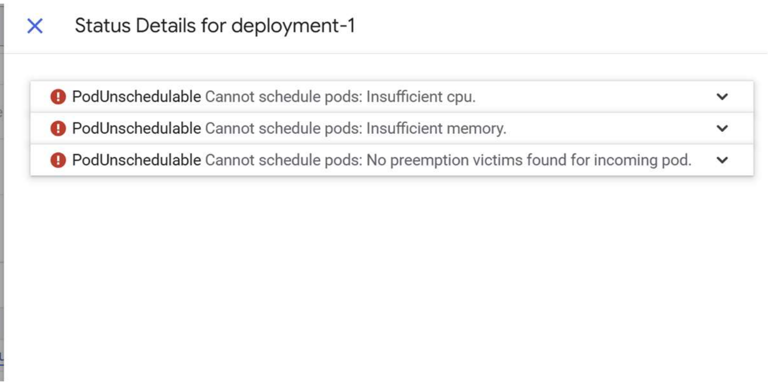
Run a web application behind an external HTTP(S) load balancer by configuring the Ingress resource.

DaemonSet objects

Learn about Kubernetes' DaemonSet objects and their use in GKE to manage

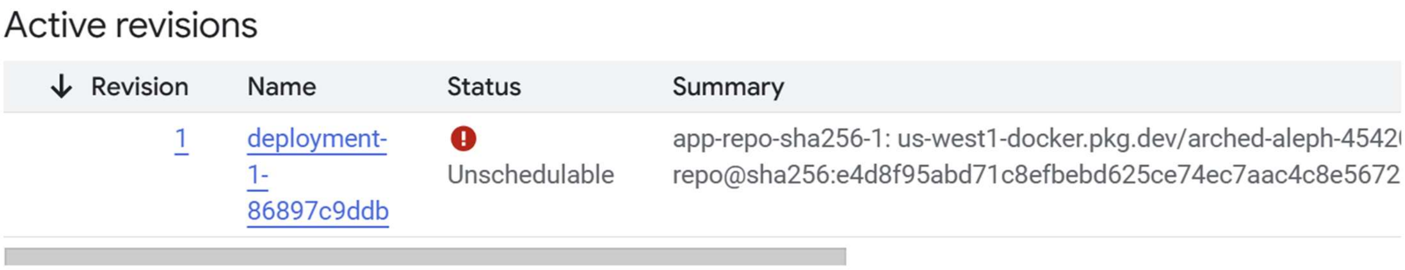


□	Name ↑	Status
□	deployment-1	Does not have minimum availability and 2 more issu

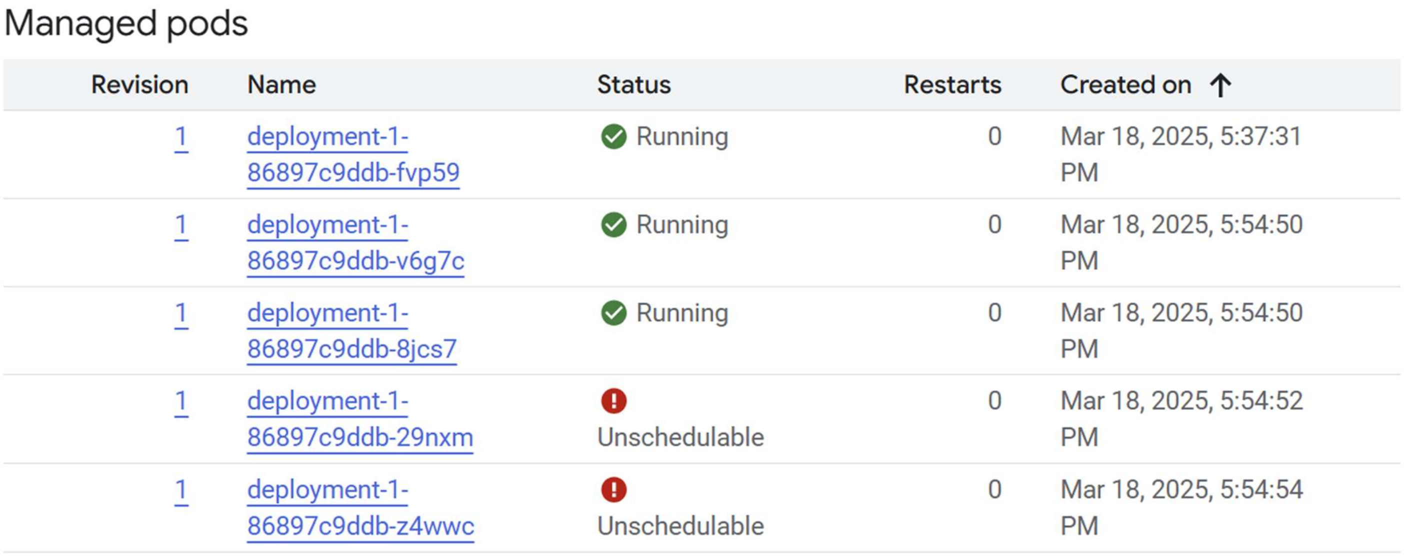

Status Details for deployment-1

- PodUnschedulable Cannot schedule pods: Insufficient cpu.
- PodUnschedulable Cannot schedule pods: Insufficient memory.
- PodUnschedulable Cannot schedule pods: No preemption victims found for incoming pod.

As seen in the screenshots above, I was finally able to get my docker image pushed to the cloud with GKE, but I was getting some errors. Apparently, I did not have sufficient CPU and memory.

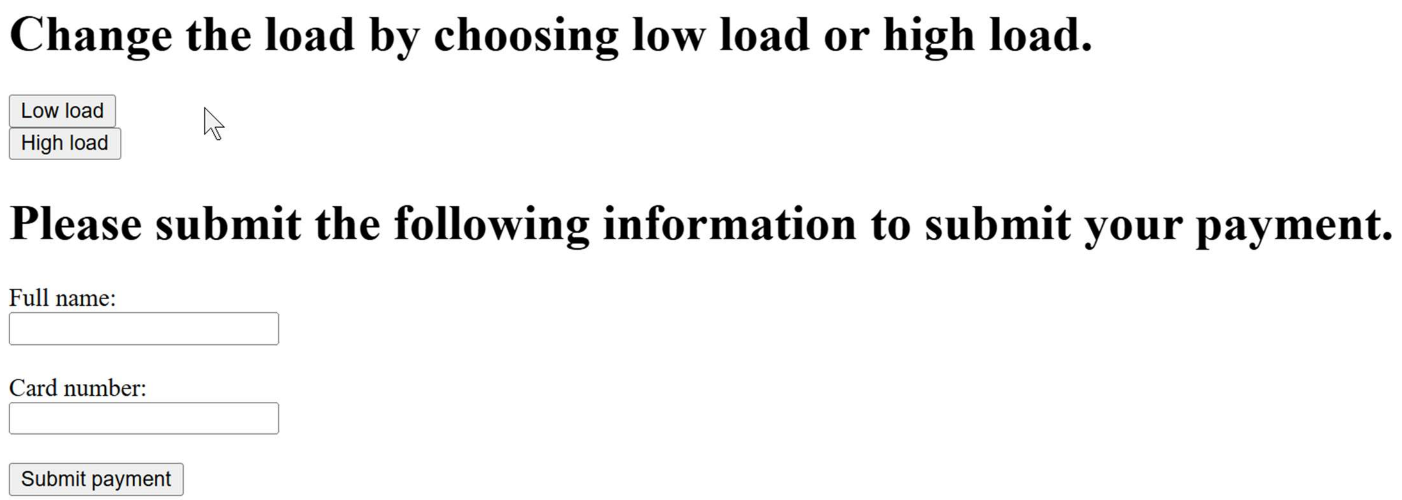

Active revisions

↓ Revision	Name	Status	Summary
1	deployment-1-86897c9ddb	Unschedulable	app-repo-sha256-1: us-west1-docker.pkg.dev/arched-aleph-4542 repo@sha256:e4d8f95abd71c8efbebd625ce74ec7aac4c8e5672


Managed pods

Revision	Name	Status	Restarts	Created on ↑
1	deployment-1-86897c9ddb-fvp59	Running	0	Mar 18, 2025, 5:37:31 PM
1	deployment-1-86897c9ddb-v6g7c	Running	0	Mar 18, 2025, 5:54:50 PM
1	deployment-1-86897c9ddb-8jcs7	Running	0	Mar 18, 2025, 5:54:50 PM
1	deployment-1-86897c9ddb-29nxm	Unschedulable	0	Mar 18, 2025, 5:54:52 PM
1	deployment-1-86897c9ddb-z4wwc	Unschedulable	0	Mar 18, 2025, 5:54:54 PM

As seen in the screenshot above, I was only able to get 3 pods running. I don't know how to get more running.



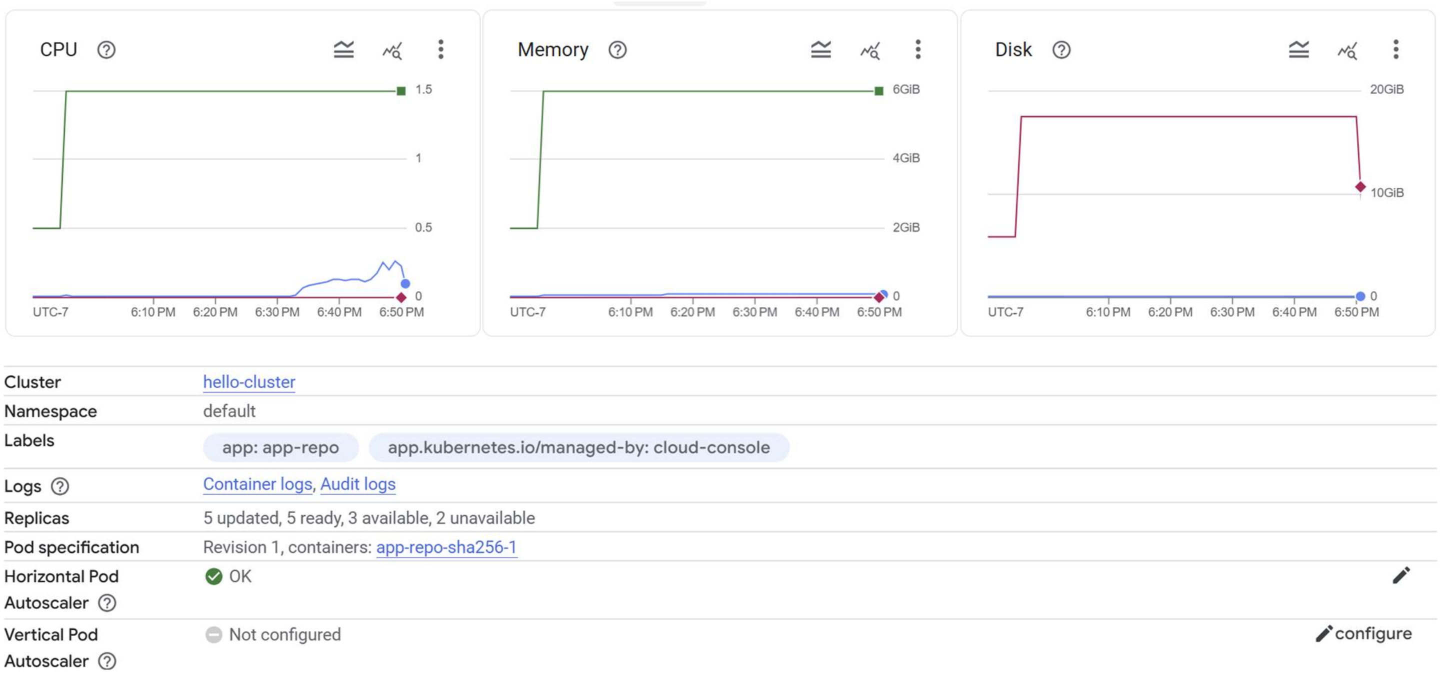
Change the load by choosing low load or high load.

Please submit the following information to submit your payment.

Full name:

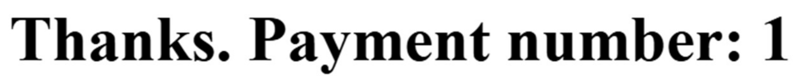
Card number:

As seen in the screenshot above, I created two buttons to adjust the load. The instructions said to "develop some code to increase the demand on the image." With Express JS, I made the "low load" button call a method once every second, while the "high load" button called a method once every millisecond. The method did not contain anything, because frankly, I didn't think it needed to. As long as the assembly code is jumping to that function, that is extra time and resources being used up by the system.



Cluster	hello-cluster
Namespace	default
Labels	app: app-repo app.kubernetes.io/managed-by: cloud-console
Logs	Container logs, Audit logs
Replicas	5 updated, 5 ready, 3 available, 2 unavailable
Pod specification	Revision 1, containers: app-repo-sha256-1
Horizontal Pod Autoscaler	✓ OK
Vertical Pod Autoscaler	Not configured

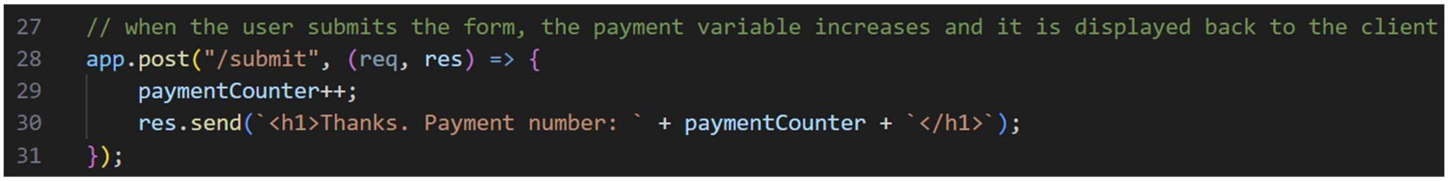
As seen in the screenshot above, I was able to increase the CPU usage by repeatedly clicking the high load and low load buttons one after another, and by entering some PII into the payment form a couple times. I still consider this to be a failure because the increase in CPU usage was not automatic. I was not able to press one button and have the CPU increase and stay at a high level. Maybe I misunderstood the instructions.



Thanks. Payment number: 1

The second half of this simple web application mimics the payment page of the payment processor. The user enters their full name and credit/debit card number. When “submit payment” is pressed, the server increments a variable and displays it on screen. This can be seen in the screenshot above. This number increases every time the submit button is pressed. A downside is that in order to go back to the previous page, the user must manually delete the rightmost subfolder of the HTTP address to get back to the main webpage.

Before I dockerized my app, I had a text file which contained the number of payments as an integer. Every time the user pressed submit, that number would increment by 1. However, when I created a docker image of my app and ran it, the text file did not update for some reason. Because of that, I just kept track of the payment number with a variable in the server file.



```
// when the user submits the form, the payment variable increases and it is displayed back to the client
app.post("/submit", (req, res) => {
    paymentCounter++;
    res.send(`<h1>Thanks. Payment number: ` + paymentCounter + `</h1>`);
});
```

In the server file, when the user clicks the submit button, the payment number is sent back to the client. Traditionally, it’s best practice for the server to send back an HTML file (instead of an HTML element), however dynamically injecting the payment number variable from the text file is not something I know how to do. Because of that, I just put the payment number variable inside an HTML element and let the server return that to the client. This can be seen in the screenshot above.

References

"How to Build a Simple Web Server with Node.js ?" 2021. GeeksforGeeks. June 24, 2021. https://www.geeksforgeeks.org/how-to-build-a-simple-web-server-with-node-js/.

GeeksforGeeks. 2020. "Express.js Res.sendFile() Function." GeeksforGeeks. July 6, 2020. https://www.geeksforgeeks.org/express-js-res-sendfile-function/.

"HTML Forms." 2021. GeeksforGeeks. August 1, 2021. https://www.geeksforgeeks.org/html-forms/.

GeeksforGeeks. 2018. "JavaScript Program to Write Data in a Text File." GeeksforGeeks. October 15, 2018. https://www.geeksforgeeks.org/javascript-program-to-write-data-in-a-text-file/.

GeeksforGeeks. 2022. "How to Dockerize an ExpressJS App ?" GeeksforGeeks. September 16, 2022. https://www.geeksforgeeks.org/how-to-dockerize-an-expressjs-app/.

stursby. 2011. "Javascript Timer Loop." Stack Overflow. December 29, 2011. https://stackoverflow.com/questions/8663246/javascript-timer-loop#8663254.