6/11/2022

S-ESE6

Portfolio v1.0

Beatrice Forslund
SEMESTER 6

Version history

Version	Description	Date
0.1	Setup of document	21/2
0.2	Continue with setting up document	7/3
0.3	Feedback and finish up for first delivery	18/3
0.4	Update in the end of sprint 1	07/4
0.5	Update in the end of sprint 4	27/05
1.0	Update of old sprints with more information, filled in last sprint and conclusions	19/06

Table of Contents

Introduction

I am 22 years old, and I come from Stockholm, Sweden. I did not have any IT or programming classes before I started Fontys, so all my knowledge come from the start of my education here. I did software as the mainstream and then cyber security as a specialization. However, I like software a lot more than cyber security so that is what I am sticking with. I wish to become a better developer this semester and also more knowledge about infrastructure and about microservices. Since I want to possibly work in the finance/bank industry later, I want to also learn about blockchain because that is the most interesting.

This semester I worked on two projects which are the group project and the individual project which includes the emerging trends.

Group project

Fontys Pedagogy is researching possibilities to support visually impaired individuals in relationships and sexuality development which is why they requested an online support community platform where sighted individuals can help the visually impaired by giving advice on e.g., dating. This would be an inclusive solution with a human touch.

Individual project

A group of hotel owners from different countries wants an online application to add their own hotel page/profile and put out available times for their rooms to book their customers on and then see an overview of all scheduled hotel bookings.

The customers want to be able to find a hotel in their area, see their hotel profile/page, book them and then see an overview of the appointment.

Learning outcomes

In this semester we focused on 8 learning outcomes which are:

- Developing and Deployment of Enterprise Software
- Context Based Research
- Preparation for Life-Long Learning
- Scalable Architectures
- Development and Operations (DevOps)
- Cloud Services
- Security by Design
- Distributed Data

For these 8 learning outcomes there are different proficiency levels, 5 levels from Undefined to Advanced.

- Undefined means that not yet show any progression towards the learning outcome.
- Orienting means to explain what the learning outcome is about and applied the acquired knowledge on a basic level.
- Beginning means ability to relate gathered theory relevant to enterprise software but did not apply the theory in a practical context yet within the semester.
- Proficient which is a passing grade for the semester, mastered the learning outcome by demonstrating that you have individually applied the gained knowledge in the context of an enterprise application.
- Advanced means that in several different situations that you deepened your knowledge on the learning outcome beyond the proficient level.

Now in next chapter I am going to discuss the learning outcomes more in detail one by one.

1. Enterprise software development as a team effort.

This learning outcome is about how I perform in a team effort, how we organize the scrum progress, how to communicate to project owner and stakeholders, stand up meetings. Efficient standups, scrum board and everyone knows what everyone has done. How well did we plan the sprints, communicate, how well did we write user stories, how did we do the feedback and sprint demos.

ID	Description	Type	Level
1.1	Sprint 0	Group project	Orienting
1.2	Sprint 1	Group project	Beginning
1.3	Sprint 2	Group project	Beginning
1.4	Sprint 3	Group project	**Proficient**

1.1 Sprint 0

In the first sprint we as a group first prepared questions for the Project owner, after that meeting, we analyzed the wishes and we created three prototypes that we then later showed the stakeholders. After that we also created a survey for the visual impaired people to get a better prioritization of tasks. With all these requirements and wishes we made a backlog and a system overview and user stories and ended the sprint with a sprint delivery.

1.2: Sprint 1

In this sprint, we learned more how to do the standups better with feedback from our teacher. We also managed to create a better backlog with user stories. Our project owner did not show up to our sprint delivery last time, so we sorted this communication issue out with our project owner. As well we also played poker with the sprint board.

1.3 Sprint 2

In this sprint, we got into a routine of doing standups every day like Gerald suggested. We know what everyone is doing, and the teamwork is going good. When some team members have problems, we all help each other out. Rick for example helped me with docker one day. Kevin creates readmes for all the team members to understand the parts of the applications he made. Overall, just going well, we also do not have much of conflicts and difference of opinions. It is easier to agree on things and we all contribute to the conversation, whereas before it was mostly two people discussing things.

1.4 Sprint 3

From last sprint we got feedback how to improve on our group project and that was to “Standups are going well. Sprint review/demo is now at the level of the PO in user stories and giving demo. For proficient: use story points and show that you apply findings from retrospectives in sprints.” So in this sprint we have worked with story points and applied the findings from the retrospectives in sprints and Gerard said in the end of the stakeholder presentation he thinks we are at a proficient level.

2. Conducting context-based research

You deliver professional products according to planning, which are the result of a structured and methodical investigation. You have a critical view towards your own and other people's work, by comparing them to alternatives, judge the structured and methodical approach and consider general accepted and ethical values. Your products are validated with stakeholders and other available research, and you can judge & communicate the relevance and value of the project in its own context.

ID	Description	Type	Level
1.1	Sprint 0	Group project/Individual	Undefined
1.2	Sprint 1	Group project	Orienting
1.3	Sprint 2	Group project	Orienting
1.4	Sprint 3	Group project/Individual	Orienting
1.5	Sprint 4	Group/Individual	**Proficient**

Substantiation

1.2: Sprint 1

In sprint 1, research questions were made first for my individual project as a help to guide the project. I then also made research questions for my emerging trends plan. Finally in the group project we made a design-oriented research plan where we conducted research questions.

For the individual I handed in my Emerging trends plan, and I also got feedback from my semester coach how to improve the research questions and methods. I also started answering the research questions in a research report.

- Emerging trends plan (In the sprint hand in folder)
- Emerging trends report (In the sprint hand in folder)

1.3 Sprint 2

We created research questions and updated them after feedback and I have also started to answer the questions and have now answered two questions.

- DevOps research questions

1.4 Sprint 3

In this sprint I have continuing researching and keep answering questions. For now, neither of them are done yet.

1.5 Sprint 4

In this sprint I finished the emerging trends research as well as in the group project. This mean my level is proficient.

- Emerging trends report (Handed in canvas)
- DevOps Research report (Handed in canvas)

3. Preparing for lifelong learning

You acquire skills required for your future career. You are aware of multiple career paths and can reflect which ones fit best, considering your (potential) skills and ambitions. You are aware of developments in software engineering and can signal trends.

ID	Description	Type	Level
1.1	Sprint 0	Individual	Orienting
1.2	Sprint 1	Individual	Beginning
1.3	Sprint 3	Individual	**Proficient**

1.1 Sprint 0

In this sprint I decided which emerging trends I will do, and I chose to do blockchain because I would maybe want to work in the finance industry later.

1.2 Sprint 1

In this sprint I made a graduation plan, which was a plan about finding a future internship and what I want to work in a carrier later. I also started answering emerging trends research questions and I still find it interesting as a future internship.

- Graduation plan
- Sign up for my minor

1.3 Sprint 3

In the beginning of the sprint, I asked my semester coach what I can do to improve this learning outcome to a proficient. I realized I do more than what I have written down. This is what I have done:

- Study abroad Week to Hasselt, Belgium
- Carrier day
- Dotnet Saturday event (https://dotnedsaturday.nl/)
- Learning Dutch to get a job in Netherlands

These are all the things I am doing to explore what I want to do for my future. The study abroad week I learnt that I enjoy working with IT and more of automotive engineering, and on the carrier day I talked to some interesting companies to have my future internship with and then the Dotnet event is to learn more about the programming language I want to do in the future, and I also learn Dutch to get a job in the future.

Me, at the conference!

4. Scalable architectures

Besides functionality, you develop the architecture of enterprise software based on quality attributes. You especially consider attributes most relevant to enterprise contexts with high volume data and events. You design your architecture with future adaptation in mind. Your development environment supports this by being able to independently deploy and monitor the running parts of your application.

ID	Description	Type	Level
1.1	Sprint 0	Group project/Individual	Orienting
1.2	Sprint 1	Individual	Beginning
1.3	Sprint 2	Group project	Beginning
1.4	Sprint 3	Individual/Group	Proficient
1.5	Sprint 4	Individual/Group	Proficient
1.6	Sprint 5	Individual	**Advanced**

1.1 Sprint 0

I started the sprint by creating an architecture diagram specifying the division of responsibilities of the different microservices we envisioned in the group project. For my individual project, I created an architecture design and started on a skeleton for the backend services. At the end of the sprint, I successfully implemented two microservices that communicate through a RabbitMQ message bus.

1.2 Sprint 1

In sprint 1 I learned how to deploy my applications on a local k3 Kubernetes cluster installed on my machine. I focused on making my application scalable which was made possible by RabbitMQ. For the architecture, I am going for an event-drive state transfer.
I also learned about the differences between Kafka and RabbitMQ.
Summary of what I have done for this learning outcome:

- Microservices & Event-driven state transfer architecture
- RabbitMQ message bus
- Kubernetes
- Technical document

Microservices & Event-driven state transfer architecture

When researching architectural styles, I learned that it is not only possible to choose one fitting architecture but that it can also be a valid choice to mix architectural styles. I decided to go with Microservices, Event-driven architecture.

Microservice Architecture

Microservices make it possible to easily scale up and down smaller parts of your application in order to make the most efficient use of your computer resources.

A large program is broken down into smaller components that can operate independently of one another when using a microservice architecture. Other machines can pick up work without adding to the burden on a machine that is already in high resource use without having to reserve resources for a complete copy of the whole application. Another benefit is fault built-in isolation. If something goes wrong, most of the system can usually continue to function while the malfunctioning component is being restored.

Event-driven Architecture

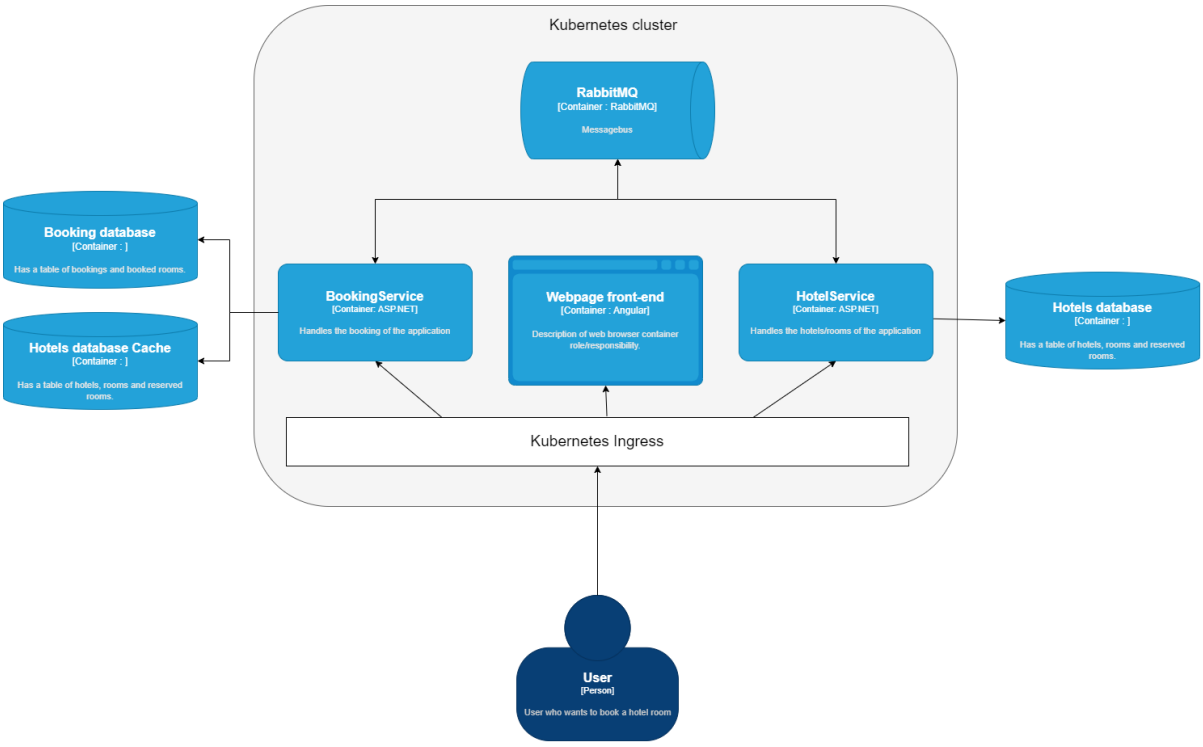
Event-driven architecture means that we have producers that send out event data and consumers which listen for specific events relevant to that consumer. Within the event-driven design, you have a choice of the event-notification pattern or event-carried state transfer.

In the event-notification pattern, most events only contain a unique identifier and some useful metadata containing methods to retrieve more information. Whenever a consumer receives an event-notification the consumer can choose to retrieve more information about the event or state by using the metadata information to query the service that made the state change.

In the event carried state transfer the services publish information about the state changes in the event. Services can copy the relevant state change and save this in a local cache of the state so they can function independently of the producing service. The benefit for services to keep a local copy of the data is that then they are more self-sustainable and if a part of the application gets overloaded, the service does not need to call for the data to the other service but can use its own local cache data instead. This makes it simpler to scale up specific microservices as they are not dependent on other microservices being available.

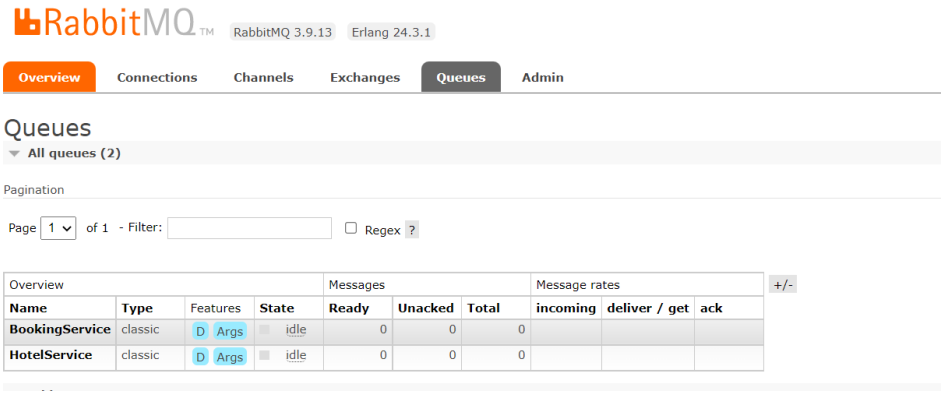
This is significantly different from event-notification where the producing microservices might have to be scaled up with the consuming microservices because scaling up the consuming microservices will add load to the producing microservices.

In my sketch, the design is using event-state transfer. Here the room is the state, and that changes based on whether it is booked or not. Later, more services can be added to handle if the room has been checked in, for example, or checked out.



C4 Model of architecture

RabbitMQ message bus



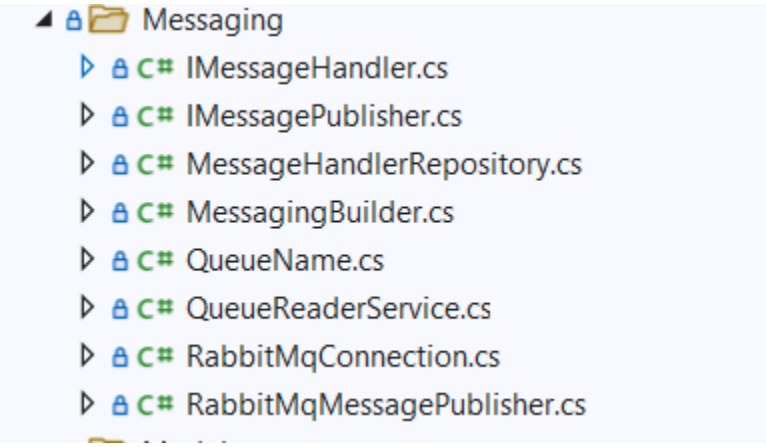
When deciding which messaging system to use, I decided on RabbitMQ and neglected Kafka, the reason being that it seemed to appeal the most to me.

In my application, I have two services that communicate with each other. The BookingService handles the bookings and HotelService the hotels and rooms. This is the flow which is also called Event-driven state transfer:

1. When a user creates a booking, it checks availability in the BookingService which will based on availability return true or false back to the user.
2. If the booking was successful, it publishes that a new booking has been made onto the RabbitMQ queue.
3. The HotelService picks the booking from the queue, checks availability and adds it to the events of bookings.
4. When the booking was successful on HotelService as well, it confirms it the booking and publishes a confirm message onto the queue.
5. BookingService listens to the queue, picks up the confirmed booking and confirms the booking in the booking table.
6. Front-end is waiting for the booking to be confirmed and has a loading screen for the user.

The RabbitMQ is also used for syncing of the hotel and rooms, so when a hotel or room was added, it also is added to BookingService database, meaning both services have duplicate data. This is because caching stores data closer to end-users. It's an excellent way to facilitate the return of data that is needed often but changes rarely.

The BookingService and HotelService have both the same messaging setup:

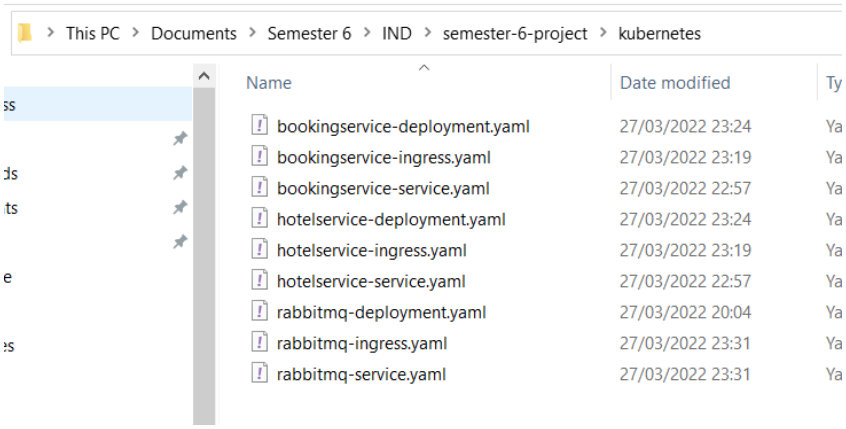


Kubernetes

The reason for Kubernetes is that it makes applications to be more secure, up-to-date, and available 24 hours a day. To make this possible, developers need to be able to roll out new updated versions quickly and easily and this is made easy to update and release with Kubernetes.

Kubernetes helps also makes it possible to ensure that these containerized applications are running, keeping track of where they are running and when they are running.

There are many reasons to use Kubernetes. It is reducing costs and saving time, portability, stability and security, scalability, and creating a base for cloud-native applications are some good arguments and the reason for me using it.



Kubernetes

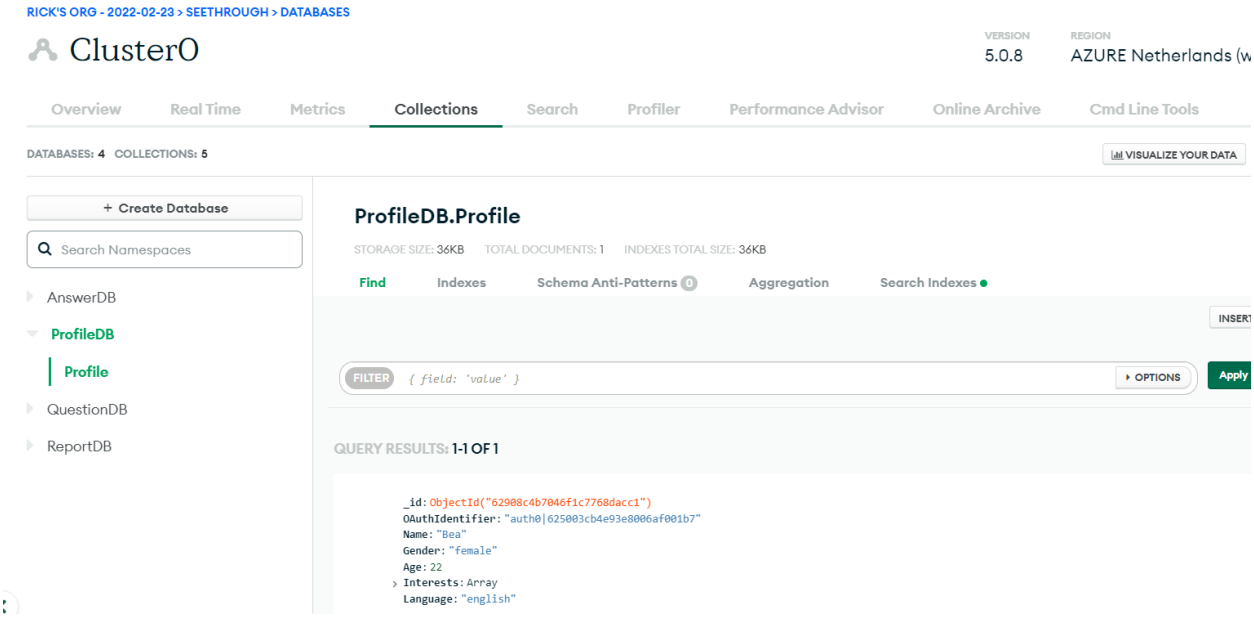
1.3 Sprint 2

In the group project, we got the advice from Marcel to start creating more microservices. I would like event-driven state transfer as well in this project, but the Team thinks otherwise. But anyways, we created many different services and started to connect them and soon also a RabbitMQ and we are also using docker-compose with a MongoDB database.

- Microservices
- MongoDB
- Docker compose
- RabbitMQ

MongoDB

The reason our group decided on a NoSQL database is that it is easier to scale up the database if needed and that was something we valued high. It is also located in the cloud which means we do not need to build and run the image in our local computer which saves CPU memory, and it is also then a lot faster.



1.4 Sprint 3

In this sprint, I tested my individual application if it is possible for it to scale, how the microservices and RabbitMQ behave if they go down, and more. The result of these can be read more about in my testing report that I created where I also show how I tested it.

Finally this sprint, I also added a scalable database in Azure which means the application is now more scalable.

- Testing the scalability
- Scalable Database
- Test report
- Kubernetes and RabbitMQ in the group project

```
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\kubernetes> kubectl apply -f bookingservice-deployment.yaml
deployment.apps/bookingservice configured
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\kubernetes> kubectl get pods
NAME                              READY   STATUS    RESTARTS        AGE
rabbitmq-0                        1/1     Running   5 (6m25s ago)   31d
bookingservice-5d76456577-zlf4s   1/1     Running   1 (6m36s ago)   2d22h
hotelservice-66dd55b85-mpkmg      1/1     Running   0               3m49s
bookingservice-5d76456577-c9jl9   1/1     Running   0               11s
bookingservice-5d76456577-zmf49   1/1     Running   0               11s
bookingservice-5d76456577-tskh4   1/1     Running   0               11s
bookingservice-5d76456577-8qggw   1/1     Running   0               11s
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\kubernetes>
```

Adding more replicas in Kubernetes

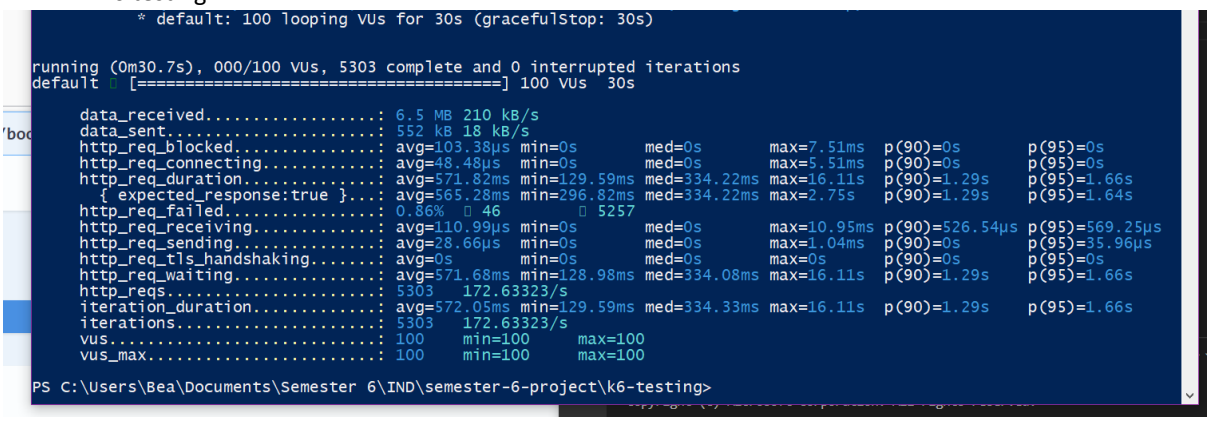
Since it is scalable, I believe I have proficient in the learning outcome. In the group project, I also added RabbitMQ and Kubernetes as well.

Not only did I test the scalability, but I tested several things. For example, also if a microservice goes down, will it pick up the RabbitMQ message from the queue? These are the topics from the test report document:

1.5 Sprint 4

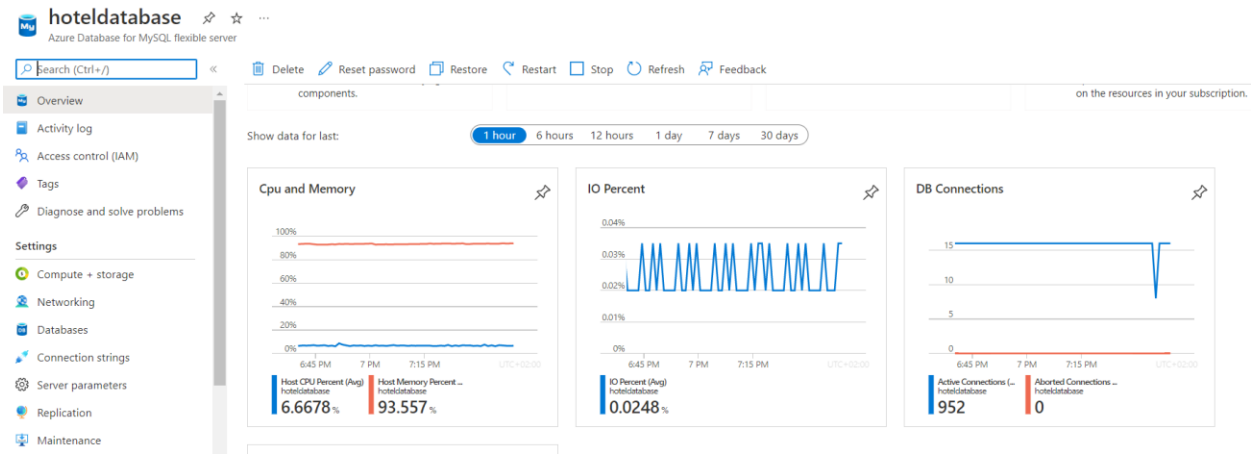
In this sprint, I started with K6 load testing. I also deployed my project to the cloud which means I can now monitor the application.

- K6 testing



For this is tried load testing on the database I have in Azure, simply by calling a GET request. For this, I tried 100 users with a 30-sec pause in between. The results from k6 testing showed that 0.86% of the requests failed, this is a quite big number. For my application I want to have at least 100 000 users, this would mean 1000 requests would fail which is a lot.

The reason for the limited numbers is because of the database, if I want it to have a lower time-out request, I need to upgrade the database and make it support more requests, but since I am on a budget this is not possible.



In this picture, it is possible to see that the Memory of the database is almost at 100%.

1.6 Final Sprint

In the previous sprint, I found out that the free azure MySQL I am using has resource limitations that affect my load/scalability tests. Since the database is cloud-hosted and can easily be scaled up after providing payment information, I decided to remove the database from my performance tests. To test the scalability of my application and to improve the scalability of the solution at the same time I decided to implement automatic horizontal scaling. I learned that in Kubernetes this can be done with what is called a HorizontalPodAutoscaler configuration. I created a hpa that scales up the application when the CPU is over a certain limit.

Horizonal Autoscaling

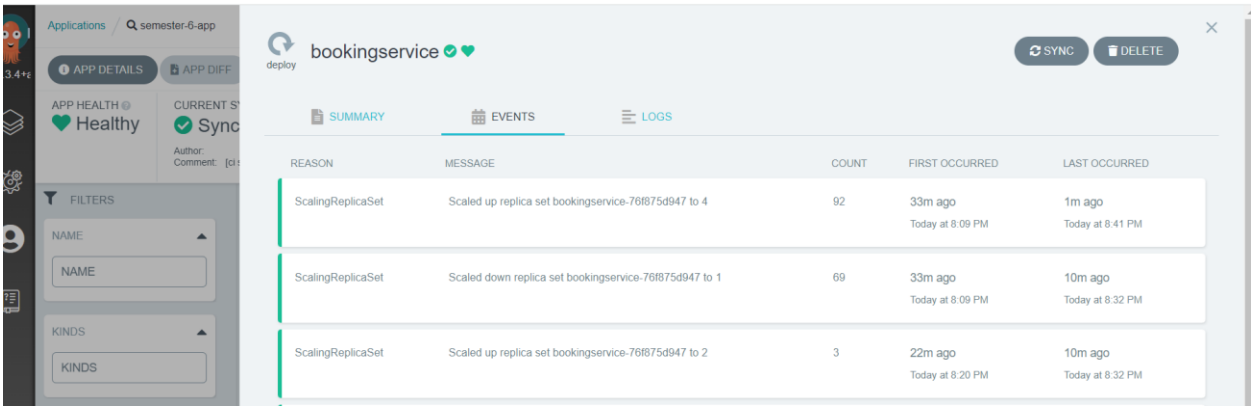
```
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\k6-testing> kubectl get horizontalpodautoscaler
NAME             REFERENCE                      TARGETS         MINPODS   MAXPODS   REPLICAS   AGE
bookingservice   Deployment/bookingservice      <unknown>/50%   1         10        1          86s
hotelservice     Deployment/hotelservice        <unknown>/50%   1         10        1          9m14s
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\k6-testing>
```

```
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\k6-testing> kubectl get pods
NAME                                        READY   STATUS      RESTARTS   AGE
bookingservice-76f875d947-2lxfd             1/1     Running     0          6m12s
bookingservice-76f875d947-gb8s4             1/1     Running     0          1s
bookingservice-76f875d947-l4lzd             1/1     Running     0          1s
bookingservice-76f875d947-znjxd             1/1     Running     0          1s
default-cert-manager-cainjector-7ff875db78-r78rr   1/1     Running     0          3d3h
default-cert-manager-controller-7c9996d6c7-dlg8n   1/1     Running     0          3d3h
default-cert-manager-webhook-85dbb5f7cc-2bv7v      1/1     Running     0          3d3h
hotelservice-56c74b7868-x765h               1/1     Running     0          6m12s
hotelservice-7d45c7bbbc-rk85m               0/1     Completed   9          3d3h
influxdb-deployment-d64d8b97b-mx4c5         1/1     Running     0          70m
rabbitmq-0                                  1/1     Running     0          3d3h
```

```
kubernetes > ! bookingservice-hpa.yaml
1  apiVersion: autoscaling/v1
2  kind: HorizontalPodAutoscaler
3  metadata:
4    name: bookingservice
5  spec:
6    scaleTargetRef:
7      apiVersion: apps/v1
8      kind: Deployment
9      name: bookingservice
10   minReplicas: 1
11   maxReplicas: 10
12   targetCPUUtilizationPer
```

```
running (0m30.7s), 000/250 VUs, 2437 complete and 0 interrupted iterations
default ✓ [==================================] 250 VUs  30s

     data_received..................: 1.3 MB 42 kB/s
     data_sent......................: 238 kB 7.7 kB/s
     http_req_blocked...............: avg=56.08ms  min=0s       med=0s     max=948.23ms p(90)=147.59ms p(95)=494.51ms
     http_req_connecting............: avg=2.84ms   min=0s       med=0s     max=37.61ms  p(90)=18.88ms  p(95)=28.14ms
     http_req_duration..............: avg=3.04s    min=962.71ms med=2.98s  max=5.31s    p(90)=3.93s    p(95)=4.41s
       { expected_response:true }...: avg=3.04s    min=962.71ms med=2.98s  max=5.31s    p(90)=3.93s    p(95)=4.41s
     http_req_failed................: 0.00%  ✓ 0         ✗ 2437
     http_req_receiving.............: avg=1.27ms   min=0s       med=0s     max=25.68ms  p(90)=4.51ms   p(95)=7.43ms
     http_req_sending...............: avg=258.67µs min=0s       med=0s     max=13.27ms  p(90)=572.2µs  p(95)=1ms
     http_req_tls_handshaking.......: avg=45.73ms  min=0s       med=0s     max=835.32ms p(90)=63.51ms  p(95)=395.18ms
```

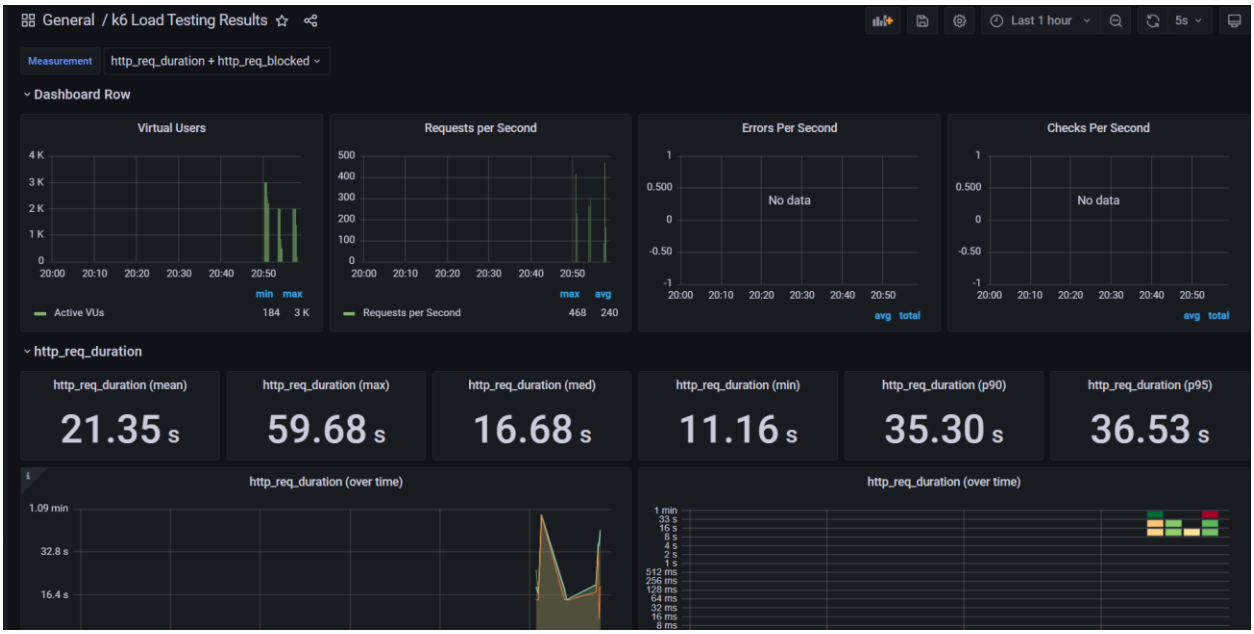
When I then tested the HTTP endpoint again with 250 users the HTTP request failed was zero and when looking at the pods of the BookingService I saw that the hpa did its job and scaled up automatically.



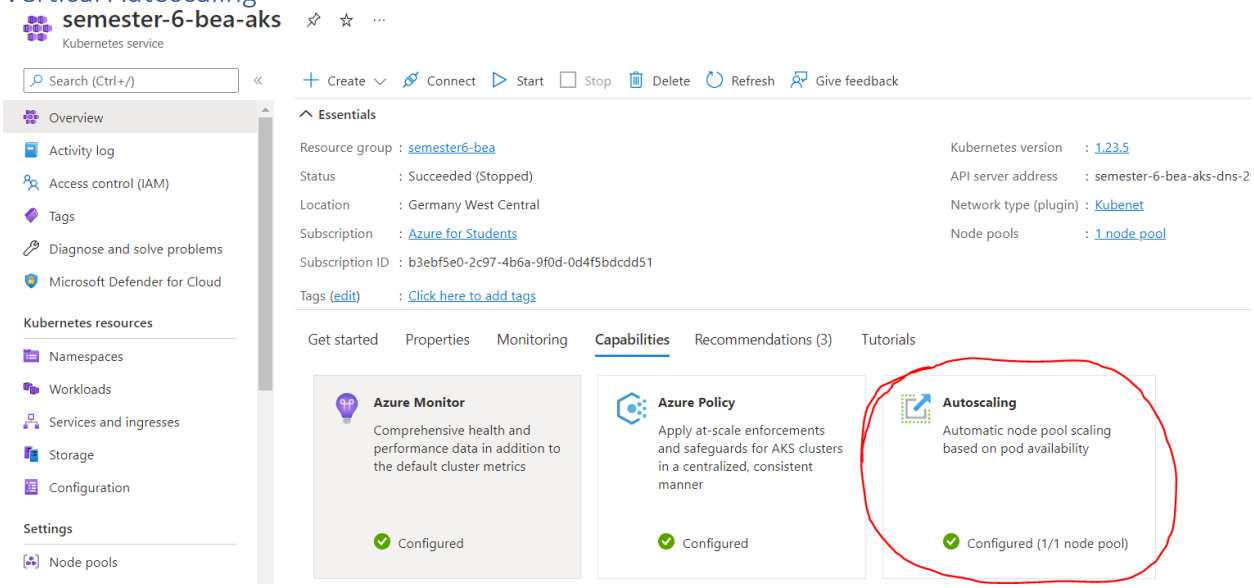
It was also possible to read in ArgoCD that it auto-scaled up the pods to be able to handle the extra load, this means that the implementation to auto-scale was successful. I tried with more users and with 1000 users we see no failing requests.

```
running (0m33.6s), 0000/1000 VUs, 2071 complete and 0 interrupted iterations
default ✓ [======================================] 1000 VUs  30s

     data_received..................: 3.9 MB 115 kB/s
     data_sent......................: 674 kB 20 kB/s
     http_req_blocked...............: avg=507.02ms min=0s    med=0s     max=2.55s    p(90)=1.64s    p(95)=2.04s
     http_req_connecting............: avg=44.6ms   min=0s    med=0s     max=150.78ms p(90)=120.92ms p(95)=128.3ms
     http_req_duration..............: avg=15.53s   min=3.73s med=15.98s max=30.8s    p(90)=25.24s   p(95)=25.46s
       { expected_response:true }...: avg=15.53s   min=3.73s med=15.98s max=30.8s    p(90)=25.24s   p(95)=25.46s
     http_req_failed................: 0.00%  ✓ 0        ✗ 2071
```

To be able to view and read the load test better and have a cool overview I added them into Grafana:



Vertical Autoscaling

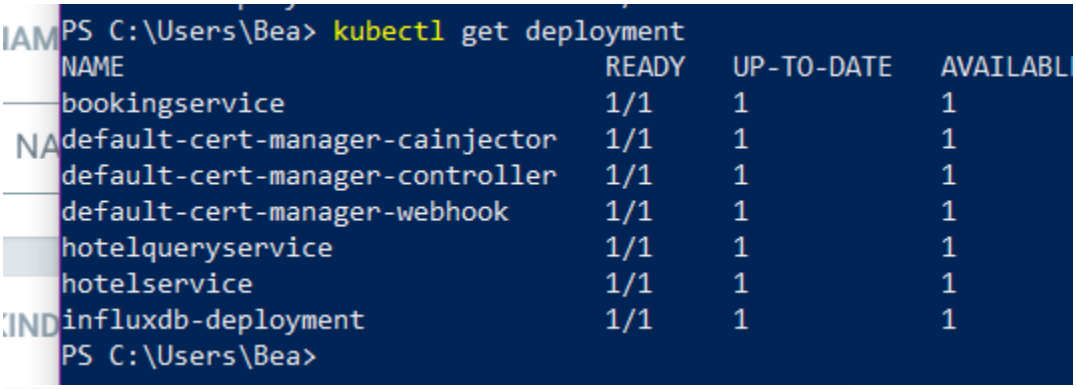


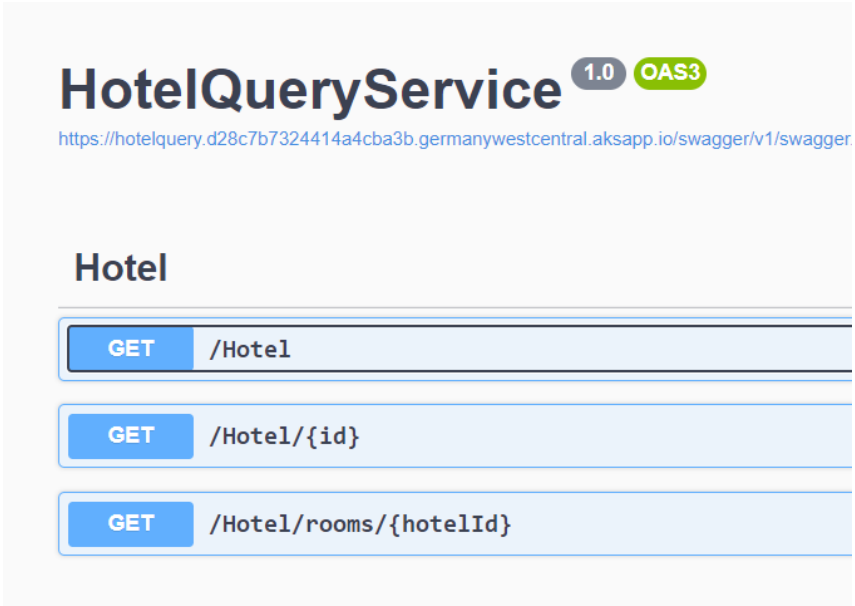
After looking at my Kubernetes service on Azure, I realized it had an option to do vertical auto-scaling as well. This scales up the node pool when a pod cannot be scheduled due to resource constraints. However, since I am using a student version the maximum it could auto-scale up to was 2 nodes.

CQRS pattern

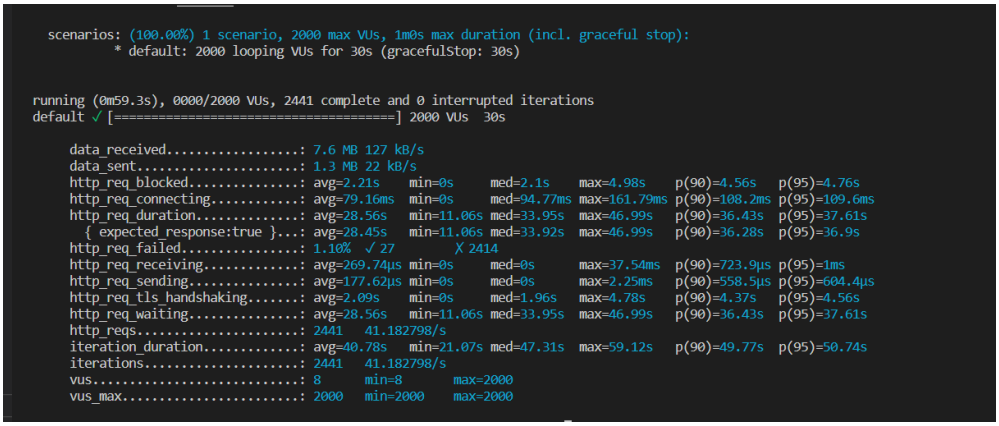
To get another level of scalable I decided to do CQRS (Command Query Responsibility Segregation), which means separating the Queries and Commands; the GET requests, and the POST requests.

The idea behind CQRS is that for example, customers can spend hours just browsing hotels without actually making a booking. Because most customers are just browsing there is a heavier load on the GET requests than the POST/PUT requests. If we would not use CQRS, and the services thus contain both query endpoints and command endpoints, we cannot scale up the query performance without also scaling up the command performance. Since there is no need to scale up the command performance this is a waste of resources. In short:
CQRS: Independent scaling. CQRS allows the read and write workloads to scale independently and may result in fewer lock contentions. This also means optimized data schemas. The read side can use a schema that is optimized for queries, while the write side uses a schema that is optimized for updates.
To implement CQRS I split the hotel service into a HotelService for handling commands and a hotelqueryservice for handling queries.





Redis



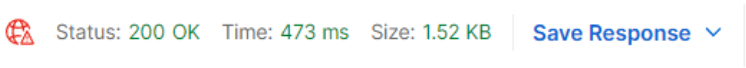
```
  scenarios: (100.00%) 1 scenario, 2000 max VUs, 1m0s max duration (incl. graceful stop):
           * default: 2000 looping VUs for 30s (gracefulStop: 30s)


running (0m59.3s), 0000/2000 VUs, 2441 complete and 0 interrupted iterations
default ✓ [======================================] 2000 VUs  30s

     data_received..................: 7.6 MB 127 kB/s
     data_sent......................: 1.3 MB 22 kB/s
     http_req_blocked...............: avg=2.21s    min=0s     med=2.1s    max=4.98s    p(90)=4.56s   p(95)=4.76s
     http_req_connecting............: avg=79.16ms  min=0s     med=94.77ms max=161.79ms p(90)=108.2ms p(95)=109.6ms
     http_req_duration..............: avg=28.56s   min=11.06s med=33.95s  max=46.99s   p(90)=36.43s  p(95)=37.61s
       { expected_response:true }...: avg=28.45s   min=11.06s med=33.92s  max=46.99s   p(90)=36.28s  p(95)=36.9s
     http_req_failed................: 1.10%  ✓ 27        ✗ 2414
     http_req_receiving.............: avg=269.74µs min=0s     med=0s      max=37.54ms  p(90)=723.9µs p(95)=1ms
     http_req_sending...............: avg=177.62µs min=0s     med=0s      max=2.25ms   p(90)=558.5µs p(95)=604.4µs
     http_req_tls_handshaking.......: avg=2.09s    min=0s     med=1.96s   max=4.78s    p(90)=4.37s   p(95)=4.56s
     http_req_waiting...............: avg=28.56s   min=11.06s med=33.95s  max=46.99s   p(90)=36.43s  p(95)=37.61s
     http_reqs......................: 2441   41.182798/s
     iteration_duration.............: avg=40.78s   min=21.07s med=47.31s  max=59.12s   p(90)=49.77s  p(95)=50.74s
     iterations.....................: 2441   41.182798/s
     vus............................: 8      min=8       max=2000
     vus_max........................: 2000   min=2000    max=2000
```

I decided to perform more load tests after implementing CQRS and scaling up the database. The results were looking good, but I still had problems with the database and requests were failing because of timeout. I discussed with my team members and my teachers about this when they mentioned using Redis could solve my problem. So, I implemented Redis into the HotelQueryService. This is also relatable to scalability because one of the bottlenecks to avoid with scalability is:
Caching stores data closer to end-users. It's an excellent way to facilitate the return of data that is needed often but changes rarely.

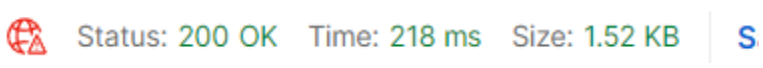
This is how I changed my controller:

```
22      [HttpGet]
        0 references
23      public async Task<ActionResult> GetAllAsync()
24      {
25          string recordKey = "Hotels_" + DateTime.Now.ToString("yyyyMMdd_hhmm");
26
27          var hotelsCache = await _cache.GetRecordAsync<Hotel[]>(recordKey);
28          if(hotelsCache == null)
29          {
30              var hotels = await _context.Hotels.Include(h => h.Rooms).ToListAsync();
31              await _cache.SetRecordAsync(recordKey, hotels);
32              return Ok(hotels);
33
34          }
35          else
36          {
37              return Ok(hotelsCache);
38          }
39
40
```

I ask if the Redis cache DB has any hotels, if not then ask the real DB for the hotels. The cache is also reset every 60 seconds, I think that is a good time.



When calling the database, the time is almost 500 milliseconds.



But after using Redis cache, the time it takes to request for data is now halved, so now it is a lot faster!

1.7 Scalable Architecture LO Conclusions

Scalability refers to a system's ability to handle changes in the workload without affecting the performance. Three things that a system needs to be scalable are **Availability**, **Performance,** and **Reliability**.

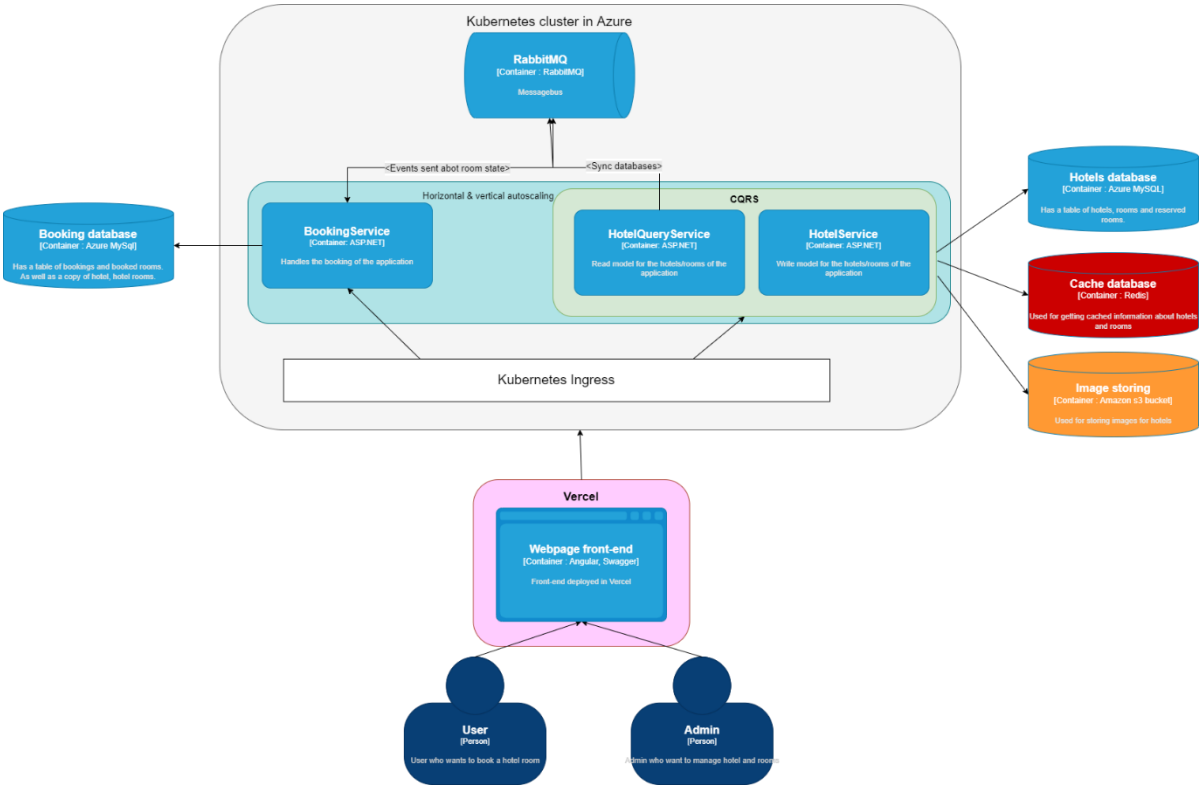
I have applied what I learned for this learning outcome to my individual application and the group project.
In my individual application:

- Microservices with async calls
- RabbitMQ with pub/sub
- Event-driven state transfer
- Kubernetes with ingress controller
- Tested the scalability
- Horizontal Auto scaling
- Vertical Auto scaling
- CQRS pattern
- Redis

In the group project:

- Cross-Platform Microservices (Both in Java and .NET)
- RabbitMQ with pub/sub (Java and .NET)
- Choosing a scalable DB (MongoDB)
- Kubernetes (adding RabbitMQ to Kubernetes cluster)

Since I applied my knowledge in multiple situations in both the individual and the group projects, I believe I have the learning outcome of an **advanced** in this learning outcome.



Final architecture diagram of my individual project

5. Development & Operations (DevOps)

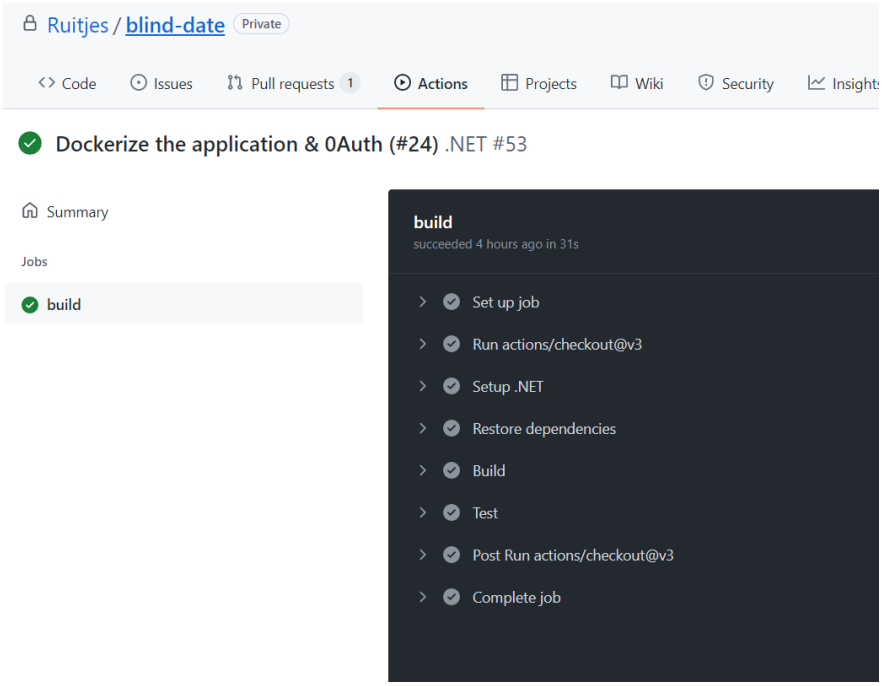
You set up environments and tools which support your chosen software development process. You provide governance for all stakeholders' goals. You aim for as much automation as possible, to enable short release times and high software quality.

ID	Description	Type	Level
1.1	Sprint 2	Group project	Beginning
1.2	Sprint 3	Group project/Individual	Proficient
1.3	Sprint 4	Group project/Individual	Proficient
1.4	Sprint 5	Individual	**Advanced**
1.5	Conclusions	Individual + Group project	**Advanced**

1.1 Sprint 2

In this sprint for the group project, I have started answering research questions for the DevOps research. I also created a successful pipeline in the group project, which now builds and tests the code. For the pipeline, I had to create a test, which means now that the group can start creating unit tests.

- Research and answer research questions about DevOps
- CI pipeline



1.2 Sprint 3

In this sprint, I worked on the DevOps research questions for the group project. This gave me a good understanding of DevOps and the CICD pipeline.

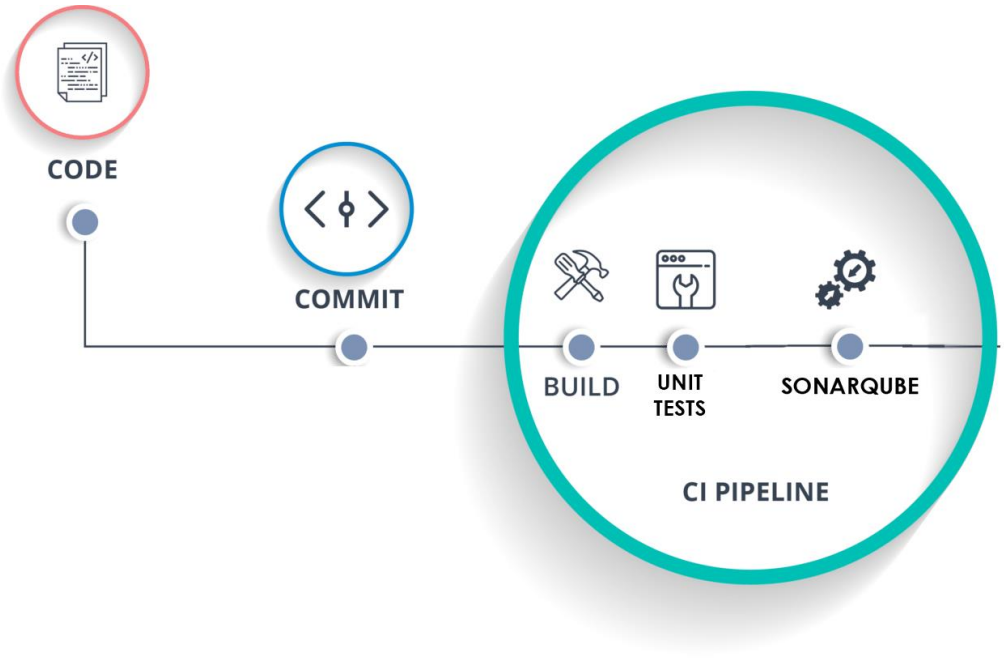
I also added sonarqube to my pipeline to check the code. Then I also followed the DevOps workshop that was provided by Fontys.

For the group project, we are using GitHub, the reason was that our group wanted the projects to be seen by the public and maybe future employers. For my individual project, I am using Gitlab, because I like the environment and I have previous knowledge about it. This also means I learn how to work with them both.

- Research questions for Design Oriented Research
- CI/CD pipeline
- Separate git environments, and learn more about the difference (GitLab & GitHub)

CI pipeline

For the application to have good quality an initial CI pipeline was made. And the reason for this is so that software can be delivered faster and without the quality going down. The ci/cd pipeline will test the application before deploying it so that all the tests are passing, and the application works as supposed to. If there is a test that is failing, the bug is easily seen and can be fixed compared to when after it has been deployed. The pipeline is also using SonarQube because SonarQube ensures security, and it is possible to check for code coverage it is a great tool to use for a good quality application.



Current CI/CD pipeline

1.3 Sprint 4

In this sprint, I deployed my application to Azure and for that, I used Azure Kubernetes and ArgoCD. In the group project, we also used ArgoCD. In the group project, we use Google Cloud Kubernetes instead of Azure Kubernetes so I am learning about the different features between Google Cloud Kubernetes and Azure Kubernetes.

Group:

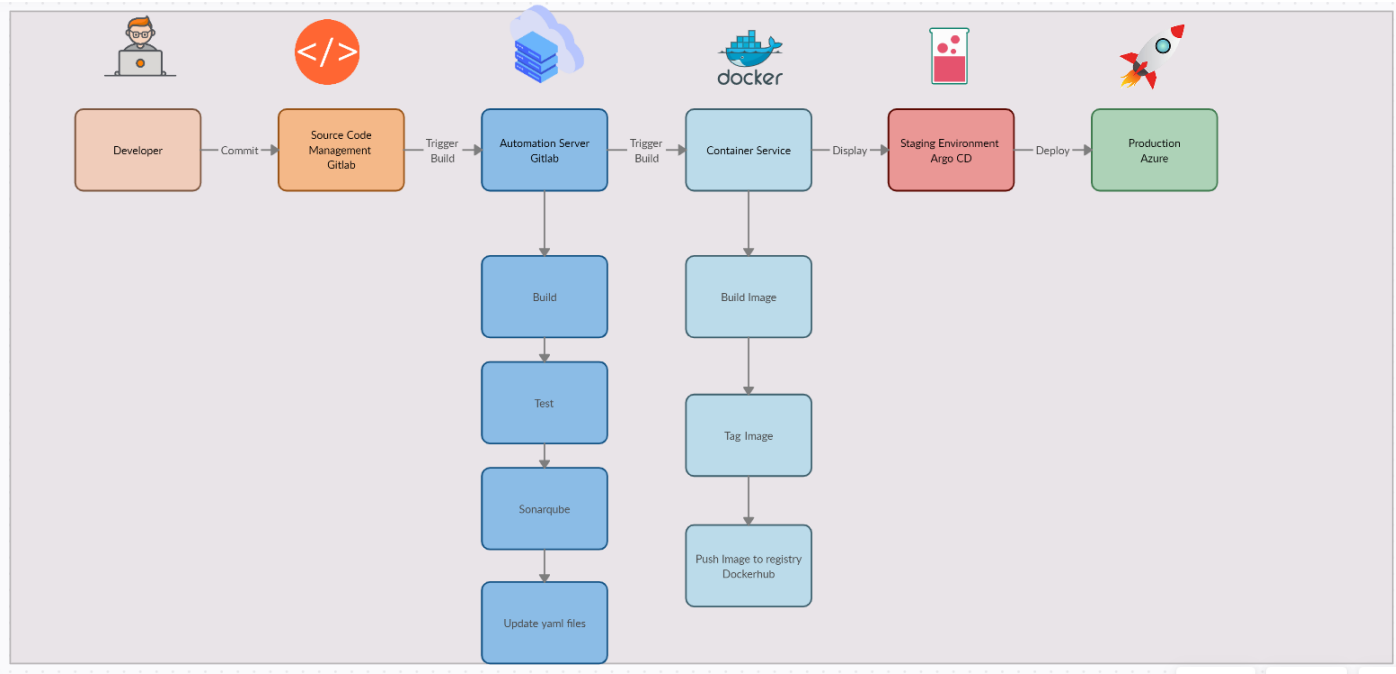
- ArgoCD
- Deployment to Google cloud

Individual:

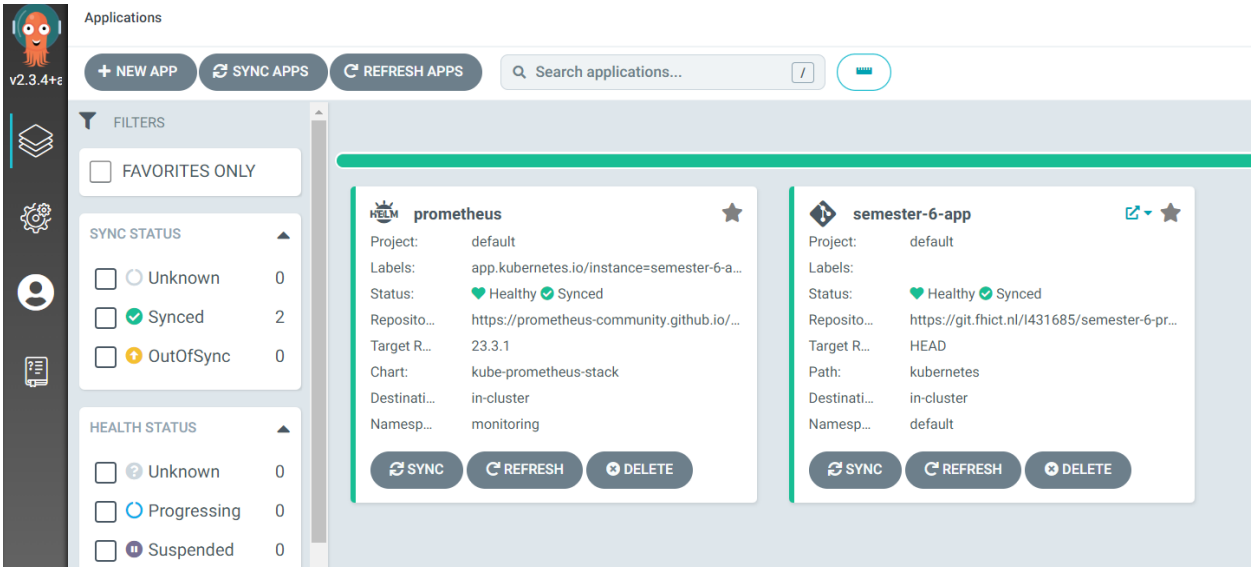
- ArgoCD
- Deployment to Azure

ArgoCD & CI/CD pipeline

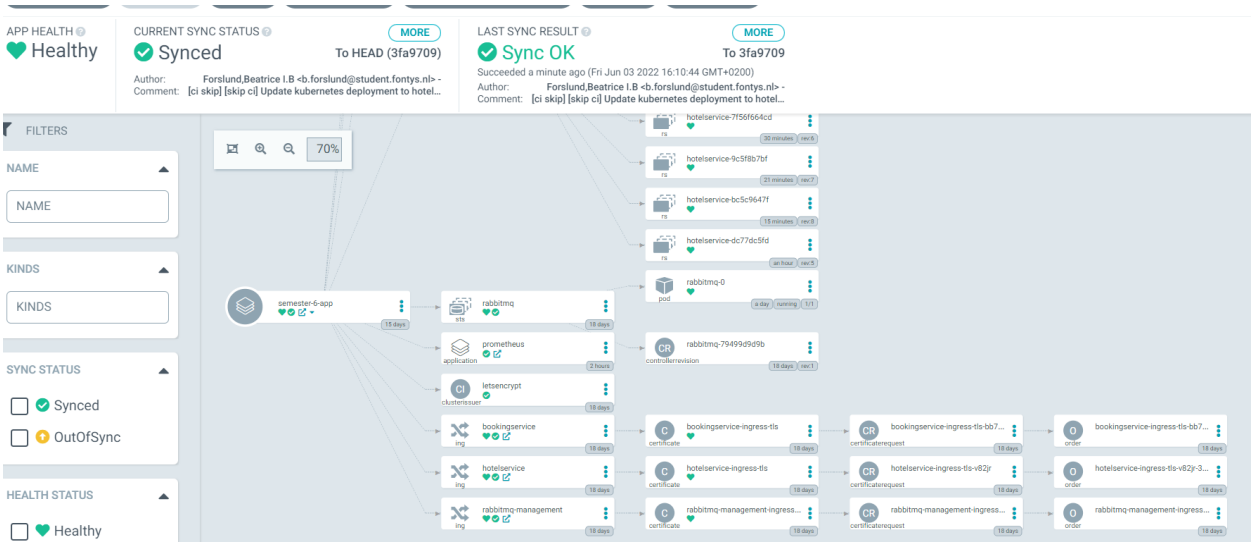
We used Argo in the group project, and I used it in my individual project because it is a popular and new tool that is used in big companies. It is also very simple to set up, and it automates the CI/CD pipeline which I wanted for my projects. ArgoCD works so that what is in git is synced to Kubernetes, so if I push it to my git repository it updates it to Kubernetes. It does that by checking if there are any changes in the YAML files. This means that every time I push an application change to git I also need to update my container image tag. I then need to update the container image version used in Kubernetes. This is a diagram of the final pipeline:



This pipeline pushes to Docker Hub and then updates the YAML file with the new image tag which triggers ArgoCD to push to my Azure Kubernetes service. This makes it possible to have the newest update on Kubernetes whenever I push to my GitLab and thus have continues deploy.



ArgoCD frontpage



Small part of ArgoCD and all pods, deployment ingress etc

1.4 Final Sprint

My goal with the final sprint was to prove that I have improved from a Beginning to an Advanced grading in the DevOps learning outcome.

Monitoring

My teacher gave me feedback that I needed to add some monitoring to my application. When talking with my team members about what they used for monitoring they all said Kubernetes Dashboard. However, in my opinion, it is not really monitoring but rather a simple management UI on top of Kubernetes.

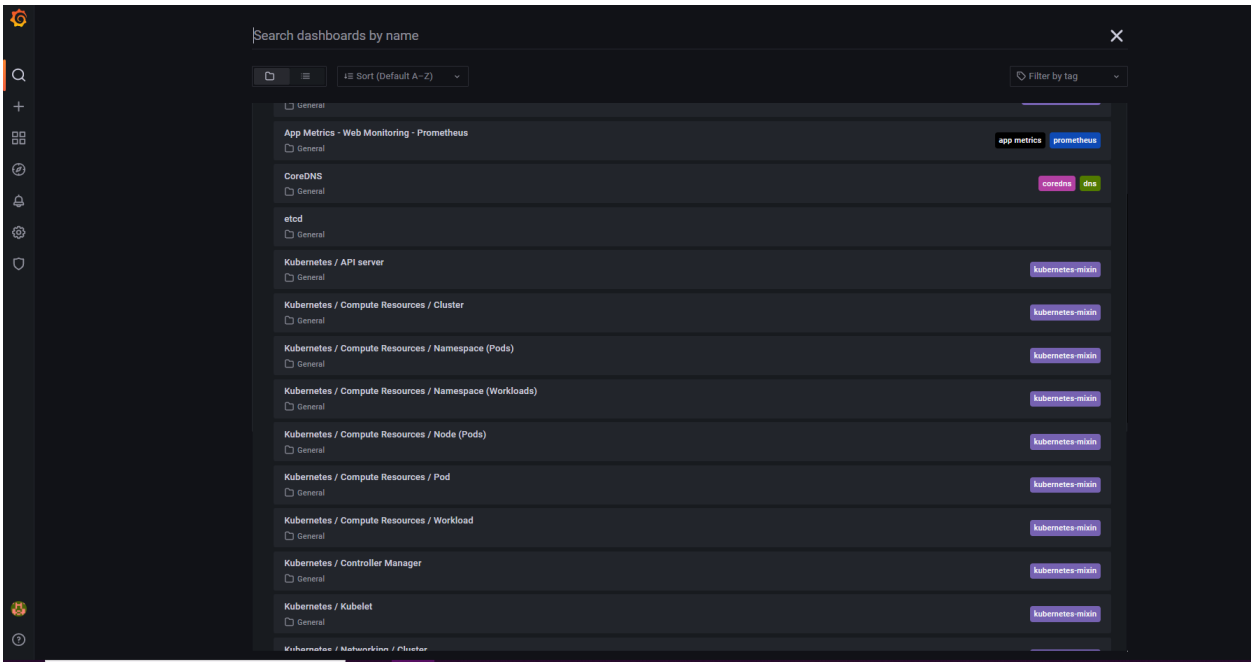
From the Kubernetes readme: "*Kubernetes Dashboard is a general-purpose, web-based UI for Kubernetes clusters. It allows users to manage applications running in the cluster and troubleshoot them, as well as manage the cluster itself.*"

I looked into monitoring tools for Kubernetes-based applications and one that was popular was Prometheus and then displaying it with Grafana.

From the Prometheus Docs it says this:

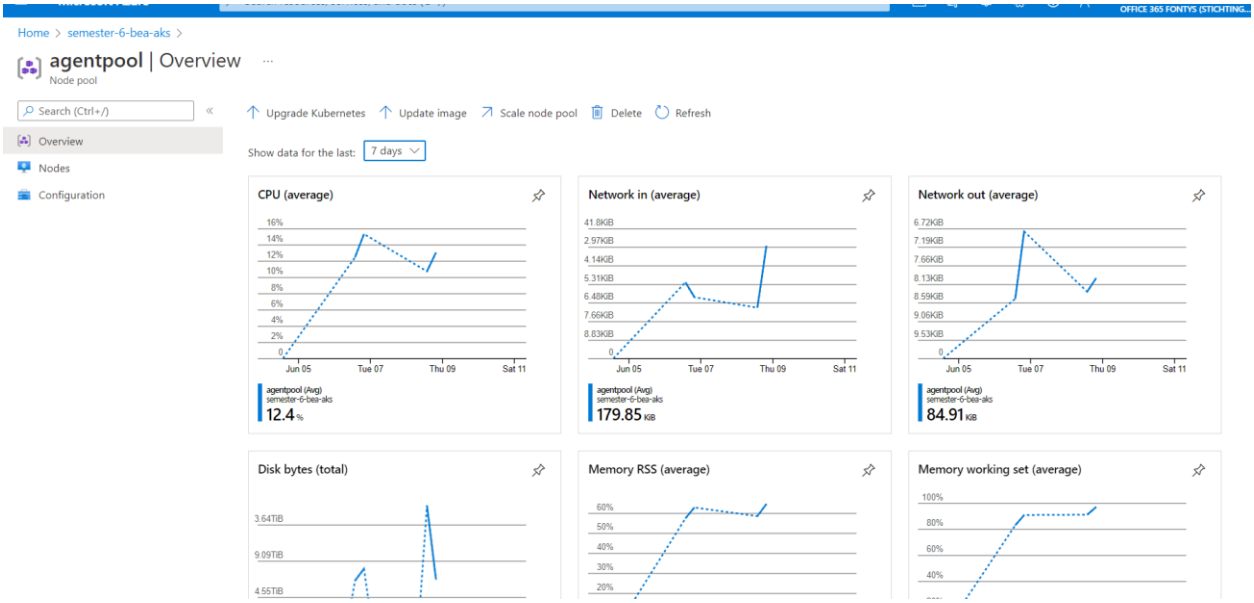
Prometheus works well for recording any purely numeric time series. It fits both machine-centric monitoring as well as monitoring of highly dynamic service-oriented architectures. In a world of microservices, its support for multi-dimensional data collection and querying is a particular strength. Prometheus is designed for reliability, to be the system you go to during an outage to allow you to quickly diagnose problems.

Which meant I decided to go for Prometheus and Grafana and this is how it looked after setting it up:



It also had a lot of other possibilities to check other types of monitoring of my Kubernetes cluster, thus it is very handy!

I then also realized that Azure has built-in monitoring, which also is cool and means I have 2 types of monitoring.



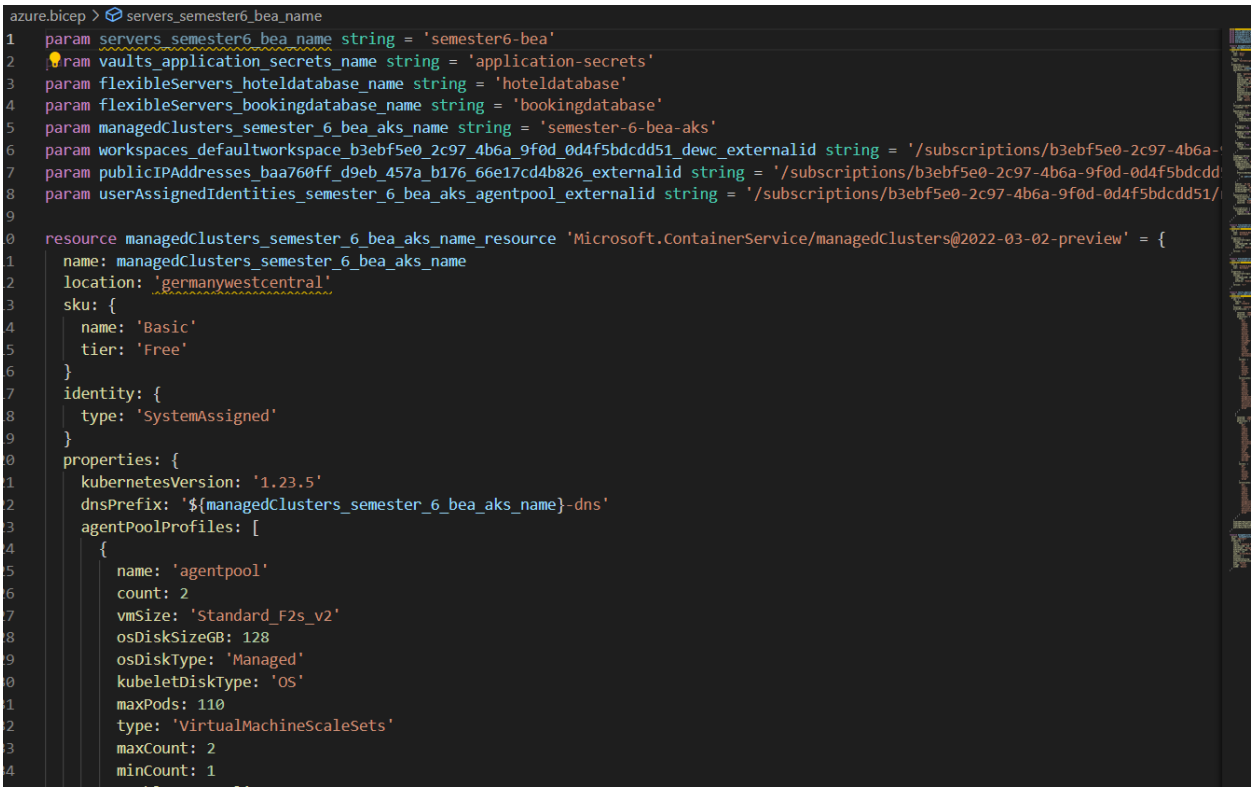
Infrastructure as code

Infrastructure as code is something I heard about at the dotnet conference I went to. It means coding the azure environment instead of manually creating a service, adjusting the options, etc. Since it is coded, it means it will stay the same every time, and automating it into the pipeline means that developers don't need to manually manage servers, operating systems, storage, and other infrastructure components each time they develop or deploy an application.

Microsoft docs describe IaC like this:

"*Infrastructure as Code (IaC) is the management of infrastructure (networks, virtual machines, load balancers, and connection topology) in a descriptive model, using the same versioning as the DevOps team uses for source code. Like the principle that the same source code generates the same binary, an IaC model generates the same environment every time it is applied. IaC is a key DevOps practice and is used in conjunction with continuous delivery*."

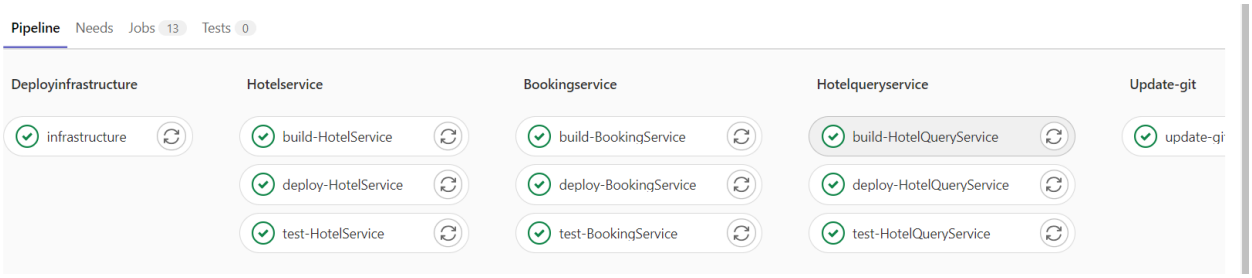
To do this, I used ARM Json templates that could be retrived from Azure website. However, ARM is being replaced by Bicep because Json is hard to read in large infrastructure environments. To do this, I simply converted the Json to a Bicep templates/language, and since Bicep is a programming language, it is possible to see coding errors in the errors list.



This is how it looks in Bicep.

This is how the ARM JSON looks like, much longer and unreadable.

Finally, I added this to my pipeline which means now my pipeline is extented with infrastructure step.



Stryker Mutation testing

Stryker is an open-source project I contributed to in my free time, and I was thinking it could be fun to integrate it and try it out in my individual project.

Stryker is a tool to test your unit tests by performing mutation testing. Everyone knows your applications should be unit tested to ensure the application performs according to the specifications, and to ensure that when the application is changed the features still work as expected. To know if you have enough unit tests we use a metric called code coverage. However, it is possible to have high unit test coverage without actually testing anything. You could for example simply forget to add assertions in your unit tests and your coverage would still be high. Mutation testing aims to solve this problem by changing your source code (for example change a + to a – or change a > to a <) and then running your unit tests. Mutation testing expects that after this change at least one unit test should fail. If so, you have sufficiently tested this part of the code. If not, you are lacking test coverage.

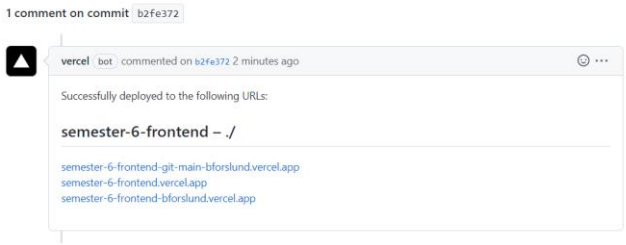
This means Stryker can help in the DevOps process by providing you with better metrics and insights into your unit test quality.

```
private static bool Between(DateTime startNewBooking, DateTime endNewBooking, Booking booking)
{
    return !((endNewBooking < booking.Start && startNewBooking < booking.Start) ||
    return !((endNewBooking < booking.Start || startNewBooking < booking.Start) ||
                        (booking.End < startNewBooking && booking.Start < startNewBooking));
}
```

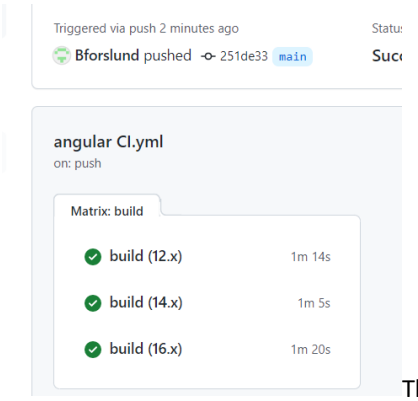
Here, this function is used in my unit tests, so the test coverage marks this function as good coverage. However, if the && becomes || this would still pass my unit tests. This means that I need a better unit test specifically for this function.

Vercel: front-end deployment

For the front-end I wanted to deploy it on Vercel, but because Fontys fhict GitLab environment cannot link with Vercel very good, I decided to put the front-end on my GitHub account. Here I also created a build pipeline for the front-end and also Vercel has built in functions to test the application and deploy it. This means I have a Ci/CD pipeline for both front-end and back-end



This is how it looks when Vercel has deployed what I push to git.



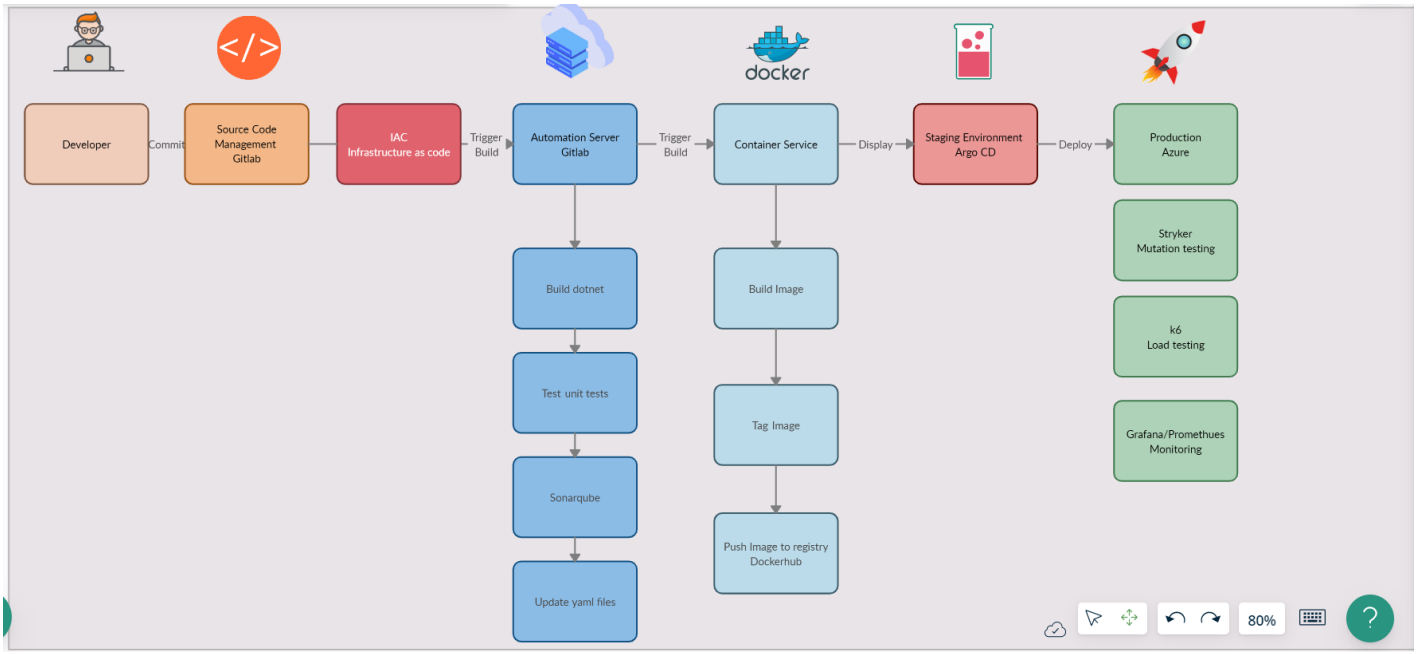
This is the pipeline which succeeded.

30 lines (25 sloc) | 673 Bytes

```yaml
name: Node.js CI

on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build:

    runs-on: ubuntu-latest

    strategy:
      matrix:
        node-version: [12.x, 14.x, 16.x]
        # See supported Node.js release schedule at https://nodejs.org/en/about/releases/

    steps:
    - uses: actions/checkout@v3
    - name: Use Node.js ${{ matrix.node-version }}
      uses: actions/setup-node@v3
      with:
        node-version: ${{ matrix.node-version }}
    - name: Install dependencies
      run: npm install
      working-directory: booking-frontend
    - name: Build
      run: npm run build
      working-directory: booking-frontend
```

1.5 DevOps LO Conclusion

For this learning outcome I have a fully automated pipeline in both backend and frontend.



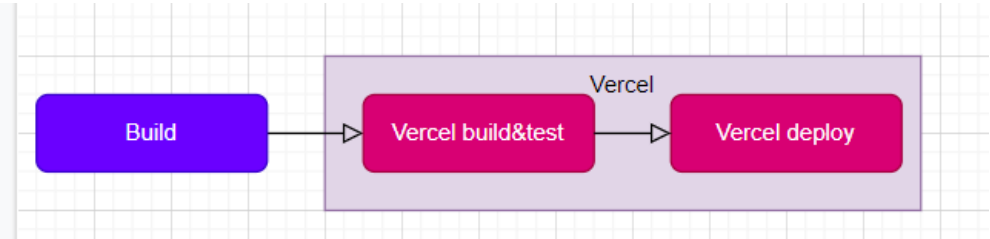
In this pipeline in **GitLab**: it starts with Infrastructure as Code, when that completes it starts building the app with dotnet build. To test the application, I have some unit tests and also SonarQube for code quality. Finally, after that it also builds and deploys the images on docker which triggers ArgoCD to deploy to my azure Kubernetes service. After production I do k6 testing, also Stryker and monitoring.

So, summarized:

- Infrastructure as code (IaC)
- Unit tests
- SonarQube
- ArgoCD
- K6 load testing
- Stryker mutation testing
- Monitoring Grafana Prometheus
- Monitoring on Azure

In the group project, I also wrote the design-oriented research on DevOps together with student Rawan. In the group project, we also used Jira as a planning tool. For the front-end I am using GitHub as environment and in my individual Gitlab because it was interesting learning both environments.

For the front-end I have a simple pipeline as well which is the environment **GitHub:**



6. Cloud Services

You can explain what a cloud platform provider is and can deploy (parts of) your application to a cloud platform. You integrate cloud services (for example: Serverless computing, cloud storage, container management) into your enterprise application, and can explain the added value of these cloud services for your application.

ID	Description	Type	Level
1.1	Sprint 2	Group project/Individual	Beginning
1.2	Sprint 3	Group project/Individual	Proficient
1.3	Sprint 4	Group project/Individual	Proficient
1.4	Sprint 5	Individual	**Advanced**
1.5	Conclusions	Individual/Group project	**Advanced**

1.1 Sprint 2

In Sprint 2 in the group project, we have added 0Auth which is a cloud provider. We also did a cloud MongoDB which I have implemented CRUD functions with. In the individual, I have started to research which DB to use.

- 0Auth authorization and authentication
- Cloud MongoDB
- Research on how to deploy the application in the group project
- Code that works with the Cloud DB

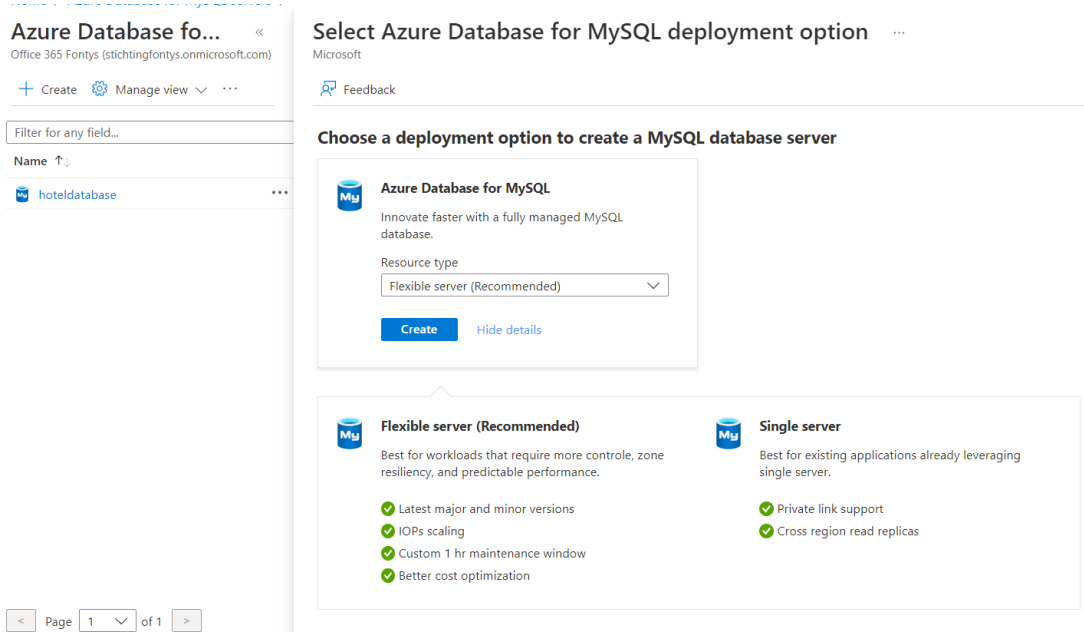
1.2 Sprint 3

In sprint 3, I worked mostly on my individual since the group project is already using cloud services at 0Auth and MongoDB. However, for my individual, I used Azure MySql Flexible Server.

- Azure MySQL Flexible Server
- Cloud Research

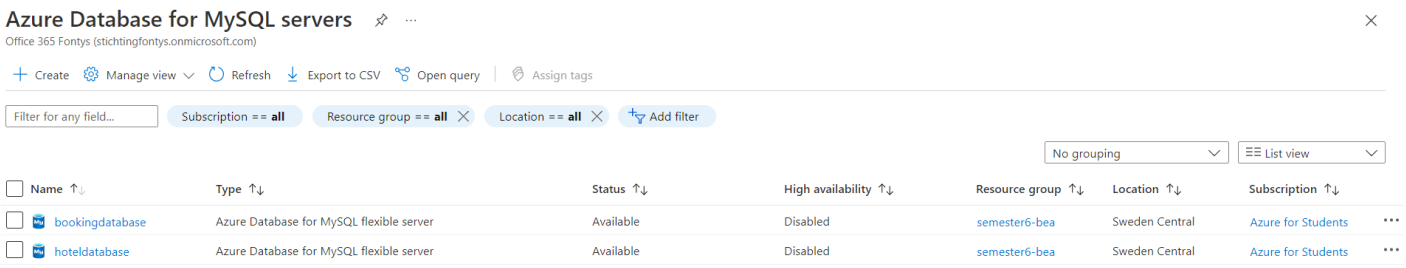
Azure MySQL Flexible Server

For my database, I wanted to use MySQL because it is what I am familiar with, and I also wanted it to be a cloud database and scalable. Azure was the easiest to set up between AWS and Google Cloud without providing a credit card, so my choice ended up being Azure. Azure is also simpler to use and is what I am familiar with since semester 3. It also has more functionalities.



These deployment options were good; I went for the recommended flexible server.

Then I created a database for each service:



1.3 Sprint 4

Group:

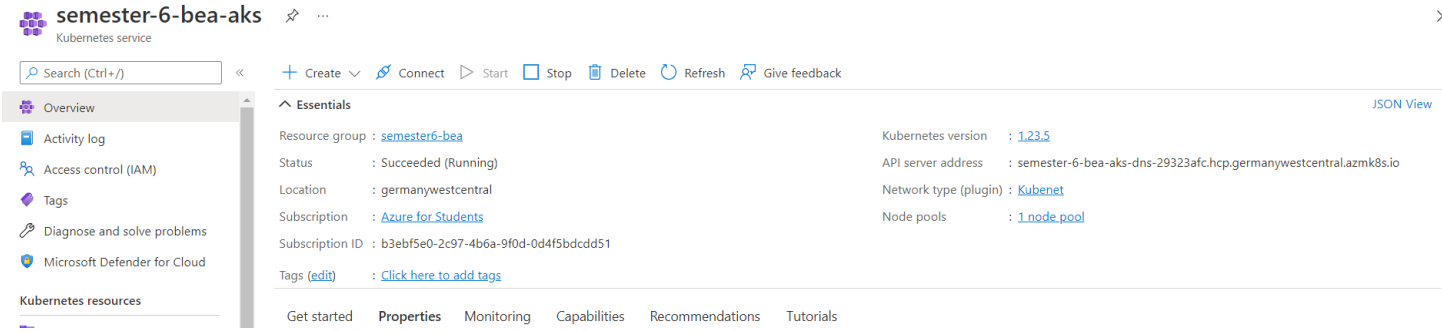
- Deployment to Google Cloud Kubernetes

The reason our group chose google cloud was that Bartosz, our teacher has credits that we can use for our group project.

Individual:

- Deployment to Azure Kubernetes service

Since I had already deployed my database to Azure, I decided that it made the most sense to keep my applications and my database close together. For that reason, I decided to use Azure Kubernetes Service and not Google Cloud Kubernetes. An added benefit is that I would learn to know both Azure Kubernetes and Google Cloud Kubernetes because there are slight differences between the two and now I could compare them.



For this I created a Kubernetes service in Azure that I deployed it to, so now I can reach my endpoints and API from the link that Azure provides. To save credits, I stop and start the service whenever I need it.

1.4 Final Sprint

For this sprint, I wanted to get an advanced for learning outcome, for this I decided to add more cloud services to my application.

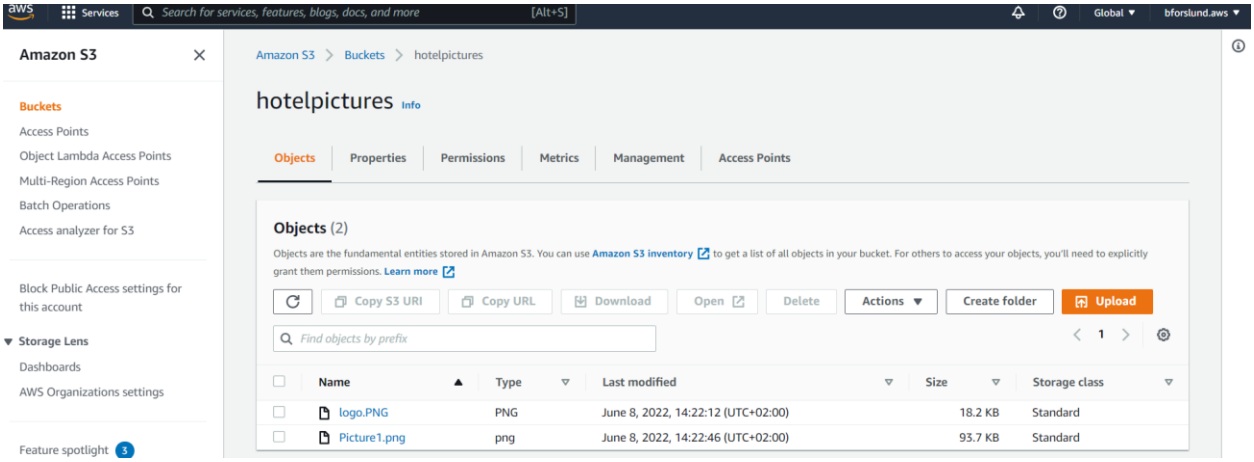
First, I added Grafana and Prometheus into the Kubernetes cluster, thus those are now in the cloud. I also deployed an influx DB that I used for saving k6 load testing results and displaying the results in Grafana.

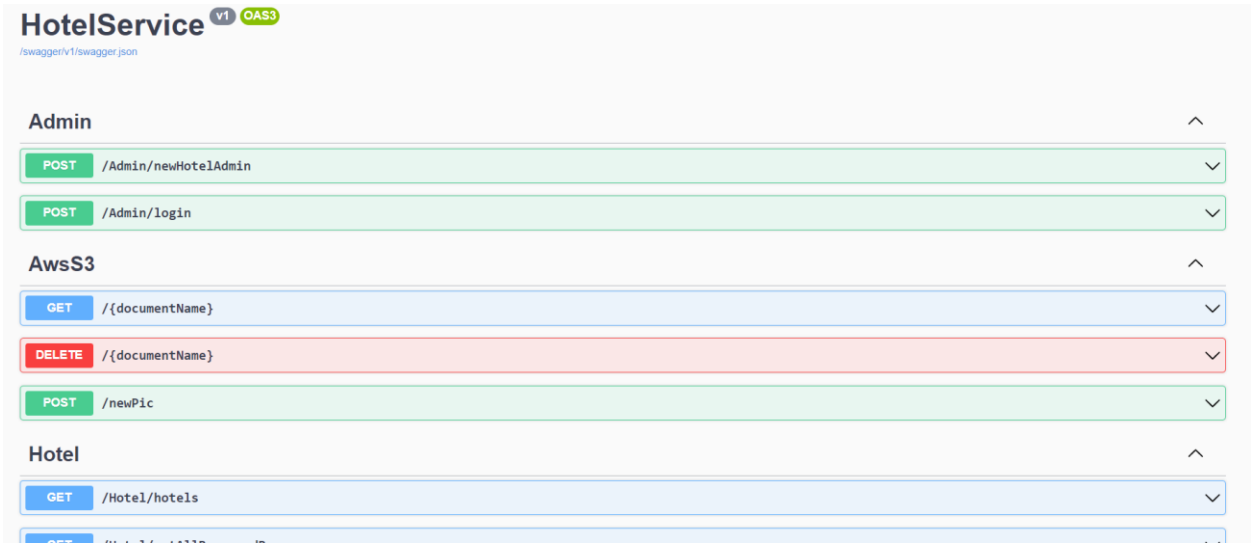
AWS S3

After that I decided to add pictures to my application, the reason for that being that as a hotel manager, of course, you want to upload pictures of the hotel you want people to book! Therefore this was a requirement and I investigated how to store the images. Since I already used Azure and Google Cloud for the individual and group project for deployment, I wanted to explore the last of the big 3 which is Amazon! From Amazon's own website:

"Amazon Simple Storage Service (Amazon S3) is an object storage service offering industry-leading scalability, data availability, security, and performance."

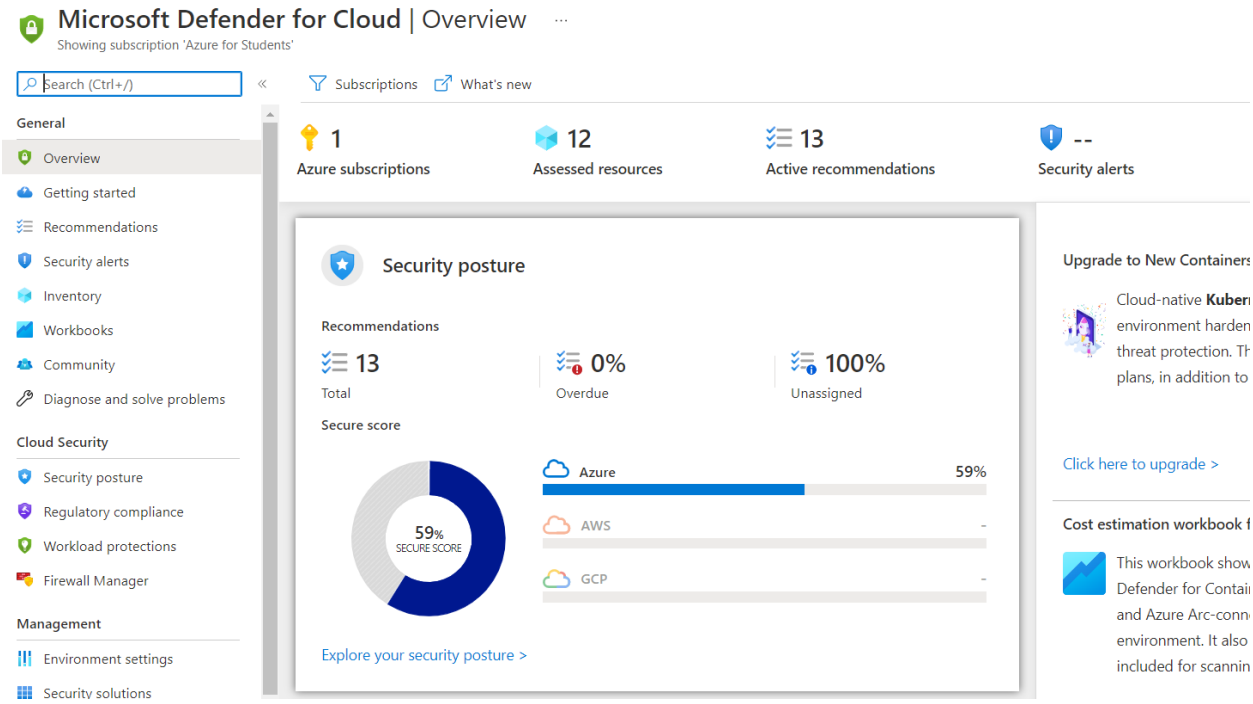
These are all requirements that are important for storing my pictures, and thus the option to use amazon was decided.





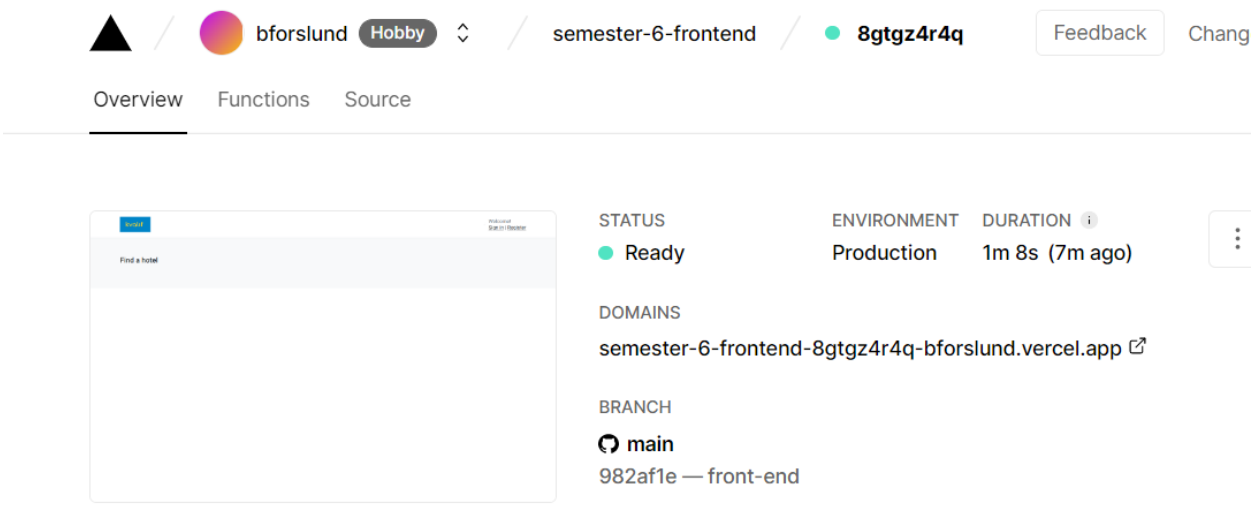
Microsoft Defender for Cloud

After thinking about what I could do more for security I talked to my friend who is in Cyber security. She told me that she was working on Microsoft Defender. I researched how I could implement Microsoft Defender and I discovered that it could integrate with my Azure MySQL database as well as my Azure Kubernetes service. While researching I found that Microsoft Defender is intended to secure all your cloud services even if they are deployed in Amazon AWS or Google Cloud which seemed like a huge benefit. More about Microsoft defender can be read in the security LO section.



Vercel

This was used to deploy the frontend.



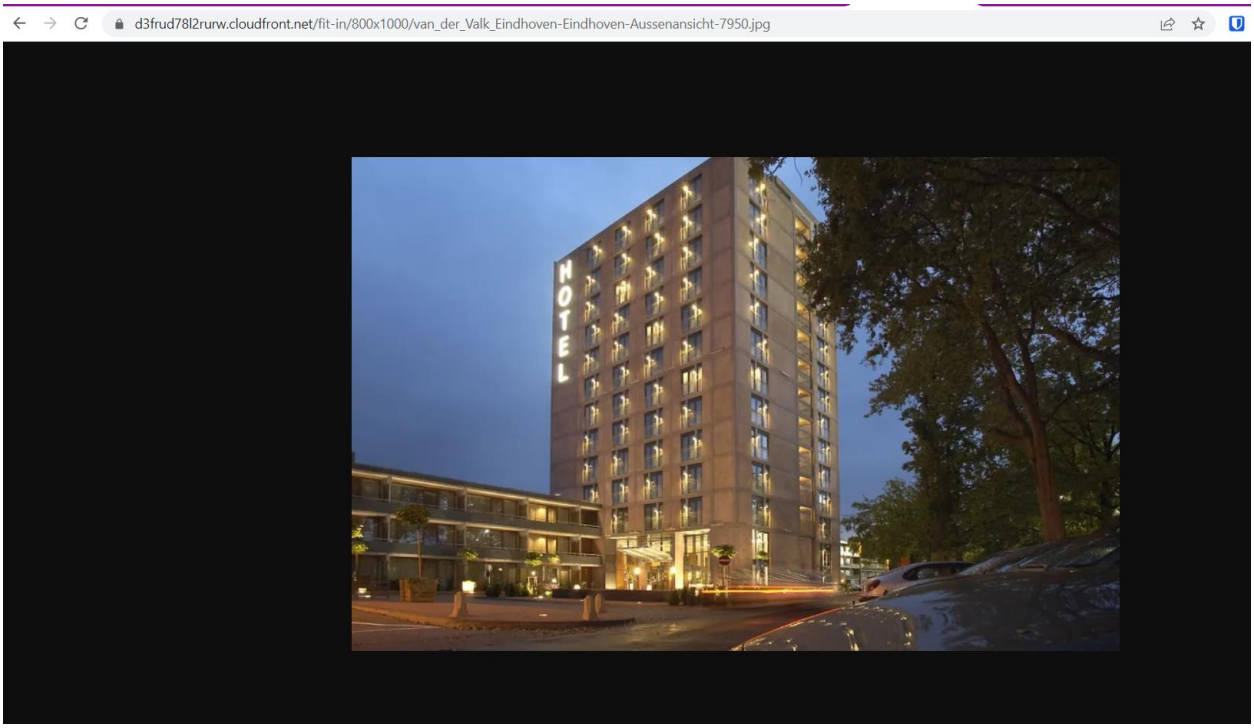
Amazon Lambda (FAAS)

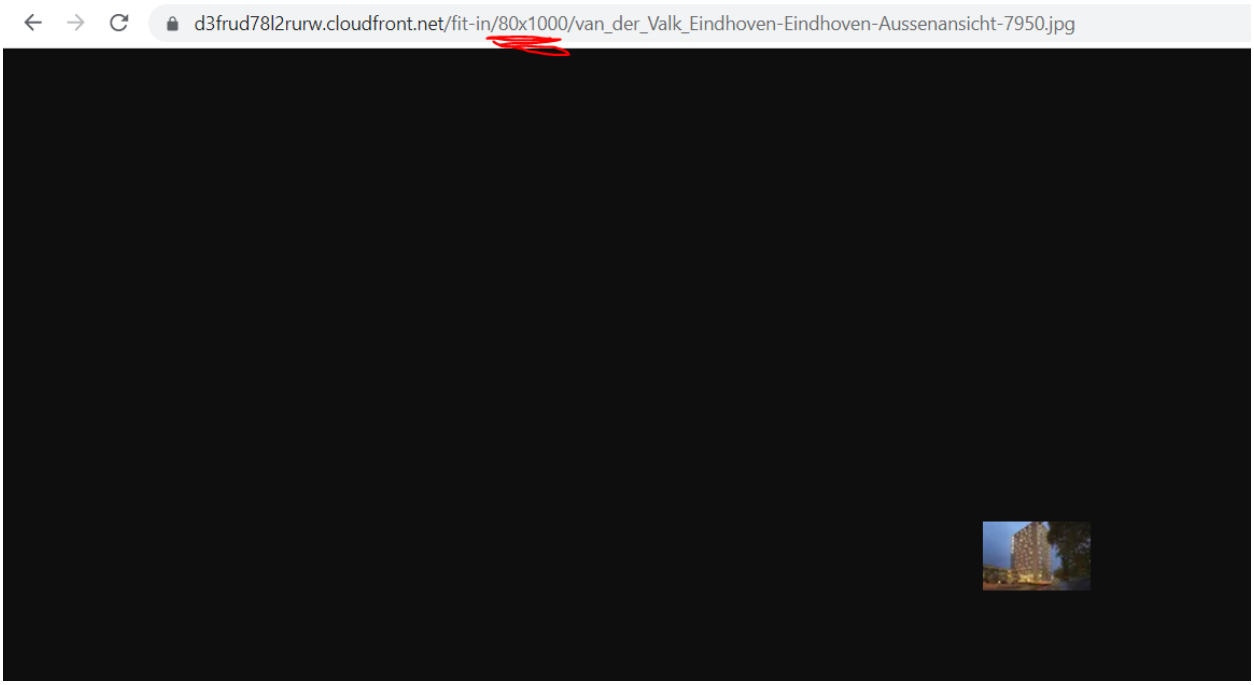
For my application, I want to be able to rescale pictures depending on if the user is using a mobile application or a desktop. I researched ways to be able to good rescale my pictures, and found out that using FAAS, function as a service is a good way to use it.

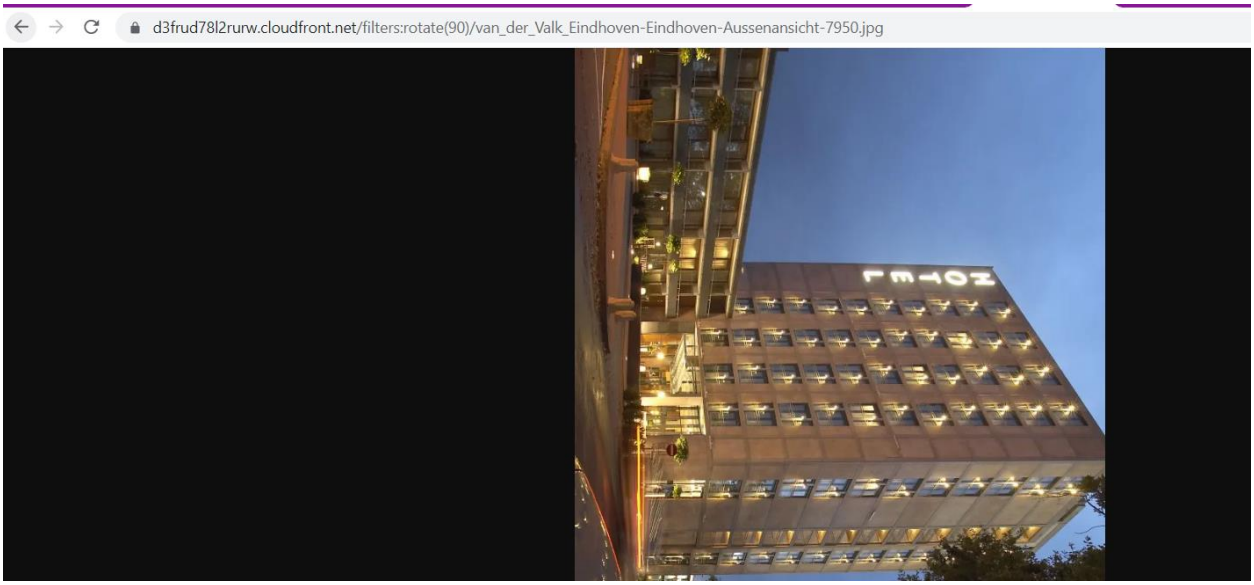
For this I am using one of the solutions that come with Amazon, they have a Serverless Image Handler that I deployed and used for rescaling pictures.

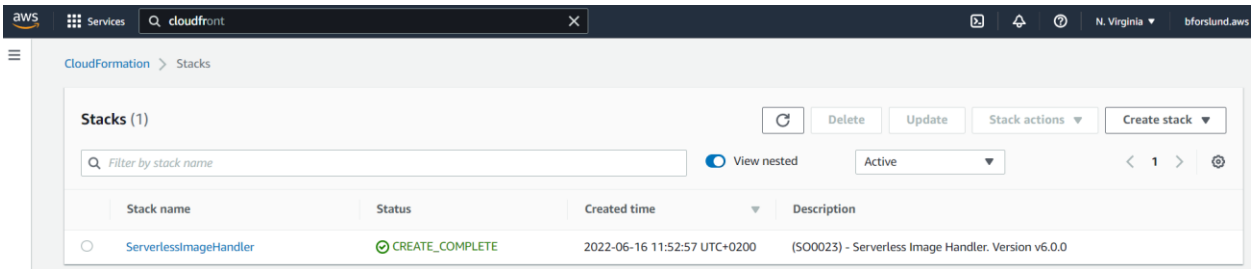
“When you redesign your website or application, you can add new dimensions on the fly, rather than working to reprocess the entire archive of images that you have stored.”

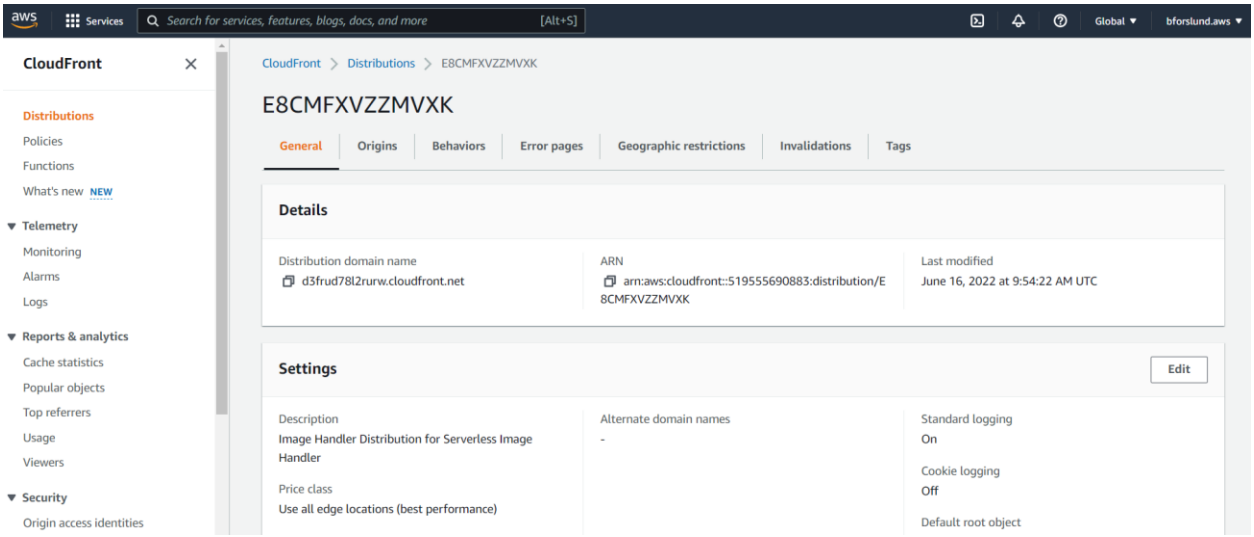
FaaS is frequently used to store or process user inputs, such as multimedia processing or other types of data

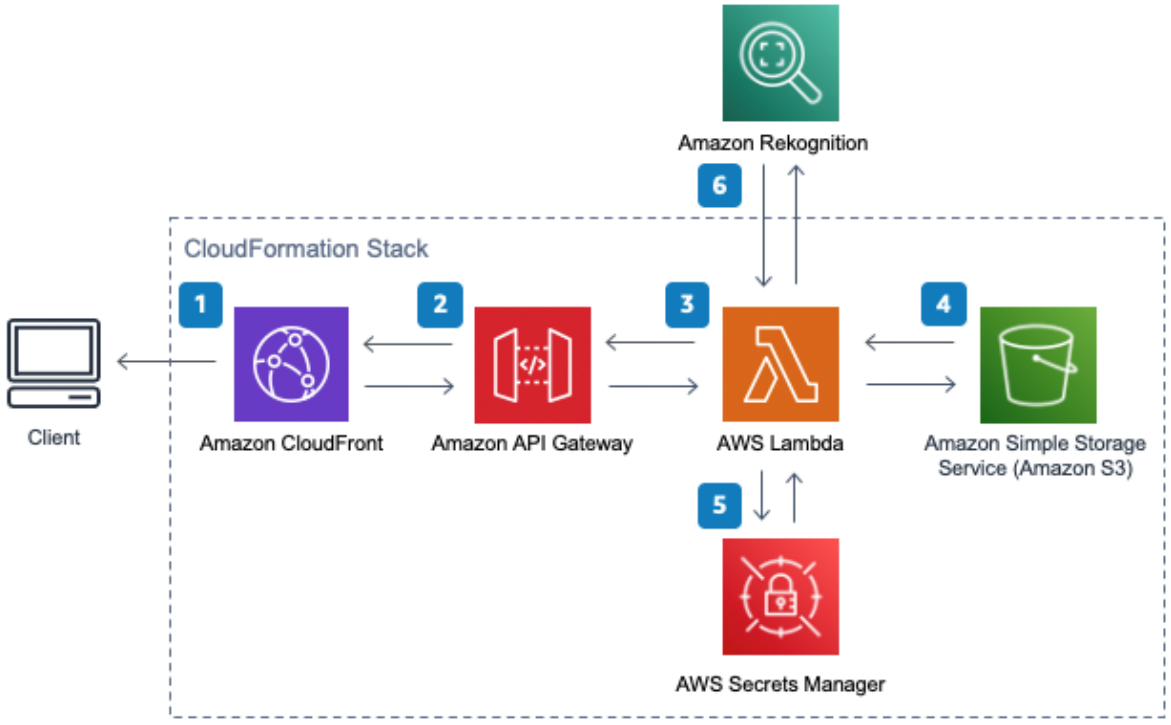



d3frud78l2rurw.cloudfront.net/fit-in/80x1000/van_der_Valk_Eindhoven-Eindhoven-Aussenansicht-7950.jpg


d3frud78l2rurw.cloudfront.net/filters:rotate(90)/van_der_Valk_Eindhoven-Eindhoven-Aussenansicht-7950.jpg


aws
Services
cloudfront
N. Virginia
bforslund.aws
CloudFormation
Stacks
Stacks (1)
Delete
Update
Stack actions
Create stack
Filter by stack name
View nested
Active
1
Stack name
Status
Created time
Description
ServerlessImageHandler
CREATE_COMPLETE
2022-06-16 11:52:57 UTC+0200
(SO0023) - Serverless Image Handler. Version v6.0.0


aws
Services
Search for services, features, blogs, docs, and more [Alt+S]
Global
bforslund.aws
CloudFront
Distributions
Policies
Functions
What's new NEW
Telemetry
Monitoring
Alarms
Logs
Reports & analytics
Cache statistics
Popular objects
Top referrers
Usage
Viewers
Security
Origin access identities
CloudFront > Distributions > E8CMFXVZZMVXK
E8CMFXVZZMVXK
General
Origins
Behaviors
Error pages
Geographic restrictions
Invalidations
Tags
Details
Distribution domain name
d3frud78l2rurw.cloudfront.net
ARN
arn:aws:cloudfront::519555690883:distribution/E8CMFXVZZMVXK
Last modified
June 16, 2022 at 9:54:22 AM UTC
Settings
Edit
Description
Image Handler Distribution for Serverless Image Handler
Price class
Use all edge locations (best performance)
Alternate domain names
-
Standard logging
On
Cookie logging
Off
Default root object


Amazon Rekognition
6
CloudFormation Stack
1
2
3
4
Client
Amazon CloudFront
Amazon API Gateway
AWS Lambda
Amazon Simple Storage Service (Amazon S3)
5
AWS Secrets Manager

1.5 Cloud Services LO Conclusions

For cloud services in my individual, I have applied and worked with four cloud providers and seven services:

- Azure Kubernetes service
- Azure Key Vault
- Azure Redis Cache
- Amazon s3 bucket
- Amazon Lambda: FAAS/serverless computing
- Microsoft Defender for Cloud
- Vercel

With Azure in my Kubernetes cluster, I also have deployed several applications (that aren't my own created services) and can also be accessed from the cloud:

- RabbitMQ
- Cert-manager
- Influx DB
- ArgoCD
- Grafana
- Prometheus

With the group project:

- Google cloud deployment
- MongoDB
- 0Auth authorization and authentication

Since I applied my knowledge both in the individual and in the group project with several instances, this means I have an **advanced** for this learning outcome.

7. Security by design

You investigate how to minimize security risks for your application, and you incorporate best practices in your whole software development process.

ID	Description	Type	Level
1.1	Sprint 1	Individual	Orienting
1.2	Sprint 2	Group project	Orienting
1.3	Sprint 3	Group project/ Individual	Proficient
1.4	Sprint 4	Individual	Proficient
1.5	Sprint 5	Individual	**Advanced**
1.6	Conclusions	Group project/ Individual	**Advanced**

1.1: Sprint 1

For this sprint, I have started looking into having to make my application secure. I followed the cyber security workshop. I also looked into JWT and OAuth. I also choose good frameworks that consider security in the individual project.

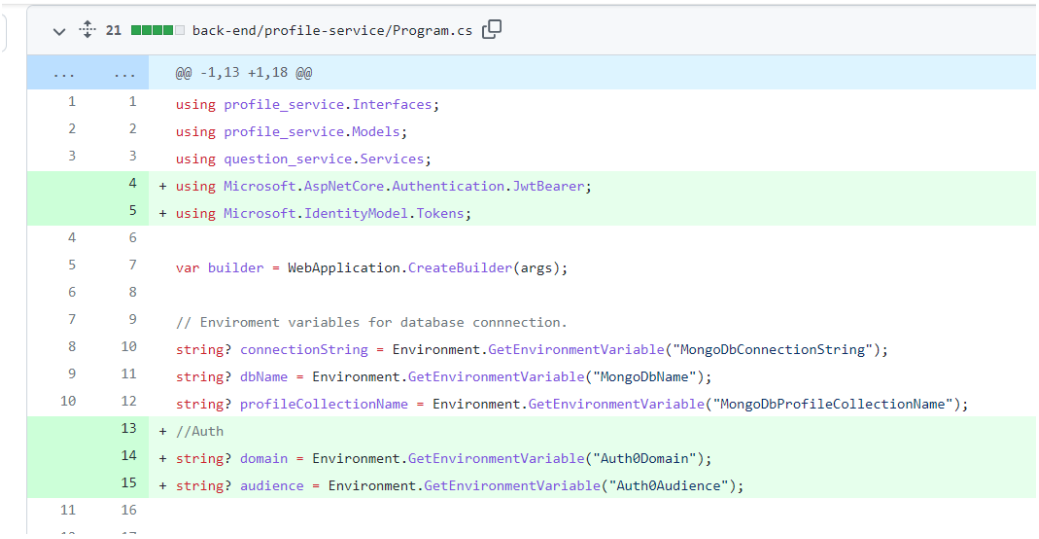
1.2 Sprint 2

For this sprint, one of my teammates added OAuth to our application using Auth0. I was reviewing the pull request and I understand how it works. Then I also adjusted the profile service to use Auth0 authentication and authorization.

OAuth

The reason we chose OAuth in the group project is that: OAuth is an open-standard authorization protocol or framework that enables "secure designated access" in apps.

OAuth does not exchange password information, instead of relying on permission tokens to establish a connection between users and service providers. OAuth is a security mechanism that allows you to authorize one application to engage with another on your behalf without disclosing your password.



1.3 Sprint 3

For this learning outcome I worked on this:

- ORM in the backend (Entity framework)
- Testing the front-end Angular security (Pen-testing)
- Separate databases & Database security: Azure Firewall
- Encrypting sensitive personal data
- Encrypting Passwords
- JWT tokens Authorization & Authentication

ORM in the backend (Entity framework)

To make it easier to update the database and minimize attacks I decided to add the Entity framework ORM.

In OWASP top 10, there is a vulnerability called A03:2021 – Injection.: This vulnerability is relevant for this application since it has a lot of data that can be manipulated, for example changing the rooms to available when they are not. SQL injection is a common security risk and since this application will use SQL queries it could be possible to break the application.

To eliminate this risk in this application I have used Entity framework and to have as little as possible input boxes as possible and server-side input validation.

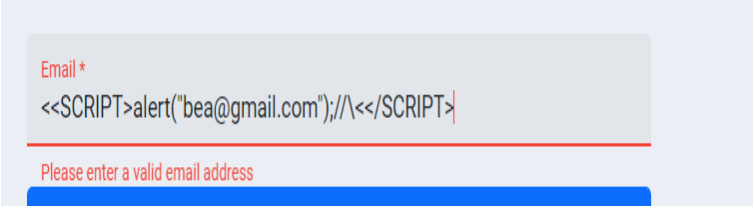
OWASP recommends using an ORM (Object Relational Mapping). Entity framework is an ORM, which makes this risk after these implementations low for this application.

Testing the security

This application could be vulnerable the to OWASP top 10 Cross-Site Scripting (XSS) vulnerability. This vulnerability is relevant because the website has HTML and javascript which is where XSS applies.

This application is using the Angular framework which specifies as one design goal:" To systematically block XSS bugs, Angular treats all values as untrusted by default". This means that the risk of this attack is low.

To confirm that Angular protects against XSS vulnerabilities I tested the application which shows it is not possible to perform cross-site scripting. This can be read in the **test report** which is in the attached sprint deliverables.



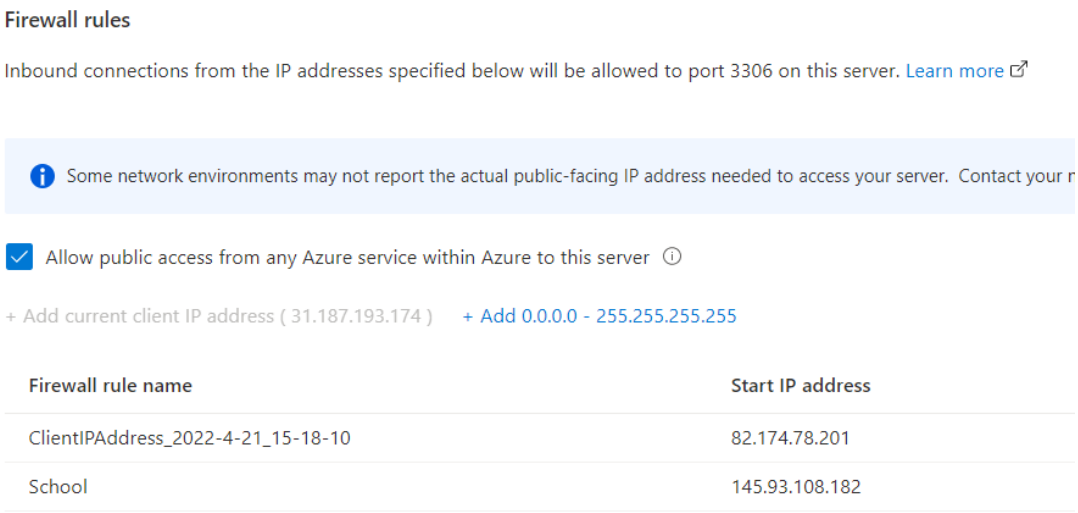
Separate databases and Database Firewall

The SQL Database firewall controls which IP addresses have access to the database. Before anyone may access the database, the firewall must be configured to allow inbound access. Inbound access is blocked for all sources by default.

In the OWASP top 10, there is a vulnerability called Broken Access Control, which means that authentication and access restriction are not properly implemented. With broken access control flaws, unauthenticated or unauthorized users may have access to sensitive files and systems, and this firewall is another preventative measure for that.

I also have two separate databases for each service which also prevents broken access control.

For this, I had to add the school IP to access the database when running it locally, but when I deployed it from the Kubernetes service, I checked the box to allow access from Azure services and disabled access from the school and my home network. Now only my Kubernetes cluster is allowed access to the databases.



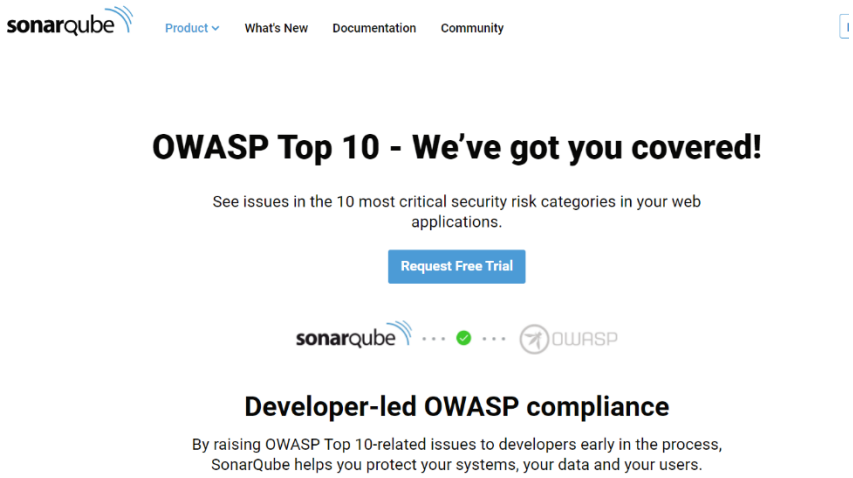
Encryption of sensitive data & Passwords

In the database, I also decided to encrypt the sensitive data and the passwords of the hotel managers. This is for both security and GDPR. This preventative measure covers the OWASP top 10 vulnerability Sensitive Data Exposure.

SonarQube

I added SonarQube Into the CI/CD pipeline because SonarQube can catch security risks in the code. It can detect code smells that can later lead to (security) bugs in the code but it can also warn about insecure libraries and code constructs. For example if you read a file from disk based on user input SonarQube can flag this code and ask you to verify that you are aware you need to sanitize user inputs before using the input to load the file. SonarQube helps you guard against path traversal vulnerabilities which are related to the OWASP top 10 broken access control security risk. From their website:

"SonarQube provides detailed issue descriptions and code highlights that explain why your code is at risk. The SonarSource Security Report facilitates communication by categorizing vulnerabilities in terms developers understand."



JWT tokens for Authorization

When deciding what Authorization framework or tools to use, I had the decision to choose between OAuth as we use in the group project, or JWT tokens. Since I already use OAuth in the group project, I thought it would be fun to learn about JWT tokens.

Without any authorization tools or frameworks, there is from OWASP top 10, The Broken Authentication vulnerability which means that the hacker has a list of common passwords and usernames and that an attack could happen because the user has a weak password. This risk is relevant for this application because uses an admin login.

According to OWASP top 10, it recommends using a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. I did this by using JWT tokens and when logging in as an admin it generates a token after login. The risk of a Broken Authentication attack becomes lower on this application after adding JWT tokens.

```
1 reference
public AuthenticateResponse Authenticate(AuthenticateRequest model)
{
    var user = _context.Admins.SingleOrDefault(x => x.Username == model.Username);

    // return null if user not found
    if (user == null) return null;

    string decryptedPassword = _cipherService.Decrypt(user.Password);

    //return null if login password is incorrect
    if (model.Password != decryptedPassword) return null;

    // authentication successful so generate jwt token
    var token = generateJwtToken(user);

    return new AuthenticateResponse(user, token);
}

1 reference
private string generateJwtToken(HotelAdmin hotelAdmin)
{
    // generate token that is valid for 7 days
    var tokenHandler = new JwtSecurityTokenHandler();
    var key = Encoding.ASCII.GetBytes(_jwtSecret.Secret);
    var tokenDescriptor = new SecurityTokenDescriptor
    {
        Subject = new ClaimsIdentity(new[] { new Claim("id", hotelAdmin.HotelId.ToString()) }),
        Expires = DateTime.UtcNow.AddDays(7),
        SigningCredentials = new SigningCredentials(new SymmetricSecurityKey(key), SecurityAlgorithms.HmacSha256Signature)
    };
    var token = tokenHandler.CreateToken(tokenDescriptor);
    return tokenHandler.WriteToken(token);
}

}
```

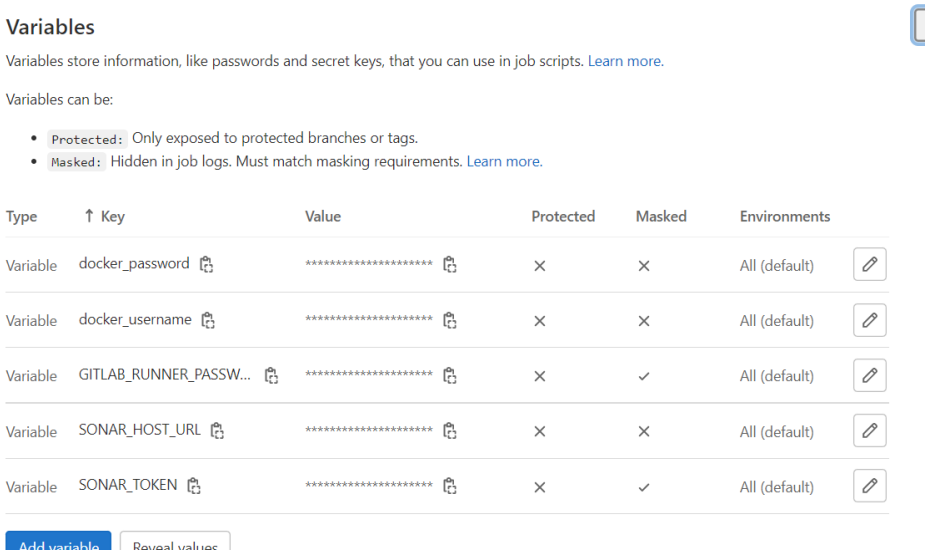
1.4 Sprint 4

For the security, for this sprint summarized:

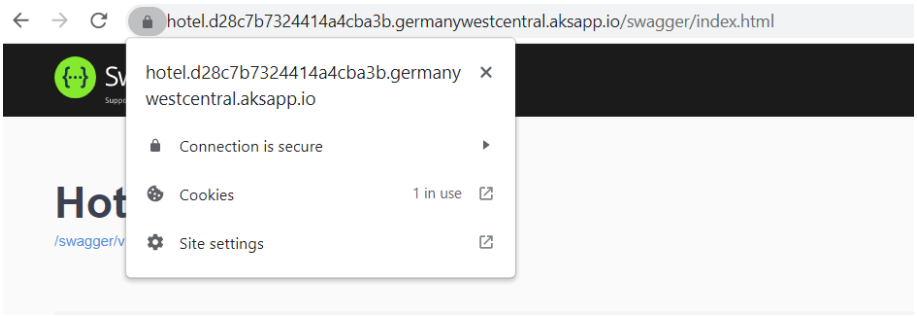
- Environment variables
- HTTPS certificate (Helm & cert-manager)
- JWT tokens Authentication

Environmental Variables

First, I added environmental variables/CICD variables and store them as protected as possible and did not hardcode any passwords.



HTTPS certificate



Then I also created an HTTPS certificate, since I noticed my website was not secure and it did not have an HTTPS site.

From OWASP top 10, a vulnerability is Sensitive Data Exposure: This vulnerability can happen when sending sensitive information over the HTTP connection, from client to server. This risk is relevant for this application since it uses passwords and users' personal information.

To prevent this vulnerability, I looked up how I could add an HTTPS certificate to my sites. I found out about the application cert-manager. It is an application to manage SSL certificates in kubernetes environments. To install cert-manager in kubernetes I used a kubernetes package manager called Helm. With helm I could easily install cert-manager to my cluster.

Not only did I have to install the cert-manager and Helm, but I learned how to configure cert-manager to use let's encrypt to request free SSL certificates.

issuer.yaml 474 Bytes

```yaml
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    preferredChain: "ISRG Root X1"
    email: noreply@fontys.nl
    privateKeySecretRef:
      name: letsencrypt
    solvers:
      - http01:
          ingress:
            ingressTemplate:
              metadata:
                annotations:
                  kubernetes.io/ingress.class: addon-http-application-routing
```

```
to the latest version and try again
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\kubernetes> kubectl get cert
NAME                              READY   SECRET                            AGE
bookingservice-ingress-tls        True    bookingservice-ingress-tls        7m13s
hotelservice-ingress-tls          True    hotelservice-ingress-tls          7m13s
rabbitmq-management-ingress-tls   True    rabbitmq-management-ingress-tls   7m13s
PS C:\Users\Bea\Documents\Semester 6\IND\semester-6-project\kubernetes>
```

I learned how to use, configure and install cert-manager so that traffic between clients and my applications is encrypted.

JWT Tokens Authentication

In the last sprint, I added authorization for hotel managers to log in and manage their hotels. In this sprint, I added authentication so that normal users cannot access certain endpoints (for example add rooms) without being authorized.

From OWASP top 10, there is a vulnerability of Broken Access Control, which means that authentication and access restriction are not properly implemented. With broken access control flaws, unauthenticated or unauthorized users may have access to sensitive files and systems, and this JWT middleware is another preventative measure for that.

```
[Authorize]
[HttpPost]
0 references
public async Task<ActionResult> AddHotelAsync(Hotel hotel)
{
    _context.Hotels.Add(hotel);
    await _context.SaveChangesAsync();
```

ntroller.cs | JwtMiddleware.cs | AdminService.cs | HotelController.cs | AdminController.cs | Au

Service | HotelService.Authentication.JwtMiddleware | attac

```
0 references
public async Task Invoke(HttpContext context, AdminService adminService)
{
    var token = context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();

    if (token != null)
        await attachUserToContextAsync(context, adminService, token);

    await _next(context);
}

1 reference
private async Task attachUserToContextAsync(HttpContext context, AdminService adminService, string token)
{
    try
    {
        string jwt = Environment.GetEnvironmentVariable("JWT_token");
        var tokenHandler = new JwtSecurityTokenHandler();
        var key = Encoding.ASCII.GetBytes(jwt);
        tokenHandler.ValidateToken(token, new TokenValidationParameters
        {
            ValidateIssuerSigningKey = true,
            IssuerSigningKey = new SymmetricSecurityKey(key),
            ValidateIssuer = false,
            ValidateAudience = false,
            // set clockskew to zero so tokens expire exactly at token expiration time (instead of 5 minutes later)
            ClockSkew = TimeSpan.Zero
        }, out SecurityToken validatedToken);

        var jwtToken = (JwtSecurityToken)validatedToken;
        var hotelId = int.Parse(jwtToken.Claims.First(x => x.Type == "id").Value);

        // attach user to context on successful jwt validation
        context.Items["admin"] = await adminService.GetAdminByHotelIdAsync(hotelId);
    }
    catch
    {
        // do nothing if jwt validation fails
        // user is not attached to context so request won't have access to secure routes
    }
}
}
```

1.5 Final Sprint

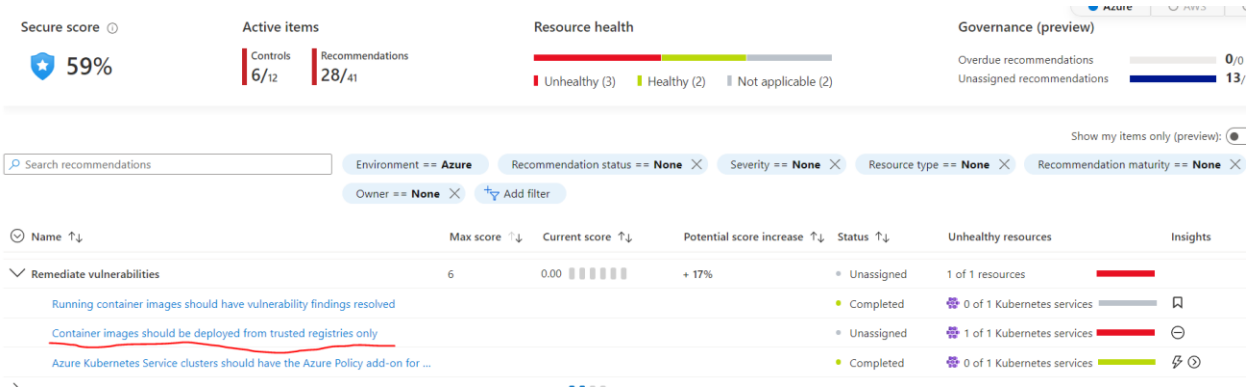
CQRS pattern

CQRS enforces security since it is easier to ensure that only the right domain entities are performing writes on the data.

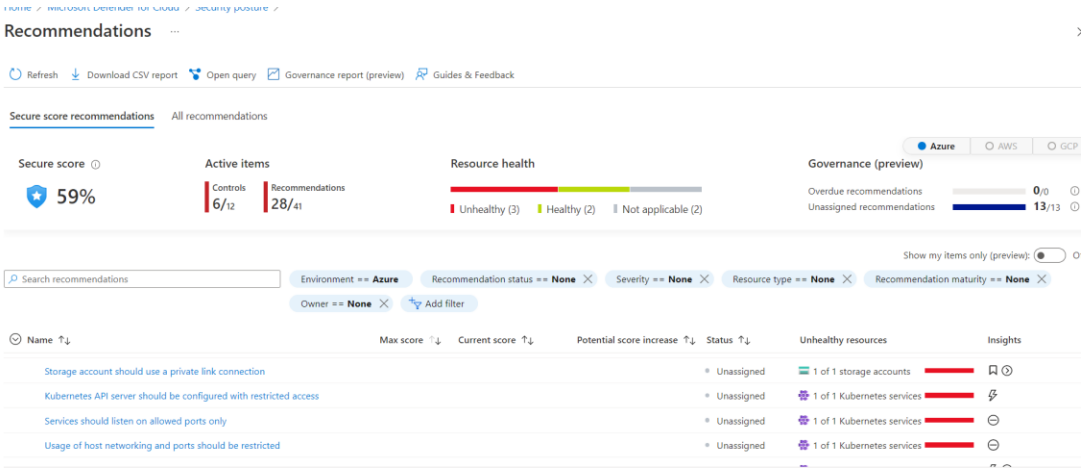
Microsoft Defender

I had a discussion with my cyber security study friend in Fontys about what options there were to secure Kubernetes, and she mentioned there was something called Microsoft Defender. I looked into Microsoft Defender and found out that it can be added to Kubernetes to warn about potential security issues such as container images in the cluster having security findings.

What Microsoft says: “Microsoft Defender for Containers is the cloud-native solution for securing your containers so you can improve, monitor, and maintain the security of your clusters, containers, and their applications.”.



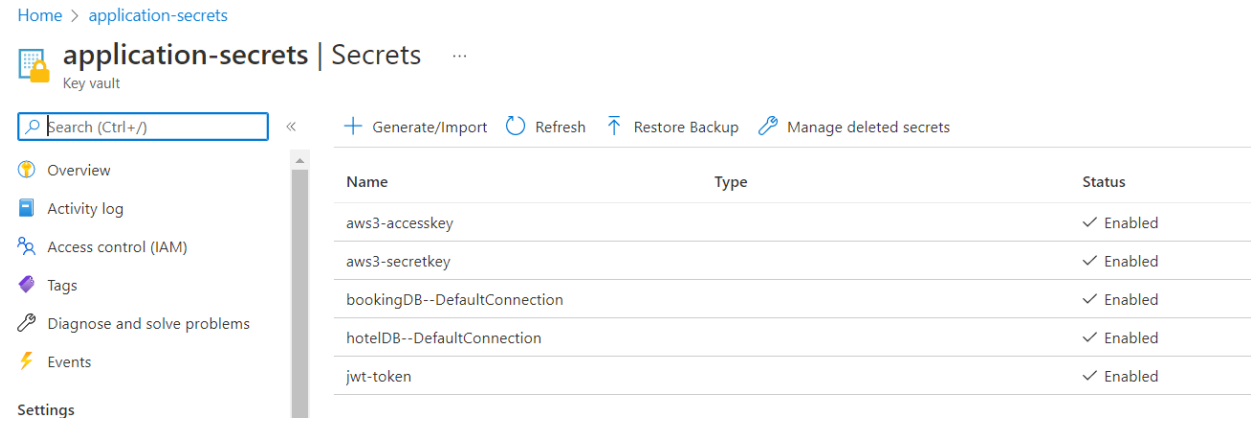
I am storing my application container images in Docker Hub, but Azure has a registry built-in that Microsoft defender considers more secure. To resolve this Microsoft Defender vulnerability finding I would have to migrate my container images from docker hub to azure container registry.



These were some more security issues that I found. If I had another sprint, I would try to fix these errors but for now, viewing them and seeing them has made me learn a lot about how I can continue securing my Kubernetes cluster.

Azure key vault

I was storing the Kubernetes secrets in the Kubernetes deployment YAML files, but that was not safe since the secrets were stored plain text on GitLab. That is why I decided to use Azure key vault. This means there are now no hard-coded passwords or passwords plain text available in source control. That limits risks as described by the OWASP top 10 vulnerability Sensitive Data Exposure.




```
25            readOnly: true
26        env:
27          - name: ASPNETCORE_ENVIRONMENT
28            value: Development
29          - name: RABBITMQ_HOSTNAME
30            value: rabbitmq-client.default.svc.cluster.local
31          - name: DbConnectionString
32            valueFrom:
33              secretKeyRef:
34                name: hotelservice-secret
35                key: connectionString
36          - name: AppConfiguration__AccessKey
37            valueFrom:
38              secretKeyRef:
39                name: hotelservice-secret
40                key: aws3-accesskey-key
41          - name: AppConfiguration__SecretKey
42            valueFrom:
43              secretKeyRef:
44                name: hotelservice-secret
45                key: aws3-secretkey-key
46          - name: AppConfiguration__BucketName
47            value: hotelpictures
48          - name: JWT_token
49            valueFrom:
50              secretKeyRef:
51                name: hotelservice-secret
52                key: jwt_token
```

Auth0

In the group project we used Auth0 for the users. My teammate Kevin added Auth0 to the project, but since I was responsible for CRUD of profiles, I had to work with Auth0 as well. This is how I got the user from the token and how I delete a user from Auth0.

```
public async Task<ActionResult<string>> DeleteAsync(string? authIdentifier)
{
    try
    {
       string mgmtToken = await GetManagementTokenAsync();
       using var mgmtClient = new ManagementApiClient(mgmtToken, new Uri("https://blind-date.eu.auth0.com/api/v2"));

       await _profiles.DeleteOneAsync(s => s.OAuthIdentifier == authIdentifier);

       await mgmtClient.Users.DeleteAsync(authIdentifier);

       return new ObjectResult("Successfully deleted") { StatusCode = 200 };
    }

    catch (Exception ex)
    {
        return new ObjectResult($"Invalid Auth0 ID given. {ex}") { StatusCode = 400 };
    }
}

public string GetUserByJWTToken()
{
    var userFromJWT = _httpContext.HttpContext.User;
    var userIdentifier = userFromJWT.Claims.Where(claim => claim.Type.Contains("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/nameidentifier")).FirstOrDefault
    return userIdentifier;
}

private async Task<string> GetManagementTokenAsync()
{
```

1.6 Security by design LO Conclusions

Since I investigated the security vulnerabilities and incorporated remediations into my applications, I believe I have a final grade of **advanced** on this learning outcome.

For the security learning outcome, I have implemented OWASP secure coding and used best practices to prevent security risks for all steps in the software development process and then also tested the security to confirm remediations were effective.

I investigated the most common and relevant security vulnerabilities in the OWASP top 10 for this application and to minimize and prevent these risks I have used these practices:

- ORM in the backend (Entity framework)
- Testing the front-end security (Pen-testing)
- Separate databases & Database security: Azure Firewall
- Encrypting sensitive personal data & Passwords
- JWT token Authorization & Authentication
- Environment/CI-CD variables in Gitlab
- HTTPS certificates (cert-manager)
- Microsoft Defender for cloud (Cloud security vulnerability reporting and remediation tool)
- Azure key vault (no hard-coded passwords)
- CQRS pattern
- SonarQube (checks security)
- Auth0 OAuth Authorization & Authentication

Then also by choosing to use services and tools that provide built-in security:

ArgoCD

- Using a pull instead of push method of application deployment lowers the risk of unauthorized persons gaining access to Kubernetes since developers no longer require access to Kubernetes to perform application deployment.
- ArgoCD provides a secure way to access application logs without having access to the Kubernetes cluster. This lowers the risk of unauthorized persons gaining access to Kubernetes because developers no longer need access to the Kubernetes cluster to be able to access application logs.

Azure

- The Azure Database for MySQL service uses the FIPS 140-2 validated cryptographic module for storage encryption of data-at-rest. Since I am using this database type my application data will be stored at rest using encryption keys from a hardware secure module.
- Data, including backups, are encrypted on disk, including the temporary files created while running queries.

AWS

- The AWS S3 bucket used for storing the hotel pictures has public access blocked by default
- Access to the S3 bucket is provided by a separate account with role based access control on the bucket.

8. Distributed data

You are aware of specific data requirements for enterprise systems. You apply best practices for distributed data during your whole development process, both for non-functional and functional requirements. You especially take legal and ethical issues into consideration.

ID	Description	Type	Level
1.1	Sprint 2	Group project/Individual	Orienting
1.2	Sprint 3	Group project/Individual	Beginning
1.3	Sprint 4	Group project/Individual	Proficient
1.4	Sprint 5	Group project/Individual	**Advanced**
1.5	Conclusions	Group project/Individual	**Advanced**

1.1: Sprint 2

For this learning outcome I contributed to an ethical design report we made in the group project. In the individual project, I am storing the personal information in a separate database and trying to minimize the amount of personal information to store. We also did an Ethics game with the semester coach and the other students.

1.2 Sprint 3

For this sprint, I added ORM (entity framework) which is good for applying data. I also hashed the data in the database to take legal issues into consideration.

- ORM
- Hashing sensitive data

```
1 reference
public async Task<Booking> CreateBookingAsync(Booking booking)
{
    Booking newBooking = new Booking(booking.HotelId, _cipherService.Encrypt(booking.ContactInfo), booking.Start, booking.Enc
    _context.Bookings.Add(newBooking);
    await _context.SaveChangesAsync();
    return newBooking;
}
1 reference
public async Task<Booking> UpdateBookingAsync(Booking updatedBooking)
{
    var existingBooking = _context.Bookings.First(a => a.Id == updatedBooking.Id);
    existingBooking.ContactInfo = _cipherService.Encrypt(updatedBooking.ContactInfo);
    await _context.SaveChangesAsync();
    return existingBooking;
}
}
```

1.3 Sprint 4

In this sprint for this learning outcome, I investigated GDPR which has information about How, where, and what data I store and how I take into consideration of GDPR rules in the application and made changes.

When dealing with the users' data it is important to think about:

1. Where I store the data
2. How it is stored
3. How it is handled

Where the data is stored

The data of the users are currently stored in Azure Database in an EU country (Sweden), this is because then EU laws apply which include GDPR rules.

The reason I chose to store the database in Azure is that according to them:

"Microsoft has an enduring commitment to protect data privacy, not as an afterthought, but built into Microsoft Azure from the ground up. Microsoft designed Azure with industry-leading security controls, compliance tools, and privacy policies to safeguard your data in the cloud, including the categories of personal data identified by the GDPR."

This is very important and the reason I chose to do Azure is also the personal data stored on Azure are according to their website:

- Well managed and protected from data breaches according to GDPR.
- The Azure Database for MySQL service uses the FIPS 140-2 validated cryptographic module for storage encryption of data-at-rest.
- Data, including backups, are encrypted on disk, including the temporary files created while running queries.
- Azure Database for MySQL takes backups of the data files and the transaction log.
- The Azure Database for MySQL service provides a guaranteed high level of availability with the financially backed service level agreement (SLA) of 99.99% uptime.

How it is stored

The personal data that is stored is email addresses. The decision was to not store more personal information than that because it is not necessary for the current solution, and because of GDPR rules the application tries to limit and minimize the amount of personal data that is stored by the users.

This is a GDPR rule of Limitation of purpose, data, and storage and mandate that no personal data, other than what is necessary, be requested.

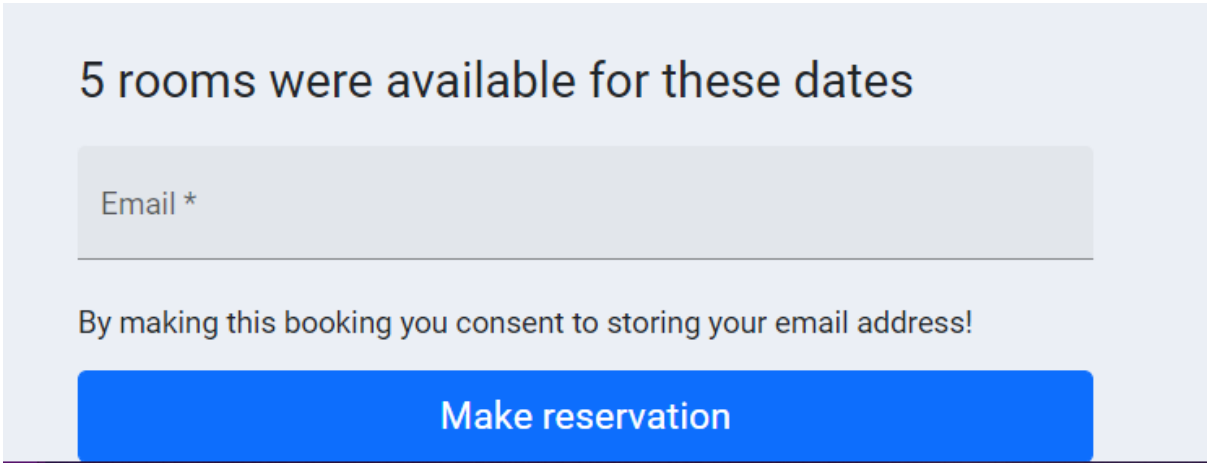
The data that is stored is also encrypted in the database to keep it safe and protected.

GDPR checklist

For more knowledge, I went through the GDPR checklist, and the personal data is according to GDPR rules handled with:

- Personal data is identified and processed in a fair way which means that I do not process data for any purpose other and the legitimate purposes.
- Personal data can be updated but also deleted from my system.
- Transparent means that I must inform data subjects about the processing activities on their personal data.
- I followed privacy by design where all sensitive data is protected and functionalities exist in my system to comply with GDPR.
- Personal data is also only saved and stored in Azure, this means it is clear where the data is stored in case it needs to be updated or removed from the system.

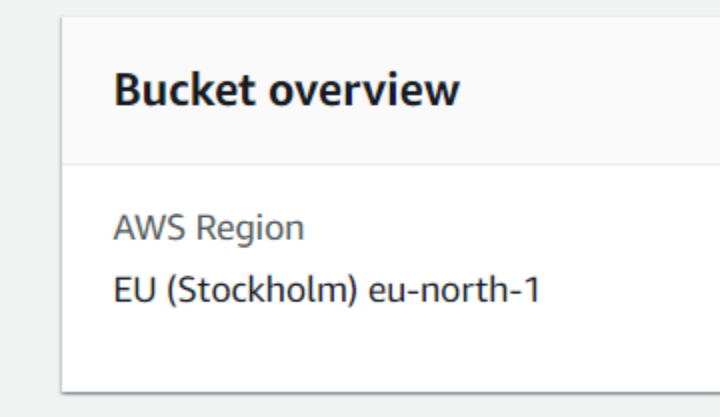
On my website, I tell them what I store so I inform the users, and finally I also hash their information to protect privacy.



1.4 Final sprint

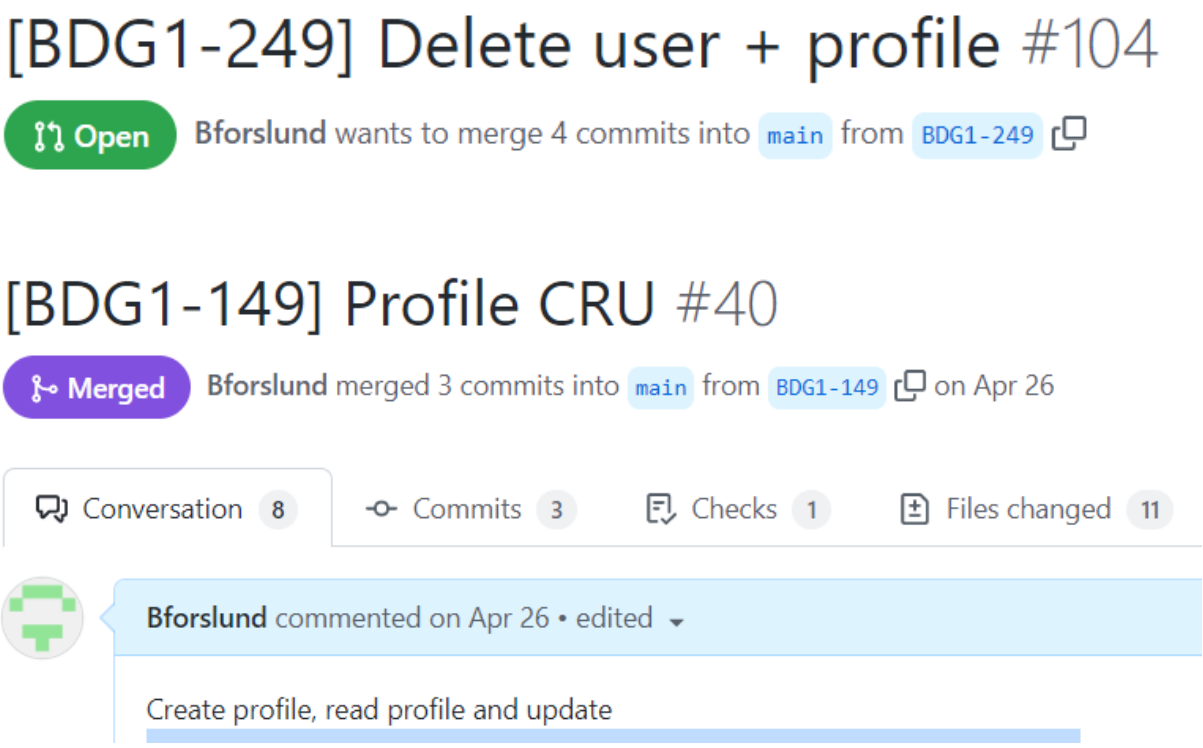
AWS S3 bucket for pictures

I also added pictures, that were stored in a safe place on Amazon and in Europe, the reason for that is GDPR.



Group project (CRUD)

For the final sprint, I made it possible to delete your account in the group project so that GDPR can be taken into consideration. In previous sprints, I implemented it so that it was possible to update your information.



Event sourcing

This is another thing I first heard about at the dotnet conference I went to. After hearing about it, I researched it on my own and I realized it applies to my individual project.

When 1000+ users try to book the same room at the same time a problem arises with concurrency in data access. This means there could be conflicting write/update, concurrent write/update, and concurrent reads.

The solution is an Event Sourcing strategy with Optimistic concurrency control, which is a good solution to deal with highly concurrent environments. Thus, I implemented it where it was needed, where the user book a room.

My model and my controller went from this:

```
6
7   namespace HotelService.Models
8   {
        7 references
9       public class Room
10      {
11
12          private string roomType;
13
            0 references
14          public int Id { get; set; }
15
            5 references
16          public string RoomType { get => roomType; set => roomType = value; }
17
            0 references
18          public Room(string roomType)
19          {
20              RoomType = roomType;
21          }
22
            0 references
23          public Room()
24          {
25
26          }
27
28      }
29  }
30
```

```
61      [HttpPost]
62      [Route("addRooms/{hotelId}")]
        0 references
63      public async Task<ActionResult> AddRoomsAsync(int hotelId, Room room)
64      {
65          var hotel = await _context.Hotels.Where(a => a.Id == hotelId).FirstOrDefaultAsync();
66          hotel.Rooms.Add(room);
67          await _context.SaveChangesAsync();
68          await _messagePublisher.PublishMessageAsync("AddRoom", new AddRoomEvent(hotel.Id, room));
69          return Ok();
70      }
71
72      [HttpGet]
73      [Route("rooms/{hotelId}")]
        0 references
74      public async Task<ActionResult> GetAllRoomsOfHotelAsync(int hotelId)
75      {
76          var hotel = await _context.Hotels.Where(a => a.Id == hotelId).Include(h => h.Rooms).FirstOrDefaultAsync();
77          return Ok(hotel.Rooms);
78      }
79
```

To this:

```
rvice                                                    HotelService.Controllers.RoomController                    AddRoomAsync(Int
        [HttpPost]
        0 references
        public async Task<ActionResult> AddRoomAsync(int hotelId, RoomProjection room)
        {
            var hotel = await _context.Hotels.Where(a => a.Id == hotelId).FirstOrDefaultAsync();

            hotel.Rooms.Add(room);
            await _context.SaveChangesAsync();

            var rooms = new Room(room.RoomNumber, room.RoomType);
            _context.RoomEvents.Add(EventContainer.Create($"hotel:{hotelId}:room:{room.RoomNumber}", 1, rooms.PendingChanges[0]));
            await _context.SaveChangesAsync();

            await _messagePublisher.PublishMessageAsync("AddRoom", new AddRoomEvent(hotel.Id, room));

            return Ok(room);
        }

        [HttpPost]
        [Route("{roomNumber}/checkIn")]
        0 references
        public async Task<ActionResult> CheckinAsync(int hotelId, int roomNumber)
        {
            var savedEvents = await _context.RoomEvents.Where(e => e.Aggregate == $"hotel:{hotelId}:room:{roomNumber}").ToListAsync();
            if (savedEvents.Count == 0)
            {
                throw new ArgumentException($"Room {roomNumber} does not exist at hotel {hotelId}");
            }

            var room = new Room(savedEvents.Select(e => e.Deserialize()));

            room.RoomIsOccupied();

            var expectedVersion = room.Version;

            foreach (var @event in room.PendingChanges)
            {
                _context.RoomEvents.Add(EventContainer.Create($"hotel:{hotelId}:room:{roomNumber}", ++expectedVersion, @event));
            }

            await _context.SaveChangesAsync();

            savedEvents = await _context.RoomEvents.Where(e => e.Aggregate == $"hotel:{hotelId}:room:{roomNumber}").ToListAsync();
            room = new Room(savedEvents.Select(e => e.Deserialize()));

            var roomProjection = await _context.RoomProjections.FirstAsync(r => r.RoomNumber == roomNumber);
            roomProjection.Available = room.Available;
            await _context.SaveChangesAsync();

            return Ok(roomProjection);
        }
```

	Id	Aggregate	Version	EventType	EventData
Edit Copy Delete	1	hotel:1:room:120	1	HotelService.EventStore.RoomRegistered	{"Available":true,"Timestamp":"2022-06-11T17:14:32...
Edit Copy Delete	2	hotel:1:room:121	1	HotelService.EventStore.RoomRegistered	{"Available":true,"Timestamp":"2022-06-11T17:14:38...
Edit Copy Delete	3	hotel:1:room:122	1	HotelService.EventStore.RoomRegistered	{"Available":true,"Timestamp":"2022-06-11T17:14:48...
Edit Copy Delete	4	hotel:1:room:130	1	HotelService.EventStore.RoomRegistered	{"Available":true,"Timestamp":"2022-06-11T17:15:43...
Edit Copy Delete	5	hotel:1:room:121	2	HotelService.EventStore.RoomOccupied	{"Available":false,"RoomNumber":121,"Timestamp":"2...
Edit Copy Delete	6	hotel:1:room:121	3	HotelService.EventStore.RoomAvailable	{"Available":true,"Timestamp":"2022-06-11T17:17:06...
Edit Copy Delete	7	hotel:1:room:122	2	HotelService.EventStore.RoomOccupied	{"Available":false,"RoomNumber":122,"Timestamp":"2...
Edit Copy Delete	8	hotel:1:room:130	2	HotelService.EventStore.RoomOccupied	{"Available":false,"RoomNumber":130,"Timestamp":"2...

I have added room events, which integrates with the Room model. I also have a Room projection Model which is the read model, which only stores the latest state.

Now I store all the events, which means that before I book a room I also check if we are at the latest version, check if the booking exists already and then book the room. If the booking already exists, or the version number is wrong then it will deny.

1.5 Distributed data LO Conclusions

GDPR

For this learning outcome on GDPR and legal requirements I have done:

- Possibility to access, update or delete personal data.
- Possibility to delete pictures.
- Storing the database in the EU.
- Storing pictures in the EU.
- Deploying the cluster in the EU.
- Transparent what I store.
- Data is stored protected on Azure and on Amazon AWS.

Data Quality

For this learning outcome on Data quality and distributed data I have done:

1. Completeness: For data completeness I am making sure that all required fields are marked as required. Both the frontend and the backends will force data completeness on required information. This ensures that the database will not contain unnecessary empty or null fields.

2. For Consistency: I am using Eventual consistency with event sourcing strategies, this means that the data is not at all times in sync. However, the data is always consistent for the data owner and any cached data is not allowed to be used as the source of truth. This ensures data consistency in an eventual consistent and event driven system.

3. Conformity: With the entering of the email, it is only possible to provide a valid email. Data should be stored as UTC and shown to the users in their local time zone.

4. Accuracy: The date, which is taken from the date picker is a real date on the calendar.

5. Integrity: I am using a SQL database, and ORM makes the relationships between the data.

6. Timeliness: The application uses eventual consistency and caching and therefore does not provide real-time accurate data to all users. This is however fine in the given context of a hotel booking site. The Redis Cache for caching hotel information can be out of date for 1 minute before cache invalidation takes place, it is therefore unlikely that a user will receive wildly inaccurate hotel information within that timeframe. Eventual consistency as applied in the booking of rooms might mean that a room seems to be available when a user tries to make a booking, but before the confirmation is received the data will be consistent enough to be able to decide if the booking can go through or not. This is also sufficient timeliness within the context of where it is applied. All other data is provided real-time from the database.

Since I applied my knowledge in several situations both in the individual and in the group project, this means I have **advanced** for this learning outcome.

Reflection

Sprint 1:

Our teacher had improvements on our standup and our standup is now going smoother. Our overall communication and teamwork with the group are also improving since we had a few different opinions and talked to the technical teachers about it. To improve to the next level of beginning is just keep collaborating with the group, give my opinions, and have good efficient standups and meetings.

About the context-based research, I feel it is going in the right direction. Since I just drew some questions, I believe I am just on Orienting level. To get to the next level I believe starting to research and answer one question is enough.

About the lifelong learning, In this sprint, I had to think about what I want to do in the future and find suitable internships and companies. This brought me closer to what I want to do with my life after school.

About the technical learning outcomes: It was hard in the beginning because all the team members have different experiences and knowledge. However, in the group project we ended up agreeing on certain things and the sketch was made.
On the individual, I got the feedback from my technical teacher to research more about event-driven architecture and refine my design, and then also implement it. This should take me to a beginning level.
I also am learning more about security, for example by also following workshops and researching but for the next level, I need to start applying security into the application.

Sprint 2:

About the first three learning outcomes:

I have learned how to talk to my team members about the changes to the application and they do listen. I also learned together with my team group how to do proper standups and we have created a nice backlog. However, we did not manage to deliver everything we promised to the client.

It is going well with researching the questions, and I do write down my findings/conclusions or answers to the questions. In the group project, we realized that we needed to change questions in the group project because the questions were overlapping, and I think that made me understand more what good research questions are as well as the feedback from my semester coach helped. To get to the next level I believe most questions should be answered.

About the technical learning outcomes:

I am understanding the purpose of Microservices and message busses quite well now after adding them to both projects. I studied the theory and started applying the ideas to the projects and now my application is very scalable, but still need a solution for the database and whether it should be cached. In the group project, I think we are starting to add more microservices and dividing the team to work on which microservices.

In this sprint, I started understanding what DevOps is by researching it. I also applied it by adding a Ci/CD pipeline which is a part of DevOps. Therefore I believe I am on level orienting, but once I have deployed stage to the pipeline it will be beginning.

At first, I did not write anything for cloud services, but after talking to my teachers and teammates I did more for this learning outcome than I thought. I did not know that Auth0 was a cloud provider, but after researching it, I

know now. I also looked up how to use a cloud deployment since the team wanted to try to deploy the application last minute so instead of working on functional requirements, I had to research how the cloud works and that's why I believe I should be on an Orienting level for this sprint.

Sprint 3:

I think we finally learned how to use story points, and we are also promising enough tasks for the client, and we can deliver what we promise which is a good thing. So, I believe our team is finally proficient and this did get confirmed by the semester coach.

Sprint 4:

This sprint was smooth, I got a lot of things done and I believe I should have proficient for all the learning outcomes now hopefully next sprint I can get some of them advanced.

What I struggled with this sprint was setting up Argo and updating the YAML files so that Argo gets triggered and deploys it to azure, but I managed to get it working.

Another problem was figuring out what exactly is needed to get done for Distributed data, but I checked the checklist and wrote about that and made some changes to my application that was necessary to take into consideration of the GDPR rules.

Sprint 5:

I handed in my portfolio during the last sprint, and I got proficient in all of the learning outcomes except monitoring. Since I knew that I passed the semester I felt a lot more relaxed since that was my biggest concern. I also went to a dotnet conference where I learned so much and I realized I never put so much effort into my portfolio, and I did not show what I was capable of or that I knew more than I wrote down. An example is Stryker, something I do in my free time but never thought of adding to my portfolio.

So, I wanted to change that, and show what I could do and that I have an advanced on most learning outcomes. I decided to add a lot of tools, and features, and update everything for the better. I am finally proud of my software, and I do believe I did my best. Hopefully, the teachers agree, and I can get an outstanding for the semester.

Conclusion

My time this semester was great, and I learned a lot of things.

The end result is something I am very proud of, and I achieved more than I thought was possible.

At the beginning of the semester, I barely knew that microservices existed. It was a goal of mine to learn more about microservices this semester. I believe I achieved this goal and that I have expanded my understanding of microservices. I think all subjects this semester were very interesting and I enjoyed learning more about them. I am especially interested in learning more about distributed data, event sourcing and scalability.

Another goal was to learn more about blockchain. I am happy that I could research that, and it was very interesting to learn about. I do wish I could have made the research more towards the learning outcomes instead. For example, I had a research question "how can blockchain be tested", but I wish maybe I had a research question about blockchain and scalability. I believe the reason for that is that I did not yet know what scalability was at the time.

What I found challenging at the beginning of the semester was, when the teachers were asking me what kind of platform I was going to create, and what will happen if 100 000 users use my platform at the same time. This was something I had no clue about and that whole conversation about load were like another language to me.

I also found it challenging with the team in the beginning since most of my team members already knew about microservices and API gateways. They wanted to start right away by making a final architecture design in the first weeks. That is where we had the most conflicts in the beginning since I had no clue what a microservice was and I did not agree that we should make the final architecture design so soon.

Now I can look back and see what they all meant. I can answer those questions the teachers asked me about my platform, and I can contribute to my team members when discussing architecture choices.

It was very hard, in the beginning, to not have a set of requirements or classes to guide the semester. I come from course-based where there is a lot more structure to the semester in the form of assignments and classes related to the project. This meant I did not think to research how to implement for example images, or where to store images. I only thought to implement this in the last sprint. It also meant that I was a bit stressed over not knowing what to implement next. I wish I researched a lot of the things sooner. This would have meant I could have had more time to research and implement the details and specifics of the things I implemented now.

To conclude, I have learned that I **can** do much more than I believe I can. I think since this was such a great learning experience and I have learned so many new things, considering the level I had at the beginning of the semester I believe I have an outstanding for the semester.