Emil Hellgren
A2103167

Learning Diary
R504TL114-3001

8.2.2024 – 15.5.2024

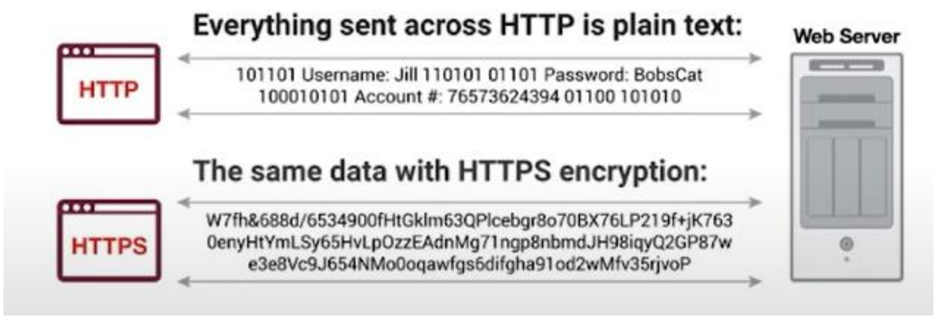
This learning diary goes through the following subject areas (based on Lapland UAS Cloud Computing):

Course organization and theory
Research in Cloud Computing
Expanding Research Cloud Computing
Cloud-based Frontend Deployment
Cloud Deployment Foundations
Advanced Cloud Deployment Strategies
Cloud Databases
Ethics, Energy, Costs, and Security in Cloud Computing

8/9/10 February 2024, Module 1, Course theory

Terminology.

HTTP / HTTPS
- Hypertext Transfer Protocol (/ Secure)
- HTTPS encrypted, HTTP plain text

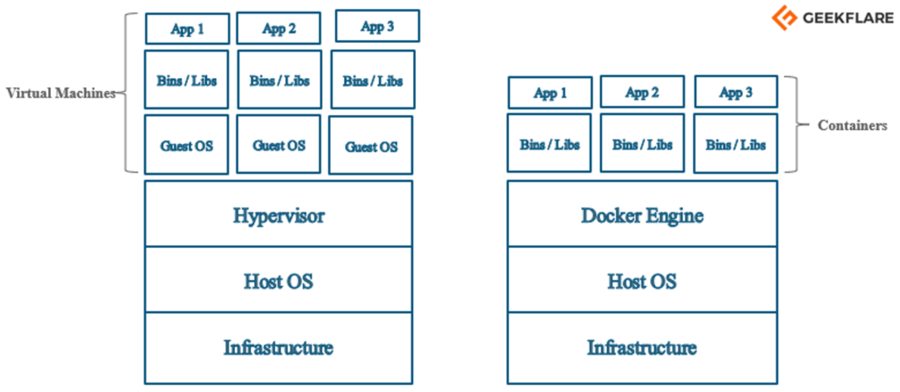

Everything sent across HTTP is plain text:
HTTP
101101 Username: Jill 110101 01101 Password: BobsCat
100010101 Account #: 76573624394 01100 101010
Web Server
The same data with HTTPS encryption:
HTTPS
W7fh&688d/6534900fHtGklm63QPlcebgr8o70BX76LP219f+jK763
0enyHtYmLSy65HvLpOzzEAdnMg71ngp8nbmdJH98iqyQ2GP87w
e3e8Vc9J654NMo0oqawfgs6difgha91od2wMfv35rjvoP

Git
- Version control
- Standard for version control in software development

Pipeline
- Automated process that allows devs and DevOps professionals to efficiently compile, build and deploy.

Virtualization
- Process of creating a virtual version of something
- Virtual computer hardware platforms, storage devices, computer network resources

Containerization
- Lightweight alternative to full machine virtualization that involves encapsulating an application in a container with its own operating environment.



https://geekflare.com/docker-vs-virtual-machine/

Docker
- Containerization platform
- Allows devs to package apps into containers.

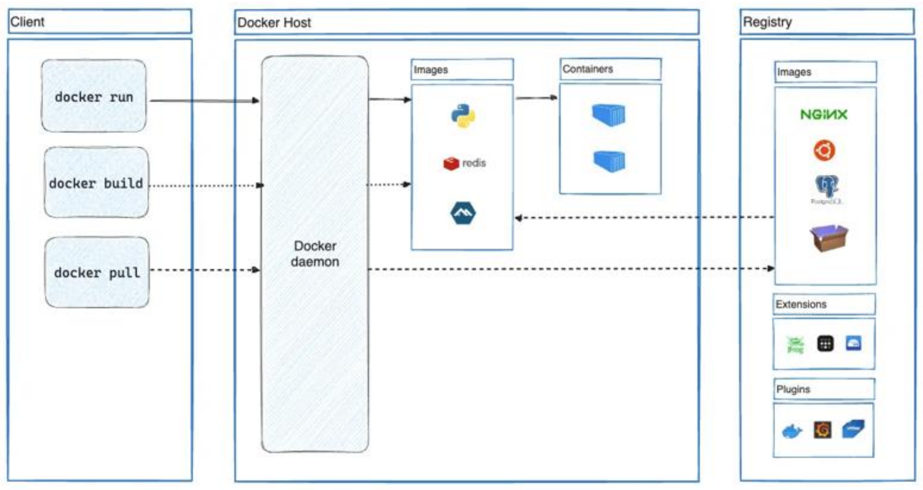
Docker Client: Docker Command Line Interface (CLI)

Docker Host: Docker server (kind a like docker app) that is running

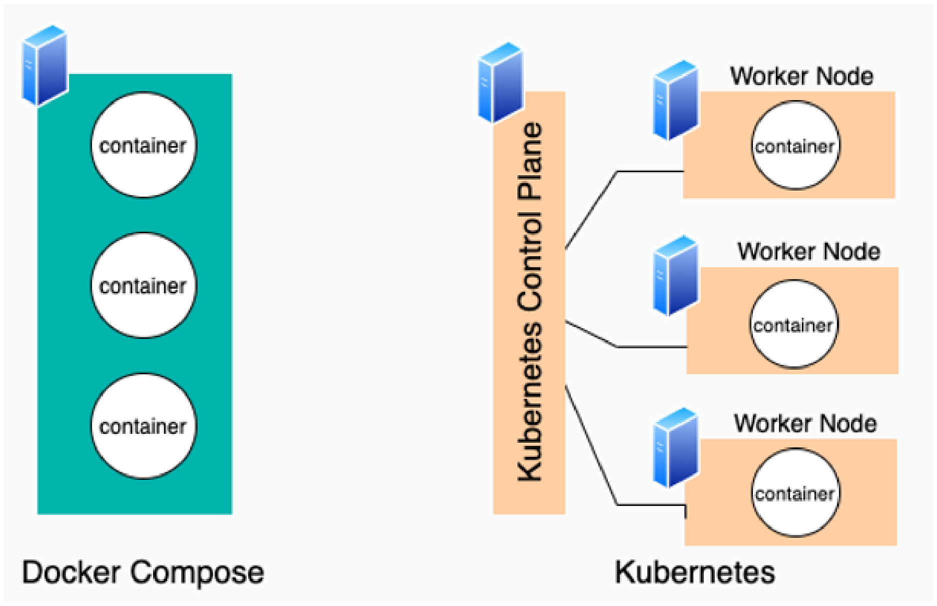
Docker Images: Docker Images are the blueprints of Docker containers. An image is a lightweight, standalone, and executable software package that includes everything needed to run a software application

Docker Containers: A Docker Container is a runtime instance of a Docker image.

Docker Registry: The Docker Registry is where Docker images are stored. The registry can be public or private. The most well-known Docker registry is Docker Hub, which is a public cloud-based registry that allows you to share container images with your team, customers, or the Docker community at large

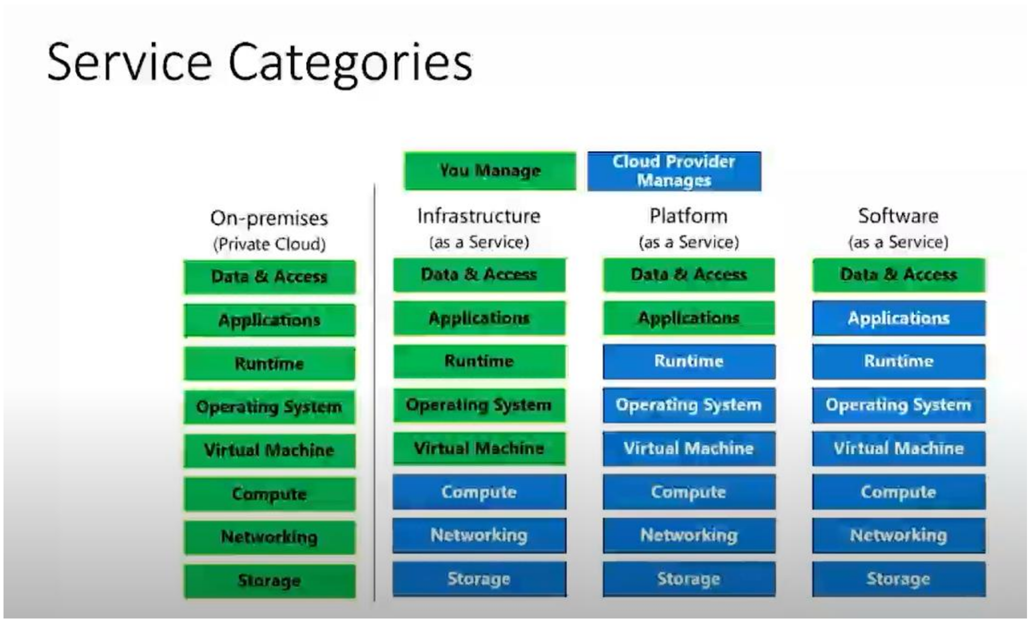


Kubernetes
- Container orchestration platform
- Place to run and manage containers
- Can handle deployment of containers in multiple servers
- Can automatically load balance, deploy the app in the least used server
- Automated Rollouts + Rollbacks, no build breaks so the app is always running.
- Can also scale performance by increasing or decreasing number of backend containers running



Cloud Service Offerings
- Public, Private and Hybrid Cloud
    - Public cloud sells services to anyone on the internet
    - Private cloud is a proprietary network or a data center that supplies hosted services to limited number of people
    - Hybrid cloud is a combination of the previous two
- PaaS (Platform as a Service)
    - Cloud computing services that provide a platform allowing customers to develop, run and manage apps without building and maintaining the infrastructure.

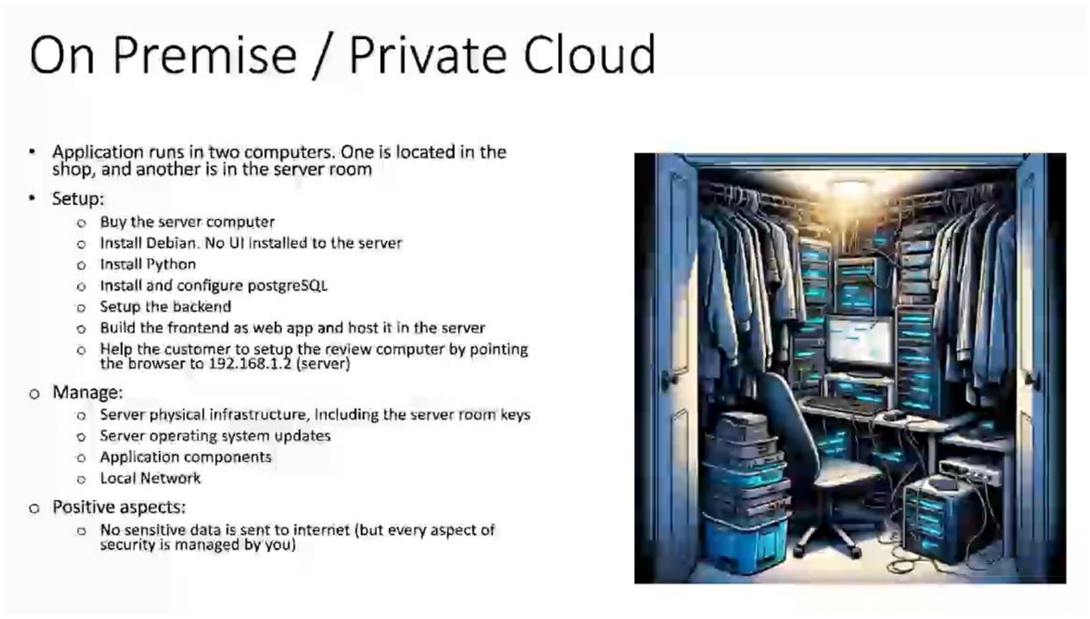
- IaaS (Infrastructure as a Service)
  - Online services that provide high-level APIs used to dereference various low-level details of underlying network infra like physical computing resources, location, data partitioning, scaling, security, backup
- SaaS (Software as a Service)
  - Method of delivering apps over the Internet as a service.
  - Frees users from software and hardware management



Service Categories

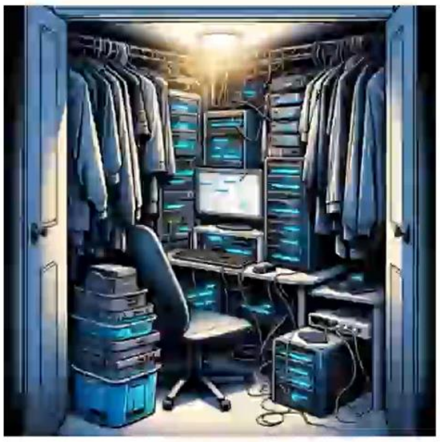
🟩 You Manage 🟦 Cloud Provider Manages

On-premises (Private Cloud)	Infrastructure (as a Service)	Platform (as a Service)	Software (as a Service)
🟩 Data & Access	🟩 Data & Access	🟩 Data & Access	🟩 Data & Access
🟩 Applications	🟩 Applications	🟩 Applications	🟦 Applications
🟩 Runtime	🟩 Runtime	🟦 Runtime	🟦 Runtime
🟩 Operating System	🟩 Operating System	🟦 Operating System	🟦 Operating System
🟩 Virtual Machine	🟩 Virtual Machine	🟦 Virtual Machine	🟦 Virtual Machine
🟩 Compute	🟦 Compute	🟦 Compute	🟦 Compute
🟩 Networking	🟦 Networking	🟦 Networking	🟦 Networking
🟩 Storage	🟦 Storage	🟦 Storage	🟦 Storage



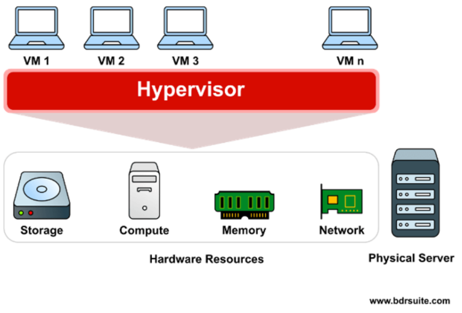
On Premise / Private Cloud

- Application runs in two computers. One is located in the shop, and another is in the server room
- Setup:
  - Buy the server computer
  - Install Debian. No UI installed to the server
  - Install Python
  - Install and configure postgreSQL
  - Setup the backend
  - Build the frontend as web app and host it in the server
  - Help the customer to setup the review computer by pointing the browser to 192.168.1.2 (server)
- Manage:
  - Server physical infrastructure, Including the server room keys
  - Server operating system updates
  - Application components
  - Local Network
- Positive aspects:
  - No sensitive data is sent to internet (but every aspect of security is managed by you)



Infrastructure as a Service (IaaS)

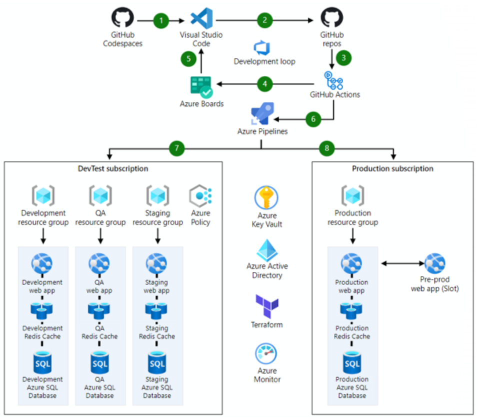
- Application runs in two computers. One is in the shop, and another is in a virtual server from an IaaS provider
- Setup:
  - buy a virtual server. Select Debian to be installed
  - Install Python
  - Install and configure postgreSQL
  - Setup the backend
  - Build the frontend as web app and host it in the server
  - Help the customer to setup the review computer by pointing the browser to https://back.end.fi (virtual server located somewhere)
- Manage:
  - Server operating system updates
  - Application components
- Provider manages:
  - Sever snapshots (periodical backups of the whole setup)
  - Network



https://www.bdrsuite.com/blog/physical-servers-vs-virtual-machines-what-are-the-differences/

Platform as a Service

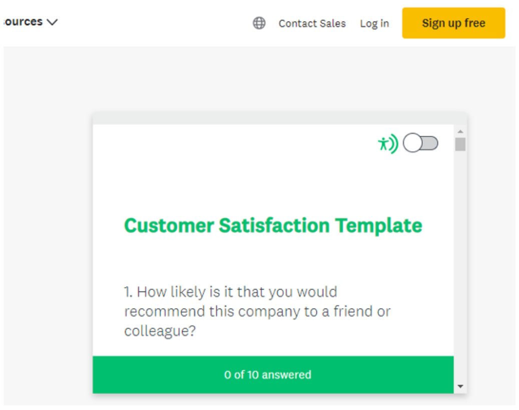
- Application runs in two places. One is located in the shop, and another is in a PaaS provider data center
- Setup:
  - Select Paas-provider and setup the build pipeline. Create python Fast API build configs
  - Connect the backend into the provided/purchased postgreSQL-database
  - Build the pipeline. App is automatically hosted
  - Help the customer to setup the review computer by pointing the browser to https://backend.paasprovider.com
- Manage:
  - Application build pipeline
- Provider manages:
  - Any physical infrastructure
  - Database install, security and backups
  - Build environment



https://learn.microsoft.com/en-us/azure/architecture/solution-ideas/articles/dev-test-paas

Software as a Service

- Buy a subscription to a cloud service that does what we want.
- Setup:
  - Create account in f.ex. https://www.surveymonkey.com/
- Manage:
  - Build the form using the web Ui
- Provider manages:
  - Everything, but access and data

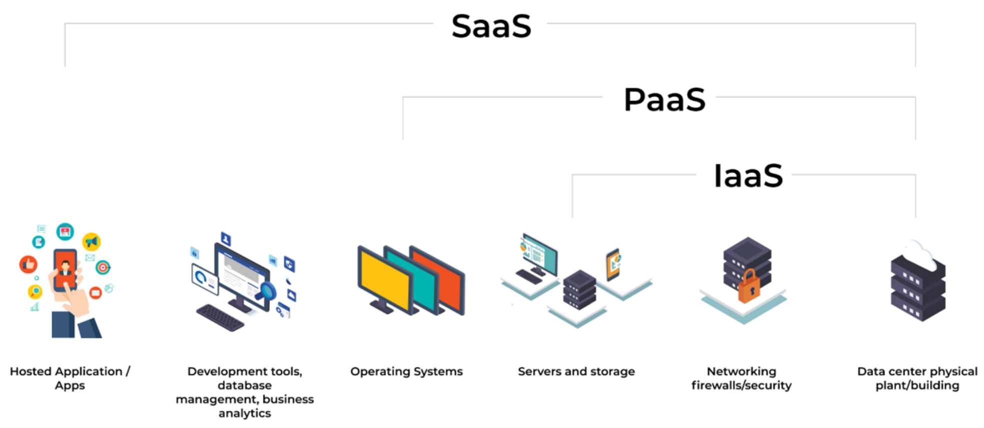


Customer Satisfaction Template

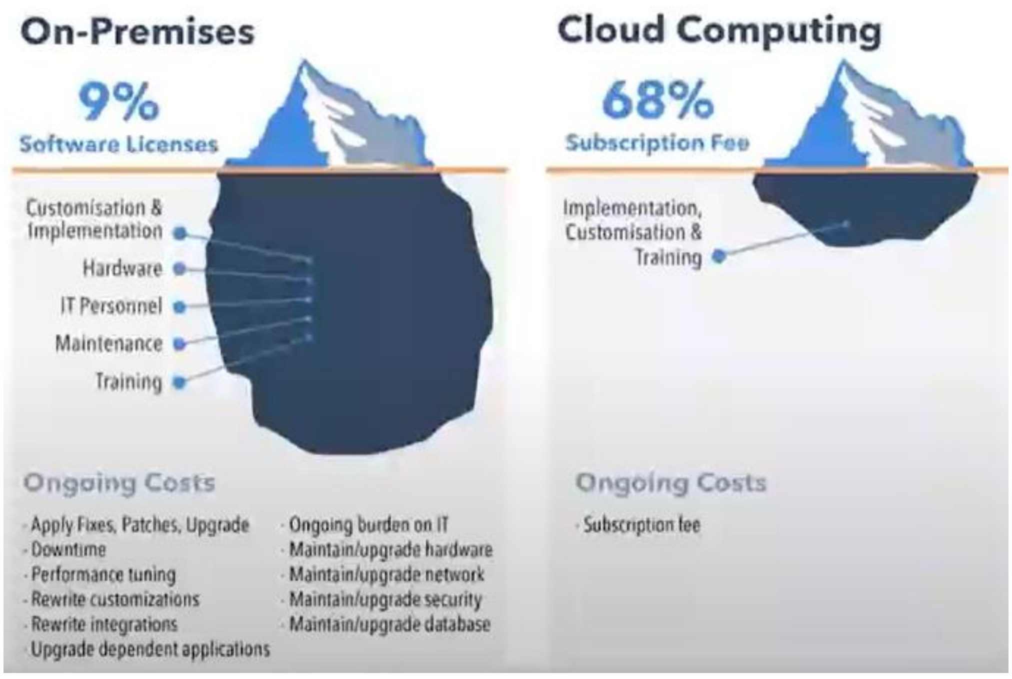
1. How likely is it that you would recommend this company to a friend or colleague?

0 of 10 answered

Recap:


SaaS
PaaS
IaaS
Hosted Application / Apps
Development tools, database management, business analytics
Operating Systems
Servers and storage
Networking firewalls/security
Data center physical plant/building

https://tezhost.com/cloud-computing/iaas-vs-saas-vs-paas/



Cost estimator tool:
https://cloud.google.com/products/calculator?hl=en

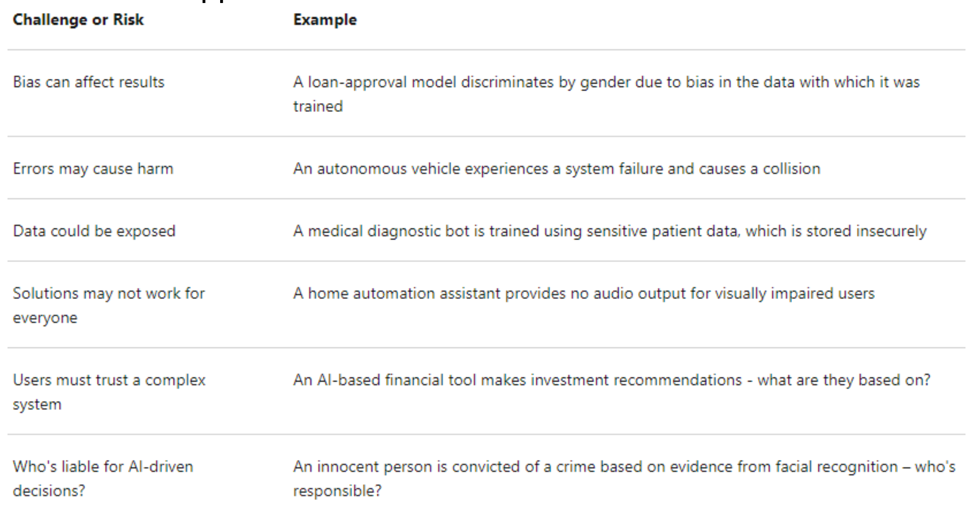
Online learning platforms:

- Free:
  - AWS, Google Cloud, Azure
- P2P:
  - coursera, Udemy, LearnDash

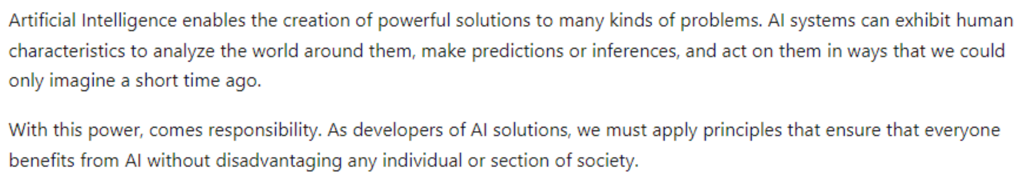
Exercise 1 (Module 1):

- computer vision: image classification, object detection, semantic segmentation, image analysis, face detection analysis, optical character recognition

- knowledge mining: term used to describe solutions that involve extracting information from large volumes of often unstructured data to create a searchable knowledge store
- risks with ai application dev:

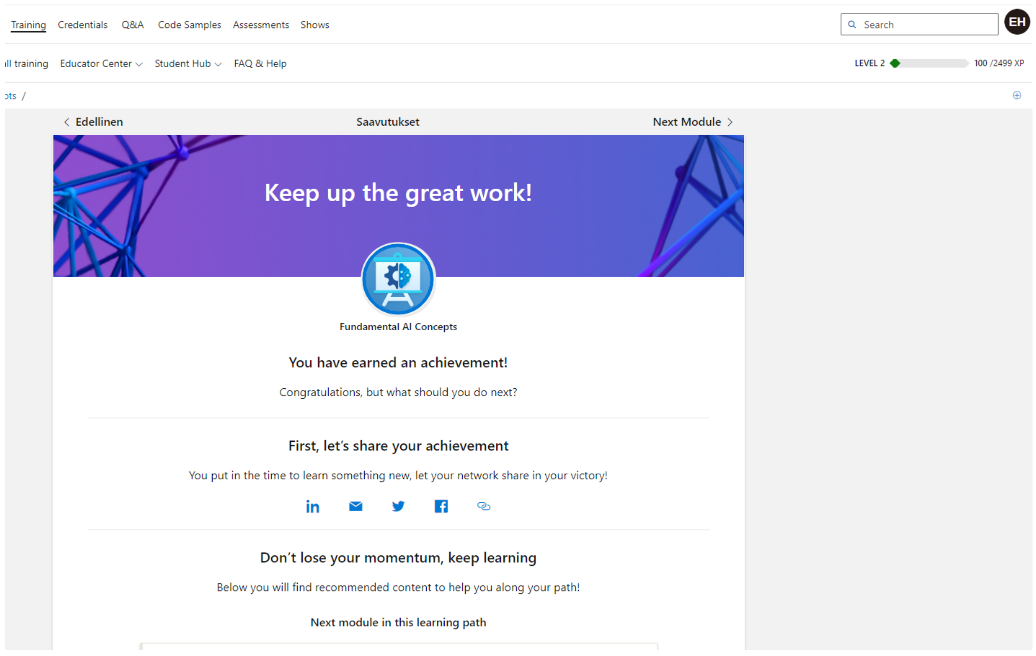


Challenge or Risk	Example
Bias can affect results	A loan-approval model discriminates by gender due to bias in the data with which it was trained
Errors may cause harm	An autonomous vehicle experiences a system failure and causes a collision
Data could be exposed	A medical diagnostic bot is trained using sensitive patient data, which is stored insecurely
Solutions may not work for everyone	A home automation assistant provides no audio output for visually impaired users
Users must trust a complex system	An AI-based financial tool makes investment recommendations - what are they based on?
Who's liable for AI-driven decisions?	An innocent person is convicted of a crime based on evidence from facial recognition – who's responsible?



Artificial Intelligence enables the creation of powerful solutions to many kinds of problems. AI systems can exhibit human characteristics to analyze the world around them, make predictions or inferences, and act on them in ways that we could only imagine a short time ago.

With this power, comes responsibility. As developers of AI solutions, we must apply principles that ensure that everyone benefits from AI without disadvantaging any individual or section of society.



Keep up the great work!

You have earned an achievement!

Congratulations, but what should you do next?

First, let's share your achievement

You put in the time to learn something new, let your network share in your victory!

Don't lose your momentum, keep learning

Below you will find recommended content to help you along your path!

Next module in this learning path

Exercise 2 (Module 1):

Cloud deployment plan for an app with relational database, backend and frontend. Specifying tech stack, cloud provider and IaaS/PaaS.

Tech Stack:
- ReactJS as frontend

- Flask as backend
- MS SQL Server as db

Azure as Cloud provider. Service model would be PaaS to make the load on the Cloud end as light as possible for dev (me).

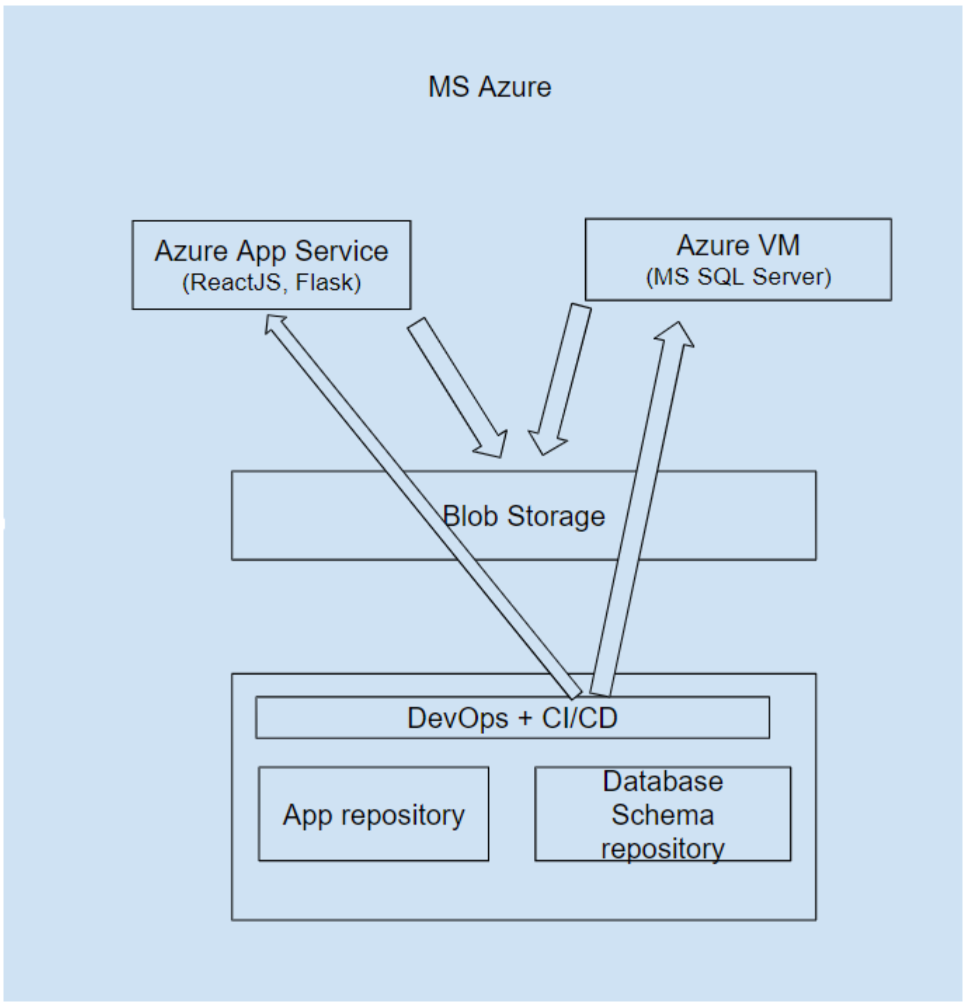
I would use App Service for hosting the front-/backend for the simplified deployment.

Azure VM for hosting my MS SQL server, I want the hands-on control for my database.

Blob Storage for the object / unstructured data storage and backups.

DevOps for the pipeline and version control.

Summasummarum, I use Azure in my everyday development work and I find it logical and somewhat simple to use.



4 March 2024, Module 2

Task:

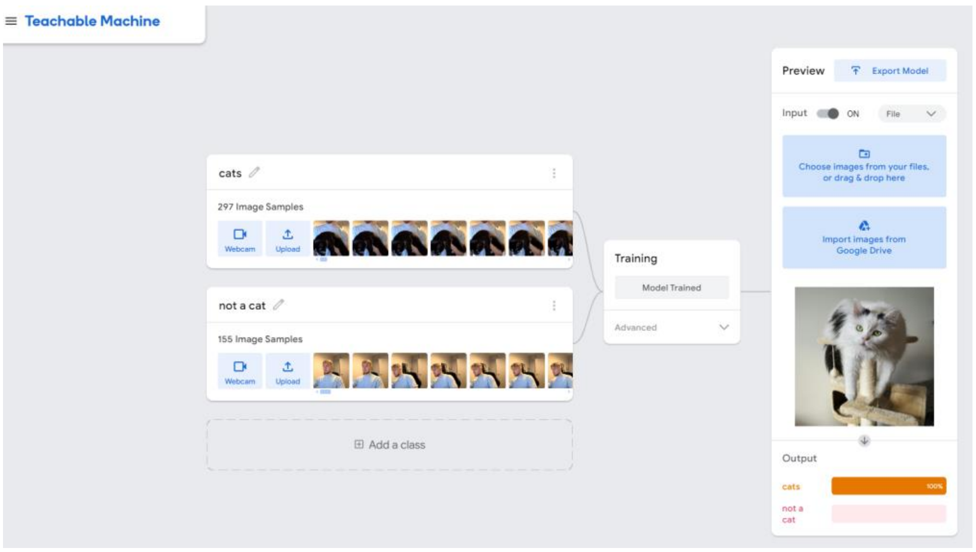
Create a basic ML model using Teachable Machine by choosing a type of model (image, pose, or sound), collecting your own dataset for training, and training your model. Integrate the trained model into an HTML file in a way that it performs a practical function of your choice.

Save in the Google Drive Folder as attachment

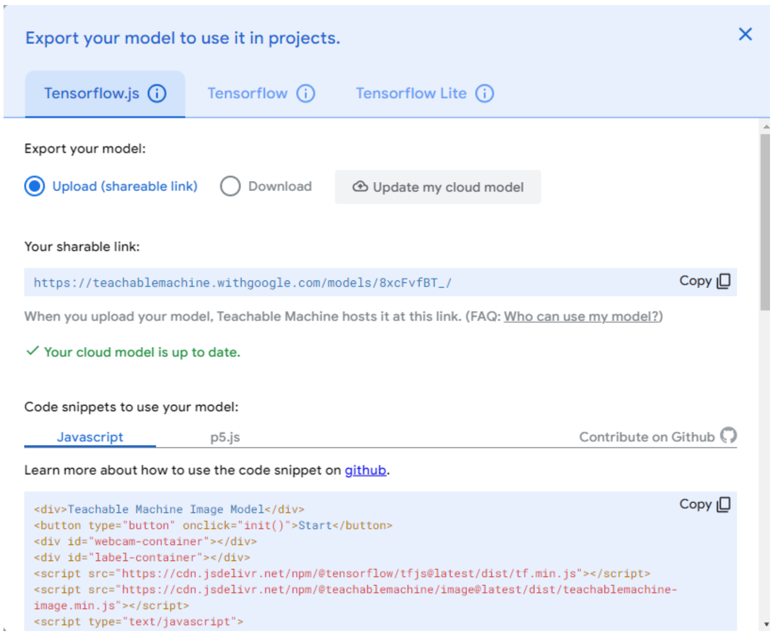
- HTML file with practical implementation

For the task I chose to do an img model, one-pager for helping people to know if they have a cat or not!

I trained the Teachable Machine with 300 cat pictures and 150 human pictures to recognize the difference.



Then I exported the base:


Export your model to use it in projects.

Tensorflow.js | Tensorflow | Tensorflow Lite

Export your model:

[✓ Upload (shareable link)] Download | Update my cloud model

Your sharable link:

`https://teachablemachine.withgoogle.com/models/8xcFvfBT_/` Copy

When you upload your model, Teachable Machine hosts it at this link. (FAQ: Who can use my model?)

✓ Your cloud model is up to date.

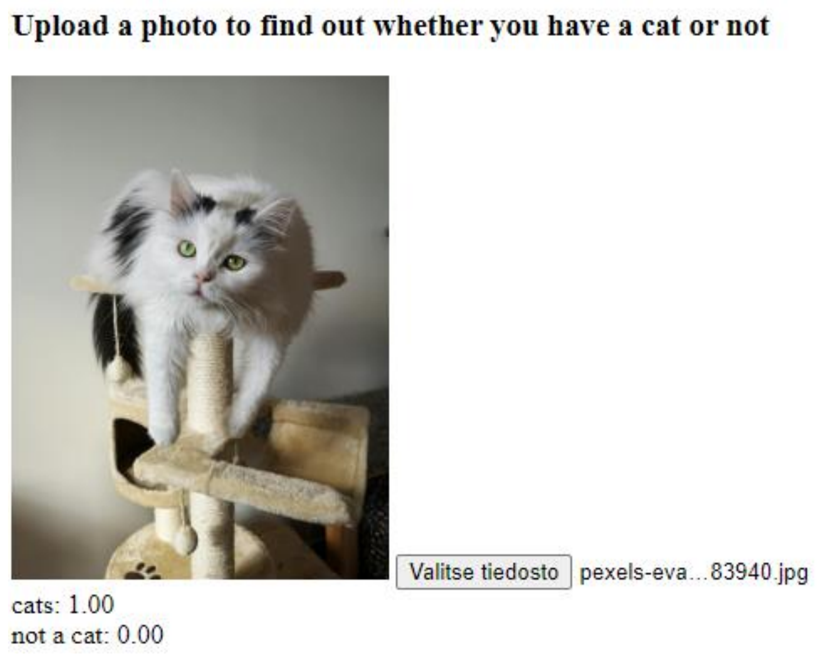
Code snippets to use your model:

Javascript | p5.js | Contribute on Github

Learn more about how to use the code snippet on github.

```
<div>Teachable Machine Image Model</div>
<button type="button" onclick="init()">Start</button>
<div id="webcam-container"></div>
<div id="label-container"></div>
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablemachine-
image.min.js"></script>
<script type="text/javascript">
```
Copy

Added btn upload and the project was ready. End result:


Upload a photo to find out whether you have a cat or not

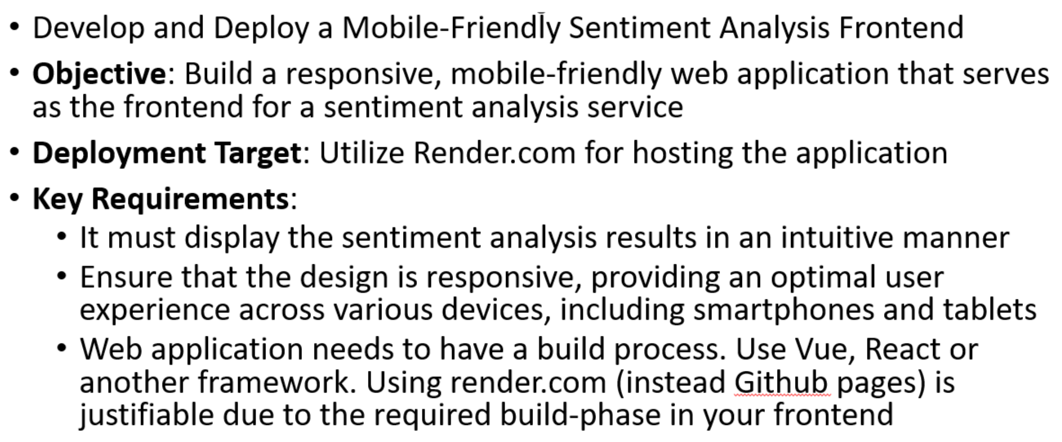
cats: 1.00
not a cat: 0.00

Source-code can be found in my Github:

https://github.com/ehellgre/cloud-computing-2024/tree/master/ml-image-model

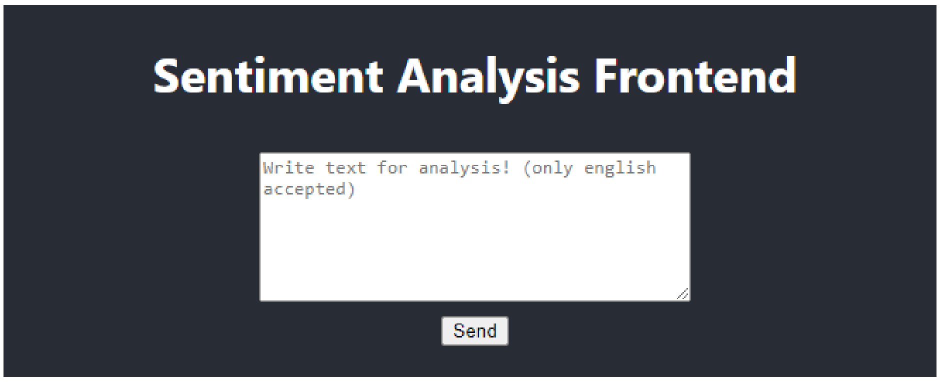
13-14 May 2024, Module 4

Main Task:


- Develop and Deploy a Mobile-Friendly Sentiment Analysis Frontend
- **Objective**: Build a responsive, mobile-friendly web application that serves as the frontend for a sentiment analysis service
- **Deployment Target**: Utilize Render.com for hosting the application
- **Key Requirements**:
  - It must display the sentiment analysis results in an intuitive manner
  - Ensure that the design is responsive, providing an optimal user experience across various devices, including smartphones and tablets
  - Web application needs to have a build process. Use Vue, React or another framework. Using render.com (instead Github pages) is justifiable due to the required build-phase in your frontend

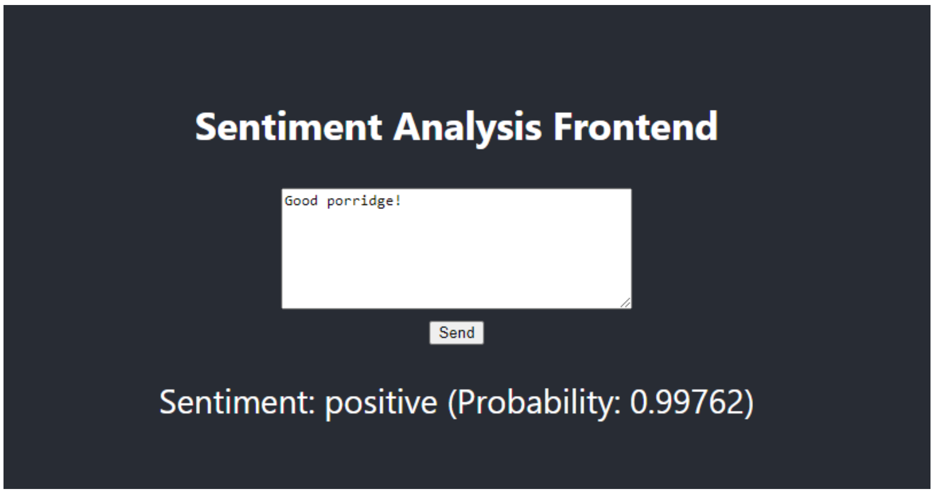
I quickly developed the needed frontend using React. I also used a public api for the sentiment analysis (https://rapidapi.com/symanto-symanto-default/api/sentiment-analysis9/).

Result frontend:


Sentiment Analysis Frontend

Write text for analysis! (only english accepted)

=>

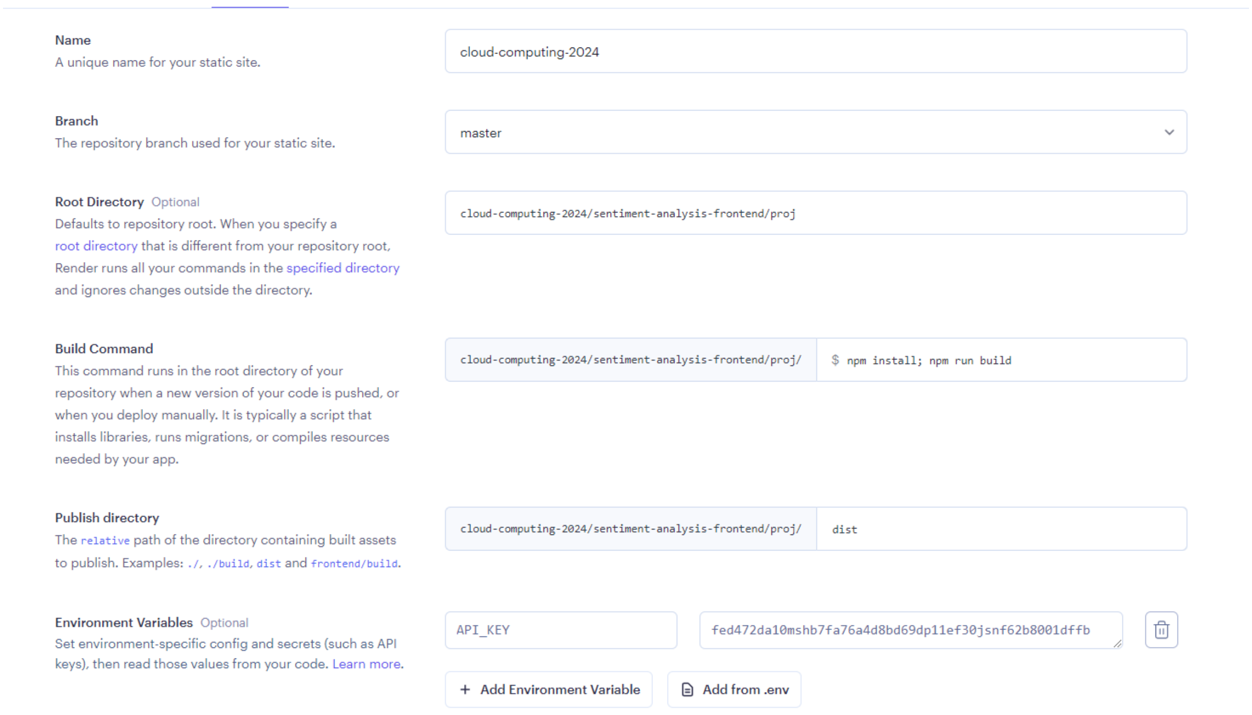

Sentiment Analysis Frontend

Good porridge!

Sentiment: positive (Probability: 0.99762)

Source code again visible in my Github repo: https://github.com/ehellgre/cloud-computing-2024/tree/master/sentiment-analysis-frontend

Setting up the deployment:


Name
A unique name for your static site.

cloud-computing-2024

Branch
The repository branch used for your static site.

master

Root Directory Optional
Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.

`cloud-computing-2024/sentiment-analysis-frontend/proj`

Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.

`cloud-computing-2024/sentiment-analysis-frontend/proj/` `$ npm install; npm run build`

Publish directory
The `relative` path of the directory containing built assets to publish. Examples: `./`, `./build`, `dist` and `frontend/build`.

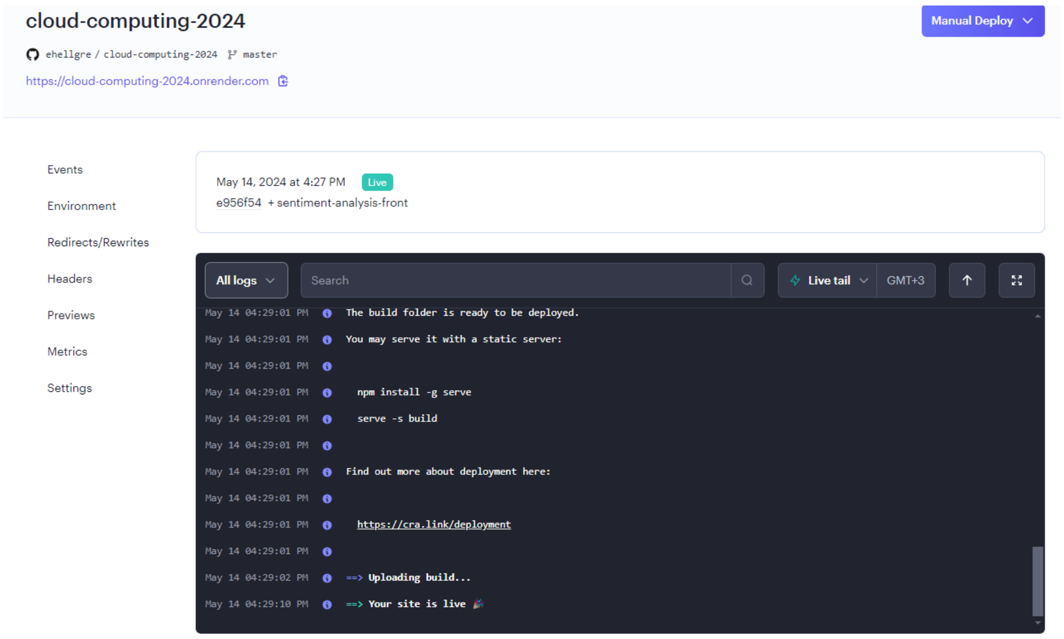
`cloud-computing-2024/sentiment-analysis-frontend/proj/` `dist`

Environment Variables Optional
Set environment-specific config and secrets (such as API keys), then read those values from your code. Learn more.

`API_KEY` `fed472da10mshb7fa76a4d8bd69dp11ef30jsnf62b8001dffb`

First build failed, correct publish dir should be build instead of dist.

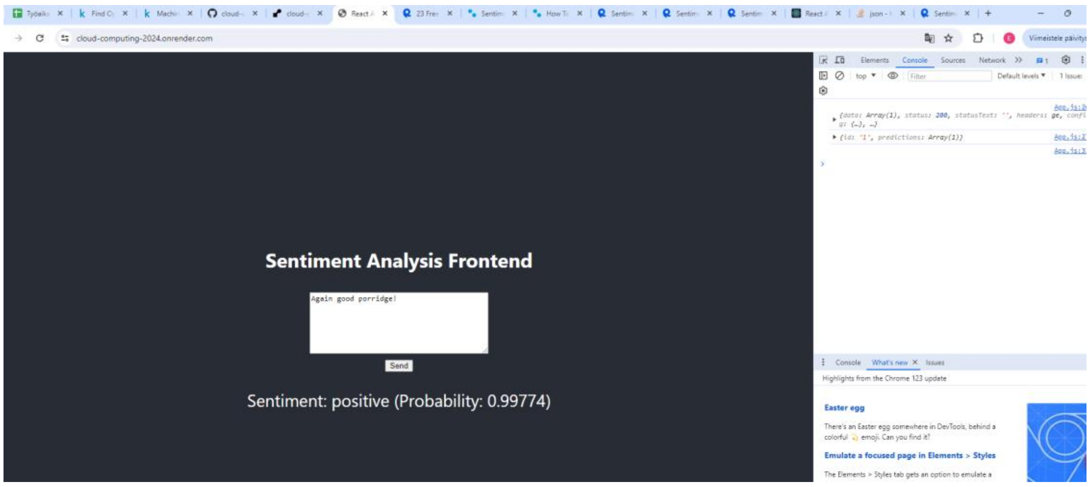
After that change, build succeeded!


cloud-computing-2024
Manual Deploy
ehellgre / cloud-computing-2024 master
https://cloud-computing-2024.onrender.com
Events
Environment
Redirects/Rewrites
Headers
Previews
Metrics
Settings
May 14, 2024 at 4:27 PM Live
e956f54 + sentiment-analysis-front
All logs
Search
Live tail
GMT+3

```
May 14 04:29:01 PM  The build folder is ready to be deployed.
May 14 04:29:01 PM  You may serve it with a static server:
May 14 04:29:01 PM
May 14 04:29:01 PM    npm install -g serve
May 14 04:29:01 PM    serve -s build
May 14 04:29:01 PM
May 14 04:29:01 PM  Find out more about deployment here:
May 14 04:29:01 PM
May 14 04:29:01 PM    https://cra.link/deployment
May 14 04:29:01 PM
May 14 04:29:02 PM  ==> Uploading build...
May 14 04:29:10 PM  ==> Your site is live 🎉
```

Bugfix: API_KEY changed to REACT_APP_API_KEY

Deployed result:


cloud-computing-2024.onrender.com
Elements
Console
Sources
Network
top
Filter
Default levels
1 Issue:
Sentiment Analysis Frontend
Again good porridge!
Send
Console
What's new
Issues
Highlights from the Chrome 123 update
Easter egg
There's an Easter egg somewhere in DevTools, behind a colorful emoji. Can you find it?
Emulate a focused page in Elements > Styles
The Elements > Styles tab gets an option to emulate a

```
▸ {data: Array(1), status: 200, statusText: '', headers: ge, config: {…}, …}
▸ {id: '1', predictions: Array(1)}
```

Sentiment: positive (Probability: 0.99774)

13-14 May 2024, Module 5

Tasks:

Main task

- Develop a sentiment analysis backend in Python. Create a GitHub to CSC OpenShift pipeline using a webhook. Host the backend on CSC Rahti OpenShift using the pipeline, ensuring it's accessible over HTTPS. Modify the frontend from Module 4 to integrate with this new sentiment analysis backend, using HTTPS for all connections. Reflect on, what you have learned about deployment
- Additionally, as a refresher on Docker, create a Dockerfile and a docker-compose file. Run the backend locally using Docker. Reflect on the Docker process in your learning diary. Ensure that your working docker-compose file and Dockerfile are included in the repository when you submit the link to the repository

Follow-Up Task

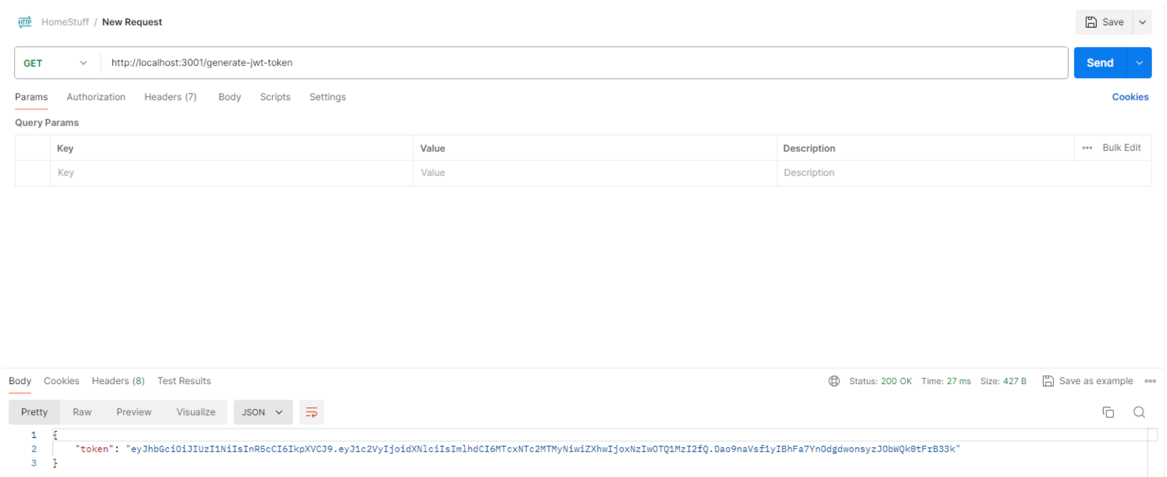
- Focus on security. Add token-based or JWT authentication to the backend. Briefly reflect on the security aspects of the backend.

First I created a Node.js server using express.

Then I skipped straight to the follow-up task and created a simple jwt-token based auth. Users must know this static bearer token to access the api.

Generation of the token:

```
app.get('/generate-jwt-token', (req, res) => {
    const payload = { user: "user" };
    const token = jwt.sign(payload, JWT_TOKEN, { expiresIn: '60d' });
    res.json({ token });
});
```



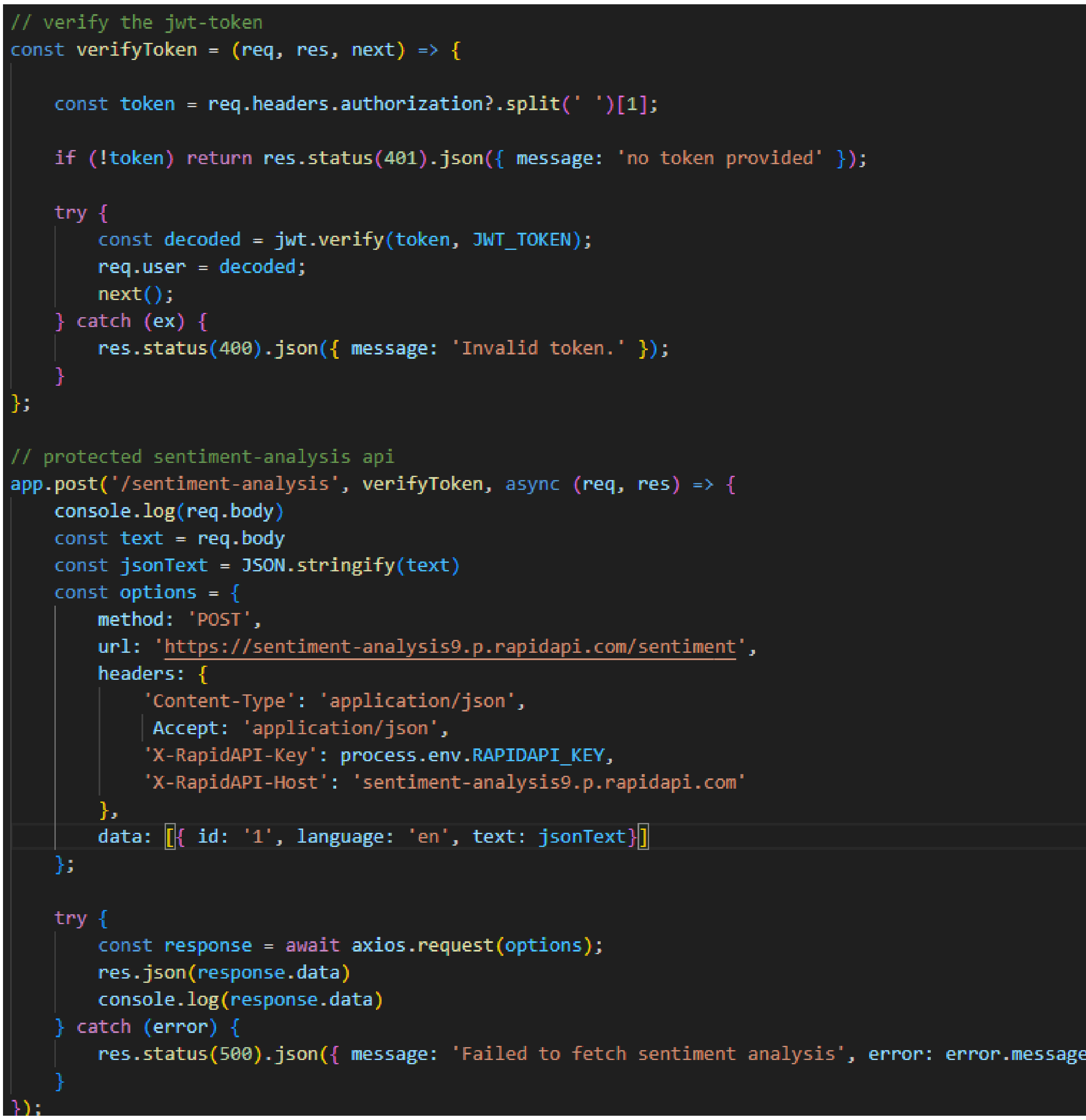
GET http://localhost:3001/generate-jwt-token

Key	Value	Description
Key	Value	Description

Status: 200 OK Time: 27 ms Size: 427 B

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoidXNlciIsImlhdCI6MTcxNTc2MTMyNiwiZXhwIjoxNzIwOTQ1MzI2fQ.Dao9naVsf1yIBhFa7YnOdgdwonsyzJObWQk0tFrB33k"
}
```

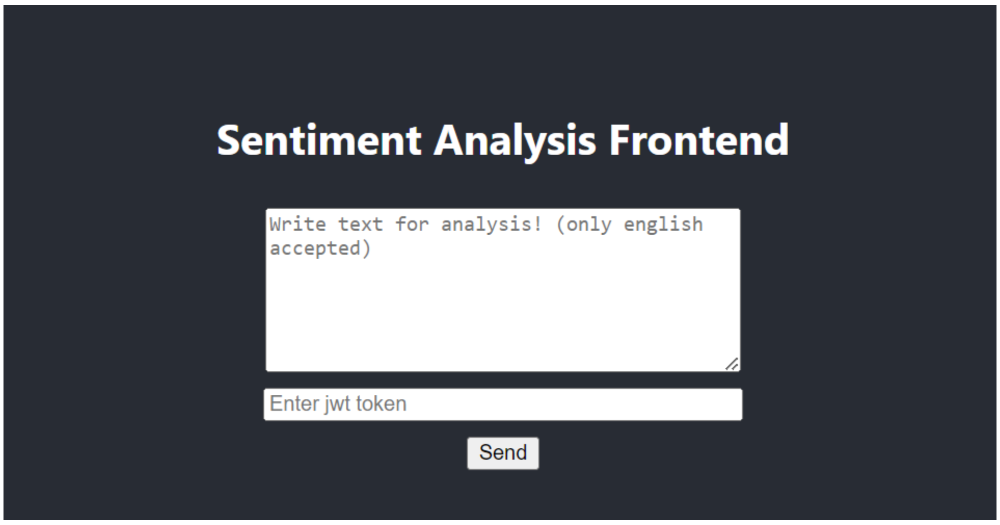
Then I created a function that verifies the jwt-token and the api for the sentiment-analysis.



```
// verify the jwt-token
const verifyToken = (req, res, next) => {

    const token = req.headers.authorization?.split(' ')[1];

    if (!token) return res.status(401).json({ message: 'no token provided' });

    try {
        const decoded = jwt.verify(token, JWT_TOKEN);
        req.user = decoded;
        next();
    } catch (ex) {
        res.status(400).json({ message: 'Invalid token.' });
    }
};

// protected sentiment-analysis api
app.post('/sentiment-analysis', verifyToken, async (req, res) => {
    console.log(req.body)
    const text = req.body
    const jsonText = JSON.stringify(text)
    const options = {
        method: 'POST',
        url: 'https://sentiment-analysis9.p.rapidapi.com/sentiment',
        headers: {
            'Content-Type': 'application/json',
            Accept: 'application/json',
            'X-RapidAPI-Key': process.env.RAPIDAPI_KEY,
            'X-RapidAPI-Host': 'sentiment-analysis9.p.rapidapi.com'
        },
        data: [{ id: '1', language: 'en', text: jsonText}]
    };

    try {
        const response = await axios.request(options);
        res.json(response.data)
        console.log(response.data)
    } catch (error) {
        res.status(500).json({ message: 'Failed to fetch sentiment analysis', error: error.message
    }
});
```

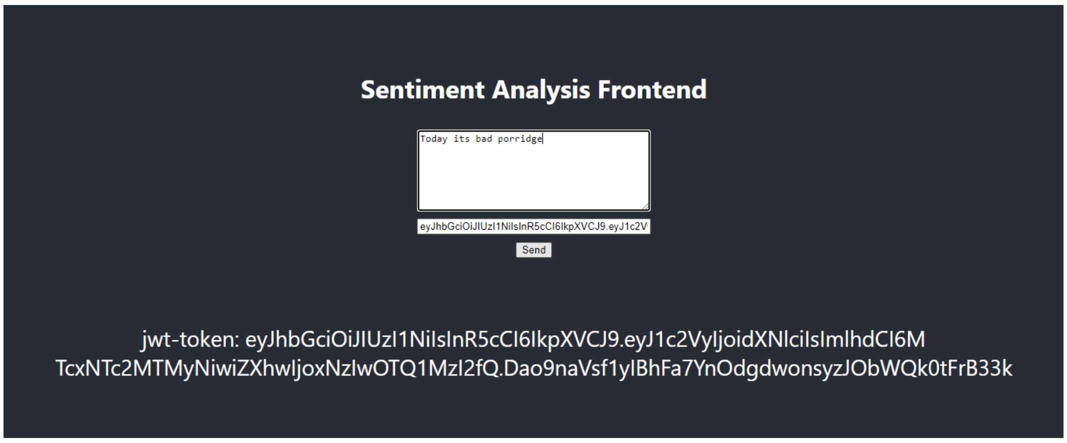
Then I did the needed changes to the frontend to hit the correct endpoint and added input for the jwt-token

Dev testing:



Sentiment Analysis Frontend

(jwt-token visible for testing purposes)

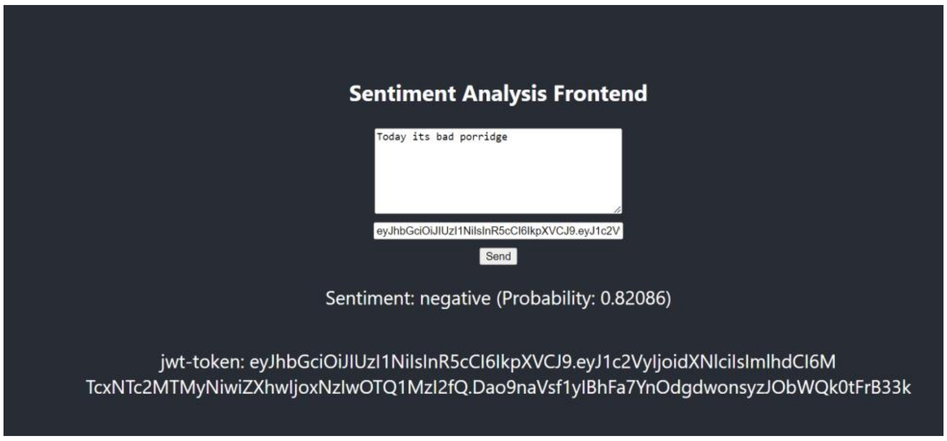


Sentiment Analysis Frontend

```
Today its bad porridge
```

jwt-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoidXNlciIsImlhdCI6M
TcxNTc2MTMyNiwiZXhwIjoxNzIwOTQ1MzI2fQ.Dao9naVsf1yIBhFa7YnOdgdwonsyzJObWQk0tFrB33k

=>



Sentiment Analysis Frontend

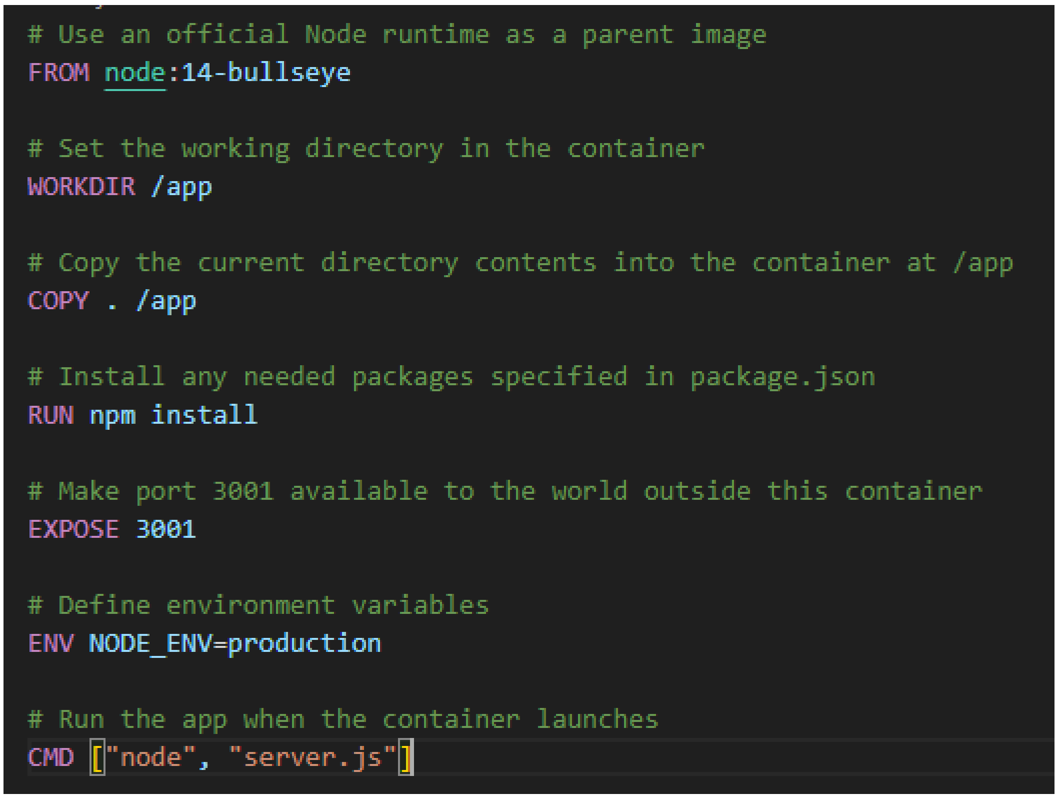
```
Today its bad porridge
```

Sentiment: negative (Probability: 0.82086)

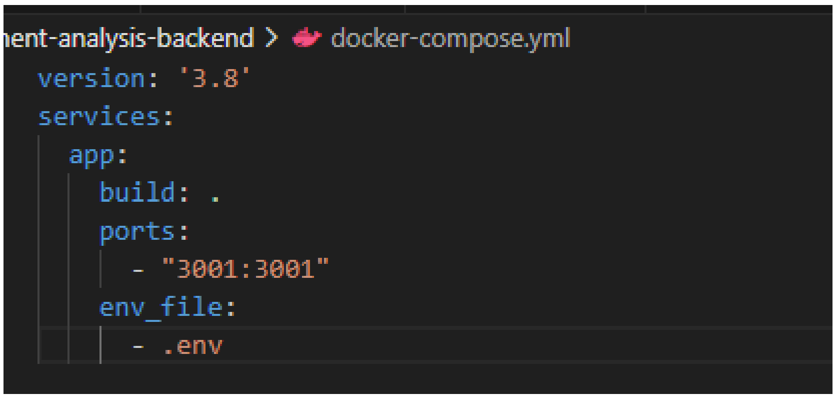
jwt-token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoidXNlciIsImlhdCI6M
TcxNTc2MTMyNiwiZXhwIjoxNzIwOTQ1MzI2fQ.Dao9naVsf1yIBhFa7YnOdgdwonsyzJObWQk0tFrB33k

Everything worked! Now lets try docker on local environment.

I created needed docker + docker-compose files:

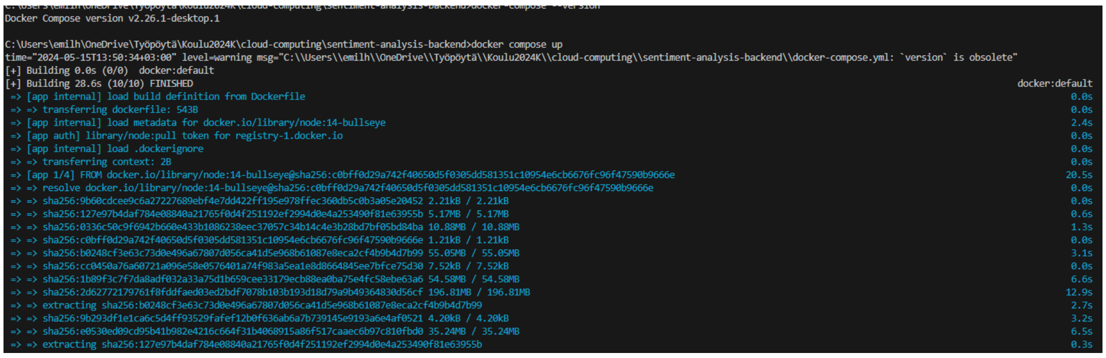


```dockerfile
# Use an official Node runtime as a parent image
FROM node:14-bullseye

# Set the working directory in the container
WORKDIR /app

# Copy the current directory contents into the container at /app
COPY . /app

# Install any needed packages specified in package.json
RUN npm install

# Make port 3001 available to the world outside this container
EXPOSE 3001

# Define environment variables
ENV NODE_ENV=production

# Run the app when the container launches
CMD ["node", "server.js"]
```

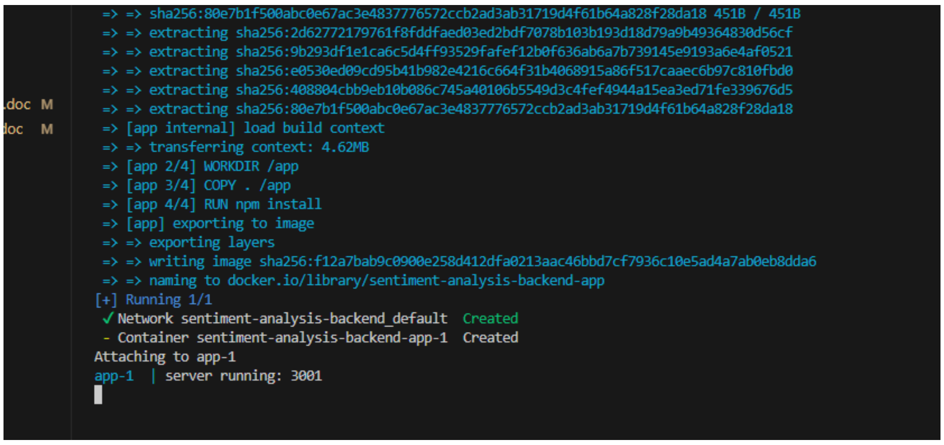


ent-analysis-backend > docker-compose.yml

```yaml
version: '3.8'
services:
  app:
    build: .
    ports:
      - "3001:3001"
    env_file:
      - .env
```



```
C:\Users\emilh\OneDrive\Työpöytä\Koulu2024K\cloud-computing\sentiment-analysis-backend>docker-compose --version
Docker Compose version v2.26.1-desktop.1

C:\Users\emilh\OneDrive\Työpöytä\Koulu2024K\cloud-computing\sentiment-analysis-backend>docker compose up
time="2024-05-15T13:50:34+03:00" level=warning msg="C:\\Users\\emilh\\OneDrive\\Työpöytä\\Koulu2024K\\cloud-computing\\sentiment-analysis-backend\\docker-compose.yml: `version` is obsolete"
[+] Building 0.0s (0/0)  docker:default
[+] Building 28.6s (10/10) FINISHED                                                                          docker:default
 => [app internal] load build definition from Dockerfile                                                               0.0s
 => => transferring dockerfile: 543B                                                                                   0.0s
 => [app internal] load metadata for docker.io/library/node:14-bullseye                                                2.4s
 => [app auth] library/node:pull token for registry-1.docker.io                                                        0.0s
 => [app internal] load .dockerignore                                                                                  0.0s
 => => transferring context: 2B                                                                                        0.0s
 => [app 1/4] FROM docker.io/library/node:14-bullseye@sha256:c0bff0d29a742f40650d5f0305dd581351c10954e6cb6676fc96f47590b9666e   20.5s
 => => resolve docker.io/library/node:14-bullseye@sha256:c0bff0d29a742f40650d5f0305dd581351c10954e6cb6676fc96f47590b9666e       0.0s
 => => sha256:9b60cdcee9c6a27227689ebf4e7dd422ff195e978ffec360db5c0b3a05e20452 2.21kB / 2.21kB                         0.0s
 => => sha256:127e97b4daf784e08840a21765f0d4f251192ef2994d0e4a253490f81e63955b 5.17MB / 5.17MB                         0.6s
 => => sha256:0336c50c9f6942b660e433b1086238eec37057c34b14c4e3b28bd7bf05bd84ba 10.88MB / 10.88MB                       1.3s
 => => sha256:c0bff0d29a742f40650d5f0305dd581351c10954e6cb6676fc96f47590b9666e 1.21kB / 1.21kB                         0.0s
 => => sha256:b0248cf3e63c73d0e496a67807d056ca41d5e968b61087e8eca2cf4b9b4d7b99 55.05MB / 55.05MB                       3.1s
 => => sha256:cc0450a76a60721a096e58e0576401a74f983a5ea1e8d8664845ee7bfce75d30 7.52kB / 7.52kB                         0.0s
 => => sha256:1b89f3c7f7da8adf032a33a75d1b659cee33179ecb88ea0ba75e4fc58ebe63a6 54.58MB / 54.58MB                       6.6s
 => => sha256:2d62772179761f8fddfaed03ed2bdf7078b103b193d18d79a9b49364830d56cf 196.81MB / 196.81MB                    12.9s
 => => extracting sha256:b0248cf3e63c73d0e496a67807d056ca41d5e968b61087e8eca2cf4b9b4d7b99                              2.7s
 => => sha256:9b293df1e1ca6c5d4ff93529fafef12b0f636ab6a7b739145e9193a6e4af0521 4.20kB / 4.20kB                         3.2s
 => => sha256:e0530ed09cd95b41b982e4216c664f31b4068915a86f517caaec6b97c810fbd0 35.24MB / 35.24MB                       6.5s
 => => extracting sha256:127e97b4daf784e08840a21765f0d4f251192ef2994d0e4a253490f81e63955b                              0.3s
```

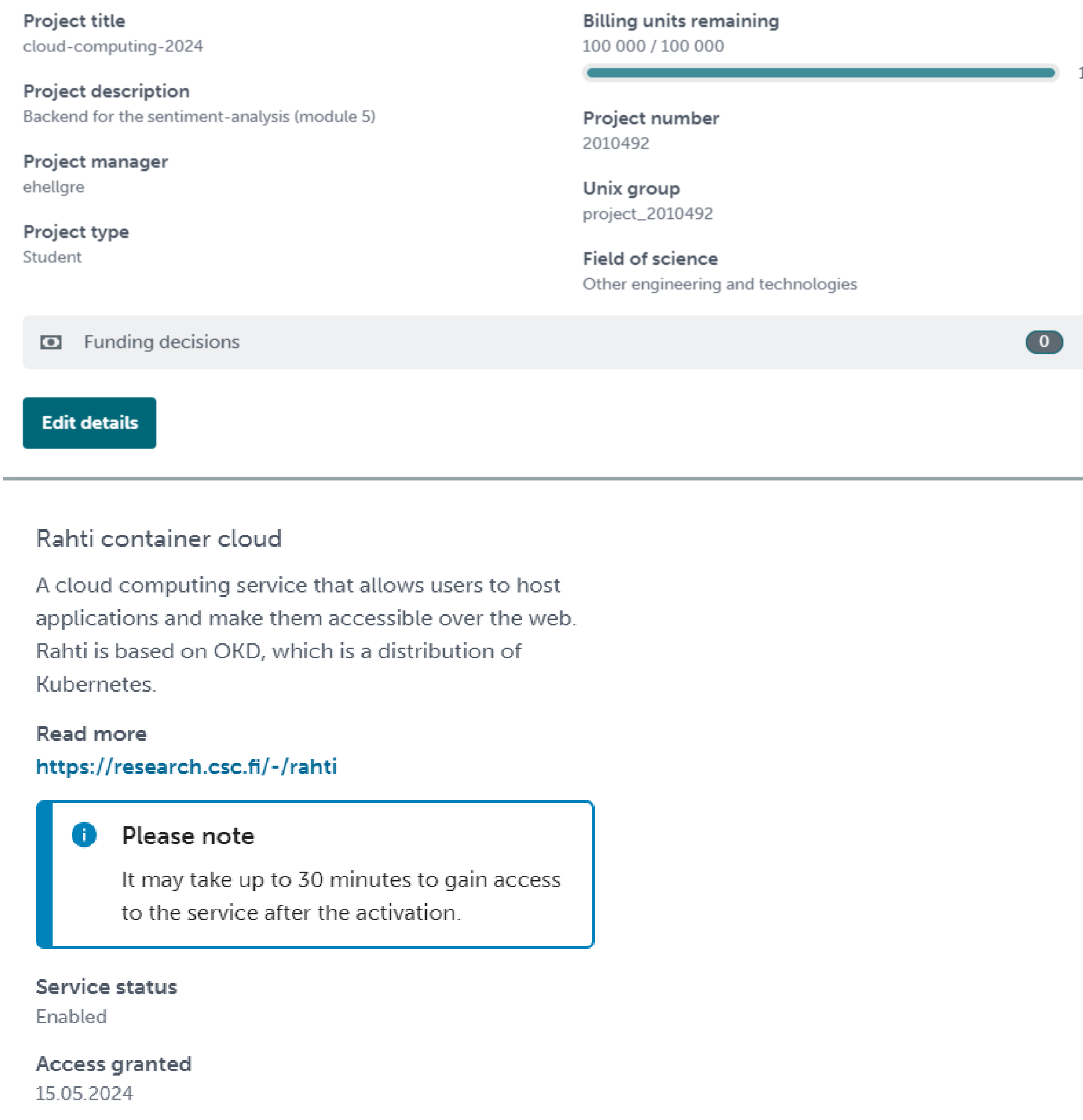


```
     => => sha256:80e7b1f500abc0e67ac3e4837776572ccb2ad3ab31719d4f61b64a828f28da18 451B / 451B
     => => extracting sha256:2d62772179761f8fddfaed03ed2bdf7078b103b193d18d79a9b49364830d56cf
     => => extracting sha256:9b293df1e1ca6c5d4ff93529fafef12b0f636ab6a7b739145e9193a6e4af0521
     => => extracting sha256:e0530ed09cd95b41b982e4216c664f31b4068915a86f517caaec6b97c810fbd0
     => => extracting sha256:408804cbb9eb10b086c745a40106b5549d3c4fef4944a15ea3ed71fe339676d5
.doc  M  => => extracting sha256:80e7b1f500abc0e67ac3e4837776572ccb2ad3ab31719d4f61b64a828f28da18
doc   M  => [app internal] load build context
     => => transferring context: 4.62MB
     => [app 2/4] WORKDIR /app
     => [app 3/4] COPY . /app
     => [app 4/4] RUN npm install
     => [app] exporting to image
     => => exporting layers
     => => writing image sha256:f12a7bab9c0900e258d412dfa0213aac46bbd7cf7936c10e5ad4a7ab0eb8dda6
     => => naming to docker.io/library/sentiment-analysis-backend-app
    [+] Running 1/1
     ✓ Network sentiment-analysis-backend_default  Created
     - Container sentiment-analysis-backend-app-1  Created
    Attaching to app-1
    app-1  | server running: 3001
```

Everything works on that end too.

Now to the CSC:

I created the CSC proj and added the Rahti service:



Project title
cloud-computing-2024

Project description
Backend for the sentiment-analysis (module 5)

Project manager
ehellgre

Project type
Student

Billing units remaining
100 000 / 100 000

Project number
2010492

Unix group
project_2010492

Field of science
Other engineering and technologies

Rahti container cloud

A cloud computing service that allows users to host applications and make them accessible over the web. Rahti is based on OKD, which is a distribution of Kubernetes.

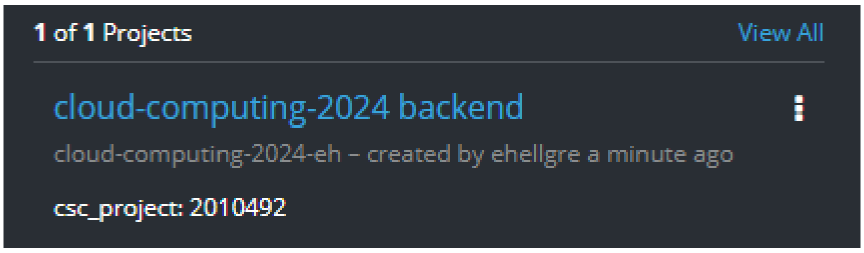
Read more
https://research.csc.fi/-/rahti

Please note
It may take up to 30 minutes to gain access to the service after the activation.

Service status
Enabled

Access granted
15.05.2024

Also Rahti CSC:


1 of 1 Projects View All

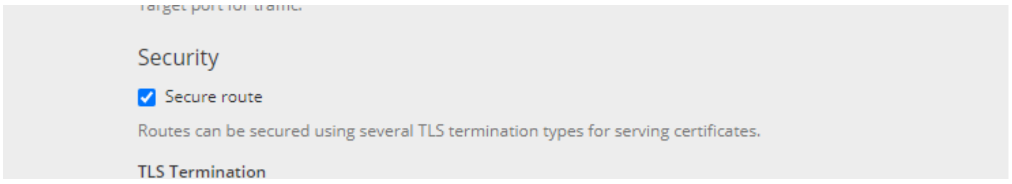
cloud-computing-2024 backend

cloud-computing-2024-eh – created by ehellgre a minute ago

csc_project: 2010492

Now that projects have been created, I will move on to the pipeline.

Making routes https:

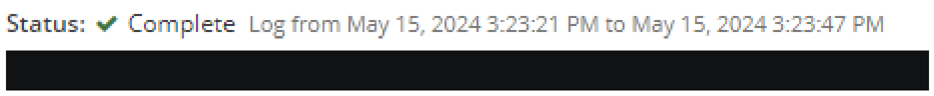

Security

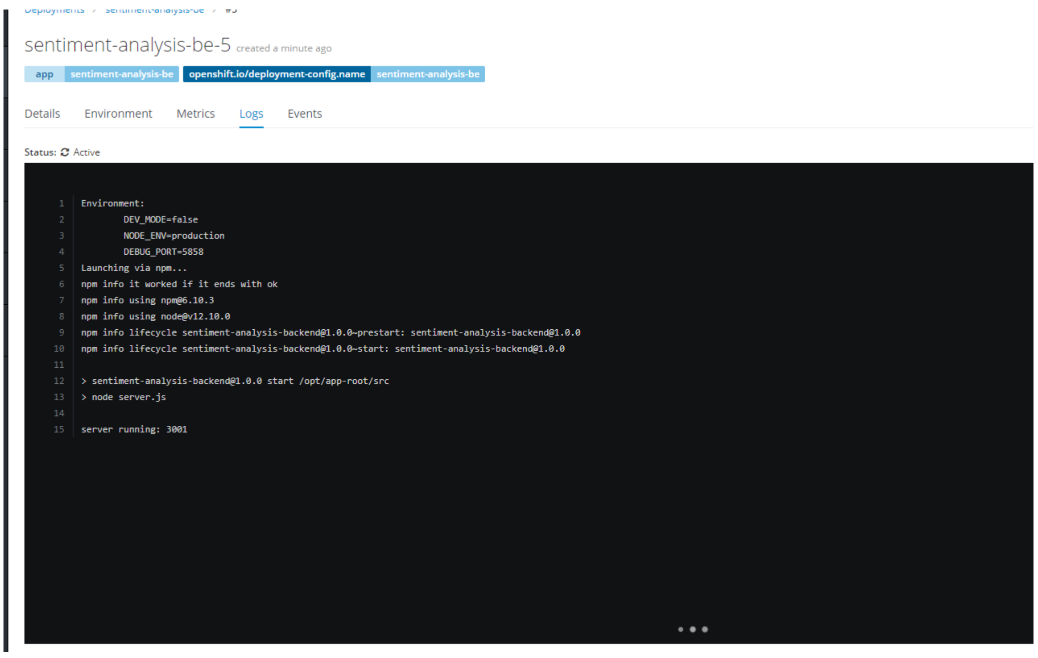
☑ Secure route

Routes can be secured using several TLS termination types for serving certificates.

TLS Termination

Build successful:


Status: ✓ Complete Log from May 15, 2024 3:23:21 PM to May 15, 2024 3:23:47 PM

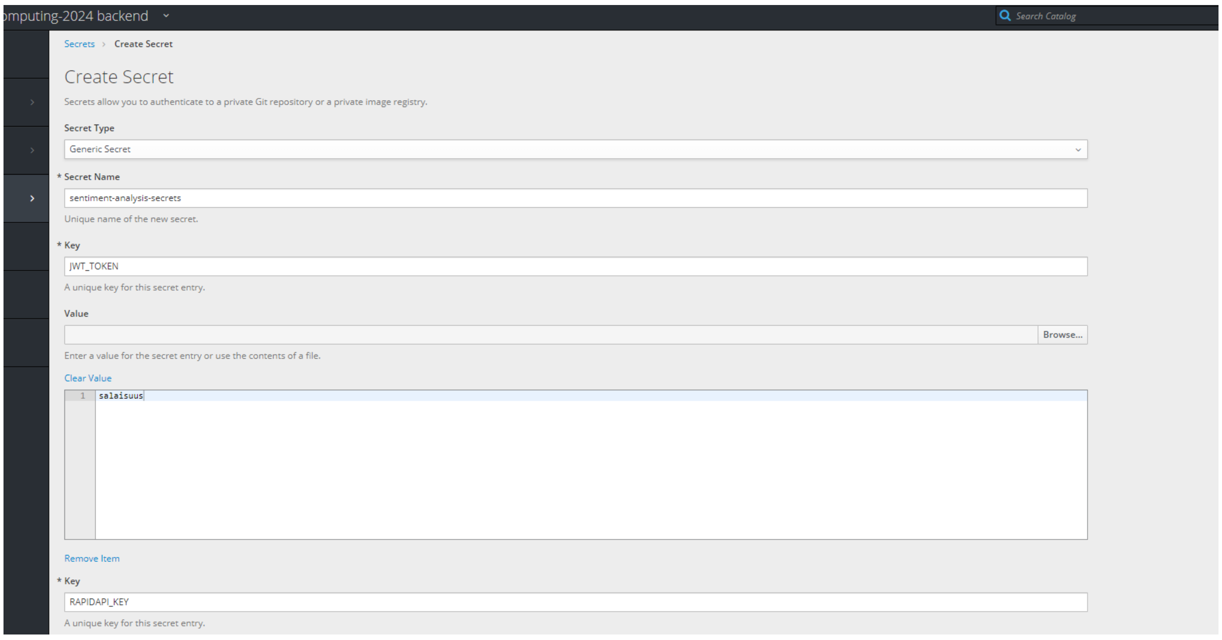

sentiment-analysis-be-5 created a minute ago

app: sentiment-analysis-be | openshift.io/deployment-config.name: sentiment-analysis-be

Status: Active

```
Environment:
        DEV_MODE=false
        NODE_ENV=production
        DEBUG_PORT=5858
Launching via npm...
npm info it worked if it ends with ok
npm info using npm@6.10.3
npm info using node@v12.10.0
npm info lifecycle sentiment-analysis-backend@1.0.0~prestart: sentiment-analysis-backend@1.0.0
npm info lifecycle sentiment-analysis-backend@1.0.0~start: sentiment-analysis-backend@1.0.0

> sentiment-analysis-backend@1.0.0 start /opt/app-root/src
> node server.js

server running: 3001
```

I also need to set the secrets, so I created new Generic Secret and placed them there:



Create Secret

Secrets allow you to authenticate to a private Git repository or a private image registry.

Secret Type: Generic Secret

* Secret Name: sentiment-analysis-secrets

Unique name of the new secret.

* Key: JWT_TOKEN

A unique key for this secret entry.

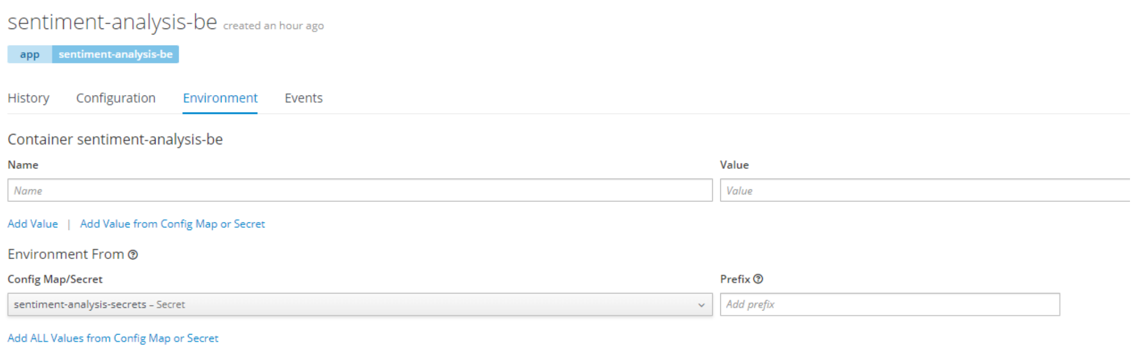
Value

Enter a value for the secret entry or use the contents of a file.

```
salaisuus
```

* Key: RAPIDAPI_KEY

A unique key for this secret entry.


sentiment-analysis-be created an hour ago

app sentiment-analysis-be
History
Configuration
Environment
Events

Container sentiment-analysis-be

Name	Value
Name	Value

Add Value | Add Value from Config Map or Secret

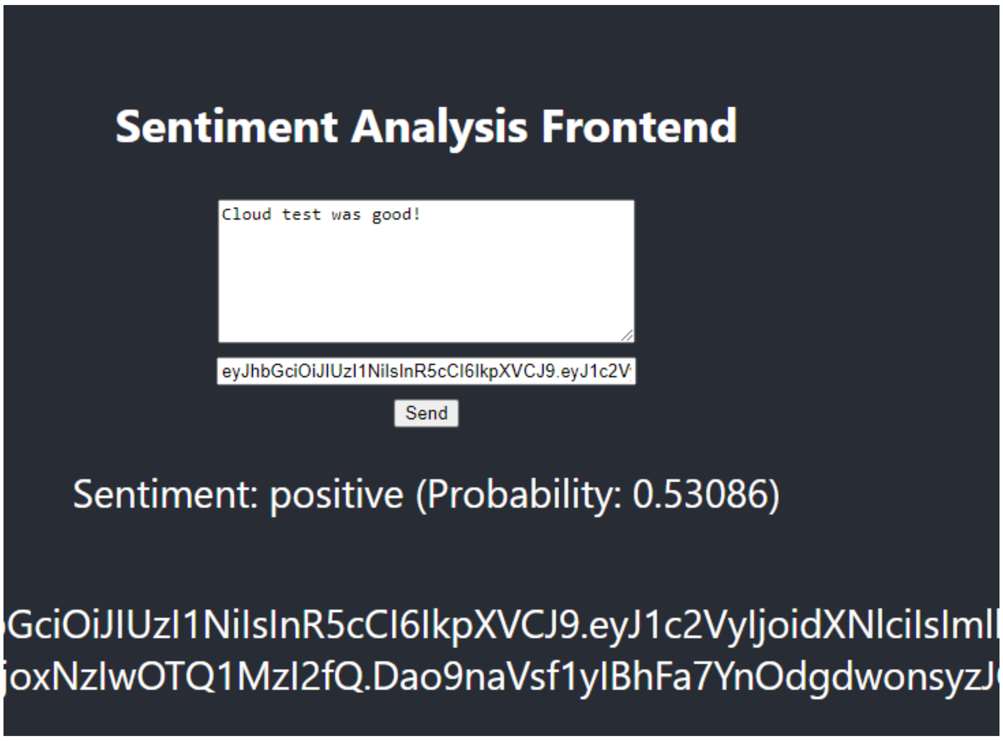
Environment From

Config Map/Secret	Prefix
sentiment-analysis-secrets – Secret	Add prefix

Add ALL Values from Config Map or Secret

⇨ Then I made the last change to the frontend to hit the new endpoint url (https://sentiment-analysis-be-cloud-computing-2024-eh.rahtiapp.fi/sentiment-analysis).

And tested on local:


Sentiment Analysis Frontend

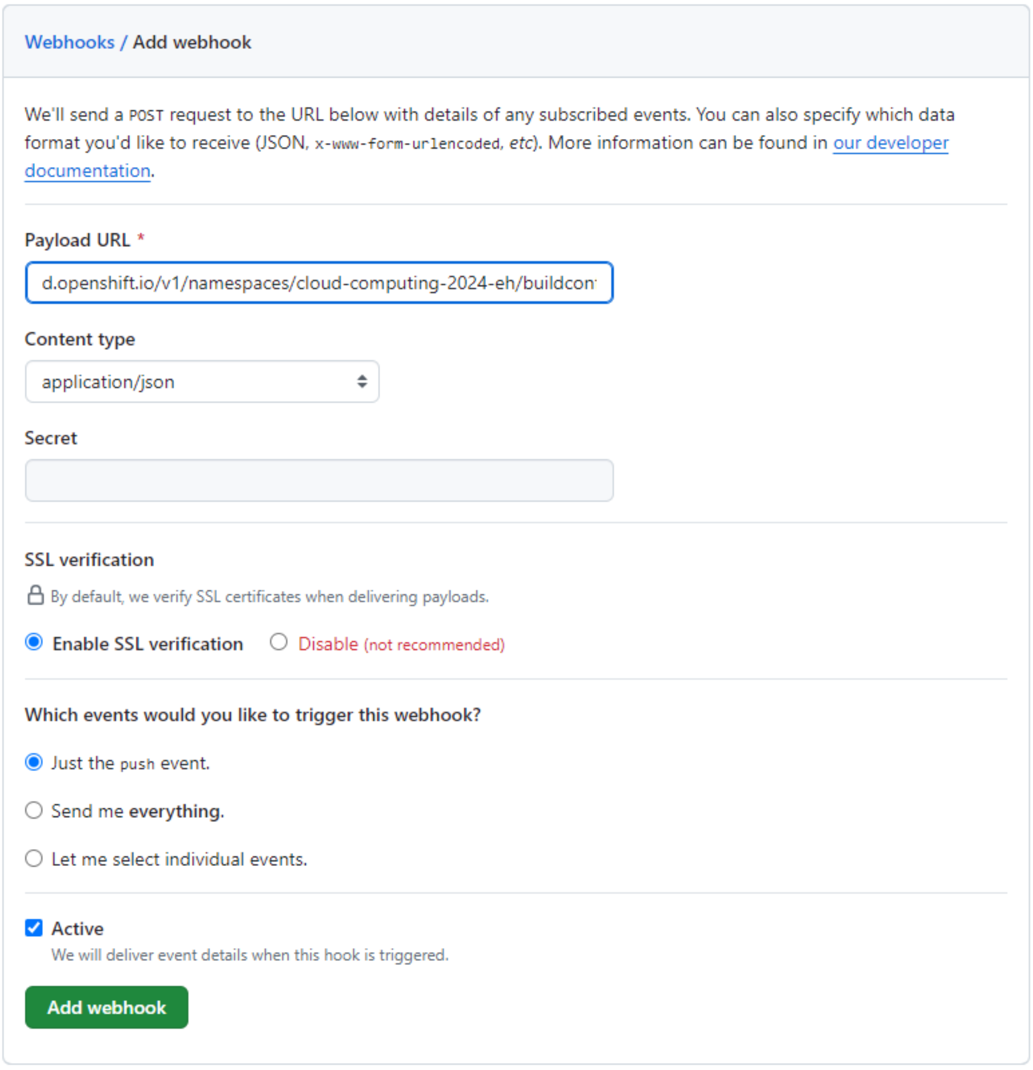
```
Cloud test was good!
```

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2V

Sentiment: positive (Probability: 0.53086)

GciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VyIjoidXNlciIsIml
joxNzIwOTQ1MzI2fQ.Dao9naVsf1yIBhFa7YnOdgdwonsyzJ

Lastly, I added the webhook to Github:



Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, *etc*). More information can be found in our developer documentation.

Payload URL *

d.openshift.io/v1/namespaces/cloud-computing-2024-eh/buildcon

Content type

application/json

Secret

SSL verification

By default, we verify SSL certificates when delivering payloads.

[✓ Enable SSL verification] Disable (not recommended)

Which events would you like to trigger this webhook?

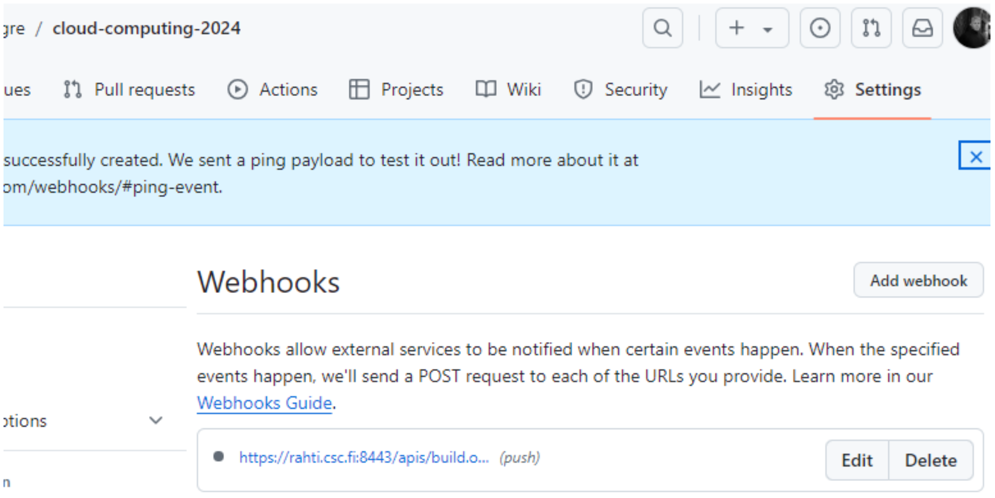
[✓ Just the push event.]

Send me **everything**.

Let me select individual events.

[✓ Active]

We will deliver event details when this hook is triggered.



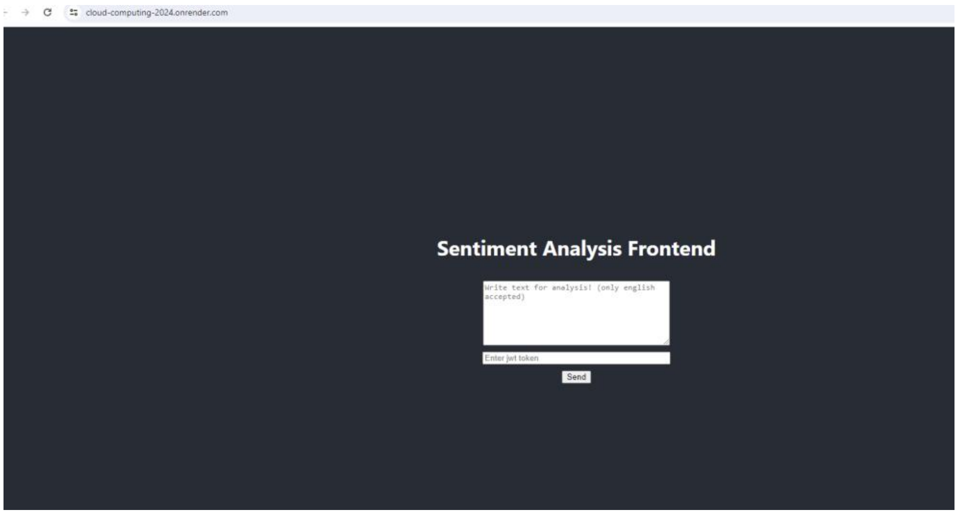
successfully created. We sent a ping payload to test it out! Read more about it at
om/webhooks/#ping-event.

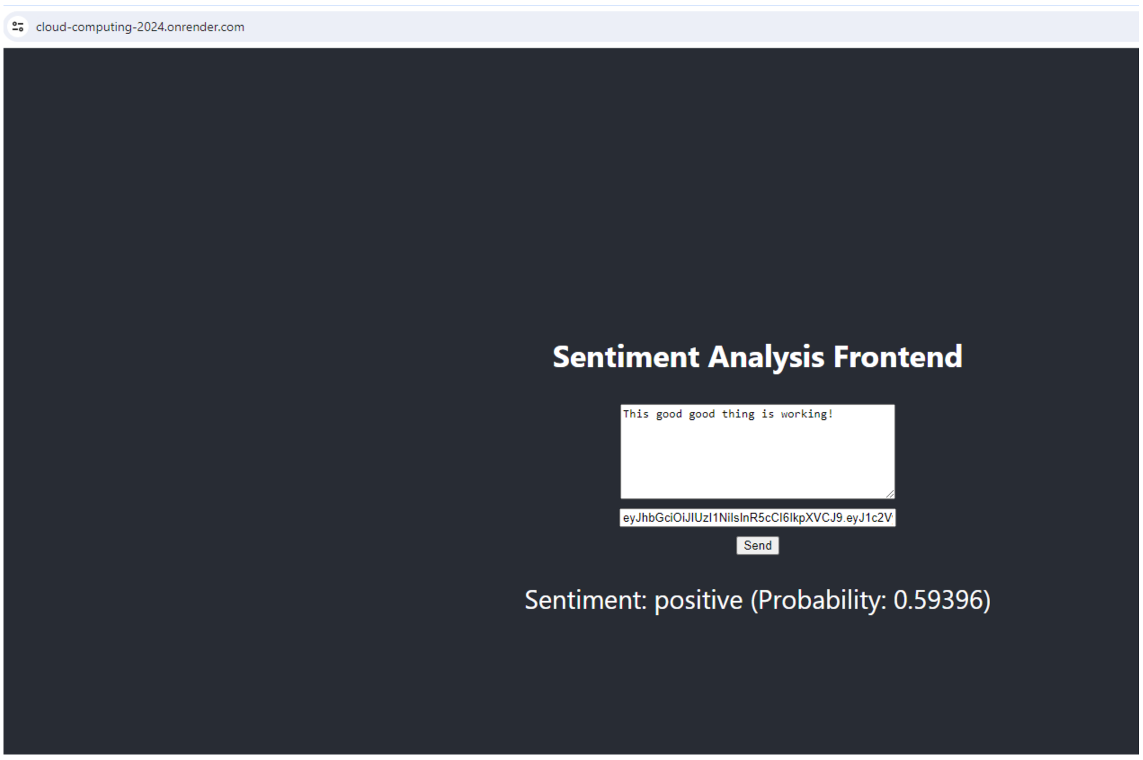
Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

● https://rahti.csc.fi:8443/apis/build.o... *(push)*

Let's test on prod:


cloud-computing-2024.onrender.com
Sentiment Analysis Frontend
Write text for analysis! (only english accepted)
Enter jwt token
Send


cloud-computing-2024.onrender.com
Sentiment Analysis Frontend
This good good thing is working!
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2V
Send

Sentiment: positive (Probability: 0.59396)

Everything works and looks good!

Emil Hellgren
Emil Hellgren