



Milestone 2

Verslag

Linux Webservices ITFactory

Nalani Van Beemen 2CCS01

Academiejaar 2023-2024

Campus Geel, Kleinhoefstraat 4, BE-2440 Geel

INHOUDSTAFEL

Inhoudsopgave

INHOUDSTAFEL	3
1 INLEIDING	4
2 INSTALLATIE VAN NODIGE PACKAGES.....	5
2.1 Installeren van kubectl, Kubernetes	5
2.2 Installeren van Kind.....	5
2.3 Kindconfig.yml	6
3 INGRESS	7
3.1 Ingress.yml.....	8
3.2 Ingress.yml.....	9
4 LIGHTTPD	10
4.1 Lighttpd-deployment.yml	10
4.2 Lighttpd-service.yml	11
4.3 Lighttpd-configmap.yml	12
5 MONGODB	13
5.1 Mongodb-deployment.yml.....	13
5.2 Mongo-service.yml	14
6 PERSISTENT VOLUME.....	15
6.1 Mongodb-pv-volume.yml.....	15
6.2 Mongodb-pv-claim.yml.....	15
7 NODE.JS	16
7.1 Dockerfile	16
7.2 App.js.....	17
8 DATABASE.....	20
9 EINDRESULTAAT	21
10 SLOT	22

1 INLEIDING

Voor het vak linux webservices heb ik een webstack opgezet in Kubernetes. Hiervoor kregen we een paar specificaties, mijn zijn:

- Webserver: lighttpd
- API: nodejs
- Database: MongoDB

Hieronder vindt u een overzicht van mijn directory structuur. Onder node_modules staan enkele packages geïnstalleerd om via nodejs een waarde uit de database te krijgen.

```
vagrant@ubuntu2204:~$ tree
.
├── api
├── app.js
├── deployment.yml
├── Dockerfile
├── docker.sh
├── get_helm.sh
├── ingress.yml
├── kindconfig
├── kindconfig2
├── kubernetes-archive-keyring.gpg
├── lighttpd
│   ├── ingress.yml
│   ├── lighttpd-configmap.yml
│   ├── lighttpd-deployment.yml
│   ├── lighttpd-service.yml
│   └── pvc.yml
├── lighttpd-service.yml
├── MongoDB
│   ├── mongodb-database-tools-ubuntu2004-x86_64-100.5.0.deb
│   ├── mongodb-database-tools-ubuntu2004-x86_64-100.5.0.deb.1
│   ├── mongodb-deployment.yml
│   ├── mongodb-org-tools_4.4.12_amd64.deb
│   ├── mongodb-pv-claim.yml
│   ├── mongodb-pv-volume.yml
│   ├── mongodb-values.yml
│   ├── mongo-secret.yml
│   ├── mongo-service.yml
│   └── mongo-statefulset.yml
├── node_modules
│   ├── accepts
│   │   ├── HISTORY.md
│   │   ├── index.js
│   │   ├── LICENSE
│   │   ├── package.json
│   │   └── README.md
│   └── array-flatten
└── array-flatten
```

2 INSTALLATIE VAN NODIGE PACKAGES

2.1 Installeren van kubectl, Kubernetes

Het installeren van Kubernetes doen we aan de hand van volgende commando's.

```
sudo apt-get update
sudo apt-get install -y ca-certificates curl
curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-archive-keyring.gpg
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-archive-keyring.gpg]
https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee
/etc/apt/sources.list.d/kubernetes.list
sudo apt-get update
sudo apt-get install -y kubectl
```

2.2 Installeren van Kind

Het installeren van Kind doen we aan de hand van volgende commando's.

```
curl -Lo ./kind https://kind.sigs.k8s.io/dl/v0.20.0/kind-linux-amd64
chmod +x ./kind
sudo mv ./kind /usr/local/bin/kind
```

2.3 Kindconfig.yml

Na het installeren van Kubernetes en Kind kunnen we onze kindconfig aanmaken. Met onze kindconfig.yml file kunnen we dan een cluster maken. De 2 extra workers zijn hier geconfigureerd om eventueel aan loadbalancing te doen verder in dit project.

Belangrijk in deze file is het zinnetje "ingress-ready=true", dit zorgt ervoor dat onze cluster van buitenaf bereikbaar is.

```

GNU nano 6.2 kindconfig2
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  kubeadmConfigPatches:
  - |
    kind: InitConfiguration
    nodeRegistration:
      kubeletExtraArgs:
        node-labels: "ingress-ready=true"
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
- role: worker
- role: worker
  
```

Daarna kunnen we een cluster maken met het volgend commando:

- kind create cluster --config=kindconfig

Met het commando `kubectl get nodes` zien we dat de control-plane en workers opgestart zijn. Hier kunnen we zien dat er twee kind workers zijn aangemaakt.

```

vagrant@ubuntu2204:~$ kubectl get nodes
NAME                 STATUS    ROLES                  AGE     VERSION
kind-control-plane   Ready    control-plane         28h     v1.25.2
kind-worker          Ready    <none>                 28h     v1.25.2
kind-worker2         Ready    <none>                 28h     v1.25.2
  
```

3 INGRESS

Ingress definieert regels voor het beheren van externe toegang tot services binnen de cluster. Met ingress is het mogelijk om HTTP- en HTTPS-routers naar services in het cluster te definiëren.

We gebruiken NGINX ingress, dit installeren we met volgend commando:

```
- kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingressnginx/master/deploy/static/provider/kind/deploy.yaml
```

Wanneer we dit commando hebben uitgevoerd kunnen we nagaan of de controller actief is. Dit doen we aan de hand van het commando: `kubectl get pods`, hierbij specificeren we dan nog `-n ingress-nginx`.

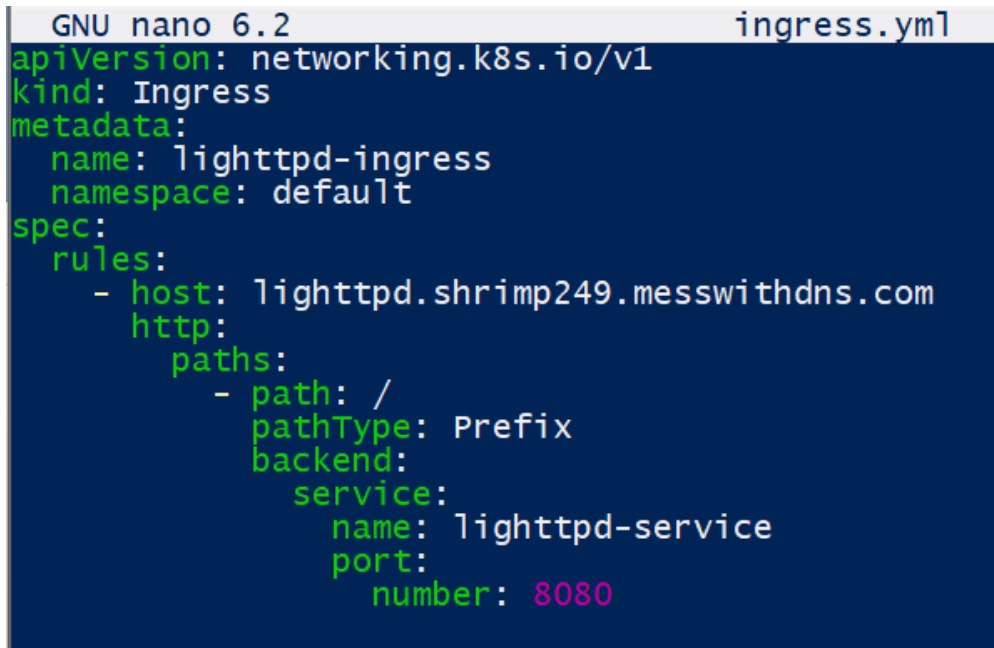
```
vagrant@ubuntu2204:~$ kubectl get pods -n ingress-nginx
```

NAME	READY	STATUS	RESTARTS	AGE
ingress-nginx-admission-create-bqf8r	0/1	Completed	0	28h
ingress-nginx-admission-patch-2nr5m	0/1	Completed	0	28h
ingress-nginx-controller-9c496784c-6vs29	1/1	Running	0	28h

Ik heb 2 ingress files geconfigureerd. Één voor mijn `lighttpd`-service. En één voor mijn `node.js`.

3.1 Ingress.yml

Deze ingress file werkt voor mijn lighttpd-service.



```
GNU nano 6.2 ingress.yml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: lighttpd-ingress
  namespace: default
spec:
  rules:
    - host: lighttpd.shrimp249.messwithdns.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: lighttpd-service
                port:
                  number: 8080
```

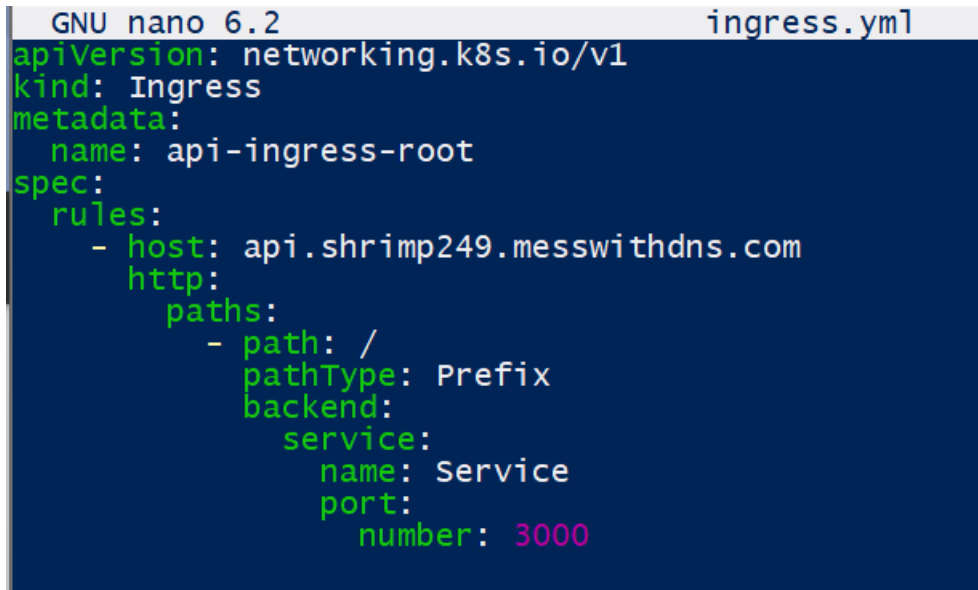
Bij host hebben we onze link van mess with dns gezet. Hierdoor is dit onze url. Deze url is 192.168.56.5 omgezet met 'mess with dns'.

Bij path specificeren we '/', hierdoor wordt de pagina van NGINX gepubliceerd door enkel het IP adres te zoeken.

Bij backend vinden we de service waar de ingress de ontvangen trafiek naar moet doorsturen. In mijn geval lighttpd webserver.

3.2 Ingress.yml

Deze ingress file werkt voor mijn api, node.js.

A screenshot of a terminal window showing the nano 6.2 text editor editing a file named ingress.yml. The file content is a Kubernetes Ingress resource definition. The text is as follows:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: api-ingress-root
spec:
  rules:
  - host: api.shrimp249.messwithdns.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: Service
            port:
              number: 3000
```

Dit is mijn tweede ingress file, dit is om te testen of mijn node.js werkt. Dit is niet volledig het geval.

Ook hier heb ik via 'mess with dns' een nieuwe url aangemaakt met hetzelfde IP adres. Hiervoor gebruiken we poort 3000.

4 LIGHTTPD

Ik kreeg de opdracht om als webserver lighttpd te gebruiken. Volgende zaken heb ik hiervoor geconfigureerd.

4.1 Lighttpd-deployment.yml

```
GNU nano 6.2 lighttpd-deployment.yml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: lighttpd-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: lighttpd
  template:
    metadata:
      labels:
        app: lighttpd
    spec:
      containers:
        - name: lighttpd
          image: jitesoft/lighttpd
          ports:
            - containerPort: 80
          volumeMounts:
            - name: lighttpd-index
              mountPath: /var/www/html
      volumes:
        - name: lighttpd-index
          configMap:
            name: lighttpd-configmap
```

Dit is mijn deployment voor mijn lighttpd. Als image pull ik jitesoft/lighttpd van dockerhub. Dit ga ik draaien op poort 80.

Onder VolumeMounts staat het mountPath gespecificeerd. Volumes wordt gebruikt om gegevens op te slaan en te delen tussen containers binnen/of tussen pods. Daarom wordt lighttpd-index gemount op het pad /var/www/html.

4.2 Lighttpd-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: lighttpd-service
spec:
  selector:
    app: lighttpd
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 80
  type: ClusterIP
```

We gebruiken het TCP protocol en dit draaien we op poort 8080. ClusterIP maakt de service alleen toegankelijk binnen het Kubernetes-cluster via een intern IP-adres. Dit is niet toegankelijk buiten de cluster.

4.3 Lighttpd-configmap.yml

```
GNU nano 6.2 lighttpd-configmap.yml
apiVersion: v1
kind: ConfigMap
metadata:
  name: lighttpd-configmap
  namespace: default
data:
  index.html: |

    <!DOCTYPE html>
    <html lang="en">
    <head>
    </head>
    <body>
      <h1><span id="user">Loading ... </span> has reached milestone 2!</h1>
      <script src="app.js"></script>
    </body>
    </html>
```

De configmap geeft onze index.html weer. Hierin maken we onze html code en voegen we het script toe om via node.js de naam uit de database te krijgen. Dit werkt niet.

5 MONGODB

5.1 Mongoddb-deployment.yml

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo
          imagePullPolicy: Always
          ports:
            - containerPort: 27017
          volumeMounts:
            - name: mongodb-volume
              mountPath: /data/db
          env:
            - name: MONGODB_ALLOW_EMPTY_ROOT_PASSWORD
              value: "0"
            - name: MONGODB_ROOT_PASSWORD
              value: secret
      volumes:
        - name: mongodb-volume
          persistentVolumeClaim:
            claimName: mongodb-pv-claim

```

Onder spec ziet u dat de mongodb container van docker gebruikt wordt. Op de poort 27017.

Onder volumeMounts wordt het persistentVolume ingesteld, dit zorgt ervoor dat de database de gegevens opslaat. Zodat na het verwijderen van de pod de data nog steeds beschikbaar is.

Met de value: secret wordt een secret gedefinieerd, dit heeft volgende inhoud:

```

apiVersion: v1
kind: Secret
metadata:
  name: mongo-secret
type: Opaque
data:
  username: YWRtaW4= # Base64-encoded username 'admin'
  password: YWRtaW4xMjM= # Base64-encoded password 'admin123'

```

5.2 Mongo-service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb
spec:
  selector:
    app: mongodb
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
  clusterIP: None
```

Poort 27017 wordt exposed voor de machine voor buitenaf.

6 PERSISTENT VOLUME

Voor het gebruiken van persistent volume gebruiken we twee files.

6.1 MongoDB-pv-volume.yml

Definieert een persistent volume object. Dit vertegenwoordigt een stuk opslag dat beschikbaar is in een cluster. Het opslagtype "manual" betekent dat het beheer van de opslag buiten Kubernetes ligt.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: mongodb-pv-volume
spec:
  storageClassName: manual
  capacity:
    storage: 20Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/mnt/data/mongodb"
```

6.2 MongoDB-pv-claim.yml

Definieert een persistentvolumeclaim object. Dit object wordt gebruikt door pods om dynamisch opslagruimte aan te vragen op basis van de beschikbare persistentVolumes

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mongodb-pv-claim
spec:
  storageClassName: manual
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
```

7 NODE.JS

7.1 Dockerfile

```
FROM node:14
WORKDIR /home/vagrant
COPY package.json package-lock.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD ["node", "app.js"]
```

Node:14 geeft aan dat dit de basisimage is voor de officiële node.js.

Op lijn twee vinden we de werkdirectory.

Ook voeren we npm install uit binnen de container.

Ook poort 3000 wordt blootgesteld.

CMD : dit stelt het standaardcommando in dat wordt uitgevoerd wanneer de container wordt gestart. Het geeft aan dat de node.js applicatie moet worden gestart door het uitvoeren van het commando node app.js.

7.2 App.js

```
const express = require('express');
const hostname = '0.0.0.0';
const port = 3000;

const app = express();

//Add a new API route
app.get('/hello', (req, res) => {
  res.send('Hello Node');
});

//Print container id
app.get('/id', (req, res) => {
  res.send(`Container ID: ${process.env.HOSTNAME}`);
});

//Start the server
app.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}`);
});
```

```
GNU nano 6.2      lightweight-configmap.yml
apiVersion: v1
kind: ConfigMap
metadata:
  name: lightweight-configmap
  namespace: default
data:
  index.html: |

  <!DOCTYPE html>
  <html lang="en">
  <head>
  </head>
  <body>
    <h1><span id="user">Loading ... </span> has reached milestone 2!</h1>
    <script src="app.js"></script>
  </body>
  </html>
```

Hieronder staat de javascript code om gegevens te kunnen verkrijgen. Wanneer we onze gegevens eruit kunnen halen kunnen we deze weergeven op ... pagina.

Dit is een voorbeeld vanop de cursussite maar dit werkt niet. Mijn eigen javascript code heb ik eruit gehaald omdat deze crashte. Daarom wou ik de code van de cursus site gebruiken maar ook deze werkt niet. Het probleem hiervan weet ik spijtig genoeg niet.

```
vagrant@ubuntu2204:~$ kubectl get pods
NAME                                READY   STATUS             RESTARTS   AGE
lighttpd-deployment-654c47687d-tqhq 1/1     Running            0           75m
mongodb-0                             1/1     Running            0           75m
mongodb-deployment-5798d4fcc9-758v6 1/1     Running            0           49m
my-nodejs-app-87bf658f6-2n2qv        0/1     CrashLoopBackOff   19 (2m31s ago) 75m
```

Dit beschrijft het probleem, maar ik vind hier geen oplossing voor.


```

vagrant@ubuntu2204:~$ kubectl describe pod my-nodejs-app-87bf658f6-2n2qv
Name:          my-nodejs-app-87bf658f6-2n2qv
Namespace:     default
Priority:       0
Service Account: default
Node:          kind-worker/172.18.0.2
Start Time:    Sun, 10 Dec 2023 18:01:30 +0000
Labels:        app=my-nodejs-app
               pod-template-hash=87bf658f6
Annotations:   <none>
Status:        Running
IP:            10.244.1.3
IPs:
  IP:          10.244.1.3
Controlled By: ReplicaSet/my-nodejs-app-87bf658f6
Containers:
  my-nodejs-app:
    Container ID:  containerd://b54e15db8b036282c5c94d246039ccc49036496f616ddd0a84f0fb0f13c0fd18
    Image:         vanbe/nalaniapi3:new
    Image ID:      docker.io/vanbe/nalaniapi3@sha256:3c593ca3d1255ac671dbffd99ed3126afc5b35b6c6ce0aa164f78271ce148835
    Port:          3000/TCP
    Host Port:     0/TCP
    State:         Waiting
      Reason:      CrashLoopBackOff
    Last State:    Terminated
      Reason:      Error
      Exit Code:    1
    Started:       Sun, 10 Dec 2023 19:14:41 +0000
    Finished:      Sun, 10 Dec 2023 19:14:42 +0000
    Ready:         False
    Restart Count: 19
    Environment:   <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-6wvx6(ro)
Conditions:
  Type           Status
  Initialized     True
  Ready           False
  ContainersReady False
  PodScheduled    True
Volumes:
  kube-api-access-6wvx6:
    Type:          Projected (a volume that contains injected data from
multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName:      kube-root-ca.crt
    ConfigMapOptional:  <nil>
    DownwardAPI:        true
QoS Class:         BestEffort
Node-Selectors:     <none>
Tolerations:        node.kubernetes.io/not-ready:NoExecute op=Exists for
300s

```

node.kubernetes.io/unreachable:NoExecute op=Exists for

300s

Events:

Type	Reason	Age	From	Message
Warning	BackOff	78s (x342 over 75m)	kubelet	Back-off restarting failed container

8 DATABASE

Voor de database heb ik MongoDB gebruikt. Dit is een eenvoudige database dat op basis van directories werkt.

We kunnen gemakkelijk verbinding maken met de database, met volgend commando:

- kubectl exec -it mongodb-0 -- /bin/bash
- mongosh --host mongodb-0 -u admin -p admin123

Met het tweede commando kunnen we inloggen met onze gegevens.

```
vagrant@ubuntu2204:~/MongoDB$ kubectl exec -it mongodb-0 -- /bin/bash
root@mongodb-0:/# mongosh --host mongodb-0 -u admin -p admin123
Current Mongosh Log ID: 65760b05821b2b9f46de4696
Connecting to: mongodb://<credentials>@mongodb-0:27017/?directConnection=true&appName=mongosh+2.1.0
Using MongoDB: 7.0.4
Using Mongosh: 2.1.0
mongosh 2.1.1 is available for download: https://www.mongodb.com/try/download/shell
For mongosh info see: https://docs.mongodb.com/mongodb-shell/

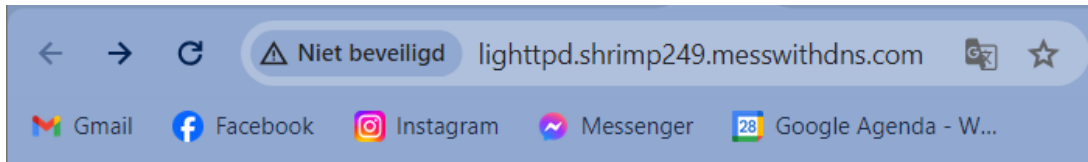
-----
The server generated these startup warnings when booting
2023-12-10T18:02:39.949+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage
see http://dochub.mongodb.org/core/prodnotes-filesystem
2023-12-10T18:02:40.756+00:00: vm.max_map_count is too low
-----

test> use Mydatabase
switched to db Mydatabase
Mydatabase> db.user.find()
[
  {
    _id: ObjectId('657603901c81c6cd26f6e080'),
    name: 'Nalani Van Beemen'
  }
]
Mydatabase>
```

Hierboven kan u de inhoud bekijken van mijn database. Hier staat enkel mijn naam in onder de collectie user in mydatabase en met tag 'name'.

9 EINDRESULTAAT

Als eindresultaat ben ik dit gekomen. Mijn pagina is goed bereikbaar maar de username krijg ik niet uit de database. Dit omdat deze code er ook niet in zit op dit moment omdat mijn app.js crasht.



Loading ... has reached milestone 2!

10 SLOT

Ik rond deze opdracht af met zowel een gevoel van voldoening en tekortkoming. Ik ben blij met wat ik bereikt heb want ik heb er heel lang en hard aan gewerkt. Toch vind ik het jammer dat ik het niet verder heb kunnen uitwerken. Het niveau voor deze opdracht lag iets te hoog voor mij.