SYSTEMDOKUMENTATION KUBERNETES-CLUSTER

Michel Gröbli

ETH Zürich

Versionstabelle

Datum	Name	Thema
03.06.2019	Michel Gröbli	Struktur
07.06.2019	Michel Gröbli	Informieren, Planen, Entscheiden
25.06.2019	Michel Gröbli	Realisieren, Kontrolle, Auswertung
28.06.2019	Michel Gröbli	Fertigstellen der Dokumentation

Inhalt

I Informieren

Aufgabe

Ich habe einen Monat (11 Tage) Zeit das Lehrlabor Projekt von Marc Vogelmann genauer unter die Lupe zu nehmen und zu verbessern/aktualisieren. Ich habe zu diesem Projekt aber noch zusätzliche Anforderungen bekommen, die ich dem Projekt von Marc Vogelmann hinzufügen soll. Dazu muss ich am Schluss eine Systemdokumentation und Benutzerdokumentation abgeben. Am 2. Juli werde ich auch eine zehn minütige Präsentation halten, bei der ich erzähle, was ich gemacht und wie ich gearbeitet habe.

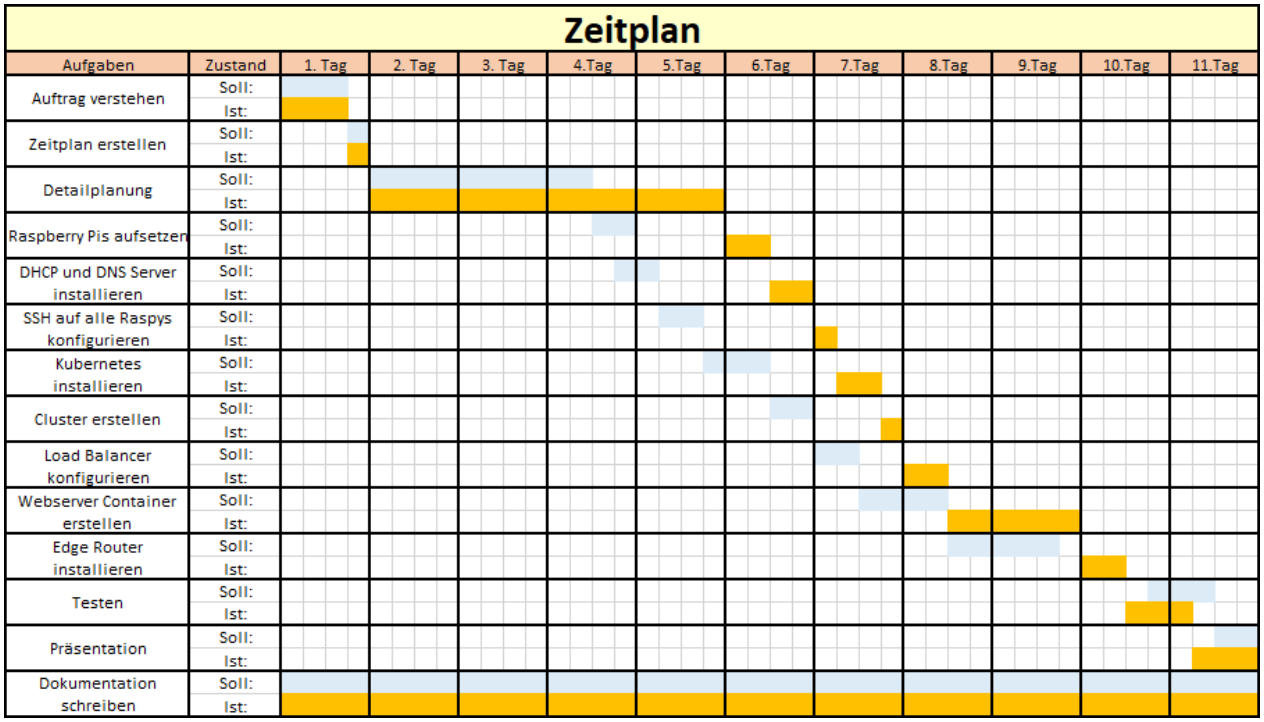
Dieses Projekt werde ich im HCP Gebäude im Raum 36.1 durchführen.
Ich bekam das Endprodukt von Marc Vogelmann und seine Dokumentation für das Projekt, womit ich mich auf das Projekt vorbereiten kann.

Momentan (grob)

Marc Vogelmann hat 13 Raspberry Pis für das Projekt eingesetzt. Einen Raspberry Pi hat er als Router, DHCP und DNS benutzt. Auf den 12 übrigen Raspberry Pis hat er Hypriot OS (Version 1.9.0) und auf dem „Router“ hat er Raspian (Version Juni 2018) installiert. Als Load Balancer hat er Traefik benutzt, der die Arbeitsprozesse auf die Raspberry Pis verteilt.

P Planen

Zeitplan



Meilensteine

Meilenstein	Was gemacht werden sollte	Datum
Erster Meilenstein	Planung fertiggestellt	07.06.2019
Zweiter Meilenstein	Raspberry Pi Umgebung fertiggestellt	12.06.2019
Dritter Meilenstein	Kubernetes Umgebung fertiggestellt	21.06.2019
Vierter Meilenstein	Dokumentation und Präsentation abgeben	02.07.2019

E Entscheiden

Fragen und Antworten

Was für einen Container Management benutze ich?

Ich werde Kubernetes als Container Management benutzen, da Kubernetes ein Orchestrations-Verfahren besitzt und Kubernetes mittlerweile bereits als Standard festgelegt wurde. Kubernetes fokussiert sich dabei stark auf die Skalierung, Verfügbarkeit und Sicherheit, die wir benötigen.

Welche Kubernetes Distribution benötige ich?

Als erstes habe ich mir Rancher näher angeschaut, da Rancher allen Anforderungen entspricht. Leider funktioniert Rancher auf ARM Prozessoren nicht. Da wir Raspberry Pis einsetzen, die einen ARM Prozessor haben, musste ich weitersuchen.

Nach weiterem Suchen traf ich auf Kubernetes Kops. Kops entsprach ebenfalls allen Anforderungen, leider ist Kops aber eine Cloudlösung, weshalb wir wieder weitersuchen müssen.

Ich habe mich entschieden Kubeadm zu benutzen, da Kubeadm keine Cloudlösung und ARM Prozessor kompatibel ist.

Was für ein Overlay-Network benutzen wir?

Ich entscheide mich für das Overlay-Netzwerk Flannel, da Flannel benutzerfreundlich ist und keine besondere Voraussetzungen benötigt. Zudem wird oft berichtet, dass Flannel sehr gut mit Kubernetes funktioniert. Eine Alternative wäre Jaguar, welches auch open-source ist und von open-day-light abstammt. Jaguar benötigt aber vxlan und ein CNIPlugin, weshalb wir Flannel vorziehen.

Welche Betriebssysteme installiere ich?

Für die Worker werde ich Raspian (Debian) verwenden, da ich Raspian am besten kenne. Für die Worker benutze ich Hypriot, da Hypriot Docker bereits vorinstalliert hat und man so weniger zusätzlich installieren muss.

Wie soll das Netzwerk aussehen?

Die Infrastruktur vom alten Kubernetes Netzwerk nicht gross verändert werden. Zudem sollten keine öffentlichen Adressen benutzt werden, da es eine Verschwendung von öffentliche Adressen wäre. Man kann auch private Adressen für die Raspberry Pis benutzen.

Welche IP-Adressen werde ich den Raspberry Pis geben?

Ich werde ihnen das private Netzwerk 192.168.0.1/24 Netz zuteilen, da sie dadurch geschützt werden und man von aussen nicht auf sie zugreifen kann.

Was benutzte ich als Load-Balancer?

Ich benutzte Traefik als Load-Balancer, da Traefik Open-Source ist und auch einen Reverse-Proxy beinhaltet. Zudem kann er auch Micro Services weiterleiten, was man mit dem Edge Router benutzen kann. Zudem ist Traefik sehr einfach zu bedienen, da Traefik die Routen automatisch erstellt und dich unterstützt, wenn man die routen manuell erstellen will.

Welche Dienste werden auf dem Cluster laufen?

Auf dem Cluster werde ich einen Webdienst in Betrieb nehmen.

Was sind unsere Sicherheitsmassnahmen?

Da wir einen Cluster haben, können mehrere Worker ausfallen und die Services werden dennoch nicht beeinträchtigt. Zudem werden wir einen 2er Cluster aus 2 Masters machen, was dazu führt, dass auch ein Master ausfallen kann und sich «nichts» verändert. Dadurch dass unsere Raspberry Pis genated sind, kann man von aussen nicht auf die Raspberry Pis zugreifen. Die Clients ausserhalb von unserem internen Netzwerk können durch den Reverse-Proxy nicht direkt mit unseren Services kommunizieren, was uns eine zusätzliche Sicherheit gewährt.

Womit werden die Cluster erstellt?

Traefik hat die Funktion, dass er zwischen den erstellten Nodes Load-Balancing durchführen kann, was das Ganze wesentlich einfacher macht, da wir da kein zusätzliches Programm benutzen müssen.

Was brauche ich damit man auf die Server zugreifen kann?

Durch Kubernetes bekommt jeder Container eine IP-Adresse eines bestimmten Netzes, welches von Kubernetes bestimmt wird. Ich werde Traefik benutzten, welche die Anfrage auf den Server zu der IP-Adresse von den Pods weiterleitet.

Container Management

Es gibt momentan mehrere Container Management Systeme. Ich habe Kubernetes ausgewählt, da Kubernetes mittlerweile als weltweiter Standard gilt und weitaus besser skaliert als andere Container Management Systeme. Zusätzlich hat Kubernetes einige Einstelllungen die Docker Swarm nicht hat wie zum Beispiel, dass Kubernetes Load-Balancing „ungesunde“ Pods ausfindig macht und löscht, was sehr wichtig ist aufgrund der High Availability.

Es gibt viele Container-Management-Systeme. Ich habe mir aber nur zwei Systeme näher angeschaut: Docker Swarm und Kubernetes.

Was es sonst noch geben würde:

- Azure Kubernetes Service AKS
- Diamante
- Googl GKE
- HyperV Containers
- Apache Mesos

Kubernetes

Kubernetes lässt sich auf jedem Betriebssystem installieren.

Kubernetes ist ein bisschen schwerer als Docker, man hat dafür aber eine bessere Skalierung, was sehr wichtig ist.

Docker Swarm

Docker Swarm ist wie Kubernetes eine Container-Orchestration. Der Vorteil bei Docker Swarm ist, dass man ein Interface hat mit bekannten Funktionen wie zum Beispiel Docker-Compose. Docker ist simpler als Kubernetes.

R Realisieren

Erste Schritte

Nachdem ich mir einen genaueren Plan gemacht habe, was ich für das Projekt brauche und was die Anforderungen sind, fing ich an das Ganze zu bauen. Da ich ein Projekt verbessere, hatte ich bereits die 13 Raspberry Pis zusammen, weshalb ich mir die nicht besorgen musste und ich keinen Plan machen musste, wie das Ganze aussieht.

So hatte ich alles zusammen um endlich mit dem Realisieren anfangen zu können. Als erstes habe ich alle SD Karten mit einem Betriebssystem, die Raspberry Pis mit Hyperiot und den «Router» mit Raspbian (mit GUI) geflashed. Nachdem ich alle Raspberry Pis aufgesetzt hatte, fing ich an den Router zu konfigurieren. Damit der Raspberry Pi überhaupt zu einem Router wird, habe ich dnsmasq installiert:

```
sudo apt-get install dnsmasq
```

Nachdem dnsmasq heruntergeladen wurde, fing ich an die einzelnen Raspberry Pis, die später zum Kubernetesnetz gehören, zu benennen. Zum Editieren habe ich dabei nano verwendet.

```
sudo nano /etc/dnsmasq.conf
```

Dort habe ich dann eine IP-Range vergeben. Zudem habe ich den einzelnen Hosts auch fixe IP-Adressen vergeben.

```
Dhcp-range,192.168.1.2,192.168.1.254, infinite

Dhcp-host, rp2,192.168.1.2, infinite
Dhcp-host, rp3,192.168.1.3, infinite
Dhcp-host, rp4,192.168.1.4, infinite
Dhcp-host, rp5,192.168.1.5, infinite
Dhcp-host, rp6,192.168.1.6, infinite
Dhcp-host, rp7,192.168.1.7, infinite
Dhcp-host, rp8,192.168.1.8, infinite
Dhcp-host, rp9,192.168.1.9, infinite
Dhcp-host, rp10,192.168.1.10, infinite
Dhcp-host, rp11,192.168.1.11, infinite
Dhcp-host, rp12,192.168.1.12, infinite
Dhcp-host, rp13,192.168.1.13, infinite
```

Nachdem ich den Router konfiguriert habe, habe ich den Raspberry Pis den „richtigen“ Namen gegeben, damit sie dann vom DHCP-Server die richtige IP-Adresse bekommen.

```
sudo raspi-config
```

Im raspi-config muss man auf die Nummer 2 drücken und den Hostnamen ändern. Der Name muss gleich sein wie der Name, den man im dnsmasq.conf File festgelegt hat (z.B. rp2).

Nachdem ich das gemacht habe, habe ich kontrolliert, dass die Namensgebung funktioniert. Dafür habe ich beim Router nslookup installiert:

```
sudo apt-get install dnsutils
```

Wenn man das Packet installiert hat, kann man den Befehl nslokup benutzen.

```
nslookup rp2
```

Wenn wir nun 192.168.1.2 als Antwort bekommen, wissen wir, dass es funktioniert.

Wenn wir aber nicht die richtige Antwort bekommen, müssen wir Datei /etc/network/interfaces ändern. Dazu geben wir folgenden Befehl ein:

```
nano /etc/network/interfaces
```

Nun gehen wir ans Ende der Datei und geben folgendes ein:

```
# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto eth0
iface eth0 inet static
 address 192.168.1.1
```

Damit die Änderungen in Kraft treten, geben wir noch zwei Befehle hinzu:

```
Ifdown eth0

Ifup eth0
```

Nun haben wir bereits die Grundstruktur. Damit man auf die Raspberry Pis etwas herunterladen kann, brauchen wir noch das IP-Forwarding, da sonst unser Router kein richtiger Router wäre. Dafür haben wir die IP-tables benutzt.

Die IP-tables muss man aber leider selber erstellen, so habe ich die Datei /etc/iptables.ipv4.nat erstellt.

```
# Generated
*filter
:INPUT ACCEPT [1600:133307]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [1197:133314]
-A FORWARD -i eth1 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth1 -o eth1 -j ACCEPT
COMMIT
# Completed on Wed
# Generated by iptables
*nat
:PREROUTING ACCEPT [236:18057]
:INPUT ACCEPT [57:4429]
:OUTPUT ACCEPT [14:908]
:POSTROUTINg ACCEPT [0:0]
-A POSTROUTING -o eth1 -j MASQUERADE
COMMIT
#Completed
```

Da wir die IP-tables auch abschalten können wollen, haben ich eine zweite Datei (/etc/iptables.ipv4.nonat) erstellt, die ich benutzen kann um das Routing zu verhindern.

```
#Generated by IPTABLES
*nat
:PREROUTING ACCEPT [2889:504160]
:INPUT ACCEPT [453:52628]
:OUTPUT ACCEPT [168:13008]
:POSTROUTING ACCEPT [5:371]
-A POSTROUTING -o wlan0 -j MASQUERADE
COMMIT
# Completed on Tue
# Generated
*filter
:INPUT ACCEPT [2032:315295]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [690:92340]
-A FORWARD -i wlan0 -o eth0 -m state --state RELATED,ESTABLISHED -j ACCEPT
-A FORWARD -i eth0 -o wlan0 -j ACCEPT
COMMIT
#Completed
```

Um das Forwarding zu aktivieren, gibt man folgendes ein:

```
sudo –i
echo 1 > /proc/sys/net/ipv4/ip-forwarding
```

Um das Forwarding zu deaktivieren, gibt man folgendes ein:

```
sudo –i
echo 0 > /proc/sys/net/ipv4/ip-forwarding
```

Um das NAT Routing zu aktivieren, gibt man folgendes ein:

```
sudo iptables-restore < /etc/iptales.ipv4.nat
```

Um das NAT Routing zu aktivieren, gibt man folgendes ein:

```
sudo iptables-restore < /etc/iptales.ipv4.nonat
```

Kubernetes installieren

Kubernetes muss man auf jedem Raspberry Pi installieren. Der einzige Raspberry Pi, der Kubernetes nicht installiert haben muss, ist der Router.

Ich habe die Kubernetes Version 1.13.5 benutzt.

```
apt-get update && apt-get install –y apt-transport-https curl
curl –s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
cat <<EOF >/etc/apt/sources.list.d/Kubernetes.list
deb https://apt.kubernetes.io/ Kubernetes-xenial main
EOF
apt-get update
apt-get install kubelet=1.13.5-00 kubeadm=1.13.5-00 kubectl=1.13.5-00
```

Wenn man das auf allen Nodes gemacht hat, muss man Kubernetes initialisieren. Dafür muss man auf dem Master einen Befehl eingeben der so aussieht:

```
kubeadm init –pod-network-cidr 10.244.0.0/16
```

Das –pod-network-cidr 10.244.0.0/16 ist dabei sehr wichtig, da wir damit festlegen, wie unser Overlaynetzwerk aussieht, wenn wir Flannel benutzen.

Daraufhin liefert uns Kubernetes einen Token, den wir bei den anderen Nodes benutzen müssen.

```
kubeadm join 192.168.1.10 –token=xxx sha255=xxx
```

Diesen Befehl muss man nun auf jedem Worker Node eingeben.
Wenn alle Raspberry Pis gejoined sind, muss man zur Master Node auf den „normalen Benutzer“ zurück um folgendes einzugeben:

```
sudo cp /etc/kubernetes/admin.conf $HOME/
sudo chown $(id -u):$(id -g) $HOME/admin.conf
export KUBECONFIG=$HOME/admin.conf
```

Diese drei Befehle muss man jedes Mal eingeben, wenn man mit Kubernetes arbeiten will und man die ssh Verbindung neu gestartet hat.

Nun kann man mit dem Befehl `kubectl get nodes` alle Nodes sehen, die mit dem Master verbunden sind. Wenn alles richtig gemacht wurde, sollten wir dort dreizehn Einträge haben und keiner ist Ready.

Die Nodes können noch nicht Ready sein, da sie noch in gar keinem Overlaynetzwerk sind und das kann man mit diesem Befehl beheben:

```
kubectl apply –f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/k8s-manifests/kube-flannel-legacy.yml
```

Wenn man nun `kubectl get pods –all-namespaces` eingibt, sollten alle Pods aufgelistet sein und alle sollten laufen.

Als nächstes installiert man einen kleinen Service auf Kubernetes:

```
kubectl run hypriot –image=hypriot/rpi-busybox-httpd –replicas=3 –port=80
```

Um den Service zu deployen, muss man noch einen Befehl zusätzlich eingeben:

```
kubectl expose deployment hypriot –port 80
```

Um zu testen, ob der Service gestartet ist, kann man die drei Endpunkte genauer ansehen:

```
kubectl get endpoints hypriot
```

Das Resultat gibt einem drei IP-Adressen aus, die man nun curlen kann:

```
curl (adresse)
```

Als Antwort erscheint folgendes:

```
HypriotOS/armv7: pirate@rp10 in ~
$ kubectl get endpoints hypriot
NAME      ENDPOINTS                                            AGE
hypriot   10.244.1.2:80,10.244.3.2:80,10.244.5.2:80   3d15h
HypriotOS/armv7: pirate@rp10 in ~
$ curl 10.244.1.2
<html>
<head><title>Pi armed with Docker by Hypriot</title>
  <body style="width: 100%; background-color: black;">
    <div id="main" style="margin: 100px auto 0 auto; width: 800px;">
      <img src="pi_armed_with_docker.jpg" alt="pi armed with docker" style="width: 800px">
    </div>
  </body>
</html>
HypriotOS/armv7: pirate@rp10 in ~
```

So wissen wir, dass wir den Service innerhalb des Clusters verwenden können. Wir müssen den Clustern aber auch ausserhalb des Clusters benutzen können.

Hierfür brauchen wir Traefik:

```
kubectl apply –f https://raw.githubusercontent.com/hypriot/rpi-traefik/master/traefik-k8s-example.yaml
```

Traefik ist perfekt für uns, da es Load-Balancing zur Verfügung stellt. Damit das Load-Balancing funktioniert, muss man eine Node auswählen, worüber der Datenfluss fliesst:

```
kubectl label node 192.168.1.11 nginx-controller=traefik
```

Mit diesem Befehl bestimmt man eine Node, auf den man von ausserhalb der Kubernetes Umgebung zugreifen kann.

Nun müssen wir dem Service angeben, dass er auf das Load-Balancing reagieren soll:

```
cat > hypriot-ingress.yaml <<EOF
apiVersion:extensions/v1beta1
kind: Ingress
metadata:
    name: hypriot
spec:
```

```
    rules:
  - http:
    paths:
    - path: /
      backend
          serviceName: hypriot
          servicePort: 80
EOF
```

Um zu kontrollieren, ob alle Pods laufen, geben wir den Befehl `kubectl get pods –all-namespace` ein.

Wenn alles auf Running ist, wissen wir, dass alles perfekt läuft.

Nun können wir den Webbrowser öffnen und die IP-Adresse angeben, die man Load-Balancing beigelegt hat, in unserem Fall 192.168.1.11.

Dashboard installieren

Da man nicht immer auf der Befehlszeile Pods erstellen und verwalten will, installieren wir ein Dashboard, welches uns alles vereinfacht.

Dafür erstellt man als Erstes das Deployment:

```
echo -n 'apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: kubernetes-dashboard
  labels:
    k8s-app: kubernetes-dashboard
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: kubernetes-dashboard
  namespace: kube-system' | kubectl apply -f –
```

Wenn man das gemacht hat, muss man das Ganze noch akzeptieren:

```
kubectl apply –f
https://raw.githubusercontent.com/kubernetes/dashboard/v1.10.1/src/deploy/alternativ/kubernetes-dashboard-arm.yaml
```

Nun muss man nur noch alles zulassen:

```
kubectl proxy –address 0.0.0.0 –accept-hosts ´.*’
```

Nun können wir uns das Dashboard im Webbrowser mit diesem Link anschauen

http://192.168.1.10:8001/api/v1/namespaces/kubesystem/services/http:kubernetes-dashboard:/proxy/

K Kontrolle

Test

Nun muss man kontrollieren, ob alle Anforderungen erfüllt wurden.

Alles das Grün gekennzeichnet ist, hat funktioniert, was hellrot angezeigt wird, hat nicht funktioniert.

	Was getestet wird	Wie es getestet wird	Resultat
1	Raspberry Pis haben den richtigen Namen	„Nslookup“ auf dem Router	Alle haben den richtigen Namen
2	Alle Rp sind mit dem Router verbunden		
3	Alle können ins Internet	Ping 8.8.8.8	Alle RP finden den Google DNS
4	Alle Worker haben eine Verbindung zum Master	„kubectl get nodes“ auf dem Master	Alle RP sind dort aufzufinden
5	Das Dashboard funktioniert	Suche im Webbrowser nach dem Dashboard	Dashboard wird angezeigt
6	Cluster funktioniert	Raspberry Pi vom Strom nehmen	Nach ausstecken des Raspberry Pis läuft es immer noch weiter
7	Man kommt auf die Load-Balancing Seite wenn man *.raspicluster.lan eingibt	Man sucht im Webbrowser nach *raspicluster.lan	Die gewünschte Seite kommt
8	Webservice funktioniert	Man sucht im Browser nach der Webseite	Man bekommt einen 404 Error

A Auswertung

Wurden alle Ziele erreicht?

Leider wurden nicht alle Ziele erreicht, da ich nicht genügend Zeit hatte, das Problem mit dem 404 Error aus dem Weg zu schaffen. Es weiss zwar, dass etwas auf dieser Seite wäre, findet die Seite aber nicht, weshalb ich stark davon ausgehe, dass das Problem bei der Konfiguration und nicht bei Traefik oder dem Load-Balancing liegt.

Fazit

Das Projekt war sehr lehrreich und hat mir stark mit dem Verständnis von Kubernetes geholfen. Ich bekam bereits einmal eine kleine (1h) Präsentation von Kubernetes, welche bei mir viele Fragen hinterlassen hat. Diese Fragen konnte ich mit diesem Projekt nach und nach selber beantworten. Ich hatte sehr viel Spass an diesem Projekt, da ich mich immer wieder gefreute habe, wenn ich ein Problem beseitigen konnte. Leider war das Projekt auch sehr anstrengend, da es noch sehr viele Bugs hat und Kubernetes nicht wirklich für Raspberry Pis geeignet ist. Das merkt man zum Beispiel darin, dass es nur Flannel als Overlay-Netzwerklösung für ARM Prozessoren gab. Ich hatte immer wieder Probleme, die sich am Schluss als Bug herausstellten, welche ich nicht beeinflussen konnte. Das Probieren mehrerer Versionen ist sehr zeitaufwendig und macht zudem nicht besonders viel Spass. Wie man bei der

Kontrolle sehen kann, konnte ich leider nicht alles fertig stellen, da ich für das letzte Problem mehr Zeit benötigte, als ich noch zur Verfügung hatte. Zur Fertigstellung des Projektes brauche ich schätzungsweise noch einen Tag mehr Zeit.

Arbeitsjournal

Tag 1

Am heutigen Tag habe ich mich mit dem Laptop richtig eingerichtet und nachgesehen, was alles gemacht werden muss, damit ich eine Checkliste erstellen kann. Die Checkliste habe ich erstellt, damit ich alle Arbeitsschritte kenne und am Schluss nachsehen kann, ob alles gemacht wurde. Offene Fragen konnte ich dabei am Ende des Tages Marc fragen.

Tag 2

An diesem Tag fing ich an mich zu informieren, damit ich weiss, was ich machen muss und wie es ungefähr realisiert wird. Da ich noch gar keine Ahnung von Kubernetes hatte, machte ich einen groben Zeitplan von dem ganzen Projekt. Zudem fing ich an, mir selber Fragen zu stellen, wie zum Beispiel, welche Kubernetes Distribution ich verwenden werde.

Tag 3

Heute habe ich versucht, alle Fragen zu beantworten. Ich hatte noch ein Standortgespräch mit Marc, der mir helfen konnte, Kubernetes besser zu verstehen. Ich konnte sehr viele Fragen beantworten, leider sind aber auch welche hinzugekommen, da ich davor gar nicht wusste, dass es benötigt wird.

Tag 4

Ich habe mich weiter informiert, was ich alles benutzen werde. Oft habe ich auch die Unterschiede zwischen zwei Systemen nicht genau verstanden, weshalb ich viel Zeit verloren habe, überhaupt zu verstehen, was der Unterschied zwischen den Systemen ist. Ein Beispiel davon ist das Overlaynetzwerk Flannel. Ich hatte heute wieder ein Standortgespräch mit Marc, bei dem ich mehrere neue Anforderungen bekommen habe, die ich auch berücksichtigen sollte.

Tag 5

Heute konnte ich die restlichen Fragen beantworten, damit ich endlich mit dem Realisieren anfangen kann. Leider ging das Informieren aber länger als geplant, weshalb ich nun hoffte, dass das Realisieren weniger lange geht, als ich bei meinem Zeitplan angegeben habe.

Am Ende des Tages konnte ich das Informieren, Planen und Entscheiden beenden und mit dem Realisieren anfangen.

Tag 6

Als erstes habe ich alle Raspberry Pis im Zimmer HCP G 36.3 aufgestellt. Dann habe ich alle Micro SD Karten geflashed und die aktuellste Hypriot Version darauf geladen. Ich habe auf allen Raspberry Pis Hypriot gebootet, ausser auf dem Router. Dort habe ich Raspbian ohne GUI installiert.

Auf dem Raspbian habe ich zudem den DHCP und DNS Service konfiguriert, damit die Worker und der Master eine IP-Adresse bekommen.

Tag 7

Heute hatte ich mein letztes Standortgespräch mit Marc. Er hat mir ein paar Fehler in meiner Dokumentation und Erweiterungsmöglichkeiten gezeigt. So versuchte ich am Anfang eine SSH Verbindung mittels Private-key mit dem Router zu erstellen, bis mir Marc gesagt hat, dass eine derartige Verbindung nicht wirklich schlau wäre, da so nur ich mich mit dem Router verbinden könnte und er das auch können muss. Ich sollte eigentlich eine Private-key Verbindung zwischen Router und Nodes erstellen. Am Ende des Tages habe ich den Master initialisiert/ aufgesetzt.

Tag 8

Ich versuchte heute alle Worker mit dem Master zu verbinden, was leider aber nicht funktioniert hat. Nach längerem recherchieren, fand ich heraus, dass der Token der beim initialisieren erstellt wird nur 24 Stunden zur Verfügung steht und das Leider bereits um ist, da ich im Wochenende war. So musste ich einen neuen Token generieren, womit ich alle mit dem Verbinden konnte.

Ich habe heute auch mitbekommen, dass Marc leider für die nächste Woche nicht zur Verfügung steht, da er mit den erst Lehrjahrlinge in ein Arbeitslager fährt.

Tag 9

Heute versuchte ich das Overlay-Netzwerk zu erstellen, leider funktionierte das aber nicht, da die Flannel Pods in einen Crashloop gerieten. Nach einer längeren Recherche fand ich heraus, dass es sich dabei um einen Bug handelt und man die Kubernetes Version 1.13.5 verwenden sollte. Nachdem ich kubeadm auf allen Nodes neugestartet habe, kam leider wieder die gleiche Fehlermeldung, die ich am Anfang nicht verstand. Gegen Ende des Tages, fand ich heraus, dass das Wiederherstellen von Kubernetes nicht alle Konfigurationen löscht, weshalb ich alle Raspberry Pis neu geflashed habe. Nun musste ich anfangen aufzupassen, dass ich die Micro Karten nicht zu oft flashe, da die Karten nur eine bestimmte Zahl geflashed wird.

Tag 10

Da ich leider nicht so weit gekommen bin, wie ich erhoffte, musste ich einen Point-of-no-return erstellen, damit ich am Schluss auch eine Präsentation abgeben kann und nicht immer noch versuche das Projekt fertig zu erstellen. Dieser Tag war bei mir der Tag 11.

Ich versuchte heute soweit zukommen wie möglich, damit ich am Tag11 am Morgen beenden kann. Nachdem ich alles neu geflashed habe und alles wieder aufgesetzt wurde, habe ich Flannel installiert und es funktionierte. So konnte ich einen kleinen Webservice starten, was ziemlich gut funktioniert hat. Nachdem der Test für den Webservice erfolgreich war, versuchte ich den mit Traefik an die Nodes ausserhalb der Kubernetes-Umgebung freizuschalten, was so ausgesehen hat, als ob es funktioniert, beim anschliessendem Testen, habe ich aber bemerkt, dass anstelle der Webseite einen 404 Error Code ausgegeben wird. Ich versuchte herauszufinden, wo der Fehler lag und kam zu dem Ergebnis, dass der Fehler bei Traefik sein muss und nicht bei dem Service, da der Service innerhalb der Kubernetes Umgebung funktioniert hat.

Tag 11

An diesem Tag habe ich das Kubernetes Dashboard erstellt und versucht, das Problem mit Traefik zu lösen. Nach einer längeren Recherche musste ich leider das Projekt abbrechen, da ich sonst bei der Präsentation nichts abgeben kann.

Tag 12

Am heutigen Tag habe ich die Dokumentation und Präsentation fertig geschrieben.