Hochschule Rhein-Waal
Fakultät Kommunikation und Umwelt

Praxissemesterbericht
SS 2020
Modul: Praxissemester

Erreichbarkeit und Orchestrierung eines Docker-basierten-Systems
Am Beispiel eines Node-RED, InfluxDB und Grafana Verbunds

Jan Sonntag

Matrikelnummer:
23307

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Problemstellung

1.1 Problemstellung

Für den rein online stattfindenden Kurs Applied Measurements and Controls aus dem Studiengang Environment and Energy mussten mehrere Instanzen des Programmier-Tools Node-RED mit Anbindung an eine Datenbank und dem Frontend-Tool Grafana, welches dem Darstellen von Grafen dient, erstellt werden. Wichtig war eine Isolierung der einzelnen Instanzen und die Erreichbarkeit über das World-Wide-Web.

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, die Implementierung der gewählten Lösung zu erklären und auf mögliche Probleme und Chancen für spätere Updates hinzuweisen. Dafür wird das System in dem oben genannten Kurs eingesetzt und getestet.

Zum Schluss wird eingeschätzt wie praktikabel die beschriebene Lösung ist und wie die Wahl der verwendeten Technologien sich darauf auswirkt.

2. Technische Grundlage

2.1 IoT / ESP

Das Internet of Thing (kurz IoT), also das Internet der Dinge, wird definiert als eine dynamische globale Netzwerkinfrastruktur, welche auf standardisierten und interoperablen Kommunikationsprotokollen basiert, in welchen sowohl physisch als auch virtuell existierende „Dinge“ eine Identität, Attribute und intelligente Schnittstellen benutzen und so in das ganze Informationsnetzwerk eingebunden werden (Vermesan & Friess, 2013, S. 3). IoT umfasst also viele kleine Dinge und integriert sie in ein großes Netzwerk. Es besteht eine Ähnlichkeit zur Maschine-zu-Maschine-Kommunikation. Der Unterschied liegt darin, dass es im IoT mehr um die Technologie an sich und wie sie angewendet wird, sowie die Überwachung und Kontrolle aus der Ferne, geht (Holler et. Al., 2014, S. 14). Besonders der Punkt „Überwachung von Sensor gestützten Daten“ ist im Zusammenhang mit weiteren Technologien dieser Arbeit entscheidend.

Holler et. Al. (2014, S.11) nennen mehrere Gründe warum sich das IoT in den letzten Jahren mehr und mehr durchgesetzt hat:

1. *An increased need for understanding the physical environment in its various forms, from industrial installations through to public spaces and consumer demands. These requirements are often driven by efficiency improvements, sustainability objectives, or improved health and safety (Singh 2012).*
2. *The improvement of technology and improved networking capabilities.*
3. *Reduced costs of components and the ability to more cheaply collect and analyze the data they produce.*

Besonders die Punkte zwei und drei von Holler et. Al. sind im studentischen Umfeld von bedeutender Wichtigkeit. Dies zeigt sich deutlich am Beispiel der Mikrokontrollerfamilie ESP. ESPs zeichnen sich durch ihre gute Performance, Kompaktheit und ganz besonders durch ihren sehr niedrigen Preis aus. Mit einfachen Mitteln ist es so möglich Sensoren oder Akteure mit modernen Technologien wie Wi-Fi oder Bluetooth zu entwickeln und zu bauen (Kurniawan, 2019).

2.2 NIG

NIG ist lediglich eine Abkürzung für die einzelnen Komponenten **N**ode-RED, **I**nfluxDB und **G**rafana und dient als Bezeichnung für das ganze System. Das ganze System mit allen Containern ist das NIG-System und besteht wiederrum aus einzelnen NIG-Systemen.

2.2.1 Node-RED

Node-RED ist ein webbasiertes grafisches open-source[1] Programmiertool, welches dazu dient Daten zu empfangen, zu senden und zu modifizieren bzw. zu verarbeiten. Als Serverarchitektur dient Node.js. (Blackstock & Lea, 2016) Programmiert wird mit einzelnen vorgefertigten Funktionsblöcken, aber es gibt auch die Möglichkeit mit eigenem JavaSkript-Code zu arbeiten.

Programme werden in Node-RED als Flow bezeichnet. In einer Node-RED-Instanz können mehrere Flows nebeneinander funktionieren und arbeiten. Auch bietet das Tool viele verschiedene Wege Daten zu empfangen und zu senden. Zu den gängigsten gehören http-Requests[2] und das MQTT[3] Protokoll. Viele weitere werden durch Plugins unterstützt. Dadurch wird eine hohe Kompatibilität zu verschiedenen Schnittstellen und Protokollen sichergestellt. Das Projekt wird durch eine große Community unterstützt und weiterentwickelt.

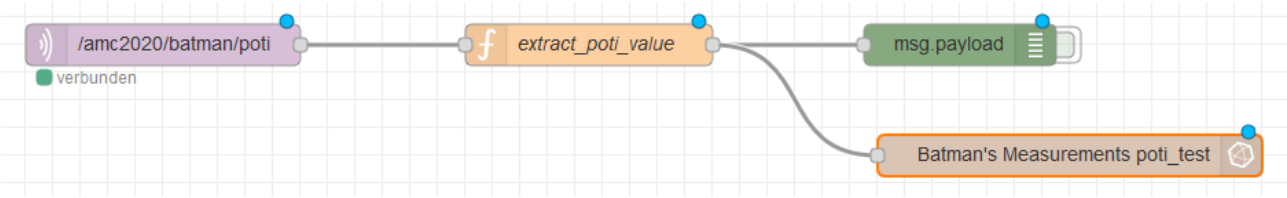


Abbildung 1 – Beispiel für einen Flow

In Abbildung 1 ist ein beispielhafter Flow zu sehen. Der Beginn des Flows ist ein „mqtt in“-Block. Dieser ist so konfiguriert, dass er mit dem MQTT-Server der HSRW verbunden ist und alle Nachrichten zum Thema „/amc2020/batman/poti“ empfängt. Werden Daten empfangen, werden diese mit einem selbst programmierten Funktionsblock ausgelesen und dann an einen Block weitergegeben, welcher die Variable auf der Konsole[4] ausgibt. Auch wird die Variable an einen

[1] Bezeichnet meist, dass das Tool/Programm frei zugänglichen Quellcode hat. Dieser kann von Jedem eingesehen werden.
[2] Das Standard-Protokoll des Internets
[3] Spezielles Netzwerkprotokoll für Maschine-zu-Maschine Kommunikations
[4] Auch bekannt als Kommandozeile, In Node-RED können jedoch keine Kommandos eingegeben werden

InfluxDB (siehe Kapitel 2.2.2) Block weitergegeben, welcher diese dann in einer Datenbank abspeichert.

2.2.2 InfluxDB

InfluxDB ist eine schemenfreie open-source Zeitreihendatenbank und zeichnet sich dabei besonders durch ihre Schnelligkeit und das Nicht-Nutzen von externen Komponenten aus (Naqvi & Yfantidou, 2017, S.8, S.36ff).

Eine Zeitreihendatenbank ist extra für Anwendungen entwickelt worden, um zeitlich nacheinander in einem größeren Zeitraum aufgenommene Daten zu speichern (What is a time series database?, o.J.). Eine solche Datenbank ist also perfekt für viele Bereiche des Internet der Dinge geeignet. Dazu zählt bspw. das Speichern von Sensordaten. Solche Messwerte können am besten in einer solchen Zeitreihendatenbank abgespeichert werden.

2.2.3 Grafana

Grafana ist ein open-source Dashboard-Tool zum Anzeigen von Zeitreihen- und momentan Messdaten. Dabei bietet es die Möglichkeiten die Daten auf verschiedene Weise zu visualisieren. Weiterhin können auch Alarme erstellt werden, sodass, sollten bestimmte Werte eine gewisse Grenze unter- oder überschreiten, ein Alarm an einen vorher definierten Dienst gesendet werden kann. Ein Punkt den Grafana besonders auszeichnet ist die Vielzahl von möglichen Datenquellen, welche direkt in Grafana implementiert sind. Weiterhin lassen sich oftmals fehlende Funktionen mit Hilfe diverser Plugins nachträglich installieren. (Grafana Features, o.J.)

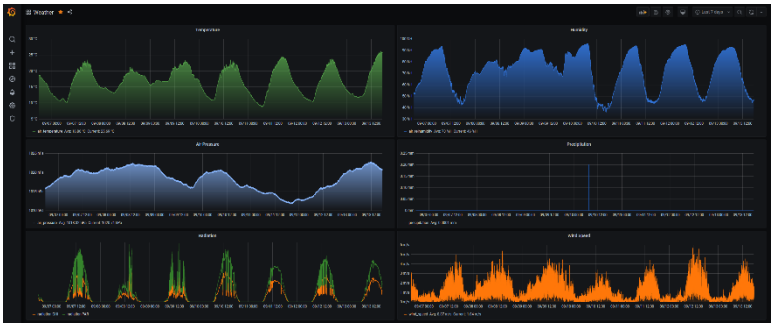


Abbildung 2 – Ein Grafana Dashboard

2.3 Server-Struktur

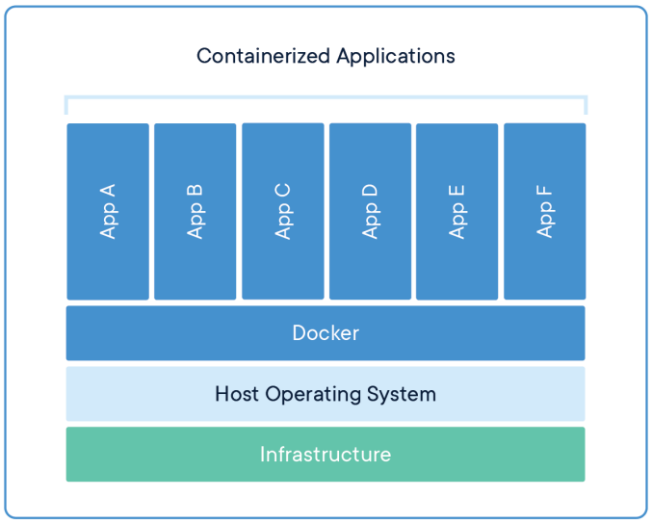
2.3.1 Docker

Bei Docker handelt es sich um ein open-source (Apache 2.0 Lizenz) System um sogenannte Container zu betreiben und zu verwalten. Dies dient im besonderen dazu Code effizient in kurzer Zeit auf diversen Systemen zum lauffähig zu machen. (Turnbull, 2019, S.7-8)

Das komplette Docker-System besteht aus drei Kern-Komponenten:

- Docker Engine
- Docker Images
- Docker Containers

Ein Docker Image ist dabei die Bauanleitung für einen Docker Container. Das Image wird Schritt für Schritt mit Hilfe von einfachen Befehlen, wie bspw. „Kopiere Datei“ oder „Öffne einen Port“ aufgebaut (Turnbull, 2019, S. 12). Zur Laufzeit werden dann die Docker Images zu Docker Containern.



Quelle: (o.J.) What is a Container?, Verfübar unter: <https://www.docker.com/resources/what-container> [aufgerufen am 14. September 2020].

Abbildung 3 – Aufbau eines Docker-Systems

Docker Container enthalten Applikationen, also einzelne Programme und deren notwendige systeminternen Bibliotheken. Dabei sind die Container vom Host-Betriebssystem isoliert, was ihre Sicherheit und die des ganzen Systems erhöht. Dadurch, dass sie lediglich die Anforderungen für diese spezifische Software enthalten und nicht auch einen Betriebssystem-Kernel[5], sind sie deutlich kleiner und schneller als eine virtuelle Maschine. Eine virtuelle Maschine virtualisiert und emuliert Hardware, wohingegen Docker Container auf Ebene der Software agieren. (What is a Container?, o. J.)

Die Docker Engine ist das Bindeglied zwischen den Docker Containern und dem Host Betriebssystem. Sie regelt die Kommunikation und verteilt dabei System Ressourcen und gibt Zugriff auf Basisfunktionalität des Betriebssystems. Kontrolliert werden kann die Docker Engine mit Hilfe einer Restful-API[6]. (Turnbull, 2019, S. 10) Durch diese Restful API werden mehrere Orchestrationstools von Docker unterstützt. Diese Tools ermöglichen es auf einfache Weise viele verschiedene Container gleichzeitig zu verwalten und zu betreiben. Beispiele dafür sind Kubernetes und Docker Compose. Docker Compose ermöglicht es eine Konfigurationsdatei zuschreiben, welche alle Services, persistenten Speicherorte, Netzwerke und Abhängigkeiten beschreibt und das für mehrere Docker Container gleichzeitig (Smith, 2017, S. 32).

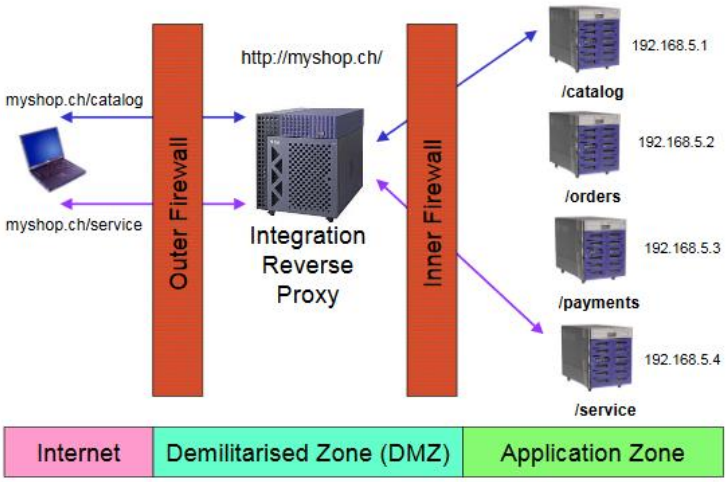
2.3.2 Reverse Proxy

Nach Sommerlad gibt es zwei verschiedene Typen von Reverse Proxys:

- Protection Reverse Proxy
- Integration Reverse Proxy

[5] Organisationseinheit eines Betriebssystems für Prozesse und Daten
[6] **Re**presentational **S**tate **T**ransfer – Eine Ressource kann mit mehreren Methoden erstellt, modifiziert und gelöscht werden | **A**pplication **P**rogramming **I**nterface – Eine Programmierschnittstelle

Wichtig für den hier in der Thesis beschriebenen Anwendungsfall ist vorerst nur das Prinzip des Integration Reverse Proxys. Dieser ermöglicht es mehrere Server unter einem gewöhnlichen Hostnamen und einer IP laufen zu lassen. Der Reverse Proxy kann dann die einkommenden Anfragen auf die unterschiedlichen Backend-Server verteilen. Unter einer Adresse können so mehrere Server und Anwendungen verwaltet werden. Auch ist es einfach möglich die Applikationen von einem Backend-Server auf einen anderen zu verschieben ohne, dass dies nach außen hin sichtbar wird. Ein weiterer Vorteil ist die verschlüsselte Kommunikation zwischen dem Server und dem Client/Nutzer. (Sommerlad, 2003) Die Integration neuer Applikationen ist deutlich einfacher als bei einer direkten Anbindung an das Internet da lediglich die Verbindung zwischen Client und Reverse Proxy verschlüsselt (bspw. https-Protokoll) sein muss. Die Verbindung zwischen Reverse Proxy und Backend-Server funktioniert dann ganz ohne Verschlüsselung (bspw. http-Protokoll).



Quelle: Sommerlad, Peter (2003) Reverse Proxy Patterns, Conference: Proceedings of the 8th European Conference on Pattern Languages of Programms (EuroPLoP '2003), Irsee: Peter Sommerlad

Abbildung 4 – Wie ein Reverse Proxy funktioniert

2.4 Kontext

Das ganze System steht im engen Zusammenhang zu dem Modul Applied Measurement and Control des Studiengangs Environment and Energy an der Hochschule Rhein-Waal. In diesem Kurs soll den Studierenden vermittelt werden wie Sie eigene hardwarenahe Systeme aus Sensoren und Aktuatoren selbst erstellen können. Dabei werden sowohl Software-, als auch Hardwareaspekte betrachtet. (Modulhandbuch Environment and Energy, 2018)

Die Möglichkeit dauerhaft bereitstehende Tools wie Node-RED, Grafana und eine Zeitreihendatenbank zu verwenden, bietet einen großen Mehrwert für die Studierenden und gibt Ihnen die Chance mehr praktische Erfahrung im Bereich Mess- und Kontrollsysteme zu erlangen.

3. Analyse

3.1 Szenarien

3.1.1 Ein neuer Kurs beginnt

Zu Beginn eines Semesters tragen sich alle Studierenden in den Kurs ein, bspw. über die Lernplattform Moodle. Die Studenten haben dann die Möglichkeit an den Vorlesungen und Übungen des Kurses teilzunehmen. Im Laufe des Semesters erklärt der Professor verschiedene Technologien

und Möglichkeiten zum Thema Internet der Dinge und das Sammeln von sensorgestützten Daten. Um den Studierenden die Möglichkeit zu geben auch praktische Erfahrung zu diesen Themen zu sammeln, werden jenen mehrere Tools zur Verfügung gestellt. Die Studierenden finden sich in kleinen Gruppen zusammen und erhalten dann Zugriff auf Instanzen von Node-RED und Grafana. Der Professor muss für jede Gruppe solche Instanzen zur Verfügung stellen und entscheiden wie er die Instanzen aufteilt und welche Login-Daten für die jeweilige Gruppe verwendet werden sollen. Hat er dies entschieden und alles geplant kann er die Instanzen auf einem von ihm gewählten Server aufsetzen und die Login-Daten an die Studierenden weitergeben. Anschließend erklärt er diesen wie die Tools zu benutzen sind und welche Möglichkeiten sie bieten.

3.1.2 Ein Kurs endet

Wenn ein Kurs endet und die Prüfungsleistungen im Hochschulnotensystem eingetragen wurden, hat der Professor mehrere Möglichkeiten mit den Instanzen zu verfahren. Wenn auf dem gewählten Server noch Ressourcen frei sind, könnte er die Instanzen weiter betreiben und so den Studierenden die Möglichkeit geben diese weiter zu verwenden. So könnte er den Studierenden mitteilen, dass sie die Instanzen für ein weiteres Jahr benutzen können bis diese abgeschaltet und die Daten endgültig gelöscht werden würden. Auch hat der Professor die Option die Instanzen und jegliche Daten Dritter komplett zu löschen. Vorangehend könnte er die persistenten Daten den jeweiligen Gruppen zu kommen lassen, um diesen die Möglichkeit zu geben an den Projekten weiter zu arbeiten.

3.1.3 Eine Gruppe hat ein Problem

Sollte die Gruppe ein Problem haben, welches Sie nicht selbst lösen kann, wird diese Gruppe den Professor darauf ansprechen. Es besteht die Möglichkeit, dass es sich um ein Problem mit der verwendeten Software an sich handelt. Der Professor kann dann den Stand der momentanen Arbeit der Studierenden ansehen und entscheiden ob das Problem aus der Implementierung dieser hervorgeht. Sollte dies der Fall sein kann er die Studierenden darauf hinweisen und ihnen im Idealfall sogar Tipps geben, wie sie etwaige Probleme beheben können. Sollte andererseits der Fall eintreten, dass das Problem bei der verwendeten Software liegt, muss der Professor schauen ob die Software intern Fehler auf der Konsole ausgibt und wenn dies der Fall ist recherchieren ob das Problem schon bekannt ist. Aus dieser Recherche muss er dann entscheiden ob das Problem nicht behoben werden kann oder ob er das Problem durch bspw. einen Neustart der Software oder durch Modifizierung dieser beheben kann. Sollte es keine Lösung geben und die Software ist im momentanen Stadium irreparabel und nicht benutzbar sein, muss diese inklusive Löschung aller Daten neu aufgesetzt werden.

3.1.4 Backup

Der Professor muss im Besonderen zu späteren Zeitpunkten der Projekte regelmäßig Backups der persistenten Daten aller Instanzen anlegen. Dabei besteht für ihn die Möglichkeit diese Daten lediglich in einem anderen Verzeichnis auf dem Server zu speichern. Diese Option hat den Vorteil, dass es sehr schnell und einfach zu händeln ist. Nachteilig ist wiederrum, dass sollte das ganze Betriebssystem kompromittiert oder fehlerhaft sein, alle Daten inklusive Backups nicht mehr vorhanden wären. Deswegen kann der Professor sich auch entscheiden die Backups auf einem externen Datenspeichermedium zu speichern. Dies dauert zwar länger, dafür sind die Daten aber sicherer vor Systemfehlern.

3.2 Anforderungsanalyse

3.2.1 Funktionale Anforderung

3.2.1.1 Neues System hinzufügen

Für einen Systemadministrator muss es möglich sein während des laufenden Betriebs ein neues System hinzuzufügen, ohne dass andere gerade laufende Systeme davon beeinflusst werden.

3.2.1.2 System entfernen

Für einen Systemadministrator muss es möglich sein während des laufenden Betriebs ein bereits bestehendes System zu entfernen, ohne dass andere gerade laufende Systeme davon beeinflusst werden.

3.2.1.3 Einzelnes System verwalten

Für einen Systemadministrator muss es möglich sein während des laufenden Betriebs ein bereits bestehendes System einer Gruppe zu verwalten, ohne dass andere gerade laufende Systeme davon beeinflusst werden.

Dazu gehört das Neustarten, Stoppen und Starten einzelner Systemkomponenten, aber auch das Sichern von Dateien eines bestimmten Systems.

3.2.1.4 Diverse Dateninputs in Node-RED

Node-RED muss so konfiguriert sein, dass es für einen Nutzer möglich ist viele verschiedene Arten von Dateninputs zu verwenden. Dazu gehört auch das nachträglich installieren und deinstallieren von Plugins.

Zudem sollte es standardmäßig einen speziellen Endpunkt für http-Requests geben.

3.2.1.5 Datenbank

Der Nutzer muss zu jeder Zeit die Möglichkeit haben Daten in einer Zeitreihendatenbank zu speichern und diese zu jedem Zeitpunkt auch wieder abzurufen. Auch muss er die Möglichkeit haben die

Datenbank zumindest teilweise verändern zu können. Bspw. neue Tabellen anlegen oder bereits vorhandene Tabellen wieder löschen.

3.2.2 Nicht-funktionale Anforderung

3.2.2.1 Erreichbarkeit

Das System muss stets erreichbar sein für die Kursteilnehmer. Dies ist eine Grundvoraussetzung, was vor allem an der Relevanz für die Prüfungsleistung liegt. Sollte aus etwaigen Gründen eine Erreichbarkeit nicht gewährleistet werden können, kann es nicht nur zu Problemen mit der Prüfungsleistung kommen, sondern im Extremfall sogar zu Fehlern oder Lücken in Messreihen, welche die Daten vielleicht nicht mehr nutzbar machen.

3.2.2.2 Sicherheit

Alle verwendeten Komponenten im System müssen nach aktuellen Sicherheitsmaßnahmen geschützt sein. Dies gilt nicht nur für unbefugte Zugriffe auf Komponenten von außen. Die Verbindungen zwischen dem Server und dem Client müssen verschlüsselt und so für Andere unlesbar sein. Auch sollte verhindert werden, dass die Datenbank oder andere Dienste direkt aus dem Internet erreicht werden können.

Weiterhin ist es wichtig, dass auch bei Systemausfall die Daten stets permanent in einem Zwischenspeicher liegen und nicht nur in einer virtuellen Umgebung. Die Sicherheit der Daten ist besonders für Studenten wichtig, da es bei Ihnen um eine Prüfungsleistung geht.

3.2.2.3 Ressourcen

Bei den Ressourcenanforderung handelt es sich sowohl um Zeit-, als auch um Hardwareressourcen. Das System muss in möglichst kurzer Zeit voll und ganz einsatzbereit sein. Dafür sollten zumindest die wichtigsten Funktionen implementiert sein.

Auch steht nur eine endliche Menge an Hardwareressourcen zur Verfügung. Eine mögliche Lösung muss mit minimalen Anforderungen an Hardwareressourcen lauffähig und trotzdem sowohl simpel aufgebaut als auch vollfunktional sein.

3.2.2.4 Wiederverwendbarkeit

Eine der wichtigsten Anforderungen stellt die Wiederverwendbarkeit dar. Dieses System darf nicht darauf ausgerichtet sein nur einmal benutzt zu werden. Es muss auch in den folgenden Semestern weiter zum Einsatz kommen können und stets erweiterbar und veränderbar sein. Dabei ist es wichtig, dass eine gute Dokumentation vorliegt sodass Jeder das System von Grund auf neu aufbauen und einsetzen kann.

3.3 Fazit

Die unterschiedlichen Szenarien zeigen, dass es wichtig ist, dass das eingesetzte System besonders fehlerfrei funktioniert. Es darf für den Professor und auch für die Studierenden kein Hindernis sein es zu benutzen, sondern muss einen sofortigen Mehrwert bieten. Besonders die Sicherheit der Daten der Studierenden hat sich als ein wichtiger Aspekt des Systems herausgestellt.

Eine der nicht-funktionalen Anforderung wurde bei der Ideenfindung zu diesem System explizit gewünscht: Die Wiederverwendbarkeit. Dies bedeutet im Umkehrschluss aber nicht, dass die anderen Anforderungen weniger Bedeutung haben.

4. Architektur

4.1 Verwendete Architektur

Das System lässt sich grundlegend in zwei Hauptteile unterteilen:

- Das Host-Betriebssystem
- Die Docker-Systeme

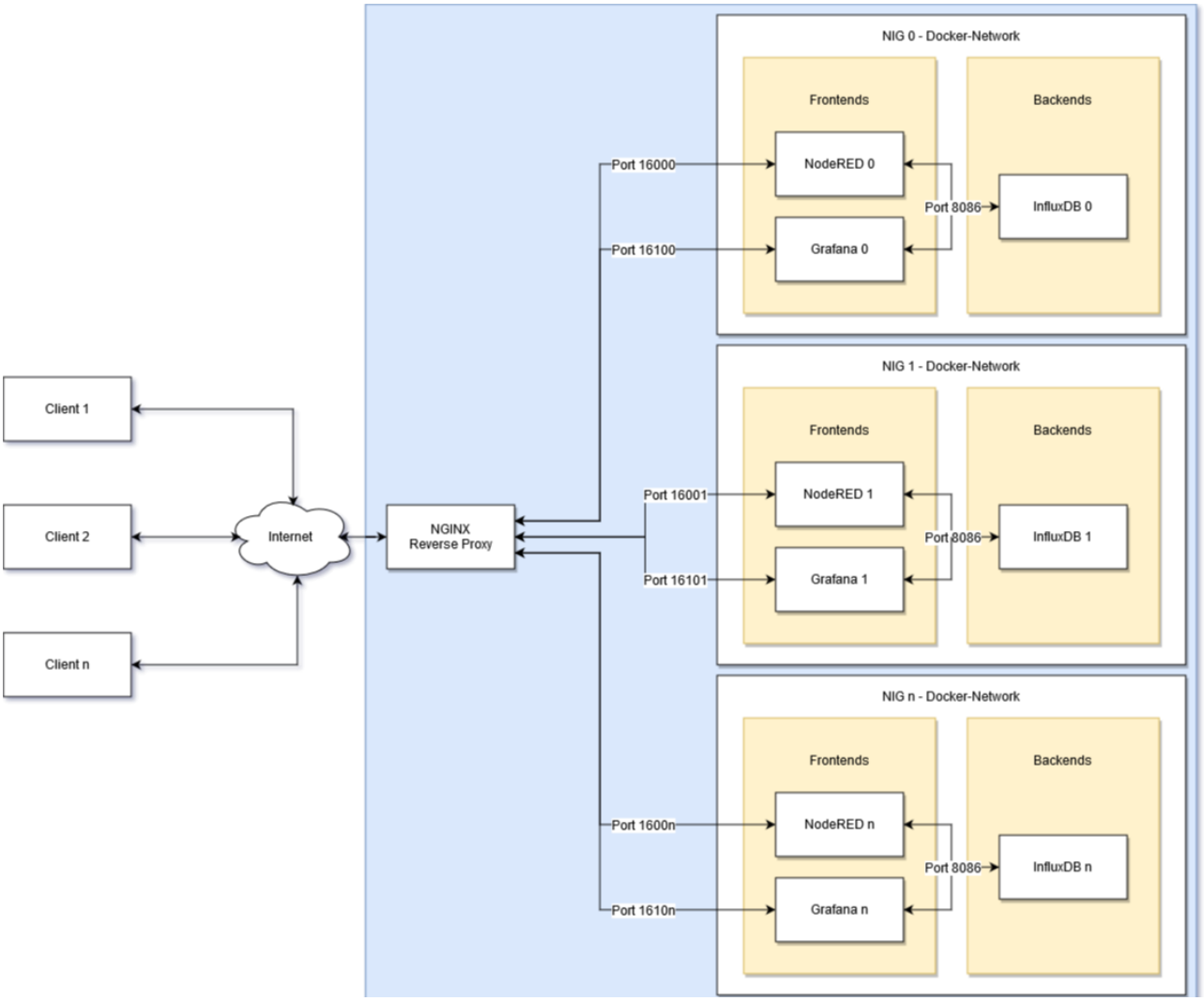


Abbildung 5 – Verbindungen im NIG-System

Im Host-Betriebssystem läuft ein NGINX Reverse Proxy. Dieser leitet alle Anfragen auf die einzelnen Docker-Container um. Wohin eine Anfrage umgeleitet wird hängt von der URL ab. Alle Docker-Container sind unter einer Subdomain erreichbar. Eine Unterscheidung findet lediglich durch den angegebenen Verzeichnispfad statt. Dieser Verzeichnispfad kann bei Erstellung eines neuen NIG-Systems definiert werden. Eine mögliche URL kann bspw. so aussehen:

nig.eolab.de/groupZ/grafana

Der Teil „groupZ“ im Verzeichnispfad ist die Root-Adresse des NIG-Systems. Danach folgend kann der Reverse Proxy auf Node-RED oder Grafana umleiten. Das System würde sich auch durch weitere Container erweitern lassen. Ein weiterer wichtiger Teil der auf dem Host-Betriebssystem abläuft ist die persistente Speicherung von Daten. Die Container sind so eingerichtet, dass die Daten sowohl im einzelnen Container enthalten sind als auch auf dem Host-Betriebssystem selbst. Auch wenn ein Container nicht mehr funktionieren oder gar gelöscht werden sollte, sind die Daten noch verfügbar. Dies ermöglicht auch die Daten einzelner Container zu sichern und die jeweiligen Container und Systeme direkt zu verwalten und zu ändern.

Der andere Hauptteil sind die Docker-NIG-Systeme. Jedes NIG-System hat sein eigenes virtuelles Netzwerk. Es gibt lediglich zwei Ports, welche von außen offen sind. Diese Ports sind jeweils einmal für Node-RED und einmal für Grafana. Da Docker es ermöglicht einzelne Ports (umzulenken) ist es möglich mehrere Instanzen derselben Software nebeneinander zu betreiben. Ein Beispiel:

Standardmäßig nutzt Node-RED den Port 1880. Dieser Port lässt sich normalerweise ohne weiteres auch nicht ändern. Da Node-RED aber in einem eigenen Container ist, kann dieser extra dafür konfiguriert werden. So wird bspw. nach außen hin für das Host-Betriebssystem aus dem Port 1880 der Port 16000. Die Software im Inneren des Containers wird davon nicht beeinflusst und funktioniert weiterhin normal.

Eine andere Node-RED Instanz in einem anderen Container benutzt dann bspw. Port 16001 und so weiter. Gleiches gilt für Grafana. Der Container der Datenbank hat für das Host-Betriebssystem keinen offenen Port, aber im jeweiligen virtuellen Docker-Netzwerk schon. Dadurch, dass der Container von außen nicht erreichbar ist und es auch nicht sein soll muss der Port nicht verändert werden. Die Kommunikation funktioniert weiter über den Standardmäßigen Port von InfluxDB (Port 8086).

In Abbildung 6 ist ein beispielhafter und vereinfachter https-Request vom Client dargestellt. Erkennbar ist der Zusammenhang der unterschiedlichen Ports. Es ist lediglich ein Teil der Kommunikation dargestellt.

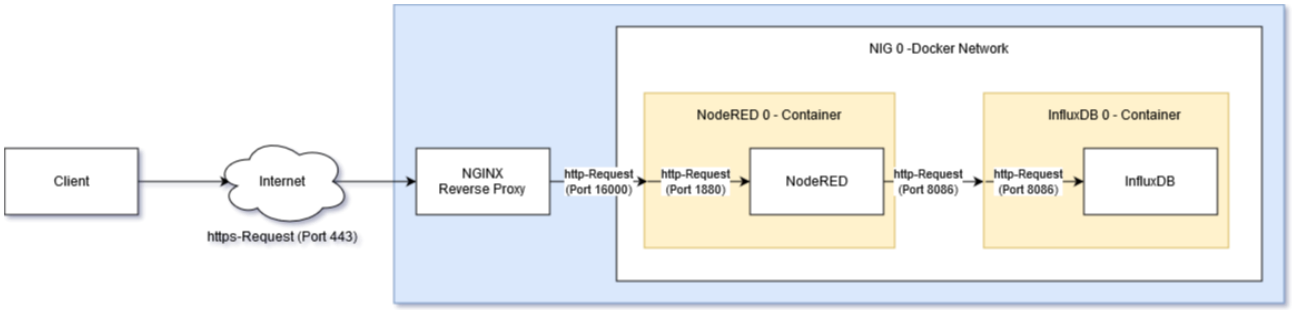


Abbildung 6 – Beispielhafter Informationsfluss

Der Client sendet über das Internet einen verschlüsselten https-Request, welchen der NGINX Reverse Proxy auf Port 443 empfängt, intern leitet er den Request auf den Port 16000 um, damit er mit dem Node-RED-Container kommuniziert. Der Port wird dann durch die Docker Engine in dem Container selbst auf 1880 umgeleitet. Ein solcher Request könnte bspw. verursachen, dass Node-RED etwas in der Datenbank abrufen möchte. Dafür sendet Node-RED einen Request in dem virtuellen NIG-Netzwerk auf Port 8086 des InfluxDB-Containers. Diese Anfrage wird dann ohne jegliche Portumleitung an InfluxDB übergeben.

4.2 Alternative Architektur

Neben der gewählten Architektur gab es noch mehrere alternative Implementierungen. Die einzigen festgelegten Komponenten waren Node-RED und Grafana. Dies lies Raum für verschiedene Möglichkeiten. Jedoch muss stets beachtet werden, dass von der Planung dieses Projektes bis hin zur Umsetzung und Nutzung nur sehr wenig Zeit lag. Es standen lediglich einige wenige Tage zur Verfügung, deswegen musste die simpelste und gleichzeitig eine gut funktionierende Lösung gewählt werden.

Neben InfluxDB gibt es noch weitere Zeitreihendatenbanken. Eine davon ist TimescaleDB. TimescaleDB basiert auf PostgreSQL, einer relationalen Datenbank (TimescaleDB, 2020). Der Unterschied zwischen den beiden verschiedenen Datenbanken hat sich als sehr gering herausgestellt, zumindest bezogen auf den hier beschriebenen Anwendungsbereich. Der größte Unterschied liegt für die Studierenden darin, dass InfluxDB eine SQL-ähnliche Sprache verwendet und TimescaleDB volle SQL-Kompatibilität bietet. Da jedoch die Datenbankabfragen in Grafana selbst über einzelne Blöcke zusammengestellt werden können, hat dieser Punkt weniger Auswirkungen auf die Entscheidung für InfluxDB gehabt. InfluxDB wurde letzten Endes gewählt, da sich nach einiger Recherche herausgestellt hat, dass der Verbund aus Node-RED, InfluxDB und Grafana bereits häufger schon

ohne große Probleme eingesetzt worden ist. Da die Sicherheit und schnelle Ausbringung des Systems wichtige Aspekte des Projektes waren, wurde sich letztendlich für InfluxDB entschieden.

Neben Docker gab es auch noch die Alternative, dass jede Gruppe eine eigene virtuelle Maschine erhält, auf welche sie über das Internet zugreifen kann. Diese Möglichkeit bietet die Vorteile, dass die Studierendengruppen deutlich mehr Freiheiten bei der Implementierung von neuen Technologien hätten und sie Softwareprobleme selbst beheben könnten. Die größten Probleme dieser Lösung lagen dafür bei Komplexität, Erreichbarkeit und Fehleranfälligkeit. Auch würde diese Lösung zu viele Hardware-Ressourcen verbrauchen, welche gar nicht zur Verfügung standen. Die Komplexität einer solchen Lösung würden zudem auch den Rahmen der Vorlesung übersteigen. Auch wäre es deutlich schwieriger die einzelnen Systeme wie Node-RED und Grafana per URL aus dem Internet erreichbar zu machen. Dies würde bspw. von Studierenden ein Grundwissen in Server-Administration verlangen. Weiterhin sind ganze virtuelle Betriebssysteme deutlich anfälliger für Fehler als einzelne Docker-Container.

4.3 Fazit

Nach Abwägung der verschiedenen Softwarearchitekturen ist die Lösung dann die in Kapitel 4.1 beschriebene Architektur geworden. Sie hat alle Anforderungen auf eine simple Weise erfüllt. Dies soll nicht implizieren, dass die gewählte Lösung perfekt und fehlerfrei ist, aber in Anbetracht der zur Verfügung stehenden Zeit und Ressourcen ist diese Lösung mehr als ausreichend.

5. Implementierung

5.1 Anforderung

Das NIG-System benötigt mehrere Komponenten, um zu funktionieren. Als Betriebssystem wird die neuste Version 20.04.1 LTS Version von Ubuntu verwendet. Diese Version bietet garantierte Updates bis April 2025. So kann das NIG-System über mehrere Jahre sicher betrieben werden. Für Docker und docker-compose reicht eine Standardinstallation vollkommen aus. Empfohlen wird Docker-Version 19.03.9 und docker-compose 1.25.5. NGINX muss bereits als Reverse Proxy konfiguriert und installiert sein. Auch ist es wichtig, dass bereits ein SSL-Zertifikat für die NIG-URL zur Verfügung steht. Für Teile des NIG-Systems wird zudem noch die JavaSkript-Runtime Node (Version 12.18.0) und der Paketmanager npm (Version 6.14.4) benötigt. Mit Hilfe des extra zu installierenden Moduls bcrypt können so automatisch verschlüsselte Passwörter für NodeRED generiert werden.

Je nach Anzahl und Umfang der einzelnen NIG-Systeme wird unterschiedliche Menge an Hardware-Ressourcen benötigt. Sollten diese Systeme eher sporadisch und nicht im Dauereinsatz benutzt werden reicht eine einfache Server-Umgebung aus. Dabei ist die Menge an RAM (Empfohlen: 16GiB) und Datenspeicher (Empfohlen: 100 GiB) wichtiger als die Anzahl der CPU-Kerne (Empfohlen: 4 Kerne / 4 Threads). Die genannten Anforderungen ermöglichen es den Server neben dem NIG-System auch noch für weitere Anwendungen zu betreiben. Bspw. ein Wiki für die Studierenden.

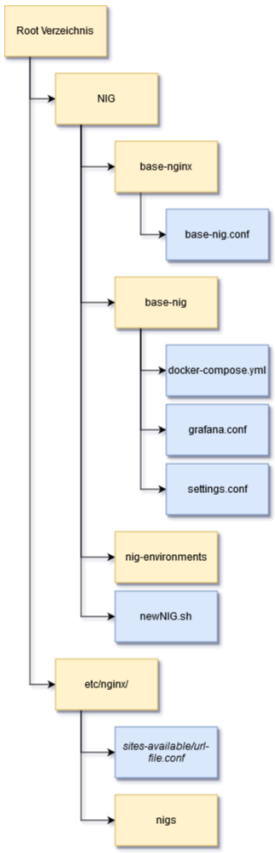


Abbildung 7 – Ordner- und Dateistruktur

5.2 Grundstruktur

Alle NIG-Dateien sollten in einem Ordner namens „NIG“ im Home-Verzeichnis eines Root-Users liegen. In diesem Ordner befinden sich drei Unterordner sowie ein Skript zum Anlegen eines neuen NIG-Systems. Der Ordner „base-nginx“ enthält eine Basis-Konfiguration für ein NIG-System, welche von NGINX später geladen wird. Im Ordner „base-nig“ befinden sich die Basis-Konfigurationsdateien für die einzelnen Container. Die „docker-compose.yml“ beschreibt die Container, welche gestartet werden sollen. Die „grafana.conf“ konfiguriert Grafana für den Einsatz hinter einem Reverse Proxy und legt die Zugangsdaten fest und die „settings.conf“ führt dasselbe für Node-RED durch. Im letzten Ordner „nig-environments“ werden alle aktiven NIG-Systeme abgelegt inklusive aller persistenten Daten. Auch die einzelnen Konfigurationsdateien der Systeme sind hier zu finden. Zu Beginn ist dieser Ordner leer. Die Datei „newNIG.sh“ wird verwendet, um automatisch ein neues System hinzuzufügen.

Weitere Dateien befinden sich im Standard-Ordner von NGINX. Hier muss ein Ordner mit dem Namen „nigs“ angelegt werden. In diesem werden die Konfigurationsdateien für die einzelnen URLs zu den NIG-Systemen festgelegt. Auch muss die URL-Konfigurationsdatei im „sites-available“ Ordner angepasst werden.

5.3 Das Skript

Das Skript dient zur automatischen Erstellung und Aktivierung einzelner NIG-Systeme. Dabei nimmt das Skript die Basis-Konfigurationsvorlagen (siehe Kapitel 5.2 Grundstruktur („base-nig“ und base-nginx)) und ersetzt vorgefertigte Marker[7] mit den eigentlichen Variablen. Bspw. enthält die Datei

[7] Ein ähnliches Konzept wie auch das Front-End-Framework Angular anwendet. So können in Angular JavaSkript-Variablen im HTML-Code verwendet werden.

„docker-compose.yml“ in base-nig den Marker „{{ROOT}}“. Dieser wird vom Skript dann zu dem gewählten ROOT-Namen[8] geändert.

Das Skript lässt sich in sechs Teilbereiche zusammenfassen. Im ersten Teil werden alle wichtigen Variablen definiert. Dazu zu zählen die zu verwendende Domain, der Verzeichnispfad in der URL, sowie der der persistenten Daten und die Nutzernamen, Passwörter und Ports von Node-RED und Grafana. Für InfluxDB wird immer dasselbe Passwort verwendet, da die Instanzen aus dem Internet nicht erreichbar sind. Der zweite Teil verarbeitet das angegebene Node-RED Passwort und verschlüsselt es mit Hilfe des npm-Packages „bcryptjs“. Dies ist nötig da sonst das Passwort unverschlüsselt von Node-RED gespeichert wird. Danach werden in einem dritten Schritt alle Basis-Dateien in einen neuen Root-Ordner im Verzeichnis „nig-environments“ kopiert. Der Name ist dabei die angegebene „ROOT“-Variable. Anschließend werden sowohl die Marker in der „docker-compose.yml“ und der „grafana.conf“ geändert. Dabei handelt es sich um die oben vordefinierten Variablen. Im fünften Schritt werden die NGINX Basis-Dateien kopiert und verändert, sodass die gewählte URL auf die richtigen Ports der Container verweist. Der letzte Schritt ändert noch die Zugriffsrechte auf die persistenten Dateien und startet das NIG-System anschließend.

5.4 NGINX

NGINX dient hier als Integration Reverse Proxy, damit alle Container aus dem Internet erreichbar sind. Für die NIG-Systeme wird eine Funktion der NGINX-Konfiguration benutzt, die ermöglicht Konfigurations-Snippets[9] aus anderen Dateien in eine andere Konfigurationsdatei einzufügen. Diese heißt “include”. Eine so referenzierte Datei wird eins zu eins übernommen. In der Hauptkonfigurationsdatei von NGINX werden alle Konfigurationsdateien aus dem Ordner „sites-available“ inkludiert. In diesem Ordner liegen üblicherweise die Konfigurationsdateien für einzelne URLs. In der Konfigurationsdatei für die gewählte URL werden die einzelnen Endpunkte des Reverse Proxy definiert. Diese heißen dann “server”. Für das NIG-System wird lediglich ein Server benötigt, welcher nur für die NIG-Domain verwendet wird. Da alle einzelnen NIG-Systeme unter einer Domain funktionieren und erreichbar sein sollen wird auch nur ein SSL Zertifikat benötigt. Folgend werden alle Dateien per Wildcard[10]-Zugriff aus dem Ordner „nigs“ importiert.

In diesem Ordner liegen alle Konfigurationsdateien der einzelnen NIG-Systeme. Diese enthalten sogenannten „locations“. Eine „location“ ist in diesem Fall eine Instanz von Node-RED oder Grafana. Für jedes einzelne NIG-System gibt es zwei verschiedene URLs. Node-RED ist dann bspw. unter „nig.example.com/groupA/node-red“ erreichbar. „groupA“ ist in diesem Fall die ROOT-Variable,

[8] Die Variable ROOT
[9] Kleine Code und Konfigurationsteile
[10] Ein Platzhalter für alle Zeichen

welche im Skript angegeben wurde. Für Grafana muss in der URL lediglich „node-red“ durch „grafana“ ersetzt werden. So lassen sich alle NIG-Systeme einzeln und unkompliziert erreichen.

5.5 Docker / Docker-Compose

Docker im Zusammenspiel mit docker-compose wird zur Orchestrierung, also der Verknüpfung einzelner Container verwendet. docker-compose ermöglicht es in diesem Fall Container mit Hilfe einer Konfigurationsdatei zu starten. Diese ist wie folgt aufgebaut:

Alle Container eines einzelnen Systems sind in einem eigenen separaten Netzwerk, welches im Namen zum eindeutigen Referenzieren den ROOT-Namen des Systems enthält. In diesem Netzwerk sind drei Container: Node-RED, InfluxDB und Grafana. Alle erhalten als Namen den eigentlichen Namen des Dienstes sowie den ROOT-Namen des Systems. Dies ermöglicht auch einzelne Container später zu verwalten. Allerdings bedeutet dies auch, dass der ROOT-Name eindeutig sein muss. Für Node-RED wird noch eine Zeitzone und eine Verknüpfung zu den vorher erstellten und kopierten Konfigurationsdateien und dem persistenten Datenspeicherort definiert. Die weiteren Umgebungsvariablen[11] wurden durch das Skript festgelegt. Diese enthalten die zu nutzende URL, sowie den Nutzernamen und das Kennwort für einen Studierenden, um sich einzuloggen. Auch muss für Node-RED festgelegt werden mit welchem Port es für Komponenten außerhalb des Netzwerks erreichbar ist. Dies wurde auch im Skript festgelegt. Für InfluxDB wird lediglich der persistente Speicherort definiert sowie drei feste Umgebungsvariablen. Jede InfluxDB-Instanz erhält als Datenbanksname, -nutzernamen und -kennwort dieselben Variablen. Diese können in Node-RED und Grafana zum Datenzugriff verwendet werden. Einen Port für die Kommunikation außerhalb des Netzwerks ist nicht nötig. Die Konfiguration des Grafana-Containers ähnelt sehr der von Node-RED. Auch hier werden die Verlinkungen zu den persistenten Datenspeicherorten angegeben, sowie die im Skript festgelegten Werte für den zu verwendenden Port außerhalb des Netzwerks und die Login-Daten für Grafana.

Mit dieser Konfigurationsdatei ist es möglich ein System zu starten, zu beenden oder auch neu zu starten. Auch kann die Konfiguration auf andere Server verlagert oder mit diesen geteilt werden. Eine weitere Möglichkeit besteht auch darin im Nachhinein Details der Konfiguration zu ändern bzw. zu evaluieren.

[11] Umgebungsvariablen sind Variablen, welche in diesem Fall lediglich in einem Container abrufbar sind. Ein Programm kann diese zur Laufzeit auslesen. Node-RED verwendet dies bspw. für die Login-Daten.

6. Anwendung

6.1 Das Skript benutzen

Möchte der Professor nun für eine Studierendengruppe ein neues NIG-System hinzufügen, dann benutzt er dafür das vorgefertigte Skript. Dieses funktioniert wie in Kapitel 5.3 beschrieben. Vor der ersten Benutzung muss sichergestellt sein, dass alles richtig eingerichtet und konfiguriert wurde. Dann müssen im Skript alle notwendigen Variablen verändert werden. Dabei muss darauf geachtet werden, dass sich sowohl die festgelegten Ports als auch der ROOT-Name mit keinem anderen System überschneiden darf. Ansonsten funktioniert die automatische Einrichtung nicht. Sollten mehrere NIG-Systeme nacheinander bereitgestellt werden empfiehlt es sich zur Planung eine Tabelle anzulegen in der alle URLs (ROOT-Namen), Ports und Login-Daten geplant und dokumentiert werden. Dies hilft auch zu einem späteren Zeitpunkt falls etwas an Systemen geändert werden muss. Sind alle Variablen im Skript geändert, kann der Professor das Skript über die Kommandozeile ausführen und nach kurzer Zeit ist das NIG-System für die Studierenden verfügbar.

6.2 „Schulung“

Ein weiterer wichtiger Aspekt der Anwendung ist neben dem reinen Aufsetzen und Generieren der Systeme, die Erklärung für die Studierenden wie sie das System benutzen können und welche Möglichkeiten es bietet. Dabei müssen die einzelnen URLs, sowie die Login-Daten an die jeweiligen Studierendengruppen verteilt werden. Auch kann es für die Studierenden interessant sein wie das System aufgebaut ist und funktioniert.

Um den Studierenden einen guten Einstieg in das System zu geben müssen auch die grundlegenden Konzepte und Funktionsweisen der einzelnen Komponenten erklärt werden. Besonders gut funktioniert dies an Hand eines einfachen Beispiels. So empfiehlt es sich bspw. mit einem ESP die Raumtemperatur zu messen und an Hand dessen zu zeigen wie man die Daten an Node-RED schickt, dann dort verarbeitet, weiter an die Datenbank gibt, um sie zu speichern und abschließend in Grafana anzuzeigen. Auch wenn sich das zunächst trivial anhört kann es für Studierende trotzdem zunächst kompliziert wirken, im Besonderen, wenn sie weniger IT-Erfahrung besitzen. Hilfreich ist es weitere Quellen für gute Tutorials anzugeben. Diese finden sich bspw. im Falle von Node-RED und Grafana direkt auf der Projekt-Seite.

Auch wenn die „Schulung“ nicht direkter Bestandteil des IT-Systems ist, ist es wichtig diese durchzuführen. Ansonsten würde das System mit größter Wahrscheinlichkeit nicht genutzt werden.

7. Ergebnisse und Ausblick

7.1 Ergebnis

Das Ergebnis des NIG-Projektes ist ein solides System, welches schnell auch auf weiteren Servern und in anderen Semestern weiterverwendet werden kann. Dabei erfüllt es alle funktionalen und nicht-funktionalen Anforderungen, welche gestellt wurden. Lediglich das Szenario „Ein Kurs endet“ (Kapitel 3.1.2) wurde zumindest nicht automatisiert (bspw. durch ein Skript) implementiert, aber es kann immer noch manuell von dem Professor oder Systemadministrator durchgeführt werden.

Die Lösung stellte sich im Test im Kurs Applied Measurement and Control als äußert praktikabel und mit zu nächst keinen Problemen heraus. Die verwendeten Technologien haben zum reibungslosen Ablauf beigetragen. Auch lässt sich das System auf weitere Tools und Systeme übertragen und so weiter nutzen.

Ein Makel des Systems ist jedoch das Skript (Kapitel 5.3). Dieses funktioniert zwar muss aber mit Bedacht ausgeführt werden, da es keine Überprüfung auf bspw. überschneidende ROOT-Namen bietet.

7.2 Ausblick

Das Projekt bietet viele Möglichkeiten weiterentwickelt zu werden. Momentan können neue NIG-System nur über das Skript (Kapitel 5.3) hinzugefügt werden und immer nur ein System auf einmal. Eine mögliche Funktion in Zukunft könnte es sein, mehrere Systeme gleichzeitig zu generieren und die Daten, also die notwendigen Variablen, aus einer CSV-Datei zu beziehen. Noch einfacher für den Endanwender würde es werden, wenn sogar eine einfache GUI zum Verwalten aller Systeme zur Verfügung stehen würde. Darüber könnten dann alle NIG-Systeme einzeln aufgerufen werden und dann bspw. neugestartet, gelöscht oder ein Backup erstellt werden. Bis dahin könnte das momentan bestehende Skript auch erweitert werden, sodass das Skript mit Kommandozeilenparametern benutzt werden kann. Dann müsste der Anwender nicht immer die Variablen in der Datei selbst ändern. Besonders einfach zu benutzen wäre das gesamte NIG-System, wenn es selbst inklusive eines NGINX-Reverse-Proxys innerhalb eines Docker-Containers wäre. Sollte dann auch eine GUI vorhanden sein, kann das System innerhalb weniger Minuten einsatzbereit sein.

7.3 Zusammenfassung

Zusammenfassend lässt sich sagen, dass die gewählte Lösung in Anbetracht des Entwicklungsaufwandes sehr gut funktioniert und sowohl von Lehrenden als auch den Studierenden gut angenommen wurde. Dabei bietet es noch eine Vielzahl an Möglichkeiten und Erweiterungen für zukünftige Semester.

Literatur- und Quellenverzeichnis

Blackstock, Michael und Lea, Rodger (2016) FRED: A Hosted Data Flow Platform for the IoT Built Using Node-RED, 1st Workshop on the Mashup of things and APIs (MOTA), Trento: ACM.

(o.J.) Grafana Features, Verfügbar unter: <https://grafana.com/grafana/> [aufgerufen am 14. September 2020].

Holler, Jan et. Al. (2014) From Machine-to-Machine to the Internet of Things, Oxford: Elsevir.

Kurniawan, Agus (2019) Internet of Things Projects with ESP32, Birmingham: Packt Publishing.

Naqvi, Syeda Noor Zehra und Yfantidou, Sofia (2017) Time Series Databases and InfluxDB, Advanced Database at Univerité Libre De Bruxelles, Brüssel: Université Libre de Bruxelles.

Rhine-Waal University of Applied Sciences – Faculty of Communication and Environments (2018) Handbook of Modules for the Degree Programme Environments and Energy, B.Sc., Version 1.32, Hochschule Rhein-Waal.

Smith, Randall (2017) Docker Orchestration, Birmingham: Packt Publishing

Sommerlad, Peter (2003) Reverse Proxy Patterns, Conference: Proceedings of the 8th European Conference on Pattern Languages of Programms (EuroPLoP '2003), Irsee: Peter Sommerlad

(o.J) TimescaleDB, Verfügbar unter: <https://www.timescale.com/products> [aufgerufen am 14. September 2020].

Turnball, James (2019) The Docker Book, Version 18.09.2, Verfügbar unter: <https://books.google.de/books?id=4xQKBAAAQBAJ&lpg=PA1&> [aufgerufen am 14. September 2020].

Vermesan, Ovidiu und Friess, Peter (2013) Internet of Things, Aalborg: River Publishers.

(o.J.) What is a Container?, Verfübar unter: <https://www.docker.com/resources/what-container> [aufgerufen am 14. September 2020].

(o.J.) What is a Time Series Database?, Verfügbar unter: < https://www.influxdata.com/time-series-database/> [aufgerufen am 14. September 2020].

Quellcode

Der Quellcode und eine technische Dokumentation zum Thema Installation lässt sich hier finden:

https://wiki.eolab.de/doku.php?id=user:jan001:nigdocu:aufbau

Hiermit erkläre ich, Jan Sonntag, dass ich die hier vorliegende Arbeit selbst-ständig und ohne unerlaubte Hilfsmittel angefertigt habe. Informationen, die anderen Werken oder Quellen dem Wortlaut oder dem Sinn nach entnommen sind, habe ich kenntlich gemacht und mit exakter Quellenangabe versehen. Sätze oder Satzteile, die wörtlich übernommen wurden, wurden als Zitate gekennzeichnet. Die hier vorliegende Arbeit wurde noch an keiner anderen Stelle zur Prüfung vorgelegt und weder ganz noch in Auszügen veröffentlicht. Bis zur Veröffentlichung der Ergebnisse durch den Prüfungsausschuss werde ich eine Kopie dieser Studienarbeit aufbewahren und wenn nötig zugänglich machen.

Goch, 14.09.2020
Jan Sonntag