Projektdokumentation

LB-Arbeit Modul 347

Daniel Kovac und Kristian Lubina

Inhalt

Einleitung

Ziel der LB-Arbeit

Die LB des Moduls 347 umfasst ein Projekt zur Einrichtung mehrerer Application Stacks in einer Cloud-Umgebung. Am Ende des Projekts sollen die folgenden Produkte bereitstehen:

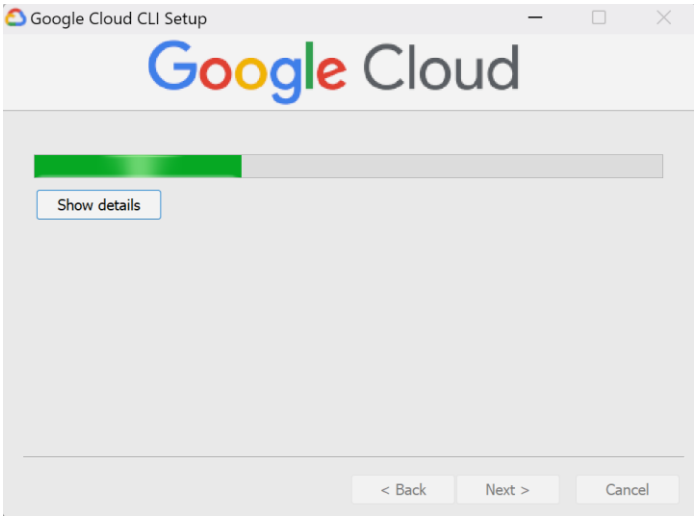
- Konfigurationsfiles für die Integration einer komplett lauffähigen
  - Wordpress
  - MediaWiki
  - Jira

Ziel dieser Dokumentation

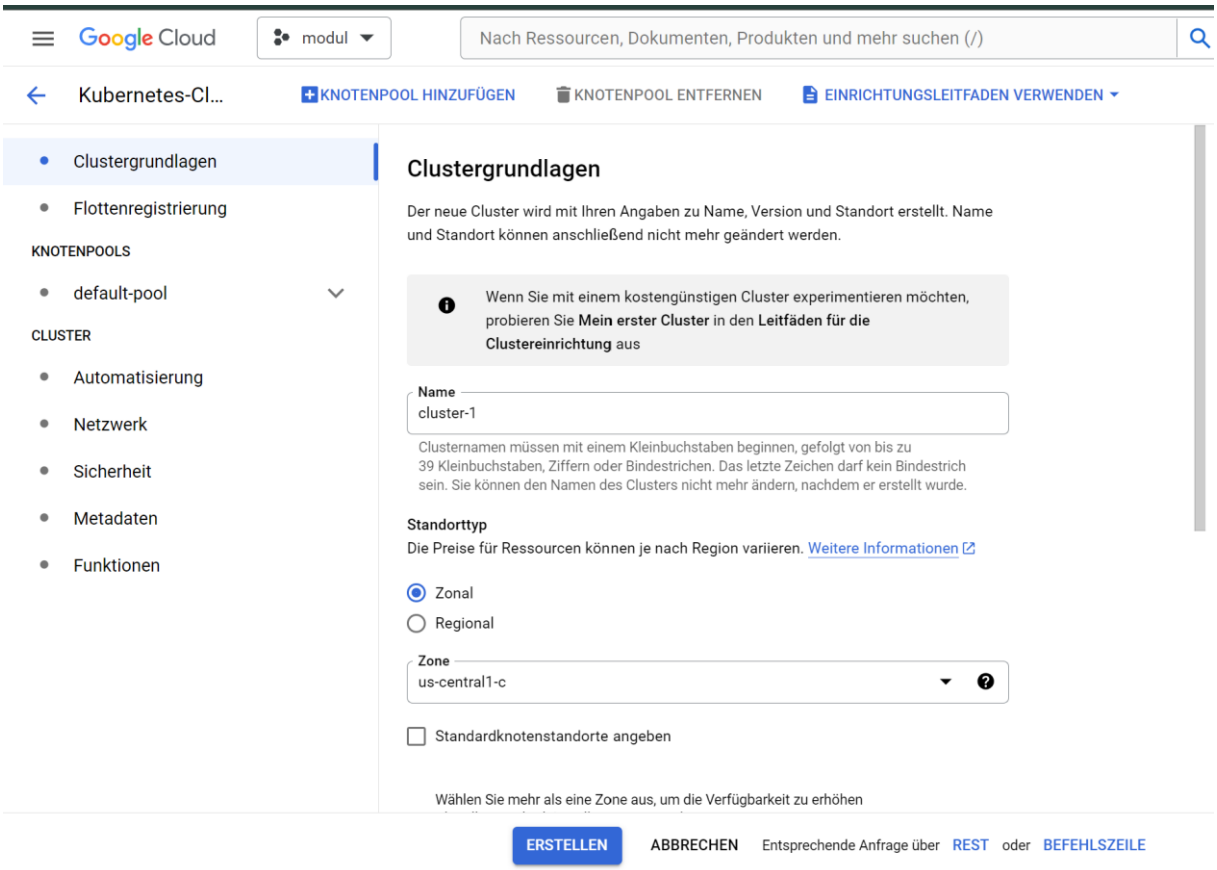
Die Projektdokumentation dient zunächst als Starthilfe für die Entwicklung und später zur Dokumentation des Fortschritts. Wenn Änderungen am System vorgenommen werden, sollten diese auch in die Dokumentation aufgenommen werden. Die Dokumentation sollte stets den aktuellen Zustand widerspiegeln.

Cloud einrichten

Zuerst haben wir uns über die Cloud Gedanken gemacht und uns vorgestellt, wie alles aussehen sollte. Am Anfang hatten wir Schwierigkeiten, eine Cloud einzurichten. Kristian hatte keine gültige Karte und konnte sie nicht bald erneuern, weil er keine Zeit hatte, da wir gerade danach eine Blockwoche hatten. Daniel konnte sich auch nicht bei DigitalOcean registrieren da DigitalOcean spinnte und seine Daten nicht annahm.



Auf Google Cloud konnten wir uns endlich einmal regisitieren was wir bei den anderen Anbietern nicht konnten. Endlich konnten wir mal anfangen mit der Arbeit loszulegen und unseren Kubernetes-Cluster anzulegen.



Beim Aufbau unserer Infrastruktur haben wir zunächst einen Kubernetes-Cluster angelegt. Dabei wurden häufig Standardkonfigurationen verwendet. Für die meisten unserer Cluster habe ich den Namen "modul 347" und die Zone "europe-central" gewählt. Dieser Prozess des Erstellens und Löschens von Clustern wurde mehrmals wiederholt, wobei stets die gleichen Standardeinstellungen verwendet wurden. Diese routinemäßige Auswahl führte allerdings später zu Problemen, da es zu hohen CPU-Auslastungen kam, die unsere Kapazitäten regelmäßig überstiegen.

Wordpress Microservice

Einrichtung

Wir haben einige Herausforderungen gehabt, als wir unseren WordPress-Microservice in der Google Cloud einrichteten. Zunächst hatten wir Schwierigkeiten mit den YAML-Dateien und konnten die Pods nicht richtig konfigurieren, was zu ständigen "Pending"-Statusmeldungen führte.

```
C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get pods
NAME                              READY   STATUS     RESTARTS   AGE
mein-wordpress-5c68fdf764-8gmfd   0/1     Init:0/1   0          5m12s
mein-wordpress-mariadb-0          0/1     Running    0          5m12s

C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get services
NAME                    TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)                      AGE
kubernetes              ClusterIP      34.118.224.1     <none>           443/TCP                      71m
mein-wordpress          LoadBalancer   34.118.236.83    35.188.135.154   80:31449/TCP,443:31811/TCP   5m22s
mein-wordpress-mariadb  ClusterIP      34.118.226.115   <none>           3306/TCP                     5m22s
```

Etliche male haben wir den Cluster neu erstellen und verschiedene Ansätze ausprobieren. Die YAML-fies die wir erstellt haben funktionierten leider nicht und wir ware uns nicht sicher wieso.

```
C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>gcloud container clusters get-credentials modul347 --region europe-central2 --project soni
c-proxy-421310
Fetching cluster endpoint and auth data.
kubeconfig entry generated for modul347.

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl cluster-info
Kubernetes control plane is running at https://34.118.17.217
GLBCDefaultBackend is running at https://34.118.17.217/api/v1/namespaces/kube-system/services/default-http-backend:http/proxy
KubeDNS is running at https://34.118.17.217/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://34.118.17.217/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl apply -f mysql-secret.yaml
Error from server (BadRequest): error when creating "mysql-secret.yaml": Secret in version "v1" cannot be handled as a Secret: illegal base64 data at input
byte 0

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl apply -f mysql-pvc.yaml
persistentvolumeclaim/mysql-pvc created

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl apply -f mysql-deployment.yaml
deployment.apps/mysql created
service/mysql created

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl apply -f wordpress-pvc.yaml
persistentvolumeclaim/wordpress-pvc created

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl apply -f wordpress-deployment.yaml
deployment.apps/wordpress created
service/wordpress created
```

Wir erhielten dasselbe Ergebnis wie zu Beginn und drehten uns im Kreis.

```
C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl get pods
NAME                         READY   STATUS    RESTARTS   AGE
mysql-86bd756d9-4pcm6        0/1     Pending   0          7m19s
wordpress-865f49c944-c4d4k   0/1     Pending   0          7m19s

C:\Users\Blasmiren\OneDrive\Desktop\modul347\wordpress\kubernetes>kubectl get services
NAME         TYPE           CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
kubernetes   ClusterIP      34.118.224.1    <none>          443/TCP        85m
mysql        ClusterIP      None            <none>          3306/TCP       7m31s
wordpress    LoadBalancer   34.118.239.58   34.118.86.253   80:31228/TCP   7m29s
```

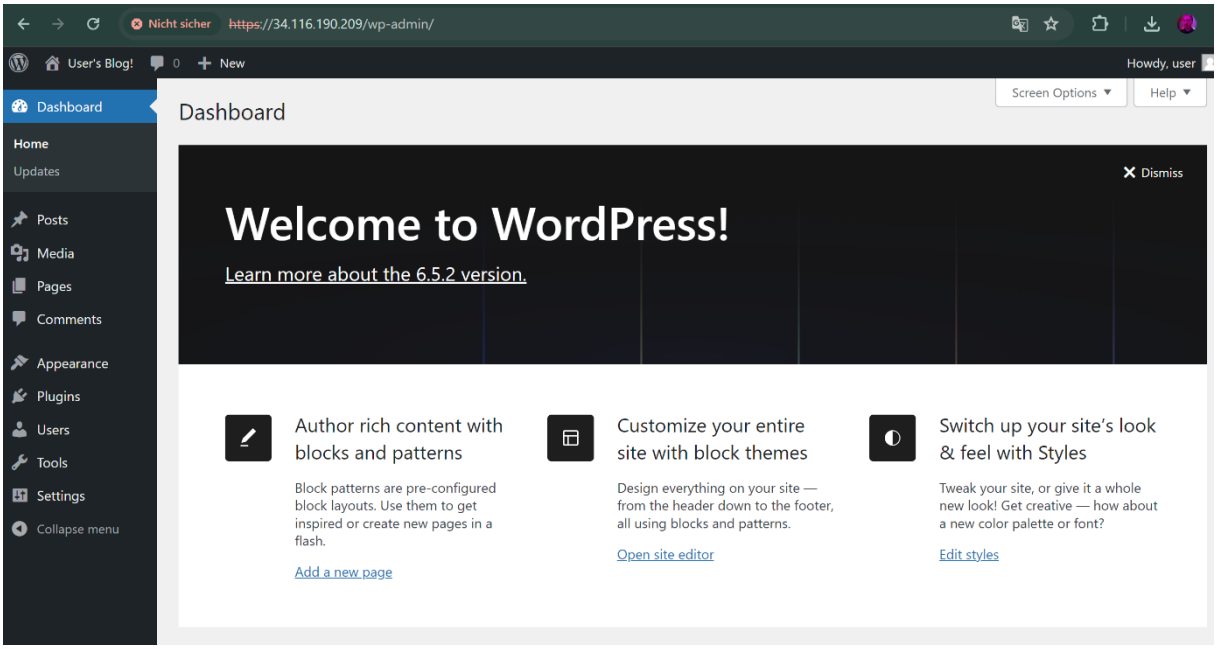
Letztendlich setzten wir Helm ein, um die Installation zu erleichtern. Schliesslich gelang es uns, die Dienste zum Laufen zu bringen, wobei wir kontinuierlich den Status der Pods überprüften.

```
C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get pods --namespace wordpress
NAME                            READY   STATUS    RESTARTS   AGE
my-wordpress-67bdcfb65d-2whh5   1/1     Running   0          6m3s
my-wordpress-mariadb-0          1/1     Running   0          6m2s

C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get svc --namespace wordpress
NAME                 TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)                      AGE
my-wordpress         LoadBalancer   10.81.202.135   34.116.190.209   80:32412/TCP,443:30206/TCP   6m17s
my-wordpress-mariadb   ClusterIP      10.81.196.59    <none>           3306/TCP                     6m17s

C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get secret --namespace wordpress my-wordpress -o jsonpath="
{.data.wordpress-password}" | base64 --decode
Der Befehl "base64" ist entweder falsch geschrieben oder
konnte nicht gefunden werden.

C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>
```

Nachdem wir das Passwort entziffert und die IP-Adresse herausgefunden haben, können wir dann endlich auf die URL zugreifen.

```
C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get secret --namespace wordpress my-wordpress -o jsonpath="
{.data.wordpress-password}"
Z25EMlNlQ1RMSg==
C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>
```

Nachdem wir das Passwort entziffert und die IP-Adresse herausgefunden haben, konnten wir erfolgreich auf die URL zugreifen. WordPress läuft nun reibungslos. Die Anwendung von Helm hat uns eine Menge geholfen.



Testkonzept und Testprotokoll

Testfall-Nummer	Ziel	Schritte	Erwartetes Ergebnis
1	Zugriff auf die Hauptseite	Zuerst öffne ich einen Webbrowser. Dann gebe ich die externe IP-Adresse oder den Domain-Namen unseres MediaWiki-Servers ein.	Die Hauptseite von MediaWiki wird erfolgreich geladen.
2	Anmeldung als Administrator	Ich klicke auf der Hauptseite auf „Anmelden“ und gebe dann unsere Administrator-Zugangsdaten ein.	Ich bin erfolgreich angemeldet und sehe das Admin-Dashboard.
3	Erstellung einer neuen Seite	Nach dem Anmelden wähle ich „Seite erstellen“, trage einen Titel und etwas Inhalt ein und speichere die Seite.	Die neue Seite wird erstellt und zeigt den eingegebenen Inhalt an.
4	Bearbeitung einer bestehenden Seite	Ich navigiere zu einer bestehenden Seite, klicke auf „Bearbeiten“, füge zusätzlichen Text hinzu oder ändere den bestehenden und speichere die Änderungen.	Die Seite wird aktualisiert und zeigt den geänderten Inhalt korrekt an.

JIRA-Microservice

Einrichtung

Nun hat auch Kristian endlich eine Cloud richtig eingerichtet und konnte effektiver an diesem Projekt mitarbeiten. Er entschied sich für Microsoft Azure, was für Schüler, die GitHub Pro nutzen, den Vorteil bietet, keine Kreditkartendaten zu verlangen. Die anfängliche Einrichtung war allerdings alles andere als einfach.

Anschließend haben wir kubectl apply verwendet, um die benötigten Ressourcen zu erstellen

- kubectl apply -f jira-postgres-volume-claim.yaml

```
kristian [ ~ ]$ kubectl apply -f jira-postgres-volume-claim.yaml
persistentvolumeclaim/jira-postgres-volume-claim created
persistentvolumeclaim/jira-volume-claim created
```

- kubectl apply -f jira-deployment.yaml

```
kristian [ ~ ]$ kubectl apply -f jira-deployment.yaml
deployment.apps/jira-deployment created
service/jira-service created
```

Die Dienste im Namespace jira starteten nicht korrekt, was die Fehlersuche nervenaufreibend machte. kubectl describe pod zeigte uns, dass wichtige Konfigurationsdateien fehlten oder fehlerhaft waren.

```
kristian [ ~ ]$ kubectl get pods -n jira
NAME                        READY   STATUS              RESTARTS
jira-7574f9d647-c47s2       0/1     ContainerCreating   0
postgres-69f8ccf56f-88fhz   1/1     Running             0
kristian [ ~ ]$ kubectl describe pod jira-7574f9d647-c47s2 -n jira
Name:             jira-7574f9d647-c47s2
Namespace:        jira
Priority:         0
Service Account:  default
Node:             aks-nodepool1-95361945-vmss000000/10.224.0.4
Start Time:       Fri, 03 May 2024 17:30:57 +0000
Labels:           app=jira
                  pod-template-hash=7574f9d647
Annotations:      <none>
Status:           Running
IP:               10.244.0.12
IPs:
  IP:           10.244.0.12
Controlled By:  ReplicaSet/jira-7574f9d647
```

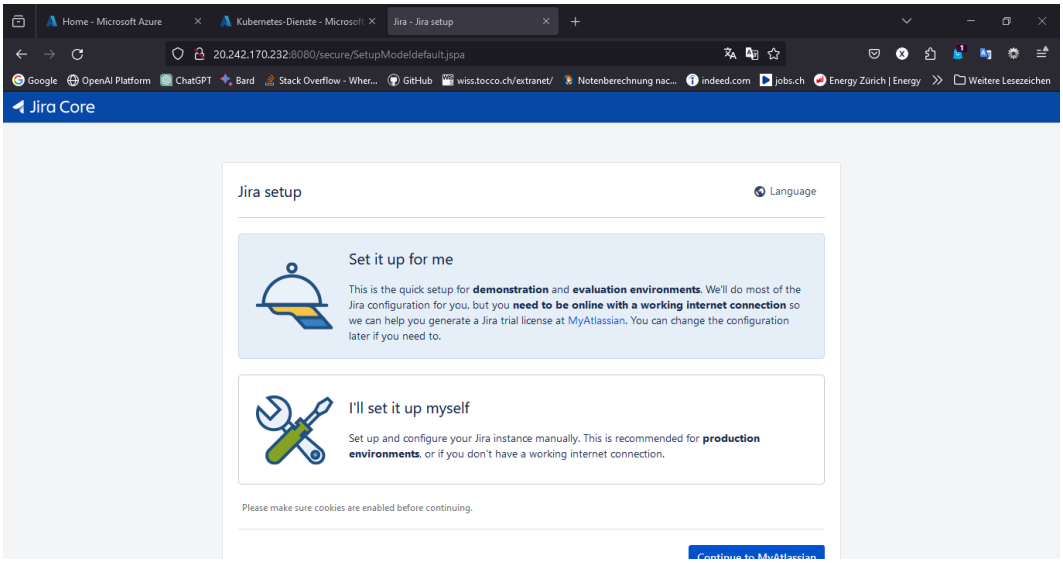
Um den Status des Jira-Pods genauer zu verstehen, haben wir die Logs mit kubectl logs überprüft. Die Logs zeigten, dass verschiedene Konfigurationsdateien generiert wurden und Jira erfolgreich gestartet wurde und dass doch alles funktioniert.

```
kristian [ ~ ]$ kubectl logs jira-7574f9d647-c47s2 -n jira
INFO:root:Generating /etc/container_id from template container_id.j2
INFO:root:Generating /opt/atlassian/jira/conf/server.xml from template server.xml.j2
INFO:root:Generating /opt/atlassian/jira/atlassian-jira/WEB-INF/classes/seraph-config.xml from template seraph-config.xml.j2
INFO:root:Generating /var/atlassian/application-data/jira/dbconfig.xml from template dbconfig.xml.j2
INFO:root:User is currently root. Will downgrade run user to jira
WARNING:root:Unsetting environment var ATL_DB_PASSWORD
INFO:root:Running Jira with command '/opt/atlassian/jira/bin/start-jira.sh', arguments ['/opt/atlassian/jira/bin/start-jira.sh', '-fg']
executing as current user
```

Um sicherzustellen, dass die externen IP-Adressen korrekt konfiguriert sind, haben wir die Services mit kubectl get svc überprüft. Auch die Nodes haben wir mit kubectl get nodes geprüft, um den Status der Kubernetes-Umgebung zu überprüfen.

```
kristian [ ~ ]$ kubectl get svc -n jira
NAME       TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
jira       NodePort    10.0.30.68   <none>        8080:30673/TCP   2m58s
postgres   ClusterIP   None         <none>        5432/TCP         3m15s
kristian [ ~ ]$ kubectl get nodes -o wide
NAME                                STATUS   ROLES   AGE   VERSION   INTERNAL-IP   EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTAINER-RUNTIME
aks-nodepool1-95361945-vmss000000   Ready    agent   14m   v1.28.5   10.224.0.4    <none>        Ubuntu 22.04.4 LTS   5.15.0-1060-azure   containerd://1.7.15-1
kristian [ ~ ]$ kubectl edit svc jira -n jira
service/jira edited
kristian [ ~ ]$ kubectl get svc -n jira
NAME       TYPE           CLUSTER-IP   EXTERNAL-IP      PORT(S)          AGE
jira       LoadBalancer   10.0.30.68   20.242.170.232   8080:30673/TCP   9m8s
postgres   ClusterIP      None         <none>           5432/TCP         9m25s
```

Abschliessend haben wir die Jira-Setup-Seite aufgerufen, um zu überprüfen, ob der Dienst ordnungsgemäss funktioniert. Was es auch tat, obwohl mehrere Probleme und Hindernissen überwindet haben, um überhaupt so weit zu kommen.



Testkonzept und Testprotokoll

Testfall	Ziel	Schritte	Erwartetes Ergebnis
1	Überprüfen, ob die Jira-Weboberfläche zugänglich ist,	Den Webbrowser öffnen. Dann gebe ich die externe IP-Adresse des Jira-Services gefolgt von :8080 ein. Ich sehe die Jira-Anmeldeseite.	Die Anmeldeseite von Jira ist korrekt angezeigt worden.
2	Überprüfen, ob sich Benutzer erfolgreich in Jira anmelden können.	Ich folge den Schritten aus Testfall 1, um die Anmeldeseite zu öffnen. Ich gebe die Anmeldeinformationen	Die Anmeldung ist erfolgreich.

		eines vorhandenen Benutzers ein. Ich überprüfe, ob die Anmeldung erfolgreich ist und die Benutzeroberfläche geladen wird.	
3	Überprüfen, ob Benutzer in der Lage sind, ein neues Projekt zu erstellen.	Nachdem ich mich in Jira angemeldet habe (siehe Testfall 2), klicke ich auf "Projekte" und wähle "Projekt erstellen". Ich gebe die erforderlichen Informationen ein und klicke auf "Erstellen". Dann überprüfe ich, ob das Projekt erfolgreich erstellt wurde.	Das neue Projekt sollte erfolgreich erstellt und in der Projektliste angezeigt werden.
4	Überprüfen, ob Benutzer in der Lage sind, ein neues Ticket zu erstellen.	Nachdem ich mich in Jira angemeldet habe (siehe Testfall 2), öffne ich ein Projekt und klicke auf "Ticket erstellen". Ich fülle die Felder für das neue Ticket aus und klicke auf "Erstellen". Dann überprüfe ich, ob das Ticket erfolgreich erstellt wurde.	Das neue Ticket sollte erfolgreich im Projekt angezeigt werden.

MediaWiki Microservice

Einrichtung

Wir versuchen gerade, MediaWiki als Microservice einzurichten, ähnlich wie wir es zuvor mit WordPress gemacht haben. Der Prozess beginnt vielversprechend. Wir konfigurieren den Dienst, setzen die Einstellungen und optimieren die Parameter.

```
C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get svc --namespace mediawiki
NAME                 TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)                      AGE
my-mediawiki         LoadBalancer   10.81.201.155   34.116.137.254   80:31297/TCP,443:31020/TCP   79s
my-mediawiki-mariadb ClusterIP      10.81.199.140   <none>           3306/TCP                     79s

C:\Users\Daniel Kovac\AppData\Local\Google\Cloud SDK>kubectl get pods --namespace mediawiki
NAME                   READY   STATUS    RESTARTS   AGE
my-mediawiki-mariadb-0 1/1     Running   0          87s
```

```
PS C:\Users\Daniel Kovac> kubectl get secret --namespace mediawiki my-mediawiki -o jsonpath="{.data.mediawiki-password}"
 | ForEach-Object { [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String($_)) }
0UMmAhAWJH
PS C:\Users\Daniel Kovac>
```

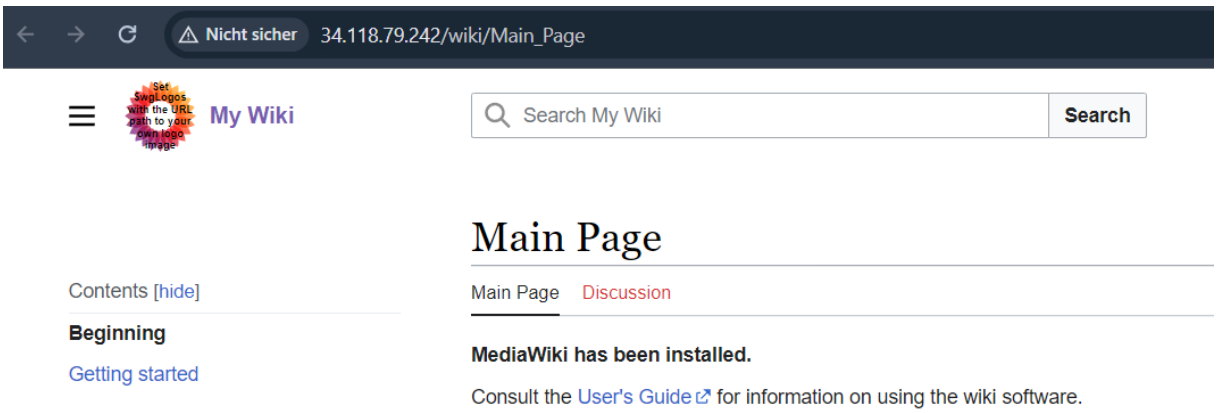
Zunächst sieht alles gut aus, doch dann stellen wir frustriert fest, dass die Seite nicht erreichbar ist. Wir vermuten, dass wir irgendwo einen Fehler gemacht haben. Als wir einige Tage später weiter daran arbeiten wollen, erleben wir die nächste Enttäuschung: die gesamten Cluster ist plötzlich nicht mehr erreichbar.

Wir beginnen erneut und erstellen ein neues Kubernetes-Cluster in der Google Cloud. Dabei folgen wir sorgfältig allen Schritten, um das Helm-Repository einzurichten. Dieser Prozess erfordert viel Geduld und Konzentration.

Trotz der Schwierigkeiten bleibt unser Team entschlossen. Wir befolgten die Schritte, die uns das Helm Repo zuwies und die Installation belief ohne Probleme dieses mal.

```
kovac_daniel@cloudshell:~ (modul-422213)$ kubectl get all
NAME                          READY   STATUS    RESTARTS   AGE
pod/my-mediawiki-mariadb-0    1/1     Running   0          74s

NAME                             TYPE           CLUSTER-IP       EXTERNAL-IP     PORT(S)                      AGE
service/kubernetes               ClusterIP      10.103.160.1     <none>          443/TCP                      37m
service/my-mediawiki             LoadBalancer   10.103.172.79    34.118.79.242   80:30942/TCP,443:31290/TCP   74s
service/my-mediawiki-mariadb     ClusterIP      10.103.163.246   <none>          3306/TCP                     74s

NAME                                       READY   AGE
statefulset.apps/my-mediawiki-mariadb      1/1     75s
kovac_daniel@cloudshell:~ (modul-422213)$  export SERVICE_IP=$(kubectl get svc --namespace default my-mediawiki --template "{{ range (index .status.loadBalancer.ingress 0) }}{{ . }}{{ end }}")
 echo "Mediawiki URL: http://$SERVICE_IP/"
Mediawiki URL: http://34.118.79.242/
kovac_daniel@cloudshell:~ (modul-422213)$      echo Username: user
    echo Password: $(kubectl get secret --namespace default my-mediawiki -o jsonpath="{.data.mediawiki-password}" | base64 -d)
Username: user
Password: mwambwxjwN
```

Schliesslich gelingt es uns, den Cluster erfolgreich einzurichten. Die Erfahrung war äusserst anstrengend und fordernd, aber letztlich auch lohnend. Ohne die Verwendung von Helm wäre der gesamte Prozess deutlich aufwendiger und komplizierter gewesen, da Helm uns dabei hilft, die verschiedenen Komponenten effektiv zu verwalten und zu



installieren.

Testkonzept und Testprotokoll

Testfall-Nummer	Ziel	Schritte	Erwartetes Ergebnis

1	Zugriff auf die Hauptseite	Zuerst öffne ich einen Webbrowser. Dann gebe ich die externe IP-Adresse oder den Domain-Namen unseres MediaWiki-Servers ein.	Die Hauptseite von MediaWiki wird erfolgreich geladen.
2	Anmeldung als Administrator	Ich klicke auf der Hauptseite auf „Anmelden“ und gebe dann unsere Administrator-Zugangsdaten ein.	Ich bin erfolgreich angemeldet und sehe das Admin-Dashboard.
3	Erstellung einer neuen Seite	Nach dem Anmelden wähle ich „Seite erstellen“, trage einen Titel und etwas Inhalt ein und speichere die Seite.	Die neue Seite wird erstellt und zeigt den eingegebenen Inhalt an.

Grafana und Prometheus

Einrichtung

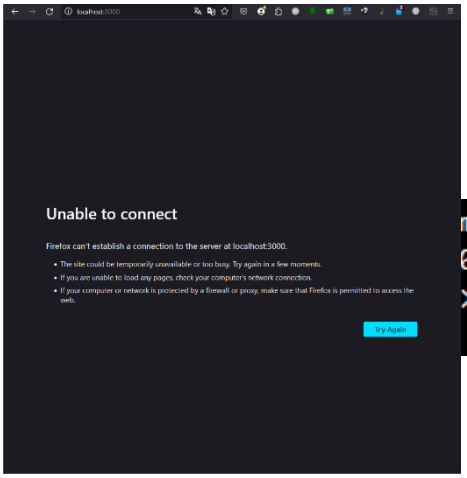
Wir haben versucht, Grafana und Prometheus auf einer Kubernetes-Umgebung zu installieren. Wir begannen mit der Installation der benötigten Tools und richteten das System ein, um die Dienste bereitzustellen. Die Installation schien auf Umwegen zunächst erfolgreich zu sein, wie die Logs und der Status der Pods in den Screenshots zeigen.

```
kristian [ ~ ]$ mkdir -p ~/bin
kristian [ ~ ]$ curl -fsSL -o ~/bin/helm https://get.helm.sh/helm-v3.14.4-linux-amd64.tar.gz
kristian [ ~ ]$ chmod +x ~/bin/helm
kristian [ ~ ]$ echo 'export PATH=$PATH:~/bin' >> ~/.bashrc
kristian [ ~ ]$ source ~/.bashrc
```

Zuerst bereiteten wir die Umgebung vor, indem wir kubectl verwendeten, um die Pods zu prüfen. Die kubectl-Befehle und ihre Ausgaben in den Screenshots zeigten, dass sowohl Grafana als auch Prometheus korrekt liefen. Trotz dieser erfolgreichen Installation hatten wir am Ende jedoch keinen Zugriff auf die Dienste.

```
kristian [ ~ ]$ kubectl get pods --namespace default
NAME                                                READY   STATUS    RESTARTS
alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0
grafana-696dff4c95-h4s45                            1/1     Running   0
prometheus-grafana-d5679d5d7-g4hd7                  3/3     Running   0
prometheus-kube-prometheus-operator-868cf75976-kmhr6   1/1     Running   0
prometheus-kube-state-metrics-547454f49d-q7h5f      1/1     Running   0
prometheus-prometheus-kube-prometheus-prometheus-0   2/2     Running   0
prometheus-prometheus-node-exporter-hg96g           1/1     Running   0
```

Wir verwendeten die Port-Forwarding-Funktion von kubectl, um den Zugang zu Grafana zu ermöglichen, wie im Screenshot sichtbar ist.



Trotz des erfolgreichen Port-Forwardings konnten wir keine Verbindung zum Server aufbauen, was in einem anderen Screenshot als "Unable to connect" im Browser sichtbar ist.

```
mespace default port-forward $POD_NAME 3000
00 -> 3000
> 3000
```

Nachdem wir festgestellt hatten, dass der Zugriff nicht funktionierte, überprüften wir auch das Admin-Passwort von Grafana, um sicherzustellen, dass wir

```
kristian [ ~ ]$ kubectl get secret --namespace default grafana -o jsonpath="{.data.admin-password}" | base64 --decode
 ; echo
q5P2ejLBEzDXVAvVixO2hYf2aK3PZrW0sg0u6Ab5
kristian [ ~ ]$
```

die korrekten Zugangsdaten hatten. Diese wurden korrekt angezeigt, was im Screenshot sichtbar ist. Trotzdem konnten wir keine Verbindung herstellen.

Letztendlich blieb der Zugriff auf die installierten Dienste erfolglos, obwohl die Installationen an sich ohne Probleme verlaufen waren.

Testkonzept und Testprotokoll

Testfall-Nummer	Ziel	Schritte	Erwartetes Ergebnis
1	Grafana-Pod-Bereitstellung	Ich führe kubectl get pods --namespace default aus, um nach dem Grafana-Pod zu suchen und seinen Status zu überprüfen.	Der Grafana-Pod ist erfolgreich bereitgestellt und läuft im Status "Running".
2	Zugriff auf Grafana	Ich führe kubectl --namespace default port-forward $POD_NAME 3000 aus und versuche, auf http://localhost:3000 zuzugreifen.	Der Zugriff auf die Grafana-Weboberfläche ist nicht erfolgreich.
3	Prometheus-Pods überprüfen	Ich führe kubectl get pods --namespace default aus, um nach den Prometheus-Pods zu suchen und ihren Status zu überprüfen.	Die Prometheus-Pods sind erfolgreich bereitgestellt und laufen im Status "Running".

Infrastruktur-Diagramm

Hilfestellungen

Zweck	Quelle
Für jegliche und verschiedenste Fragen	Chat-GPT
Benutzung von Helm	https://helm.sh/docs/
Nachschauen, ob jemand ähnliche Probleme wie wir hatten	https://stackoverflow.com/
Offizielle Kubernetes-Dokumentation	https://kubernetes.io/docs/
Suchen nach vorgefertigten Artefakten	https://artifacthub.io/
Diskutieren wo deren Stand ist und ob sie gleiche Herausforderungen hatten.	Mitschüler

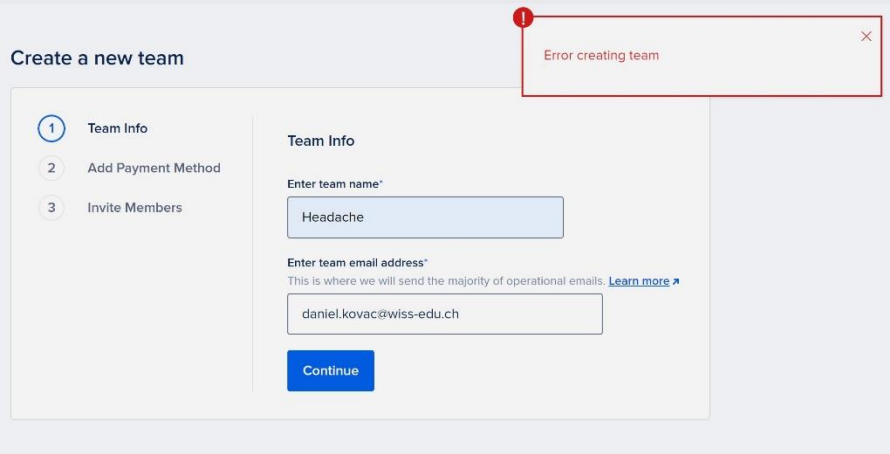
Arbeitsjournal

Arbeitsjournal Daniel Kovac

Arbeitsfortschritt:

Datum: 19.04.2024

- Wir haben begonnen uns Gedanken darüber zu machen wie wir vorgehen sollen, wir mussten auch erstmal viel dokumentieren und lernen, weil all dies ein grosses Thema ist und alles für uns relativ neu war.
- Wir haben damit begonnen den perfekten Cloud Anbieter zu finden. Unsere Lehrperson hat uns Digital Ocean empfohlen aber ich hatte Probleme damit. Schon zu beginn an hatte ich damit zu kämpfen und auch nachdem ich dem support geschrieben habe ging es nicht rund. Danach ging ich einfach rüber zu Google Cloud. Dort funktioniere alles erstmal.
- Wir haben uns schlau gemacht, recherchiert und rumprobiert.
- Jedes Teammitglied hat begonnen, die Konfigurationsdateien für die verschiedenen Microservices vorzubereiten und mit den yaml Files zu experimentieren.
- Jedes Teammitglied hat sein persönliches Arbeitsjournal begonnen, um den individuellen Beitrag zum Projekt festzuhalten.
- Wir haben die ersten Schritte für die Erstellung der Projektdokumentation unternommen, einschließlich der Definition von Anforderungen und der Testpläne.



Datum: 26.04.2024

- Wir hatten diese Woche eine Block-Woche. Die anderen haben sich während dieser Woche mehr mit dem Modul347 beschäftigt als mit dem Modul426. Ich und Kristian nicht da und das Modul426 auch wichtig war und wir nicht den ganzen Unterricht verpassen wollten. Deswegen waren wir während der Präsentation noch nicht so weit.
- Während des Unterrichtes haben wir uns für das Modul426 konzentriert, zuhause haben wir aber probiert weiterzuarbeiten. Sehr, sehr, sehr viele Kopfschmerzen hat es uns gebracht. (Obwohl es am Schluss relativ simpel erscheint)
- Wir haben zusammen die Präsentation erstellt und am Samstag vorgetragen (Mit dem jetzigen Stand, wo wir uns aktuell befinden, wäre die Präsentation um einiges besser geworden)

Datum: -- 02.05.2024

- Wir haben viel Zeit damit verbracht damit es lauft und momentan sieht es nicht schlecht aus. Einige Tage haben wir in den Ferien zusammen verbracht damit es funktioniert. Zu guter Letzt konnte ich MediaWiki und WordPress Aufsetzten und Kristian konnte JIRA aufsetzten. Es gibt noch einiges zu tun aber dies werden wir hoffentlich auch noch lösen.
- Wir schreiben zusammen weiter an der Dokumentation

Herausforderungen und Lösungen:

- Wir sind auf viele Schwierigkeiten bei der Konfiguration von Kubernetes gestossen, einrichten einer Cloud, mit den Yaml files und mit dem Aufsetzten der Application Stacks in einer Cloud Umgebung allgemein. Aber zu guter Letzt sind wir weit gekommen und haben viele Herausforderungen lösen können.

Arbeitsjournal Kristian Lubina

19. April 2024

- Ich habe hier begonnen zu überlegen, wie ich vorgehen sollte. Zunächst musste ich viel noch anschauen, da mir das Thema immer noch sehr neu war. Auch habe ich schon begonnen die Präsentation für die nächste Woche vorzubereiten.

20. April.2024

- Ich begann damit, den idealen Cloud-Anbieter zu finden. Unsere Lehrperson empfahl uns DigitalOcean, aber ich hatte Schwierigkeiten damit. Ich konnte nicht direkt eine Cloud einrichten, das kam erst später.
- Jedes Teammitglied begann sein persönliches Arbeitsjournal, um den individuellen Beitrag zum Projekt festzuhalten.
- Ich beschloss mit Daniel dass ich während der ganzen Arbeit auf die Dokumentation achten während dieser LB-Arbeit.

22. April. 2024- 26. April. 2024

- Diese Woche hatten wir eine Blockwoche. Die anderen haben sich mehr mit Modul 347 als mit Modul 426 beschäftigt. Wir haben uns Modul 426 gewidmet, um nicht den ganzen Unterricht zu verpassen.
- Wir haben zusammen die Präsentation über den damaligen Stand unserer LB-Arbeit erstellt und sie am Samstag vorgetragen.

29. April. 2024 - 02.Mai.2024

- Erst in der letzten Woche konnte ich wirklich versuchen loszulegen, oder es zumindest zu versuchen. Wir hatten in dieser Woche auch eine andere Prüfung in Modul 320, die wir abgeben mussten. Das alles bedeutete viel Arbeit auf einmal in so einer kurzen Zeit.
- Ich habe viel Zeit darauf verwendet, die Dienste zum Laufen zu bringen. Ich habe mit Daniel einige Tage in den Ferien zusammen verbracht, um die Systeme einzurichten. Schließlich habe ich Jira eingerichtet.
- Wir schreiben zusammen weiter an der Dokumentation. Wir sind auf viele Schwierigkeiten bei der Konfiguration von Kubernetes gestoßen, beim Einrichten der Cloud, mit den YAML-Dateien und allgemein bei der Einrichtung der

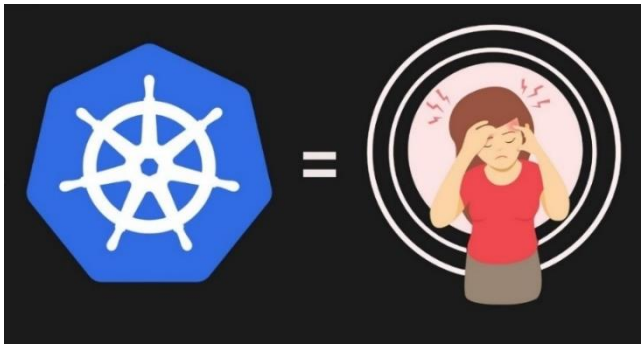
Application Stacks in einer Cloud-Umgebung. Letztendlich sind wir weit gekommen und haben viele Herausforderungen lösen können.

Herausforderungen und Lösungen:

Der Anfang war besonders herausfordernd, vor allem da die Hilfsmittel schwer zu finden waren und das Modul zu einem ungünstigen Zeitpunkt kam, als wir wenig Zeit hatten. Wir hatten viele Schwierigkeiten bei der Konfiguration von Kubernetes, der Einrichtung einer Cloud und der Verwendung von YAML-Dateien. Trotzdem haben wir es geschafft, die Herausforderungen zu meistern.

Persönliche Fazit

Persönliches Fazit Daniel Kovac



Das Modul war äusserst anspruchsvoll und erforderte erhebliche Anstrengungen. Die Arbeit war weit zeitintensiver als in anderen Modulen. Besonders anstrengend war es, nach einer Blockwoche sofort mit einer so grossen Aufgabe konfrontiert zu werden.

Diese Aufgabe war keine leichte Herausforderung. Der Arbeitsaufwand war deutlich grösser als die vorgesehene Zeit von 8 x 45 Minuten pro Person. Wir haben nun zahlreiche Stunden und Tage in diese Aufgabe investiert und hoffen, dass wir sie einigermassen gut gelöst haben.

Persönliches Fazit Kristian Lubina

Ich war bereits durch die Gespräche über das Modul stark beansprucht. Durch eine effiziente Aufgabenverteilung wir versuchten wir das bestmögliche Ergebnis zu erzielen.

Der Zeitpunkt für diese Aufgabe war ungünstig, insbesondere im Hinblick auf die vorhergehende Blockwoche und auch dass der grösste Teil von uns noch ein Praktikum sucht (nur 4 von 21 Schüler haben eine Praktikumsstelle). Die Kombination aus erheblichem Arbeitsaufwand und unglücklichem Timing machte die Arbeit besonders anspruchsvoll. Dennoch gelang es uns, uns gut zu organisieren und die Herausforderung gemeinsam probieren zu meistern.