Informatik Compute Cloud (ICC)

Die Informatik Compute Cloud ist eine vom AI Labor zur Verfügung gestellte Container Cloud Umgebung, in der Mitglieder des Departments Applikationen in Form von Docker Containern betreiben können. Der Betrieb befindet sich derzeit in der Beta-Phase. Es ist jedem Mitglied des Departments Informatik möglich, sich in der Cloud einzuloggen. Zuvor wird derzeit der Login in Gitlab (https://gitlab.informatik.haw-hamburg.de) empfohlen, da Berechtigungen in der Cloud über eine Anbindung an Gitlab konfiguriert werden. Bei weiteren Fragen hilft Ihnen das ICC Team icc@informatik.haw-hamburg.de

Ein Tutorial für die Verwendung der ICC zusammen mit Gitlab finden Sie hier zum Download: icc_tutorial_hello_world.pdf

Ein FAQ mit häufigen Fragen findet sich hier::docu:informatikcomputecloudfaq

Hilfe und andere Nutzer zur ICC, Gitlab und Nexus finden Sie im Slack Team. Sie können mit einer HAW Kennung über den folgenden Einladungslink Beitreten: Slack HAW-HH-AIL Invite Link

Ticket System für Bug Reports und längere Nachverfolgung von Problemen oder Requests zusätzlich zur Slack via Gitlab: https://gitlab.informatik.haw-hamburg.de/icc/icc-docs/issues

Alternativ erreichen Sie das ICC Team per Mail via icc@informatik.haw-hamburg.de

Für weitere Details zur ICC klicken Sie auf den Weiterlesen Link.

Installation / Vorbereitungen

kubectl - Command Line für Kubernetes

Um innerhalb der Informatik Compute Cloud auf Container oder Konfigurationsobjekte zugreifen zu können, benötigen Sie das Werkzeug `kubectl`. Sie können dies unter folgendem Link für viele Betriebssysteme herunterladen (Windows, MacOS, Linux, nur für die x86_64). Stellen Sie sicher, dass sie es an einen Ort kopieren, der in Ihrer PATH Variable enthalten ist bzw. dass Sie Ihre PATH Variable entsprechend anpassen:

- Linux: https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/linux/amd64/kubectl
- macOS: https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/darwin/amd64/kubectl
- Windows: https://storage.googleapis.com/kubernetes-release/release/v1.14.0/bin/windows/amd64/kubectl.exe

Login

Um sich mit der Informatik Compute Cloud zu verbinden, ist ein Login gegen die Infrastruktur notwendig. Dies geschieht mithilfe des 'kubelogin' Werkzeuges, welches Sie unter den folgenden

Links für Ihr Betriebssystem herunterladen können:

- Linux: kubelogin_linux –> Nach dem Download umbenennen in kubelogin
- macOS : kubelogin_macos –> Nach dem Download umbenennen in kubelogin
- Windows: kubelogin.exe

Führen Sie das Programm anschliessend in einem Terminal (OSX & Linux) bzw. einer Command Sitzung (Windows) aus und folgen Sie den Anweisungen. Unter Linux und macOS müssen Sie die Datei ggf. noch via '`chmod +x kubelogin`' ausführbar machen. Das Werkzeug fragt Ihre Credentials ab und erzeugt eine config Datei in Ihrem Heimverzeichnis im Unterordner .kube.

Das Kubelogin Werkzeug ruft ein Token aus dem Cluster ab, mit dem Sie zukünftig authentifiziert werden. Dieses Token hat eine beschränkte Gültigkeit von 12 Stunden, sodass Sie 'kubelogin' erneut aufrufen müssen, sobald 'kubectl' eine Meldung darüber ausgibt, dass Sie nicht authentifiziert sind. Ein expliziter Logout ist somit nicht notwendig.

Kontext und Namespace

kubectl arbeitet mit sog. Kontexten um verschiedene Konfigurationen zu verwalten. Ein Kontext ist immer aktiv, sodass dieser bei Nutzung von kubectl genutzt wird.

Kubernetes selbst definiert Namespaces um Ressourcen logisch zu trennen. In der ICC erhält jeder Nutzer einen eigenen Namespace. In diesem darf nur der Nutzer selbst arbeiten.

Die geschriebene Kubeconfig setzt die ICC automatisch als aktiven Kontext und definiert Ihren privaten Namespace (identisch mit Ihrer Nutzerkennung) als automatisch gesetzt. D.h. jedweder Aufruf von 'kubectl' wird automatisch mittels des ICC Kontextes sowie gegen Ihren privaten Namespace durchgeführt werden. Beachten Sie dies, falls Sie in mehreren Projekten gleichzeitig arbeiten und ggf. verschiedene Kontexte oder Namespaces verwenden müssen.

Kurzzeitiger Wechsel von Kontext und Namespace

Um einen anderen Kontext oder Namespace als den per Standard eingestellten zu verwenden, können Sie auf die Parameter –context und -n zurückgreifen. '`kubectl –context my-context …`' würde bspw. den Kontext „my-context“ nutzen, während '`kubectl -n my-namespace …`' für den jeweiligen Aufruf den Namespace „my-namespace“ annimmt.

Arbeiten in Gruppen: Synchronisation von Gitlab mit Kubernetes

Das AI Labor bietet Gitlab (https://gitlab.informatik.haw-hamburg.de) als Dienst zum Hosting von Git Repositories, Durchführen von Build Pipelines sowie genereller Gruppenarbeit im Department Informatik an. Da für Veranstaltungen, Projekte etc. Gruppen und Projekt in Gitlab angelegt werden, wurde eine Synchronisation dieser Gruppen und Projekte mit der ICC implementiert.

Die Synchronisation bewirkt, dass jede Gruppe, jedes Projekt in Gruppen sowie jedes private Projekt von Benutzern automatisch in Form von Namespaces in Kubernetes reflektiert wird. Anders ausgedrückt gibt es zu jeder Gruppe und jedem Projekt einen entsprechenden Namespace in Kubernetes.

Zusätzlich werden in Kubernetes Rollen definiert, die steuern, welcher Nutzer in welchen Namespaces welche Operationen ausführen darf. Diese Rollen orientieren sich an den Rollen in Gitlab und sind grob wie folgt gemapped:

Gitlab	Kubernetes / ICC
Owner	Darf Pods, Deployments etc. erstellen, löschen und Logs auslesen
Master	wie Owner
Developer	Darf Pods, Deployments etc. listen und Logs dafür auslesen, aber nichts ändern
Reporter	wie Developer
Guest	darf nichts

Niemand darf DaemonSets deployen oder privilegierte Pods starten! Das ist nur nach gesonderter Genehmigung durch einen Admin möglich. Sprecht uns dazu bitte an.

Aufgrund dieser Mechaniken ist es möglich die Berechtiungen für den Kubernetes Cluster über Zugehörigkeiten zu Gitlab Projekten und Gruppen zu steuern.

Gitlab	Kubernetes	Beispiel
Persönliches Projekt	Namespace des gleichen Namens	student-Bob → student-bob
Gruppe	Namespace des gleichen Namens	Foo-Group → foo-group
Sub-Gruppe	Namespace des gleichen Namens, geprefixt mit „$GruppenName$-„	Foo-Group/bar-subgroup → foo-group-bar-subgroup
Projekte	Namespace des gleichen Namens, geprefixt mit „$GruppenName$-“ und „$Sub-GruppenName$-“ falls anwendbar	Foo-Group/bar-subgroup/MyProject → foo-group-bar-subgroup-myproject

Löschen von Gitlab Projekten/Gruppen und Kubernetes Namespaces

Achtung: Wenn eine Gitlab Entität angelegt wird, entsteht auch sogleich der Namespace in Kubernetes. Genauso wird dieser aber auch gelöscht, wenn Sie ihr Projekt oder Ihre Gruppe in Gitlab entfernen.

Beim Entfernen eines Namespaces in der ICC werden ALLE Ressourcen darin gelöscht und alle PersistentVolumes wieder freigegeben. Somit ist das Löschen eines Gitlab Projektes höchst effizient zum Aufräumen, aber eben auch gefährlich.

Zugriff von einer Workstation auf Services / Schnittstellen innerhalb des Clusters

Um von Ihrer Workstation / Ihrem Laptop aus auf im Cluster laufenden Schnittstellen zuzugreifen, bietet sich der Proxy-Modus von kubectl an. Um diesen zu nutzen, starten Sie in einem Terminal 'kubectl proxy'. Daraufhin wird kubectl auf Ihrem Rechnern einen Endpunkt anbieten über welchen Sie via HTTPS auf die von Ihnen im Cluster gehosteten Schnittstellen zugreifen können. Die zu verwendende Adresse folgt dabei folgendem Namensschema: `http://localhost:8001/api/v1/namespaces/namespace_name/services/service_name[:port_name]/proxy`

Zugriff auf Logs & Alerting

Bei einzelnen Pods kann problemlos mit `kubectl logs <PodName>` gearbeitet werden. Wird ein Deployment jedoch größer, oder hat man z.B. Datenbank-Cluster oder andere Dienste mit mehreren Pods zu verwalten, wird die Arbeit häufig mühsam und schwierig.

Resourcennutzung von Pods anzeigen lassen

Dies funktioniert zurzeit nicht aufgrund eines Zertifikatsproblems.

Mit dem folgenden Befehl lassen sich die aktuell genutzten Resourcen aller Pods in einem Namespace anzeigen:

```
kubectl -n <NameDesNamespaces> top pods
```

Die Einheit „m“ steht dabei für millicores (also 1/1000 CPU Kern) und „Mi“ für Megabyte.

Diese Funktion lässt sich u.a. gut benutzen um vernünftige ResourceRequests an Deployments etc. zu schreiben –> https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/#resource-requests-and-limits-of-pod-and-container

Dienste weltweit zur Verfügung stellen

Der Zugriff via ***kubectl proxy*** ist für die Entwicklung oder im Rahmen eines Praktikums völlig ausreichend. Ist jedoch gewünscht einen Service innerhalb der ICC zu hosten und diesen dauerhaft und gesichert in der Welt verfügbar zu machen, so reicht dieses Verfahren nicht aus. Stattdessen muss Ihr Service über einen Gateway vom Cluster in die Welt verfügbar gemacht werden. Die ICC unterstützt diesen Betrieb von Services, Webseiten etc. **via HTTPS** ausdrücklich.

Um Ihren Dienst zur Verfügung zu stellen, benötigen Sie folgende Dinge:

1. Eine konfigurierte Domäne
2. Eine Ingress Beschreibung für Ihren Dienst in Ihrem Namespace

Rechtliche Auflagen

Um den Datenschutz und den Wettbewerbsregeln gerecht zu werden, muss der von Ihnen zu Verfügung gestellte Dienst entweder über eine Zugangsbeschränkung oder über ein korrektes Impressum mit Informationen zum Datenschutz verfügen.

Eine Zugansbeschränkung muss den Zugriff auf den Dienst auf den Kreis der berechtigten Personen beschränken, z.B. über die Verwendung eines Passwortes.

Für die Form und den Inhalt des Impressums wenden Sie sich bitte an die Rechtsabteilung der HAW.

Vorlagen für Impressum und Datenschutzerklärung

- Impressum (docx) | Impressum (pdf)
- Datenschutzerklärung (docx) | Datenschutzerklärung (pdf)

HowTo

Die Domäne müssen Sie extern verwalten oder bei uns einrichten lassen. Für das zugehörige Zertifikat jedoch **verfügt die ICC über einen Mechanismus zur vollautomatischen Erzeugung von weltweit gültigen Let's Encrypt Zertifikaten!** Hier sind die Schritte im Einzelnen:

(Die Punkte 1.a.II und 1.b sollen in Zukunft auch noch automatisiert werden)

1. Sie benötigen eine vollwertige (Sub-)Domain (www.example.com) entweder von:

1. einem externen Anbieter (z.B. von strato.de, etc.). **Die Registrierung erfolgt durch Sie über den jeweiligen Anbieter!**
   1. **Konfigurieren Sie** bei **Ihrem Anbieter** Ihre Domain, sodass diese als **CNAME** Eintrag auf den Namen `icc-k8s-api.informatik.haw-hamburg.de` zeigt!
   2. Schicken Sie Ihren Domainnamen an icc@informatik.haw-hamburg.de. Wir richten dann eine entsprechende Weiterleitung im Gateway Server ein, sodass Ihre Seite erreichbar ist.
2. aus dem Department Informatik (example.informatik.haw-hamburg.de). **Die Registrierung und DNS Eintragung erfolgt über AI Labor Mitarbeiter** (Mail mit gewünschter Sub-Domain an icc@informatik.haw-hamburg.de)
3. Legen Sie eine Ingress Ressource (https://kubernetes.io/docs/concepts/services-networking/ingress/) in Ihrem Namespace für Ihren Domainnamen und Ihre Services gemäß dieser Vorlage an:

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  annotations:
    kubernetes.io/tls-acme: "true"
    kubernetes.io/ingress.class: "nginx"
  name: NAME_DES_INGRESS
  namespace: IHR_NAMESPACE
spec:
  tls:
  - hosts:
    - IHRE_DOMAIN.informatik.haw-hamburg.de
    secretName: NAME_IHRES_SECRETS
  rules:
  - host: IHRE_DOMAIN.informatik.haw-hamburg.de
    http:
      paths:
```

```
      - path: /
        backend:
            serviceName: NAME_IHRES_SERVICES
            servicePort: NAME_DES_SERVICE_PORTS
```

Durch die Ingress Ressource wird vom Cluster ein Zertifikat für Ihre Domäne bei Let's Encrypt angefordert und als Secret in Ihren Cluster hinterlegt.

Hinweis 1: Das Feld secretName bestimmt wie das Secret mit dem automatisch erzeugten SSL Zertifikat in Ihrem Namespace heißen wird. Dieses Secret darf es noch nicht geben, wenn Sie die Ingress Ressource für Ihren Namespace anlegen!

Hinweis 2: Verwenden Sie für den Namen des Secrets nur Zahlen, Kleinbuchstaben und Bindestriche. Ansonsten kann das Zertifikat nicht richtig beantragt und eingerichtet werden.

Hinweis 3: Let's Encrypt hat eine Mengenbeschränkung von 20 Zertifikaten für vollständige Domains pro Woche. Es kann also sein, dass Sie ihr Zertifikat nicht sofort bekommen, falls es ein sehr hohes Registrierungsaufkommen geben sollte. **Im Sinne der Fairness bitten wir darum, sich den Namen und die Notwendigkeit von Zertifikaten gut im Vorfeld zu überlegen.**

Die Ingress Ressource definiert, wie Ihre Services unter Ihrer Domain erreichbar sind. Weitere Details und Einzelheiten entnehmen Sie bitte der Ingress Dokumentation des Kubernetes Projektes: https://kubernetes.io/docs/concepts/services-networking/ingress/

Dienste hochverfügbar zur Verfügung stellen (H/A Deployment)

Wer die ICC nutzt um einen Dienst oder eine Technologie zu testen oder sonstige relativ kurzweilige Rechenaufgaben auszuführen, benötigt kein H/A (High Availability) Deployment. Für alle anderen, die Dienste für Dritte und damit „produktiv“ anbieten wollen und ggf. sogar Datenbanken betreiben möchten, ist es nötig sich einige zusätzliche Gedanken um das Deployment dieser Dienste zu machen.

Zustandslos vs. Zustandsbehaftet

Für Applikationen kann von Zustandslosigkeit gesprochen werden, wenn Aufrufe an Ihre API keinerlei Annahmen über den Zustand des Arbeitsspeichers oder den Inhalt der Festplatte machen müssen, um erfolgreich beantwortet werden zu können. Ist das nicht der Fall, spricht man bei der Applikation von einer zustandsbehafteten App. Datenbanken sind in der Regel **nicht zustandslos**.

Betriebsmodel in Kubernetes

Alle Komponenten in Kubernetes sind darauf ausgelegt ausfallen zu können, ohne andere Komponenten in Mitleidenschaft zu ziehen. Aufgrund dieser Tatsache ist es auch jederzeit möglich jeden Bestandteil der Clusterlösung zu aktualisieren, ohne, dass andere Teile davon beeinträchtigt sind. Ist zusätzlich jede Komponente mehrfach vorhanden und hat man sich Gedanken um ein sog. Rolling-Update gemacht, kann man sogar den ganzen Cluster und alle seine Bestandteile updaten, ohne dass der Cluster jemals offline ist (Zero Downtime).

Bei einem solchen Model bleibt es nicht aus, dass Knoten im Cluster ebenfalls neugestartet werden müssen. Für die auf diesen Knoten laufenden Pods bedeutet das, dass sie vom Knoten entfernt und anderswo gestartet werden. Das heißt in der Folge aber auch, dass die Apps in diesen Pods demselben Betriebsmodel folgen sollten, wie der Kubernetes Cluster selbst es tut. **Apps sind folglich unter der Annahme zu entwickeln, dass sie jederzeit ausgeschaltet und anderswo neugestartet werden können! (siehe** https://12factor.net/ **)**

Der Neustart von Pods wird indes nicht nur durch Updates ausgelöst, sondern kann auch mal durch ein notwendiges Re-Scheduling zur Optimierung der Resourcennutzung oder als Folge anderer Probleme wie etwa Netzwerksegmentierung etc. erfolgen.

H/A für zustandslose Applikationen: Normale Deployments/Pods

Ist die Applikation zustandslos, kann man diese mit K8s Deployments ausrollen, die dann wiederum dafür sorgen, dass die spezifizierte Anzahl Pods (Feld: replicas) im Cluster gestartet werden. Durch die Zustandslosigkeit nimmt die App keinen Anstoß daran plötzlich ausgeschaltet und anderswo neugestartet zu werden. Durch die Verwendung mehrerer identischer Replicas derselben App sowie der Nutzung der Kubernetes Service Entität zur Bündelung der Anfragen an einen Service, wird auf diese Weise Hochverfügbarkeit gewährleistet. Die ICC verfügt derzeit über mehr als 30 Knoten, sodass ein gleichzeitiger Ausfall aller Knoten recht unwahrscheinlich ist.

H/A für zustandsbehaftete Applikationen: StatefulSets

Ist die App nicht zustandslos, etwa bei Datenbanken, so ist die Verwendung eines Statefulsets ratsam. Statefulsets funktionieren zunächst genauso wie Deployment, haben jedoch den Unterschied, dass die erzeugten Pods einzigartige und nicht arbiträre Namen bekommen. Ein Pod einer MongoDB aus einem Deployment heitß etwa „mongodb-deploy-12z352“, während ein Pod eines MongoDB StatefulSets „mongodb-1“ heißen könnte. Die zufallsgenerierte ID weicht also einer klaren Pod-ID, die auch wiederverwendet wird. Kubernetes folgt hierbei der „at-most-once“ Semantik, d.h. es gibt jeden Pod mit seinem Namen nur genau einmal.

Entsprechend anders geht Kubernetes auch beim Löschen von StatefulSet Pods vor. Diese werden nicht beliebig entfernt, sondern nach Möglichkeit ganz zuletzt gelöscht (etwa bei Aufräumarbeiten). Wird ein StatefulSet heruntergefahren geschieht dies in umgekehrter Reihenfolge zur Nummerierung. Bei geplanten Neustarts von Knoten, etwa für Updates, wird sichergestellt, dass von einem StatefulSet immer eine definierte Anzahl von Instanzen läuft, um die Konsistenz des jeweiligen Dienstes nicht zu gefährden.

Mit dem Verlust einzelner Knoten umzugehen ist indes Aufgabe der jeweiligen Applikation.

Mehr zu StatefulSets unter: https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

Ausfall von StatefulSet Pods

Manchmal kommt es vor, dass Pods eine StatefulSets in den Zustand „unknown“ kommen. In diesem Fall werden die betroffnen Pods nicht gelöscht und neugestartet, da Kubernetes nicht wissen kann, ob der Pod möglicherweise noch läuft. „unknown“ heißt in diesem Fall nämlich nur, dass das Kubelet des Hostes, auf dem der Pod läuft aktuell keine Statusupdates herausgibt. Dies kann viele Gründe haben und auch nur vorübergehender Natur sein. Würde Kubernetes den Pod anderswo neustarten, könnte es zur Verletzung der „at-most-once“ Semantik kommen, was auch für die Applikation fatale Folgen haben könnte.

Zur Behebung dieser Situation gibt es 3 Optionen:

1. Löschen des Knotens aus Kubernetes

2. Knoten bzw. Kubelet wieder in einen positiven Zustand bringen (aká beim ICC Betriebsteam Bescheid sagen!)

3. Die Löschung des Pods erzwingen

Empfohlen als Best Practice ist 2.. Punkt 1 würde nur greifen, wenn das ICC Team entschieden hat einen Knoten aus dem Cluster zu entfernen, in diesem Fall würden wir den Knoten aber freiräumen (kubectl drain), sodass der Zustand wie oben beschrieben nicht erreicht werden würde.

Punkt 3 ist wie oben beschrieben kritisch, kann aber auf eigene Verantwortung gemacht werden (kubectl delete po <Name> –force –grace-period=0).

Mehr zu den Hintergründen etc. ist hier zu lesen: https://kubernetes.io/docs/tasks/run-application/force-delete-stateful-set-pod/

H/A Betrieb von Datenbanken

Grundsätzlich gilt hier alles, was unter „H/A für zustandsbehaftete Applikationen: StatefulSets“ geschrieben wurde. Zusätzlich ist jedoch darauf hinzuweisen, dass die Datenbank entsprechend konfiguriert werden muss, um mit dem Hinzufügen und Entfernen von Knoten gut umgehen zu können. Da jede Datenbank anders funktioniert, ist diese Konfiguration zwischen verschiedenen DBMS nicht identisch und kann hier daher nicht wiedergegeben werden.

Jedoch lohnt es sich zunächst eine google Recherche durchzuführen, da es für viele DBMS bereits Templates und Anleitungen für den H/A Betrieb in Kubernetes gibt. Beispiele sind etwa MySQL, MongoDB, InfluxDB um nur einige zu nennen. Einge Datenbanken, wie z.B. Riak, Cassandra sind sogar für einen verteilten/HA Einsatz von Grund auf entworfen worden.

Batch Jobs ausführen

Das Atom des K8s Schedulings ist der Pod. Dieser sagt jedoch noch nichts darüber aus, wie die Ausführungsumgebung den Pod behandelt. Normale Services werden als Deployment gestartet und entsprechend stellt K8s sicher, dass jederzeit die gewünschte Anzahl Pods zu diesem Deployment verfügbar ist.

Möchte man nun aber einen Batch Job ausführen, so will man in der Regel, dass dieser genau einmal vollständig durchläuft. Gelegentlich gibt es komplexere Anforderungsszenarien mit mehreren voneinander abhängigen Jobs.

für diese Zwecke beitet Kubernetes den Resourcentyp „Job“ an. Jobs lassen sich definieren und deployen wie Deployments, werden dann aber vom Laufzeitsystem dahingehend anders behandelt, als dass sie nur solange neugestartet werden, bis sie einmal den Zustand „completed“ erreicht haben. Was das bedeutet, lässt sich jeweils festlegen.

Details und Beispiele zu Jobs finden sich in der Kubernetes Dokumentaton: https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/

Cron Jobs

Analog zu (Batch) Jobs, gibt es auch noch CronJobs ion Kubernetes. Sie funktionieren ähnlich wie ihr Pendant in der Unix Welt. Hier die Doku:
https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

Persistente Speicherung von Daten

Wenn Pods zustandsbehaftet sind und diese Daten dauerhaft gespeicher werden soll, so sind weitere Schritte notwendig, da sonst mit dem Löschen des StatefulSets die Daten verloren sind.

Persistente Daten in der ICC werden auf mit Hilfe von Ceph redundant abgelegt und nach bestem Wissen und Gewissen gepflegt.

ACHTUNG: Die ICC bietet keinen Backup-Dienst an! Ein vollständiger Datenverlust kann geschehen (z.B. auch durch Fehlbedienung). Daher ist jeder Anwender selbst aufgefordert sich um externes Backup zu kümmern.

Persistentem Speicherplatz anlegen

Das AI-Labor bietet derzeit begrenzten persistenten Speicher als Block Storage über einen CEPH Cluster an (http://ceph.com/ceph-storage/block-storage/). Dieser Storage kann in der ICC in From von Volumes mit festgelegter Größe genutzt werden. Um ein Storage Volume anzufordern gehen Sie wie folgt vor:

1. Legen Sie in Ihrem Projekt eine YAML Datei 'my-pvc.yaml' für einen PersistentVolumeClaim (https://kubernetes.io/docs/concepts/storage/persistent-volumes/#persistentvolumeclaims) mit folgendem Inhalt an:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my-storage-claim
  labels:
    service: my-service
```

```
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 5Gi
```

Die Storage Größe können Sie variieren, in dem sie den Wert in der letzten Zeile verändern. **Bedenken Sie jedoch, dass Storage nur begrenzt zur Verfügung steht und Volumes (derzeit!) nicht vergrößer- oder verkleinerbar sind!** Der Zugriffsmodus steht auf `ReadWriteOnce`, was bedeutet, dass Lesend wie Schreibend nur von einem Pod auf dieses Volume zugreifbar ist. Alternativen finden Sie in der Kubernetes Dokumentation.

1. Legen sie den Claim mittels `kubectl apply -f my-pvc.yaml` im Cluster an
2. Referenzieren Sie das Volume in einer YAML Beschreibung eines Pods und geben Sie einen Mountpfad an:

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: dockerfile/nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: my-volume
  volumes:
    - name: my-volume
      persistentVolumeClaim:
        claimName: my-storage-claim
```

1. Sobald der Pod erzeugt wird, wird der PersistentVolumeClaim durch Kubernetes an ein PersistentVolume gebunden, welches gegen den CEPH Cluster provisioniert wird. Je nach angeforderter Größe kann die Provisionierung zwischen wenigen Sekunden (< 1GB) bis hin zu mehreren Minuten in Anspruch nehmen.
2. Etwaige Fehler sind via `kubectl get pvc` und `kubectl describe pvc <PVCName>` einsehbar

WICHTIG: Das erzeugte PersistentVolume bleibt solange erhalten, wie der PersistentVolumeClaim vorhanden ist. Sie können also Ihre Pods und Services löschen ohne Ihre Daten zu verlieren. Löschen Sie aber den PersistentVolumeClaim, so sind auch die Daten weg.

Deployment einer Applikation aus Gitlab in die ICC

Pipelines in Gitlab haben die Möglichkeit über das Services Feature von Gitlab eine Kubernetes Integration aufzusetzen. Die Integration bewirkt, dass in der Build Pipeline spezielle K8s Variablen zur

Verfügung stehen, mit denen das Deployment via kubectl durchgeführt werden kann. Weiterhin gibt es einige Features von Gitlab selbst wie etwa Monitoring der Applikationen und ein Rollback auf frühere Versionen, die dadurch aktiviert werden.

Für diese Integration wird ein Token, ein Zertifikat, der Namespace-Name in K8s und eine URL benötigt. Diese Werte werden durch den Gitlab-K8s-Integrator in der ICC für jedes Projekt in Gitlab vollautomatisch für Sie gesetzt, sodass Sie die Integration nur noch zu nutzen brauchen. Auch wird ein CI/CD Environment mit Namen „icc-dev“ angelegt, welches Sie in Ihrer Pipeline für den Deploy Job referenzieren müssen. Mehr dazu im nachfolgenden How-To Abschnitt.

Randbedingungen & Empfehlungen

- Die im jeweiligen Projekt hinterlegte K8s-Konfiguration gilt nur für den jeweilig assoziierten Namespace. Sie können also nicht aus Projekt A in den Namespace zu Projekt B deployen!
- Den **Namen des mit Ihrem Projekt assoziierten Namespaces** finden Sie in den Settings der Kubernetes Integration unter ***Projekt→CI/CD→Clusters→KubernetesService***
- Die K8s Konfiguration ist **<u>derzeit</u> nicht änderbar.** Etwaige Änderungen werden spätestens beim nächsten Synchronisationslauf des Gitlab Integrators überschrieben.
- Legen Sie sich für das Deployment Ihrer Dienste idealerweise ein **spezielles Deployment Projekt** mit einem guten Namen an und verwalten Sie hier Ihre YAML Dateien
- Verwenden Sie ggf. Gitlab Pipeline Trigger um beim Neubau eines Services in einem anderen Projekt, die Pipeline Ihres Deployment Projektes anzustoßen

HowTo

Um die Integration nutzen zu können sollte Ihr Build-Plan als Ergebnis ein oder mehrere Images für Ihre Applikation erzeugt und in einer Registry abgelegt haben. Zusätzlich benötigen Sie Kubernetes YAML Dateien in dem mit Ihrem Build-Plan assoziierten Repository, die ihr Deployment in Kubernetes beschreiben. **Wenn Ihnen diese Begriffe / Konzepte nichts sagen, empfehlen wir Ihnen dringend das ICC-Gitlab-Tutorial durchzuarbeiten bevor Sie fortfahren.**

Haben Sie all dies beisammen, so können Sie in Ihrer .gitlab-ci-yml Datei eine weitere Stage mit einem weiteren Job anlegen um ein Deployment gegenüber Kubernetes durchzuführen. Zu diesem Zweck halten wir ein Image mit kubectl in der jeweils kompatiblen Version im Nexus vor Verfügung. Hier ist eine Beispiel gitlab-ci.yml aus dem Tutorial Beispiel:

(das vollständige Beispiel finden Sie in Gitlab unter folgender Adresse: https://gitlab.informatik.haw-hamburg.de/ail/haw-world/tree/master)

```
# .gitlab-ci.yml
stages:
    - dockerize
    - deploy

variables:
 DOCKER_HOST: "tcp://localhost:2375"
 DOCKER_REGISTRY: "docker-hub.informatik.haw-hamburg.de"
 SERVICE_NAME: "haw-world"
```

```
createImage:
 stage: dockerize
 image: docker-hub.informatik.haw-hamburg.de/icc/docker-dind
 services:
   - docker-hub.informatik.haw-hamburg.de/icc/docker-dind
 script:
   - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN $DOCKER_REGISTRY
   - docker build -t $SERVICE_NAME:latest .
   - docker tag $SERVICE_NAME:latest
$DOCKER_REGISTRY/$CI_PROJECT_NAMESPACE/$SERVICE_NAME:$CI_PIPELINE_ID
   - docker tag $SERVICE_NAME:latest
$DOCKER_REGISTRY/$CI_PROJECT_NAMESPACE/$SERVICE_NAME:latest
   - docker push
$DOCKER_REGISTRY/$CI_PROJECT_NAMESPACE/$SERVICE_NAME:$CI_PIPELINE_ID
   - docker push $DOCKER_REGISTRY/$CI_PROJECT_NAMESPACE/$SERVICE_NAME:latest

deploy_image:
  environment:
    name: ICC-K8s
  stage: deploy
  image: docker-hub.informatik.haw-hamburg.de/icc/kubectl:v1.10.3
  script:
  - sed
"s/$\DOCKER_REGISTRY/$\CI_PROJECT_NAMESPACE/$SERVICE_NAME/$\DOCKER_REGISTRY/
$\CI_PROJECT_NAMESPACE/$SERVICE_NAME:$CI_PIPELINE_ID/g" deploy.yaml>
deploy_new.yaml
  - kubectl apply -f deploy_new.yaml
```

Der sed Befehl bewirkt, dass der Image Name in der deploy.yaml mit dem neuen Namen mit angehängter PIPELINE_ID ersetzt wird. Die PIPELINE_ID dient hier also als Versionszähler. Die neu erzeugte yaml Datei wird dann zum Deployment genutzt, aber **nicht im Repository gespeichert**

Beachten Sie, dass der ***deploy_image*** Job das automatisch erzeugte Environment ***icc-dev*** referenziert! Die Verwendung von kubectl gelingt hier nun ohne weitere Konfiguration, da Gitlab alle erforderlichen Verbindungs- und Authentifikationsparameter via Umgebungsvariablen in das Build Environment injiziert.

Kopieren von Dateien aus einem Gitlab-Build mittels SSH in das eigene Pub Verzeichnis

1. Da später auch Änderungen an Dateien im HOME vorgenommen werden müssen, logt man sich mittels SSH in das HOME ein: ssh aaXXX@ssh.informatik.haw-hamburg.de. Siehe auch https://userdoc.informatik.haw-hamburg.de/doku.php?id=docu:ssh
2. Mittels ssh-keygen wird ein privater und ein öffentlicher Schlüssel erzeugt. Fragen nach einem Passwort werden einfach mit Enter beantwortet. Ein Passwort würde im Build-Prozess zu einer Abfrage führen, die dann den Prozess abbricht. Mit ssh-keygen -f icc.key wird die Schlüsseldatei icc.key und icc.key.pub erzeugt.
3. Der Inhalt der Datei icc.key.pub, der öffentliche Schlüssel, wird nun an die Datei

.ssh/authorized_keys angehängt.

4. In Gitlab wird unter Settings→CI/CD eine Secret-Varaible angelegt mit dem Namen SSH_PRIVATE_KEY. Der Wert der Variablen ist der Inhalt der Datei icc.key inklusive der Begrenzer. Die Variable darf nicht auf Protected gesetzt werden.
5. Füge die folgende Before-Sektion zur Datei .gitlab-ci.yml hinzu:

```
before_script:
  ##
  ## Create the SSH directory and give it the right permissions
  ##
  - mkdir -p ~/.ssh
  - chmod 700 ~/.ssh

  ##
  ## Copy the private key from the Variable into the identity file.
Permisisons
  ## must be set to user-only, otherwise ssh-tools will complanine.
  ##
  - echo "$SSH_PRIVATE_KEY"> ~/.ssh/id_rsa
  - chmod 700 ~/.ssh/id_rsa

  ##
  ## Use ssh-keyscan to scan the keys of your private server.
  ##
  - ssh-keyscan ssh.informatik.haw-hamburg.de>> ~/.ssh/known_hosts
  - chmod 644 ~/.ssh/known_hosts
```

Die Datei kann nun mittels scp kopiert werden, beispielsweise:

```
## Copy file to HOME
  - scp build.zip aaXXX@ssh.informatik.haw-hamburg.de:pub/.
```

Es ist abschließend empfehlenswert beide Schlüsseldateien zu löschen, da der private Schlüssel kein Passwort hat und sowieso in Gitlab vorhanden ist. Der öffentliche Schlüssel ist in der Liste im Pub HOME eingetragen. Falls erforderlich können jederzeit neue Schlüssel erzeugt werden.

E-Mails aus der ICC versenden

Der E-Mail Versand von Diensten aus der ICC ist grundsätzlich möglich! Sie sollten sich dazu zunächst ein explizites E-Mail Konto für Ihren Dienst, Ihr Projekt oder Ihre Gruppe besorgen. Dieses E-Mail Konto können Sie beim Betriebsteam des HAW-Mailers via E-Mail beantragen. Kontaktinformationen finden Sie unter: https://www.haw-hamburg.de/online-services/kontakt.html

Sobald Sie über ein extra Konto mit Zugangsdaten verfügen, sprechen Sie den Mailserver aus Ihrer Applikation wie folgt an:

Feld	Wert
Server	haw-mailer.haw-hamburg.de

Feld	Wert
Port	587
Benutzername	Nutzername Ihres neuen Mailkontos
Passwort	Passwort Ihres neuen Mailkontos
Sicherheit	TLS (starttls)
Authentifikation	PlainAuth

Verwendung von GPUs

Die ICC stellt eine limitierte Anzahl an GPUs zur Verfügung, deshalb sind Interessenten gebeten sich eigenständig mittles Slack, Mattermost oder per Mail an icc@informatik.haw-hamburg.de zu wenden.

Generelle Voraussetzung ist, dass das Container-Image GPU-Unterstützung hat, dazu könnte es zum Beispiel auf den NVIDIA Cuda Ubuntu Images basieren.

Damit ein Container auf eine GPU zugreifen kann, muss zuerst eine *toleration* gesetzt sein, damit der Container die Erlaubnis hat auf einen der GPU-Knoten zu laufen:

```
tolerations:
- key: "gpu"
  operator: "Equal"
  value: "true"
```

Nun muss nur noch ein *resource-limit* gesetzt werden, damit dem Container eine GPU zugeordnet wird (wichtig hierbei ist, dass man keine fraktale einer GPU anfragen kann und *request == limit*):

```
resources:
  limits:
    nvidia.com/gpu: 1
```

Beispiel

Im Beispiel von TensorFlow werden spezielle Basisimages zur Verfügung gestellt (die wiederum auf den NVIDIA Cuda Ubuntu Images basieren), die GPUs innerhalb von Containern unterstützen. Mit folgendem Deployment könnte man nun einen Tensorflow-Container starten, der ein Jupyter Notebook auf Port 8888 hostet:

```
apiVersion: v1
kind: Service
metadata:
  namespace: my-namespace
  labels:
    app: tensorflow
  name: tensorflow
spec:
  ports:
  - name: http
```

```
    port: 8888
    targetPort: 8888
    protocol: TCP
  selector:
    app: tensorflow
---
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: tensorflow
  namespace: my-namespace
spec:
  replicas: 1
  strategy:
    type: Recreate
  template:
    metadata:
      labels:
        app: tensorflow
    spec:
      tolerations:
      - key: gpu
        operator: "Equal"
        value: "true"
      tolerations:
        - key: gpu
          operator: "Equal"
          value: "true"
      containers:
        - name: tensorflow
          image: tensorflow/tensorflow:1.4.1-gpu-py3
          imagePullPolicy: Always
          resources:
            limits:
              nvidia.com/gpu: 1
          ports:
          - containerPort: 8888
            name: http
            protocol: TCP
```

FAQ - Informatik Compute Cloud (ICC)

Ich bekomme eine Fehlermeldung bzgl. unzureichender Rechte. Was kann ich tun?

Prüfen Sie zunächst, ob Sie eingeloggt sind: Login in ICC

Hat dies Ihr Problem nicht gelöst, so versuchen Sie möglicherweise einen Ressourcentyp anzulegen, der in den Standardrollen nicht erlaubt ist. Dazu gehören z.B. DaemonSets, RoleBindings oder auch ServiceAccounts. Eine weitere Möglichkeit ist, dass Sie versuchen einen Pod zu deployen, dessen

SecurityContext den Wert `privileged: true` setzt und somit mit Root Rechten laufen würde.

In beiden Fällen ist diese Ausführung zunächst nicht vorgesehen und ist in den allermeisten Fällen auch nicht notwendig. Wenn Sie sicher wissen, dass Sie ohne erweiterte Berechtigungen nicht weiterkommen, melden Sie sich bitte bei icc@informatik.haw-hamburg.de. Sollten wir Ihrem Wunsch entsprechen können, so erhalten Sie einen ServiceAccount, den Sie an Ihre Deployment-Definitionen schreiben.