This is VS

Im Rahmen des Praktikums für verteilte Systeme im Wintersemester 2015 sollen Sie Server- und Client-Komponenten in Java implementieren. Als Abgabe sollen die Komponenten auf einem Docker-Host ausgeführt werden.

Die Motivation für diese Form der Abgabe ist die Minimierung des Administrationsaufwandes für die Studierenden. Auch Kollisionen zwischen den Teams sollen so gering wie möglich sein. Als Alternative zu einer vollständigen (virtuellen Linux) Maschine erlaubt ein Docker Container eine maximale Automatisierung von Erstellung, Installation und Ausführung eines Netzwerk-Servers.

Die Container sehen sich gegenseitig als eigenständige Maschinen in einem eigens für sie geschaffenem virtuellen Netzwerk. Außerdem agieren die Container isoliert voneinander, so dass Ihre Dienste sich nicht gegenseitig in die Quere kommen.

Vorbereitung

Um Ihre Anwendung in die Docker-Umgebung zu bekommen, müssen ein paar Vorbereitungen getroffen werden. Dies muss in der Regel nur einmal erledigt werden.
Das System nutzt zum Datenaustausch die OwnCloud der HAW. Legen Sie in der Informatik OwnCloud eine Ordner nach dem Muster

```
VSP_<username>
```

an. Im Folgenden wird dieser Ordner als $OC_USERDIR bezeichnet

Geben Sie den Ordner für den User **„VS Docker Robot”** frei.
Wichtig: den User, **nicht** die Gruppe „vs_docker“

Jeder, der selbst etwas unter seinem Benutzernamen in der Dockerumgebung starten möchte, benötigt ein solches geteiltes Verzeichnis.

Container Verzeichnisse

Für jeden Container wird in Ihrem freigegebenen Ordner $OC_USERDIR ein verzeichnis erstellt. Dieses muss dem Template
docker_{nr}
entsprechen. Jedes der erstellten Verzeichnisse wird automatisch erkannt und später angeboten, um einen Container daraus zu bauen.

Beispiel: $OC_USERDIR/docker_0/
abz104/docker_0/

Java Container

Für die von Ihnen, die sich für Java als Entwicklungssprache entschieden haben (oder Derivate wie Scala), haben wir den Prozess zum erzeugen eines Containers vereinfacht und man muss sich gar nicht mit den Details von Docker auseinander setzen.

Ausführbare Jar
Basis Ihrer Anwendung muss für diesen Fall eine ausführbare Jar sein. Diese wird durch den von uns vorbereiteten Container gestartet
Zum Erstellen:

jar cvfe vsp_<username>.jar MyMainClass *.class

Mehr Details unter: http://www.skylit.com/javamethods/faqs/createjar.html

Das richtige Verzeichnis

Für einen Java-Container werden die benötigten Dateien in den Unterordner java eines Container Verzeichnisses gelegt.

Die resultierende Jar-Datei muss also in einen Ordner namens **$OC_USERDIR/docker_0/java** abgelegt werden. Die zu startende Jar muss unbedingt folgender Namenskonvention ensprechen, um automatisch als zu startende Jar erkannt zu werden: **vsp_<username>.jar**.

Abhängigkeiten
Benötigte Bibliotheken müssen ebenso in den **$OC_USERDIR/docker_0/java** Ordner platziert werden. Jede Datei mit der Endung .jar wird automatisch dem Classpath hinzugefügt. Hierbei ist der Name bis auf die Endung irrelevant.

Allgemeiner Container

Entwickelt man nicht unter Java oder benötigt andere Systempakete, oder Java-Versionen, als welche von der automatischen Containererstellung angeboten wird, muss man seinen eigenen Container kreieren.
Hierzu muss ein Docker-File erstellt werden, welches benutzt werden kann, um mittels docker build einen lauffähigen Container zu bauen.

Für einen allgemeinen Container wird ein verzeichnis
„$OC_USERDIR/docker_{nr}/container"
angelegt und die erstellte DockerFile und alle benötigten Anwendungsdaten hinein kopiert.
Hierbei wird der „container"-Ordner als Build-Context verwendet!

Es können nicht sowohl ein Java als auch ein Container Verzeichnis erzeugt werden. In diesem Fall wird das Java-Verzeichnis bevorzugt.
Eine nette Anleitung zum umgehen mit Docker-Files findet sich unter:

- https://www.digitalocean.com/community/tutorials/docker-explained-using-dockerfiles-to-automate-building-of-images

- https://docs.docker.com/userguide/dockerimages/
- https://docs.docker.com/articles/dockerfile_best-practices/

VS-Docker

Zur Verwaltung Ihrer Container, wird eine Nutzeroberfläche zur Verfügung gestellt.
Diese erreichen Sie unter
https://vs-docker.informatik.uni-hamburg.de
bzw. derzeit ohne DNS...
https://141.22.34.15 (Es muss leider das Zertifikat akzeptiert werden so...)
Dort loggen Sie sich mit Ihrer Helios-Kennung ein. Die in der OwnCloud abgelegten Docker-Verzeichnisse werden Ihnen dort angezeigt als baubare Container.
Da mittels Synchronisation gearbeitet wird, kann es zu leichten Verzögerungen kommen zwischen Ablegen einer neuen Datei und der Aktualisierung auf dem vs-docker.

Um einen Container zu bauen, drücken Sie in der Oberfläche bei dem Gewünschten „Buildable“ auf „build“. Hierbei wird automatisch ein Container erstellt und gestartet. Dieser Container trägt automatisch den Namen „<user>_<container>“ und ist unter diesem Namen auch im Netzwerk der Container erreichbar. Außerdem bekommt er eine IP, welche in der Oberfläche angezeigt wird. Die Container sind nicht direkt von außen zu erreichen.

Ihre gestarteten Container werden in der Oberfläche angezeigt und Sie können diese Verwalten, wie z.B. stoppen, neu starten, löschen.

Container Zugriff von außen

Sie können Ihre Services von außerhalb erreichen. Jedoch funktioniert das nur über einen HTTP-Tunnel über den VS-Proxy. Denn generell sind nur wenige Ports verfügbar in dem Netz der HAW. Das schränkt den Einsatz etwas ein, aber alle Ports Ihrer Container können per HTTP angesprochen werden.
Jeder Container ist erreichbar unter

<vsdocker>/cnt/<container ip>/<port>

Direkter Zugriff über VPN

Es existiert ein extra VPN, um direkt auf die Container zugreifen zu können. Hierbei gibt es hinter dem VPN-Gateway keine Einschränkungen der Ports.
Die Konfiguration für den VPN-Zugriff ist Benutzer-Spezifisch und wird Ihnen von der vsdocker-Oberfläche generiert. Dort kann die Konfiguration heruntergeladen werden.

Eigentlich kann diese Konfiguration direkt genutzt werden...
Da leider derzeit das DNS Probleme macht, muss eine Zeile geändert werden:
remote vs-docker2016 443 tcp
ändern Sie bitte zu
remote 141.22.34.15 443 tcp

Danach können Sie sich mit dem VPN verbinden mit
openvpn –config <user>.ovpn

Yellow Pages Service

Ihre Services werden angemeldet bei einem speziellen Service. Dieser Service hat die Container IP 172.18.0.5. Sollte dieser mal dort nicht erreichbar sein, schickt mir direkt eine Mail!
Der Service horcht auf dem Port 4567 und die API für den Service ist in der OwnCloud zu finden.

https://141.22.34.15/cnt/172.18.0.5/4567/services

Der Service erforder eine korrekte Anfrage! Wenn Sie also JSON senden, setzen sie auch den entsprechenden content header!

Disclaimer

Die Art dieser Übungen ist immernoch in der Test-Phase! Es ist mit Schluckauf zu rechnen und alle Vorschläge sind herzlich willkommen!