D 30.11.2018
R IT-Projekt
P Maximilian Falkenstein
Max Schrimpf

Container-Init-System

Changelog

- 13.08.2018 v1alpha1: Initialer Draft, Vorstudie (mf)
- 22.08.2018: Erste Punkte der Evaluation, Anpassung der Soll-Requirements um “Emulation von pseudo-Terminals” (mf)
- 30.11.2018: Aktualisierung des WP, Einarbeitung Feedback (jv), Berücksichtigung, dass Implementation neu durch (jv) erfolgt.

Involvierte Personen

Name	**Bereich**	**Verantwortung**
Maximilian Falkenstein (mf)	VSETH IT	Initialer Draft des Workpackage, initiale Umsetzung mit SCinit
Max Schrimpf (ms)	VSETH IT Projektleiter	Koordination der beteiligten Parteien, Architektur
Jan Veen (jv)	VIS	Feedback auf SCinit, eigene Implementation von ‘cinit’ basierend auf Feedback

Status

- Abgeschlossen

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch

Vorstudie

Jeder Docker-Container startet seitens Docker genau einen Prozess. Grundsätzlich ist es bei Docker best-practice auch nur einen Prozess zu verwenden, da dies verschiedenen Vorteile wie einfachere Skalierbarkeit etc. bringt. Im VSETH sind wir jedoch der Meinung, dass es auch gewisse Nachteile gibt jeden Prozess in einen einzelnen Container zu packen, da so für ein "kleines Python App" schon mindestens ein nginx webserver container und ein python container benötigt werden und somit unnötig die Komplexität an einer Stelle, die mit der eigentlichen Applikation nichts zu tun hat, erhöht wird. Die Skalierbarkeit ist für den VSETH auf Grund der relativ geringen Last und der quasi nicht vorhandenen Lastspitzen kein Argument. Der VSETH plant daher konzeptionell einen Container pro Applikation (bzw. git Repository) und nicht einen Container pro Prozess.

Seit der Adaption einer Containerinfrastruktur im VIS vor ungefähr einem Jahr hat der VIS sich mit allen Problemen auseinandergesetzt, die dieses Work Package anspricht. Bei keinem dieser Probleme hat der VIS eine abschliessende, zufriedenstellende Lösung gefunden.

Jedoch bestehen die meisten Apps, die bspw. bisher im VIS deployed wurden, aus mehr als einem Prozess (und sei es nur dass erst eine Config-Datei geschrieben wird bevor der eigentliche Dienst gestartet wird). Aus diesem Grund muss es einen ersten Hauptprozess geben, der sich darum kümmert, dass alle anderen Prozesse gestartet werden. Ausserdem müsste er als Prozess mit der Nummer 1 nach UNIX-Konventionen besondere Aufgaben, die des sogenannten init-Prozesses (also beispielsweise das Handling von Zombie-Prozessen), wahrnehmen. In der Praxis findet dies allerdings oft in Docker Containern nicht statt (was zu verschiedenen Diskussionen in der Community führt[1] [2] [3]).

Benutzergruppen / Last

Bei einem init-System handelt sich um eine Systemkomponente, die für den Benutzer unsichtbar ist und über keinerlei zentrale Schnittstelle verfügt. Jedoch wird nach der derzeitigen Planung jeder einzelne Container, der auf der SIP betrieben wird, das entsprechende Container-Init System verwenden. Es ist somit als zentrale und hochkritische Komponente anzusehen bei der Fehler (Crashes) fatale Auswirkungen haben können.

Zudem müssen EntwicklerInnen innerhalb des VSETH ein geeignetes Interface besitzen, um Container entsprechend ihrer Vorgaben zu starten. Sie sind somit als die Benutzer des Init-Systems anzusehen.

[1] https://medium.com/@nagarwal/an-init-system-inside-the-docker-container-3821ee233f4b

[2] https://blog.phusion.nl/2015/01/20/docker-and-the-pid-1-zombie-reaping-problem/

[3] https://blog.codebeat.co/docker-has-no-init-and-where-it-got-us-210118c87849

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch

Requirements

Muss-Requirements:

- Starten von verschiedenen Prozessen, was durch die Art, wie SiP Container funktionieren, essentiell ist
- Setzen von Benutzer+Gruppe bei gestarteten Prozessen um möglichst keinen Prozess im Container mit root-Rechten laufen zu lassen, da Container eben doch keine völlige Abschottung bieten und somit Probleme in einzelnen Komponenten potentiell Auswirkungen auf andere besitzen.
- Setzen von Arbeitsverzeichnis und Umgebungsvariablen bei gestarteten Prozessen, da einige Software durch Umgebungsvariablen oder ein Config-File im "aktuellen Verzeichnis" konfiguriert wird.
- Wahrnehmung von Init-Aufgaben (Signal-forwarding und "Zombies" reapen)
- Weiterleitung von stdout/stderr der Kind-Prozesse auf den stdout/stderr des init Prozesses zwecks Statusübersicht der Kind-Prozesse
- Kennzeichnung von Prozessen als abhängig voneinander (A darf nicht vor B starten etc.) um einfach bewirken zu können, dass beispielsweise ein Konfigurationsskript vor der eigentlichen Software ausgeführt wird

Soll-Requirements:

- Setzen von Linux Capabilities bei gestarteten Prozessen um selbst bei Prozessen die normalerweise Root-Rechte benötigen (etwa Webserver, die auf privilegierten Ports (etwa Port 80 HTTP) arbeiten) auf diese verzichten zu können
- Emulation von pseudo-Terminals falls normalerweise Pipes benutzt werden für Software die sonst nicht funktioniert

Marktanalyse

Tool	bash	supervisord	Phusion my_init	Yelp dumb-init	Docker tini	NodeJS (not a joke)
Zombies reapen	Nein	Nein	Ja	Ja	Ja	Nein
Signals forwarden	Ja, mit Einschränkungen	Scheinbar ausser SIGTERM nicht	Ja	Ja	Ja	Nur manche

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch

Start von mehreren Prozessen	Ja, aber nicht gleichzeitig mit 'Signals forwarden'	Ja	Nur extern	Nur extern	Nein	Implizit
Security Features (Benutzer + Gruppe setzen)	Ja, aber nur mit externen tools	Ja, aber Capabilities nur mit externen Tools	Nein	Nein	Nein	Nein
Security Features (Capabilities)	Ja, aber nur mit externen tools	Ja, aber nur mit externen Tools	Nein	Nein	Nein	Nein
Dependencies	Ja	Nein	Nein	Nein	Nein	Implizit

Es stellt sich relativ schnell heraus, dass die meisten Programme, die der VIS bisher an dieser Stelle benutzt hat (bspw. supervisord, simple bash scripts) sowie alle vielversprechenden Tools, die wir während einer Marktanalyse finden können die gewünschten Aufgaben entweder nicht oder nur unvollständig wahrnehmen.

Vorschlag

Es wird ein eine eigene simple implementation eines Init-Systems mit den gewünschten Features entwickelt. Simple containter init (SCInit) wird erst als Proof-of-Concept entwickelt und anschliessend bezüglich Benutzbarkeit, etc getestet. Zur Evaluation sollte es in den VSETH-SiP-Apps benutzt werden.

Evaluationskriterien

Nicht funktionale Requirements

Bei einer Eigenentwicklung sollten an dieses System sehr hohe Anforderungen gestellt werden, die über die beschriebenen funktionalen Anforderungen hinausgehen:

- **Bedienbarkeit**: Es muss Nutzern so einfach wie möglich gemacht werden, mit dem System zu interagieren. Das bedeutet zum einen eine ausführliche, an Beispielen aufgebaute Bedienungsanleitung, zum anderen das Ausgeben von klaren, kurzen und unmissverständlichen Fehlermeldungen.
- **Unterstützung**: Es muss sichergestellt werden, dass Programmfehler im eigenen Code oder Abhängigkeiten zeitnah repariert werden. Das muss über die gesamte erwartete Lebensdauer des Programms sichergestellt sein (dies ist eine Anfroderung, die direkt aus der Entscheidung eine entsprechende Lösung selbst zu entwickeln hervorgeht).
- **Stabilität**: Die den Nutzern des Programms präsentierte Schnittstelle sollte stabil genug sein, dass Änderungen wie im Punkt darüber beschrieben, einfach in die abhängende Applikation übernommen werden können.

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch

- **Wartbarkeit**: Es muss für Neueinsteiger in den Code einfach sein, Erweiterungen und Reparaturarbeiten durchzuführen. Das bedeutet insbesondere, dass durch ausreichende Tests sichergestellt wird, dass Neueinsteiger einfach verifizieren können, ob ihre Änderungen die Qualität des Produkts verschlechtert haben. Darüber hinaus müssen interne Designentscheidungen und die Architektur des Programms in einer Form dokumentiert werden, dass es möglich ist, daraus die Funktionalität des Codes abzuleiten und der Code ein leicht zu erkennender Spiegel dieser Architektur ist.

Evaluationsablauf.

- Test des Init-Systems mit verschiedenen VSETH-Applikationen
- Einholen von Feedback beim technischen Lenkungsgremium des IT-Projekts / Verwendung im VIS für Feedback von Dritten
- Gibt es signifikante Nachteile gegenüber der Lösung, dass man mit mehreren Containern arbeitet?

Sign-Off

An verschiedenen IT-Executive Meetings wurde das Thema diskutiert und anschliessend ein Sign of für 24h - 32h Entwicklungsaufwand des Prototyps gegeben.

Evaluation

Zunächst wurde ein Proof-of-Concept entwickelt, der alle oben aufgeführten Muss-Requirements erfüllt, und auf GitHub gestellt. Anschliessend wurde der SiP Container für das Helfertool der TU München als erstes damit versehen. Das Helfertool selbst ist eine normale Django-Applikation, im Container müssen deswegen nginx, uwsgi und celery ausgeführt werden. Es stellte sich dabei heraus, dass für diese Konstellation bereits die beiden 'Soll-Features' (Capabilities und Dependencies) erforderlich waren:

- **Capabilities**: nginx kann aus Sicherheitsgründen als nicht-Root-User gestartet werden
- **Dependencies**: Vor dem Start des Helfertools muss ein Konfigurationsfile generiert werden, d.h. Der Prozess der das Konfigurationsfile generiert muss vor den anderen Prozess ausgeführt werden.

Ausserdem fiel auf, das nginx access und error Logs nur in Dateien oder pseudo-Terminals (PTYs) ausgeben kann, nicht jedoch in Pipes[4] . Die ursprüngliche Implementation von SCInit hat aufgrund der einfacheren Handhabung Pipes benutzt um stdout und stderr von Kindprozessen abzufangen, deshalb musste SCInit darüber hinaus noch um eine einfach Terminalemulation erweitert werden, die Soll-Requirements mussten damit entsprechend angepasst werden

Das Helfertool mit diesen Anpassungen und SCInit lief über mehrere Monate stabil.

[4] Erklärung Unterschied Pipes und Pseudo-Terminals: Stackoverflow

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch

Feedback - Evaluation im VIS

Da der VIS bereits eine Container-Infrastruktur besitzt und verwendet wurde SCinit dem VIS zum internen Test übergeben. Dabei ergab sich folgendes Feedback von (jv):

Grundsätzlich war die Portierung auf scinit einfach und es passte sich gut in die bestehende Infrastruktur ein.

Grössere Probleme bereitete das Aufsetzen der Build Pipeline. Der Build Prozess von scinit ist nicht ausreichend dokumentiert und Abhängigkeiten unsauber beschrieben. Generell wäre es wünschenswert, scinit als Distributionspaket anzubieten, um diese Probleme zu vermeiden. Konkrete Probleme sind stichpunktartig gelistet:

- Build steps are not complete (submodule initialisation)
- Default build type is Debug
- Runtime dependencies are not described
- APT packaging appreciated

Die Evaluation hat ausserdem weitere konkrete Wünsche ergeben, die das System erfüllen könnte:

- Bug: Not all lines are prefixed by scinit
- Setting a working directory for a process
- Feature to forward the entire environment
- Feature to give all privileges
- Restart non-terminating processes when crashed (debugging)
- Shutdown container if any exit status != 0

Die Dokumentation für Entwickler und Integratoren ist dürftig. Hier besteht im Falle einer Umsetzung Verbesserungsbedarf (Liste nicht abschliessend):

- process dependencies should be documented to take lists of other programs
- note on quoting rules for yaml
- args recommendation: Every space splitting to one list entry
- Explain env whitelist
- Explain capabilities

Generell möchte ich aber betonen, dass scinit viele wichtiges Problem beim Betreiben einer Containerinfrastruktur erkannt hat und ein vielversprechender Kandidat ist, um diese Probleme zu lösen. Ich empfehle daher das weitere Verfolgen dieses Vorschlags.

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch

Mögliche Weitere Features

Im Rahmen der Umsetzung des Proof of Concepts wurde offensichtlich, dass folgende Features für die Weiterführung des Projekts und die einfachere Handhabung durch Entwickler noch hilfreich wären:

Da es sich bei SCinit um eine Eigenentwicklung handelt können diese als teil der Umsetzung bei entsprechendem Feedback implementiert werden.

- Integration mit Kubernetes Security Policies: SCInit weiss ob ein Container Root-Rechte braucht und was für Capabilities gefordert werden. Prinzipiell könnte man dies Kubernetes mitteilen, sodass der Container clusterseitig auf die minimal erforderlichen Privilegien eingeschränkt wird
- Status-Schnittstelle: Ein einfaches Unix-Socket das es ermöglicht den aktuellen Status der Kindprozesse abzufragen. Damit könnte man dann bspw. einen einfachen Health-Check implementieren der nur healthy ist wenn es auch alle Apps im Container sind
- Support für regelmässige Aufgaben (Cron-Jobs). Einfache Konfigurierbarkeit und Status-Abfrage wären enorm hilfreich. Ausserdem wäre es hilfreich, wenn logs entsprechender Jobs einfach einsehbar wären (mit den Logs der anderen Prozesse)

Umsetzung

Das Projekt wurde im VSETH grösstenteils als positiv aufgenommen. Im VIS stellte sich auch klar das Bedürfnis für ein entsprechendes init-System.

Da das Projekt allerdings aufgrund anderer Prioritäten im VSETH über längere Zeit nicht weiterverfolgt wurde ergab sich die Situation, dass (jv) bereits mit einer abweichenden Implementation der gleichen Requirements begonnen hatte.

Um keine divergierenden Implementation bereits vor der Einführung des init Systems zu besitzen wurde anschliessend beschlossen nur die VIS-Version von (jv) weiter zu verfolgen.

VSETH — CAB E 27 — Universitätstrasse 6 — 8092 Zürich — +41 44 632 42 98 — vseth@vseth.ethz.ch