Remote Processing V1

Versionen

Datum	Notiz	Version
31.12.2019	Erste Veröffentlichung	1.0

Autoren

Name	Institution	Email
Fabian Bruckner	Fraunhofer ISST	fabian.bruckner@isst.fraunhofer.de

Motivation für Remote Processing

D° erlaubt es Entwicklern auf einfache Art und Weise und mit geringem Overhead datenverarbeitende Applikationen (sogenannte Data Apps) mit integrierten Usage Control Mechanismen zu entwickeln. Hierdurch wird ein Beitrag zur Wahrung und/oder Erlangung der eigenen digitalen Souveränität geleistet. Die so entwickelten Applikationen können in entsprechende Arbeitsabläufe integriert werden und setzten die integrierten Policies automatisiert durch. Ein Aspekt der dem souveränen Umgang mit den eigenen Daten in solchen Abläufen sehr häufig entgegenwirkt sei mit einem kleinen Beispiel umrissen:

Zwei Entitäten (beispielsweise Geschäftspartner) einigen sich auf den (unidirektionalen) Austausch von Daten, um beliebige Mehrwerte zu erzeugen. Außerdem einigen sich die beiden Parteien darüber, wie die geteilten Daten genutzt und verarbeitet werden dürfen. Anschließend wird der entsprechend vereinbarte Arbeitsablauf eingerichtet und der Partner, welcher über die Daten verfügt, sendet seine Daten an die andere Partei welche die Verarbeitung vornimmt, um daraus die vereinbarten Mehrwerte zu generieren.

Für die Entität, welche die Daten bereitgestellt hat, stellt sich nun die Frage, wie sichergestellt werden kann, dass die geteilten Daten nur wie vereinbart genutzt werden. Zum einen kann der Partner darauf hoffen, dass sein Gegenüber gemäß Treu und Glauben handelt und Vereinbarungen, sowie geschlossene Verträge einhält. Zusätzlich können technische Mechanismen verwendet werden, welche die korrekte Datennutzung sicherstellen sollen (beispielsweise D°). Dennoch ergibt sich das Problem, dass die zu schützenden Daten die eigenen Systeme verlassen haben und an einen Dritten übermittelt wurden. Hierdurch geht die Kontrolle an den eigenen Daten verloren und die digitale Souveränität ist zumindest gefährdet.

Um dieser Problematik entgegenzuwirken verfügt D° über eine Remote Processing Technologie. Diese kehrt die Richtung des Datenflusses um und entfernt die Notwendigkeit die eigenen Daten zur Verarbeitung an Dritte zu übermitteln. Stattdessen wird die datenverarbeitende Applikation an die Partei, welche über die Daten verfügt, gesendet. Anschließend werden die Data App auf seinen Systemen ausführt. Somit ist es zu keinem Zeitpunkt notwendig, dass die Daten die eigenen System verlassen was sehr förderlich für die Souveränität an den eigenen Daten ist.

Kompilat vs. Programmcode

Die (sowohl technisch als auch organisatorisch) einfachste Möglichkeit zur Umsetzung des Remote Processing besteht darin, die ausführbaren Applikationen direkt zu versenden. Der Datenbesitzter nimmt die Data App entgegen, führt diese aus und sendet die berechneten Ergebnisse zurück oder erlaubt der Gegenstelle das Abrufen der Ergebnisse. Problematisch an diesem Vorgehen ist, dass die ausführbare Data App dem Ausführenden quasi als Black Box zur Verfügung gestellt wird. In der Data App können sich, neben der vereinbarten Datenverarbeitung, beliebige Funktionalitäten verbergen, welche im schlimmsten Fall das ausführende System korrumpieren können. Somit können sich schlimmere Folgen ergeben, als durch die Herausgabe der Daten an Dritte, wie in der Ausgangssituation beschrieben.

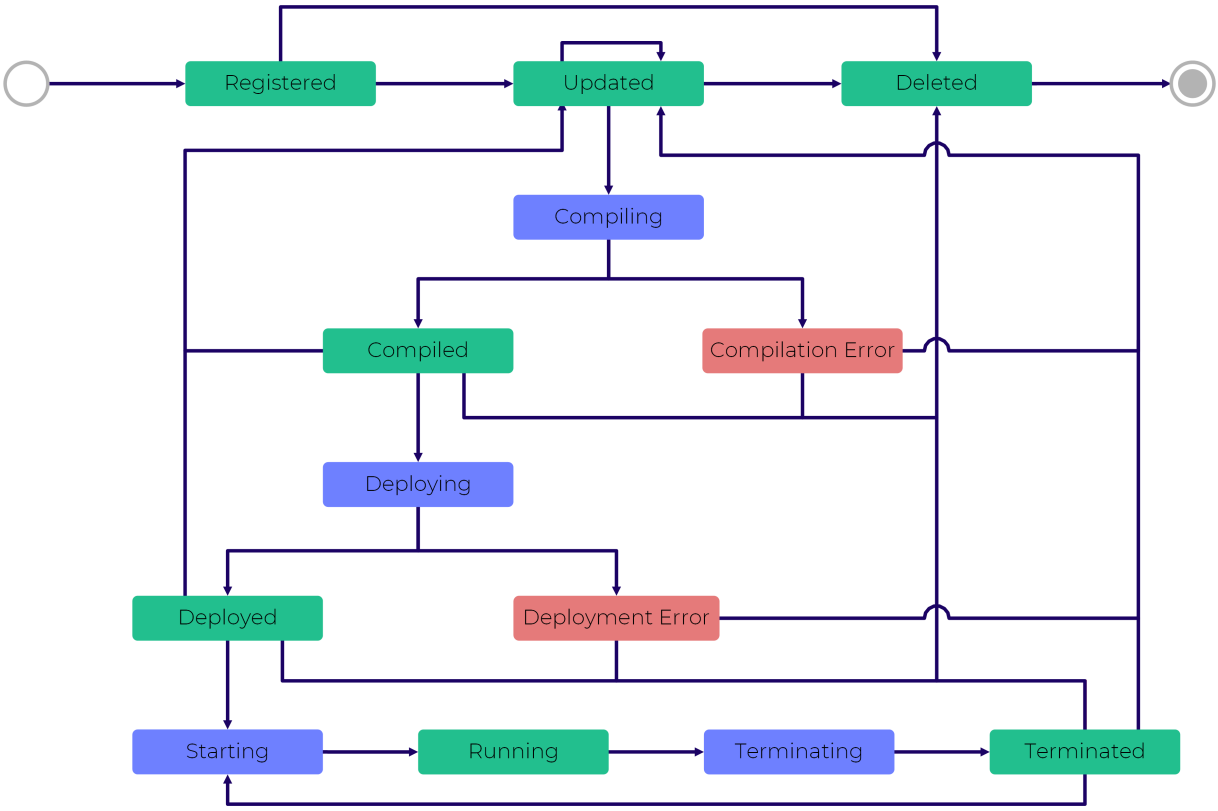
Aus diesem Grund arbeitet die Remote Processing Technologie von D° ausschließlich mit Programmcode. Statt dem fertigen Kompilat wird dem Ausführer der Data App der Speicherort des Programmcode Repositories mitgeteilt. Der Ausführer beschafft sich eine Kopie dieses Repositories und übersetzt selber eine ausführbare Data App aus den bezogenen Quellen. Hierdurch besteht für den Ausführenden zumindest die theoretische Möglichkeit die Applikation auf ihr Verhalten hin zu überprüfen, bevor eine Übersetzung und Ausführung vorgenommen wird.

Durch dieses Vorgehen wird dem Datenbesitzter die Möglichkeit gegeben qualifizierte Entscheidungen darüber zu treffen, welche Data Apps Zugriff auf seine Daten erhalten. Dies ist ein wichtiger Aspekt für die Souveränität an den eigenen Daten.

Prozessbeschreibung

Im Kontext des Remote Processings durchlaufen die verwendeten Data Apps diverse Zustände. Diese Zustände sowie deren Übergänge werden nachfolgend erläutert.

In der nachfolgenden Abbildungen sind die Zustände sowie die möglichen Zustandsübergänge als Zustandsdiagramm gezeigt. Blaue Knoten sind solche, die nach einer bestimmten Zeit oder dem eintreten eines Ereignisses automatisch verlassen werden. Der Anwender kann nur das Betreten dieser Zustände verursachen, kann aber keinen Einfluss darauf nehmen, wann der Zustand wieder verlassen wird. Grüne Zustände können nur durch Ereignisse, die der Anwender auslöst, verlassen werden. Fehlerzustände werden über rote Knoten gekennzeichnet, welche semantisch gleich zu den grünen Zuständen sind.



Registriert

Die Registrierung einer Data App ist der erste Schritt, um eine Data App für die Verwendung mit Remote Processing verfügbar zu machen. Es muss dem Remote Processing Prozess mitgeteilt werden, wo der Programmcode für die Data App bezogen werden kann und welche Anmeldedaten hierfür notwendig sind. Im aktuellen Entwicklungsstand werden ausschließlich git-Repositories unterstützt.

Der Remote Processing Prozess erzeugt zunächst eine UUID, welche bei weiteren Anfragen zur Identifikation der Data App verwendet wird. Anschließend wird das angegebene Repository geklont, die Adresse des Repositories persistiert und der aktualisierte Zustand des Remote Processing Servers im eigenen git-Repository abgelegt. Dabei werden die Metadaten des geklonten Data App Repositories vorher gelöscht, um den Speicherplatzbedarf zu reduzieren. Die zugewiesene UUID wird an den Aufrufer zurückgegeben und bei allen nachfolgenden Anfragen zur Identifikation verwendet.

Hinweis

Die Anmeldedaten, die an den Remote Processing Server gesendet werden, werden weder persistiert, noch in den Log geschrieben.

Aktualisiert

Unter der Angabe der zuvor vergebenen UUID und der Anmeldedaten für das dazugehörige Repository kann der Programmcode einer Data App aktualisiert werden. Dabei werden auch eventuell vorhandene Artefakte der Containervirtualisierung (Images und Container) entfernt, da diese sich auf alte Versionen beziehen.

Nach dem Update wird der geänderte Zustand des Remote Processings in das git-Repository geschrieben.

Gelöscht

Sobald eine Data App nicht mehr für das Remote Processing verwendet werden soll ist es sinnvoll sie vom Remote Processing Server zu löschen. Dabei werden sowohl der Programmcode und Metadaten (bspw. Position des Programmcode-Repositories), als auch Artefakte der Containervirtualisierung vom System entfernt.

Wird dieser Zustand von einer Data App erreicht, ist dies nicht mehr zu ändern. Selbst wenn die Data App erneut verwendet werden soll, ist es notwendig sie neu zu registrieren und im Folgenden die neu zugewiesene UUID zu verwenden.

In Übersetzung

Sobald der Übersetzungsvorgang der Data App gestartet wird, wird der D°-Compiler mit dem Inhalt des geklonten Respositories aufgerufen. Dies findet asynchron statt um die Responsivität des Systems zu erhöhen.

Hieraus ergibt sich, dass das Repository welches die zu bauende Data App enthält alle notwendigen Ressourcen enthalten muss und das diese im Wurzelverzeichnis des Repositories liegen.

Bedingt durch den Aufbau des D°-Compilers werden aktuell alle Übersetzungsanfragen sequentialisiert.

Übersetzt

Wird die Übersetzung des Data App Repositories ohne Fehlermeldung beendet geht die Data App über in den Übersetzt Zustand. Von diesem Zustand aus kann die Data App bereitgestellt werden.

Übersetzungsfehler

Falls bei der Übersetzung der Data App ein Fehler auftritt geht die Data App in diesen Zustand über.

Im aktuellen Entwicklungsstand von D° ist es nicht möglich die Fehlermeldung abzufragen. Diese ist nur in den Log-Ausgaben des Remote Processing Servers zu finden.

In Bereitstellung

Bevor eine erfolgreich übersetzte Data App gestartet werden kann, ist es notwendig diese mittels Containervirtualisierung vom ausführenden System zu isolieren. Die Remote Processing Technologie in D° verwendet zu diesem Zweck Docker.

Dabei wird die Bereitstellung durch den Anwender angestoßen und findet anschließend asynchron im Hintergrund statt. Der Remote Processing Server kann mehrere Data Apps gleichzeitig in entsprechende Docker-Images verpacken. Es findet somit zunächst keine Sequentialisierung statt.

Der Betreiber des Remote Processing kann konfigurieren, wie viele Bereitstellungs-Prozesse gleichzeitig ausgeführt werden können, bevor die Anfragen Sequentialisiert werden.

Sobald der Bereitstellungsprozess tatsächlich ausgeführt wird, sammelt der Remote Processing Server die notwendigen Dateien (Dockerfile, ausführbare Data App, ...) ein und erzeugt ein temporäres Verzeichnis für die Dauer des Prozesses.

Bereitgestellt

Ist die Erzeugung des Images erfolgreich geht die Data App in diesen Zustand über.

Bereitstellungsfehler

Wenn bei der Erzeugung des Docker Images ein Fehler auftritt geht die Data App in diesen Zustand über.

Im aktuellen Entwicklungsstand von D° ist es nicht möglich die Fehlermeldung abzufragen. Diese ist nur in den Log-Ausgaben des Remote Processing Servers zu finden.

Hochfahren

Nach erfolgreicher Bereitstellung der Data App kann sie gestartet werden. Dabei wird aus dem Docker Image ein Container erzeugt und hochgefahren.

Sofern die Data App über HTTP-Schnittstellen angesprochen wird, wird dem Container ein zufälliger Port zugewiesen, über den die Kommunikation erfolgt.

Da dem hochgefahrenen Container im aktuellen Entwicklungsstand nicht zu entnehmen ist, wie weit der Startprozess der enthaltenen Data App fortgeschritten ist, wird der Zustand der Data App aktuell nach einem festen Zeitintervall in den nächsten Zustand überführt.

In Ausführung

Nach dem Hochfahren eines Data App Containers geht die dazugehörige Data App nach einem festen Zeitintervall in diesen Zustand über.

Herunterfahren

Wird eine Data App die sich aktuell in Ausführung befindet, nicht länger benötigt ist es sinnvoll den entsprechenden Container zu beenden. Dieser Prozess wird durch den Benutzer angestoßen und findet anschließend asynchron im Hintergrund statt.

Da Data Apps im aktuellen Zustand keine Information darüber geben, ob sie vollständig terminiert sind, wird in diesem Zustand eine feste Zeit verblieben bevor die Data App in den Terminiert Zustand übergeht.

Terminiert

Data Apps gehen automatisiert in diesen Zustand über, sobald dem Docker Container ausreichend Zeit gegeben wurde, um ordnungsgemäß heruntergefahren zu werden.

Unbekannter Zustand

Falls beim Start des Remote Processing Servers bereits Data Apps im System vorliegen, aber aus beliebigen Gründen die Metadaten nicht geladen werden können, wird die Data App in diesen Zustand versetzt. Von diesem Zustand aus ist es notwendig die Data App zu aktualisieren, um die Data App wieder in einen definierten Zustand zu überführen.

Unbekannte Data App

Werden Anfragen mit einer unbekannten UUID gestellt, welche keiner Data App zugeordnet werden kann, wird dieser Pseudo-Zustand verwendet.

Allgemeiner Aufbau

In der nachfolgenden Abbildung ist der Aufbau des Remote Processing Servers schematisch dargestellt. Es ist erkennbar, dass das sich die Komponenten des Remote Processing in fünf Gruppen einteilen lassen.

D°-Compiler

Der D°-Compiler ist in seiner jeweils aktuellen Form ein Modul, welches im Remote Processing zur Übersetzung der Data Apps verwendet wird.

Git Repository

Das integrierte git Repository des Remote Processing ist der zentrale Speicherort für die Resourcen aller Data Apps und notwendiger Metadaten. Dieses Repository kann darüber hinaus auf einem (externen) git Server abgelegt werden.

Durch die Verwendung eines git Repositories zur Datenspeicherung ist es möglich nachträglich Aussagen über den Zustand einzelner Data Apps oder des Remote Processing als Ganzes zu einem bestimmten Zeitpunkt zu treffen.

Es muss dabei beachtet werden, dass kein besonderes Augenmerk auf den Schutz der git-History gelegt wird. Sofern dies notwendig ist, muss die Integrität dieser Daten anderweitig auf dem entsprechendem git Server sichergestellt werden.

Docker Host

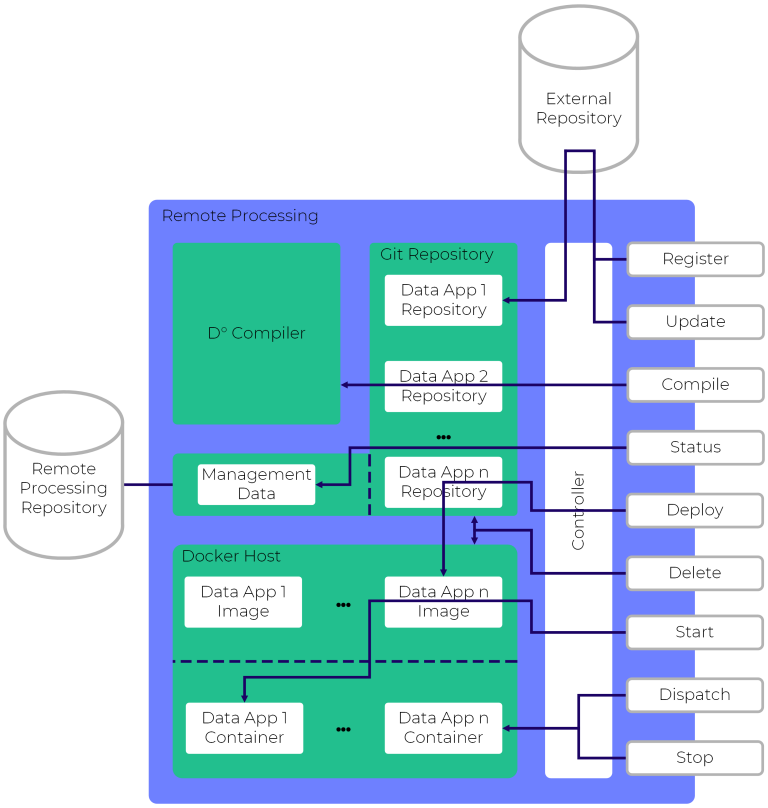
Der Docker Host, welche die Docker Container und Images erzeugt, abspeichert und ausführt, kann auf der selben Maschine wie der Remote Processing Server laufen. Es ist aber ebenso Möglich hierfür eine andere Maschine zu verwenden. Es ist nur in jedem Fall notwendig, dass die Docker API auf dem Docker Host aktiviert ist und die Maschine durch den Remote Processing Server erreicht werden kann.

Controller

Der Controller nimmt alle Anfragen der Nutzer über die Schnittstelle entgegen und leitet sie an die entsprechenden Komponenten weiter.

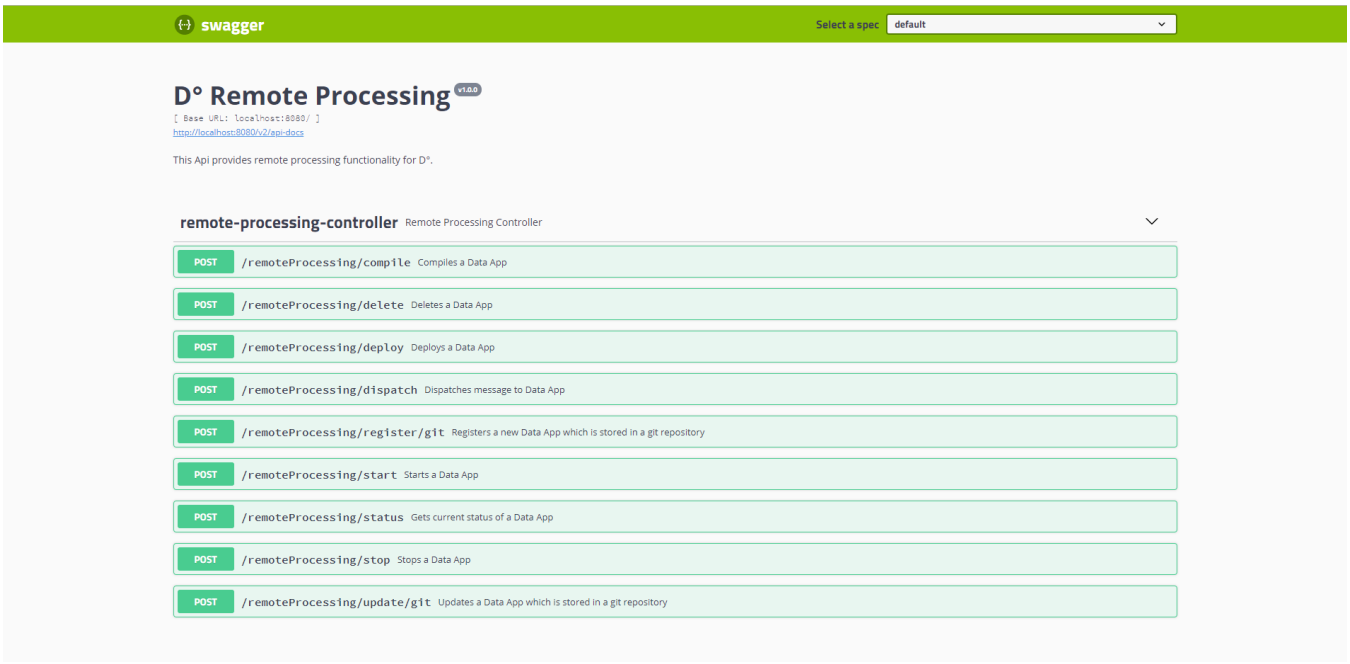
Schnittstelle

Die Schnittstelle des Remote Processing ist eine HTTP-Api, welche sowohl serialisierte JSON-Objekte entgegennimmt und auch zurückgibt. Die einzelnen Endpunkte der Schnittstelle sind in der Abbildung als Rechtecke im rechten Bereich zu sehen. Die ausgehenden Pfeile illustrieren die ausgelösten Abläufe und laufen durch die betroffenen Elemente.



An dieser Stelle wird aus Gründen des Umfangs keine detaillierte Spezifikation der API vorgenommen, welche hochgradig redundant zu den vorherigen Informationen wäre. Stattdessen sei auf die Swagger Oberfläche des Remote Processing Servers verwiesen. Diese ist unter dem Pfad `/swagger-ui.html` erreichbar und enthält Informationen über alle Endpunkte der Remote Processing API.

Der nachfolgende Screenshot zeigt die Swagger Oberfläche, auf der alle Endpunkte zu sehen sind.


swagger
Select a spec
default
D° Remote Processing
v1.0.0
[Base URL: localhost:8080/]
http://localhost:8080/v2/api-docs
This Api provides remote processing functionality for D°.
remote-processing-controller
Remote Processing Controller
POST /remoteProcessing/compile Compiles a Data App
POST /remoteProcessing/delete Deletes a Data App
POST /remoteProcessing/deploy Deploys a Data App
POST /remoteProcessing/dispatch Dispatches message to Data App
POST /remoteProcessing/register/git Registers a new Data App which is stored in a git repository
POST /remoteProcessing/start Starts a Data App
POST /remoteProcessing/status Gets current status of a Data App
POST /remoteProcessing/stop Stops a Data App
POST /remoteProcessing/update/git Updates a Data App which is stored in a git repository