Proposal

Speicherung von Informationen zu Release-Artefakten

Zielsetzung

Immer wieder tauchen Fragen zu installierten Versionen auf. Zum Beispiel:

- Ist Version x.y.z eines Produkts freigegeben?
- Welche Service-/Programmversionen sind in Produkt x.y.z enthalten?
- In welchen Versionen eines Produkts ist Service/Programm x.y.z enthalten?
- In welchen Produkten wird aktuell noch ein Modul/Bibliothek mit Version x.y.z verwendet?
- Entsprechen die Lizenzen aller verwendeten Programme/Module/Bibliotheken unseren Richtlinien?
- In welche Version eines Services/Produkts ist eine Codeänderung eingeflossen?

Es soll möglich werden, diese auf effiziente Weise zu beantworten.

Ausgangszustand

Einen Teil dieser Fragen kann bisher mittels Auswertung des Master-Verzeichnisbaumes ermittelt werden. Ein Teil (z.B. Lizenzen von Modulen/Bibliotheken) können nur manuell geprüft werden. Insbesondere Codeänderungen können nicht durchgehend bis zum fertigen Produkt nachvollzogen werden.

Der neue Entwicklungsprozess bietet die Möglichkeit, alle notwendigen Daten automatisiert zu ermitteln. Allerdings bedarf es einer effizienten Möglichkeit zur Speicherung und Abfrage dieser Daten. Wenn man die bis jetzt vorhandenen Einschränkungen beseitigen will, ist die Auswertung der Daten on Demand nicht mehr möglich, da diese zu umfangreich und verteilt sind.

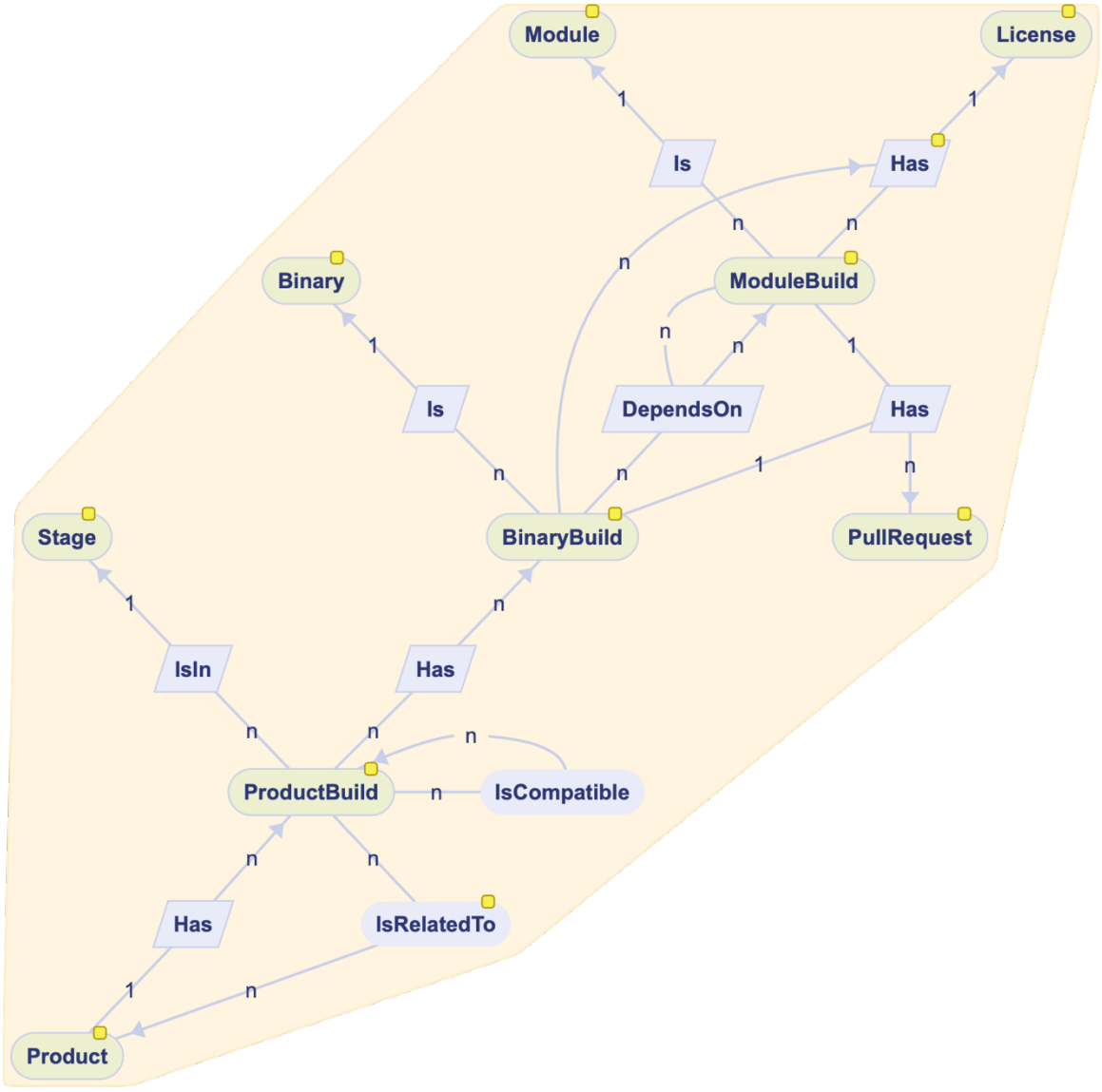
Zielzustand

Mittels einer Graph-Datenbank werden alle benötigten Informationen gespeichert und miteinander verknüpft. Die Speicherung erfolgt dabei als Teil des Buildprozesses, da dann sowieso alle Informationen verfügbar sind. Alte Daten können über entsprechende Skripte aus unseren Repositories gezogen und in die Datenbank übertragen werden.

Die Datenbank kann mittels entsprechender Queries abgefragt werden. Dies kann direkt geschehen oder über Interfaces (z.B. Robbie/Hubot), um eine einfache Verwendung zu ermöglichen. Dafür soll die Datenbank über eine entsprechendes REST-API angesprochen werden können.

Auch Änderungen der Daten (z.B. Hinzufügen neuer Daten im Laufe eines Builds) erfolgen über diese REST-API. Ein Interface ist hier nicht oder nur für einzelne Aufgaben notwendig (z.B. manuelle Bestätigung einer Lizenzzuordnung, wenn die automatische Zuordnung nicht sicher erfolgen konnte).

Der folgende Graph wurde größtenteils im Rahmen eines Tests am P5-Projekt realisiert. Die Knoten Product, Stage, Binary, Module, License enthalten das Attribut „Name“. ProductBuild, BinaryBuild, ModuleBuild enthalten zusätzlich noch Versionsinformationen. PullRequest enthält einen Link auf GitHub.



Mit diesem Graph können alle oben genannten Fragen beantwortet werden. Im Test mit allen verfügbaren historischen Daten zum PLOSSYS 5 Produkt lagen die Antwortzeiten maximal bei

einigen Sekunden. Eine Optimierung der Datenstrukturen und entsprechende Indizierung sollten diese guten Zahlen in den meisten Fällen noch drastisch reduzieren.

Erste Maßnahmen

- Konzept für Betrieb einer hochverfügbaren Datenbank: Es muss geklärt werden, wie die Graph-Datenbank aufgesetzt werden kann, so dass die Buildprozesse sie jederzeit sicher aus der Cloud ansprechen können. Insbesondere ein Konzept für Backup und Restore ist notwendig, da diese Daten eine Single Source of Truth für unsere Entwicklung sein wird.
- Einbindung in unsere Buildprozesse: Die Buildpipeline muss über die REST-API der Datenbank neue Daten hinzufügen.
- Einfaches Abfrage-Interface: Für den Anfang wäre eine Erweiterung von Robbie um entsprechende Fragen (z.B. „What services are included in product XYZ?“) wohl die schnellste Methode.