# Caching Strategie für Bitbucket API-Abfragen

|  |  |
| --- | --- |
| **Fragestellung** | **Wie sollen API-Abfragen auf Bitbucket (code.sbb.ch) im Controller optimal gecached werden?**  Der ESTA Tekton Controller benötigt vielfältige Informationen über Repositories von Bitbucket, welche über dessen API abgefragt werden. Es werden häufig dieselben Anfragen gemacht oder mehrere API-Requests hintereinander, um komplexe Informationen, wie z.B. ob ein Repository «tekton-enabled» ist, zu gewinnen. Die teilweise langen Antwortzeiten von Bitbucket, insbesondere bei kumulierten Abfragen und die potenziell hohe Last, die der ESTA Tekton Controller auf Bitbucket damit verursacht, legen ein Caching seitens Controller nahe. Dieses wurde bereits zu Beginn der Entwicklung sehr rudimentär eingebaut und soll nun verfeinert werden. Konkret stellt sich die Frage nach den richtigen Werten für Vorhaltezeit, Grösse/Limits und Eviction Strategie. |
| **Rahmenbedingung** | * Caching ist bereits vorhanden aber "schlecht" oder "falsch" eingesetzt |
| **Annahmen** | * Die Performance des Bitbucket APIs bleibt konstant |
| **Alternativen / Varianten** | Variante 0 (status quo) Die aktuelle Cache-Strategie sieht ein statisches Caching mit einer automatisierten Eviction/Refresh alle 10 Minuten via Scheduler für alle gecachten Ressourcen. Das Caching von Multi-Resultaten ist nicht korrekt implementiert (Quarkus Cache kann kein Multi).   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Einfach | Nicht sehr effektiv | | Out-of-sync für max. 10 Minuten | Inkonstante Antwortzeiten | |  | Multis (Repo-Listing, Branches) nicht richtig gecached |  Variante 2 Die Caching Lifetime resp. Eviction-Strategie soll für die einzelnen Ressourcen individuell definiert und implementiert werden. Die von Bitbucket empfangenen Webhooks sollen als Informationsquelle für die Invalidierung und den allfälligen Neuaufbau eines Caches herangezogen werden. Über die Webhooks wird der Controller proaktiv über Änderungen an einem bestimmten Repo informiert. So kann z.B. der Cache für die Branches oder Tags eines Repos länger vorgehalten und erst bei einem Update von Bitbucket invalidiert werden. Um das Risiko von verpassten Webhooks zu vermindern, können sämtliche Caches periodisch (z.B. 1x täglich) invalidiert und neu aufgebaut werden.  Gleichzeit soll das Caching von Listen (tekton-enabled Repositories, Branches, Tags) korrekt mit den Quarkus Caching Mittlen refactored werden.   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Optimierte Vorhaltezeit | Möglicher out-of-sync für eine limitierte Zeit | | Konstant schnelle Antwortzeiten | Komplexere Invalidierungslogik | |  | Quarkus 2.7.x Update notwendig | |
| **Entscheidung** | **Variante 2** |
| **Begründung** | Beim Caching besteht eindeutig Handlungs- und Optimierungsbedarf. Die Verwendung von ESTA Tekton und somit auch die Last auf das Bitbucket API wird zukünftig sicher zunehmen, so dass ein  optimiertes Caching als sinnvoll und notwendig erachtet wird. |
| **Wer** | [Wallrapp Manuel (IT-PTR-EXT-EXT2 - Extern)](file:///C:\display\~ue64007) [Brüderli Thomas (IT-PTR-CEN2-SL2)](file:///C:\display\~u233658) |
| **Wann** | 26.01.2022 |