# Shared Caches für Tekton Pipelines

|  |  |
| --- | --- |
| **Fragestellung** | Wie sollten geteilte und persistente Caches für Tekton Pipelines aufgebaut/konfiguriert werden, damit ein Optimum an Performance, Disk/Volume Space und Parallelität erreicht werden kann.  Build Pipelines für Maven, NPM und andere Technologien, welche Dependencies herunterladen müssen, ist ein Cache in Form eines Persistent Volumes notwendig, um die Build-Performance zu optimieren und damit nicht bei jedem Build alle Dependencies neu geladen werden müssen. Viele Projekte haben eine gemeinsame Schnittmenge von Dependencies und somit ist es sinnvoll den Cache (z.B. Maven) für alle Java-Projekte zu teilen. Das ist aktuell so umgesetzt mit einem PVC "shared-XXX-cache", welches für Pipelines als Workspace gemountet wird. Auf Openshift (AWS) steht aktuell nur ReadWriteOnce Storage Class für PVCs zur Verfügung und diese hat [gewisse Einschränkungen](https://kubernetes.io/docs/concepts/storage/persistent-volumes/#access-modes) für den gleichzeitigen Zugriff von parallel laufenden Pipelines: "the volume can be mounted as read-write by a single node". Wollen also mehrere Pipelines (Task Pods) dieses Volume read-write mounten, muss Pod2 warten, bis Pod1 terminiert, was potenziell zu Verzögerungen in den Build Pipelines führt.  Es gilt nun weitere Möglichkeiten zur Optimierung der Caching-Architektur zu prüfen und abzuwägen. [ESTA-5103](https://flow.sbb.ch/browse/ESTA-5103?src=confmacro) - Der Jira-Vorgang existiert nicht oder Sie sind nicht anzeigeberechtigt. |
| **Rahmenbedingung** | * Tekton läuft auf Openshift auf AWS Clustern * Es steht keine Storage Class von Typ ReadWriteMany zur Verfügung |
| **Annahmen** | * Pro Batch Node können 8 Build Pods parallel laufen (allocatable: 16 CPU / 24Gi Memory; Pod limit: 2 CPU / 3 Gi) * Bei hoher Last muss davon ausgegangen werden, dass Pipeline Tasks (Pods) auf unterschiedlichen Nodes gestartet werden. * Die Grösse eines Projekt-Caches liegt zwischen wenigen MB und 2 GB (Ø 500M) * Die verfügbare Ephemeral Storage Size pro Node ist relativ klein (~1GB) * Kosten für Openshift Block Storage (PVC) gemäss [CON Preisliste](https://confluence.sbb.ch/display/CLEW/Costs+of+resources+on+Openshift+platform) (SATA non-prod): CHF 2.91 / GB / Monat * Für S3 Datentransfer fallen [keine Kosten an](https://aws.amazon.com/s3/pricing/), da "intern" von S3 zu einem anderen AWS Service (EC2) transferiert wird   Einfache Analyse eines Maven Caches (ESTA Tekton Controller): du -sh /workspace/cache: **211M** tar czf cache-dump.tgz /workspace/cache/: **166M / 5.717s** Analyse eines NPM Caches (ESTA Tekton UI):du -sh /workspace/cache**: 121M** tar czf npm-cache-dump.tgz /workspace/cache: **112M / 3.765s** |
| **Alternativen / Varianten** | Variante 1 **Ein geteilter Cache für alle Projekte eines Build Stacks (Status quo)**  Konzept: ein gesharter Cache pro Build Namespace (ähnlich wie auf einem lokalen Computer)   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Keine Änderungen erforderlich | Blockiert parallel laufende Pipelines (auf mehreren Nodes) | | Keine redundanten Daten | Grosses Cache Volume | | Schnelles Cache Warming | Cleanup: wipe all |   **Processing Overhead**: 0s  **Kosten pro Montat**: ~ CHF 28 (fix pro Namespace; 2 x 5G Cache Size) Variante 2 **Separate Caches pro Projekt/Repository**  Konzept: ein Cache Volume pro Projekt resp. Repository. Dieser Cache enthält (nur) die Dependencies für ein spezifisches Projekt und die read-write Einschränkungen verhindern lediglich parallele Builds desselben Projekts. Der Cache muss für jedes Projekt neu aufgebaut werden, was Redundanz und langsame erste Builds mit sich bringt.   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Minimale Änderungen erforderlich | Viele redundante Daten | | Kleine Cache Volumes | Langsames Cache Warming | | Blockierung auf ein Projekt beschränkt | Cleanup: wipe all (pro Repo) |   **Processing Overhead**: 0s  **Kosten pro Monat**: ~ CHF 56 (dynamisch pro Repo; Annahme: 10 Repos à 2G Cache Size) Variante 3 **Mehrere dynamisch allozierte Caches (konfigurierbare Anzahl)**  Konzept: mehrere gesharte Caches ähnlich wie Variante 1, welche dynamisch den zu startenden Pipelines zugewiesen werden. Die Blockierung von parallelen Builds wird damit entschärft. Der ESTA Tekton Controller "verwaltet" die Caches und prüft vor dem Start einer Pipeline, welches Cache Volume verfügbar ist. Das ergibt einen gewissen Verwaltungsaufwand und zusätzliche Logik im Controller die potenziell "out-of-sync" sein kann und im schlimmsten Fall einen Lock von Pipelines zur Folge haben kann.   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Blockierung kann per Config beeinflusst werden | Redundante Daten | | Gute Verteilung von Cache Daten | Zusätzliche Logik im Controller | |  | Nicht "frei gegebene" Caches blockieren neue Pipelines | |  | Cleanup: wipe all |   **Processing Overhead**: ~ 1s  **Kosten pro Monat**: ~ CHF 56 (konfigurierbar pro Namespace; Annahme: 4 Caches à 5G Cache Size)  **POC** implementiert in Pipeline [Templates](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-templates/compare/commits?sourceBranch=refs%2Fheads%2Ffeature%2FESTA-5103-cache-name-param-poc&targetBranch=refs%2Fheads%2Fmaster) und [Controller](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-controller/compare/commits?sourceBranch=refs%2Fheads%2Ffeature%2FESTA-5103-multiple-shared-caches-poc&targetBranch=refs%2Fheads%2Fmaster). Variante 4 **Cache-Dumps werden zentral gespeichert und in Pipeline Workspace kopiert (Github Style)**  Konzept: Cache Daten werden beim Pipeline Start von einer zentralen Storage (Service) mit einem bestimmten Key (pro Projekt/Artefakt/Version) in den Pipeline Workspace oder ein Lokales Volume geladen und nach dem Build wieder gepackt und in die Storage zurück geschrieben. Dabei gibt es einen Fallback auf den Cache eines anderen Projekts desselben Build Stacks, um nicht wie Variante 2 für jedes Projekt mit einem leeren Cache starten zu müssen. Dieser Ansatz wird auch von [Github Actions](https://docs.github.com/en/actions/using-workflows/caching-dependencies-to-speed-up-workflows) angeboten.  Die genaue Technologie für einen zentralen Storage Service muss noch evaluiert werden. AWS S3 oder etwas ähnliches wie [Azure Blob Storage](https://azure.microsoft.com/de-de/services/storage/blobs/). Es benötigt zudem einen Dienst oder Job um nicht mehr verwendete Cache-Entries periodisch zu löschen, um die Storage Grösse unter Kontrolle zu halten.   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Keine Blockierung durch PVCs | Komplexe Implementation | | Kleine Caches | Zusätzlicher Dienst erforderlich | | Cleanup: inkrementell nach Usage | Datentransfer (copy) vor und nach dem Build |   **Processing Overhead**: ~ 11s (~ 3s (Query+Download) + 0.831s (Unpack) + 5.712s (Tar GZ) + 1.363s (Upload))  **Kosten pro Monat**: ~ USD 1.24 (Annahme: 10 Repos mit je 10 Cache Entries à 500M = 50GB; 400 Builds pro Monat (→ 4 COPY/LIST/PUT Operations pro Build + je 1GB Datentransfer))  **POC** mit S3 implementiert in [Pipeline Templates](https://code.sbb.ch/projects/KD_ESTA/repos/esta-tekton-pipeline-templates/compare/commits?sourceBranch=refs%2Fheads%2Ffeature%2FESTA-5103-cache-download-upload-poc&targetBranch=refs%2Fheads%2Fmaster). Variante 5 **Varninsh Cache als Proxy für bin.sbb.ch**  Pro Tekton Build Namespace wird ein [Varnish Cache](https://varnish-cache.org) Deployment mit persistenter Storage (File Storage Backend via PVC) hochgefahren, welches als Proxy für Maven und NPM Repos konfiguriert wird.   |  |  | | --- | --- | | **Vorteile** | **Nachteile** | | Keine Blockierung durch PVCs | Zusätzliches Deployment | | Keine Redundanz | Wenig Knwohow vorhanden | | Cleanup: automatisch | Kann kein SSL; erfordert zusätzlichen Proxy |   **Processing Overhead**: 0s  **Kosten pro Monat**: ~ CHF 117 (fix pro Namespace; 5G Cache Size + zusätzliche CPU + Memory für Varnish Pods) |
| **Entscheidung** | **Variante 4** |
| **Begründung** | Am ARCH-Sync wurde diese Variante mit S3 Storage als kostengünstigste und am besten skalierbare Option gewählt und in einem [Folgemeeting](https://confluence.sbb.ch/display/ESTA/2022-07-29+ESTA+Tekton+Cache+Konzept) im Detail besprochen. Eine Cache Eviction-Policy kann direkt in S3 konfiguriert werden, so dass nur wenige Logick für das Management des Caches selbst implementiert werden muss. |
| **Wer** | [Jeanneret Julien (IT-PTR-CEN1-BDE3)](file:///C:\display\~u231154) [Spirig Lukas (IT-PTR-CEN2-SL3)](file:///C:\display\~u228304) |
| **Wann** | 29.07.2022 |