# Tekton Resolvers Review

ClusterTasks werden mit Tekton v0.41.0 [deprecated](https://tekton.dev/docs/pipelines/deprecations/#deprecation-table). Das neue Konzept nennt sich [Resolvers](https://tekton.dev/docs/pipelines/resolution-getting-started/) und lässt vielseitige Möglichkeiten zum laden von Pipelines und Tasks zu. Siehe auch den [Blogpost von RedHat](https://cloud.redhat.com/blog/migration-from-clustertasks-to-tekton-resolvers-in-openshift-pipelines).

Nachfolgend haben wird die verschiedenen Resolvers analysiert und deren Eignung für den Einsatz in ESTA Tekton beurteilt.

[ESTA-5478](https://flow.sbb.ch/browse/ESTA-5478) - ESTA Tekton Task & Pipeline Resolvers Konzept Closed

# Cluster Resolver

Dies ist der direkte Ersatz für ClusterTasks und die taskRef kann einfach umgestellt werden.

+ Die Tasks sind OpenShift kompatibel mit RedHat Images  
- Unterstützt keine Versionierung

**Fazit**: bedingt brauchbar

# Git Resolver

Mit entsprechender [Default Config](https://tekton.dev/docs/pipelines/git-resolver/#configuration) wäre die Integration recht einfach:

taskRef:

resolver: git

params:

- name: org

value: KD\_ESTA

- name: repo

value: esta-tekton-pipeline-templates

- name: pathInRepo

value: tasks/esta-cache-load.yaml

- name: revision

value: 0.14.7

"revision" könnte via Template oder Pipeline Parameter definiert werden.

+ Möglicher Ersatz für Custom Repos und Tasks  
- Bei Github (public Repos), haben die meisten Tasks Image-Referenzen, die auf OpenShift geblockt sind (nicht via [bin.sbb.ch](http://bin.sbb.ch))  
- Default Config mit Bitbucket Server ([authenticated API](https://tekton.dev/docs/pipelines/git-resolver/#authenticated-api)) ist global und benötigt ein Secret mit Bitbucket Access Token (read-only)  
- Resolver unterstützt nur HTTP Access Tokens, welche pro Projekt resp. Repo ausgestellt werden können. Zugriff auf mehrere Projekte in Bitbucket (pro Tekton Namespace) ist also nicht möglich.

**git-resolver-config** Quelle erweitern

kind: ConfigMap

apiVersion: v1

metadata:

name: git-resolver-config

namespace: openshift-pipelines

...

data:

default-url: ''

# NOTE: omitting the namespace config will read the configured secret from the Tekton namespace

# api-token-secret-namespace: openshift-pipelines

api-token-secret-name: bitbucket-resolver-token

api-token-secret-key: token

server-url: 'https://code.sbb.ch'

default-revision: master

scm-type: bitbucketserver

fetch-timeout: 1m

default-org: KD\_ESTA

**Fazit**: brauchbar

# Hub Resolver

- Die meisten Tasks haben Image-Referenzen, die auf Openshift geblockt sind (nicht via [bin.sbb.ch](http://bin.sbb.ch))

**Fazit**: unbrauchbar

## Bundles Resolver

+ Einfache Erstellung von Bundles mit `tkn bundle push`  
+ Einfache Integration in die Pipeline Templates möglich (für Task-Referenzen)  
+ Versionierung via Pipeline Parameter möglich (siehe Beispiel unten BUNDLE\_IMAGE: [esta.docker.bin.sbb.ch/esta/esta-tekton-pipeline-bundle:latest](http://esta.docker.bin.sbb.ch/esta/esta-tekton-pipeline-bundle:latest))

taskRef:

  resolver: bundles

  params:

    - name: bundle

      value: $(params.BUNDLE\_IMAGE)

    - name: name

      value: esta-cache-load

**Fazit**: brauchbar

# Fazit

Als brauchbare Konzepte haben sich [Bundles](#TektonResolversReview-BundlesResolver) und [Git Resolver](#TektonResolversReview-GitResolver) gezeigt. Der ESTA Tekton Task Katalog könnte mit wenig Aufwand als Bundle released und publiziert werden. Auch Custom Tasks könnten über ein Bundle referenziert werden, allerdings müssten die Kunden selber diese Bundles erstellen und pflegen.

Die Pipeline Templates lassen sich jedoch nicht über einen Resolver einbinden, aus dem einfachen Grund, dass ESTA Tekton PipelineRuns mit inline PipelineSpec benötigt, um die Pipelines vor dem Start noch zu modifizieren. Dies lässt sich nur bewerkstelligen, indem die Pipeline Templates direct als Kubernetes Ressource eingelesen werden können. Ein "Import" der Pipeline Templates in den spezifischen Namespace ist also zwingend notwendig und in diesem Zug können auch die Tekton Tasks direkt importiert werden und müssen nicht via Resolver geladen werden. So bleibt sichergestellt, dass die Pipelines und die Tasks konsistent und miteinander kompatibel sind.

Die Anforderung, Tasks und Pipelines pro Namespace oder gar pro Repo zu parametriesieren (z.B. esta-maven-build mit höheren Memory Requests, siehe [ESTA-5461](https://flow.sbb.ch/browse/ESTA-5461)), kann mit keinem der oben genannten Konzepte direkt umgesetzt werden. Hier muss die taskRef im Pipeline Template angepasst werden und verschiedene Varianten des Tasks im Katalog (egal ob lokal oder via Resolver) bereitgestellt werden. Dazu sollte der Einsatz von Helm Templating für die Generierung des Task Katalogs in Betracht gezogen werden (→ [ESTA-5484](https://flow.sbb.ch/browse/ESTA-5484)). Die esta-tekton-pipeline-templates können parallel als Helm Chart und als Tekton Bundle released werden, um die grösstmögliche Flexibilität anzubieten.

## Custom Pipelines und Tasks

Die aktuelle Implementierung von Custom Tasks und Custom Pipelines in ESTA Tekton sollte indes erweitert werden, so das Pipelines und Tasks auch via Bundles oder Git Resolver referenziert werden können. Damit könnte der direkte Import von Custom Repositories in den Namespace abgelöst werden. Die Abnehmer müssten in diesem Fall selber Bundles erstellen. Dafür könnten wir aber geeignete Werkzeuge (z.B. einen Bundle Build Task in Tekton) anbieten.

# Weitere Schritte

* ~~Git Resolver für Bitbucket (code.sbb.ch) testen und dokumentieren~~
* Resolver Unterstützung für Custom Tasks und Pipelines in estaTektonPipeline.json einbauen (taskRef erweitern) →  [ESTA-5495](https://flow.sbb.ch/browse/ESTA-5495) - ESTA Tekton Resolvers für Custom Tasks/Pipelines Open
* esta-tekton-pipeline-templates als Helm Chart anbieten →  [ESTA-5484](https://flow.sbb.ch/browse/ESTA-5484) - ESTA Tekton: Tekton Templates als Helm Chart Closed