Errata / Ergänzungen GitOps, Stand 19-11-2025

5.7.11 [korrigiert/ergänzt:] Unterstützte Variablen-Substitution in Tasks und Pipelines

Tekton unterstützt mannigfaltige Variablen-Substitutionen in Tasks und Pipelines, die weitestgehend selbsterklärend sind (siehe dazu *https://tekton.dev/docs/pipelines/variables/*).

[…]

Soll nun beispielhaft ein (`kubectl`-)Task automatisch in dem Namespace ausgeführt werden, in dem auch der PipelineRun gestartet wird, könnte der betreffende Step im Task wie folgt aussehen, nachstehend bezogen auf unsere Picalc-Pipeline.

In der Task-Sektion der Pipeline:

```
[...]
    - name: deploy-to-cluster
      taskRef:
        name: deploy-using-kubectl
        kind: Task
      runAfter:
        - source-to-image
      workspaces:
        - name: yaml-source
          workspace: pipeline-pvc
      params:
        - name: pathToYamlFile
          value: "$(params.pathToYamlFile)"
       […]
        - name: namespace
          value: "$(context.pipelineRun.namespace)"
```

Und im Task selbst:

```
    - name: run-kubectl
      image: my/k8s-kubectl
      command: ["kubectl"]
      args:
        - apply
        - -n
        - $(params.namespace)
        - -f
        - $(workspaces.yaml-source.path)/$(params.pathToYamlFile)
```

Ein wichtiger Benefit, bezogen auf Trigger und diesen konkreten Fall: Es sind keine Ergänzungen an TriggerBinding/TriggerTemplate erforderlich.

5.21.3 [ergänzt] High-Level-Arbeitsweise der Pipeline und Tasks

Download und Einbindung des Windows-ISO

Die Tekton-Pipeline lädt das Windows Server 2022-Installations-ISO direkt aus dem Internet per HTTPS herunter [...]

Es handelt sich also nicht um ein rein flüchtiges In-Memory-Image, sondern um eine persistent gespeicherte Disk, die aber im Verlauf der Pipeline nur temporär genutzt und am Ende wieder gelöscht wird (siehe weiter unten).

[ergänzt:]

Um den Overhead zu verringern bzw. für Air-Gapped Systeme, hinterlegen Sie das ISO natürlich auf einem lokalen HTTP-Server bzw. stellen es in Ihrer unternehmenseigenen Registry zur Verfügung. Beachten Sie für letzteren Fall, dass Sie das ISO als Container Image in der Registry hinterlegen müssen. In der Pipeline müssen Sie im Task *import-win-iso* den Parameter `spec.source.http.url` auf `spec.source.registry.url` umstellen.

5.21.4 [ergänzt] Die Tasks im Detail

Ein wichtiger Punkt bei den folgenden Tasks sind die ***Sysprep***- (Systemvorbereitung) und *unattended*-spezifischen Aufgaben [...]

[ergänzt] Customisieren des Installationsprozesses

Wenn Sie den Installationsprozess der Windows VM anpassen möchten, können Sie das Sysprep entsprechend modifizieren.

Siehe zu diesem Exkurs auch: *https://developers.redhat.com/articles/2024/09/09/create-windows-golden-image-openshift-virtualization#customize_the_installation_process_*

Die für das Sysprep bzw. unattended Setup verwendete ConfigMap (im Folgenden: `w2k22-server-autounattend-custom`), die im selben Project liegen muss wie die Pipeline bzw. der ausgeführte Run, muss die folgende Struktur aufweisen:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: w2k22-server-autounattend-custom
data:
```

```
  autounattend.xml: |
    <?xml version="1.0" encoding="utf-8"?>
    <unattend xmlns="urn:schemas-microsoft-com:unattend"
xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/State">
      #Insert your custom configurations here:
    </unattend>
  post-install.ps1 |
    #Insert your commands to perform customizations here:
    […]
```

Details zum Thema Sysprep bzw. unattended Setups finden Sie unter anderem hier:

https://learn.microsoft.com/de-de/windows-hardware/manufacture/desktop/update-windows-settings-and-scripts-create-your-own-answer-file-sxs

https://learn.microsoft.com/de-de/windows-hardware/manufacture/desktop/sysprep--generalize--a-windows-installation

Damit die Pipeline darüber informiert ist, das ein customisiertes Sysprep zum Einsatz kommt, muss diese im PR entsprechend hinterlegt werden:

```
apiVersion: tekton.dev/v1
kind: PipelineRun
metadata:
    generateName: w2k22-server-installer-run-
spec:
    params:
[...]
    -   name: autounattendConfigMapName
        value: w2k22-server-autounattend-custom
[...]
    pipelineRef:
[...]
```

[...]

5.21.6 [korrigiert/ergänzt] Erzeugte Objekte

[...]

Zwei Mankos dieser Pipeline bleiben allerdings, die noch gefixt bzw. ergänzt werden sollten:

- [korrigiert/ergänzt] Durch die parallele Löschung (im `Cleanup` der `Finally`-Sektion) von ConfigMaps, ISO und VM bleibt nach Abschluss des Runs das temporäre PVC (und damit PV) der VM, das für die Image-Erstellung verwendet wurde, des Öfteren hängen. Es kann dann nur noch manuell per PVC-Finalizer-Abschuss entsorgt werden. Hier kann eine Sequenzialisierung mit `when` und Status-Abfragen helfen. Red Hat sagt zwar, dass sich das »Zombie«-PVC entfernt, wenn der PR gelöscht wird, aber zum einen klappt das nicht immer (und falls doch, dauert es bis zur Löschung manchmal mehrere Minuten); zum anderen möchte man die PRs aus Debug-Gründen vielleicht nicht sofort löschen (umso mehr, da im »Archiv« der Tekton Results API gerne mal die Logs der TaskRuns verloren gehen).
- Die Disk-Sizes [...]

6.2.5 [korrigiert/ergänzt:] Login auf dem Argo-CD-Server (CLI)

[korrigiert] Um Aktionen durchführen zu können (z. B. das Hinzufügen von Repos mit der Argo CD-CLI), müssen Sie sich zunächst gegen den Argo CD-Server über die bereitgestellte Route bzw. den Ingress authentifizieren. Hier sehen Sie das für eine OpenShift-Route:

```
# ADMIN_PASSWD=$(oc get secret openshift-gitops-cluster \
 -n openshift-gitops -o jsonpath='{.data.admin\.password}' | base64 -d)
# echo $ADMIN_PASSWD
cA6rno0yB4auU1wvLiXeOk5dJVSjRMpD
# argocd login \
  openshift-gitops-server-openshift-gitops.apps.opns0.local1.site \
  --username admin --password "${ADMIN_PASSWD}" --grpc-web \
  --loglevel debug
'admin:login' logged in successfully
Context 'openshift-gitops-server-openshift-gitops.apps.opns0.local1.site' updated
```

[neu/ergänzt:] CLI Login -Probleme

Gibt es bei der CLI Anmeldung Probleme, z. B. wenn Sie in einen Timeout laufen:

```
{"level":"fatal","msg":"context deadline exceeded","time":"2025-11-15T16:39:18+01:00"}
```

können Sie das Problem wie folgt eingrenzen.

Stellen Sie zunächst sicher, dass der ArgoCD Server selbst funktioniert (Login per UI okay), die Route ebenfalls funktioniert und DNS-technisch auflösbar ist:

```
#  nslookup \
   openshift-gitops-server-openshift-gitops.apps.opns0.local1.site
Server:         192.168.99.63
Address:        192.168.99.63#53
Name:   openshift-gitops-server-openshift-gitops.apps.opns0.local1.site
Address: 192.168.99.79
#       curl       -vk       https://openshift-gitops-server-openshift-
gitops.apps.opns0.local1.site/
*   Trying 192.168.99.79:443...
* Connected to openshift-gitops-server-openshift-gitops.apps.opns0.local1.site
(192.168.99.79) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
*  CAfile: /etc/pki/tls/certs/ca-bundle.crt
* TLSv1.0 (OUT), TLS header, Certificate Status (22):
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
[…]
TLSv1.2 (IN), TLS header, Unknown (23):
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< accept-ranges: bytes
< content-length: 788
[…]
```

Bei Self-Signed Certificates müssen Sie bis OpenShift Version 4.18 (OpenShift GitOps 1.17/1.18) in jedem Fall den Zusatzschalter `--insecure` verwenden. Unter OpenShift 4.19 ist gegebenenfalls zusätzlich der Schalter `--skip-test-tls` erforderlich:

```
# argocd login \
  openshift-gitops-server-openshift-gitops.apps.opns0.local1.site \
  --username admin --password "${ADMIN_PASSWD}" \
  --insecure --loglevel debug --grpc-web --skip-test-tls
'admin:login' logged in successfully
Context 'openshift-gitops-server-openshift-gitops.apps.opns0.local1.site' updated
```

[neu:] Alternatives CLI Login per --core

Alternativ zu den oben angeführten Login-Verfahren können Sie sich zu Testzwecken, oder beispielsweise wenn das Login per Route / Ingress nicht funktionieren sollte, auch per `argocd login --core` anmelden:

Beim „normalen“ argocd-Login über Route/Ingress (argocd login <route-host> ...) spricht das CLI mit dem Argo-CD-API-Server, der hinter einer HTTP(S)-Route oder einem Ingress hängt, und benutzt dessen eigene Authentifizierung und Session-Logik. Dabei laufen alle Operationen über die Argo-CD-API (gRPC/gRPC-Web), der Server hält Status, RBAC und SSO-Integration zusammen und verhält sich aus Sicht des CLI wie jede andere »externe« Argo-CD-Installation.

Im `--core`-Modus (`argocd login --core`) gibt es dagegen kein Login zum Argo-CD-Server: das CLI nutzt direkt die Kubernetes-API und die im Cluster gespeicherten Argo-CD-Ressourcen, so als wäre der Kube-API-Server der „Backend-Endpunkt“ für das CLI. Authentifizierung, Autorisierung und Kontext kommen ausschließlich aus der aktuellen kubeconfig; Argo-CD-Server, Route/Ingress und deren TLS-/SSO-Konfiguration werden übergangen und spielen in diesem Modus keine Rolle. Dies sollte natürlich nur auf Testsystemen gestattet sein.

6.4.8 [neu/ergänzt] OpenShift und Prometheus Error 'PrometheusOperatorRejectedResources'

https://access.redhat.com/solutions/6706741

Sollte unter OpenShift nach dem Setup der OpenShift GitOps Metrics-Settings und dem Setup des User-Workload-Monitorings von Prometheus ein `PrometheusOperatorRejectedResources` -Alert auftauchen, checken Sie die Logs des Prometheus Operators im Namespace `openshift-user-workload-monitoring`. Dort werden Sie wahrscheinlich Meldungen wie z. B. die folgende finden, bezogen auf z. B. den Namespace openshift-user-workload-monitoring, aber gegebenenfalls auch für den Namespace openshift-gitops oder andere:

```
Event occurred" object.name=openshift-gitops-operator-metrics-monitor
object.namespace=openshift-gitops-operator fieldPath="" kind=ServiceMonitor
apiVersion=monitoring.coreos.com/v1 type=Warning reason=InvalidConfiguration
message="ServiceMonitor openshift-gitops-operator-metrics-monitor was rejected due to
invalid configuration: it accesses file system via bearer token file which Prometheus
specification prohibits"
```

Um das Problem zu lösen, setzen Sie in den betreffenden Namespaces das folgende Label:

```
# o label namespace openshift-gitops-operator \
  openshift.io/cluster-monitoring=true
namespace/openshift-gitops-operator labeled
```

```
# o label namespace openshift-user-workload-monitoring \
   openshift.io/cluster-monitoring=true
namespace/openshift-user-workload-monitoring not labeled
```

Zur Kontrolle können Sie optional den Prometheus-Operator des Namespace `openshift-user-workload-monitoring` durchstarten und dessen Logs prüfen; die Alerts in der UI sollten ebenfalls verschwunden sein.

9.7.1 [korrigiert/ergänzt] [ESO-]Setup

Im Folgenden wird der ESO unter OpenShift mit Vault-Backend aufgesetzt.

ESO

Zunächst installieren Sie den ***External Secrets Operator*** (im betrachteten Stand die Community Version 0.11.0) aus dem OperatorHub.

Achtung: OpenShift External Secrets Operator

Wenn Sie die ***OpenShift Variante des External Secrets Operators*** verwenden (als TP verfügbar seit OpenShift 4.19) existiert dort keine ***OperatorConfig*** CR! Stattdessen muss dort (am einfachsten für ein PoC mit den Default Settings) die CR ***ExternalSecrets*** erzeugt werden, um die passenden Controller zu generieren. Die Bezeichnung ist extrem schlecht gewählt von Red Hat, da sie zu leicht mit der CR ***ExternalSecret*** zu verwechseln ist.

https://docs.redhat.com/en/documentation/openshift_container_platform/4.19/html/security_and_compliance/external-secrets-operator-for-red-hat-openshift

Sobald der Operator ausgerollt ist, erzeugen Sie eine ***OperatorConfig***-CR in dem Namespace, in dem auch der ESO ausgerollt wurde. Typischerweise wäre das in der Community Version `openshift-operators` oder für die OpenShift Variante `external-secrets-operator`.

Für einen einfachen Test reichen die Default-Settings der ***OperatorConfig***-CR bzw. ***ExternalSecrets***-CR. Ist alles implementiert, sollten Sie die folgenden ESO-Pods im Namespace `openshift-operators` sehen (hier für die Community Variante):

```
# o get pods -n openshift-operators | grep external-secret
external-secrets-operator-controller-manager-[…]  1/1   Running   0   13m
sample-external-secrets-66dfdb97c4-lnp7q          1/1   Running   0   57s
sample-external-secrets-cert-controller-7b56[…]   1/1   Running   0   57s
sample-external-secrets-webhook-575f87546-ldtm6   1/1   Running   0   57s
```

In der OpenShift Variante des ESO finden sich die Controller im Namespace `external-secrets`:

```
# o get pods -n external-secrets
NAME                                                READY   STATUS    RESTARTS   AGE
external-secrets-78944c9c88-5x9cb                   1/1     Running   0          2m49s
external-secrets-cert-controller-7cdd75d687-89vm2   1/1     Running   0          2m49s
external-secrets-webhook-59ff6ff965-nxptf           1/1     Running   0          2m49s
```