

# Container-Image-Digests verwenden

Dieses Dokument zeigt Entwicklern und Operatoren, die Container-Images erstellen und bereitstellen, was Digests sind und wie sie funktionieren. Ein Container-Image-Digest (<https://github.com/opencontainers/image-spec/blob/master/descriptor.md#digests>) identifiziert ein Container-Image eindeutig und unveränderlich. Wenn Sie Images per Digest bereitstellen, vermeiden Sie die Nachteile der Bereitstellung durch Image-Tags (<https://github.com/opencontainers/distribution-spec/blob/master/spec.md>).

In diesem Dokument werden die Image-Digests ausführlich erläutert. Sie erfahren, was Image-Digests sind, wie Sie sie finden und wie Sie deren Verwendung in Kubernetes-Clustern durchsetzen. Informationen zum Hinzufügen von Image-Digests zu Kubernetes-Manifesten finden Sie in der Anleitung zum Verwenden von Container-Image-Digests mit Kubernetes-Manifesten (</solutions/using-container-image-digests-in-kubernetes-manifests>).

Bei den Befehlen in diesem Dokument wird davon ausgegangen, dass Sie Zugriff auf eine Linux- oder macOS-Shell-Umgebung haben, auf der bereits Tools wie die Google Cloud CLI (`/sdk`), Docker, `cURL`, `jq` (<https://stedolan.github.io/jq/>) und `pack` (<https://buildpacks.io/docs/tools/pack/>) installiert sind. Alternativ können Sie Cloud Shell (`/shell`) verwenden, wo diese Tools bereits vorinstalliert sind.

Wenn Sie mit Container-Images arbeiten, benötigen Sie eine Möglichkeit, auf die von Ihnen verwendeten Images zu verweisen. Image-Tags (<https://github.com/opencontainers/distribution-spec/blob/master/spec.md>) sind eine gängige Methode, um auf verschiedene Überarbeitungen eines Images zu verweisen. Ein gängiger Ansatz besteht darin, Images beim Erstellen mit einer Versionskennung zu versehen, z. B. `v1.0.1`, um auf eine Version zu verweisen, die Sie 1.0.1 nennen.

Mit Tags können Sie Ihre Images ganz einfach anhand von menschenlesbaren Strings nachschlagen. Tags sind jedoch änderbare Verweise. Das bedeutet, dass sich das von einem Tag referenzierte Image ändern kann, wie im folgenden Diagramm dargestellt:



Wie im vorherigen Diagramm gezeigt: Wenn Sie ein neues Image mit demselben Tag wie ein vorhandenes Image veröffentlichen, verweist das Tag nicht mehr auf das vorhandene Image sondern auf Ihr neues Image.

Da Tags änderbar sind, haben sie folgende Nachteile, wenn Sie sie zum Bereitstellen eines Images verwenden:

- In Kubernetes kann das Bereitstellen über Tags zu unerwarteten Ergebnissen führen. Angenommen, Sie haben eine vorhandene Deployment-Ressource, die auf ein Container-Image mit dem Tag `v1.0.1` verweist. Zur Behebung eines Fehlers oder zur Durchführung einer kleinen Änderung erstellt der Build-Prozess ein neues Image mit dem gleichen Tag `v1.0.1`. Neue Pods, die aus Ihrer Deployment-Ressource erstellt werden, können entweder das alte oder das neue Image verwenden, auch wenn Sie die Deployment-Ressourcenspezifikation nicht ändern. Dieses Problem betrifft auch andere Kubernetes-Ressourcen wie StatefulSets, DaemonSets, ReplicaSets und Jobs.
- Wenn Sie zum Scannen oder Analysieren von Images Tools verwenden, sind die Ergebnisse dieser Tools nur für das gescannte Image gültig. Um sicherzustellen, dass Sie das gescannte Image bereitstellen, können Sie sich nicht auf das Tag verlassen, da sich das Image, auf das das Tag verweist, möglicherweise geändert hat.
- Wenn Sie die Binärautorisierung (/binary-authorization) mit Google Kubernetes Engine (GKE) (/kubernetes-engine/docs) verwenden, ist die Tag-basierte Bereitstellung nicht zulässig, da das genaue Image, das beim Erstellen eines Pods verwendet wird, nicht ermittelt werden kann.

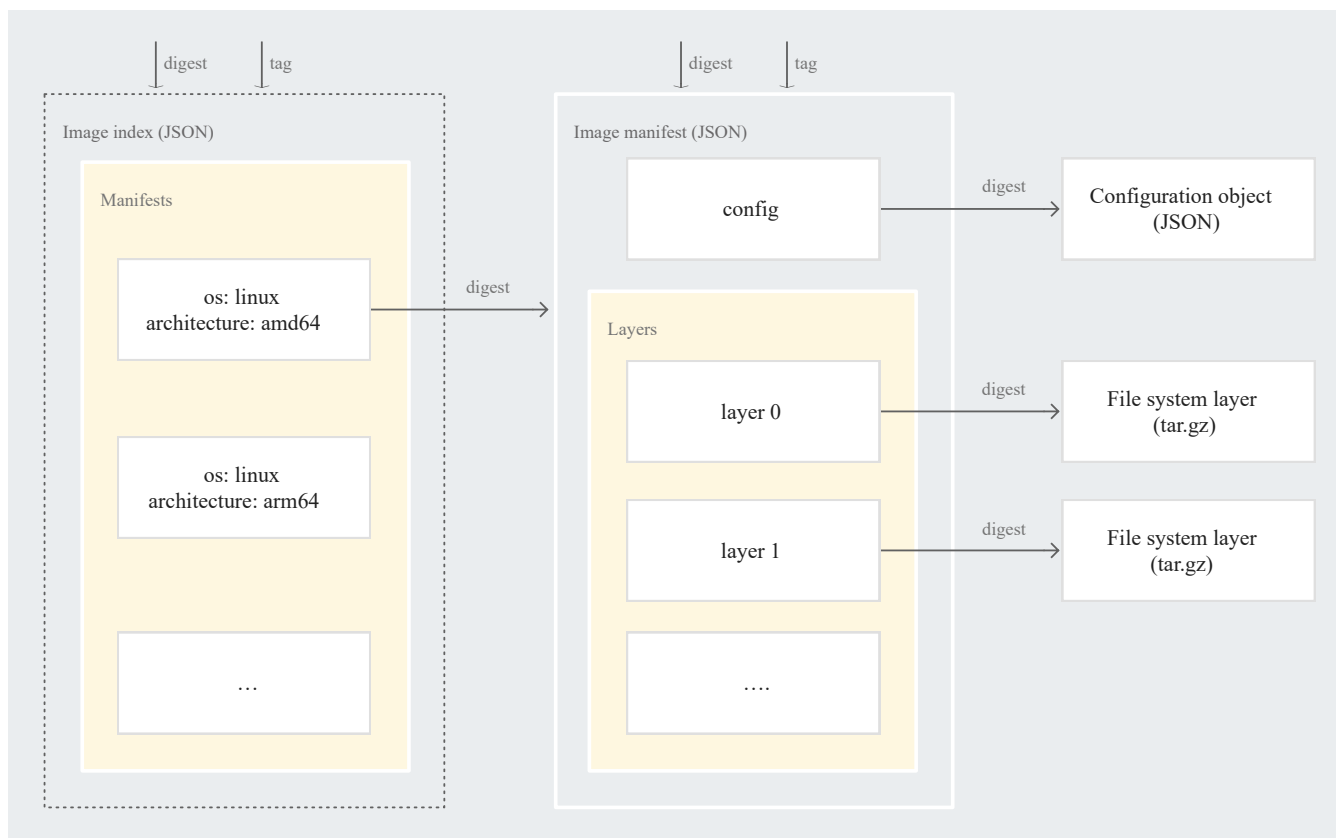
Wenn Sie Ihre Images bereitstellen, können Sie einen Image-Digest verwenden, um die Nachteile von Tags zu vermeiden. Sie können Ihren Images auch bei Bedarf Tags hinzufügen. Dies ist jedoch nicht erforderlich.

## Struktur eines Images

Ein Image besteht aus den folgenden Komponenten:

- Ein Image-Manifest  
(<https://github.com/opencontainers/image-spec/blob/master/manifest.md#oci-image-manifest-specification>)
- Ein Konfigurationsobjekt  
(<https://github.com/opencontainers/image-spec/blob/master/manifest.md#image-manifest-property-descriptions>)
- Ein Array mit einer oder mehreren Dateisystem-Layern  
(<https://github.com/opencontainers/image-spec/blob/master/layer.md#image-layer-file-system-changeset>)
- Ein optionaler Image-Index  
(<https://github.com/opencontainers/image-spec/blob/master/image-index.md#oci-image-index-specification>)

Diese Komponenten werden im folgenden Diagramm dargestellt:



Das vorherige Image zeigt zusätzliche Details zu Image-Komponenten:

- Das Image-Manifest ist ein JSON-Dokument, das einen Verweis auf das Konfigurationsobjekt, die Dateisystem-Layers und optionale Metadaten enthält.
- Das Image-Manifest verweist auf das Konfigurationsobjekt und die einzelnen Dateisystem-Layer mithilfe ihrer **digest**-Attribute. Der Wert eines **digest**-Attributs ist ein kryptografischer Hash der Inhalte, auf die sich der Digest bezieht, und die normalerweise mit dem SHA-256 (<https://github.com/opencontainers/image-spec/blob/master/descriptor.md#sha-256>)-Algorithmus berechnet werden.
- Die Digest-Werte werden verwendet, um unveränderliche Adressen für die Objekte zu erstellen. Dieser Vorgang wird als Content-Addressed Storage ([https://wikipedia.org/wiki/Content-addressable\\_storage](https://wikipedia.org/wiki/Content-addressable_storage)) bezeichnet und bedeutet, dass Sie Image-Manifeste, Image-Indexe, Konfigurationsobjekte und Layer anhand ihrer Digests abrufen können.
- Der Image-Digest ist der Hash des des Image-Index oder des Image-Manifest-JSON-Dokuments.

- Das Konfigurationsobjekt ist ein JSON-Dokument, das Attribute des Images (<https://github.com/opencontainers/image-spec/blob/master/config.md#properties>) definiert, z. B. CPU-Architektur, Einstiegspunkt, verfügbare Ports und Umgebungsvariablen.
- Das Dateisystemarray definiert die Reihenfolge, in der die Containerlaufzeit die Layer stapelt. Die Layer werden als TAR-Dateien ([https://wikipedia.org/wiki/Tar\\_\(computing\)](https://wikipedia.org/wiki/Tar_(computing))) verteilt und in der Regel mit dem Dienstprogramm gzip (<https://www.gzip.org/>) komprimiert.
- Der optionale Image-Index, auch als Manifestliste (<https://docs.docker.com/registry/spec/manifest-v2-2/#manifest-list>) bezeichnet, bezieht sich auf ein oder mehrere Image-Manifeste. Die Referenz ist der Digest des Image-Manifests. Ein Image-Index ist nützlich, wenn Sie mehrere verwandte Images für verschiedene Plattformen (<https://github.com/opencontainers/image-spec/blob/master/image-index.md#image-index-property-descriptions>) wie amd64- und arm64-Architekturen erstellen.

Weitere Informationen finden Sie im Abschnitt Image-Manifeste, -Digests und -Tags kennenlernen (`#exploring_image_manifests_digests_and_tags`).

## Image-Digests finden

Sie müssen zuerst den Digest suchen, um Image-Digests für die Bereitstellung zu verwenden. Anschließend können Sie den Digest mit Ihrem Bereitstellungsbefehl verwenden oder in Ihre Kubernetes-Manifeste einfügen.

Sie können den Digest eines Images je nach aktueller Situation auf unterschiedliche Weise abrufen. Die folgenden Abschnitte enthalten Beispiele für verschiedene Produkte und Tools.

Führen Sie in den folgenden Abschnitten die Befehle in Cloud Shell oder in einer Shell-Umgebung aus, in der die `gcloud` CLI, Docker, `cURL` und `jq` bereits installiert sind.

## Container Registry

- Für Images, die in der Container Registry (`/container-registry`) gespeichert sind, können Sie den Befehl `gcloud container images describe` (`/sdk/gcloud/reference/container/images/describe`) verwenden, um den Digest für ein Image

abzurufen. Geben Sie dazu den Namen und ein Tag an. Verwenden Sie das Flag `--format` (/sdk/gcloud/reference/topic/formats), um nur den Digest anzuzeigen:






```
gcloud container images describe \  
  gcr.io/google-containers/pause-amd64:3.2 \  
  --format 'value(image_summary.digest)'
```

Die Ausgabe sieht etwa so aus, wobei der Digest-Wert variieren kann:

```
sha256:4a1c4b21597c1b4415bdbecb28a3296c6b5e23ca4f9feeb599860a1dac6a0108
```

## Artifact Registry

- Für Images, die in Artifact Registry (/artifact-registry) gespeichert sind, können Sie den Befehl `gcloud artifacts docker images describe` (/sdk/gcloud/reference/beta/artifacts/docker/images/describe) verwenden.

```
gcloud artifacts docker images describe \  
  LOCATION  -docker.pkg.dev/PROJECT  /REPOSITORY  /IMAGE  :TAG  \  
  --format 'value(image_summary.digest)'
```

Dabei gilt:

- LOCATION:** der regionale oder multiregionale Standort (/artifact-registry/docs/repo-locations) Ihres Repositorys
- PROJECT:** Ihre Google Cloud-Projekt-ID (/resource-manager/docs/creating-managing-projects#identifying\_projects)
- REPOSITORY:** Name Ihres Repositorys
- IMAGE:** Name Ihres Images
- TAG:** Ihr Image-Tag

## Cloud Build

Für Images, die mit Cloud Build (/build) erstellt wurden, können Sie den Image Digest mithilfe des Befehls `gcloud builds describe` (/sdk/gcloud/reference/builds/describe) mit dem Flag `--format` abrufen. Dieser Ansatz funktioniert unabhängig von der Registry, die Sie zum Veröffentlichen des Images verwendet haben.

- Führen Sie für einen bereits abgeschlossenen Build die folgenden Schritte aus:

1. Rufen Sie eine Liste der Builds für Ihr Projekt ab:

```
gcloud builds list
```

Notieren Sie sich eine ***BUILD\_ID***.

2. Rufen Sie den Image-Digest ab:

```
gcloud builds describe BUILD_ID \
  --format 'value(results.images[0].digest)'
```

Ersetzen Sie ***BUILD\_ID*** durch die eindeutige ID, die Cloud Build Ihrem Build zugewiesen hat.

- Rufen Sie den Image-Namen und den Digest für den neuesten Build von Cloud Build für Ihr aktuelles Projekt ab:

```
gcloud builds describe \
  $(gcloud builds list --limit 1 --format 'value(id)') \
  --format 'value(separator="@")(results.images[0].name,results.images[0]
```

- Wenn Ihr Build mehrere Images erstellt hat, filtern Sie die Ausgabe und erhalten Sie den Digest eines der Images:

```
gcloud builds describe BUILD_ID --format json \
  | jq -r '.results.images[] | select(.name=="YOUR_IMAGE_NAME") | .digest'
```

Ersetzen Sie *YOUR\_IMAGE\_NAME* durch den Namen eines der Images aus der Datei `cloudbuild.yaml` (/build/docs/build-config#images).

- Wenn Sie einen Build mit dem Befehl `gcloud builds submit` (/sdk/gcloud/reference/builds/submit) an Cloud Build senden, können Sie den Image-Digest der Ausgabe in einer Umgebungsvariablen erfassen:

```
IMAGE_DIGEST=$(gcloud builds submit \
  --format 'value(results.images[0].digest)' | tail -n1)
```

## Cloud Native Buildpacks

- Wenn Sie Cloud Native Buildpacks (/blog/products/containers-kubernetes/google-cloud-now-supports-buildpacks) und den Google Cloud-Builder verwenden, um Images zu erstellen und zu veröffentlichen, können Sie den Image-Namen und den Digest mit dem Flag `--quiet` und dem Befehl `pack` erfassen:

```
pack build --builder gcr.io/buildpacks/builder:v1 --publish --quiet \
  gcr.io/PROJECT /IMAGE > image-with-digest.txt
```

Dabei gilt:

- *PROJECT*: Ihre Google Cloud-Projekt-ID (/resource-manager/docs/creating-managing-projects#identifying\_projects)
- *IMAGE*: Name Ihres Images

Die Datei `image-with-digest.txt` enthält den Image-Namen und den Digest.

Verwenden Sie das Flag `--tag`, wenn Sie dem Image zusätzliche Tags hinzufügen möchten.

## Docker-Client

- Für Images, die in einer beliebigen Registry gespeichert sind, können Sie Docker verwenden, um den Image-Digest zu erhalten:



1. Laden Sie das Image in Ihren lokalen Docker-Daemon (<https://docs.docker.com/config/daemon/>) herunter:

```
docker pull docker.io/library/debian:buster
```

2. Rufen Sie den Image-Digest ab:

```
docker inspect docker.io/library/debian:buster \
| jq -r '.[0].RepoDigests[0]' \
| cut -d'@' -f2
```

Die Ausgabe sieht dann ungefähr so aus:

```
sha256:8414aa82208bc4c2761dc149df67e25c6b8a9380e5d8c4e7b5c84ca2d04bb24
```

- Listen Sie alle Images und Digests in Ihrem lokalen Docker-Daemon auf:

```
docker images --digests
```

Die Ausgabe zeigt Digests für Images mit einem Digest-Wert. Ein Container-Image muss in einer Container Registry veröffentlicht werden, bevor ein Digest verfügbar ist.

- Wenn Sie experimentelle Funktionen des Docker-Clients (<https://docs.docker.com/engine/reference/commandline/cli/#experimental-features>) aktivieren, können Sie Image-Manifeste und -Digests erhalten, ohne das Image zuerst in Ihren lokalen Docker-Daemon herunterzuladen. Wenn Sie diesen Befehl mit einem Tag verwenden, wird der Image-Index zurückgegeben, sofern er vorhanden ist. Andernfalls wird das Image-Manifest zurückgegeben.

Rufen Sie den Digest aus dem Image-Index des Images `docker.io/library/debian:buster` für die CPU-Architektur `amd64` und das Betriebssystem `linux` ab:

```
docker manifest inspect --verbose docker.io/library/debian:buster | \
jq -r 'if type=="object"
then .Descriptor.digest
else .[] | select(.Descriptor.platform.architecture=="amd64" and .D
end'
```

Die Ausgabe sieht dann ungefähr so aus:

```
sha256:60cb30babcd1740309903c37d3d408407d190cf73015aeddec9086ef3f393a5d
```

## crane und gcrane

Mit den Open-Source-Befehlszeilentools [crane](https://github.com/google/go-containerregistry/blob/master/cmd/crane/README.md)

(<https://github.com/google/go-containerregistry/blob/master/cmd/crane/README.md>) und [gcrane](https://github.com/google/go-containerregistry/blob/master/cmd/gcrane/README.md)

(<https://github.com/google/go-containerregistry/blob/master/cmd/gcrane/README.md>) können Sie

den Digest eines Images abrufen, ohne das Image an einen lokalen Docker-Daemon zu übertragen. Dies ist nützlich, wenn Sie mit großen Images arbeiten oder in Umgebungen, in denen Docker nicht verfügbar ist.

1. Laden Sie crane und gcrane in Ihr aktuelles Verzeichnis herunter:

```
curl -L https://github.com/google/go-containerregistry/releases/download/v0
```

2. Image-Digests abrufen:

```
./gcrane digest gcr.io/distroless/static:nonroot
```

crane und gcrane haben weitere Funktionen, die in diesem Dokument nicht behandelt werden. Weitere Informationen finden Sie in der Dokumentation zu [crane](https://github.com/google/go-containerregistry/blob/master/cmd/crane/README.md#crane)

(<https://github.com/google/go-containerregistry/blob/master/cmd/crane/README.md#crane>) und

[gcrane](https://github.com/google/go-containerregistry/blob/master/cmd/gcrane/README.md#gcrane)

(<https://github.com/google/go-containerregistry/blob/master/cmd/gcrane/README.md#gcrane>).

# Verwendung von Image-Digests in Kubernetes-Deployments erzwingen

Wenn Sie die Verwendung von Digests für Images, die Sie in Ihren Kubernetes-Clustern bereitstellen, erzwingen möchten, verwenden Sie [Policy Controller](#) (/anthos-config-management/docs/concepts/policy-controller) oder [Open Policy Agent \(OPA\) Gatekeeper](#) (<https://github.com/open-policy-agent/gatekeeper>), um die Option zu aktivieren. Policy Controller basiert auf dem Open-Source-Projekt Gatekeeper.

Policy Controller und Gatekeeper bauen beide auf der [OPA-Richtlinien-Engine](#) (<https://github.com/open-policy-agent/opa>) auf. Policy Controller und Gatekeeper bieten einen [validierenden Zulassungs-Webhooks von Kubernetes](#) (<https://kubernetes.io/docs/reference/access-authn-authz/extensible-admission-controllers/>) zur Durchsetzung von Richtlinien sowie [benutzerdefinierte Ressourcendefinitionen \(CRDs\)](#) (<https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/>) für [Einschränkungsvorlagen](#) (<https://github.com/open-policy-agent/gatekeeper#constraint-templates>) und [Einschränkungen](#) (<https://github.com/open-policy-agent/gatekeeper#constraints>).

Einschränkungsvorlagen enthalten Richtlinienlogiken, die mit einer deklarativen Sprache namens [Rego](#) (<https://www.openpolicyagent.org/docs/latest/#rego>) ausgedrückt werden. Im Folgenden finden Sie eine Einschränkungsvorlage, die prüft, ob Container und [init-Container](#) (<https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>) in einer Kubernetes-Ressourcenspezifikation Images mit Digests verwenden:

```
library/general/imagedigests/template.yaml
```

```
(https://github.com/open-policy-agent/gatekeeper-library/blob/HEAD/library/general/imagedigests/template.yaml)
```

```
'github.com/open-policy-agent/gatekeeper-library/blob/HEAD/library/general/imagedigests/template.yaml')
```

```
apiVersion: templates.gatekeeper.sh/v1beta1
kind: ConstraintTemplate
metadata:
  name: k8simagedigests
  annotations:
    description: >-
      Requires container images to contain a digest.

  https://kubernetes.io/docs/concepts/containers/images/
```

```

spec:
  crd:
    spec:
      names:
        kind: K8sImageDigests
      validation:
        openAPIV3Schema:
          type: object
          description: >-
            Requires container images to contain a digest.

            https://kubernetes.io/docs/concepts/containers/images/
        properties:
          exemptImages:
            description: >-
              Any container that uses an image that matches an entry in this l
              from enforcement. Prefix-matching can be signified with `*`. For

              It is recommended that users use the fully-qualified Docker imag
              in order to avoid unexpectedly exempting images from an untruste
            type: array
            items:
              type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8simagedigests

        import data.lib.exempt_container.is_exempt

        violation[{"msg": msg}] {
          container := input.review.object.spec.containers[_]
          not is_exempt(container)
          satisfied := [re_match("@[a-z0-9]+([+._-][a-z0-9]+)*:[a-zA-Z0-9=_-]+",
            not all(satisfied)
          msg := sprintf("container <%v> uses an image without a digest <%v>", [
        }

        violation[{"msg": msg}] {
          container := input.review.object.spec.initContainers[_]
          not is_exempt(container)
          satisfied := [re_match("@[a-z0-9]+([+._-][a-z0-9]+)*:[a-zA-Z0-9=_-]+",
            not all(satisfied)
          msg := sprintf("initContainer <%v> uses an image without a digest <%v>

```

```

libs:
- |
  package lib.exempt_container

  is_exempt(container) {
    exempt_images := object.get(object.get(input, "parameters", {}), "
    img := container.image
    exemption := exempt_images[_]
    _matches_exemption(img, exemption)
  }

  _matches_exemption(img, exemption) {
    not endswith(exemption, "*")
    exemption == img
  }

  _matches_exemption(img, exemption) {
    endswith(exemption, "*")
    prefix := trim_suffix(exemption, "*")
    startswith(img, prefix)
  }

```

Die vorherige Richtlinie enthält einen regulären Ausdruck als Eingabe für die Funktion `re_match` (<https://www.openpolicyagent.org/docs/v0.22.0/policy-reference/#regex>). Dieser reguläre Ausdruck entspricht dem Container-Image-Digest und basiert auf dem Digest-Format in der Open Container Initiative Image-Spezifikation (<https://github.com/opencontainers/image-spec/blob/master/descriptor.md#digests>).

Einschränkungen wenden die Richtlinie auf Kubernetes-Ressourcen an. Dazu werden sie mit Attributen wie `kind` und `namespace` abgeglichen. Die folgende Beispieleinschränkung wendet die Richtlinie aus der Einschränkungsvorlage auf alle Pod-Ressourcen im Namespace `default` an.

[library/general/imagedigests/samples/container-image-must-have-digest/constraint.yaml](https://github.com/open-policy-agent/gatekeeper-library/blob/HEAD/library/general/imagedigests/samples/container-image-must-have-digest/constraint.yaml)  
 (<https://github.com/open-policy-agent/gatekeeper-library/blob/HEAD/library/general/imagedigests/samples/container-image-must-have-digest/constraint.yaml>)

[rary/blob/HEAD/library/general/imagedigests/samples/container-image-must-have-digest/constraint.yaml](https://github.com/open-policy-agent/gatekeeper-library/blob/HEAD/library/general/imagedigests/samples/container-image-must-have-digest/constraint.yaml))

```

apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sImageDigests
metadata:
  name: container-image-must-have-digest
spec:
  match:
    kinds:
      - apiGroups: ["" ]
        kinds: [ "Pod" ]
    namespaces:
      - "default"

```

Nachdem Sie die Einschränkungsvorlage und die Einschränkung erstellt haben, müssen alle neuen Pods im Namespace `default` Image Digests verwenden, um auf [Container-](https://kubernetes.io/docs/concepts/containers/) (<https://kubernetes.io/docs/concepts/containers/>) und [Init-Container-Images](https://kubernetes.io/docs/concepts/workloads/pods/init-containers/) (<https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>) zu verweisen.

Das vollständige Beispiel finden Sie in der Richtlinie der Gatekeeper-Richtlinie in der [Richtlinie imagedigests](https://github.com/open-policy-agent/gatekeeper-library/tree/master/library/general/imagededigests) (<https://github.com/open-policy-agent/gatekeeper-library/tree/master/library/general/imagededigests>).

## Manifeste, Digests und Tags von Images kennenlernen

In diesem Abschnitt werden vorhandene Images in Registries mit Befehlszeilentools wie `curl` und `docker` untersucht. Führen Sie in diesem Abschnitt die Befehle in der Cloud-Shell oder in einer Shell-Umgebung mit bereits installierten Tools wie der `gcloud` CLI, `Docker`, `cURL` und `jq` aus. Die folgenden Befehle verwenden öffentliche Images in [Container Registry](https://container-registry) (`/container-registry`).

- Rufen Sie das Manifest des Images `gcr.io/google-containers/pause-amd64:3.2` mithilfe von `cURL` und der [Manifest-URL](https://github.com/opencontainers/distribution-spec/blob/master/spec.md) (<https://github.com/opencontainers/distribution-spec/blob/master/spec.md>) ab:

```
curl -s https://gcr.io/v2/google-containers/pause-amd64/manifests/3.2
```

Die Ausgabe sieht etwa so aus:

```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
  "config": {
    "mediaType": "application/vnd.docker.container.image.v1+json",
    "size": 759,
    "digest": "sha256:80d28bedfe5dec59da9ebf8e6260224ac9008ab5c11dbbe16ee"
  },
  "layers": [
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 296534,
      "digest": "sha256:c74f8866df097496217c9f15efe8f8d3db05d19d678a02d0"
    }
  ]
}
```

Der Abschnitt `config` hat ein `Digest`-Attribut, mit dem Sie das Konfigurationsobjekt abrufen können. Ebenso hat jedes Layer ein `digest`-Attribut, mit dem Sie die TAR-Datei für diese Ebene abrufen können.

- Wenn das Image den optionalen Image-Index enthält, gibt eine HTTP-GET-Anfrage an die Manifest-URL unter Verwendung eines Tags den Image-Index anstelle des Image-Manifests zurück.

Rufen Sie den Image-Index des Images `gcr.io/google-containers/pause:3.2` ab:

```
curl -s https://gcr.io/v2/google-containers/pause/manifests/3.2
```

Die Ausgabe sieht etwa so aus:

```
{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.list.v2+json",
  "manifests": [
    {
      "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
      "size": 526,
      "digest": "sha256:4a1c4b21597c1b4415bdbecb28a3296c6b5e23ca4f9feeb5"
```

```

    "platform": {
      "architecture": "amd64",
      "os": "linux"
    }
  },
  {
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "size": 526,
    "digest": "sha256:bbb7780ca6592cfc98e601f2a5e94bbf748a232f91165186",
    "platform": {
      "architecture": "arm",
      "os": "linux",
      "variant": "v7"
    }
  },
  {
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "size": 526,
    "digest": "sha256:31d3efd12022ffeffb3146bc10ae8beb890c80ed2f073635",
    "platform": {
      "architecture": "arm64",
      "os": "linux"
    }
  },
  {
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "size": 526,
    "digest": "sha256:7f82fec72730a6aeb70713476fb6f7545ed1bbf32cadd74",
    "platform": {
      "architecture": "ppc64le",
      "os": "linux"
    }
  },
  {
    "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
    "size": 526,
    "digest": "sha256:1175fd4d728641115e2802be80abab108b8d9306442ce354",
    "platform": {
      "architecture": "s390x",
      "os": "linux"
    }
  }
]
}

```



- Filtern Sie das Ergebnis, um den Image-Digest für die gewünschte Plattform zu extrahieren. Rufen Sie den Digest des Image-Manifests für die CPU-Architektur **amd64** und das Betriebssystem **linux** ab:

```
curl -s https://gcr.io/v2/google-containers/pause/manifests/3.2 | \
  jq -r '.manifests[] | select(.platform.architecture=="amd64" and .platf
```

Durch das Filtern in diesem Befehl wird simuliert, wie Containerlaufzeiten wie containerd (<https://github.com/containerd/containerd/blob/master/docs/content-flow.md#image-format>) das Image auswählen, das mit der Zielplattform aus dem Image-Index übereinstimmt.

Die Ausgabe sieht etwa so aus:

```
sha256:4a1c4b21597c1b4415bdbecb28a3296c6b5e23ca4f9feeb599860a1dac6a0108
```

### Der Image-Digest

(<https://github.com/opencontainers/image-spec/blob/master/descriptor.md#digests>) ist das Ergebnis der Anwendung eines kollisionssicheren Hashs ([https://wikipedia.org/wiki/Collision\\_resistance](https://wikipedia.org/wiki/Collision_resistance)) auf das Image-Manifest, typischerweise der SHA-256-Algorithmus (<https://github.com/opencontainers/image-spec/blob/master/descriptor.md#sha-256>).

- Rufen Sie den Digest des Images **gcr.io/google-containers/pause-amd64:3.2** ab:

```
curl -s https://gcr.io/v2/google-containers/pause-amd64/manifests/3.2 \
  | shasum -a 256 \
  | cut -d' ' -f1
```

Die Ausgabe sieht etwa so aus:

```
4a1c4b21597c1b4415bdbecb28a3296c6b5e23ca4f9feeb599860a1dac6a0108
```

Anhand des Image-Digest-Werts können Sie auf dieses Image verweisen:

`gcr.io/google-containers/pause-amd64@sha256:4a1c4b21597c1b4415bdbecb28a3296`

- Verwenden Sie das Konzept inhaltsadressierbarer Speicher ([https://wikipedia.org/wiki/Content-addressable\\_storage](https://wikipedia.org/wiki/Content-addressable_storage)), um das Image-Manifest abzurufen. Verwenden Sie dazu den Digest als Referenz:

```
curl -s https://gcr.io/v2/google-containers/pause-amd64/manifests/sha256:4a
```

- Viele Container Registries geben den Digest von Manifesten, Image-Indexen, Konfigurationsobjekten und Dateisystemebenen im `Docker-Content-Digest` Header als Antwort auf HTTP-HEAD-Anfragen zurück. Ruft den Digest des Image-Indexes des Image `gcr.io/google-containers/pause-amd64:3.2` ab:

```
curl -s --head https://gcr.io/v2/google-containers/pause/manifests/3.2 \
  | grep -i Docker-Content-Digest \
  | cut -d' ' -f2
```

Die Ausgabe sieht etwa so aus:

```
sha256:927d98197ec1141a368550822d18fa1c60bdae27b78b0c004f705f548c07814f
```

Der Header `Docker-Content-Digest` wurde nicht von Spezifikationen der Open Container Initiative Distribution vorgegeben

(<https://github.com/opencontainers/distribution-spec/blob/master/spec.md#checking-if-content-exists-in-the-registry>)

, sodass dieser Ansatz möglicherweise nicht mit allen Container Registries funktioniert.

Eine Verwendung mit der Container Registry (/container-registry) und Artifact Registry (/artifact-registry) ist möglich.

- So rufen Sie ein Image-Konfigurationsobjekt mithilfe des Digest-Werts aus dem Image-Manifest ab:

1. Rufen Sie den Konfigurations-Digest ab:

```
CONFIG_DIGEST=$(curl -s https://gcr.io/v2/google-containers/pause-amd64
| jq -r '.config.digest')
```

2. Verwenden Sie den Konfigurations-Digest, um das Konfigurationsobjekt abzurufen, und formatieren Sie die Ausgabe mit `jq`, um sie leichter lesbar zu machen:

```
curl -sL https://gcr.io/v2/google-containers/pause-amd64/blobs/$CONFIG_
| jq
```

Die Ausgabe sieht etwa so aus:

```
{
  "architecture": "amd64",
  "config": {
    "Env": [
      "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/b:",
    ],
    "Entrypoint": [
      "/pause"
    ],
    "WorkingDir": "/",
    "OnBuild": null
  },
  "created": "2020-02-14T10:51:50.60182885-08:00",
  "history": [
    {
      "created": "2020-02-14T10:51:50.60182885-08:00",
      "created_by": "ARG ARCH",
      "comment": "buildkit.dockerfile.v0",
      "empty_layer": true
    },
    {
      "created": "2020-02-14T10:51:50.60182885-08:00",
      "created_by": "ADD bin/pause-amd64 /pause # buildkit",
      "comment": "buildkit.dockerfile.v0"
    },
    {
      "created": "2020-02-14T10:51:50.60182885-08:00",
```

```

        "created_by": "ENTRYPOINT [\"/pause\"]",
        "comment": "buildkit.dockerfile.v0",
        "empty_layer": true
    }
],
"os": "linux",
"rootfs": {
    "type": "layers",
    "diff_ids": [
        "sha256:ba0dae6243cc9fa2890df40a625721fdbea5c94ca6da897acdd814d7"
    ]
}
}

```

- Gehen Sie so vor, um Dateisystem-Layer mithilfe von Digest-Werten aus dem Image-Manifest abzurufen:

1. Rufen Sie den Digest des Layers ab, das Sie abrufen möchten:

```
LAYER_DIGEST=$(curl -s https://gcr.io/v2/google-containers/pause-amd64,
| jq -r '.layers[0].digest')
```

2. Rufen Sie mit dem Layer-Digest die TAR-Datei des Layers ab und listen Sie die Inhalte auf:

```
curl -sL https://gcr.io/v2/google-containers/pause-amd64/blobs/$LAYER_I
| tar --list
```

Dieses Layer hat nur eine Datei namens `pause`.

- So suchen Sie Tags, die mit einem Image-Digest verknüpft sind:

1. Definieren Sie den Digest, den Sie aufrufen möchten:

```
IMAGE_DIGEST=$(curl -s https://gcr.io/v2/google-containers/pause-amd64,
| shasum -a 256 \
| cut -d' ' -f1)
```

Die Umgebungsvariable `IMAGE_DIGEST` enthält den Digest des Images, auf das das Tag 3.2 verweist.

## 2. Verwenden Sie den Endpunkt der Image-Tags-Liste `/tags/list`

(<https://github.com/opencontainers/distribution-spec/blob/master/spec.md>), um Tag-Informationen aufzulisten und die Tags für den Digest-Wert zu extrahieren:

```
curl -s "https://gcr.io/v2/google-containers/pause-amd64/tags/list?n=1" | jq ".manifest.\"sha256:$IMAGE_DIGEST\".tag"
```

Die Ausgabe sieht etwa so aus:

```
[  
  "3.2"  
]
```

- Fügen Sie ein Zugriffstoken (<https://developers.google.com/identity/protocols/oauth2>) in den Anfrageheader `Authorization` ein, um das Manifest eines Images mithilfe von cURL aus einem privaten Repository der Container Registry abzurufen:

```
curl -s -H "Authorization: Bearer $(gcloud auth print-access-token)" \  
  https://gcr.io/v2/PROJECT / IMAGE / manifests / DIGEST
```

Dabei gilt:

- *PROJECT*: Ihre Cloud-Projekt-ID
- *IMAGE*: Name Ihres Images
- *DIGEST*: Ihr Image-Digest im Format `sha256:DIGEST_VALUE`.

## Nächste Schritte

- [Container-Image-Digests mit Kubernetes-Manifesten verwenden](#)  
(/solutions/using-container-image-digests-in-kubernetes-manifests).
- [Best Practices für das Erstellen von Containern](#)  
(/solutions/best-practices-for-building-containers).
- [Best Practices für den Betrieb von Containern](#)  
(/solutions/best-practices-for-operating-containers).
- Weitere Informationen zu Images finden Sie in den Spezifikationen für das [Image-Format](#)  
(https://github.com/opencontainers/image-spec/blob/master/spec.md#image-format-specification)  
und die [Verteilung](#)  
(https://github.com/opencontainers/distribution-spec/blob/master/spec.md#distribution-specification)  
der Open Container Initiative.
- Referenzarchitekturen, Diagramme, Anleitungen und Best Practices zu Google Cloud  
entdecken. Weitere Informationen zu [Cloud Architecture Center](#) (/architecture)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](#) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](#) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](#) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2021-01-08 UTC.