Namespace: `default`

- Inspecting and modifying a Gatekeeper constraint template and constraint
- Creating a Pod that follows the constraint

IMPORTANT: OPA Gatekeeper is installed in the cluster with version 3.11.0.

Creating the Gatekeeper Constraint Template and Constraint

Inspect the following YAML manifests in the current directory:

- `allowed-repos-constraint-template.yaml`: Defines a Gatekeeper constraint template for defining allowed container registries.

```
    command: ["/bin/sh"]
root@controlplane:~$ cat allowed-repos-constraint-template.yaml
apiVersion: templates.gatekeeper.sh/v1
kind: ConstraintTemplate
metadata:
  name: k8sallowedrepos
  annotations:
    metadata.gatekeeper.sh/title: "Allowed Repositories"
    metadata.gatekeeper.sh/version: 1.0.0
    description: >-
      Requires container images to begin with a string from the specified list.
spec:
  crd:
    spec:
      names:
        kind: K8sAllowedRepos
      validation:
        openAPIV3Schema:
          type: object
          properties:
            repos:
              description: The list of prefixes a container image is allowed to have.
              type: array
              items:
                type: string
  targets:
    - target: admission.k8s.gatekeeper.sh
      rego: |
        package k8sallowedrepos

        violation[{"msg": msg}] {
          container := input.review.object.spec.containers[_]
          satisfied := [good | repo = input.parameters.repos[_] ; good = startswith(container.image, repo)]
          not any(satisfied)
          msg := sprintf("container <%v> has an invalid image repo <%v>, allowed repos are %v", [container.n
        }

        violation[{"msg": msg}] {
          container := input.review.object.spec.initContainers[_]
          satisfied := [good | repo = input.parameters.repos[_] ; good = startswith(container.image, repo)]
          not any(satisfied)
          msg := sprintf("initContainer <%v> has an invalid image repo <%v>, allowed repos are %v", [contain
        }
```

- `allowed-repos-constraint.yaml`: Specifies a list of allowed container registries for certain resource types.

```
root@controlplane:~$ cat gcr-allowed-repos-constraint.yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sAllowedRepos
metadata:
  name: repo-is-gcr
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: [""]
  parameters:
    repos:
      - ""root@controlplane:~$
```

- `pod.yaml`: Defines a Pod using a given container image.

```
    repos:
      - ""root@controlplane:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  containers:
  - name: busybox
    image: busybox:1.27.2
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo hello; sleep 10; done"]root@controlplane:~$
```

Modify the file `allowed-repos-constraint.yaml` so that container images can only be pulled from the registry with the domain prefix `gcr.io/` when a Pod is created. Create the constraint template and the constraint objects from the YAML manifests. Do not create the Pod yet.

Creating the Gatekeeper Constraint Template and Constraint

The constraint has been prepared for you. You will need to fill in the blanks. Leave the value for the attribute `spec.match.kinds[0].apiGroups` as is. The attribute value for `spec.match.kinds[0].apiGroups` should be `["Pod"]` and the attribute value for `spec.parameters.repos[0]` should be `"gcr.io/"`. The finalized YAML manifest in the file `gcr-allowed-repos-constraint.yaml` is shown here:

```
root@controlplane:~$ cat gcr-allowed-repos-constraint.yaml
apiVersion: constraints.gatekeeper.sh/v1beta1
kind: K8sAllowedRepos
metadata:
  name: repo-is-gcr
spec:
  match:
    kinds:
      - apiGroups: [""]
        kinds: ["Pod"]
  parameters:
    repos:
      - "gcr.io/"
root@controlplane:~$
```

```
root@controlplane:~$ kubectl apply -f allowed-repos-constraint-template.yaml
constrainttemplate.templates.gatekeeper.sh/k8sallowedrepos created
root@controlplane:~$ kubectl apply -f gcr-allowed-repos-constraint.yaml
k8sallowedrepos.constraints.gatekeeper.sh/repo-is-gcr created
root@controlplane:~$ kubectl get all
NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.96.0.1    <none>        443/TCP   15m
root@controlplane:~$ kubectl get constraint
NAME          ENFORCEMENT-ACTION   TOTAL-VIOLATIONS
repo-is-gcr                        18
root@controlplane:~$ kubectl get constraint/k8
error: the server doesn't have a resource type "constraint"
root@controlplane:~$ kubectl get constrainttemplate.templates.gatekeeper.sh/k8sallowedrepos
NAME             AGE
k8sallowedrepos   67s
root@controlplane:~$ kubectl get k8sallowedrepos.constraints.gatekeeper.sh/repo-is-gcr
NAME          ENFORCEMENT-ACTION   TOTAL-VIOLATIONS
repo-is-gcr                        18
root@controlplane:~$
```

Creating a Pod Following the Constraint

Modify the `pod.yaml` file in the current directory. Make sure to assign the `busybox` image from the container registry at `gcr.io/google-containers`. Wait for the Pod to transition to the Running status.

```
root@controlplane:~$ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: busybox
spec:
  containers:
  - name: busybox
    image: gcr.io/google-containers/busybox:1.27.2
    command: ["/bin/sh"]
    args: ["-c", "while true; do echo hello; sleep 10; done"]
root@controlplane:~$
```

```
root@controlplane:~$ kubectl apply -f pod.yaml
pod/busybox created
root@controlplane:~$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
busybox   1/1     Running   0          7s
root@controlplane:~$
```