Gatekeeper installation using the gatekeeper-operator

Updated May 31 2021 at 8:28 AM - English

Gatekeeper helps your openshift resource schemas meet policy standards. These will help to ensure the presence of an appropriate set of labels and annotations, security policies, or volumes.

Until version 3.3, gatekeeper was only validating that resources were in compliance with user-defined policies. With the introduction of gatekeeper mutation, it's now able to actively enforce resource schema based on user policies and validate incoming resources.

Gatekeeper can be installed by adding the gatekeeper-operator to your cluster.
This can be done by adding the following subscription.

```
---
apiVersion: operators.coreos.com/v1alpha1
kind: CatalogSource
metadata:
  name: redhat-operators
  namespace: openshift-marketplace
spec:
  displayName: Red Hat Operators
  image: "registry.redhat.io/redhat/redhat-operator-index:v4.7"
  publisher: Red Hat
  sourceType: grpc
---
apiVersion: operators.coreos.com/v1
kind: OperatorGroup
metadata:
  annotations:
    olm.providedAPIs: Gatekeeper.v1alpha1.operator.gatekeeper.sh
  name: global-operators
  namespace: openshift-operators
---
apiVersion: operators.coreos.com/v1alpha1
kind: Subscription
metadata:
  name: gatekeeper-operator-product
  namespace: openshift-operators
spec:
  channel: stable
  installPlanApproval: Automatic
  name: gatekeeper-operator-product
  source: redhat-operators
  sourceNamespace: openshift-marketplace
  startingCSV: gatekeeper-operator-product.v0.1.2
```

To add the subscription save the above description to a file and execute:

```
oc apply -f <filename>
```

Once the gatekeeper-operator is installed, create the gatekeeper resource. The gatekeeper resource is consumed by the gatekeeper-operator.

Once the gatekeeper resource has been created, the gatekeeper-operator will perform the installation based on the configuration information provided.

Here's an example of a basic gatekeeper resource with mutation support enabled:

```
apiVersion: operator.gatekeeper.sh/v1alpha1
kind: Gatekeeper
metadata:
  name: gatekeeper
spec:
  mutatingWebhook: "Enabled"
```

With the gatekeeper resource, you can control the details of the gatekeeper installation. Below is a gatekeeper example showing all possible values. Descriptions of non-obvious items have been provided in comments. Please note that the basic example provided above should suffice in most cases and the additional values shown below should only be used when a specific need arises.

```
apiVersion: operator.gatekeeper.sh/v1alpha1
kind: Gatekeeper
metadata:
  name: gatekeeper
spec:
  image:
    image: <> # Gatekeeper image. This overrides the default image used by the
operator.
    imagePullPolicy: Always # Image pull policy used in the gatekeeper pods
  audit:
    replicas: 1 # Number of gatekeeper audit pods
    logLevel: DEBUG
    auditInterval: 10s
    constraintViolationLimit: 55
    auditFromCache: Enabled
    auditChunkSize: 66
    emitAuditEvents: Enabled
    resources: # Resources needed for the audit pod
      limits:
        cpu: 100m
        memory: 20Mi
      requests:
        cpu: 100m
        memory: 20Mi
  validatingWebhook: Enabled # Enables/disables the gatekeeper validation.
  webhook:
    replicas: 2 # Number of the gatekeeper validation/mutation pods
    logLevel: ERROR
    emitAdmissionEvents: Enabled
    failurePolicy: Ignore
    namespaceSelector: # Namspace selector allowing to filter namespaces being
validated/mutated. The filtering is done at webhook level
      matchExpressions:
      - key: admission.gatekeeper.sh/enabled
        operator: Exists
    resources: # Resources needed for the gatekeeper controller pod
      limits:
        cpu: 100m
        memory: 20Mi
      requests:
        cpu: 100m
        memory: 20Mi
  nodeSelector:
    region: "EMEA"
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        - labelSelector:
            matchLabels:
              auditKey: "auditValue"
          topologyKey: topology.kubernetes.io/zone
  tolerations:
    - key: "Example"
      operator: "Exists"
      effect: "NoSchedule"
```

```
  podAnnotations:
    some-annotation: "this is a test"
    other-annotation: "another test"
```

Product(s) Red Hat Advanced Cluster Security for Kubernetes **Category** Install

Article Type General

People who viewed this article also viewed

Using gatekeeper mutation

Article - 31 mai 2021

Unable to set specific SCC using Gatekeeper

Solution - 2 mars 2022

OPA Gatekeeper is not enforcing deployment replicas using oc or kubectl scale

Solution - 20 juin 2022

Comments