拉勾教育
—互联网人实战大学—

《Kubernetes 原理剖析与实战应用》

正范

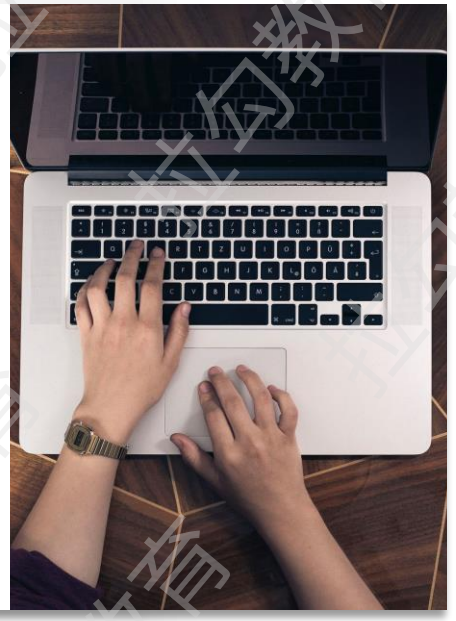
— 拉勾教育出品 —

20｜资源优化：Kubernetes 中有 GC（垃圾回收）吗？

前言

Garbage Collector 即垃圾回收，简称 GC

用来清理一些不用的资源



Kubelet GC

拉勾教育
互联网人实战大学

GC 在 Kubelet 中

不仅可以清理无用的容器，还可以清理未使用的镜像以达到节省空间的目的

Kubelet GC

Kubelet 会对容器每分钟执行一次 GC 操作

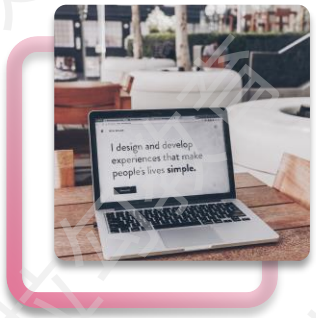
对容器镜像每 5 分钟执行一次 GC 操作避免节点出现资源紧缺的情况

启动时并不会立即执行 GC 操作

启动 1 分钟后开始执行第一次对容器的 GC 操作

启动 5 分钟后开始执行第一次对容器镜像的回收操作

Kubelet GC



--minimum-image-ttl-duration

表示一个镜像在清理前的最小存活时间

--image-gc-high-threshold

表示磁盘使用率的上限阈值，默认值是 90%

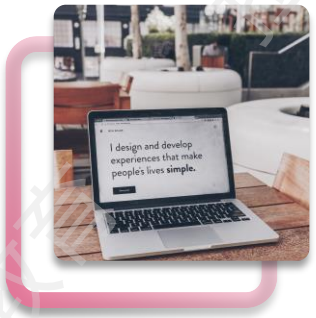
即当磁盘使用率达到 90% 的时候会触发对镜像的 GC 操作

--image-gc-low-threshold

表示磁盘使用率的下限阈值，默认值是 80%

即当磁盘使用率降到 80% 的时候，GC 操作结束

Kubelet GC



--minimum-container-ttl-duration

表示已停止的容器在被清理之前最小的存活时间
默认值是 1 分钟
即容器停止超过 1 分钟才会被标记可被 GC 清理

--maximum-dead-containers-per-container

表示一个 Pod 内可以保留的已停止的容器数量，默认值是 2

Kubernetes 是以 Pod 为单位进行容器管理的

--maximum-dead-containers

表示在本节点上可以保留的已停止容器的最大数量
默认值是240

Kubelet GC

强烈建议将 **--maximum-dead-containers-per-container 设置为一个足够大的值**

以便每个容器至少有一个退出的实例

https://kubernetes.io/zh/docs/concepts/cluster-administration/kubelet-garbage-collection/#deprecation

Kubernetes 内部对象的 GC

创建好一个 Deployment 以后

kube-controller-manager 会创建对应的 **ReplicaSet**

Kubernetes 内部对象的 GC

拉勾教育
互联网人实战大学

```
// OwnerReference contains enough information to let you identify an owning
// object. An owning object must be in the same namespace as the dependent, or
// be cluster-scoped, so there is no namespace field.
type OwnerReference struct {
    // API version of the referent.
    APIVersion string `json:"apiVersion" protobuf:"bytes,5,opt,name=apiVersion"`
    // Kind of the referent.
    // More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
    Kind string `json:"kind" protobuf:"bytes,1,opt,name=kind"`
    // Name of the referent.
    // More info: http://kubernetes.io/docs/user-guide/identifiers#names
    Name string `json:"name" protobuf:"bytes,3,opt,name=name"`
    // UID of the referent.
    // More info: http://kubernetes.io/docs/user-guide/identifiers#uids
    UID types.UID `json:"uid" protobuf:"bytes,4,opt,name=uid,casttype=k8s.io/apimachinery/pkg/types.UID"`
    // If true, this reference points to the managing controller.
    // +optional
```

Kubernetes 内部对象的 GC


```
    // More info: http://kubernetes.io/docs/user-guide/identifiers#names
    Name string `json:"name" protobuf:"bytes,3,opt,name=name"`
    // UID of the referent.
    // More info: http://kubernetes.io/docs/user-guide/identifiers#uids
    UID types.UID `json:"uid"
protobuf:"bytes,4,opt,name=uid,casttype=k8s.io/apimachinery/pkg/types.UID"`
    // If true, this reference points to the managing controller.
    // +optional
    Controller *bool `json:"controller,omitempty" protobuf:"varint,6,opt,name=controller"`
    // If true, AND if the owner has the "foregroundDeletion" finalizer, then
    // the owner cannot be deleted from the key-value store until this
    // reference is removed.
    // Defaults to false.
    // To set this field, a user needs "delete" permission of the owner,
    // otherwise 422 (Unprocessable Entity) will be returned.
    // +optional
    BlockOwnerDeletion *bool `json:"blockOwnerDeletion,omitempty"
protobuf:"varint,7,opt,name=blockOwnerDeletion"`
}
```

Kubernetes 内部对象的 GC


```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 annotations:
  deployment.kubernetes.io/desired-replicas: "2"
  deployment.kubernetes.io/max-replicas: "3"
  deployment.kubernetes.io/revision: "1"
 creationTimestamp: "2020-09-03T07:22:35Z"
 generation: 1
 labels:
  k8s-app: kube-dns
  pod-template-hash: 5644d7b6d9
 name: coredns-5644d7b6d9
 namespace: kube-system
 ownerReferences:
 - apiVersion: apps/v1
  blockOwnerDeletion: true
  controller: true
  kind: Deployment
  name: coredns
  uid: 37ae660a-dba8-4ff9-a152-7d6f420e624d
```

Kubernetes 内部对象的 GC


```
  generation: 1
  labels:
    k8s-app: kube-dns
    pod-template-hash: 5644d7b6d9
  name: coredns-5644d7b6d9
  namespace: kube-system
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: Deployment
    name: coredns
    uid: 37ae660a-dba8-4ff9-a152-7d6f420e624d
  resourceVersion: "1542272"
  selfLink: /apis/apps/v1/namespaces/kube-system/replicasets/coredns-5644d7b6d9
  uid: fa3d9859-43d4-484b-9716-7536243acd0f
spec:
  replicas: 2
  ...
status:
  ...
```

Kubernetes 内部对象的 GC


```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: "2020-09-03T07:22:35Z"
  generateName: coredns-5644d7b6d9-
  labels:
    k8s-app: kube-dns
    pod-template-hash: 5644d7b6d9
  name: coredns-5644d7b6d9-sz4qj
  namespace: kube-system
  ownerReferences:
  - apiVersion: apps/v1
    blockOwnerDeletion: true
    controller: true
    kind: ReplicaSet
    name: coredns-5644d7b6d9
    uid: fa3d9859-43d4-484b-9716-7536243acd0f
  resourceVersion: "1542270"
  selfLink: /api/v1/namespaces/kube-system/pods/coredns-5644d7b6d9-sz4qj
  uid: c52d630b-1840-4502-88d1-b67bed2dd625
spec:
  ...
```

Kubernetes 内部对象的 GC

Deployment（owner）—> ReplicaSet (dependent)

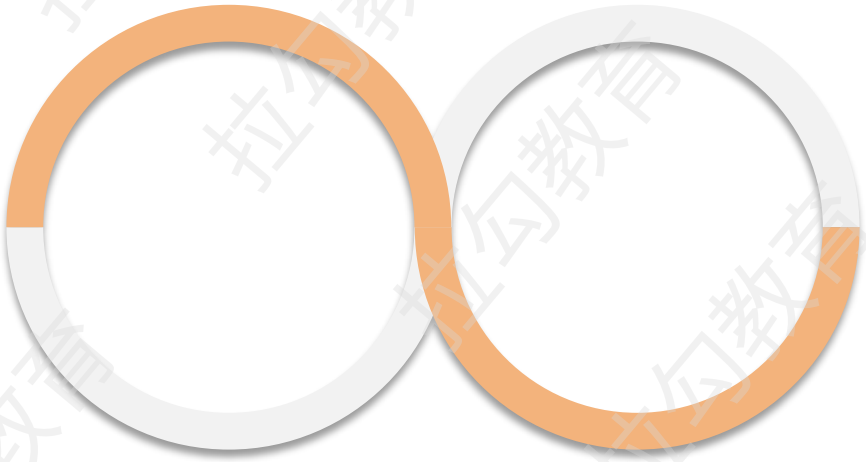
ReplicaSet (owner) —> Pod (dependent)

Kubernetes 内部对象的 GC

拉勾教育
互联网人实战大学

Kubernetes 两种模式

后台（Background）模式



前台（Foreground）模式

Kubernetes 内部对象的 GC

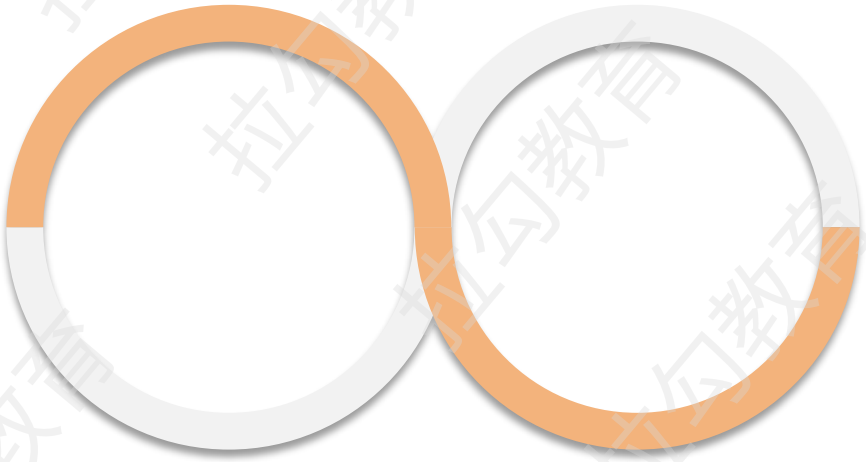
拉勾教育
互联网人实战大学

Kubernetes 两种模式

后台（Background）模式

发送完请求

Kuberentes 会立即删除主对象



前台（Foreground）模式

Kubernetes 内部对象的 GC

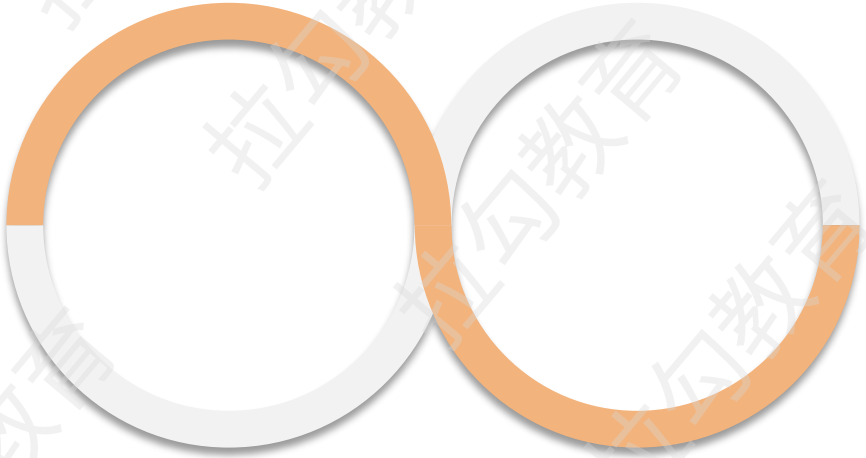
拉勾教育
互联网人实战大学

Kubernetes 两种模式

后台（Background）模式

发送完请求

Kuberentes 会立即删除主对象



前台（Foreground）模式

先删除其所属的对象

然后再删除主对象

Kubernetes 内部对象的 GC

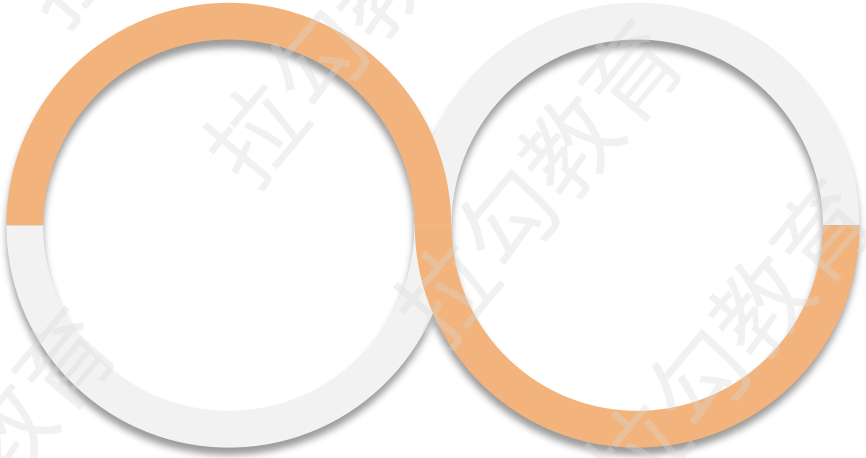
拉勾教育
互联网人实战大学

Kubernetes 两种模式

后台（Background）模式

发送完请求

Kuberentes 会立即删除主对象



前台（Foreground）模式

先删除其所属的对象

然后再删除主对象

Kubernetes 内部对象的 GC

如果删除对象时，并不想自动删除其附属对象

那么这些附属对象就“孤立”存在了，即**孤立对象（Orphaned）**

Kubernetes 内部对象的 GC


```
$ kubectl proxy --port=8080
$ curl -X DELETE localhost:8080/apis/apps/v1/namespaces/default/replicasets/my-replicaset \
  -d '{"kind":"DeleteOptions","apiVersion":"v1","propagationPolicy":"Background"}' \
  -H "Content-Type: application/json"
```

Kubernetes 内部对象的 GC

拉勾教育
互联网人实战大学

```
$ kubectl proxy --port=8080
$ curl -X DELETE localhost:8080/apis/apps/v1/namespaces/default/replicasets/my-replicaset \
  -d '{"kind":"DeleteOptions","apiVersion":"v1","propagationPolicy":"Foreground"}' \
  -H "Content-Type: application/json"
```

Kubernetes 内部对象的 GC

拉勾教育
互联网人实战大学

```
$ kubectl proxy --port=8080
$ curl -X DELETE localhost:8080/apis/apps/v1/namespaces/default/replicasets/my-repset \
  -d '{"kind":"DeleteOptions","apiVersion":"v1","propagationPolicy":"Orphan"}' \
  -H "Content-Type: application/json"
```

Kubernetes 内部对象的 GC

拉勾教育
互联网人实战大学

```
$ kubectl delete replicaset my-repset --cascade=false
```

写在最后

Kubernetes 默认开启了 GC 的能力

不管是对于内部的各种 API 对象

还是对于 kubelet 节点上的冗余镜像以及退出的容器

Next：《21 | 优先级调度：你必须掌握的 Pod 抢占式资源调度》

拉勾教育

—互联网人实战大学—

关注拉勾「教育公众号」
获取更多课程信息

L / A / G / O / U