**加我微信可进群学习交流：**

**微信号：**

luckylucky421302

也可通过扫描下面二维码添加



**课程更新的知识点会通过微信公众号免费分享给大家，可以关注我的公众号**

****

**下文需要的gb-frontend.tar.gz镜像包在百度网盘地址如下：**

链接：https://pan.baidu.com/s/1RjJiRYDjXK3UaPRIrIKV-A

提取码：387q

**1.Replicaset概念**

Kubernetes中的ReplicaSet主要的作用是维持一组[Pod](https://draveness.me/kubernetes-pod)副本的运行，它的主要作用就是保证一定数量的 Pod 能够在集群中正常运行，它会持续监听这些 Pod 的运行状态，在 Pod 发生故障时重启pod，pod数量减少时重新运行新的 Pod 副本，因此，它通常被用来保证特定数量相同的Pods的可用性。

**2.replicaset怎么工作**

ReplicaSet由字段定义，包括一个选择器，该选择器指定如何找到它所管理的Pod、维护多少个pod，以及pod的模板。ReplicaSet通过创建和删除Pod来满足期望的pod数量。当ReplicaSet需要创建新的Pod时，它将使用其Pod模板。ReplicaSet通过Pods的[metadata.ownerReferences](https://kubernetes.io/docs/concepts/workloads/controllers/garbage-collection/" \l "owners-and-dependents)字段链接到其Pod，该字段指定当前对象所拥有的资源。由ReplicaSet获取的所有Pod在其ownerReferences字段中都有其自己的ReplicaSet的标识信息。通过此链接，ReplicaSet可以知道它正在维护的Pod的状态，并据此计划。

ReplicaSet通过使用其选择器标识要获取的新Pod。如果存在没有OwnerReference的Pod或OwnerReference不是[控制器](https://kubernetes.io/docs/concepts/architecture/controller/" \t "_blank)，并且它与ReplicaSet的选择器匹配，它将由所述的ReplicaSet立即获取

**3.什么时候使用replicaset**

replicaSet确保在任何给定时间都运行指定数量的Pod副本。但是，Deployment是一个高级概念，用于管理副本集，并提供对Pod的声明性更新以及许多其他有用的功能。因此，除非你需要自定义更新编排或根本不需要更新，否则我们使用Deployment而不是直接使用replicaset。这实际上意味着你可能永远不需要操纵ReplicaSet对象：改用Deployment，然后在spec部分中定义你的应用程序。

**4.例子，使用replicaset部署一个应用**

cat replicaset.yaml

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: frontend

labels:

app: guestbook

tier: frontend

spec:

replicas: 3

selector:

matchLabels:

tier: frontend

template:

metadata:

labels:

tier: frontend

spec:

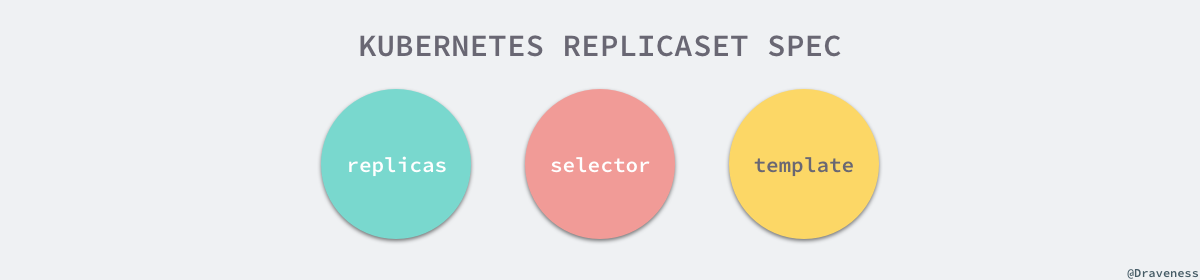
containers:

- name: php-redis

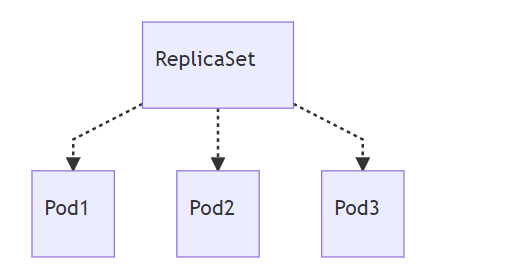
image: yecc/gcr.io-google\_samples-gb-frontend:v3

kubectl apply -f replicaset.yaml

这里的YAML文件除了常见的apiVersion、kind和metadata属性之外，规格中总共包含三部分重要内容，也就是 Pod 副本数目 replicas、选择器 selector 和 Pod 模板 template，这三个部分共同定义了 ReplicaSet 的规格：



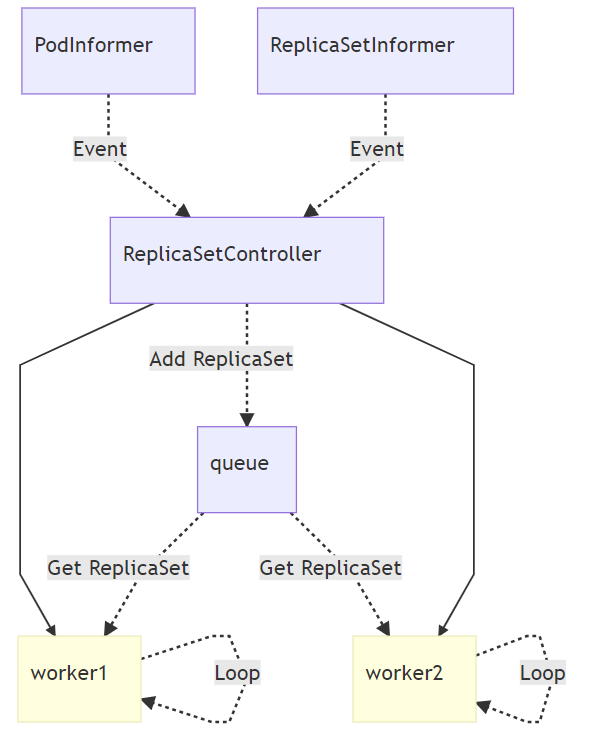
同一个 ReplicaSet 会使用选择器 selector 中的定义查找集群中自己持有的 Pod 对象，它们会根据标签的匹配获取能够获得的 Pod，下面就是持有三个 Pod 对象的 Replica 拓扑图：



被 ReplicaSet 持有的 Pod 有一个metadata.ownerReferences指针指向当前的 ReplicaSet，表示当前 Pod 的所有者，这个引用主要会被集群中的[垃圾收集器](https://draveness.me/kubernetes-garbage-collector)使用以清理失去所有者的 Pod 对象。

**5.实现原理**

所有 ReplicaSet 对象的增删改查都是由 ReplicaSetController 控制器完成的，该控制器会通过 Informer 监听 ReplicaSet 和 Pod 的变更事件并将其加入持有的待处理队列:



ReplicaSetController 中的 queue 其实就是一个存储待处理 ReplicaSet 的『对象池』，它运行的几个 Goroutine 会从队列中取出最新的数据进行处理，上图展示了事件从发生到被处理的流向。

**6.编写replicaset资源清单文件**

与所有其他 Kubernetes API对象一样，ReplicaSet需要apiVersion、kind、和metadata，ReplicaSet 也需要[.spec](https://git.k8s.io/community/contributors/devel/api-conventions.md" \l "spec-and-status" \t "_blank)部分。对于replicaset副本集来说，kind只能是replicaset，在kubernetes1.9+之后，apiversion默认的版本是apps/v1，apps/v1beta2已经被废弃了。

**Pod template**

.spec.template是一个[Pod模板](https://kubernetes.io/docs/concepts/workloads/Pods/pod-overview/" \l "pod-templates)，还需要在其上放置标签。在我们的replicaset.yaml示例中，我们有一个标签：tier: frontend。注意不要与其他控制器的选择器重叠。对于 [重启策略](https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#restart-policy)，.spec.template.spec.restartPolicy 唯一允许的取值是 Always，这也是默认值.

Pod模板参考：

<https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/#pod-templates>

重启策略参考：

<https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#restart-policy>

**pod selector**

.spec.selector字段是[标签选择器](https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/)。可以选择它所匹配的拥有相同标签的pod。在我们的 replicaset.yaml示例中，选择器为：

matchLabels:

tier: frontend

在ReplicaSet中，.spec.template.metadata.labels必须匹配spec.selector，否则将被API拒绝。

**注意：**对于指定相同.spec.selector但不同的.spec.template.metadata.labels和.spec.template.spec字段的2个replicaset，每个replicaset都会忽略另一个replicaset创建的Pod。

**Replicas**

通过设置 .spec.replicas 您可以指定要同时运行多少个 Pod。 在任何时间运行的 Pod 数量可能高于或低于 .spec.replicas 指定的数量，例如在副本刚刚被增加或减少后、或者 Pod 正在被优雅地关闭、以及替换提前开始。

如果您没有指定 .spec.replicas, 那么默认值为 1

**7.删除**

如果我们在 Kubernetes 集群中删除一个 ReplicaSet 持有的 Pod，那么控制器会重新同步 ReplicaSet 的状态并启动一个新的 Pod，但是如果删除集群中的 ReplicaSet 所有相关的 Pod 也都会被删除：

kubectl delete rs example

**8.总结**

Kubernetes 中的 ReplicaSet 并不是一个工程师经常需要直接接触的对象，常用的 Deployment ，其实使用 ReplicaSet 实现了很多复杂的特性，例如滚动更新，虽然作为使用者我们并不会经常直接与 ReplicaSet 这一对象打交道，但是如果需要对 Kubernetes 进行一些定制化开发，可能会用 ReplicaSet 和其他对象实现一些更复杂的功能。

[Deployment](https://kubernetes.io/docs/concepts/workloads/controllers/deployment/) 是一个高级 API 对象，它以 kubectl rolling-update 的方式更新其底层replicaset及其Pod，虽然ReplicaSets可以独立使用，但如今，它们主要由Deployments用作协调Pod创建，删除和更新的机制，使用deployment时，你不必担心管理它们创建的replicaset。部署拥有并管理其副本集。因此，建议你在需要副本集时使用deployment。