Job

Job 负责批处理任务，即仅执行一次的任务，它保证批处理任务的一个或多个 Pod 成功结束

特殊说明

- **spec.template格式同Pod**
- **RestartPolicy仅支持Never或OnFailure**
- **单个Pod时，默认Pod成功运行后Job即结束**
- `.spec.completions` **标志Job结束需要成功运行的Pod个数，默认为1**
- `.spec.parallelism` **标志并行运行的Pod的个数，默认为1**
- `spec.activeDeadlineSeconds` **标志失败Pod的重试最大时间，超过这个时间不会继续重试**

Example

```
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    metadata:
      name: pi
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi", "-wle", "print bpi(2000)"]
      restartPolicy: Never
```

CronJob Spec

- **spec.template格式同Pod**
- **RestartPolicy仅支持Never或OnFailure**
- **单个Pod时，默认Pod成功运行后Job即结束**
- `.spec.completions` **标志Job结束需要成功运行的Pod个数，默认为1**
- `.spec.parallelism` **标志并行运行的Pod的个数，默认为1**
- `spec.activeDeadlineSeconds` **标志失败Pod的重试最大时间，超过这个时间不会继续重试**

CronJob

Cron Job 管理基于时间的 Job，即：

- 在给定时间点只运行一次
- 周期性地在给定时间点运行

使用条件：当前使用的 Kubernetes 集群，版本 >= 1.8（对 CronJob）

典型的用法如下所示：

- 在给定的时间点调度 Job 运行
- 创建周期性运行的 Job，例如：数据库备份、发送邮件

CronJob Spec

- `.spec.schedule`：调度，必需字段，指定任务运行周期，格式同 Cron
- `.spec.jobTemplate`：Job 模板，必需字段，指定需要运行的任务，格式同 Job
- `.spec.startingDeadlineSeconds` ：启动 Job 的期限（秒级别），该字段是可选的。如果因为任何原因而错过了被调度的时间，那么错过执行时间的 Job 将被认为是失败的。如果没有指定，则没有期限
- `.spec.concurrencyPolicy`：并发策略，该字段也是可选的。它指定了如何处理被 Cron Job 创建的 Job 的并发执行。只允许指定下面策略中的一种：
  - `Allow`（默认）：允许并发运行 Job
  - `Forbid`：禁止并发运行，如果前一个还没有完成，则直接跳过下一个
  - `Replace`：取消当前正在运行的 Job，用一个新的来替换

  注意，当前策略只能应用于同一个 Cron Job 创建的 Job。如果存在多个 Cron Job，它们创建的 Job 之间总是允许并发运行。
- `.spec.suspend` ：挂起，该字段也是可选的。如果设置为 `true`，后续所有执行都会被挂起。它对已经开始执行的 Job 不起作用。默认值为 `false`。
- `.spec.successfulJobsHistoryLimit` 和 `.spec.failedJobsHistoryLimit` ：历史限制，是可选的字段。它们指定了可以保留多少完成和失败的 Job。默认情况下，它们分别设置为 `3` 和 `1`。设置限制的值为 `0`，相关类型的 Job 完成后将不会被保留。

Example

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
```

```
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

```
$ kubectl get cronjob
NAME      SCHEDULE      SUSPEND   ACTIVE    LAST-SCHEDULE
hello     */1 * * * *   False     0         <none>
$ kubectl get jobs
NAME               DESIRED   SUCCESSFUL   AGE
hello-1202039034   1         1            49s
$ pods=$(kubectl get pods --selector=job-name=hello-1202039034 --output=jsonpath=
{.items..metadata.name})
$ kubectl logs $pods
Mon Aug 29 21:34:09 UTC 2016
Hello from the Kubernetes cluster

# 注意，删除 cronjob 的时候不会自动删除 job，这些 job 可以用 kubectl delete job 来删除
$ kubectl delete cronjob hello
cronjob "hello" deleted
```

CrondJob 本身的一些限制

创建 Job 操作应该是 *幂等的*