

Chap. 04

Replication and other controllers

黃舟淵 洪胤勛

```
git clone https://github.com/luksa/kubernetes-in-action.git  
cd kubernetes-in-action/Chapter04
```

Liveness probes

- **HTTP GET probe**

send HTTP GET request, wait for HTTP response

- **TCP socket probe**

open TCP connection

- **Exec probe**

execute an arbitrary command and check exit code

```
apiVersion: v1
kind: pod
metadata:
  name: kubia-liveness
```

```
spec:
```

```
  containers:
```

```
  - image: luksa/kubia-unhealthy
```

```
    name: kubia
```

```
    livenessProbe:
```

```
      httpGet:
```

```
        path: /
```

```
        port: 8080
```

**This is the image
containing the
(somewhat)
broken app.**



**A liveness probe that will
perform an HTTP GET**



**The path to
request in the
HTTP request**



**The network port
the probe should
connect to**



pod's describe

- **initialDelaySeconds**: 容器啟動後幾秒之後進行第一次探測任務。
- **periodSeconds**: 執行探測的頻率間隔, 默認是10秒, 最小1秒。
- **timeoutSeconds**: 探測超時時間, 默認1秒, 最小1秒。
- **successThreshold**: 繼上次失敗後, 最少連續探測成功次數才被認定為成功, 默認是 1;
 - 對於 **liveness** 必須是 1, 最小值是 1。
- **failureThreshold**: 連續探測失敗多少次才被認定為失敗, 默認是 3, 最小值是 1;
 - **liveness** 達標情況下將重啟 pod。
 - **readiness** 在此標準下, 認定 pod 未準備就緒。

```
$ kubectl describe po kubia-liveness
```

```
Name: kubia-liveness
```

```
...
```

```
Containers:
```

```
  kubia:
```

```
    Container ID: docker://480986f8
```

```
    Image: luksa/kubia-unhealthy
```

```
    Image ID: docker://sha256:2b208508
```

```
    Port:
```

```
    State: Running
```

```
      Started: Sun, 14 May 2017 11:41:40 +0200
```

The container is currently running.

```
    Last State: Terminated
```

```
      Reason: Error
```

```
      Exit Code: 137
```

```
      Started: Mon, 01 Jan 0001 00:00:00 +0000
```

```
      Finished: Sun, 14 May 2017 11:41:38 +0200
```

The previous container terminated with an error and exited with code 137.

```
    Ready: True
```

```
    Restart Count: 1
```

```
    Liveness: http-get http://:8080/ delay=0s timeout=1s
              period=10s #success=1 #failure=3
```

The container has been restarted once.

```
...
```

```
Events:
```

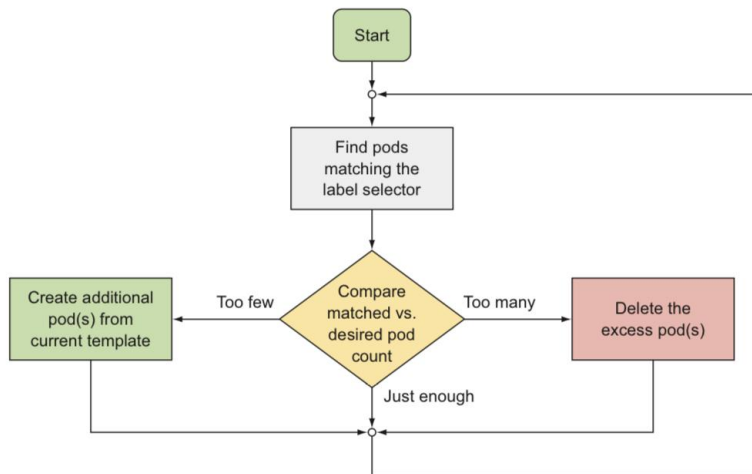
```
... Killing container with id docker://95246981:pod "kubia-liveness ..."
    container "kubia" is unhealthy, it will be killed and re-created.
```

Effective liveness probes

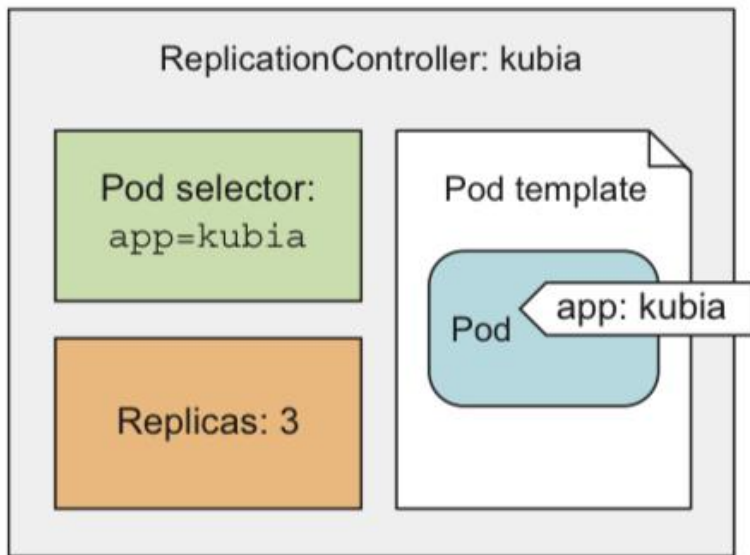
- Specific URL(/health)
- Keep probes light(short)
- implement retry loops
- liveness Probs wrap-up

Replication Controllers

make sure that an exact number of pods always matches its label selector.



struct of ReplicationController



A label selector, which determines what pods are in the ReplicationController's scope

A replica count, which specifies the desired number of pods that should be running

A pod template, which is used when creating new pod replicas

YAML definition of ReplicationController

```
apiVersion: v1
```

```
kind: ReplicationController
```

```
metadata:
```

```
  name: kubia
```

```
spec:
```

```
  replicas: 3
```

```
  selector:
```

```
    app: kubia
```

```
template:
```

```
  metadata:
```

```
    labels:
```

```
      app: kubia
```

```
  spec:
```

```
    containers:
```

```
      - name: kubia
```

```
        image: luksa/kubia
```

```
        ports:
```

```
          - containerPort: 8080
```

This manifest defines a
ReplicationController (RC)

The name of this
ReplicationController

The desired number
of pod instances

The pod selector determining
what pods the RC is operating on

The pod template
for creating new
pods

kubia-rc.yaml

Creating new Pods

```
$ kubectl describe rc kubia
```

```
Name:          kubia
Namespace:     default
Selector:      app=kubia
Labels:        app=kubia
Annotations:    <none>
Replicas:      3 current / 3 desired
Pods Status:   4 Running / 0 Waiting / 0 Succeeded / 0 Failed
Pod Template:
  Labels:      app=kubia
  Containers:  ...
```

The actual vs. the
desired number of
pod instances

Number of
pod instances
per pod
status

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
kubia-53thy	1/1	Terminating	0	3m
kubia-oini2	0/1	ContainerCreating	0	2s
kubia-k0xz6	1/1	Running	0	3m
kubia-q3vkg	1/1	Running	0	3m

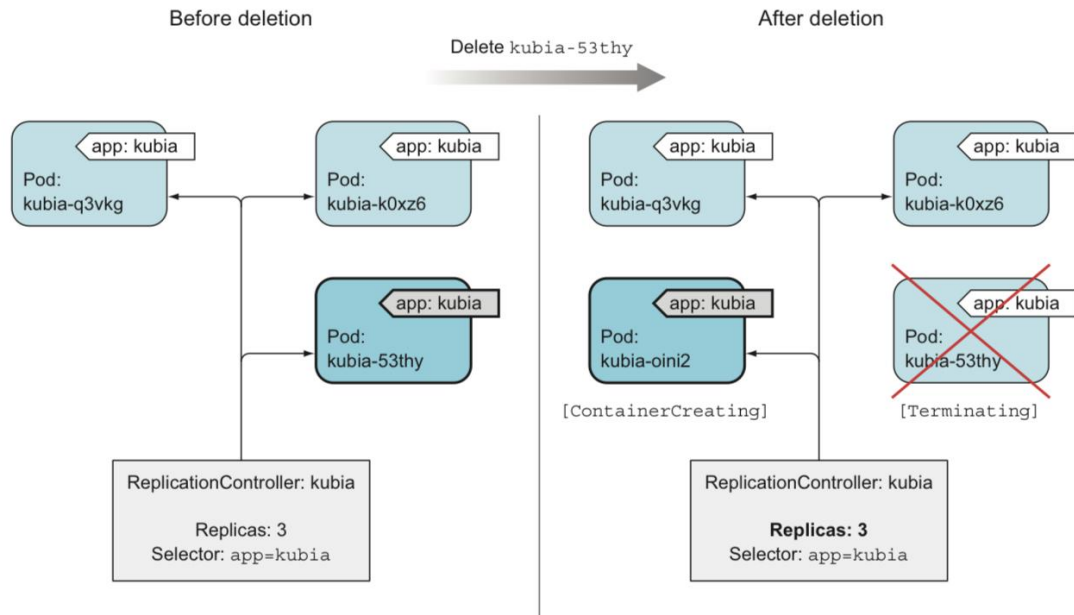


Figure 4.4 If a pod disappears, the ReplicationController sees too few pods and creates a new replacement pod.

The controller creates a new pod, doesn't respond to the deletion itself, but the resulting state – the inadequate number of pods(or the unfit of the number of pods)

Relabelling

\$ kubectl label pod kubia-dmdck app=foo – overwrite

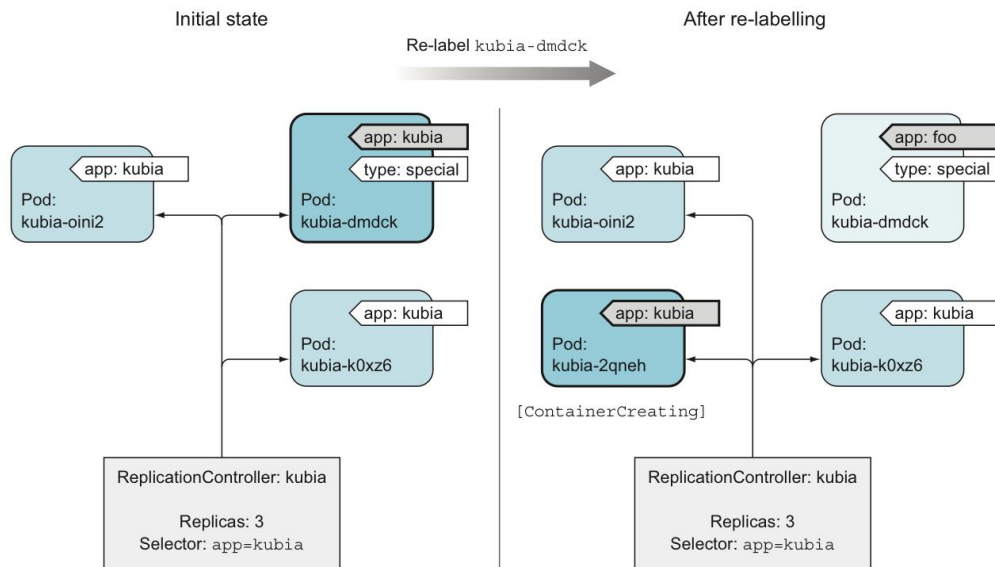
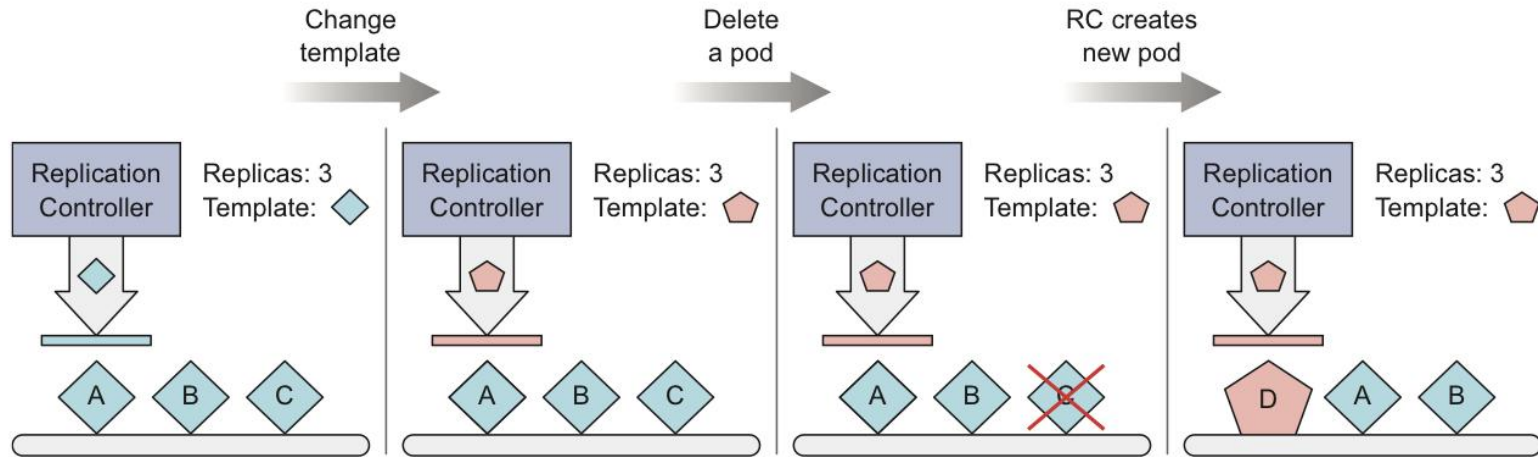


Figure 4.5 Removing a pod from the scope of a ReplicationController by changing its labels

Changing pods template

\$ kubectl edit rc kubia



Horizontally scaling pods

```
$ kubectl scale rc kuba -replicas=10
```

```
$ kubectl edit rc kuba
```

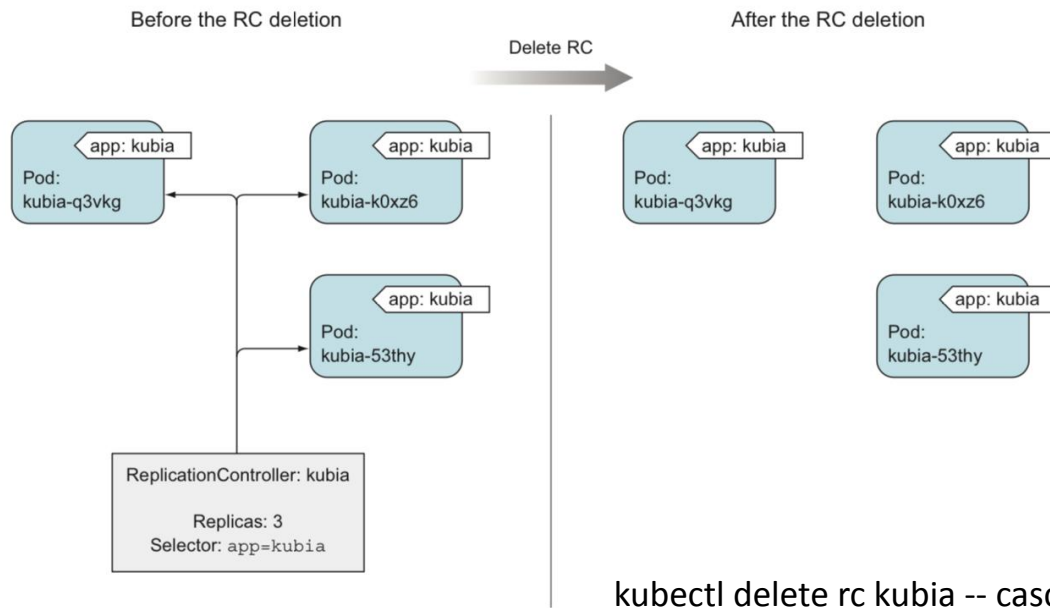
```
metadata:  
  ...  
spec:  
  replicas: 3  
  selector:  
    app: kuba  
  ...
```

← Change the number 3
to number 10 in
this line.

```
$ kubectl get rc
```

NAME	DESIRED	CURRENT	READY	AGE
kuba	10	10	4	21m

Deleting Replication controller



kubectl delete rc kubia -- cascade=false

ReplicaSets

- 比 ReplicationControllers 有更多的 label selector 的設定
 - 一個 key, 多個 value
 - env=production & env=devel
 - 選定 label 的 key, 不管 value
 - env=*

ReplicaSets

```
apiVersion: apps/v1beta2
```

```
kind: ReplicaSet
```

```
metadata:
```

```
  name: kubia
```

```
spec:
```

```
  replicas: 3
```

```
  selector:
```

```
    matchLabels:
```

```
      app: kubia
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: kubia
```

```
    spec:
```

```
      containers:
```

```
        - name: kubia
```

```
          image: luksa/kubia
```

kubia-replicaset.yaml

spec.selector.matchLabels

跟 ReplicationController 的 selector 一樣

pod 的 template

ReplicaSets

```
apiVersion: apps/v1beta2
kind: ReplicaSet
metadata:
```

```
  name: kubia
```

```
spec:
```

```
  replicas: 3
```

```
  selector:
```

```
    matchExpressions:
```

```
      - key: app
```

```
        operator: In
```

```
        values:
```

```
          - kubia
```

```
  template:
```

```
    metadata:
```

```
      labels:
```

```
        app: kubia
```

```
    spec:
```

```
      containers:
```

```
        - name: kubia
```

```
          image: luksa/kubia
```

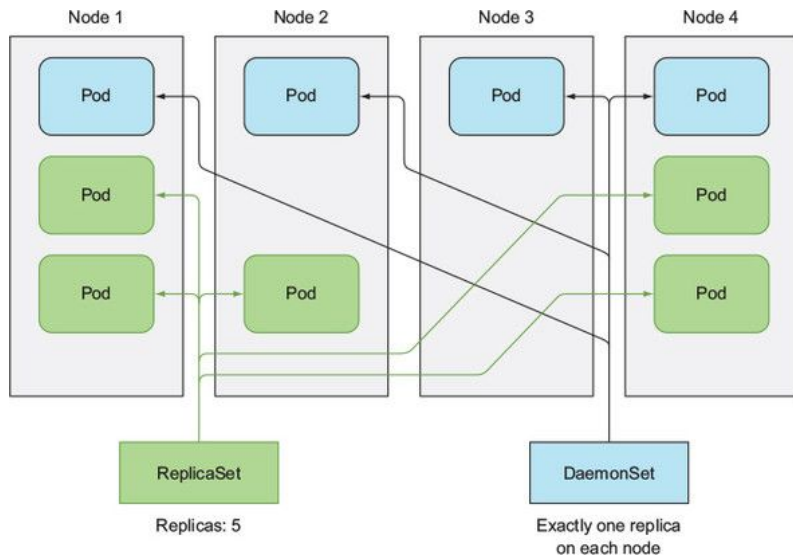
kubia-replicaset-matchexpressions.yaml

- matchExpressions.operator
 - In: label 的 value 必須跟指定的一樣
 - NotIn: label 的 value 必須跟指定的不一樣
 - Exists: label 的 key 一樣 (不用指定 values)
 - DoesNotExist: label 的 key 不一樣 (不用指定 values)

pod 的 template

DaemonSets

在每個 node 上佈署一個 pod



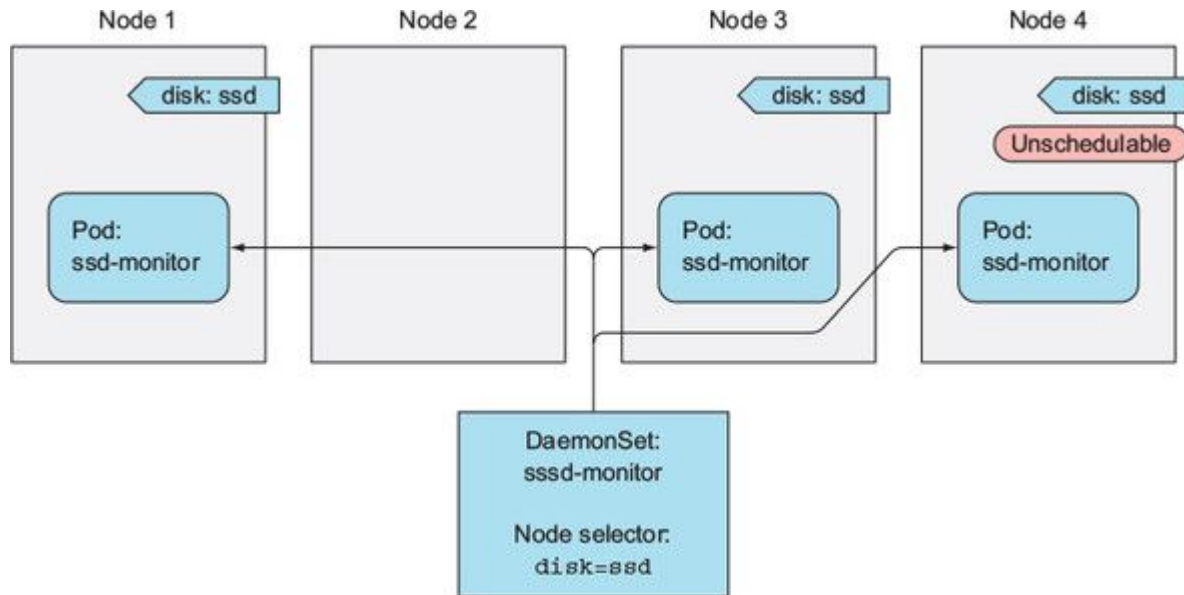
使用時機：收集每個 node 上的 log, 或 node 上的 monitor

DaemonSets

- 不像 ReplicaSet/ReplicaController, 可以指定要多少個 replicas
- 如果有 node 離開 cluster, DaemonSet 不會在別的 node 上建立新 pod
- 如果有新的 node 加入 cluster, DaemonSet 會在那個 node 上建立 pod
- 如果不小心把 pod 刪除了, DaemonSet 會重新建立新的 pod

DaemonSets

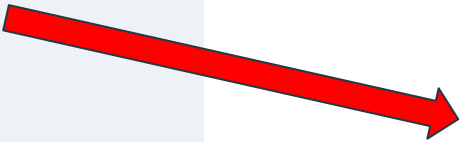
- DaemonSet 預設在所有 node 上建立 pod
- 如要指定某些 node, 則用 nodeSelector (使用 label 篩選)



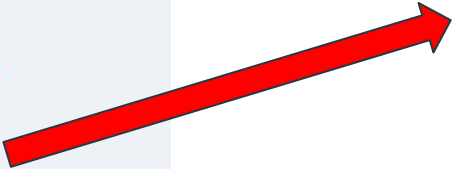
DaemonSets

```
apiVersion: apps/v1beta2
kind: DaemonSet
metadata:
  name: ssd-monitor
spec:
  selector:
    matchLabels:
      app: ssd-monitor
  template:
    metadata:
      labels:
        app: ssd-monitor
    spec:
      nodeSelector:
        disk: ssd
      containers:
        - name: main
          image: luksa/ssd-monitor
```

ssd-monitor-daemonset.yaml



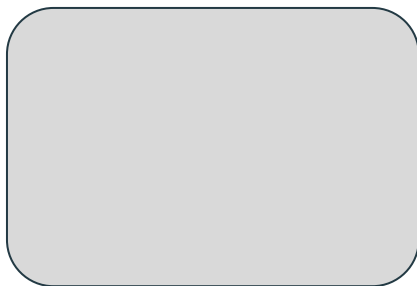
指定 pod 的 label



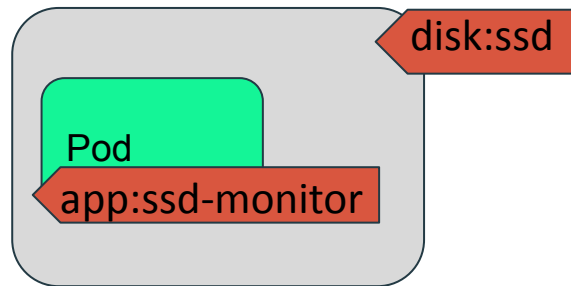
指定哪個 node

DaemonSets

1. `kubectl create -f ssd-monitor-daemonset.yaml`
 - 1.1. `kubectl get ds` # ds for DaemonSet
 - 1.2. `kubectl get po`
2. `kubectl label node minikube disk=ssd`
 - 2.1. `kubectl get po`



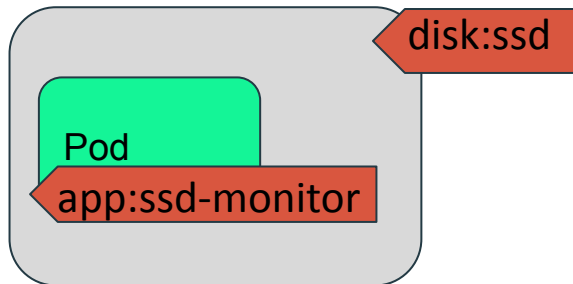
minikube



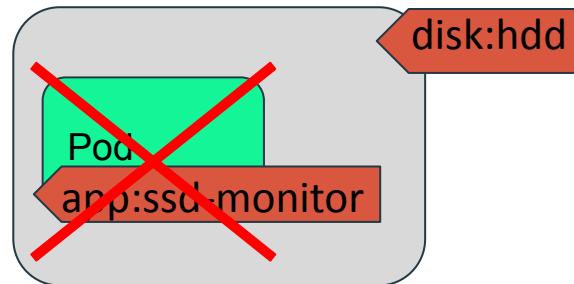
minikube

DaemonSets

- `kubectl label node minikube disk=hdd --overwrite`
- `kubectl get po`



minikube



minikube

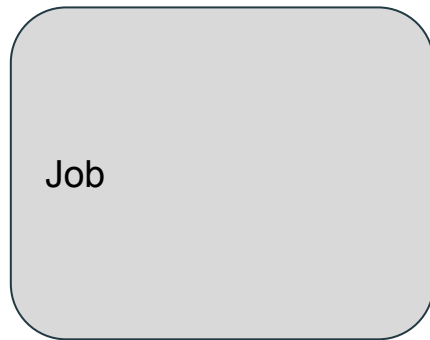
Job



continuous tasks

K8s 保持 pod 的運行
pod 不會自己停止運作

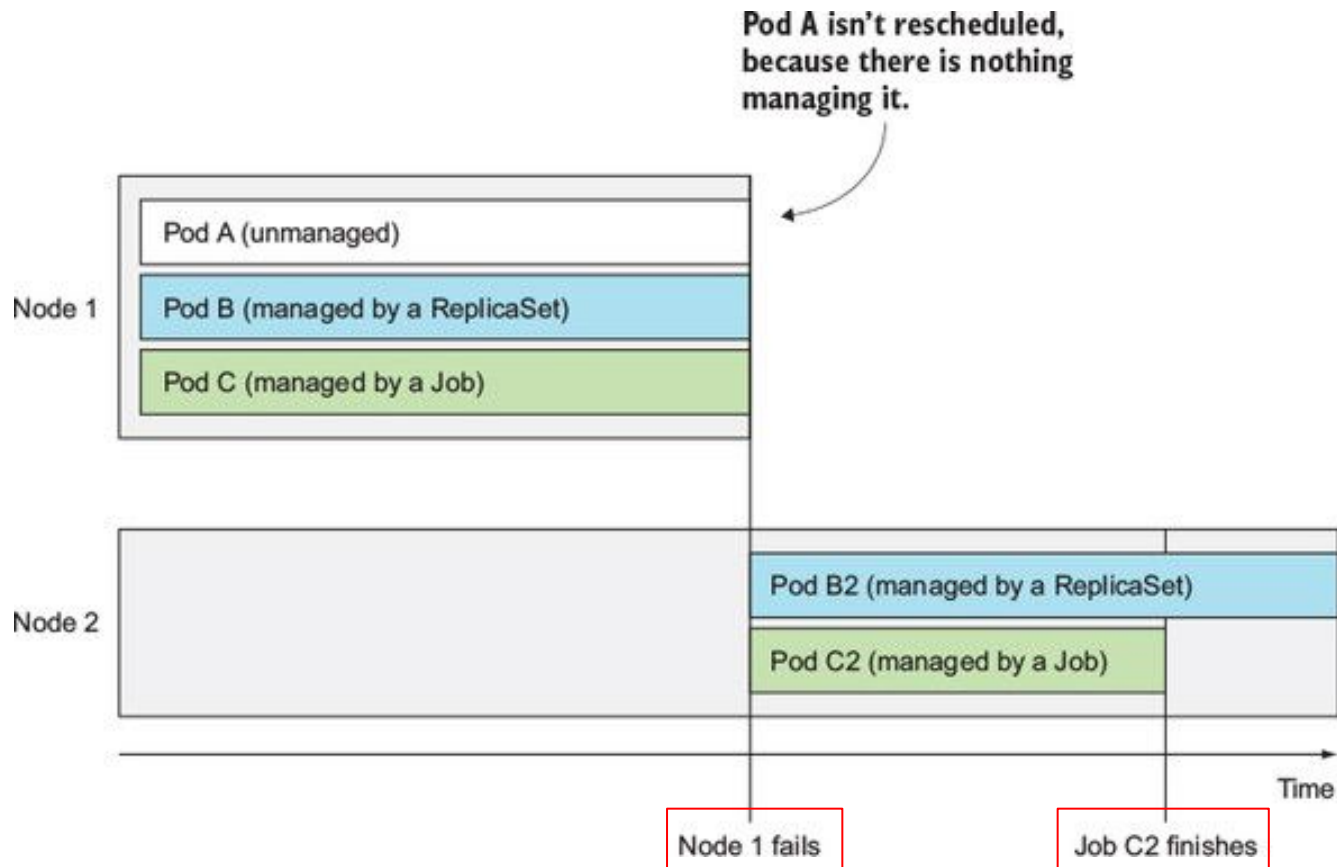
V.S.



completable tasks

K8s 保持 pod 的運行
當 pod 完成任務後，就會停止，
並不會重新啟動

Job



Job

```
apiVersion: batch/v1
kind: Job
metadata:
  name: batch-job
spec:
  template:
    metadata:
      labels:
        app: batch-job
    spec:
      restartPolicy: OnFailure
      containers:
        - name: main
          image: luksa/batch-job
```

batch-job.yaml

需要明確指定 restartPolicy
預設為 Always, Job 不能使用這個設定,
必須設為 **onFailure** 或 **Never**

- Always: 每次都會重啟
- onFailure: 回傳不是 0 的值, 才重啟
- Never: 永遠不會重啟

Job

kubectl get jobs

前

NAME	DESIRED	SUCCESSFUL	AGE
batch-job	1	0	2s

後

NAME	DESIRED	SUCCESSFUL	AGE
batch-job	1	1	9m

kubectl get po

前

NAME	READY	STATUS	RESTARTS	AGE
batch-job-28qf4	1/1	Running	0	4s

後

NAME	READY	STATUS	RESTARTS	AGE
batch-job-28qf4	0/1	Completed	0	2m

Job

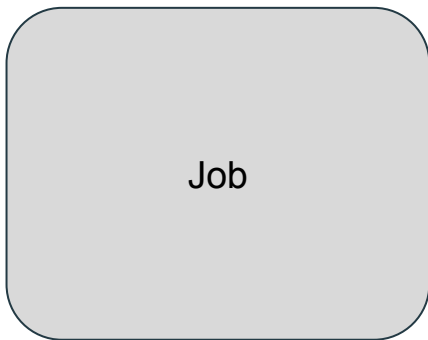
multi-completion-batch-job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: multi-completion-batch-job
spec:
  completions: 5
  template:
    metadata:
      labels:
        app: batch-job
    spec:
      restartPolicy: OnFailure
      containers:
      - name: main
        image: luksa/batch-job
```

multi-completion-parallel-batch-job.yaml

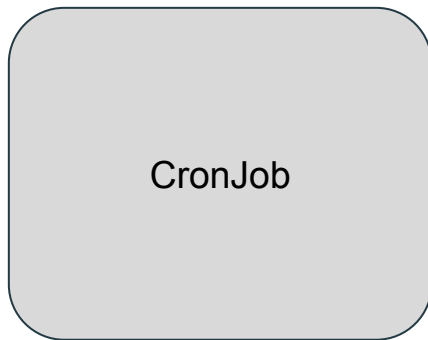
```
apiVersion: batch/v1
kind: Job
metadata:
  name: multi-completion-batch-job
spec:
  completions: 5
  parallelism: 2
  template:
    metadata:
      labels:
        app: batch-job
    spec:
      restartPolicy: OnFailure
      containers:
      - name: main
        image: luksa/batch-job
```

CronJob



建立好後，
就會馬上運行

V.S.



會有固定的時間去運行，
類似 unix 的 cron

CronJob

- `crontab -e` # 編輯 crontab
- `crontab -l` # 列出 crontab
- `crontab -r` # 刪除 crontab

星號 (*)	代表接受任意時刻，例如若在月份那一欄填入星號，則代表任一月份皆可。
逗號 (,)	分隔多個不同時間點。例如若要指定 3:00、6:00 與 9:00 三個時間點執行指令，就可以在第二欄填入 3,6,9。
減號 (-)	代表一段時間區間，例如若在第二欄填入 8-12 就代表從 8 點到 12 點的意思，也就是等同於 8,9,10,11,12。
斜線加數字 (/n)	n 代表數字，這樣寫的意思就是「每隔 n 的單位」的意思，例如若在第一欄填入 */5 就代表每間隔五分鐘執行一次的意思，也可以寫成 0-59/5。

```
# _____ 分鐘 (0 - 59)
# | _____ 小時 (0 - 23)
# | | _____ 日 (1 - 31)
# | | | _____ 月 (1 - 12)
# | | | | _____ 星期幾 (0 - 7, 0 是週日, 6 是週六, 7 也是週日)
# | | | | |
# * * * * * /path/to/command
```

Reference:

<https://blog.gtwang.org/linux/linux-crontab-cron-job-tutorial-and-examples/>

CronJob

cronjob.yaml

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: batch-job-every-fifteen-minutes
spec:
```

```
  schedule: "0,15,30,45 * * * *"
```

每天每小時的 0分, 15分, 30分, 45分執行

```
  jobTemplate:
```

```
    spec:
```

```
      template:
```

```
        metadata:
```

```
          labels:
```

```
            app: periodic-batch-job
```

```
        spec:
```

```
          restartPolicy: OnFailure
```

```
          containers:
```

```
            - name: main
```

```
              image: luksa/batch-job
```

pod 的 template

Q&A

- exit code 是 $128 + N$, 前面的 128 有甚麼特殊意義嗎

Exit Code Number	Meaning	Example
1	Catchall for general errors	let "var1 = 1/0"
2	Misuse of shell builtins (according to Bash documentation)	empty_function() {}
126	Command invoked cannot execute	/dev/null
127	"command not found"	illegal_command
128	Invalid argument to exit	exit 3.14159
128+n	Fatal error signal "n"	kill -9 \$PPID of script
130	Script terminated by Control-C	Ctl-C
255*	Exit status out of range	exit -1

Reference: <https://tldp.org/LDP/abs/html/exitcodes.html>

Q&A

1. 既然 liveness probe 是偵測 container 的健康度的方法, 那有什麼方法可以偵測整個 pod 的健康度?
 - a. pod 的健康度是由container是否健康來判定, 若有其中一個container 有問題則整個pod重啟, 似乎沒有單獨的方式可以檢測pod不含container的健康度, 若container皆為健康則k8s將那個pod為健康
2. *liveness probe* 檢測機制依據 pod 屬性挑選, 網頁可使用 HTTP GET 依據網頁回傳判定, 能夠參照 http status code, 那選擇 Exec 自行撰寫的程式有什麼規則可以參照, 能否由此方法從 exit code 得到更多資訊?
 - a. return 0 代表正常, 其他代表有錯誤

1. ReplicaSet可以用兩種不同的label去篩選嗎？

Pod Selector

The `.spec.selector` field is a [label selector](#). A ReplicationController manages all the pods with labels that match the selector. It does not distinguish between pods that it created or deleted and pods that another person or process created or deleted. This allows the ReplicationController to be replaced without affecting the running pods.

ref: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/>

2. 在 replica set and replica controller 中，當 pod 發生failed，會根據 pod template 產生新的 pod，但設定與原本不同的 template意義為何？

- a. 在不影響原來在運行的服務下，逐一更新pod至新的版本(可能有漏洞或可能需要更新)，並且同時可以檢測新版本是否穩定

1. **ReplicationControllers 發現 pod 過多時會移除 pod, 會以什麼原則挑選要移除的 pod?**
 - a. 先依照 node 的資源, 看要刪除哪個 node 上的 pod, 再隨機從 node 上的任一 pod 刪除

2. **yaml 設定檔中的 apiVersion 能指定 api 版本, 從不同 group 釋出的 api 版本差異? 哪個版本比較適合本地自架的?**
 - a. `$ kubectl api-versions` # 可用的 api
 - b. `$ kubectl api-resources -o wide` # 每個 object 要使用的 api

1. 在cronjob中難道不能設定, 設定年分執行的process嗎?

cronjob沒有辦法, 但可以用cron搭配

```
# Example of job definition:
# .----- minute (0 - 59)
# | .----- hour (0 - 23)
# | | .----- day of month (1 - 31)
# | | | .----- month (1 - 12) OR jan,feb,mar,apr ...
# | | | | .---- day of week (0 - 6) (Sunday=0 or 7)
# | | | | |
# * * * * * command to be executed
0 0 1 1 * [[ $(date "+\%Y") == 2020 ]] && command1
0 0 1 1 * (( $(date "+\%Y") % 3 == 0 )) && command2
0 0 1 1 * (( $(date "+\%Y") % 3 == 1 )) && command3
```

ref:<https://stackoverflow.com/questions/7724316/how-can-i-specify-time-in-cron-considering-year>

1. 如果我要安排一個工作在明日凌晨三點執行, 只執行一次。我該怎麼做? 用 Job 還是 CronJob ?
 - a. CronJob

2. CronJob可以安排時間執行, 那可不可以執行完後在一定的時間 內移除?
 - a. job/CronJob 執行完就會移除