Health Check

1. About K8S health check

Health check is the important feature of the k8s orchestrating. It's used in the automatic restarting containers by default. In addition to this, there are two kinds of ways for health check. Liveness and Readiness. What's the difference? And how to use these?

- The liveness health check configure file.

> ```
> apiVersion: v1
> kind: Pod
> metadata:
> labels:
> test: liveness
> name: liveness
> spec:
> restartPolicy: OnFailure
> containers:
>  - name: liveness
>    image: busybox
>    args:
>    - /bin/sh
>    - -c
>    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 60
>    livenessProbe:
>      exec:
>        command:
>        - cat
>        - /tmp/healthy
>      initialDelaySeconds: 10
>      periodSeconds: 5
> ```

From this configure file, we define the livenessProbe. Every 5 seconds, the probe will detect and execute the command. If failed, will restart the pod again.

- The Readiness health check configure file.

> ```
> apiVersion: v1
> kind: Pod
> metadata:
> labels:
> test: readiness
> name: readiness
> spec:
> restartPolicy: OnFailure
> containers:
>  - name: readiness
>    image: busybox
>    args:
>    - /bin/sh
>    - -c
>    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 60
>    readinessProbe:
>      exec:
>        command:
>        - cat
>        - /tmp/healthy
>      initialDelaySeconds: 10
>      periodSeconds: 5
> ```

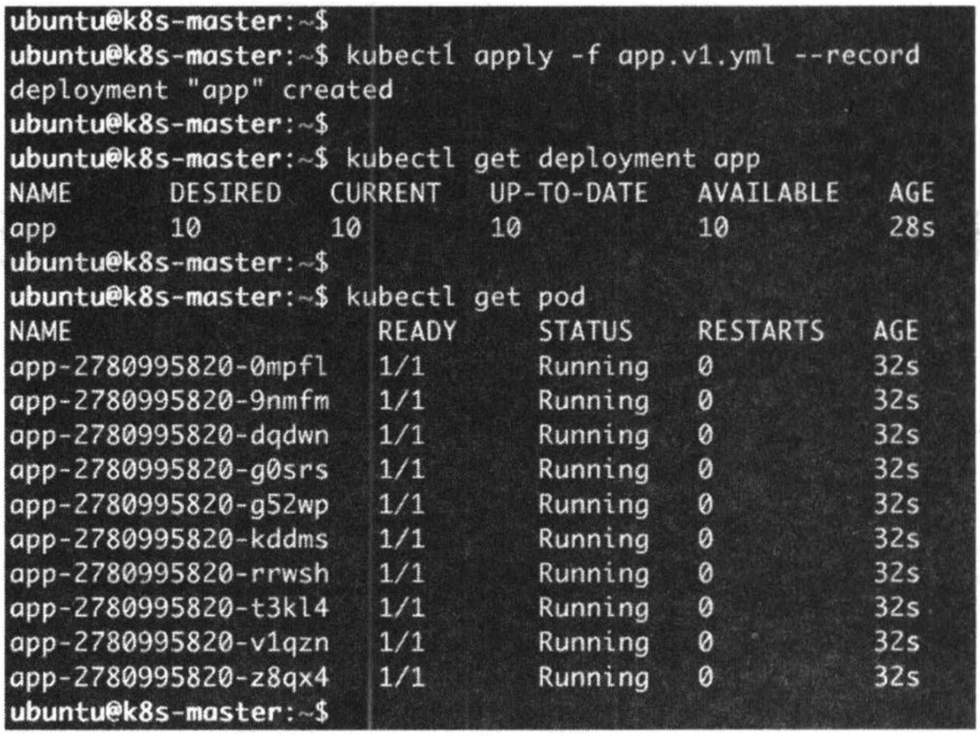
The file is similar to liveness. Just changed the key value. For this one, when it was failure, will restart the pod one time, after that, set the readiness no use.

2. Health Check practice in the rolling update

How the health check used in the rolling update? Image one case, you update the application from V1 to V2, but in fact V2 application is wrong, you didn't use the checking method to verify this. How will you solve this problem? We can use health check.

- First, deploy 10 copies with v1.

```
apiVersion: apps/v1
kind: Deployment
metadata:
name: app
spec:
replicas: 10
selector:
matchLabels:
 run: app
template:
metadata:
 labels:
   run: app
spec:
 containers:
     - name: app
       image: busybox
       args:
       - /bin/sh
       - -c
       - sleep 10; touch /tmp/healthy; sleep 30000
       readinessProbe:
         exec:
           command:
           - cat
           - /tmp/healthy
         initialDelaySeconds: 10
         periodSeconds: 5
```



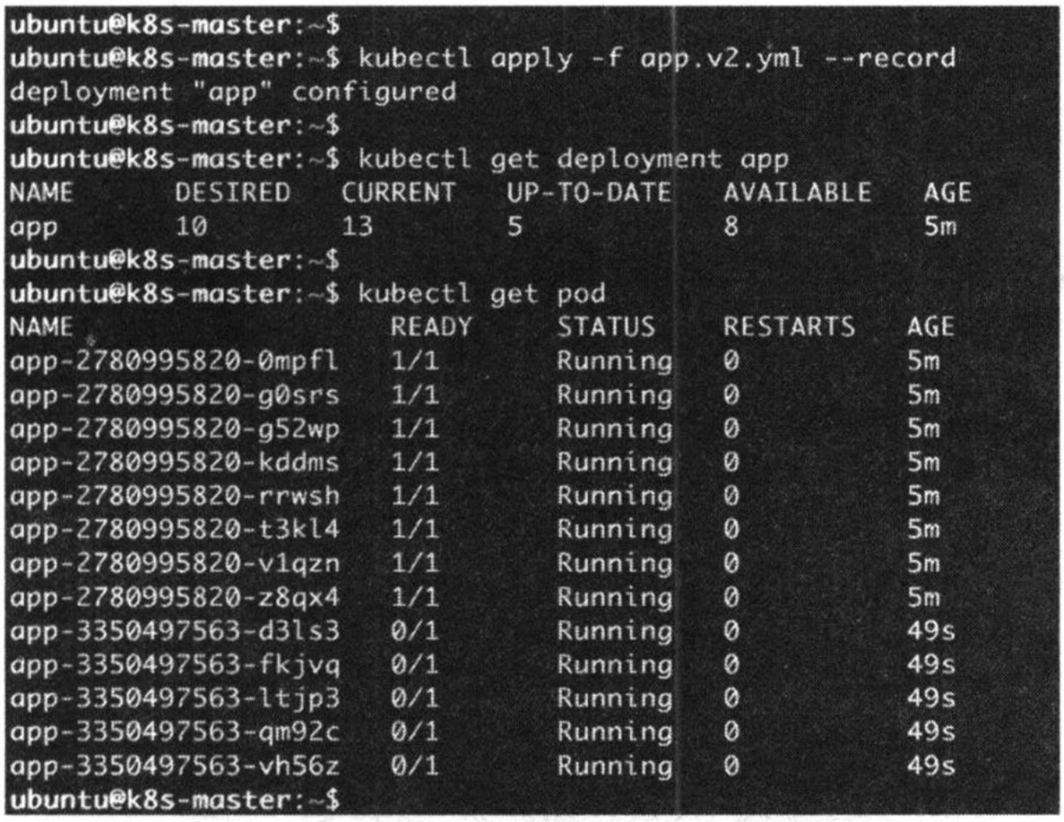
```
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl apply -f app.v1.yml --record
deployment "app" created
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get deployment app
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
app       10        10        10           10          28s
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get pod
NAME                   READY     STATUS    RESTARTS   AGE
app-2780995820-0mpfl   1/1       Running   0          32s
app-2780995820-9nmfm   1/1       Running   0          32s
app-2780995820-dqdwn   1/1       Running   0          32s
app-2780995820-g0srs   1/1       Running   0          32s
app-2780995820-g52wp   1/1       Running   0          32s
app-2780995820-kddms   1/1       Running   0          32s
app-2780995820-rrwsh   1/1       Running   0          32s
app-2780995820-t3kl4   1/1       Running   0          32s
app-2780995820-v1qzn   1/1       Running   0          32s
app-2780995820-z8qx4   1/1       Running   0          32s
ubuntu@k8s-master:~$
```

- Then rollling update the wrong application. It will trigger the probe to detect.

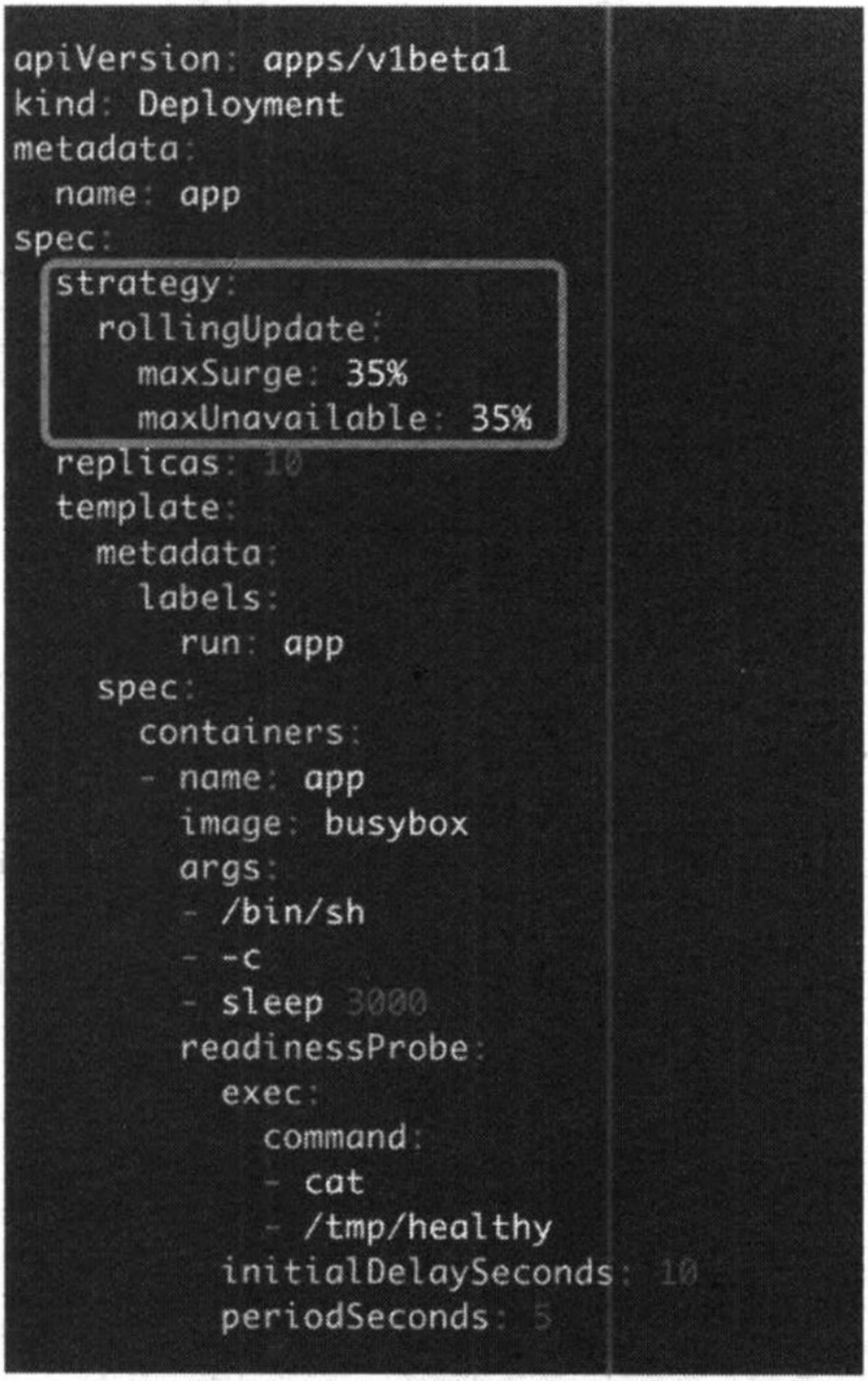
```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
name: app
spec:
replicas: 10
selector:
matchLabels:
 run: app
template:
metadata:
 labels:
   run: app
spec:
 containers:
     - name: app
       image: busybox
       args:
       - /bin/sh
       - -c
       - sleep 3000
       readinessProbe:
         exec:
           command:
           - cat
           - /tmp/healthy
         initialDelaySeconds: 10
         periodSeconds: 5
```

When use this configure file to deploy, will fail. Below is the result.



```
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl apply -f app.v2.yml --record
deployment "app" configured
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get deployment app
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
app       10        13        5            8           5m
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get pod
NAME                   READY     STATUS    RESTARTS   AGE
app-2780995820-0mpfl   1/1       Running   0          5m
app-2780995820-g0srs   1/1       Running   0          5m
app-2780995820-g52wp   1/1       Running   0          5m
app-2780995820-kddms   1/1       Running   0          5m
app-2780995820-rrwsh   1/1       Running   0          5m
app-2780995820-t3kl4   1/1       Running   0          5m
app-2780995820-v1qzn   1/1       Running   0          5m
app-2780995820-z8qx4   1/1       Running   0          5m
app-3350497563-d3ls3   0/1       Running   0          49s
app-3350497563-fkjvq   0/1       Running   0          49s
app-3350497563-ltjp3   0/1       Running   0          49s
app-3350497563-qm92c   0/1       Running   0          49s
app-3350497563-vh56z   0/1       Running   0          49s
ubuntu@k8s-master:~$
```



```yaml
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: app
spec:
  strategy:
    rollingUpdate:
      maxSurge: 35%
      maxUnavailable: 35%
  replicas: 10
  template:
    metadata:
      labels:
        run: app
    spec:
      containers:
      - name: app
        image: busybox
        args:
        - /bin/sh
        - -c
        - sleep 3000
        readinessProbe:
          exec:
            command:
            - cat
            - /tmp/healthy
          initialDelaySeconds: 10
          periodSeconds: 5
```

maxSurge and maxUnavailable are used to define the rolling update copies and failure copies.

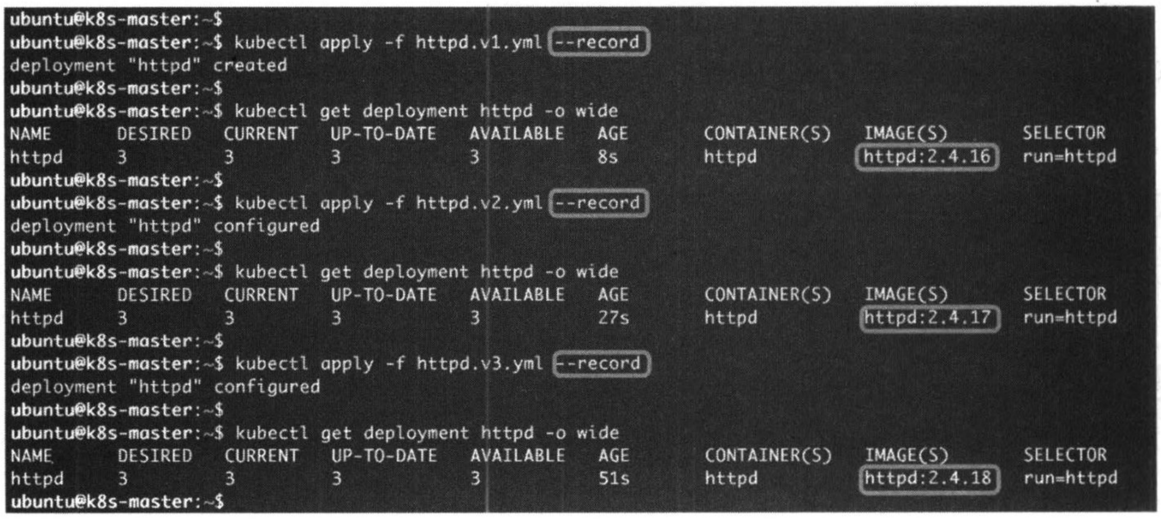
3. Rolling update

This section is not for health check. Only the summary of the learning before. This section comprises with rolling update and rolling back. Rolling update is very easy, just run the command, will deploy the change automatically. We mainly discuss the rolling back.

When you do the rolling update, you can record the revision. When you want to roll back, you can location this version.

- Kuebctl apply -f rollback.yaml --record

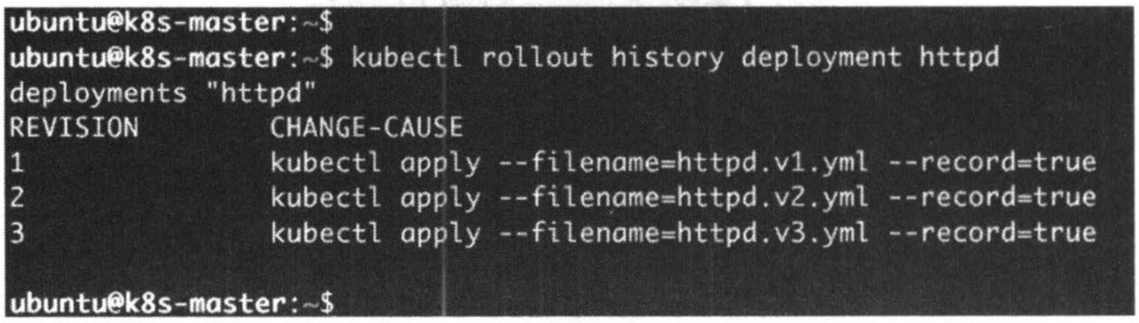
  When add the record parameter, will record the reversion.



```
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl apply -f httpd.v1.yml --record
deployment "httpd" created
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get deployment httpd -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE      CONTAINER(S)   IMAGE(S)       SELECTOR
httpd     3         3         3            3           8s       httpd          httpd:2.4.16   run=httpd
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl apply -f httpd.v2.yml --record
deployment "httpd" configured
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get deployment httpd -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE      CONTAINER(S)   IMAGE(S)       SELECTOR
httpd     3         3         3            3           27s      httpd          httpd:2.4.17   run=httpd
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl apply -f httpd.v3.yml --record
deployment "httpd" configured
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get deployment httpd -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE      CONTAINER(S)   IMAGE(S)       SELECTOR
httpd     3         3         3            3           51s      httpd          httpd:2.4.18   run=httpd
ubuntu@k8s-master:~$
```

- Kubectl rollout history deployment http

  This will show the all deployment history version.

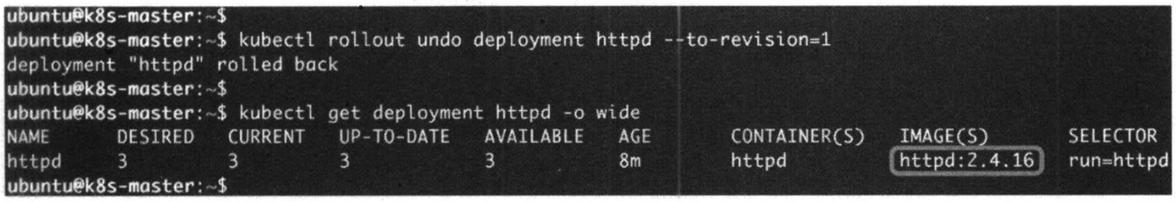


```
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl rollout history deployment httpd
deployments "httpd"
REVISION        CHANGE-CAUSE
1               kubectl apply --filename=httpd.v1.yml --record=true
2               kubectl apply --filename=httpd.v2.yml --record=true
3               kubectl apply --filename=httpd.v3.yml --record=true

ubuntu@k8s-master:~$
```

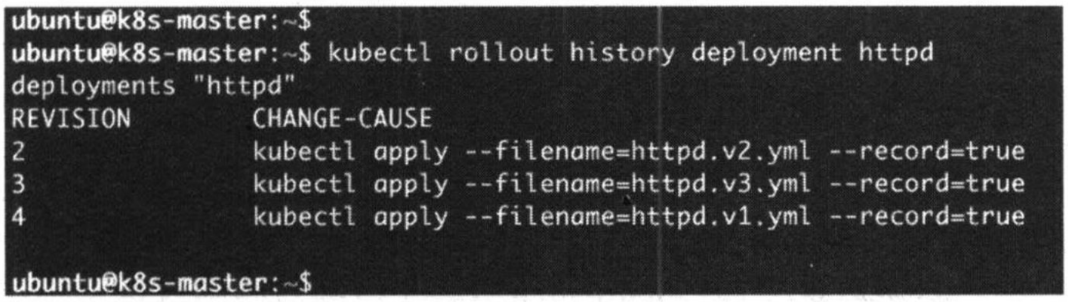
- Kubectl rollout undo deployment http --to-revision=1

  This will rollout the version.



```
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl rollout undo deployment httpd --to-revision=1
deployment "httpd" rolled back
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl get deployment httpd -o wide
NAME      DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE      CONTAINER(S)   IMAGE(S)       SELECTOR
httpd     3         3         3            3           8m       httpd          httpd:2.4.16   run=httpd
ubuntu@k8s-master:~$
```

  Also, when you rollout successful, the version will change as your deployment.



```
ubuntu@k8s-master:~$
ubuntu@k8s-master:~$ kubectl rollout history deployment httpd
deployments "httpd"
REVISION        CHANGE-CAUSE
2               kubectl apply --filename=httpd.v2.yml --record=true
3               kubectl apply --filename=httpd.v3.yml --record=true
4               kubectl apply --filename=httpd.v1.yml --record=true

ubuntu@k8s-master:~$
```