K8S Deployment Strategies

1. Background

We always deploy the final version applications to the production env. There are several ways to ensure the release production stably and safety. Below sections are k8s deployment strategies.

2. The Strategies

There are 4 ways to do the production release. 1) Rolling-update. 2) Recreate. 3) Blue/Green. 4) Canary. Below is the details.

- Recreate Update

  This is very clumsy way to update. Destory V1 application then create V2 application. Will hit service down issue in a period time.

Recreate

- **Rolling update.**

  This is the most commonly used.

  - **Performing an automatic rolling update with a RC**

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: kubia-v1
spec:
  replicas: 3
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
      - image: luksa/kubia:v1
        name: nodejs
---
apiVersion: v1
kind: Service
metadata:
  name: kubia
spec:
  type: NodePort
  selector:
    app: kubia
  ports:
  - port: 80
    targetPort: 8080
    nodePort: 30007
```

  To run this app, will create a ReplicationController and NodePort service to enable to access the app externally. Then open another terminal to monitor the service.

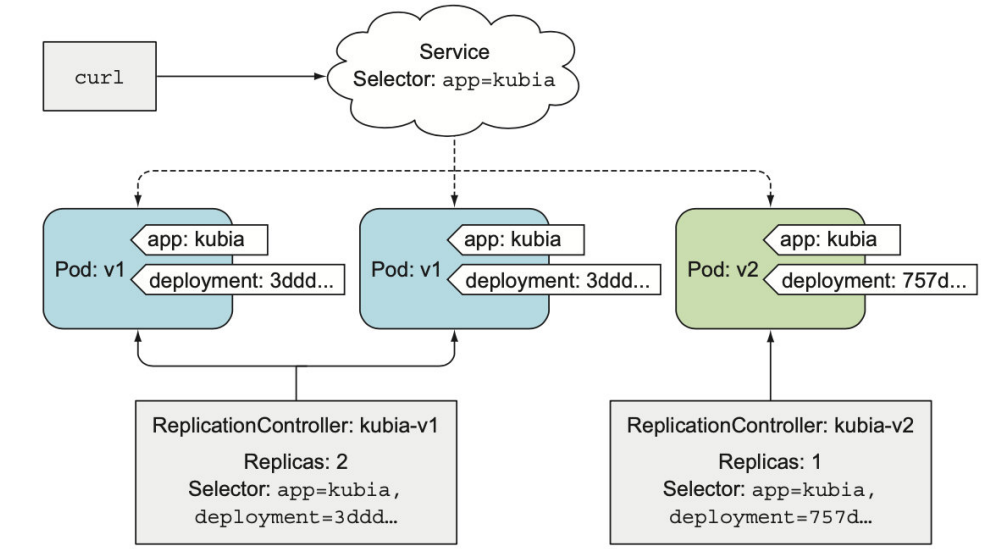
```
ylsccnu1_gmail_com@liansong-instance:~$ while true;do curl 10.128.0.2:30007;done
This is v1 running in pod kubia-v1-p9vvx
This is v1 running in pod kubia-v1-p9vvx
This is v1 running in pod kubia-v1-p9vvx
This is v1 running in pod kubia-v1-lwlfp
This is v1 running in pod kubia-v1-lxz85
This is v1 running in pod kubia-v1-lwlfp
This is v1 running in pod kubia-v1-lwlfp
This is v1 running in pod kubia-v1-p9vvx
This is v1 running in pod kubia-v1-lxz85
```

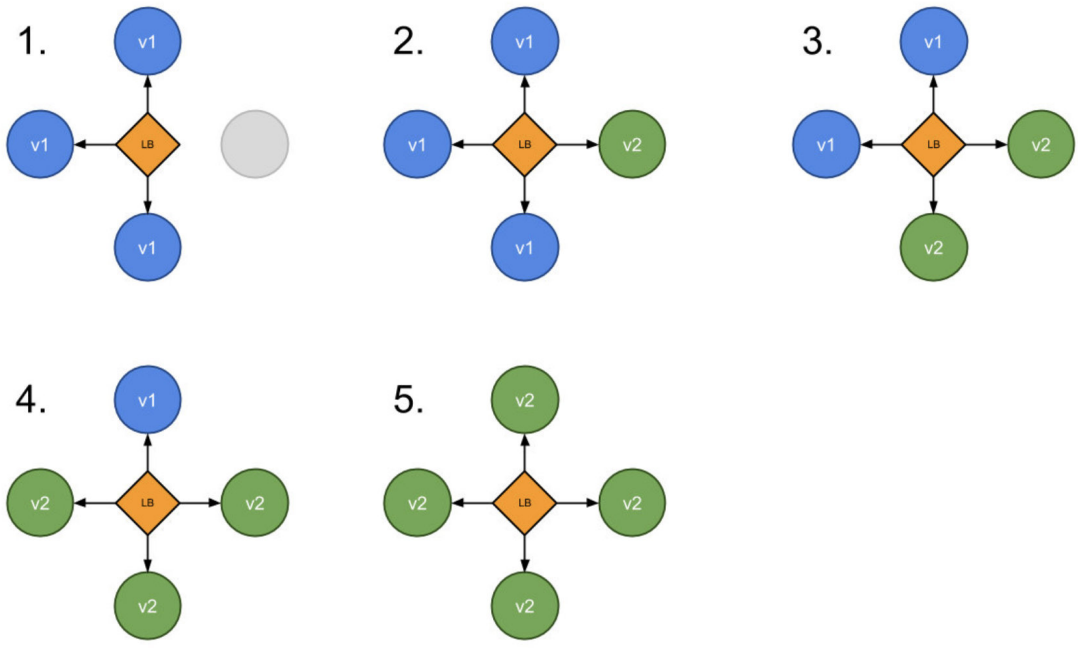
Then performing a rolling update with kubectl, will create version 2 of the app. Execute below command:

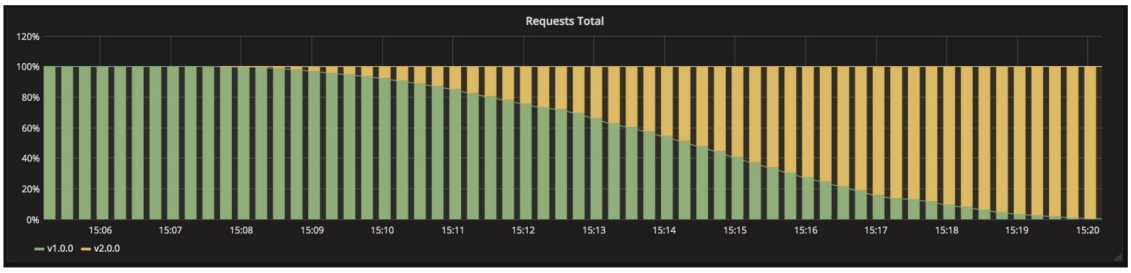
```
kubectl rolling-update kubia-v1 kubia-v2 --image=luksa/kubia:v2
```

```
ylsccnu1_gmail_com@liansong-instance:~/Deployment_Strategy$ kubectl rolling-update kubia-v1 ku
Command "rolling-update" is deprecated, use "rollout" instead
Created kubia-v2
Scaling up kubia-v2 from 0 to 3, scaling down kubia-v1 from 3 to 0 (keep 3 pods available, do
Scaling kubia-v2 up to 1
Scaling kubia-v1 down to 2
Scaling kubia-v2 up to 2
```

```
This is v2 running in pod kubia-v2-blrs2
This is v1 running in pod kubia-v1-lxz85
This is v1 running in pod kubia-v1-lwlfp
This is v1 running in pod kubia-v1-lwlfp
This is v2 running in pod kubia-v2-blrs2
This is v2 running in pod kubia-v2-blrs2
This is v2 running in pod kubia-v2-rcq7m
```







rolling-update requests

- **Using Deployment for updating apps declaratively**

  A deployment is a high-level resource meant for deploying applications and updating them declaratively, instead of doing it through a RC or RS, which are both considered lower-level concepts.


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubia-v1
spec:
  replicas: 3
  selector:
    matchLabels:
      app: kubia
  template:
    metadata:
      name: kubia
      labels:
        app: kubia
    spec:
      containers:
      - image: luksa/kubia:v1
        name: nodejs
---
apiVersion: v1
kind: Service
metadata:
  name: kubia
spec:
  type: NodePort
  selector:
    app: kubia
  ports:
  - port: 80
```

Also to run this application, after completed, change the v1 deployment container to v2, then deploy with the new version.

```
kubectl apply -f kubia-deploy-v2.yaml
```

After that, you can notice that there will reserve the V1 RS. So it's very easy to rollback the version from V2 to V1.

```
ylsccnul_gmail_com@liansong-instance:~/Deployment_Strategy$ kubectl get rs
NAME                  DESIRED   CURRENT   READY   AGE
kubia-v1-6c99f46f5    3         3         3       33s
kubia-v1-74fb644f8    0         0         0       2m39s
```

Ways of modifying Deployments and other resources

Over the course of this book, you've learned several ways how to modify an existing object. Let's list all of them together to refresh your memory.

Table 9.1 Modifying an existing resource in Kubernetes

Method	What it does
`kubectl edit`	Opens the object's manifest in your default editor. After making changes, saving the file, and exiting the editor, the object is updated. Example: `kubectl edit deployment kubia`
`kubectl patch`	Modifies individual properties of an object. Example: `kubectl patch deployment kubia -p '{"spec": {"template": {"spec": {"containers": [{"name": "nodejs", "image": "luksa/kubia:v2"}]}}}}'`
`kubectl apply`	Modifies the object by applying property values from a full YAML or JSON file. If the object specified in the YAML/JSON doesn't exist yet, it's created. The file needs to contain the full definition of the resource (it can't include only the fields you want to update, as is the case with `kubectl patch`). Example: `kubectl apply -f kubia-deployment-v2.yaml`
`kubectl replace`	Replaces the object with a new one from a YAML/JSON file. In contrast to the `apply` command, this command requires the object to exist; otherwise it prints an error. Example: `kubectl replace -f kubia-deployment-v2.yaml`
`kubectl set image`	Changes the container image defined in a Pod, ReplicationController's template, Deployment, DaemonSet, Job, or ReplicaSet. Example: `kubectl set image deployment kubia nodejs=luksa/kubia:v2`

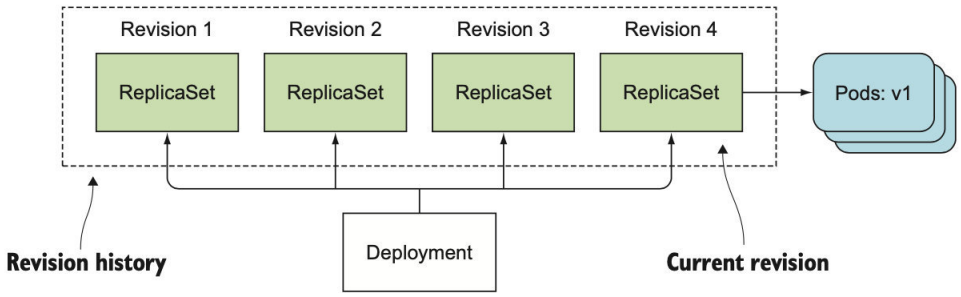
All these methods are equivalent as far as Deployments go. What they do is change the Deployment's specification. This change then triggers the rollout process.

Use this command to roll back the deployment.

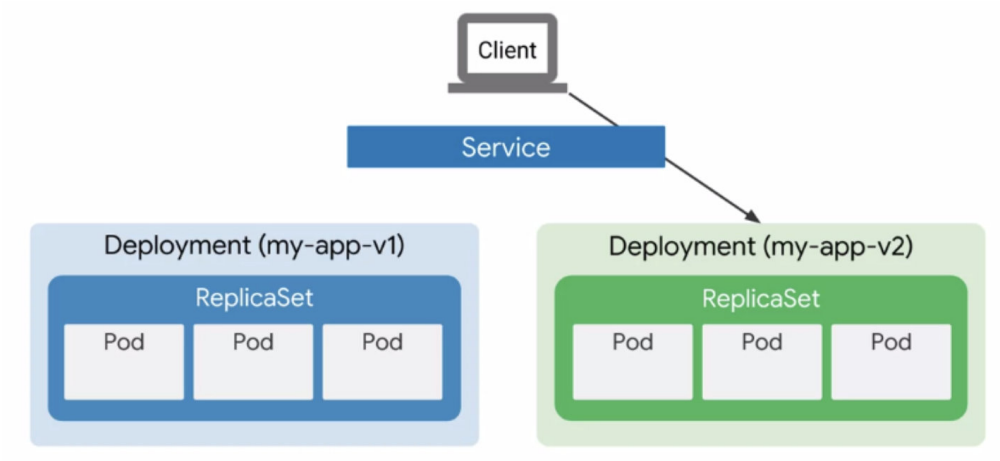
```
kubectl rollout undo deployment kubia
```

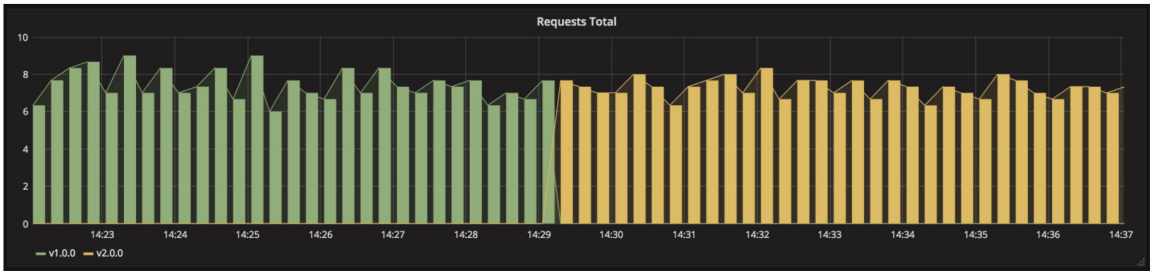
```
kubectl rollout history deployment kubia
```

```
kubectl rollout undo deployment kubia --to-revision=1
```



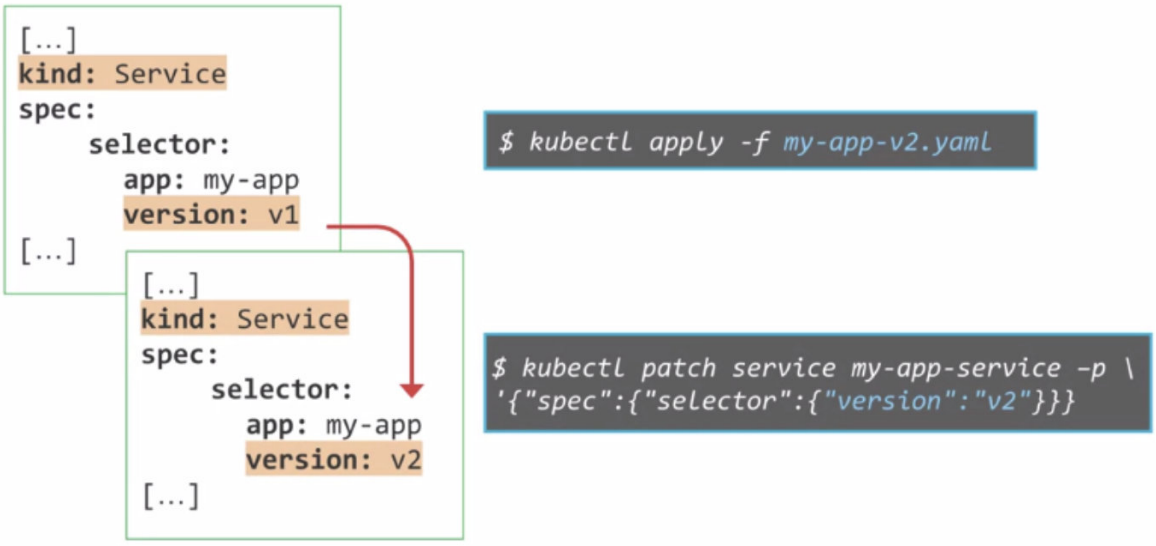
- **Blue / Green**



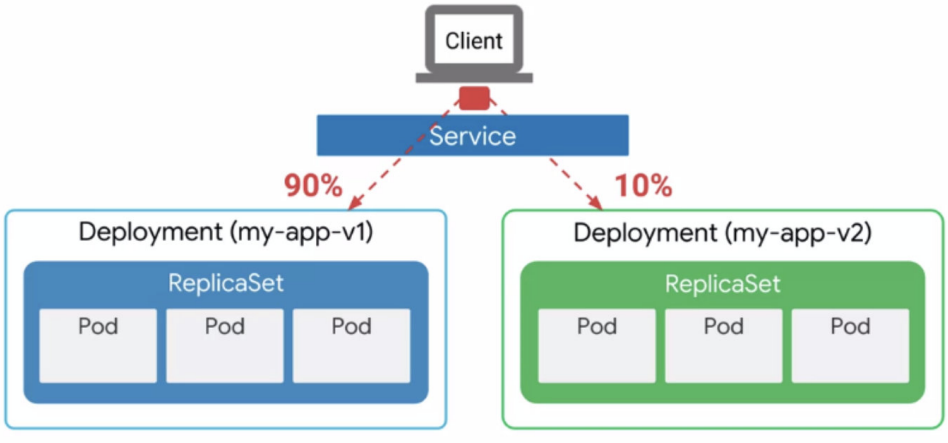


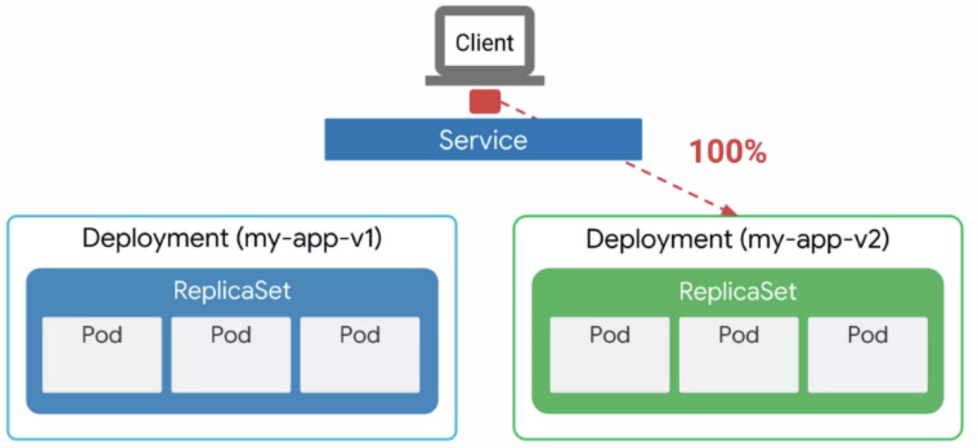
blue/green request

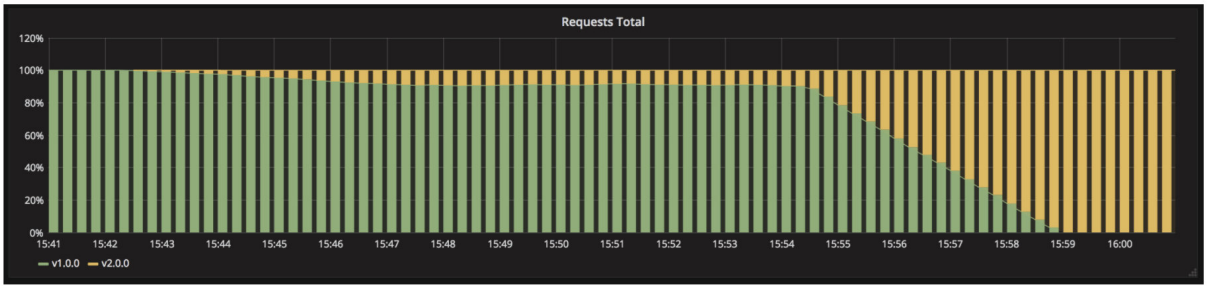
Actually this deployment needs V1 and V2 applications are existing. The client connect the Deployment through the labels.



- **Canary Deployment**







canary requests