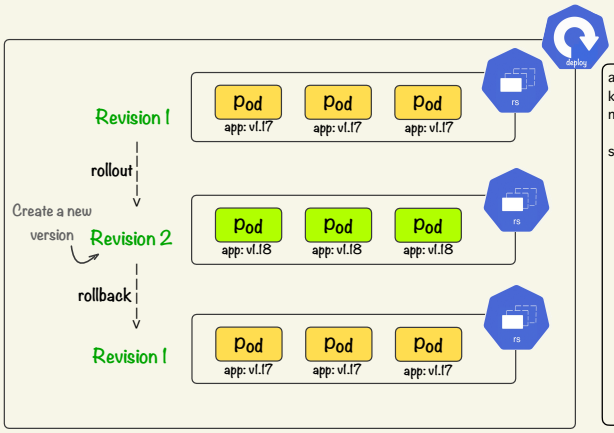
Rollout & Rollback

Rollout is the process of updating a Deployment or ReplicaSet to a new version of your application
Rollback is performed by updating container with the previous version of the container image



Deployment — Create → ReplicaSet-1

rollback / rollout

ReplicaSet-2

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-imp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.17
        ports:
        - containerPort: 80
```

After adding the 'strategy' section to the YAML file and applying it to the Kubernetes cluster using the 'kubectl apply' command, Kubernetes will start a RollingUpdate for the Deployment

```
kubectl apply -f deployment.yaml
```

Update image to nginx:1.2

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-imp
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 1
      maxSurge: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.18
        ports:
        - containerPort: 80
```

When you perform an upgrade to a deployment, Kubernetes creates a new replica set with the updated container image and configuration, and gradually replaces the pods managed by the old replica set with the pods managed by the new replica set

During a Rolling update , the 'maxUnavailable' and 'maxSurge' settings determine the rate at which replicas are replaced, ensuring that the application remains available and stable throughout the update process

'maxUnavailable' specifies the maximum number of replicas that can be unavailable during the update process. This parameter ensures that the application always has a minimum number of replicas available, even during the update process. For example, if you set 'maxUnavailable' to 1, Kubernetes will not terminate more than one replica at a time during the update process, ensuring that the application always has at least one replica available.

'maxSurge' specifies the maximum number of new replicas that can be created during the update process. This parameter ensures that the update process is efficient and does not overload the system with too many new replicas at once. For example, if you set 'maxSurge' to 1, Kubernetes will not create more than one new replica at a time during the update process, ensuring that the application remains stable and functional throughout the update.

you can also run a rolling update in Kubernetes using the 'kubectl' command

- To perform a rolling update using the 'kubectl' command, you need to have a Deployment defined in Kubernetes

```
kubectl create deployment nginx-imp —image nginx:1.17 —replicas 3
```

create a Deployment with the 'nginx:1.17' image and three replicas

- use the 'set' command in 'kubectl' to update the image used by the Deployment

```
kubectl set image deployment/nginx-imp nginx=nginx:1.18
```

After executing the 'set' command, Kubernetes will start a rolling update for the 'nginx-imp' Deployment

pod name-Replicaset id-pod id

```
kubectl get pod,rs
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx-imp-f32gt99mnj-ihd7t      1/1     Running   0          10m
...
pod/nginx-imp-f32gt99mnj-ki34f      1/1     Running   0          10m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-imp-f32gt99mnj   1         1         1       10m
replicaset.apps/nginx-imp-ht5g34kpz2   0         0         0       12m
```

Old Replicaset — New Replicaset

- You can monitor the progress of the rolling update by running the following command

```
kubectl rollout status deployment/nginx-imp
```

If you want to pause the rolling update at any time, you can use this command:

```
kubectl rollout pause deployment/nginx-imp
```

```
kubectl get pod,rs
NAME                                READY   STATUS    RESTARTS   AGE
pod/nginx-imp-ht5g34kpz2-ihd7t      1/1     Running   0          10m
...
pod/nginx-imp-ht5g34kpz2-ki34f      1/1     Running   0          10m

NAME                                DESIRED   CURRENT   READY   AGE
replicaset.apps/nginx-imp-f32gt99mnj   0         0         0       14m
replicaset.apps/nginx-imp-ht5g34kpz2   1         1         1       16m
```

- If you want to undo the update and roll back to the previous version, you can use the following command:

```
kubectl rollout undo deployment/nginx-imp
```

- You can use 'kubectl rollout history' command to view the revision history of a Deployment, including the rollout status, the version of the Deployment, and the date and time of the revision

```
kubectl rollout history deployment nginx-imp
deployment.apps/nginx-imp
REVISION  CHANGE-CAUSE
1         kubectl create deployment nginx-imp --image=nginx:1.17 --replicas=5
2         kubectl set image deployment/nginx-imp nginx=nginx:1.18
```

The 'CHANGE-CAUSE' field in the 'kubectl rollout history' output is an annotation that is added to the Deployment when it is updated using the 'kubectl set' command This annotation can be useful for tracking changes and providing additional information about the update process

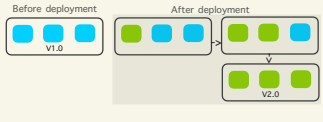
To change the 'CHANGE-CAUSE' annotation for a Deployment in Kubernetes, you can use the 'kubectl annotate' command

```
kubectl rollout undo deployment/nginx-imp --to-revision=2
kubectl annotate deployment nginx-imp kubernetes.io/change-cause="updated to nginx 1.19" --overwrite
```

```
kubectl rollout history deployment nginx-imp
deployment.apps/nginx-imp
REVISION  CHANGE-CAUSE
1         kubectl create deployment nginx-imp --image=nginx:1.17 --replicas=3
2         updated to nginx 1.18
```

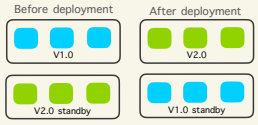
Some of Deployment strategies to perform rollouts

Rolling updates are performed by gradually replacing instances of an old version of a container with instances of a new version. (default deployment strategy)



Recreate strategy deletes all the old Pods before creating new ones. This can result in some downtime for your application

Blue/Green strategy creates a new set of Pods running the updated version of your application alongside the old set of Pods running the previous version



Canary strategy updates a small percentage of Pods with the new version of your application, while the rest of the Pods continue to run the previous version