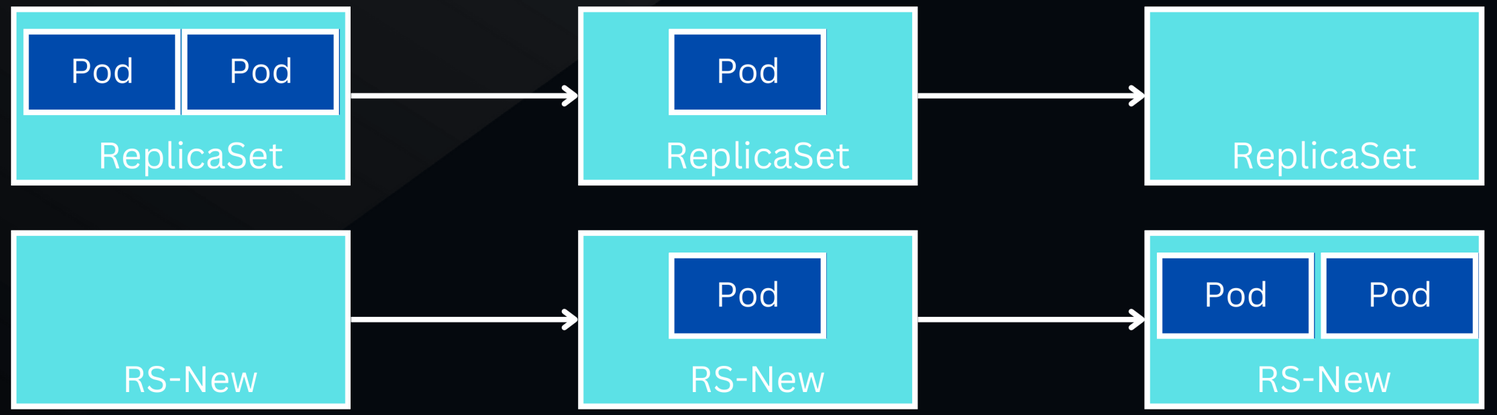
DEPLOYMENT

- https://kubernetes.io/docs/concepts/workloads/controllers/deployment/
- It is higher-level abstraction that manages **ReplicaSets** and provides declarative updates to **pods**

QUESTION

- What is **difference between deployment and replicaset** then?
- Answer: It becomes clear if you upgrade your pod
  - let's say we have the existing RS, and it controls the pod, now we want to upgrade the pod version from 1.0.1 to 1.0.2.
  - In this situation, we need to create new RS(say RS-new)
  - Then, we need to scale down the original RS by one and increase the new RS(RS-new) by one
  - And repeat again until we release all of RS-new
  - However, the deployment does automatically when we update the image version



DEPLOYMENT YML

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80

```

```
 ~/git/learn-k8s/ k get deployment
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
nginx-deployment    3/3     3            3           4m
```

```
 ~/git/learn-k8s/ k get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-85996f8dbd   3         3         3       4m29s
```

```
 ~/git/learn-k8s/ k get pod
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-85996f8dbd-2j684   1/1     Running   0          2m28s
nginx-deployment-85996f8dbd-4bn9v   1/1     Running   0          2m28s
nginx-deployment-85996f8dbd-ql68v   1/1     Running   0          2m28s
```

ROLLOING UPDATE

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        # changed 1.14.2 to 1.22.1
        image: nginx:1.22.1
        ports:
        - containerPort: 80

```

```
 ~/git/learn-k8s/ k rollout status deployment nginx-deployment
Waiting for deployment "nginx-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 1 out of 3 new replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 2 out of 3 new replicas have been updated...
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas are pending termination...
Waiting for deployment "nginx-deployment" rollout to finish: 1 old replicas are pending termination...
deployment "nginx-deployment" successfully rolled out
```

```
 ~/git/learn-k8s/ k rollout history deployment/nginx-deployment
deployment.apps/nginx-deployment
REVISION  CHANGE-CAUSE
1         <none>
2         <none>
```

DEPLOYMENT STRATEGY

- **.spec.strategy** specifies the strategy used to replace old Pods by new ones. **.spec.strategy.type** can be **"Recreate"** or **"RollingUpdate"**. "RollingUpdate" is the default value.
  - All existing Pods are killed before new ones are created when .spec.strategy.type==Recreate

```yaml
spec:
  ...
  strategy:
    type: "Recreate"
  ...
```

```
Events:
  Type    Reason             Age   From                   Message
  ----    ------             ----  ----                   -------
  Normal  ScalingReplicaSet  21m   deployment-controller  Scaled up replica set nginx-deployment-85996f8dbd to 3
  Normal  ScalingReplicaSet  13m   deployment-controller  Scaled up replica set nginx-deployment-579c9dfc44 to 1
  Normal  ScalingReplicaSet  13m   deployment-controller  Scaled down replica set nginx-deployment-85996f8dbd to 2 from 3
  Normal  ScalingReplicaSet  13m   deployment-controller  Scaled up replica set nginx-deployment-579c9dfc44 to 2 from 1
  Normal  ScalingReplicaSet  13m   deployment-controller  Scaled down replica set nginx-deployment-85996f8dbd to 1 from 2
  Normal  ScalingReplicaSet  13m   deployment-controller  Scaled up replica set nginx-deployment-579c9dfc44 to 3 from 2
  Normal  ScalingReplicaSet  13m   deployment-controller  Scaled down replica set nginx-deployment-85996f8dbd to 0 from 1
  Normal  ScalingReplicaSet  59s   deployment-controller  Scaled down replica set nginx-deployment-579c9dfc44 to 0 from 3
  Normal  ScalingReplicaSet  59s   deployment-controller  Scaled up replica set nginx-deployment-85996f8dbd to 3 from 0
```

COMMON DEPLOYMENT

- **Rolling update strategy**: This strategy replaces old replicas with new ones in a controlled manner, **one at a time**, and monitors the application's health before continuing with the update. This strategy provides minimal downtime and allows you to roll back to the previous version if needed.
- **Blue-green deployment strategy**: This strategy deploys **two identical environments** (blue and green), but only one environment is active (blue) while the other (green) is idle. The new version of the application is deployed to the inactive environment (green), and after thorough testing, traffic is switched to the green environment, making it active while the blue environment is idle. This strategy provides zero-downtime and a quick rollback in case of failure.
- <continue>

COMMON DEPLOYMENT

- **Canary deployment strategy**: This strategy allows you to test new versions of the application by gradually **routing a small percentage of traffic to the new version** while the majority of traffic is still sent to the previous version. This strategy reduces the impact of the update on the production environment while allowing you to collect feedback and make adjustments.
- **A/B testing deployment strategy**: This strategy deploys **multiple versions of the application** and routes traffic between them to test different features or changes in functionality. This strategy helps you to validate assumptions and test different versions before fully deploying a new version.

ROLLBACK

```
 ~/git/learn-k8s/ k get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-579c9dfc44   0         0         0       38m
nginx-deployment-85996f8dbd   3         3         3       46m
```

```
 ~/git/learn-k8s/ k rollout undo deployment nginx-deployment
deployment.apps/nginx-deployment rolled back
 ~/git/learn-k8s/ k get rs
NAME                          DESIRED   CURRENT   READY   AGE
nginx-deployment-579c9dfc44   3         3         3       39m
nginx-deployment-85996f8dbd   0         0         0       47m
```