Rollout Command

You can see the status of your rollout by running the command

object

```
kubectl rollout status deployment/myapp-deployment
```

Name of the object

To see the history & revision of rollout

```
kubectl rollout history deployment/myapp-deployment
```

This shows you the "revisions" & "history" of our deployment

deployments "myapp-deployment"

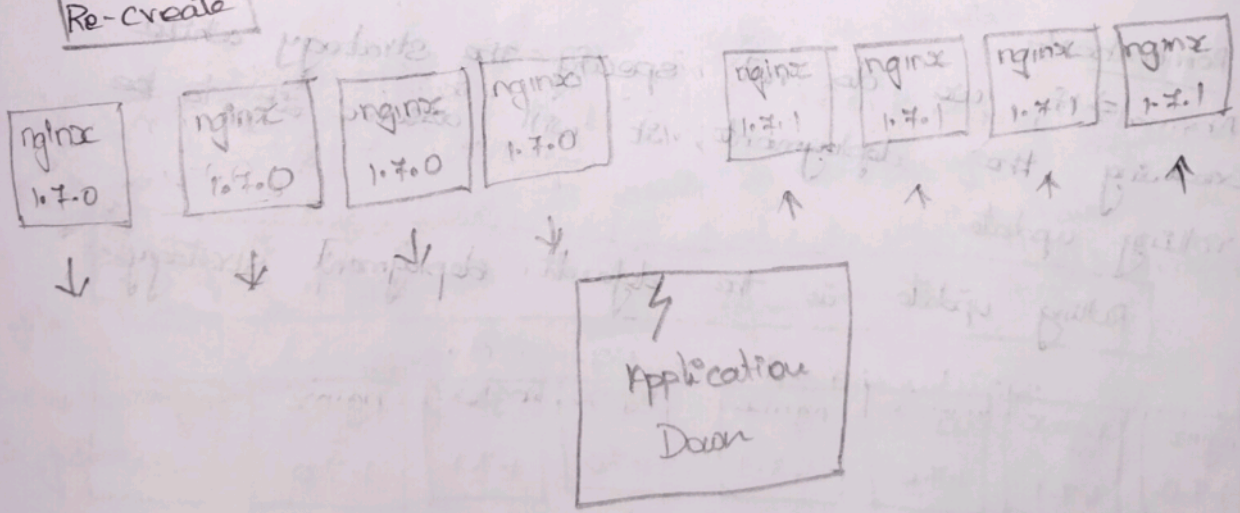
REVISION	CHANGE-CAUSE
1	<none>
2	kubectl apply --filename = deployment-definition .yml --record = true

Deployment Strategy

⇒ Re-Create

⇒ Rolling update

Re-create



⇒ 4 replica instance of web-application is deployed

⇒ One way to upgrade this 4 replicas to newer version is to destroy all of these & create newer versions of application instances

⇒ Meaning:- 1st Destroy the 4 running instances & then deploy 4 new instances of the new application version

Problem:- During the period after the older versions are down & before any newer version is up, the application is down and inaccessible to users.

⇒ This is known as Recreate Strategy, & this is not default deployment strategy

Rolling update

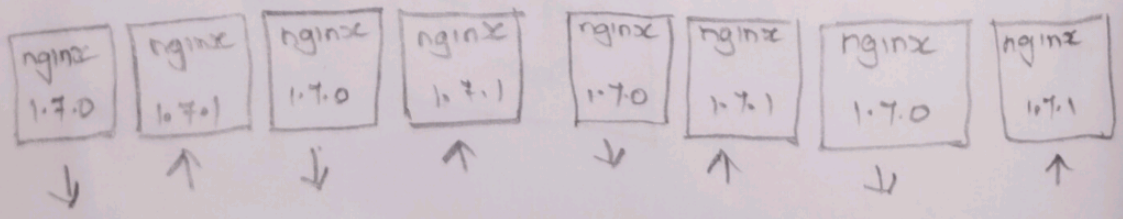
⇒ In rolling update, where we do not destroy all of them at once, instead we take down the older version & bring up a newer version one by one

⇒ This way the application never goes down & the upgrade is seamless

Remember:-

⇒ If we do not specify the strategy while creating the deployment, it will assume it to be rolling update

> Rolling update is the default deployment strategy



kubectl apply

update - It could be different things such as updating

* Application version [by using the version of docker Containers used]'s

* Updating labels * Updating no. of. replicas etc.,

⇒ If we have deployment-definition-file, it will be easy for us to modify this file by just opening the file & edit the image version, save & exit, then kubectl apply

>kubectl apply -f deployment-definition.yml
deployment "myapp-deployment" configured

>once we run the kubectl apply command to apply the changes, a new rollout is triggered & a new revision of the deployment is created

There is another way to do the same thing

⇒ we can use the "kubectl set image" command to update the image of your application"

>kubectl set image deployment/myapp-deployment \
nginx-container = nginx:1.9.1
deployment "myapp-deployment" image is updated

Remember:-

Doing it this way will result in the deployment definition file having a different configuration, so we have to be cautious when using the same definition file to make changes in future

38 Difference b/w the recreate & Rolling update

⇒ The difference can also be seen, when you view the deployments in details

```
Kubectl describe deployment myapp-deployment
```

⇒ you will notice when the recreate strategy was used the events indicate that the old replica set was scaled down to zero first & then the new replicaset scaled up to 5

```
Name:           myapp-deployment
Namespace:
CreationTimestamp:
Labels:
Annotations:
Selectors:
Replicas:
StrategyType:   Recreate
MinReadySeconds: 0

Events:
Type    Reason             Age   From                   Message
----    ------             ----  ----                   -------
Normal  ScalingReplicaSet  11m   deployment-controller  scaled up replicaset myapp-deployment to 5
Normal  ScalingReplicaSet  1m    deployment-controller  scaled down replicaset myapp-deployment to 0
Normal  ScalingReplicaSet  56s   deployment-controller  scaled up replicaset myapp-deployment to 5
```

When Rolling Update strategy was used the old replica set was scaled down one at a time, simultaneously scaling up the new replicaset one at a time

StrategyType: RollingUpdate

Events:

Type	Reason	Age	From	Message
Normal	ScalingReplicaSet	1m	deployment-Controller	Scaled up replicaset myapp-deployment to 5
				Scaled up replicaset myapp-deployment-2idx to 2
				Scaled down replicaset myapp-deployment-2idx to 4
				Scaled up replicaset myapp-deployment-2idx to 3
				Scaled down replicaset myapp-deployment-2idx to 3
				Scaled up replicaset myapp-deployment-2idx to 4
				Scaled down replicaset myapp-deployment-2idx to 2

Upgrades

Lets look at how a deployment performs an upgrade

⇒ When a new deployment is created to deploy 5 replicas

0:29 * It first creates a replicaset automatically, which
5:13 in turn creates the no. of. pods required to meet the no.of. replicas

when you upgrade your application, the k8s deployment object creates a new replicaset & starts deploying the containers there at the same time, taking down the pods in the old replicaset following a "rolling update" strategy

This can be seen when you try to list the replicasets

```
kubectl get replicasets
```

NAME	DESIRED	CURRENT	READY	AGE
myapp-deployment-<id>	0	0	0	22m
myapp-deployment-<not the above pod id, its unique id>	5	5	5	20m

Here, we see the old replica set with zero pods & the new replicaset with 5 pods

5:13 Rollback

Once you upgrade your application, you realise something is wrong with the new version of the build you used to upgrade

So you would like to rollback your update

K8s deployments allows you to rollback to a previous revision.

To undo a change:-

`kubectl rollout undo deployment/myapp-deployment`

deployment "myapp-deployment" rolled back

⇒ The deployment will then destroy the pods in the new replicaset & bring the older ones up in the old replicaset & your application is back to its older format.

⇒ When you compare the output of the kubectl get replicasets command before & after the rollback.

Before the rollback:-

⇒ The first replicaset had zero pods & new replicaset had 5 pods & this is reversed after the rollback is finished

Before Rollback

`kubectl get replicasets`

NAME	DESIRED	CURRENT	READY	AGE
myapp-deployment-<id>	0	0	0	22m
myapp-deployment-<another id>	5	5	5	20m

After rollback

`kubectl get replicasets`

NAME	DESIRED	CURRENT	READY	AGE
myapp-deployment-<id>	5	5	5	22m
myapp-deployment-<id>	0	0	0	20m

Initially for the first time, if you want to create the deployment object, then use

```
Kubectl Create -f deployment-definition.yml
```

To list the created deployment object

```
Kubectl get deployments
```

If you're updating labels, images, no.of. replicas etc, in the manifest file, post editing the manifest file. use

```
Kubectl apply -f deployment-definition.yml
```

To set the image using imperative command, which is not recommended because the deployment file in local will not get changed, once the object is updated with manifest & imperative command.

```
Kubectl set image deployment/myapp-deployment
                  nginx=nginx:1.9.1
```

Status:-

```
Kubectl rollout status deployment/myapp-deployment
```

```
Kubectl rollout history deployment/myapp-deployment
```

Rollback:-

```
Kubectl rollout undo deployment/myapp-deployment
```