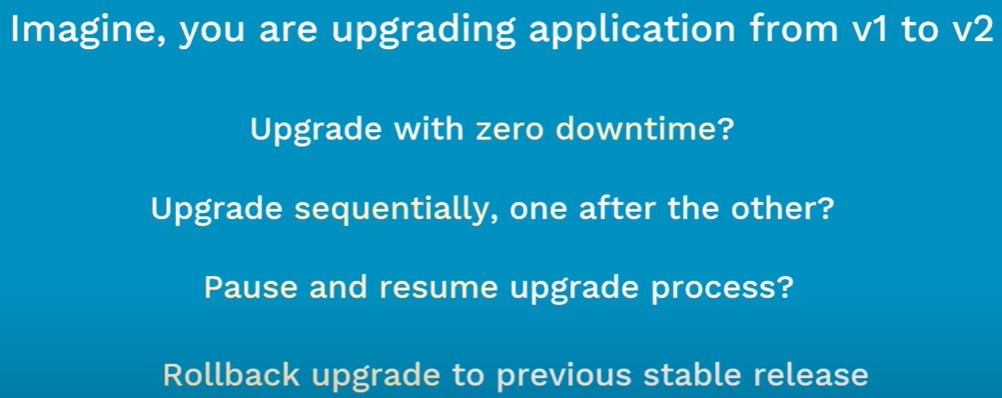
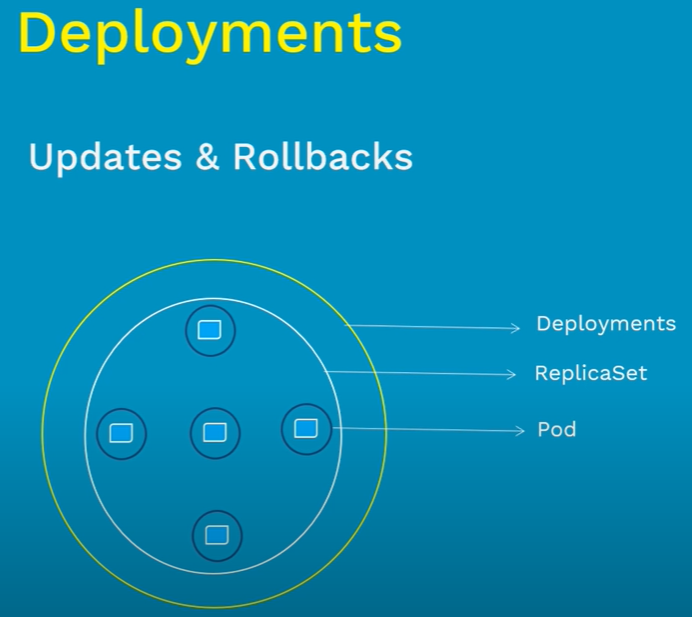
**Deployments:**

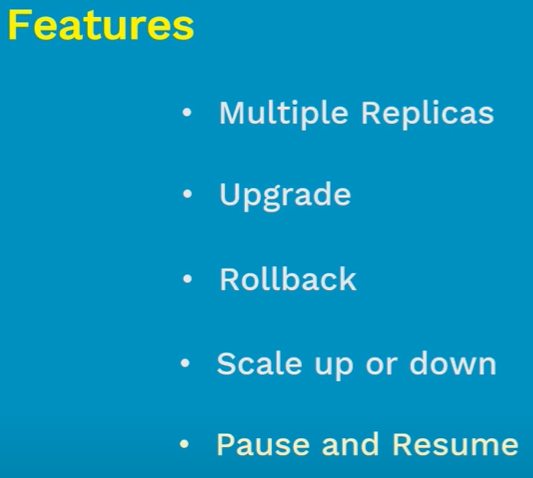


* Deployment is a controller just like any other controllers such as replication controllers or replicasets
* Deployment is all about updates and rollbacks.
* Replicaset does not provide update and rollback services.

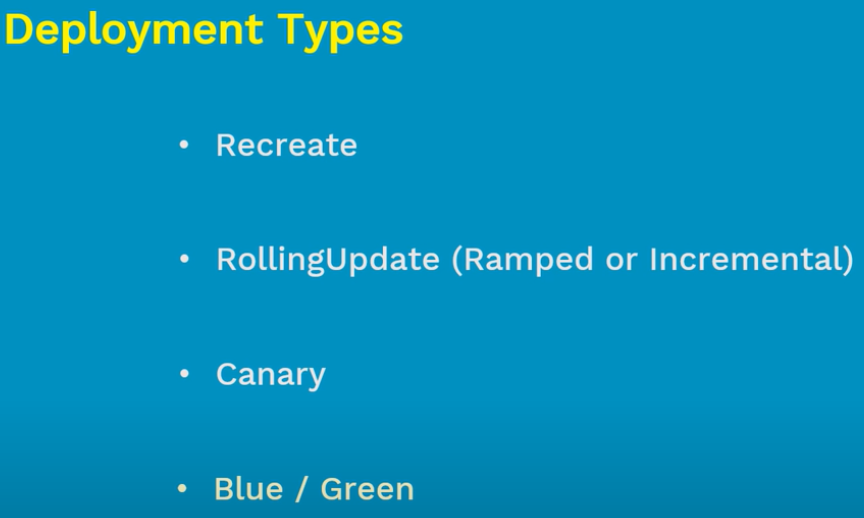


**Features:**

* When we create deployment, Kubernetes will automatically create a replicaset behind it. If we don’t mention replicas, then it will be taken as 1 by default.

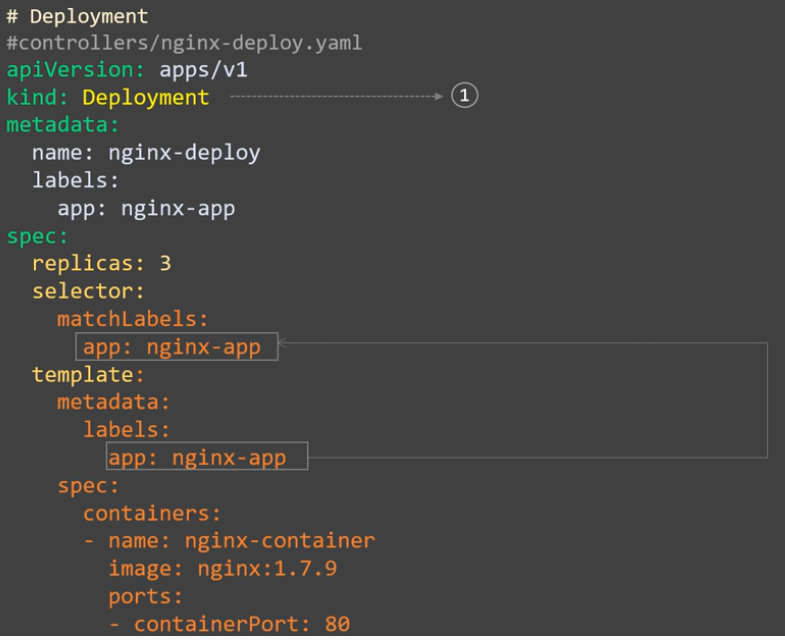


**Types of deployments:**



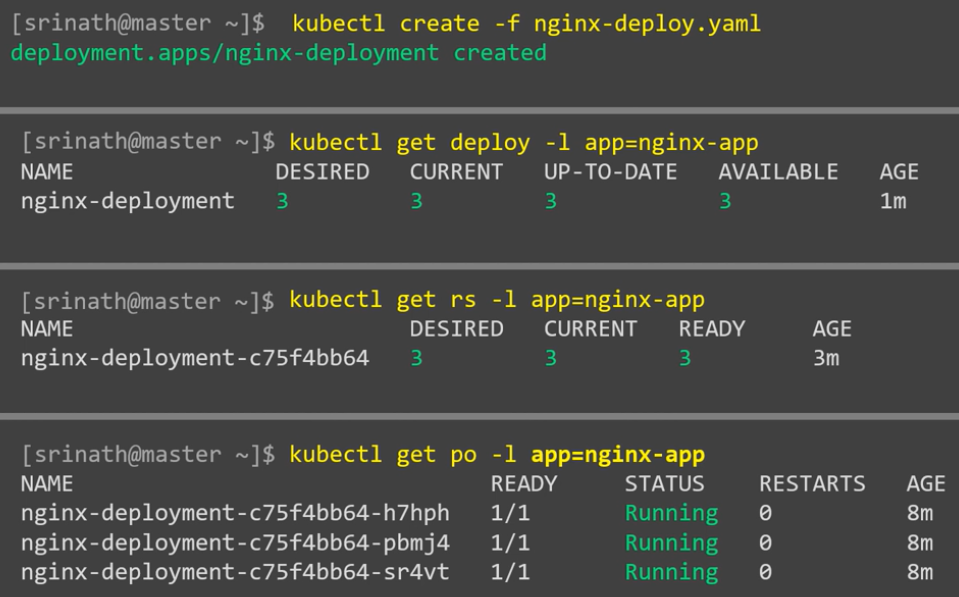
* Let’s say we want to upgrade from version A to B. The **“recreate”** strategy is a dummy deployment which shuts down the version A and once it makes sure it is down, then it will create version B.
* During this switching, there will be a downtime of the service.
* **“rolling update”** slowly rollout the apps by installing one after the other until all the instances are successfully rolled out.
* Let’s say there are 10 instances of version A running behind the load balancer. Then the update strategy upgrades the one instance from the pool. If that one is ready, then it will replace it with version B by removing one from version A.
* **it** is the default strategy in Kubernetes. But it takes some time.
* **“canary”** is the idle deployment method used when someone wants to test the application before deploying it.
* Let’s say we have 10 instances with version A. and if we want to deploy the version B on only 2 instance and test. If all works well. We can go for the deployment of rest 8 instances as well.
* **“blue/green”** deployment means deploying all the instances at once like green with version B deployed along side with blue version A with the same number of instances.
* Once new upgrade meets all the requirements, the traffic will be switched from version A to B. advantage of this one is instant rollout and rollback.

**Practical:**

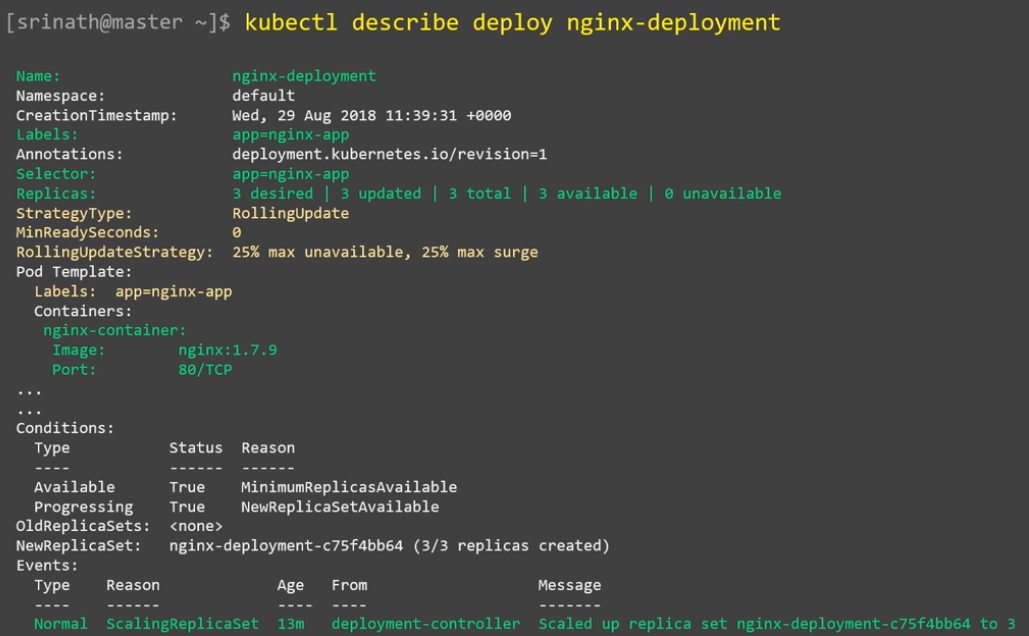


* The labels under the metadata is optional but it comes handy when we are managing deployments.

**Create & display:**



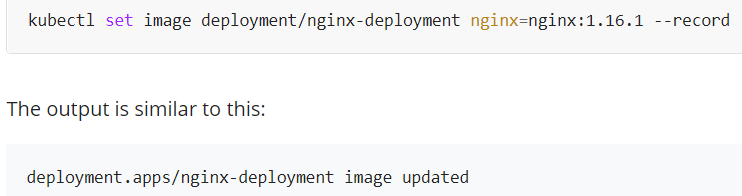
**Describe:**



* We can see the **“rolling update”** is the default strategy added to the deployment as above.

**Update:**

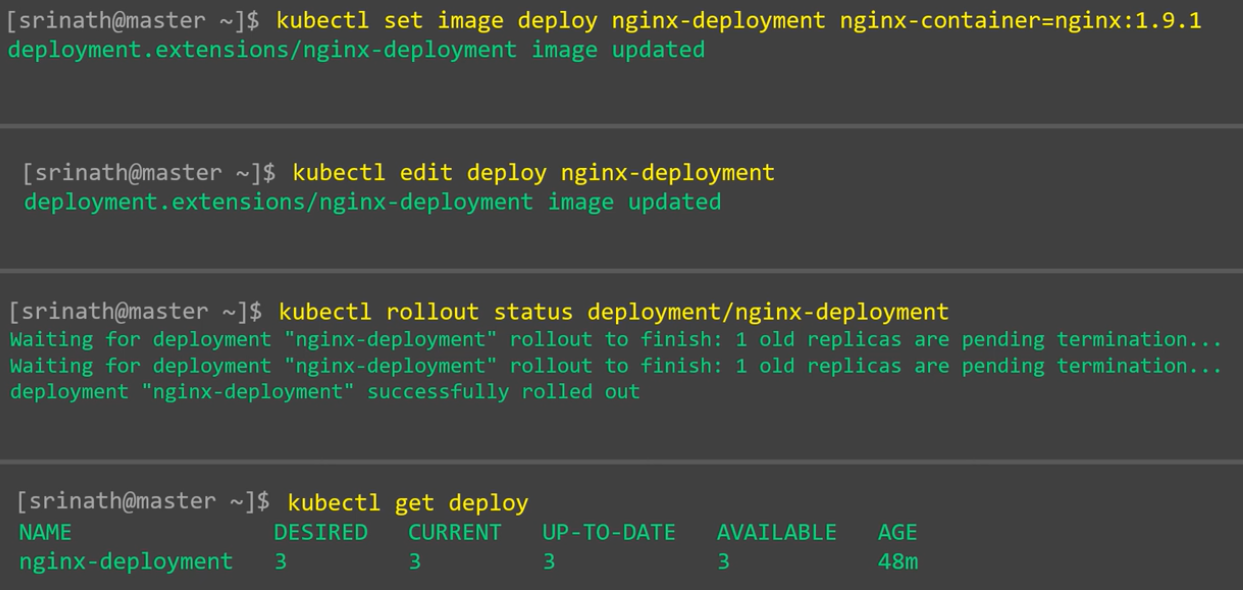
* We can update the application either with **“set”** or **“edit”** command.
* We can simply run the set command as above with the updated image details
* Or we can edit the deployment configuration with edit command as above. Once we run the above command, it will open the deployment configuration in vi editor. We can change the image details and save it.
* Once we save the configuration. It will be automatically applied on the cluster.





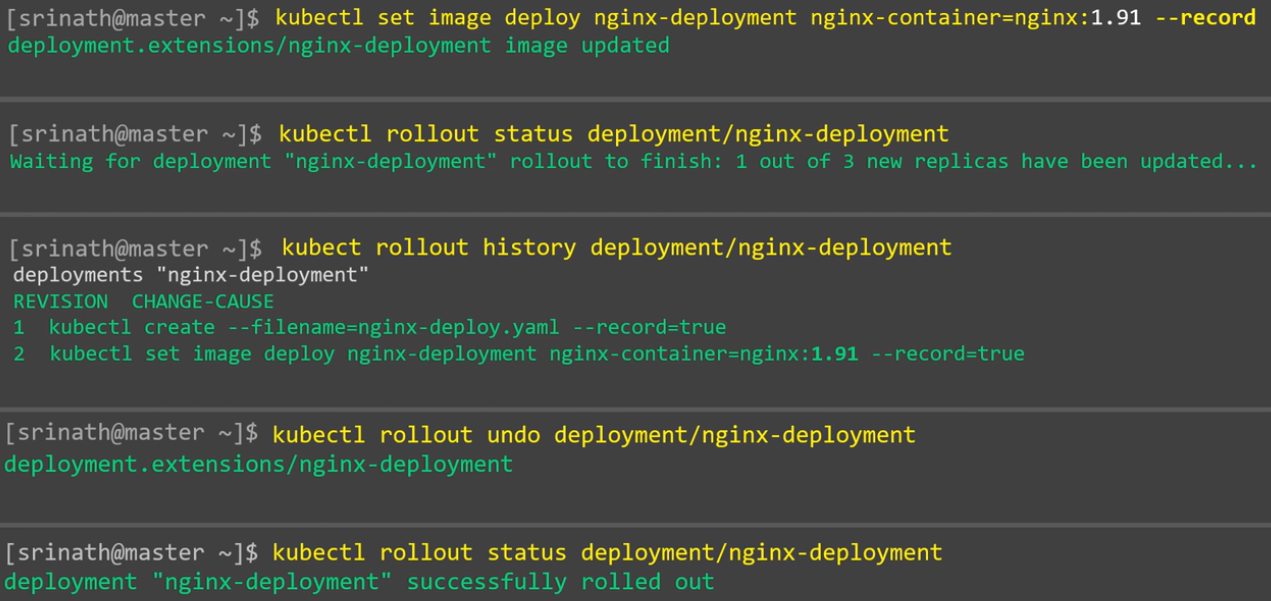
* **image deployments/tds-administration tds-administration=citfodvcr.azurecr.io/tds-administration:$(Build.BuildId)**
* We can also check the rollout status as above.





* We can also edit the yaml file and apply it again.

**Rollback:**



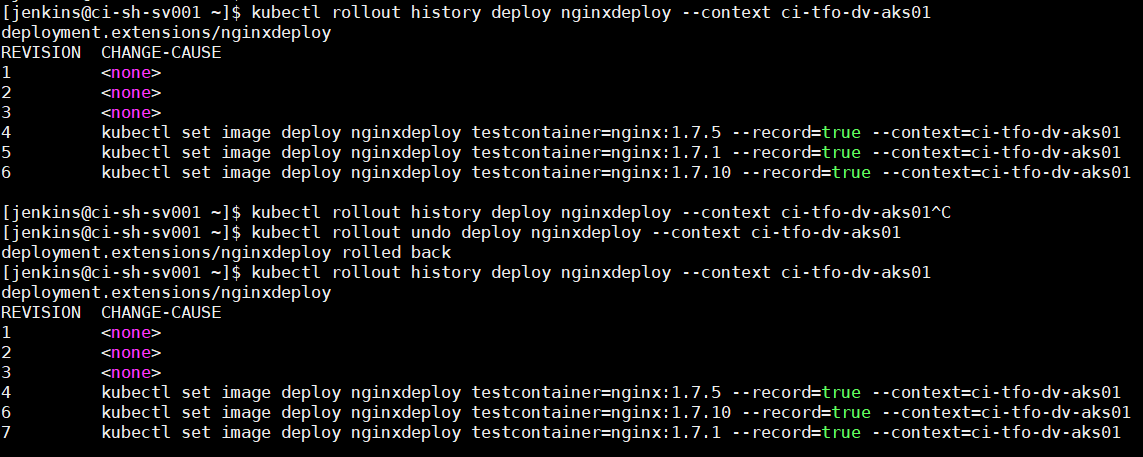
Let’s say we have given wrong nginx version as above in set command. And if we have used **“—record”** option, Kubernetes will record the command we execute.

* **kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:1.161 --record=true**
* **kubectl rollout history deploy tds-administration**

we can use the below command to see the changes of each revision.

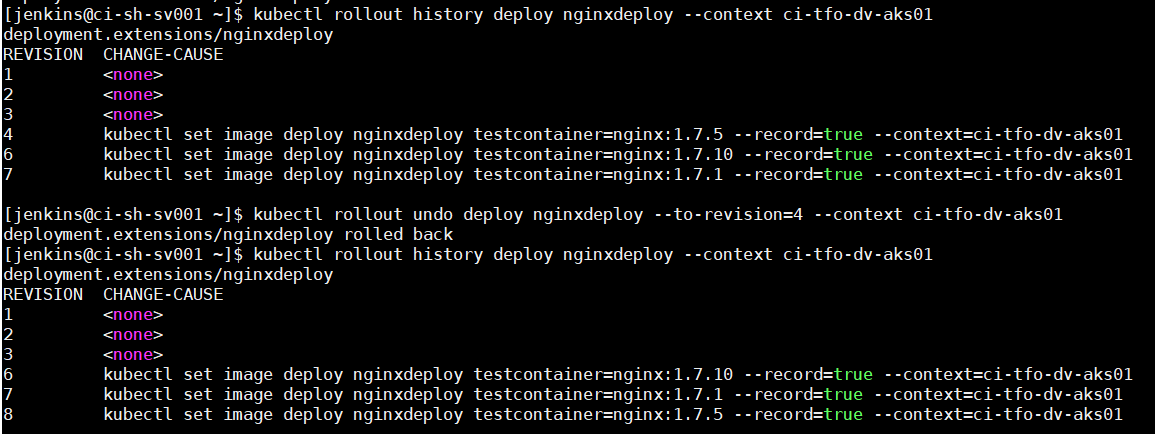
* **Kubectl rollout history deploy tds-administration –revision=2**

By default, Kubernetes rolls back to one version back if we just use undo command as below.



Alternatively, you can roll back to a specific revision by specifying it with --to-revision

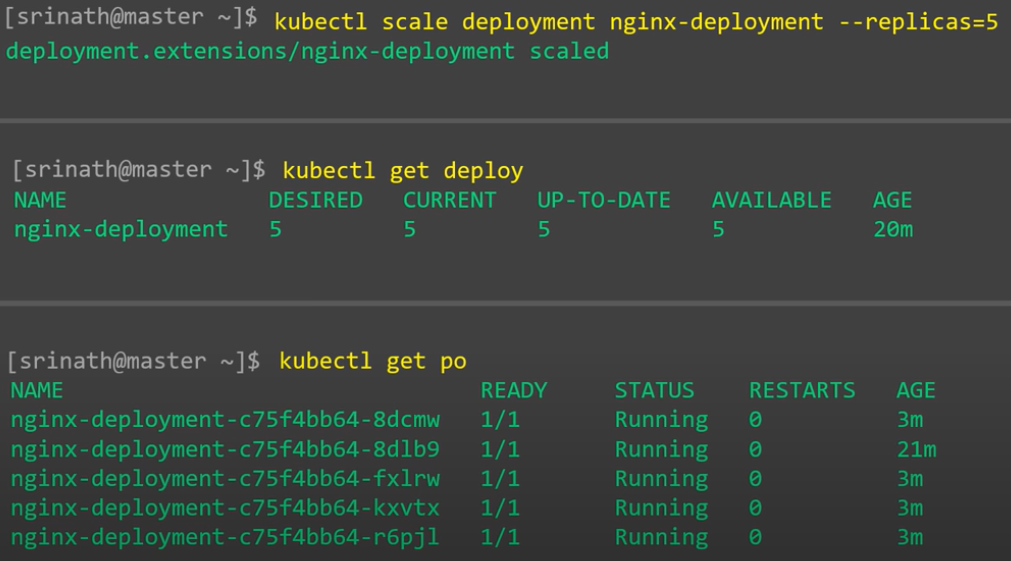
* **kubectl rollout undo deployment.v1.apps/nginx-deployment --to-revision=2**



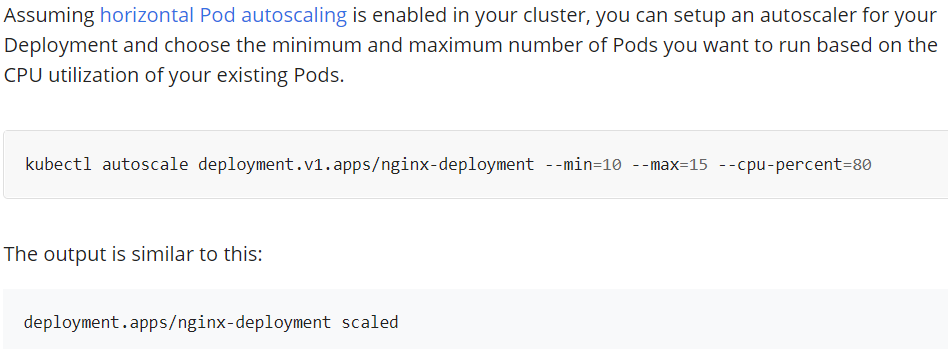
* By default, Kubernetes stores last 10 replicasets and let us rollback to any of them.
* But if we want to increase it, we can do it by changing the deployment specifications as below.



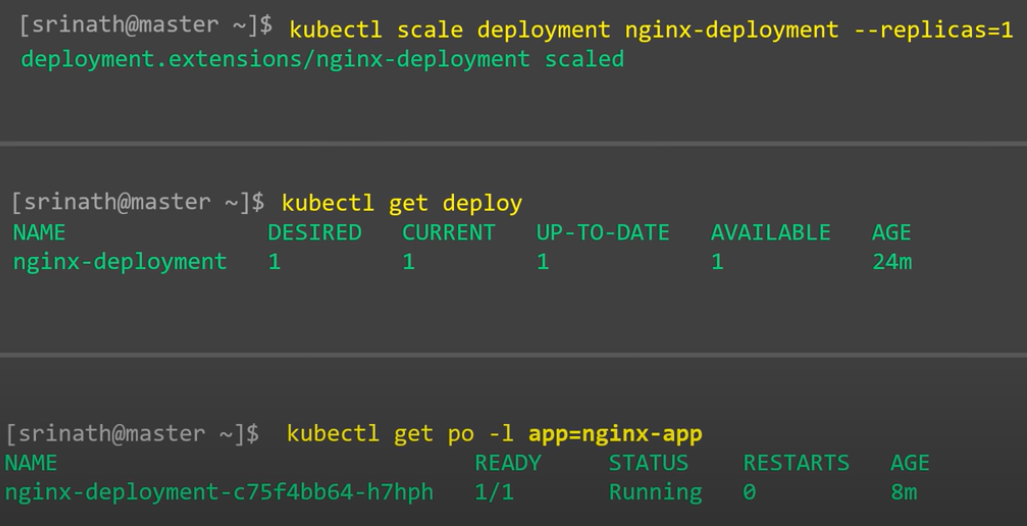
**Scaling up:**



**We can also enable autoscaling as below.**

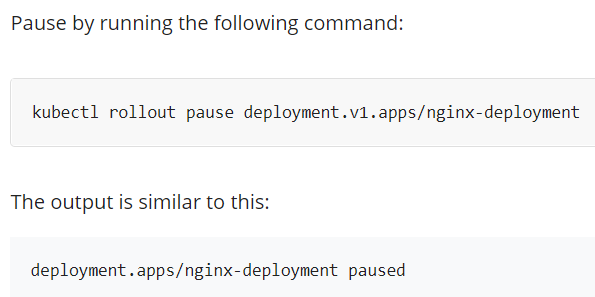


**Scaling down:**

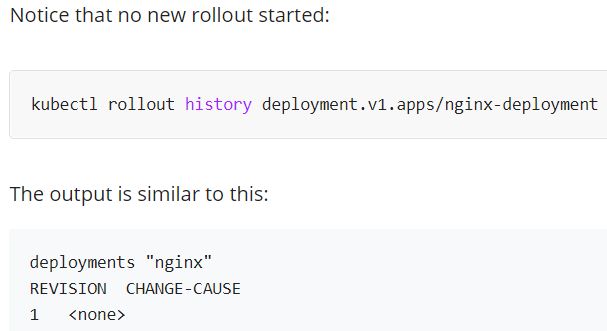


**Pause/resume deployment:**

* When we have too many updates to be done to the deployment. We can pause the deployment and make the changes. But the changes won’t be applied to the deployment immediately. It will be applied when we resume it.







* We can see the deployment rollout status as paused when we describe the deployment asbelow.



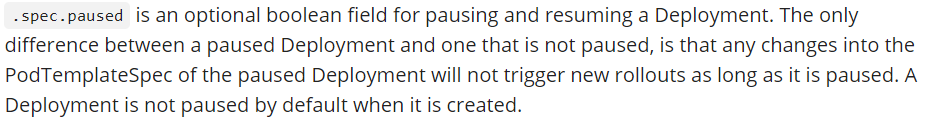
* Even when we use the set command on deployment to update the image. The changes won’t be applied as the deployment is paused.



* After we resume it, we can see the deployment is getting updated with new changes.

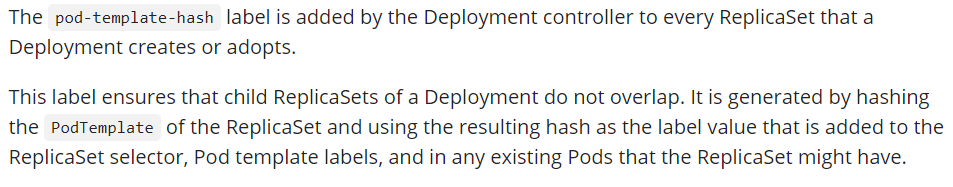
**Note:**

* Pausing the deployment does impact only the pod spec inside deployment. If we increase or decrease the replicas of a dep loyment. It will be applied even if the deployment is in paused state as increasing replicas is on deployment controller.



**Pod-template-hash value:**

* We can see pod template hash value when we get pods with “—show-labels” command.



**Delete:**

