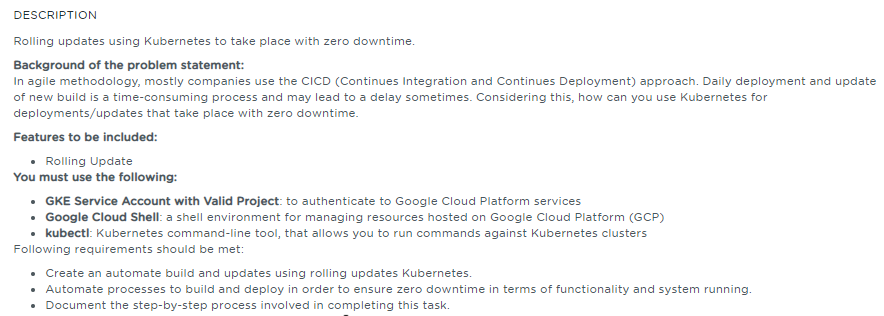
**Build Deployment using Rolling Updates - Assessment**



**Table of Contents**

[**1.** **Introduction** 1](#_Toc44792117)

[**2.** **Work Environment** 1](#_Toc44792118)

[**2.1.** **Prerequisites** 1](#_Toc44792119)

[**2.2.** **Setting up Kubernetes Cluster** 2](#_Toc44792120)

[**3.** **Performing Rolling Update** 5](#_Toc44792121)

[**3.1.** **Deployment Rolling Update** 5](#_Toc44792122)

[**3.1.1.** **Creating a Deployment** 5](#_Toc44792123)

[**3.1.2.** **Updating a Deployment** 7](#_Toc44792124)

[**3.1.3.** **Checking Rollout History of a Deployment** 8](#_Toc44792125)

[**3.1.5.** **Rolling Back to a Previous Version of Deployment** 8](#_Toc44792126)

[**3.1.5.** **Pausing and Resuming a Deployment** 9](#_Toc44792127)

[**3.2.** **StatefulSets Update** 9](#_Toc44792128)

[**3.2.1.** **Creating a StatefulSet** 9](#_Toc44792129)

[**3.2.2.** **Examining a StatefulSet** 10](#_Toc44792130)

[**3.2.3.** **Updating StatefulSet** 10](#_Toc44792131)

[**4.** **References** 11](#_Toc44792132)

# **Introduction**

The intent of this project is to perform the following activities:

* Automate the build/deployment creation and updation using Rolling updates.
* Automate the build and deploy processes to ensure zero down time in terms of functionality, and process that is running in production.

# **Work Environment**

## **Prerequisites**

Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster.

**Note:**

Not using the GKE Cluster as mentioned in the problem statement and requirement specification as am facing issue in creating the Cluster in my Trial account

Am using the SimpliLearn Practice lab session to implement this assessment.

## **Setting up Kubernetes Cluster**

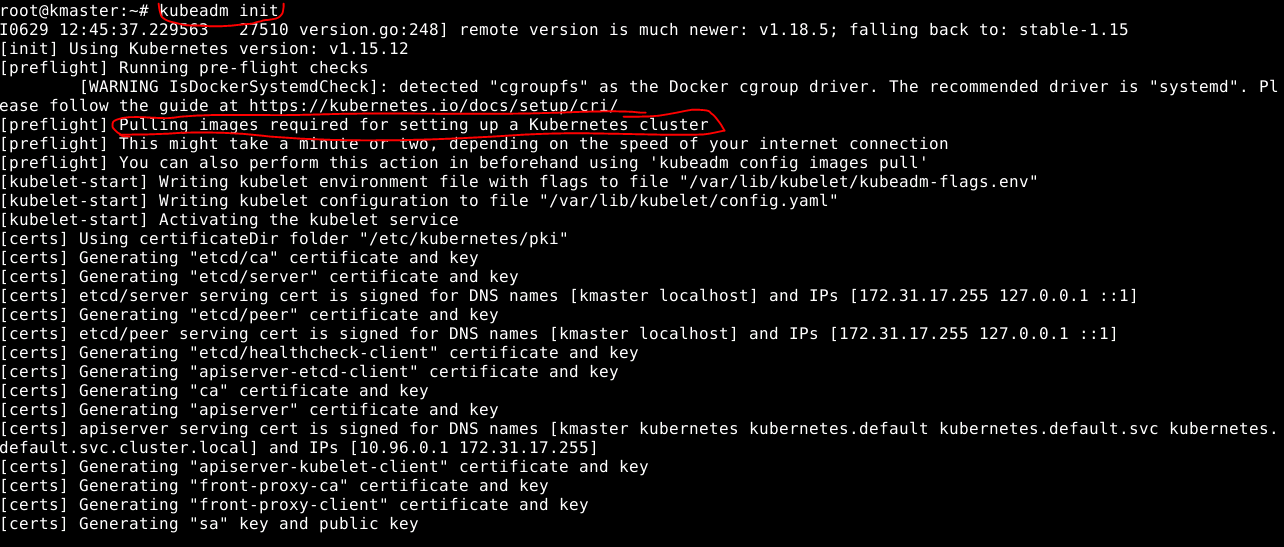
Using the SimpliLearn Practice Lab session, Kubernetes Cluster is being setup with one master and two worker nodes

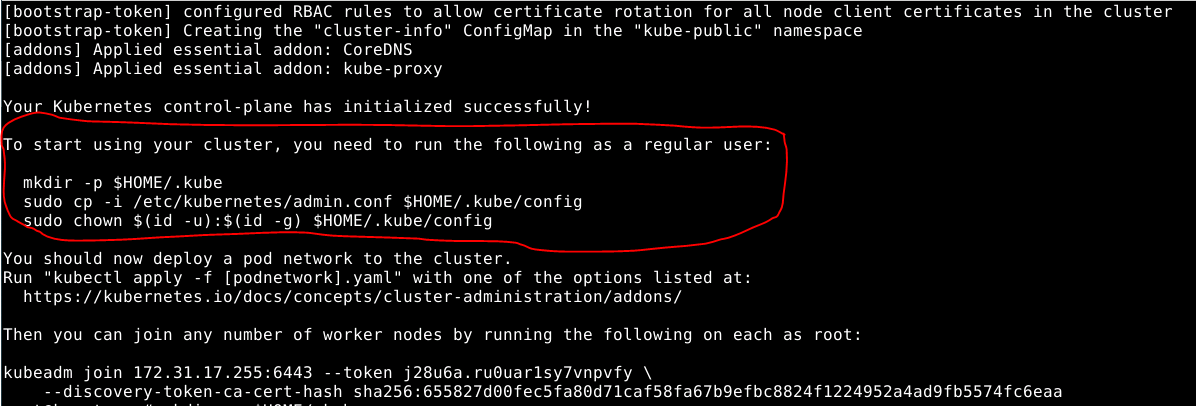
Run the below command on the master

|  |
| --- |
| $ sudo –i  $ kubeadm reset |

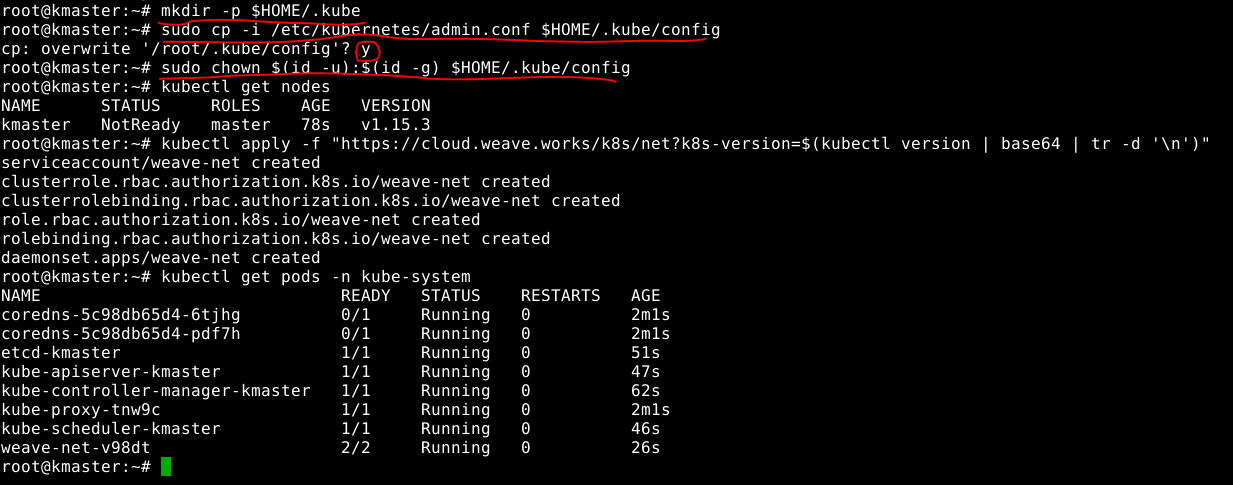
**kubeadm reset** is responsible for cleaning up a node local file system from files that were created using the kubeadm init or kubeadm join commands

Run the below command in order to set up the Kubernetes control plane

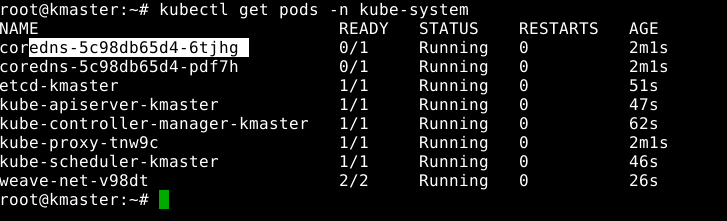




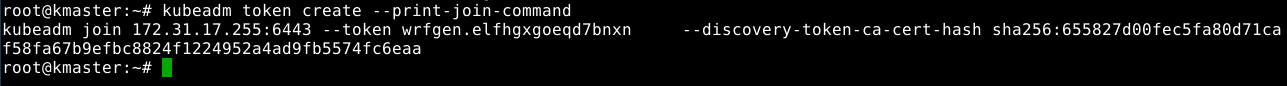
After executing the init command, below screenshot shows various commands used to initialize the configuration and set the correct permissions



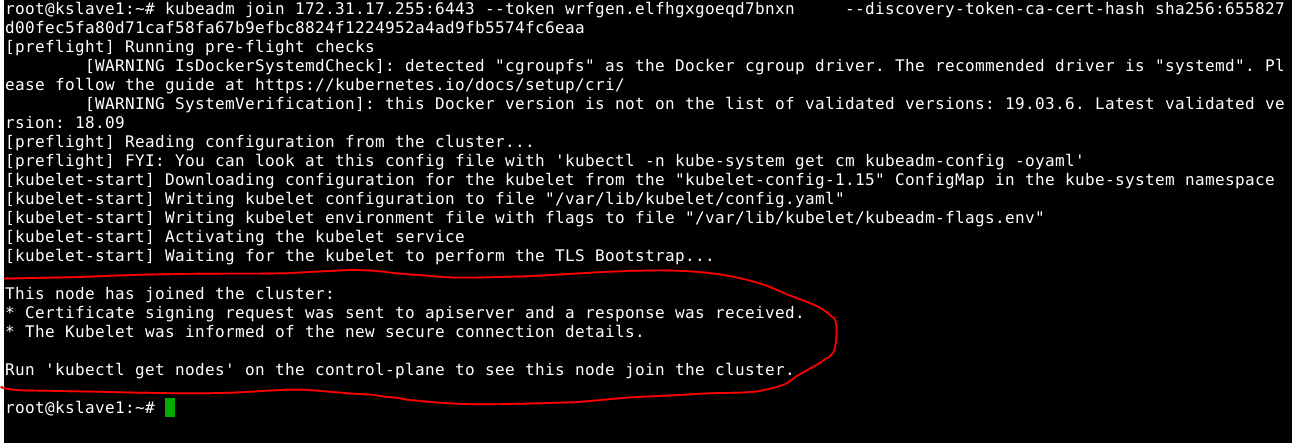
Below screenshot shows the command to verify the master has all the required components running successfully



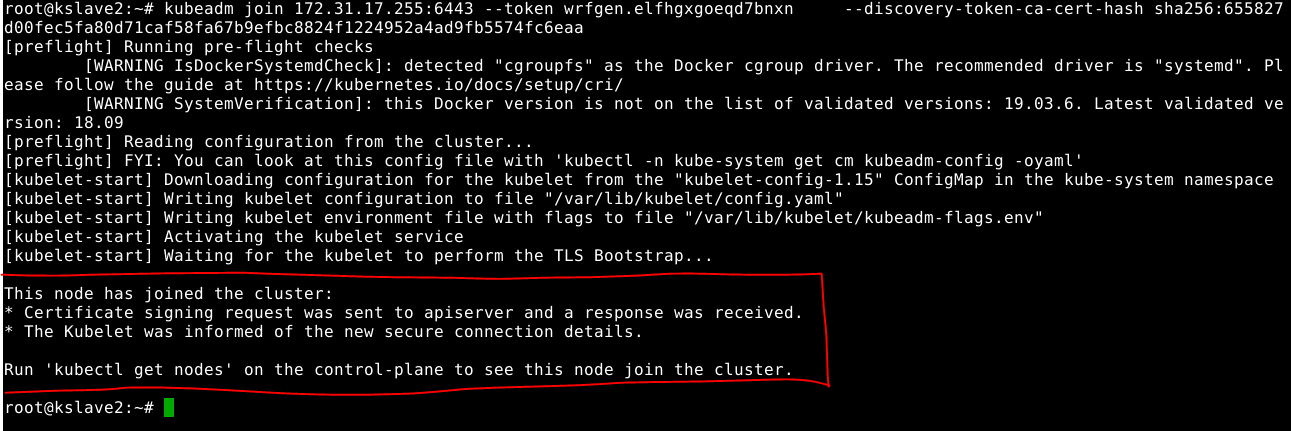
Below screenshot shows the command usage on generating the token for joining the slave nodes in the cluster



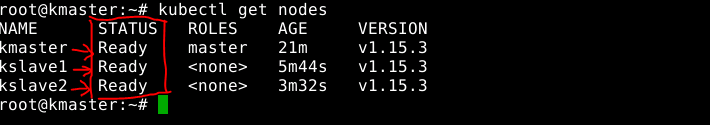
Below screenshot shows the command usage on to join the slave node (kslave1) to the cluster



Below screenshot shows the command usage on to join the slave node (kslave2) to the cluster



Below screenshot shows the all the nodes status in the cluster



# **Performing Rolling Update**

Users expect application to be available all the time and developers are expected to deploy new versions of them several times a day. In Kubernetes this is done by Rolling Update.

Rolling Update allows Kubernetes workloads like Deployment to update itself with zero downtime by incrementally updating Pods instances with the new ones. The new pods will be scheduled on Nodes with available resources.

The following objects represent Kubernetes workloads. You can trigger a rolling update on these workloads by updating their pod template:

* Deployment
* StatefulSets
* DaemonSets

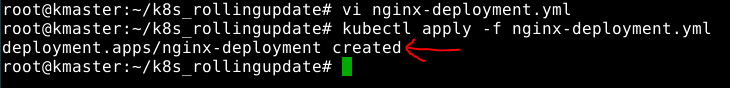
Here we shall see how to perform the Rolling updates for the Deployment and StatefulSets workloads.

## **Deployment Rolling Update**

## **Creating a Deployment**

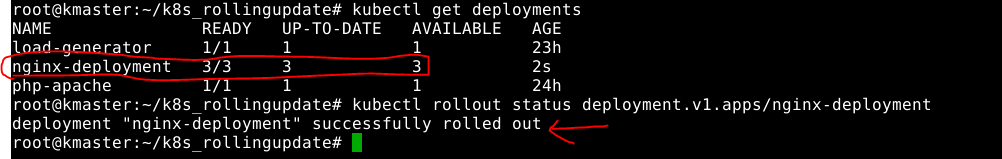
The following is an example of a Deployment definition file. It creates a ReplicaSet to bring up 3 nginx Pods:



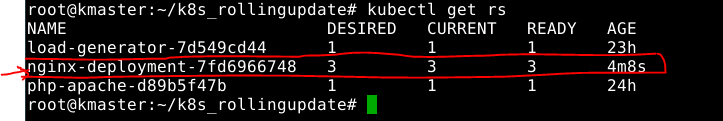


Notice that the Deployment has created all three replicas, and all replicas are up-to-date (they contain the latest Pod template) and available.

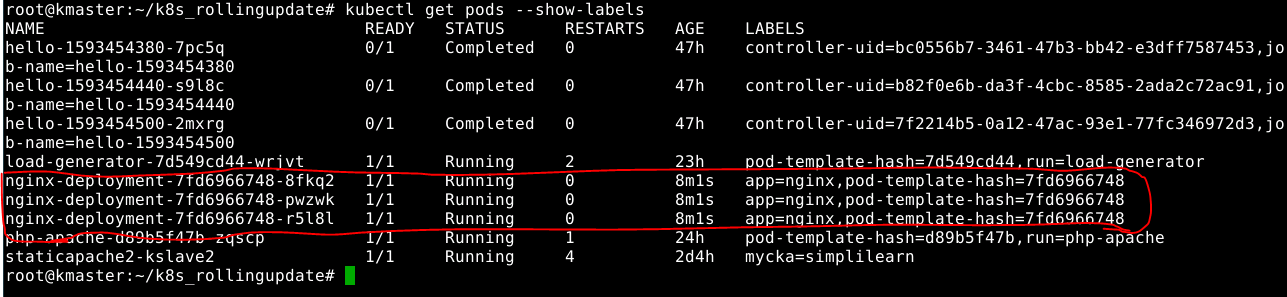
The below screenshot shows that the deployments have been created with all three replicas, and all replicas are up-to-date and available



Thebelow screenshot shows the ReplicaSets created by the deployment



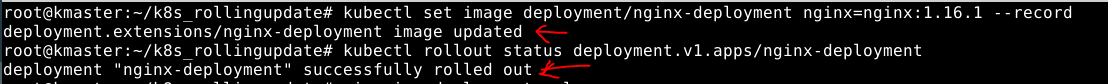
The below screenshot depicts that the created ReplicaSets ensures that there are three nginx Pods.



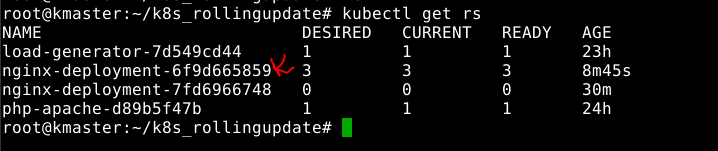
## **Updating a Deployment**

Below screenshot shows on updating the nginx Pods to use the nginx:1.16.1 image instead of the nginx:1.14.2 image.

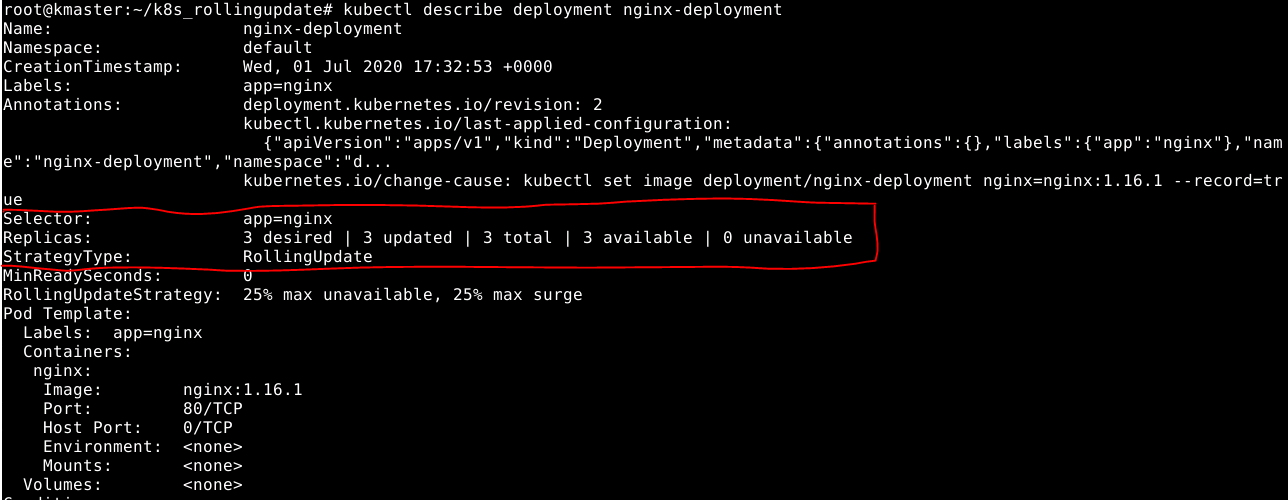
Also, it shows the rollout status of an updated deployment



Below screenshot shows the output of running the command **kubectl get rs** to see that the Deployment updated the Pods by creating a new ReplicaSet and scaling it up to 3 replicas, as well as scaling down the old ReplicaSet to 0 replicas

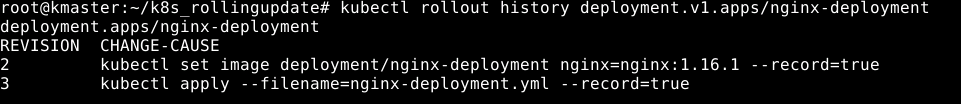


Below screenshot depicts the details of the nginx-deployment. Here you can find all the details like it’s Namespace, Annotations, Selector, Replicas, StrategyType, PodTemplate details etc as shown below



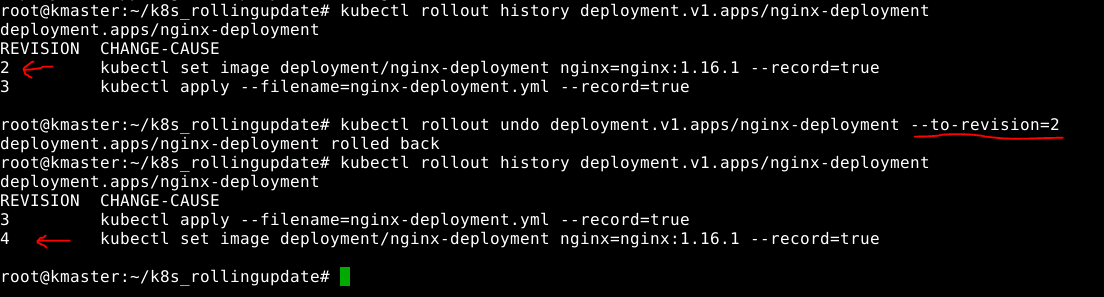
## **Checking Rollout History of a Deployment**

Below screenshot shows the command output i.e changes related to the revisions of the deployment history.



## **Rolling Back to a Previous Version of Deployment**

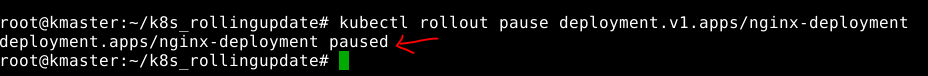
Below screenshot shows the command execution of Rolling back to the previous version i.e undo the changes and roll back to the respective revision.



## **Pausing and Resuming a Deployment**

You can pause a Deployment before triggering one or more updates and then resume it. This allows you to apply multiple fixes in between pausing and resuming without triggering unnecessary rollouts.

Below screenshot shows the deployment pause command and its output



Eventually, resume the Deployment by running the below command and observe a new ReplicaSet coming up with all the new updates which has been made pausing

|  |
| --- |
| **$ kubectl rollout resume deployment.v1.apps/nginx-deployment** |

## **StatefulSets Update**

* StatefulSets is a Kubernetes resource used to manages stateful applications.
* It manages the deployment and scaling of set of Pods, and provides guarantee about the ordering and uniqueness of these Pods.
* StatefulSets is also a Controller but unlike Deployment, it doesn’t create ReplicaSet rather itself creates Pods with a unique naming convention.
* Every replica of a StatefulSet will have its own state, and each of the pods will be creating its own PVC (Persistent Volume Claim). So a StatefulSet with 3 replicas will create 3 pods, each having its own Volume, so total 3 PVCs.

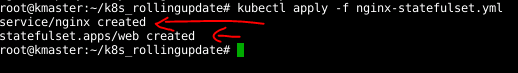
We can begin by creating a StatefulSet using the example below. It is similar to the example presented in the [StatefulSets](https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/) concept. It creates a [headless Service](https://kubernetes.io/docs/concepts/services-networking/service/#headless-services), nginx, to publish the IP addresses of Pods in the StatefulSet, web.

## **Creating a StatefulSet**

Below screenshot shows the contents of the **nginx-statefulset.yml**



Below screenshot shows the creation of Headless service and Statefulset define in **nginx-statefulset.yml**



## **Examining a StatefulSet**

Pods in a StatefulSet have a sticky, unique identity. This identity is based on a unique ordinal index that is assigned to each Pod by the StatefulSet [controller](https://kubernetes.io/docs/concepts/architecture/controller/).

Run the below command to get the StatefulSet’s Pods

|  |
| --- |
| **$ kubectl get pods –l app=nginx** |

The output is similar to this:

**NAME READY STATUS RESTARTS AGE**

**web-0 1/1 Running 0 1m**

**web-1 1/1 Running 0 1m**

## **Updating StatefulSet**

Updating StatefulSets feature can be used to upgrade the container images, resource requests and/or limits, labels, and annotations of the Pods in a StatefulSet. There are two valid update strategies, **RollingUpdate** and **OnDelete**

The *RollingUpdate* update strategy will update all Pods in a StatefulSet, in reverse ordinal order, while respecting the StatefulSet guarantees.

Patch the web StatefulSet to apply the RollingUpdate update strategy:

|  |
| --- |
| **$ kubectl patch statefulset web -p '{"spec":{"updateStrategy":{"type":"RollingUpdate"}}}'** |

The output is similar to this:

**statefulset.apps/web patched**

# **References**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Components** | **Reference** |
| 1 | Write-up document | 5.CI\_CD\_Deployment\_using\_Rolling\_updates\_writeup.docx |
| 2 | Sources document | 5.CI\_CD\_Deployment\_using\_Rolling\_updates\_sources.docx |
| 3 | Kubernetes.io | <https://kubernetes.io/docs/> |