**Pod Status**

A pod has five possible statuses:

1. **Pending:** This state shows at least one container within the pod has not yet been created.
2. **Running:** All containers have been created, and the pod has been bound to a Node. At this point, the containers are running, or are being started or restarted.
3. **Succeeded:** All containers in the pod have been successfully terminated and will not be restarted.
4. **Failed:** All containers have been terminated, and at least one container has failed. The failed container exists in a non-zero state.
5. **Unknown:** The status of the pod cannot be obtained.

If you notice a pod in an undesirable state where the status is showing as ‘error’, you might try a ‘restart’ as part of your troubleshooting to get things back to normal operations. You may also see the status ‘[CrashLoopBackOff](https://spacelift.io/blog/crashloopbackoff" \t "_blank)’, which the default when an error is encountered, and K8S trys to restart the pod automatically.

**Restart a pod**

You can use the following methods to ‘restart’ a pod with kubectl. Once new pods are re-created they will have a different name than the old ones. A list of pods can be obtained using the kubectl get pods command.

**Method 1**

kubectl rollout restart

This method is the recommended first port of call as it will not introduce downtime as pods will be functioning. A rollout restart will kill one pod at a time, then new pods will be scaled up. This method can be used as of K8S v1.15.

kubectl rollout restart deployment <deployment\_name> -n <namespace>

**Method 2**

kubectl scale

This method will introduce an outage and is not recommended. If downtime is not an issue, this method can be used as it can be a quicker alternative to the kubectl rollout restart method (your pod may have to run through a lengthy Continuous Integration / Continuous Deployment Process before it is redeployed).

If there is no YAML file associated with the deployment, you can set the number of replicas to 0.

kubectl scale deployment <deployment name> -n <namespace> --replicas=0

This terminates the pods. Once scaling is complete the replicas can be scaled back up as needed (to at least 1):

kubectl scale deployment <deployment name> -n <namespace> --replicas=3

Pod status can be checked during the scaling using:

kubectl get pods -n <namespace>

**Method 3**

[kubectl delete pod](https://spacelift.io/blog/kubectl-delete-pod) and kubectl delete replicaset

Each pod can be deleted individually if required:

kubectl delete pod <pod\_name> -n <namespace>

Doing this will cause the pod to be recreated because K8S is declarative, it will create a new pod based on the specified configuration.

However, where lots of pods are running this is not really a practical approach. Where lots of pods have the same label however, you could use that to select multiple pods at once:

kubectl delete pod -l “app:myapp” -n <namespace>

Another approach if there are lots of pods, then ReplicaSet can be deleted instead:

kubectl delete replicaset <name> -n <namespace>

**Method 4**

kubectl get pod | kubectl replace

The pod to be replaced can be retrieved using the kubectl get pod to get the YAML statement of the currently running pod and pass it to the kubectl replace command with the --force flag specified in order to achieve a restart. This is useful if there is no YAML file available and the pod is started.

kubectl get pod <pod\_name> -n <namespace> -o yaml | kubectl replace --force -f -

**Method 5**

kubectl set env

Setting or changing an [environment variable](https://spacelift.io/blog/kubernetes-environment-variables) associated with the pod will cause it to restart to take the change. The example below sets the environment variable DEPLOY\_DATE to the date specified, causing the pod to restart.

kubectl set env deployment <deployment name> -n <namespace> DEPLOY\_DATE="$(date)"