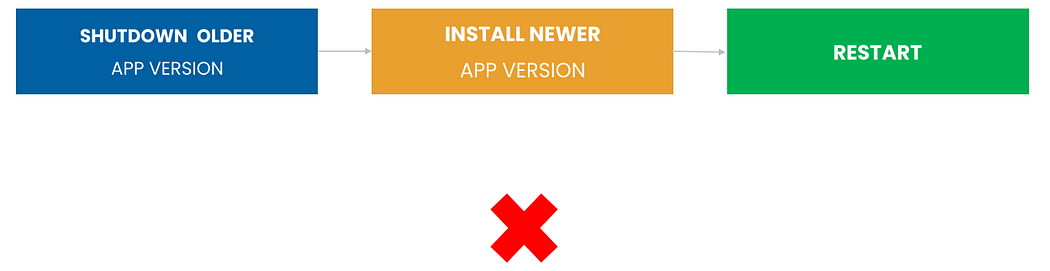
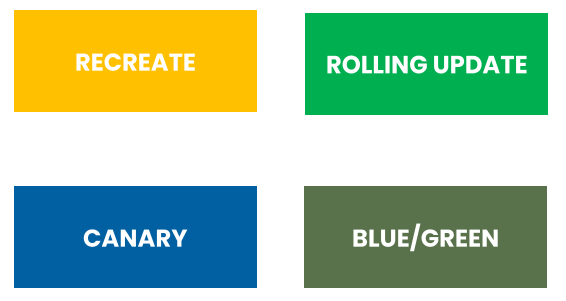
**Kubernetes Deployment Strategies**



* In the old days, we typically use to upgrade applications by just shutting down the older application and installing the newer application, and restarting the server.
* This is the biggest challenge is to speed up the deployment speed.
* Kubernetes provides an advanced platform for making cloud-native deployments easy.

Here, we will discuss various different strategies to upgrade applications from the older version to newer version capabilities.



**1. Recreate**



* Recreate deployment strategy will terminate all the running pod instances and then recreate them with the newer version.
* It is also called dummy deployment.
* This technique implies downtime of the application that depends on both the shutdown and boot duration of the application.

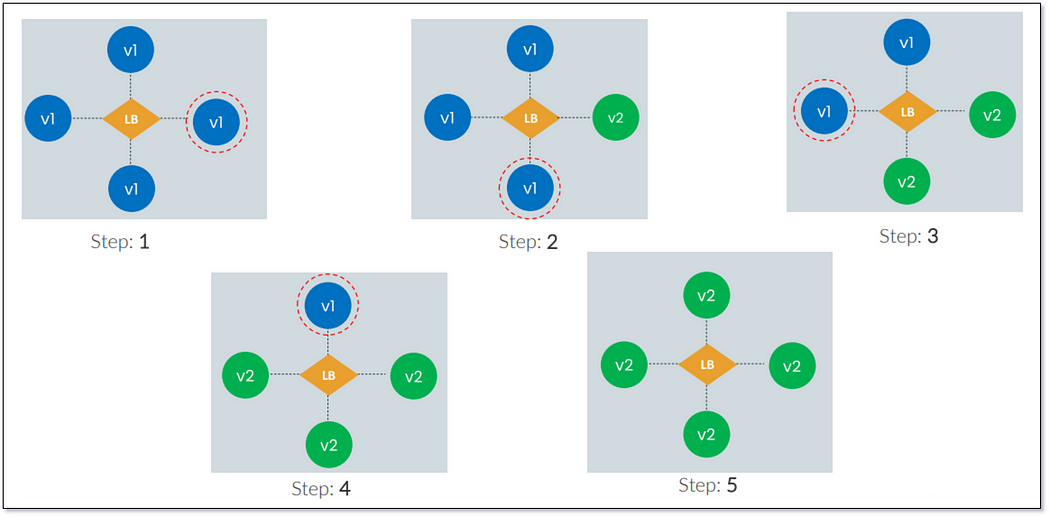
**Pros**

* When it comes to easy setup, Recreate is a good option.
* Application state entirely renewed.

**Cons**

* High impact on the user, expect downtime that depends on both shutdown and boot duration of the application.

**2. Rolling Update**



* This deployment strategy is the default strategy in Kubernetes.
* In this, pods of the previous version are slowly replaced one by one pod of the new version without any cluster downtime.
* Also, this is one of the best deployment strategies in Kubernetes.
* For example, let’s imagine an application with four pods.
* When the developers start to upgrade to the newer version, pods are upgraded one at a time until all pod instances are replaced with the newer applications.

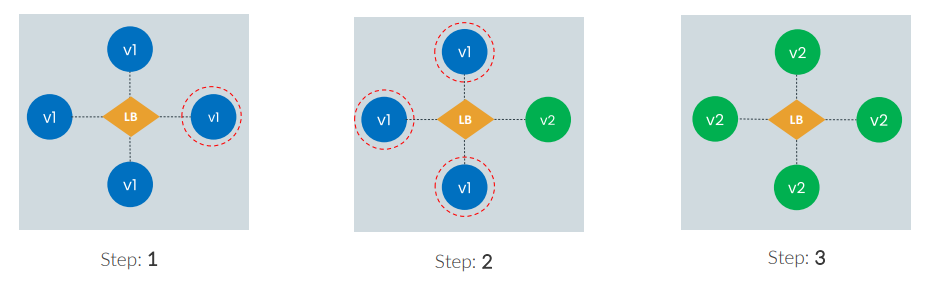
**Pros**

* Slow-release of version across instances.
* Useful for stateful applications that can handle the data.

**Cons**

* In this strategy, rollout and rollback take time.
* There is no control over the traffic.

**3. Canary**



* In this Kubernetes deployment strategy, traffic is split based on weight.
* A canary deployment consists of slowly moving production traffic from version A to version B.
* This is an ideal deployment strategy for someone who wants to test the newer version before it is 100 percent deployed.
* In the above example, First, you may upgrade from version1 to version2 and do then do some testing, If the test results are good, then you can upgrade the remaining three pod instances.

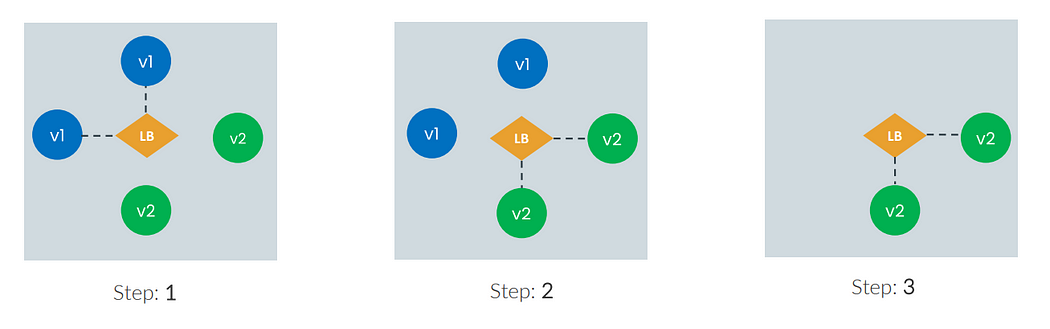
**Pros**

* In this strategy, rollback is fast.
* Convenient for monitoring application performance and error rates.

**Cons**

* The rollout is slow in this strategy.

**4. Blue/Green**



* Using this deployment strategy, we deploy the same number of the newer versions of the app along with the older version of the instance.
* Then switch the user traffic from older to newer instances, if there are any issues with the newer version, it is very easy to switch back to the older version.
* We can remove the older version when your newer one is working fine.

**Pros**

* rollout and rollback are fast.
* Versioning issues are fixed as a whole application state is changed completely.

**Cons**

* The cost factor, as this strategy requires double the resources.
* Required proper testing of application before deploying it to production.