Kubernetes Deployment Strategies

Kubernetes provides an advanced platform for making cloud-native deployments easy. Today, I will show you **various strategies to upgrade applications** from the older version to newer version capabilities.

Source: Bikram, Medium

RECREATE

ROLLING UPDATE

CANARY

BLUE/GREEN

1. Recreate

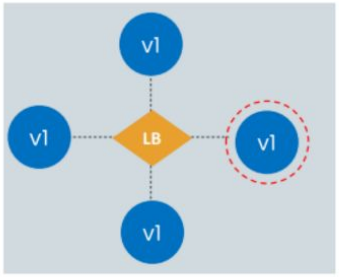
The recreate deployment strategy will terminate all the running pod instances and recreate them with the newer version. This technique implies downtime of the application that depends on both the **shutdown and boot duration** of it.

Pros

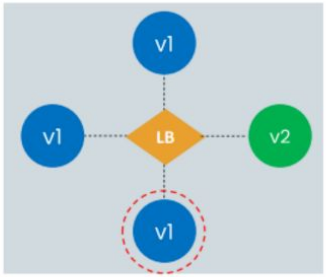
- When it comes to **easy setup,** Recreate is a good option.
- Application state entirely renewed.

Cons

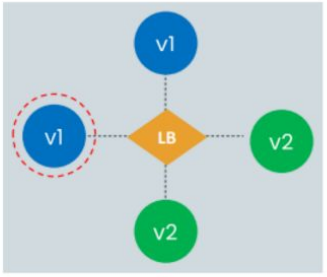
- **High impact on the user,** expect downtime that depends on both shutdown and boot duration of the application.



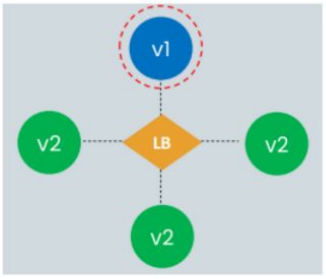
Step: 1



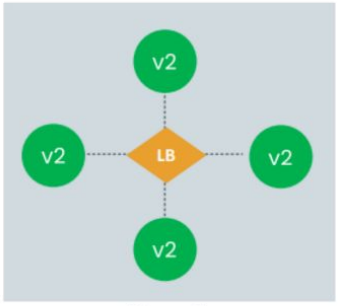
Step: 2



Step: 3



Step: 4



Step: 5

2. Rolling Update

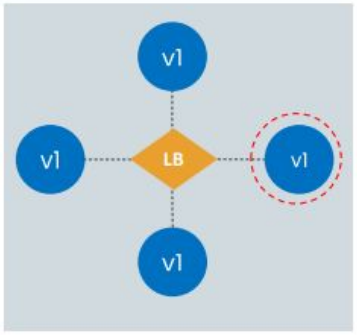
This deployment strategy is the default strategy in Kubernetes. In this, **pods of the previous version are slowly replaced** one by one pod of the new version without any cluster downtime.

Pros

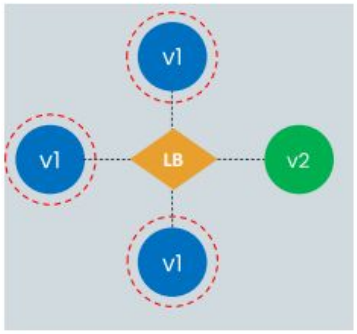
- Slow-release of version across instances.
- Useful for stateful applications **that can handle the data.**

Cons

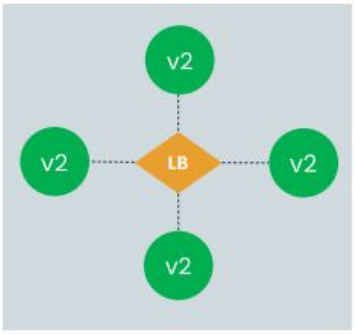
- In this strategy, rollout and rollback **take time.**
- There is no control over the traffic.



Step: **1**



Step: **2**



Step: **3**

3. Canary

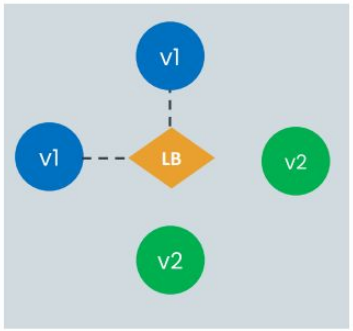
A canary deployment consists of **slowly moving production traffic** from version A to version B.
This is an ideal deployment strategy for someone who wants to test the newer version before it is 100% deployed.

Pros

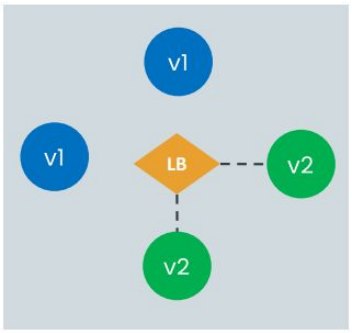
- In this strategy, **rollback is fast.**
- Convenient for monitoring of application performance and error rates.

Cons

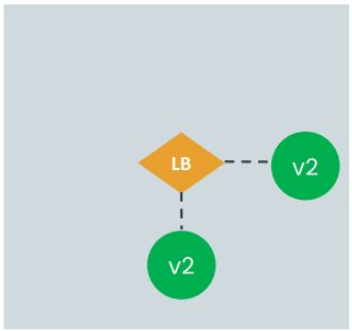
- The rollout **is slow** in this strategy.



Step: 1



Step: 2



Step: 3

4. Blue/Green

Using this deployment strategy, you can 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's easy to switch back to the older version.

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 the production.