Design proposal for best effort pod autoscaling in kubernetes with less disruptions

1. Use node affinity and anti-affinity for scheduling of nodes.

Example:

1. case1: If we want to schedule a pod on nodes which has more resources, we can use operator '**In**' in that case.
2. case2: If we do not want to schedule a pod on critical nodes which has no resources, we can use operator **'Not In'** in this case.
   In second case we can many types of label one with high, moderate and low.

Default Case:

- A process to update the label or value we use in affinity and anti-affinity
  Note: We will not schedule the pods once it reaches 80% as per our calculation (RAM * 0.9) / 300 MB. 300MB can be changed.
- We can use settings **requiredduringschedulingignoredduringexecution**
- In future we can have our custom scheduler.

2. To be extra safe we can even taint the nodes which has more resource usage [it is redundant as we are updating labels already]

3. Use priority efficiently with pods

- Documentation says that which has more priority has a less chance of getting evicted.
- We can have a process which updates the pods priority based on resource utilization and other factors in a node.
- I didn't get a solution to update the pods priority while pod running.
- If update priority does not work, we can try updating of priority classes value itself.

Important Note: All this are just a solution based on research but must be tested. if it works can be improved or can be evicted [most likely rebalancer won't work because updating of spec is not supported].

Important Resource

1. Affinity / Anti-affinity of node

2. Taint

3. Priority Classes

related links

Priority Classes

https://kubernetes.io/docs/concepts/configuration/pod-priority-preemption

kubernetes best practices

https://cloud.google.com/blog/products/gcp/kubernetes-best-practices-resource-requests-and-limits

Affinity and anti-affinity and taint

https://kubernetes.io/blog/2017/03/advanced-scheduling-in-kubernetes/