Cluster Maintainance

- Releases
  - As for now: the etcd and coredns component has only the different version to other components but the other core components should have the same version.
- OS Upgrade node
  - Some keypoints
    - Single pod replica set get the downtime
    - A single pod if not backed by the controller then it would be inaccessible due to on the maintenance node
    - Can set the pod-eviction-time by kube-controller-manager for the pods
    - Drain the node first is a good practice and then cordon it so that no pods can be scheduled on it
    - Drain can be failed if the pod doesn't have the replica set but force option will delete it which would become inaccessible
- Cluster Upgradation
  - Version mgmt
  - Supported version
  - Controlplane Upgrade
    - Mgmt of cluster get impacted
    - The worker nodes are not impacted, as they can easily get their traffic as usual
    - New pods can't be scheduled and APIserver is not available
  - Worker node upgrade strategies
    - Down and upgrade all the nodes
    - Upgrade one by one
    - Add upgraded node in the cluster and schedule pods on them
  - Kubeadm Upgrade
- Backup and Restore
  - That's why its prefer to have the declarative approach rather then the imperative
  - Approach 1: we can also get the current configuration from all namespaces by querying the API server. But this approach would work for few resources groups. But there are still some imperative commands that could be run. And we can use other third party tools to cover those resources group.
  - Approach 2: As we know that the etcd contains the state of the cluster, so instead of previous approaches, we can take the backups of the etcd. We can also see the data directory where its files are saved. before restoring the cluster from etcd backup, we have to stop the kubeapi server because it depends on the etcd. when we restore it, we have to redefine a new data directory for the etcd to prevent the nodes to join the old cluster accidentally