Best practices for Openshift

Importance of masters and ETCD

ETCD which is running on masters is key-value database that holds all data and objects of cluster. Any performance issues (timeouts, delays) can lead to unstable cluster.

Namespaces

By default, there are 3 namespaces in a K8s cluster, default, kube-public and kube-system and additionaly openshift-* namespaces in OCP4.

Pods

Don't use naked Pods (that is, Pods not bound to a ReplicaSet or Deployment) if you can avoid it.

Readiness and Liveness Probes

Readiness and liveness probes are strongly recommended; it is almost always better to use them than to forego them. These probes are essentially health checks.

Readiness probe Ensures a given pod is up-and-running before allowing the load to get directed to that pod. If the pod is not ready, the requests are taken away from your service until the probe verifies the pod is up. Liveness probe Verifies if the application is still running or not. This probe tries to ping the pod for a response from it and then check its health. If there is no response, then the application is not running on the pod. The liveness probe launches a new pod and starts the application on it if the check fails.

Resource Requests and Limits

Resource limits are the operating parameters that you provide to kubernetes that tell it two critical things about your workload: what resources it requires to run properly; and the maximum resources it is allowed to consume. The first is a critical input to the scheduler that enables it to choose the right node on which to run the pod. The second is important to the kubelet, the daemon on each node that is responsible for pod health.

```
resources:
  requests:
    cpu: 50m
    memory: 50Mi
  limits:
    cpu: 100m
    memory: 100Mi
```

Resource requests specify the minimum amount of resources a container can use Resource limits specify the maximum amount of resources a container can use.

Deployments and Replicasets

.spec.revisionHistoryLimit is an optional field that specifies the number of old ReplicaSets to retain to allow rollback. These old ReplicaSets consume resources in etcd and crowd the output of kubectl get rs. The configuration of each Deployment revision is stored in its ReplicaSets; therefore, once an old ReplicaSet is deleted, you lose the ability to rollback to that revision of Deployment. By default, 10 old ReplicaSets will be kept, however its ideal value depends on the frequency and stability of new Deployments.

This means, that with 2000 deployments you could create up to 20k replicasets and this could have huge performance impact on ETCD.

Operators

TODO

What is Operator doing?

TODO

Where does Operator run?

If Operator runs also on masters (virus or file scanner), it could have performance impact on storage (and therefor impact on ETCD).

Does Operator call OCP API?

TODO

Does Operator create CRDs?

TODO

Pipelines

TODO

3rd party software and services

TODO

Logging

Customer applications produce logs and you should consider add enough resources (CPU/RAM) to monitoring/infra nodes.

When to use a ReplicaSet

A ReplicaSet ensures that a specified number of pod replicas are running at any given time. However, a Deployment is a higher-level concept that manages ReplicaSets and provides declarative updates to Pods along with a lot of other useful features. Therefore, we recommend using Deployments instead of directly using ReplicaSets, unless you require custom update orchestration or don't require updates at all.

This actually means that you may never need to manipulate ReplicaSet objects use a Deployment instead, and define your application in the spec section.

CLEANUP

It is very important to clean up unused resources that could be referencing other unneeded resources like images or secrets.