Kubernetes High Availability

Following HA techniques in k8s

- Restart Pods: The cluster will restart misbehaving pods according to its restart policy.
- Probing: Using Health Probes to try solving the issues.
- Horizontal Scaling: As per the load, we can scale the number of replicas.

Health Probes

Some pod is down, K8s will repeatedly probe it to find the problem

Taking care of:
- Crash Mitigation
- Failover
- Monitors a failing pod
- Scaling

Types of Probes:

- Readiness Probe
- Liveness Probe
- Startup Probe

Types of Probe Tests:

- HTTP GET
When this probe executes, it sends a request to this pod, if exit code is between 200 and 399, it is considered a success otherwise it has failed the test.

- Container Command
So the cluster runs a specific command in the container, if it exits with code 0, then it has succeed otherwise failed.

- TCP Socket
Cluster attempts to open a socket in the container, it it succeeds then fine otherwise it has failed the test.

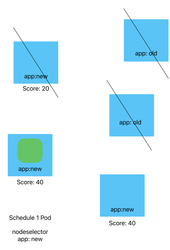
Kubernetes Pod Scheduling

Three step Process:

- Filtering Nodes
If we define a nodeselector, then the nodes that match that label are the only eligible ones.

- Prioritizing the Filtered Nodes
By using multiple criteria and its own algorithm, then the scheduler will determine a score for each node. Nodes with higher score are better for running the pod.

- Select the Best Fit Node
It will list them according to score.
Higher the score better the chances.
The highest score will be selected as the host of this pod.
If the score of two nodes match, then k8s implements round-robin style to allocate the node.



Compute Resource Requests

Minimum Resources to be allocated

Compute Resource Limits

Maximum Resources to be allocated

Millicore is a CPU Core
1 Core = 1000 Millicore
If I want to allocate 1 Core to an deployment = 1000m

Through CLI (Resource Request)

oc set resources deployment <name_of_deployment> --requests cpu=100m,memory=1G

Through YAML (Resource Limit)

Come to Spec tag and here we mention

```
resources:
    limits:
      cpu: "1000m"
      memory: "5G"
```

Kubernetes Auto Scaling

Resource:
Horizontal Pod Autoscaler (HPA)

This AutoScaling works in loops
Every 15 seconds (default) it will perform a set of steps:

1. The Autoscaler will retrieve details of the metrics from HPA Resource.
2. For each pod the HPA targets, metrics are collected for the OpenShift metric system.
3. For each pod, autoscaler will compute usage percentage, using collected metrics.
4. Autoscaler will then compute the average usage across all pods.
5. According to average it will taking scaling decisions.

CONTAINER IMAGES

docker.io/library/nginx:1.14.2

docker.io/library/nginx@SHA256:wvmpomcvopc2345oc

Image Pull Policy

1. IfNotPresent: (Default Policy)
If image is present in local storage, it will not pull the image, and if not present, it will pull the image.

2. Always:
OpenShift will verify whether an updated version of the image is available.

3. Never:
Do not pull the image, even if not present in local storage.
Deployment -- Will Fail

Pruning Images

Kubelet (our agent) has a **garbage collector** that runs every 5 mins.

File System carrying the Images, if the occupancy of that storage exceeds 85%, the garbage collector will start deleting images.

It will remove the oldest unused image first.

File System Usage drops to 80%

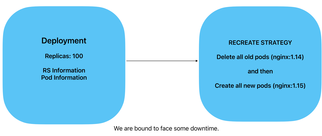
Deployment Strategies

Deployment ---- Replica Set + Pods (Replicas) + Rolling Updates + Rollback

Client

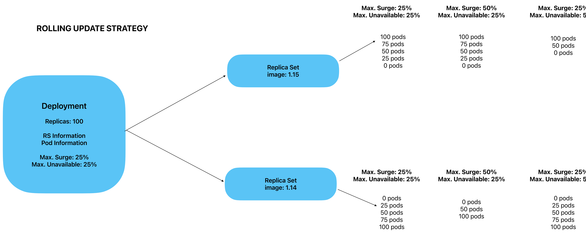
nginx:1.14
| Client wants Update
nginx:1.15

RECREATE STRATEGY



We are bound to face some downtime.

ROLLING UPDATE STRATEGY



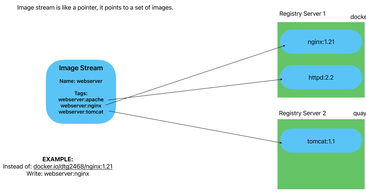
Rolling Updates

1. nginx:1.14 --> nginx:1.15 revision1
2. nginx:1.15 --> nginx:1.16 revision2
3. nginx:1.16 --> nginx:1.17 revision3

Rollback Updates
rollback - revision1

IMAGE STREAMS

Image stream is like a pointer, it points to a set of images.



EXAMPLE:
Instead of: docker.io/ltq2468/nginx:1.21
Write: webserver:nginx

BENEFITS

- Grouping Related Images
- Easier to remember names
- When an image changes, it will trigger an automatic new deployment