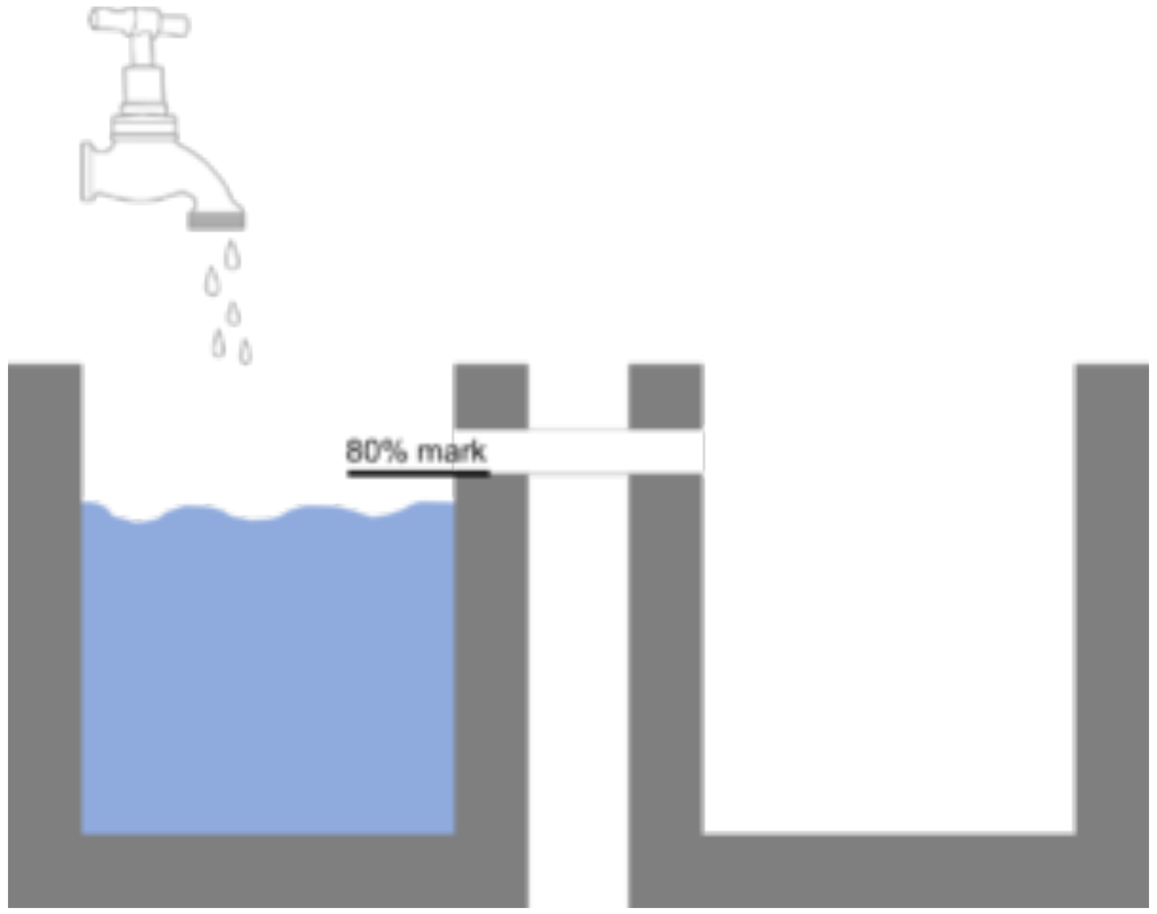
kubernetes

Kubernetes: Auto-Scaling

KUBERNETES : Advance of Kuebernetes

- **AutoScale:** Scaling the System to serve the requirement.
- Understand AutoScale: WaterTub Problem
- We want to ensure that once the first tub is **80% full**, another tub should be placed and the water should be sent to the second tub.



KUBERNETES : Advance of Kuebernetes

- Things to Understand by Above Problem.
- Tub - the unit of scaling (**What to scale?**)
- 80% mark - metric and trigger for scaling (**When to scale?**)
- Pipe - the operation which enables scaling in this case (**How to scale?**)

What to Scale?

- **Pods:** For a given application let's say you are running X replicas, In case of more Traffic Y number of Pods can also be executed on Cluster.
- For this our Nodes have enough resources available.
- **Nodes:** Capacity of all nodes put together represents your cluster's capacity. If the workload demand goes beyond this capacity, then you would have to add nodes to the cluster and make sure the workload can be scheduled and executed effectively.

When to Scale?

- By measuring a **certain metric** continuously and when the metric crosses a threshold value, then acting on it by scaling a certain resource.For this our Nodes have enough resources available.
- For example, you might want to **measure the average CPU consumption** of your pods and then trigger a scale operation if the CPU consumption crosses 80%.
- For memory intensive applications, memory consumption might be that metric.

How to Scale?

- CPU Based Scaling (HPA): Horizontal Pod Scaling
- With Horizontal Pod Autoscaling, Kubernetes **automatically scales the number of pods** in a replication controller, deployment or replica set based on observed CPU utilisation.
- Mostly this type of Scaling is based on CPU and Memory metrics.
- HPA is not applicable for **DaemonSets**.
- User can define the threshold and minimum and maximum scale to which the deployment should scale.
- Create HPA for Deployment:

```
kubectl autoscale deployment <deployment-name> --min=2 --max=5 --cpu-percent=80
```

How to Scale?

- Cluster AutoScaler
- Cluster autoscaler is used in Kubernetes to scale cluster i.e. nodes dynamically.
- CA watches the pods continuously and if it finds that a pod cannot be scheduled, then based on the PodCondition, it chooses to scale up.
- The cluster autoscaler requests a newly provisioned node if:
  There are pending pods due to not having enough available cluster resources to meet their requests and
  The cluster or node pool has not reached the user-defined maximum node count.
- Kubernetes detects the new node once it is provision by the underlying infrastructure.
- The Kubernetes scheduler allocates the pending pods to the new node.

KUBERNETES : Advance of Kuebernetes

- To enable Metrics on Cluster, Cluster has to be started with env variable: ENABLE_CUSTOM_METRICS=TRUE
- An Example:
- Run Deployment with CPU request 200m.
- 200m = 200 miliCPUs or 20% core of Running Node. If Node is 2 core then it's still 20% of Single Core.
- User can Introduce AutoScaling at 50% of CPU uses(Which is 100m/10% of Code with the Pod)

Will see you in Next Lecture…

See you in next lecture …