Kubernetes Scheduling

What is a Scheduler

A scheduler watches for newly created Pods that have no Node assigned.

For every Pod that the scheduler discovers, the scheduler becomes responsible for finding the best Node for that Pod to run on.

The scheduler reaches this placement decision taking into account the scheduling principles described

Kube-Scheduler

- kube-scheduler is the default scheduler for Kubernetes and runs as part of the control plane
- kube-scheduler is designed so that, if you want and need to, you can write your own scheduling component and use that instead.
- For every newly created pods or other unscheduled pods, kube-scheduler selects a optimal node for them to run on. However, every container in pods has different requirements for resources and every pod also has different requirements. Therefore, existing nodes need to be filtered according to the specific scheduling requirements.
- In a cluster, Nodes that meet the scheduling requirements for a Pod are called feasible nodes. If none of the nodes are suitable, the pod remains unscheduled until the scheduler is able to place it.
- The scheduler finds feasible Nodes for a Pod and then runs a set of functions to score the feasible Nodes and picks a Node with the highest score among the feasible ones to run the Pod.
- The scheduler then notifies the API server about this decision in a process called binding.
- Factors that need taken into account for scheduling decisions include individual and collective resource requirements, hardware / software / policy constraints, affinity and

Scheduling with kube-scheduler

kube-scheduler selects a node for the pod in a 2-step operation:

- Filtering
- Scoring

- The filtering step finds the set of Nodes where it’s feasible to schedule the Pod. For example, the PodFitsResources filter checks whether a candidate Node has enough available resource to meet a Pod’s specific resource requests. After this step, the node list contains any suitable Nodes; often, there will be more than one. If the list is empty, that Pod isn’t (yet) schedulable.

- In the scoring step, the scheduler ranks the remaining nodes to choose the most suitable Pod placement. The scheduler assigns a score to each Node that survived filtering, basing this score on the active scoring rules.Finally, kube-scheduler assigns the Pod to the Node with the highest ranking. If there is more than one node with equal scores, kube-scheduler selects one of these at random.

Default Policies

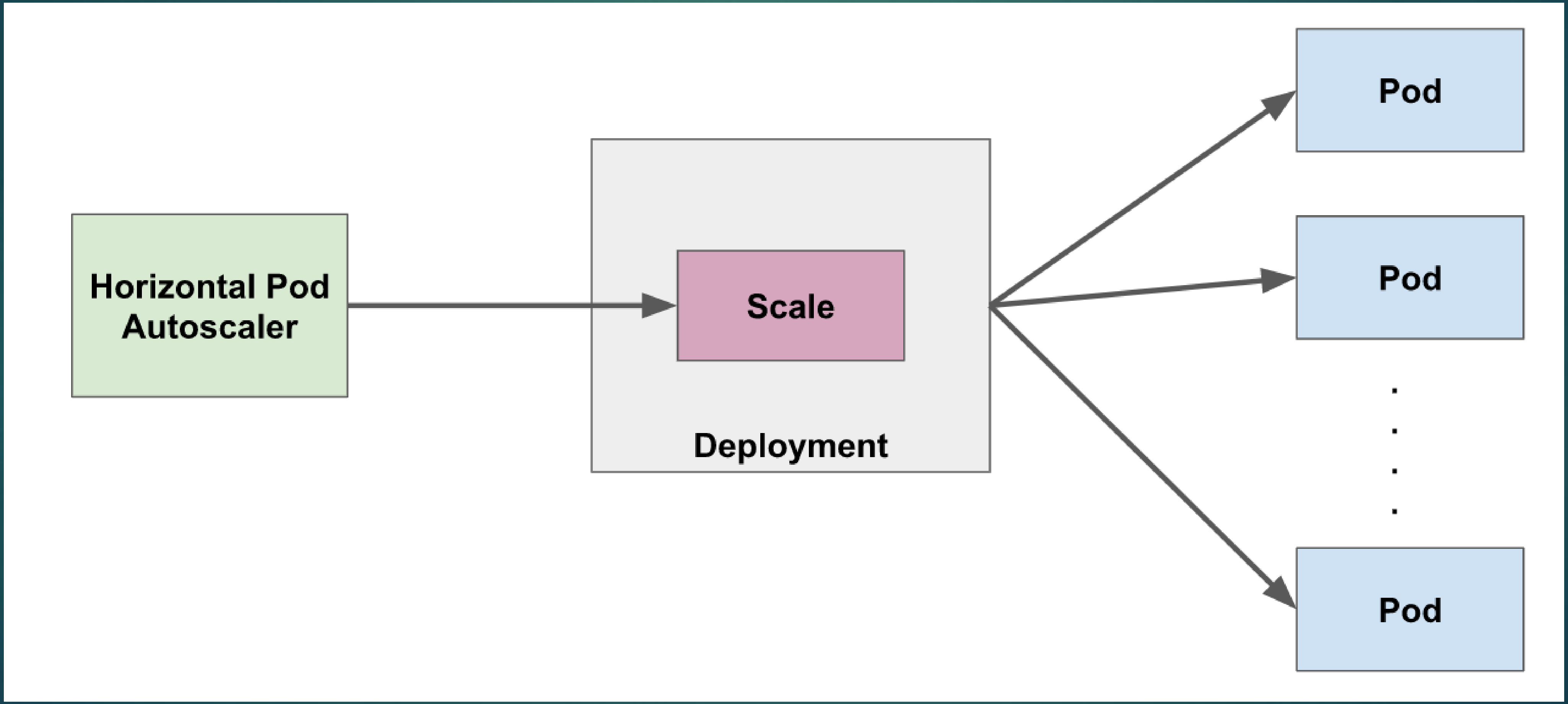
Filtering:

- PodFitsHostPorts: Checks if a Node has free ports (the network protocol kind) for the Pod ports the the Pod is requesting.
- PodFitsHost: Checks if a Pod specifies a specific Node by it hostname.
- PodFitsResources: Checks if the Node has free resources (eg, CPU and Memory) to meet the requirement of the Pod.
- PodMatchNodeSelector: Checks if a Pod's Node Selector matches the Node's label(s)

Scoring:

- SelectorSpreadPriority: Spreads Pods across hosts, considering Pods that belonging to the same Service , StatefulSet or ReplicaSet .
- InterPodAffinityPriority: Computes a sum by iterating through the elements of weightedPodAffinityTerm and adding "weight" to the sum if the corresponding PodAffinityTerm is satisfied for that node; the node(s) with the highest sum are the most preferred.
- LeastRequestedPriority: Favors nodes with fewer requested resources. In other words, the more Pods that are placed on a Node, and the more resources those Pods use, the lower

Pod AutoScaling



Horizontal Pod Autoscaling

- Horizontal Pod Autoscaling (HPA) can automatically scale the number of pods in a replication controller, deployment or replica set based CPU utilization
- It is implemented as a Kubernetes API resource and a controller
- Can also work with metrics

Affinity and Anti-Affinity

- Node affinity and Anti-affinity
- Node affinity is similar to Node Selector
- Antiaffinity allows rules using which pods can and cannot be co-located

Use Cases:

Pods should be always co-located and never be co-located on the same node

THANK YOU