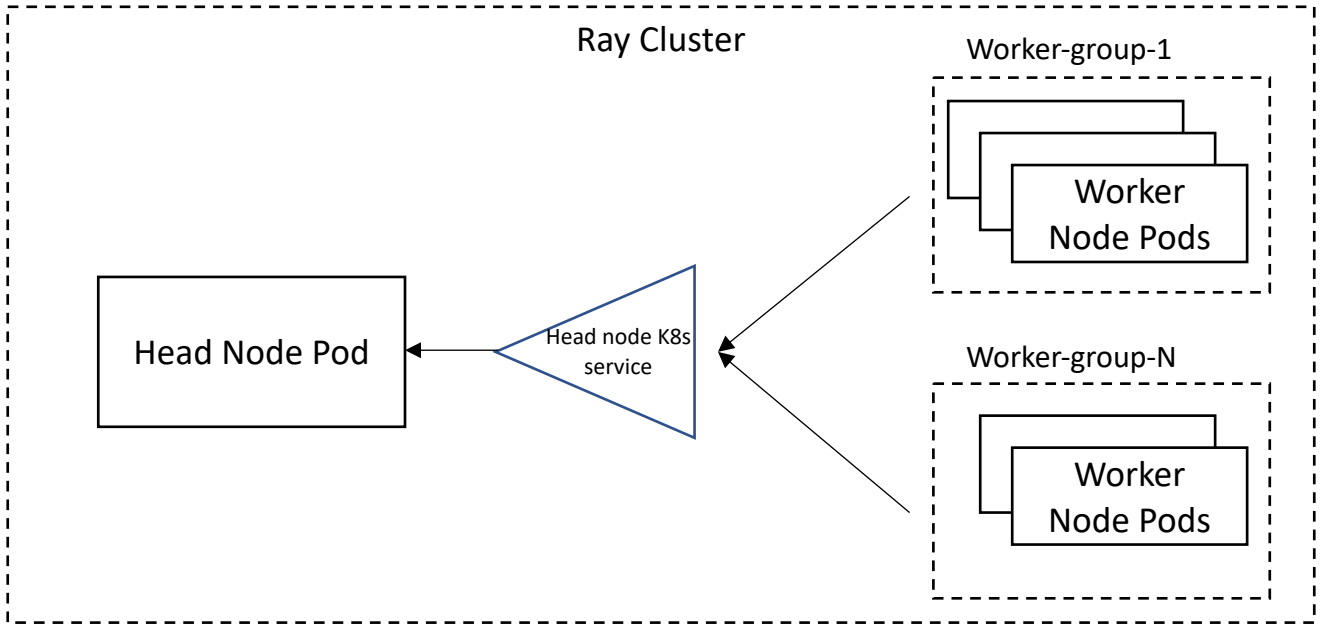
Microsoft AI

Ray AutoScaling Alternatives

Ali Kanso ✯, Edi Palencia✯, Edward Oakes, Eric Liang*, Yiran Wang**

✯ *Microsoft*

* *Anyscale*


Ray Cluster
Worker-group-1
Worker
Node Pods
Head Node Pod
Head node K8s
service
Worker-group-N
Worker
Node Pods

Autoscaling list of tasks:

The component(s) that is *managing* the lifecycle **and** *auto-scaling* a Ray cluster, must be capable of performing the following tasks:

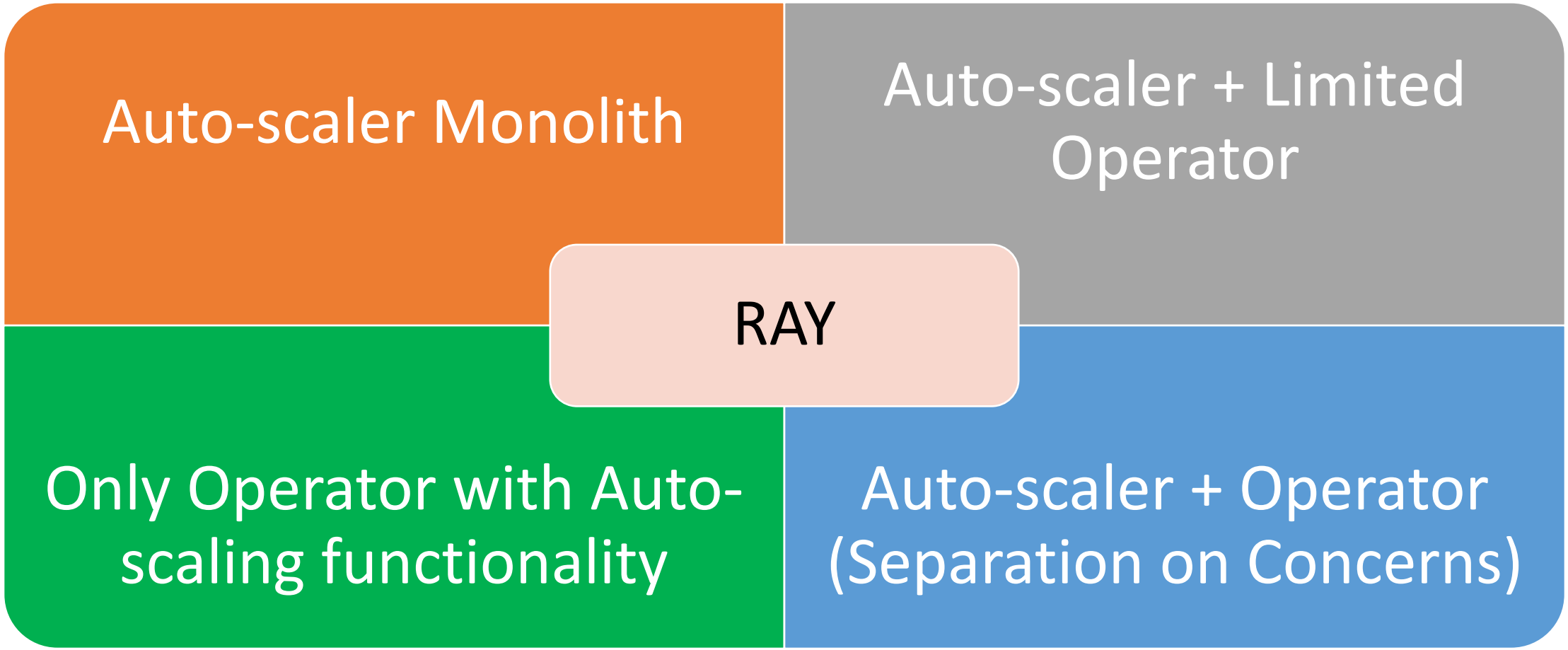
Life-Cycle Management

1. *Create a Ray cluster*
2. *Delete a Ray cluster and cleanup all the relevant resources*
3. *Failure recovery (e.g. if a physical machine fails, create the Ray nodes on another machine(s))*
4. Creating additional nodes and deleting idle nodes.

Auto-scaling

5. *Metrics collection: determine the CPU, Mem, GPU, TPU utilization in the cluster (currently there is only one threshold supported). And determine the idle nodes.*
6. *Making a scaling up/down **decision** based on the metrics collected*

Discussed Options

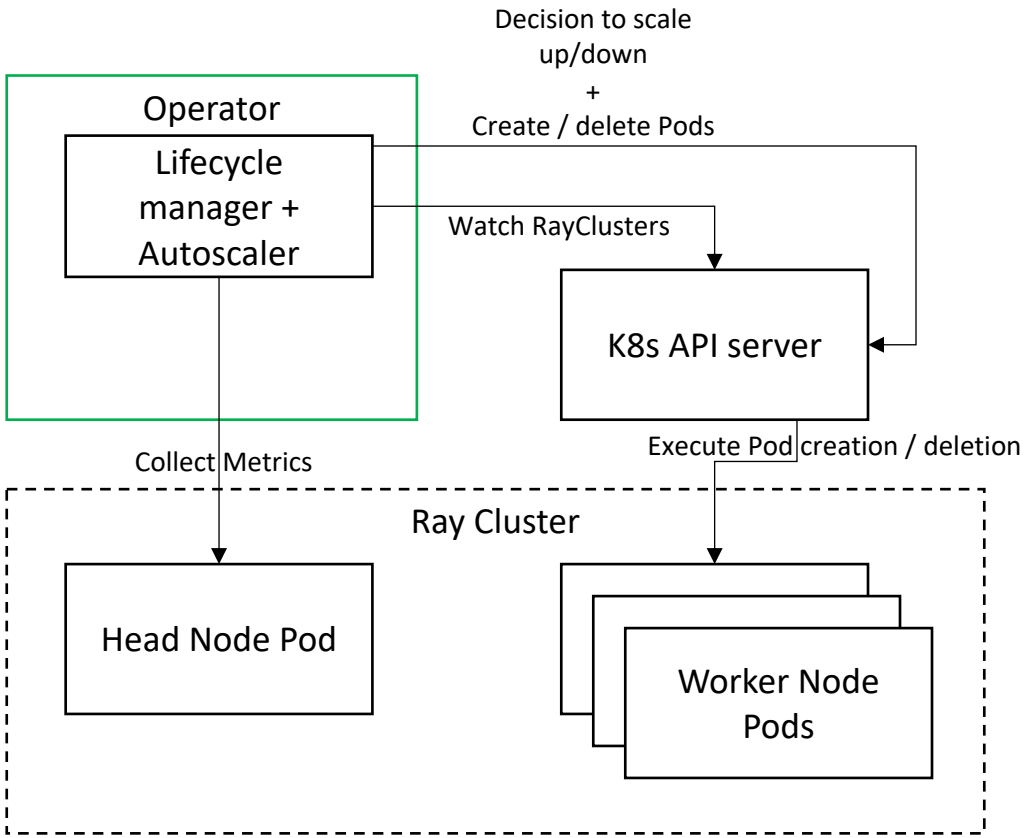


Auto-scaler monolith:

Currently, the auto-scaler in Ray , can carry out all the above tasks (1 → 6).

Pros	Cons
Has access to internal Ray-nodes information	Mixes concerns for the auto-scaler (metrics collection, decision making, autoscaling by creating/deleting nodes)
Can select which idle nodes to remove	Has excessive permissions on the Ray cluster and the K8s cluster (no RBAC in place today)
Is not limited to K8s, supports AWS, GCP, Azure, etc.	More complicated to maintain and modify
	If a k8s Pod fails, the auto-scaler might not be fast as an operator to recover the failed pod.

Only Operator with autoscaling functionality



Auto-scaler + Limited Operator:

Let the limited-Operator only take care of creating and managing the head node of the ray cluster, and delegate the task of **managing the worker nodes to the auto-scaler.**

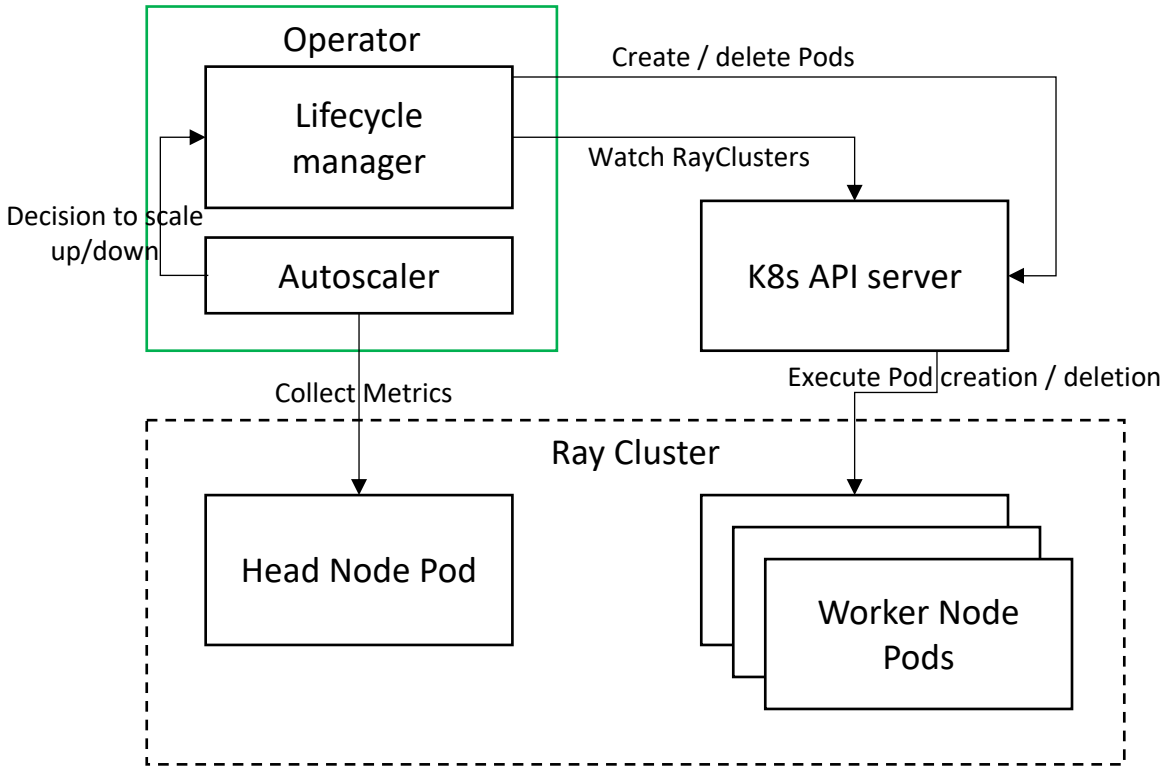
Pros	Cons
Ray head node availability is improved	Mixes concerns for the auto-scaler (metrics collection, decision making, autoscaling by creating/deleting worker nodes)
	Added complexity for adding the Operator for a trivial task of managing the head node of the ray cluster
	In addition to the concerns about excessive permissions, etc. mentioned before

Only Operator with autoscaling functionality

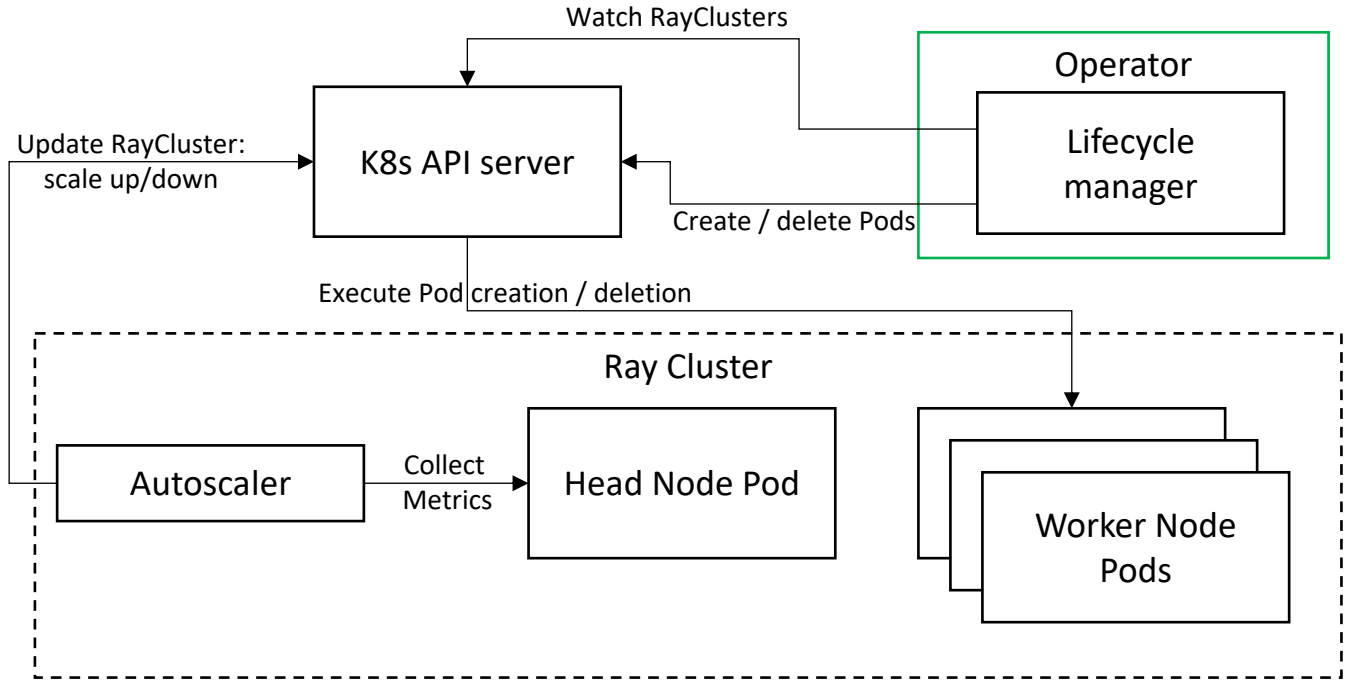
The Operator will be **responsible for tasks 1 to 6 described** above. Basically, replacing the auto-scaler logic we have today.

Pros	Cons
Ray nodes availability is improved	Requires the existence of a K8s API server. i.e. it is tied down to K8s*
Better user experience. The user only deals with K8s Yaml manifest for the cluster configuration as K8s Custom resource	Mixes concerns for the Operator (metrics collection, decision making, autoscaling by creating/deleting worker nodes)
Added security by managing through RBAC who has permissions to create/modify/delete Ray clusters on the K8s cluster	Requires a re-write of the current auto-scaler and a re-write of the current Ray Operator
Ability to enforce admission control on the Ray cluster configuration. E.g. reject configuration that are not semantically valid. Using the OpenAPI V3 validation mechanisms we can also enforce syntactical validation with minimum effort.	
Ability to enforce resource quota in the namespace of the Ray cluster.	
Leverages the ubiquitous availability of K8s as a service offering of cloud vendors .	

Only Operator with autoscaling functionality



Auto-scaler + Operator

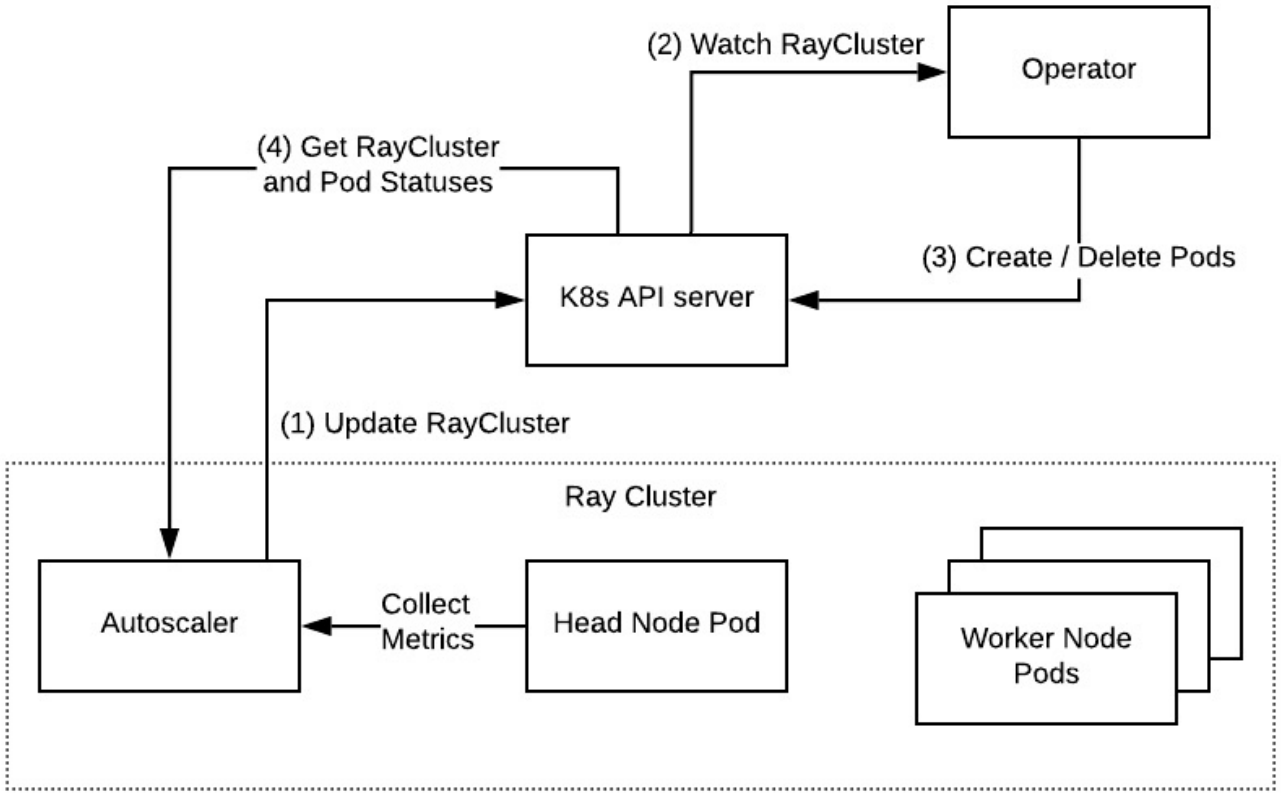


Auto-scaler + Operator

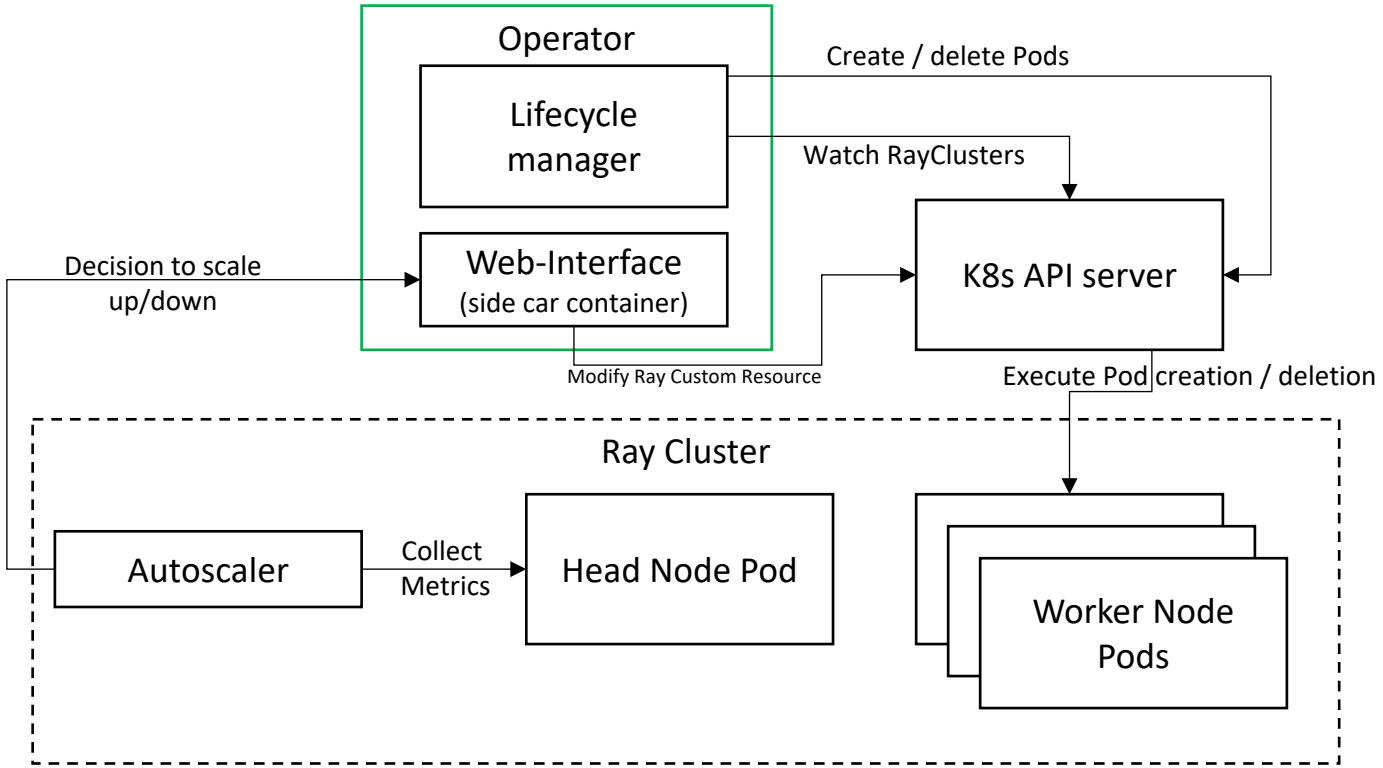
The auto-scaler makes the auto-scaling decisions (tasks 5 and 6) and the Operator takes care of the life-cycle management of the Ray cluster nodes (tasks 1, 2, 3, 4).

Pros	Cons
Ray nodes availability is improved	Requires the existence of a K8s API server. i.e. it is tied down to K8s*
Separation of concerns for the auto-scaler (metrics collection, decision making,) and the Operator manages the lifecycle of the ray nodes)	Requires the auto-scaler to convey the decisions to the Operator. E.g. by creating an auto-scaling request in K8s, or modifying the Custom Resource
Better user experience. The user only deals with K8s Yaml manifest for the cluster configuration as K8s Custom resource	
Added security by managing through RBAC who has permissions to create/modify/delete Ray clusters on the K8s cluster	
Ability to enforce admission control on the Ray cluster configuration. E.g. reject configuration that are not semantically valid. Using the OpenAPI V3 validation mechanisms we can also enforce syntactical validation with minimum effort.	
Ability to enforce resource quota in the namespace of the Ray cluster.	
Leverages the ubiquitous availability of K8s as a service offering of cloud vendors.	

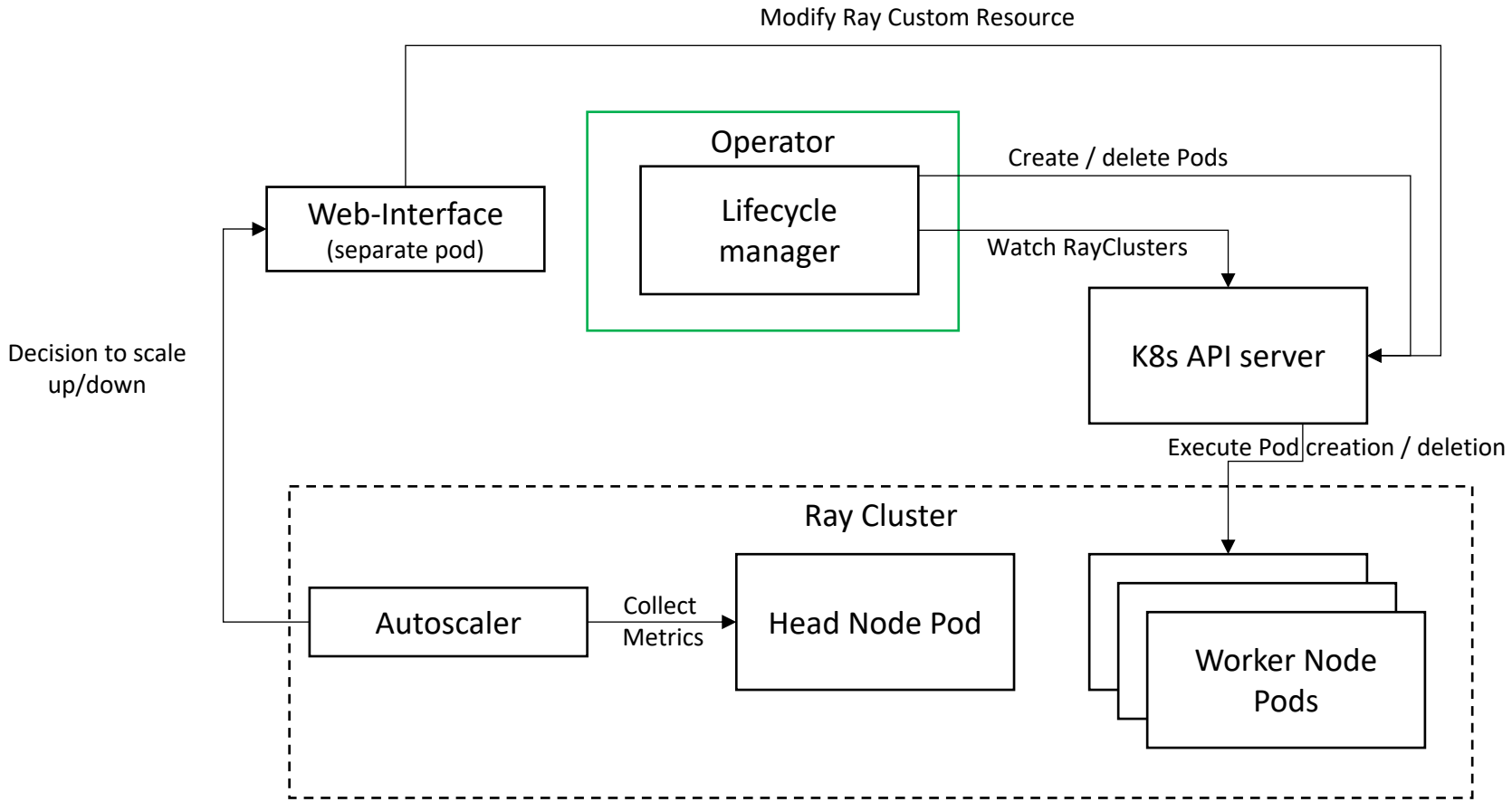
Auto-scaler + Operator: Option 1



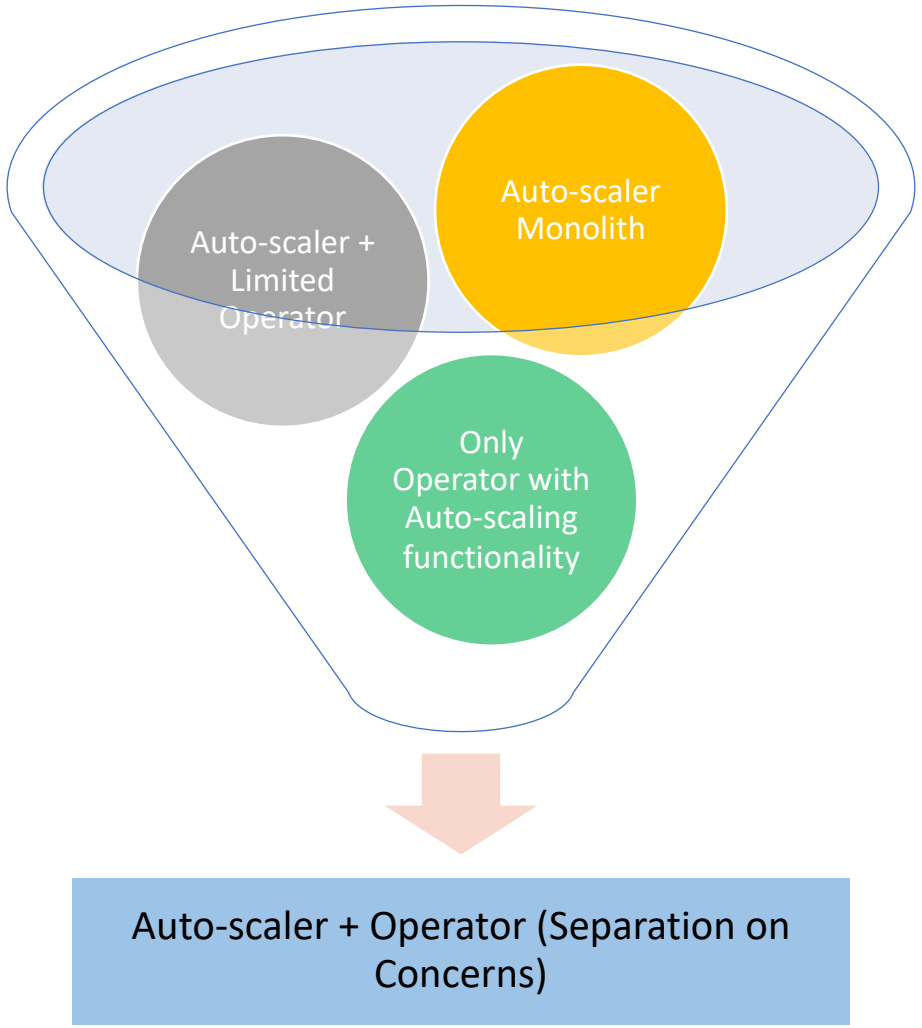
Auto-scaler + Operator + WebServer: Option 2



Auto-scaler + Operator + WebServer : Option 3



Conclusion: Auto-scaler + Operator (Separation on Concerns)



(Ray users not running on K8s clusters, can substitute the Operator with Ray Cluster-Launcher)