

Robert Zacchia

# Task 04 - Optimization-Based custom Scheduler for Minikube Deployment

---

## Optimization Algorithm

To optimize the resource scheduling I used a Genetic Algorithm which can be found in [main.py](#).

## Scheduling

For the python script to be able to control the cluster resources, it needs a [deployment](#) and a [clusterRole](#) to work properly. The clusterRole describes, which api resources of the cluster the deployment has access to (e.g. read pods).

I chose to simulate the latency score by giving labels to the different nodes, which I then read out as latency to use in my genetic algorithm. I did that to check if the scheduling starts with the correct and to make things a bit easier.

## Step by Step

```
# start minikube
minikube start -p task04 --nodes=3
kubectl config use-context task04

# check nodes are here
kubectl get nodes -o wide

# artificial response time labels
kubectl label node task04      latency-score=5 --overwrite
kubectl label node task04-m02   latency-score=1 --overwrite
kubectl label node task04-m03   latency-score=3 --overwrite

# load scheduler into cluster
cd scheduler
docker build -t meta-scheduler:latest .
cd ..
minikube -p task04 image load meta-scheduler:latest

# apply scheduler to pod
kubectl apply -f k8s/namespace.yaml
kubectl apply -f k8s/scheduler-rbac.yaml
kubectl apply -f k8s/scheduler-deployment.yaml

# check scheduler pod
# kubectl -n task04 get pods
# kubectl -n task04 logs deploy/meta-scheduler

# deploy mongodb + mongo express
```

```
kubectl apply -f k8s/mongo.yaml
kubectl apply -f k8s/mongo-express.yaml

kubectl -n task04 port-forward svc/mongo-express 9090:8081

# kubectl -n task04 rollout restart deploy/meta-scheduler

# check which pod uses which scheduler
kubectl -n task04 get pods -o custom-
columns=NAME:.metadata.name,SCHED:.spec.schedulerName
```