

K8S Report

1. The screenshot of `kubectl get nodes`

```
• student@ds-student11:~/NTHU-Scheduler-Plugin$ kubectl get nodes
NAME                STATUS    ROLES    AGE   VERSION
kind-control-plane  Ready    control-plane  50d   v1.29.2
kind-worker          Ready    <none>      50d   v1.29.2
kind-worker2         Ready    <none>      50d   v1.29.2
○ student@ds-student11:~/NTHU-Scheduler-Plugin$
```

2. The screenshot of passing all the unit tests

```
root@7274bb316515:/go/src/app# go test -v ./...
?      my-scheduler-plugins/cmd/scheduler      [no test files]
=== RUN   TestCustomScheduler_Prefilter
=== RUN   TestCustomScheduler_Prefilter/pod_is_accepted
finish adding2024/06/15 19:02:06 Pod  is in Prefilter phase.
=== RUN   TestCustomScheduler_Prefilter/pod_is_just_accepted
finish adding2024/06/15 19:02:06 Pod  is in Prefilter phase.
=== RUN   TestCustomScheduler_Prefilter/pod_is_rejected
finish adding2024/06/15 19:02:06 Pod  is in Prefilter phase.
--- PASS: TestCustomScheduler_Prefilter (0.00s)
    --- PASS: TestCustomScheduler_Prefilter/pod_is_accepted (0.00s)
    --- PASS: TestCustomScheduler_Prefilter/pod_is_just_accepted (0.00s)
    --- PASS: TestCustomScheduler_Prefilter/pod_is_rejected (0.00s)
=== RUN   TestCustomScheduler_Score
=== RUN   TestCustomScheduler_Score/least_mode
2024/06/15 19:02:06 Custom scheduler runs with the mode: Least.
2024/06/15 19:02:06 Pod  is in Score phase. Calculate the score of Node m1.
2024/06/15 19:02:06 Pod  is in Score phase. Calculate the score of Node m2.
=== RUN   TestCustomScheduler_Score/most_mode
2024/06/15 19:02:06 Custom scheduler runs with the mode: Least.
2024/06/15 19:02:06 Pod  is in Score phase. Calculate the score of Node m1.
2024/06/15 19:02:06 Pod  is in Score phase. Calculate the score of Node m2.
--- PASS: TestCustomScheduler_Score (0.00s)
    --- PASS: TestCustomScheduler_Score/least_mode (0.00s)
    --- PASS: TestCustomScheduler_Score/most_mode (0.00s)
=== RUN   TestCustomScheduler_NormalizeScore
=== RUN   TestCustomScheduler_NormalizeScore/scores_in_range
=== RUN   TestCustomScheduler_NormalizeScore/scores_out_of_range
=== RUN   TestCustomScheduler_NormalizeScore/negative_score
--- PASS: TestCustomScheduler_NormalizeScore (0.00s)
    --- PASS: TestCustomScheduler_NormalizeScore/scores_in_range (0.00s)
    --- PASS: TestCustomScheduler_NormalizeScore/scores_out_of_range (0.00s)
    --- PASS: TestCustomScheduler_NormalizeScore/negative_score (0.00s)
PASS
ok      my-scheduler-plugins/pkg/plugins      0.031s
root@7274bb316515:/go/src/app#
```

3. Explain the 3 scenarios you design to validate your implementation

1. Validating the prefilter

```

I0619 15:36:43.433640      1 log.go:194] Pod nginx1 is in Prefilter phase.
I0619 15:36:43.434689      1 log.go:194] Pod nginx1 is in Score phase. Calculate the score of Node kind-worker.
I0619 15:36:43.434728      1 log.go:194] Pod nginx1 is in Score phase. Calculate the score of Node kind-worker2.
I0619 15:36:43.442633      1 schedule_one.go:252] "Successfully bound pod to node" pod="default/nginx1" node="kind-worker" evaluatedNodes=3 feasibleNodes=2
I0619 15:36:49.960469      1 log.go:194] Pod nginx2 is in Prefilter phase.
I0619 15:36:49.961285      1 schedule_one.go:867] "Unable to schedule pod; no fit; waiting" pod="default/nginx2" err="0/3 nodes are available: Not enough po
ds in group A, minimum required is 3, preemption: 0/3 nodes are available: 3 No preemption victims found for incoming pod.."
I0619 15:36:52.668782      1 log.go:194] Pod nginx3 is in Prefilter phase.
I0619 15:36:52.669118      1 log.go:194] Pod nginx3 is in Score phase. Calculate the score of Node kind-worker.
I0619 15:36:52.669138      1 log.go:194] Pod nginx3 is in Score phase. Calculate the score of Node kind-worker2.
I0619 15:36:52.674478      1 schedule_one.go:252] "Successfully bound pod to node" pod="default/nginx3" node="kind-worker" evaluatedNodes=3 feasibleNodes=2
I0619 15:37:01.619214      1 log.go:194] Pod nginx4 is in Prefilter phase.
I0619 15:37:01.619795      1 log.go:194] Pod nginx4 is in Score phase. Calculate the score of Node kind-worker2.
I0619 15:37:01.619832      1 log.go:194] Pod nginx4 is in Score phase. Calculate the score of Node kind-worker.
I0619 15:37:01.625442      1 schedule_one.go:252] "Successfully bound pod to node" pod="default/nginx4" node="kind-worker" evaluatedNodes=3 feasibleNodes=2
I0619 15:42:10.631237      1 log.go:194] Pod nginx2 is in Prefilter phase.
I0619 15:42:10.632094      1 log.go:194] Pod nginx2 is in Score phase. Calculate the score of Node kind-worker2.
I0619 15:42:10.632128      1 log.go:194] Pod nginx2 is in Score phase. Calculate the score of Node kind-worker.
I0619 15:42:10.637505      1 schedule_one.go:252] "Successfully bound pod to node" pod="default/nginx2" node="kind-worker2" evaluatedNodes=3 feasibleNodes=2

```

- 我 declare 了 nginx1-4, 並將 nginx2 的 minavail 設成 3, 所以一開始 nginx2 進來時會被 prefilter block 住, 但是當 nginx1-4 都進來後 nginx2 則可以過去

2. Validating the Least Mode

```

I0619 15:52:41.174765      1 log.go:194] Pod nginx7 is in Prefilter phase.
I0619 15:52:41.175724      1 log.go:194] Pod nginx7 is in Score phase. Calculate the score of Node kind-worker.
I0619 15:52:41.175758      1 log.go:194] Pod nginx7 is in Score phase. Calculate the score of Node kind-worker2.
I0619 15:52:41.180979      1 schedule_one.go:252] "Successfully bound pod to node" pod="default/nginx7" node="kind-worker" evaluatedNodes=3 feasibleNodes=2

```

- 我另外 declare 了 nginx5-7, 他們則屬於另一個 group, 可以看到在 least mode 時 nginx7 會優先被放到 `kind-worker`, 利用 `kubect1 describe nodes` 可以看到 `kind-worker` 的 memory 比 `kind-worker2` 的 memory 還多, 所以可以證明 Least mode 的正確性

3. Validating the Most Mode

```

I0619 16:04:20.123431      1 log.go:194] Pod nginx8 is in Prefilter phase.
I0619 16:04:20.126220      1 log.go:194] Pod nginx8 is in Score phase. Calculate the score of Node kind-worker.
I0619 16:04:20.126260      1 log.go:194] Pod nginx8 is in Score phase. Calculate the score of Node kind-worker2.
I0619 16:04:20.140783      1 schedule_one.go:252] "Successfully bound pod to node" pod="default/nginx8" node="kind-worker2" evaluatedNodes=3 feasibleNodes=2

```

- 將 Mode 改成 Most mode 後, 再 deploy 一個 node 上去(內容跟example一樣), 可以看到這次被優先放到了 `kind-worker2`, 因此可以證明 Most mode 的正確性