1. The screenshot of kubectl get nodes

```
→ NTHU-Scheduler-Plugin git:(main) X kubectl get nodes
NAME                 STATUS   ROLES           AGE   VERSION
kind-control-plane   Ready    control-plane   13d   v1.29.2
kind-worker          Ready    <none>          13d   v1.29.2
kind-worker2         Ready    <none>          13d   v1.29.2
→ NTHU-Scheduler-Plugin git:(main) X
```

2. The screenshot of passing all the unit tests

```
Running tool: /usr/local/go/bin/go test -timeout 30s -run ^(TestCustomScheduler_PreFilter|TestCu
stomScheduler_Score|TestCustomScheduler_NormalizeScore)$ my-scheduler-plugins/pkg/plugins

=== RUN   TestCustomScheduler_PreFilter
=== RUN   TestCustomScheduler_PreFilter/pod_is_accepted
finish adding2024/06/11 23:39:03 Pod  is in Prefilter phase.
2024/06/11 23:39:03 Pod label: g1
2024/06/11 23:39:03 Pods len: 3
--- PASS: TestCustomScheduler_PreFilter/pod_is_accepted (0.00s)
=== RUN   TestCustomScheduler_PreFilter/pod_is_just_accepted
finish adding2024/06/11 23:39:03 Pod  is in Prefilter phase.
2024/06/11 23:39:03 Pod label: g1
2024/06/11 23:39:03 Pods len: 3
--- PASS: TestCustomScheduler_PreFilter/pod_is_just_accepted (0.00s)
=== RUN   TestCustomScheduler_PreFilter/pod_is_rejected
finish adding2024/06/11 23:39:03 Pod  is in Prefilter phase.
2024/06/11 23:39:03 Pod label: g1
2024/06/11 23:39:03 Pods len: 3
--- PASS: TestCustomScheduler_PreFilter/pod_is_rejected (0.00s)
--- PASS: TestCustomScheduler_PreFilter (0.00s)
=== RUN   TestCustomScheduler_Score
=== RUN   TestCustomScheduler_Score/least_mode
2024/06/11 23:39:03 Custom scheduler runs with the mode: Least.
2024/06/11 23:39:03 Pod  is in Score phase. Calculate the score of Node m1.
2024/06/11 23:39:03 Node m1 allocatable memory: 100, requested memory: 0, memory: 100
2024/06/11 23:39:03 Pod  is in Score phase. Calculate the score of Node m2.
2024/06/11 23:39:03 Node m2 allocatable memory: 200, requested memory: 0, memory: 200
--- PASS: TestCustomScheduler_Score/least_mode (0.00s)
```

```
--- PASS: TestCustomScheduler_Score/least_mode (0.00s)
=== RUN   TestCustomScheduler_Score/most_mode
2024/06/11 23:39:03 Custom scheduler runs with the mode: Least.
2024/06/11 23:39:03 Pod  is in Score phase. Calculate the score of Node m1.
2024/06/11 23:39:03 Node m1 allocatable memory: 100, requested memory: 0, memory: 100
2024/06/11 23:39:03 Pod  is in Score phase. Calculate the score of Node m2.
2024/06/11 23:39:03 Node m2 allocatable memory: 200, requested memory: 0, memory: 200
--- PASS: TestCustomScheduler_Score/most_mode (0.00s)
--- PASS: TestCustomScheduler_Score (0.00s)
=== RUN   TestCustomScheduler_NormalizeScore
=== RUN   TestCustomScheduler_NormalizeScore/scores_in_range
2024/06/11 23:39:03 Pod . Node m1's socre 1
2024/06/11 23:39:03 Pod . Node m2's socre 2
2024/06/11 23:39:03 Pod . Node m3's socre 3
--- PASS: TestCustomScheduler_NormalizeScore/scores_in_range (0.00s)
=== RUN   TestCustomScheduler_NormalizeScore/scores_out_of_range
2024/06/11 23:39:03 Pod . Node m1's socre 1000
2024/06/11 23:39:03 Pod . Node m2's socre 2000
2024/06/11 23:39:03 Pod . Node m3's socre 3000
--- PASS: TestCustomScheduler_NormalizeScore/scores_out_of_range (0.00s)
=== RUN   TestCustomScheduler_NormalizeScore/negative_score
2024/06/11 23:39:03 Pod . Node m1's socre -1000
2024/06/11 23:39:03 Pod . Node m2's socre -2000
2024/06/11 23:39:03 Pod . Node m3's socre -3000
--- PASS: TestCustomScheduler_NormalizeScore/negative_score (0.00s)
--- PASS: TestCustomScheduler_NormalizeScore (0.00s)
PASS
ok      my-scheduler-plugins/pkg/plugins        0.029s
```

3. Explain the 3 scenarios you design to validate your implementation by describing the expected results and showing the screenshots of the results.

Prefilter:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    podGroup: "A"
    minAvailable: "3"
spec:
  schedulerName: my-scheduler
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
    resources:
      requests:
        memory: "100Mi"
      limits:
        memory: "100Mi"

---
```

Spec:

Yaml檔內容是五個spec一模一樣的pod，其labels中的podGroup均為A，且

minAvailable為3，且scheduler使用的mode為Least。

理論結果:

應該要是後三個pod立即被schedule到較少memory的那個worker，且前兩個pod在

一段時間後(k8s重跑該pod的schedule流程)也會被schedule到較少memory的那個

worker。

實驗結果:

```
→ NTHU-Scheduler-Plugin git:(main) X make testPrefilter
kubectl create -f test/prefilter.yaml
pod/nginx created
pod/nginx1 created
pod/nginx2 created
pod/nginx3 created
pod/nginx4 created
kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE           NOMINATED NODE   READINESS GATES
my-scheduler-69cfc986c7-7g498       1/1     Running   0          13m   10.244.1.5    kind-worker2   <none>           <none>
nginx                               0/1     Pending   0          3s    <none>        <none>         <none>           <none>
nginx1                              0/1     Pending   0          2s    <none>        <none>         <none>           <none>
nginx2                              1/1     Running   0          2s    10.244.2.11   kind-worker    <none>           <none>
nginx3                              1/1     Running   0          2s    10.244.2.12   kind-worker    <none>           <none>
nginx4                              1/1     Running   0          2s    10.244.2.13   kind-worker    <none>           <none>
```

```
I0611 23:51:20.975608       1 log.go:194] Pod nginx is in Prefilter phase.
I0611 23:51:20.975662       1 log.go:194] Pod label: A
I0611 23:51:20.975795       1 log.go:194] Pods len: 1
I0611 23:51:21.030557       1 log.go:194] Pod nginx1 is in Prefilter phase.
I0611 23:51:21.030587       1 log.go:194] Pod label: A
I0611 23:51:21.030635       1 log.go:194] Pods len: 2
I0611 23:51:21.077217       1 log.go:194] Pod nginx2 is in Prefilter phase.
I0611 23:51:21.077396       1 log.go:194] Pod label: A
I0611 23:51:21.077506       1 log.go:194] Pods len: 3
I0611 23:51:21.078574       1 log.go:194] Pod nginx2 is in Score phase. Calculate the score of Node kind-worker.
I0611 23:51:21.078598       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 52428800, memory: 1324290048
I0611 23:51:21.078612       1 log.go:194] Pod nginx2 is in Score phase. Calculate the score of Node kind-worker2.
I0611 23:51:21.078637       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 52428800, memory: 1861160960
I0611 23:51:21.078901       1 log.go:194] Pod nginx2. Node kind-worker's socre -1324290048
I0611 23:51:21.078912       1 log.go:194] Pod nginx2. Node kind-worker2's socre -1861160960
I0611 23:51:21.129388       1 log.go:194] Pod nginx3 is in Prefilter phase.
I0611 23:51:21.129415       1 log.go:194] Pod label: A
I0611 23:51:21.129453       1 log.go:194] Pods len: 4
I0611 23:51:21.129968       1 log.go:194] Pod nginx3 is in Score phase. Calculate the score of Node kind-worker2.
I0611 23:51:21.130009       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 52428800, memory: 1861160960
I0611 23:51:21.130042       1 log.go:194] Pod nginx3 is in Score phase. Calculate the score of Node kind-worker.
I0611 23:51:21.130062       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 157286400, memory: 1219432448
I0611 23:51:21.130198       1 log.go:194] Pod nginx3. Node kind-worker2's socre -1861160960
I0611 23:51:21.130219       1 log.go:194] Pod nginx3. Node kind-worker's socre -1219432448
I0611 23:51:21.184711       1 log.go:194] Pod nginx4 is in Prefilter phase.
I0611 23:51:21.185035       1 log.go:194] Pod label: A
I0611 23:51:21.185111       1 log.go:194] Pods len: 5
I0611 23:51:21.185357       1 log.go:194] Pod nginx4 is in Score phase. Calculate the score of Node kind-worker.
I0611 23:51:21.185364       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 262144000, memory: 1114574848
I0611 23:51:21.185371       1 log.go:194] Pod nginx4 is in Score phase. Calculate the score of Node kind-worker2.
I0611 23:51:21.185375       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 52428800, memory: 1861160960
I0611 23:51:21.185423       1 log.go:194] Pod nginx4. Node kind-worker's socre -1114574848
I0611 23:51:21.185428       1 log.go:194] Pod nginx4. Node kind-worker2's socre -1861160960
```

可以從scheduler的log與pod的資訊上看到，pod確實是被schedule給較少memory

的那個worker(kind-worker)，與理論結果一致。

```
→ NTHU-Scheduler-Plugin git:(main) X kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE     IP            NODE           NOMINATED NODE   READINESS GATES
my-scheduler-69cfc986c7-7g498       1/1     Running   0          18m     10.244.1.5    kind-worker2   <none>           <none>
nginx                               1/1     Running   0          5m33s   10.244.2.14   kind-worker    <none>           <none>
nginx1                              1/1     Running   0          5m32s   10.244.2.15   kind-worker    <none>           <none>
nginx2                              1/1     Running   0          5m32s   10.244.2.11   kind-worker    <none>           <none>
nginx3                              1/1     Running   0          5m32s   10.244.2.12   kind-worker    <none>           <none>
nginx4                              1/1     Running   0          5m32s   10.244.2.13   kind-worker    <none>           <none>
→ NTHU-Scheduler-Plugin git:(main) X
```

約過了5分30秒後，前兩個pod也被schedule到較少memory的那個worker(kind-

worker) ，與理論結果一致。

Least Mode:

Spec:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    podGroup: "A"
    minAvailable: "1"
spec:
  schedulerName: my-scheduler
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
    resources:
      requests:
        memory: "1000Mi"
      limits:
        memory: "1000Mi"

---
```

Yaml檔內容中有兩個pod，其labels中的podGroup均為A，且minAvailable為1，第一個pod的memory為1000Mi，第2個pod的memory為100Mi，且scheduler使用的mode為Least。

理論結果:

應該要是這兩個pod立即被schedule到較少memory的那個worker。

實驗結果:

```
→ NTHU-Scheduler-Plugin git:(main) X make testLeastMode
kubectl create -f test/least_mode.yaml
pod/nginx created
pod/nginx1 created
kubectl get po -o wide
NAME                            READY   STATUS    RESTARTS   AGE   IP            NODE           NOMINATED NODE   READINESS GATES
my-scheduler-69cfc986c7-v2vd8   1/1     Running   0          8s    10.244.1.6    kind-worker2   <none>           <none>
nginx                           1/1     Running   0          2s    10.244.2.16   kind-worker    <none>           <none>
nginx1                          1/1     Running   0          2s    10.244.2.17   kind-worker    <none>           <none>
```

```
I0611 23:58:58.332870       1 log.go:194] Pod nginx is in Prefilter phase.
I0611 23:58:58.332912       1 log.go:194] Pod label: A
I0611 23:58:58.333008       1 log.go:194] Pods len: 1
I0611 23:58:58.334276       1 log.go:194] Pod nginx is in Score phase. Calculate the score of Node kind-worker2.
I0611 23:58:58.334306       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 52428800, memory: 1861160960
I0611 23:58:58.334338       1 log.go:194] Pod nginx is in Score phase. Calculate the score of Node kind-worker.
I0611 23:58:58.334350       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 52428800, memory: 1324290048
I0611 23:58:58.334430       1 log.go:194] Pod nginx. Node kind-worker's socre -1324290048
I0611 23:58:58.334443       1 log.go:194] Pod nginx. Node kind-worker2's socre -1861160960
I0611 23:58:58.385748       1 log.go:194] Pod nginx1 is in Prefilter phase.
I0611 23:58:58.386016       1 log.go:194] Pod label: A
I0611 23:58:58.386053       1 log.go:194] Pods len: 2
I0611 23:58:58.386378       1 log.go:194] Pod nginx1 is in Score phase. Calculate the score of Node kind-worker2.
I0611 23:58:58.386393       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 52428800, memory: 1861160960
I0611 23:58:58.386403       1 log.go:194] Pod nginx1 is in Score phase. Calculate the score of Node kind-worker.
I0611 23:58:58.386408       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 1101004800, memory: 275714048
I0611 23:58:58.386441       1 log.go:194] Pod nginx1. Node kind-worker2's socre -1861160960
I0611 23:58:58.386450       1 log.go:194] Pod nginx1. Node kind-worker's socre -275714048
```

可以從scheduler的log與pod的資訊上看到，pod確實是被schedule給較少memory的那個worker(kind-worker)，與理論結果一致。

Most Mode:

Spec:

這個spec與least mode的spec一模一樣，Yaml檔內容中有兩個pod，其labels中的podGroup均為A，且minAvailable為1，第一個pod的memory為1000Mi，第2個pod的memory為100Mi，且scheduler使用的mode為Most。

理論結果:

第一個pod應該要被schedule在當下memory最多的node上(也就是kind-worker2)，第二個pod也是同樣的道理，但此時最多memory的node會改變成kind-worker，因為第一個pod request 1000Mi的memory，讓kind-worker2的memory從4.5Gi降到3.5Gi，小於此時kind-worker的4Gi，所以第二個pod應該要被schedule在kind-worker。

實驗結果:

```
→ NTHU-Scheduler-Plugin git:(main) X make testMostMode
kubectl create -f test/most_mode.yaml
pod/nginx created
pod/nginx1 created
kubectl get po -o wide
NAME                                READY   STATUS    RESTARTS   AGE   IP            NODE           NOMINATED NODE   READINESS GATES
my-scheduler-69cfc986c7-dtmhw       1/1     Running   0          6s    10.244.2.20   kind-worker    <none>           <none>
nginx                               1/1     Running   0          2s    10.244.1.10   kind-worker2   <none>           <none>
nginx1                              1/1     Running   0          2s    10.244.2.21   kind-worker    <none>           <none>
```

```
I0612 00:07:02.397306       1 log.go:194] Pod nginx is in Prefilter phase.
I0612 00:07:02.397336       1 log.go:194] Pod label: A
I0612 00:07:02.397372       1 log.go:194] Pods len: 1
I0612 00:07:02.397905       1 log.go:194] Pod nginx is in Score phase. Calculate the score of Node kind-worker.
I0612 00:07:02.397921       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 52428800, memory: 1324290048
I0612 00:07:02.397933       1 log.go:194] Pod nginx is in Score phase. Calculate the score of Node kind-worker2.
I0612 00:07:02.397939       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 52428800, memory: 1861160960
I0612 00:07:02.398009       1 log.go:194] Pod nginx. Node kind-worker's socre 1324290048
I0612 00:07:02.398023       1 log.go:194] Pod nginx. Node kind-worker2's socre 1861160960
I0612 00:07:02.447247       1 log.go:194] Pod nginx1 is in Prefilter phase.
I0612 00:07:02.447278       1 log.go:194] Pod label: A
I0612 00:07:02.447326       1 log.go:194] Pods len: 2
I0612 00:07:02.447577       1 log.go:194] Pod nginx1 is in Score phase. Calculate the score of Node kind-worker.
I0612 00:07:02.447599       1 log.go:194] Node kind-worker allocatable memory: 1376718848, requested memory: 52428800, memory: 1324290048
I0612 00:07:02.447618       1 log.go:194] Pod nginx1 is in Score phase. Calculate the score of Node kind-worker2.
I0612 00:07:02.447633       1 log.go:194] Node kind-worker2 allocatable memory: 1913589760, requested memory: 1101004800, memory: 812584960
I0612 00:07:02.447799       1 log.go:194] Pod nginx1. Node kind-worker's socre 1324290048
I0612 00:07:02.447815       1 log.go:194] Pod nginx1. Node kind-worker2's socre 812584960
```

可以從scheduler的log與pod的資訊上看到，兩個pod確實是被schedule給各自對應的node，與理論結果一致。