

🚀 EKS Cluster Upgrade & Blue-Green Node Upgrade (Production Way) — Using Cordon + Drain (No nodeSelector)

For this I deployed application Before upgrading the Cluster and Nodes, To show that the applications still up and running

```
34.230.38.116 | 172.31.90.11 | m7i-flex.large | https://github.com/Prakashreddy14306/Helm.git
[ ec2-user@ip-172-31-90-11 ~/Helm/expense-helm/frontend ]$ kubectl get pods
NAME           READY   STATUS    RESTARTS   AGE
backend-84b755f586-2q95g   1/1     Running   0          44m
backend-84b755f586-7zw2j   1/1     Running   0          44m
frontend-7c44f58b66-5v7zc  1/1     Running   0          37m
frontend-7c44f58b66-8rc55  1/1     Running   0          37m
mysql-0          1/1     Running   0          54m
mysql-1          1/1     Running   0          49m
mysql-2          1/1     Running   0          48m

34.230.38.116 | 172.31.90.11 | m7i-flex.large | https://github.com/Prakashreddy14306/Helm.git
```

✓ The best approach is **Blue → Green upgrade** using:

- **cordon**
- **drain**
- **delete nodegroup**

👉 No changes required in Deployments / Helm charts.

✓ Step-by-Step Process (eksctl)

1 Create EKS Cluster with BLUE NodeGroup

```
eksctl create cluster -f cluster.yaml
```

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: expense
  region: us-east-1
  version: "1.29"

managedNodeGroups:
- name: blue-ng
  instanceTypes: ["c7i-flex.large", "m7i-flex.large"]
  desiredCapacity: 3
  minSize: 3
  maxSize: 6
  spot: true
  labels:
    color: blue
```

```
eksctl create cluster -f cluster.yaml
```

* To view the cluster version

```
34.230.38.116 | 172.31.90.11 | m7i-flex.large | https://github.com/Prakashreddy14306/Helm.git
[ ec2-user@ip-172-31-90-11 ~/Helm/expense-helm/mysql ]$ aws eks describe-cluster --name expense --region us-east-1 --query "cluster.version"
"1.29"
```

```
This image is designed for only for the lab purpose
of learning DevOps and not recommended at all to use
in production or any company environments.

Last login: Sun Feb 15 10:35:43 2026 from 124.123.173.124

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~ ]$ 

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~ ]$ kubectl get nodes
NAME           STATUS   ROLES      AGE    VERSION
ip-192-168-39-158.ec2.internal   Ready    <none>   3h56m   v1.29.15-eks-70ce843
ip-192-168-5-218.ec2.internal   Ready    <none>   3h56m   v1.29.15-eks-70ce843
ip-192-168-53-225.ec2.internal Ready    <none>   3h56m   v1.29.15-eks-70ce843
```

2 Upgrade EKS Control Plane

Now we are running cluster 1.29 and we upgrade this to 1.30

```
eksctl upgrade cluster \
--name expense \
--region us-east-1 \
--version 1.30 \
--approve
```

```
34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~ ]$ eksctl upgrade cluster \
--name expense \
--region us-east-1 \
--version 1.30 \
--approve
2026-02-15 13:35:47 [d] will upgrade cluster "expense" control plane from current version "1.29" to "1.30"
2026-02-15 13:43:22 [✓] cluster "expense" control plane has been upgraded to version "1.30"
2026-02-15 13:43:22 [d] you will need to follow the upgrade procedure for all of nodegroups and add-ons
2026-02-15 13:43:23 [d] re-building cluster stack "eksctl-expense-cluster"
2026-02-15 13:43:23 [✓] all resources in cluster stack "eksctl-expense-cluster" are up-to-date
2026-02-15 13:43:23 [d] checking security group configuration for all nodegroups
2026-02-15 13:43:23 [d] all nodegroups have up-to-date cloudformation templates

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~ ]$ aws eks describe-cluster --name expense --region us-east-1 --query "cluster.version"
"1.30"

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~ ]$ █
```

3 Create GREEN NodeGroup

```
eksctl create nodegroup -f green-ng.yaml
```

```
apiVersion: eksctl.io/v1alpha5
kind: ClusterConfig

metadata:
  name: expense
  region: us-east-1
  version: "1.30"
```

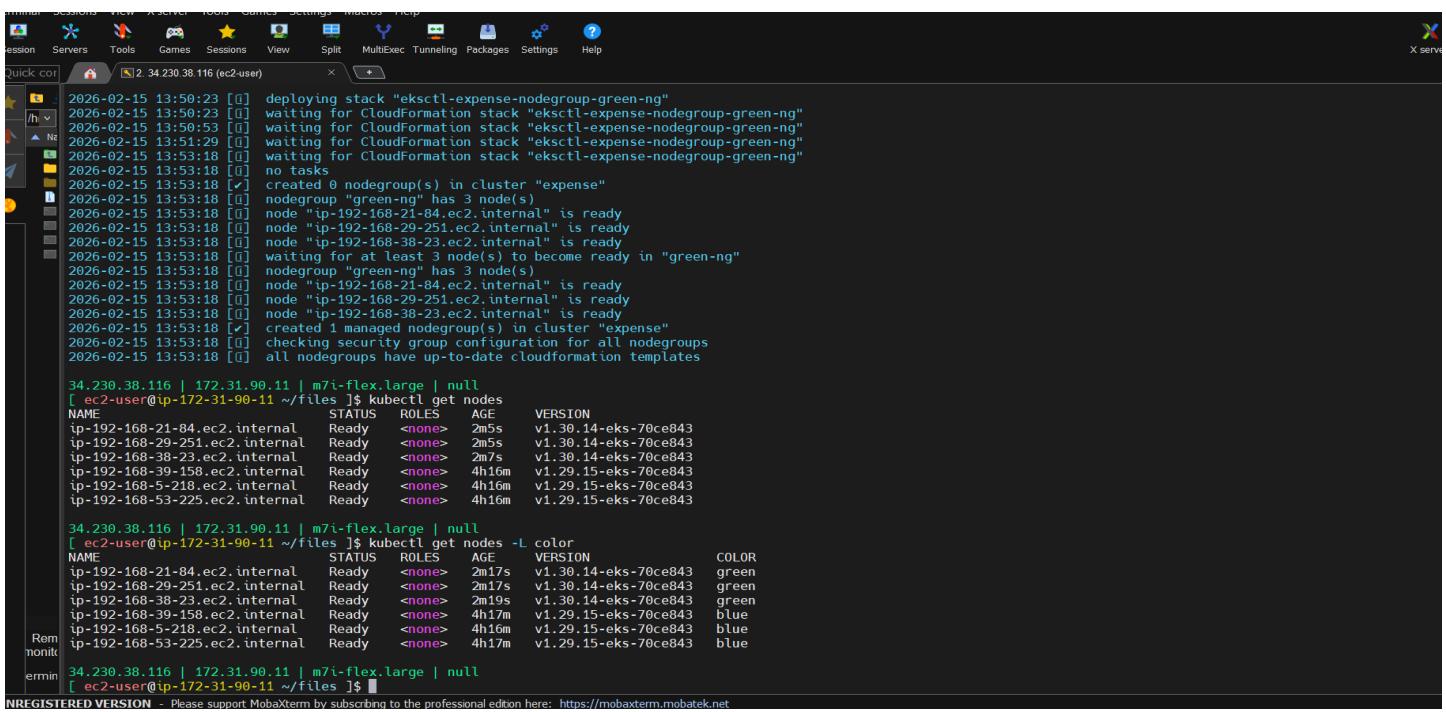
```

managedNodeGroups:
  - name: green-ng
    instanceTypes: ["c7i-flex.large", "m7i-flex.large"]
    desiredCapacity: 3
    minSize: 3
    maxSize: 6
    spot: true
    labels:
      color: green

```

Now cluster has:

- **blue-ng** (old nodes)
- **green-ng** (new nodes)



```

2026-02-15 13:50:23 [0] deploying stack "eksctl-expense-nodegroup-green-ng"
2026-02-15 13:50:23 [0] waiting for CloudFormation stack "eksctl-expense-nodegroup-green-ng"
2026-02-15 13:50:53 [0] waiting for CloudFormation stack "eksctl-expense-nodegroup-green-ng"
2026-02-15 13:51:29 [0] waiting for CloudFormation stack "eksctl-expense-nodegroup-green-ng"
2026-02-15 13:53:18 [0] waiting for CloudFormation stack "eksctl-expense-nodegroup-green-ng"
2026-02-15 13:53:18 [0] no tasks
2026-02-15 13:53:18 [✓] created 0 nodegroup(s) in cluster "expense"
2026-02-15 13:53:18 [0] nodegroup "green-ng" has 3 node(s)
2026-02-15 13:53:18 [0] node "ip-192-168-21-84.ec2.internal" is ready
2026-02-15 13:53:18 [0] node "ip-192-168-29-251.ec2.internal" is ready
2026-02-15 13:53:18 [0] node "ip-192-168-38-23.ec2.internal" is ready
2026-02-15 13:53:18 [0] waiting for at least 3 node(s) to become ready in "green-ng"
2026-02-15 13:53:18 [0] nodegroup "green-ng" has 3 node(s)
2026-02-15 13:53:18 [0] node "ip-192-168-21-84.ec2.internal" is ready
2026-02-15 13:53:18 [0] node "ip-192-168-29-251.ec2.internal" is ready
2026-02-15 13:53:18 [0] node "ip-192-168-38-23.ec2.internal" is ready
2026-02-15 13:53:18 [✓] created 1 managed nodegroup(s) in cluster "expense"
2026-02-15 13:53:18 [0] checking security group configuration for all nodegroups
2026-02-15 13:53:18 [0] all nodegroups have up-to-date cloudformation templates

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ kubectl get nodes
NAME           STATUS   ROLES      AGE   VERSION
ip-192-168-21-84.ec2.internal   Ready    <none>   2m5s   v1.30.14-eks-70ce843
ip-192-168-29-251.ec2.internal   Ready    <none>   2m5s   v1.30.14-eks-70ce843
ip-192-168-38-23.ec2.internal   Ready    <none>   2m7s   v1.30.14-eks-70ce843
ip-192-168-39-158.ec2.internal  Ready    <none>   4h16m  v1.29.15-eks-70ce843
ip-192-168-5-218.ec2.internal   Ready    <none>   4h16m  v1.29.15-eks-70ce843
ip-192-168-53-225.ec2.internal  Ready    <none>   4h16m  v1.29.15-eks-70ce843

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ kubectl get nodes -L color
NAME           STATUS   ROLES      AGE   VERSION   COLOR
ip-192-168-21-84.ec2.internal   Ready    <none>   2m17s  v1.30.14-eks-70ce843  green
ip-192-168-29-251.ec2.internal   Ready    <none>   2m17s  v1.30.14-eks-70ce843  green
ip-192-168-38-23.ec2.internal   Ready    <none>   2m19s  v1.30.14-eks-70ce843  green
ip-192-168-39-158.ec2.internal  Ready    <none>   4h17m  v1.29.15-eks-70ce843  blue
ip-192-168-5-218.ec2.internal   Ready    <none>   4h16m  v1.29.15-eks-70ce843  blue
ip-192-168-53-225.ec2.internal  Ready    <none>   4h17m  v1.29.15-eks-70ce843  blue

Remonit
ermint 34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ 

```

UNREGISTERED VERSION Please support MobaXterm by subscribing to the professional edition here: <https://mobaxterm.mobatek.net>

4 Cordon BLUE Nodes (Stop Scheduling New Pods)

Get nodes:

```
kubectl get nodes -L eksctl.io/nodegroup
```

Cordon blue nodes:

```
kubectl cordon <blue-node1>
kubectl cordon <blue-node2>
kubectl cordon <blue-node3>
```

Check status:

```
kubectl get nodes
```

You'll see:

SchedulingDisabled

```
34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ kubectl cordon ip-192-168-39-158.ec2.internal
node/ip-192-168-39-158.ec2.internal cordoned

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ kubectl cordon ip-192-168-5-218.ec2.internal
node/ip-192-168-5-218.ec2.internal cordoned

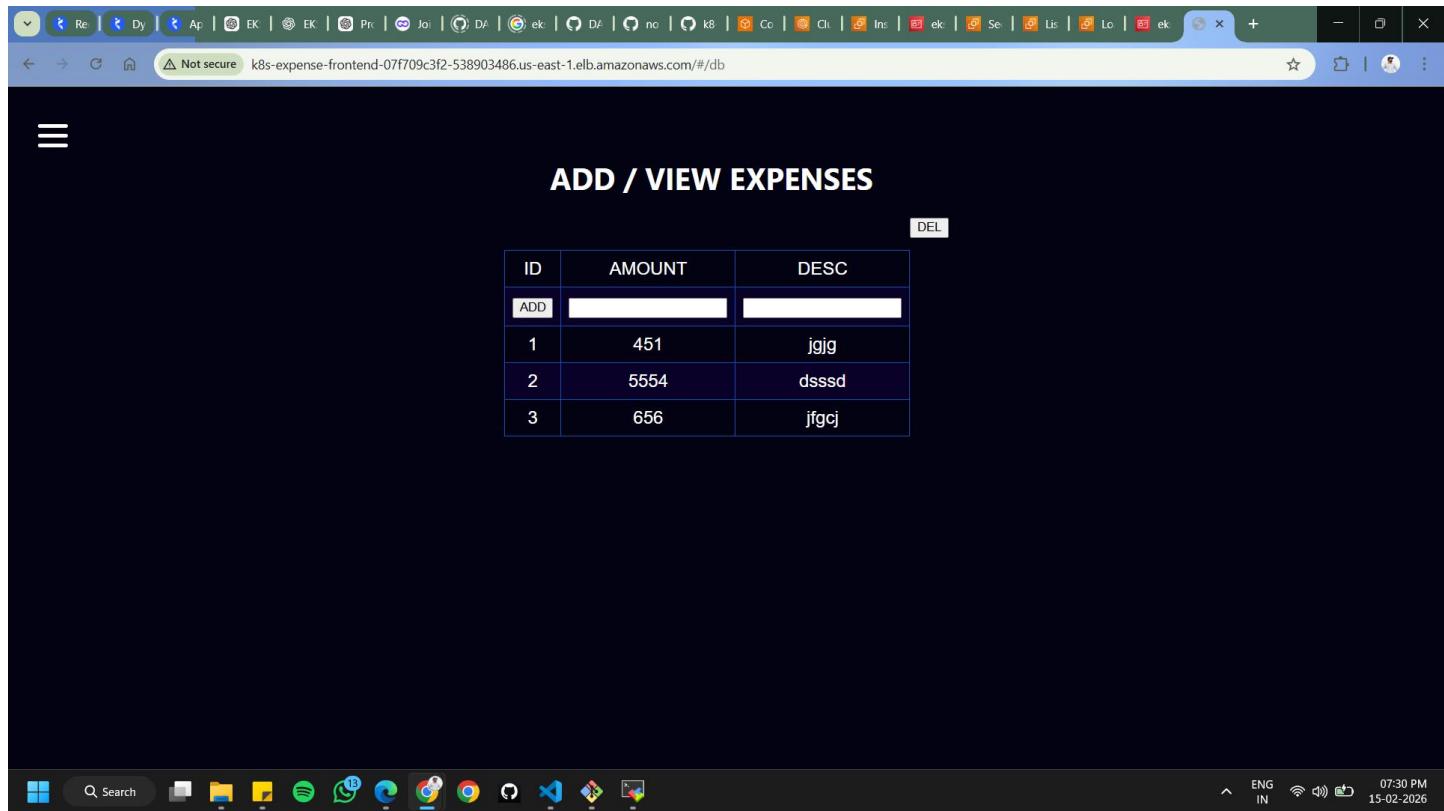
34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ kubectl cordon ip-192-168-53-225.ec2.internal
node/ip-192-168-53-225.ec2.internal cordoned

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$ kubectl get nodes -L color
NAME           STATUS      ROLES   AGE     VERSION      COLOR
ip-192-168-21-84.ec2.internal Ready      <none>  6m50s  v1.30.14-eks-70ce843 green
ip-192-168-29-251.ec2.internal Ready      <none>  6m50s  v1.30.14-eks-70ce843 green
ip-192-168-38-23.ec2.internal Ready      <none>  6m52s  v1.30.14-eks-70ce843 green
ip-192-168-39-158.ec2.internal Ready,SchedulingDisabled <none>  4h21m  v1.29.15-eks-70ce843 blue
ip-192-168-5-218.ec2.internal Ready,SchedulingDisabled <none>  4h21m  v1.29.15-eks-70ce843 blue
ip-192-168-53-225.ec2.internal Ready,SchedulingDisabled <none>  4h21m  v1.29.15-eks-70ce843 blue

34.230.38.116 | 172.31.90.11 | m7i-flex.large | null
[ ec2-user@ip-172-31-90-11 ~]$
```

 Now new pods will automatically schedule on **GREEN** nodes.

 Still the application is running



The screenshot shows a web browser window with the URL <https://k8s-expense-frontend-07f709c3f2-538903486.us-east-1.elb.amazonaws.com/#/db>. The page title is "ADD / VIEW EXPENSES". Below the title is a table with the following data:

ID	AMOUNT	DESC
1	451	jgjg
2	5554	dsssd
3	656	jfgcj

At the bottom of the browser window, the taskbar shows various application icons, and the system tray indicates the date and time as 15-02-2026 at 07:30 PM.

5 Drain BLUE Nodes (Move Pods to GREEN)

Drain nodes one-by-one:

```
kubectl drain <blue-node1> --ignore-daemonsets --delete-emptydir-data  
kubectl drain <blue-node2> --ignore-daemonsets --delete-emptydir-data  
kubectl drain <blue-node3> --ignore-daemonsets --delete-emptydir-data
```

👉 Drain evicts pods and Kubernetes recreates them on green nodes.

Blue nodes are drained and automatically all the pods are shifted to green nodes, Check age of the the pods.

NAME	PF	READY	STATUS	RESTARTS	CPU	%CPU/R	%CPU/L	MEM	%MEM/R	%MEM/L	IP	NODE	AGE
back-end-84b755f586-27rxc	●	1/1	Running	0	0	n/a	n/a	0	n/a	n/a	192.168.43.26	ip-192-168-38-23.ec2.internal	31s
backend-84b755f586-z5tvb	●	1/1	Running	1	25	n/a	n/a	16	n/a	n/a	192.168.18.222	ip-192-168-21-84.ec2.internal	73s
frontend-7c44f58b66-cb2vh	●	1/1	Running	0	2	2	1	2	4	2	192.168.4.40	ip-192-168-29-251.ec2.internal	73s
frontend-7c44f58b66-f97q6	●	1/1	Running	0	1	1	0	2	4	2	192.168.38.244	ip-192-168-38-23.ec2.internal	97s
mysql-0	●	1/1	Running	0	5	n/a	n/a	376	n/a	n/a	192.168.58.243	ip-192-168-38-23.ec2.internal	28s
mysql-1	●	1/1	Running	0	0	n/a	n/a	0	n/a	n/a	192.168.4.216	ip-192-168-29-251.ec2.internal	69s
mysql-2	●	1/1	Running	0	86	n/a	n/a	349	n/a	n/a	192.168.32.152	ip-192-168-38-23.ec2.internal	93s

6 Validate Everything Running on GREEN

```
kubectl get pods -A -o wide  
kubectl get nodes -L eksctl.io/nodegroup
```

Check the Ip addresses' all the pods are running in green nodes

NAME	STATUS	ROLES	AGE	VERSION	COLOR
ip-192-168-21-84.ec2.internal	Ready	<none>	12m	v1.30.14-eks-70ce843	green
ip-192-168-29-251.ec2.internal	Ready	<none>	12m	v1.30.14-eks-70ce843	green
ip-192-168-38-23.ec2.internal	Ready	<none>	12m	v1.30.14-eks-70ce843	green
ip-192-168-39-158.ec2.internal	Ready,SchedulingDisabled	<none>	4h27m	v1.29.15-eks-70ce843	blue
ip-192-168-5-218.ec2.internal	Ready,SchedulingDisabled	<none>	4h27m	v1.29.15-eks-70ce843	blue
ip-192-168-53-225.ec2.internal	Ready,SchedulingDisabled	<none>	4h27m	v1.29.15-eks-70ce843	blue

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS
expense	backend-84b755f586-27rxc	1/1	Running	1 (66s ago)	118s	192.168.43.26	ip-192-168-38-23.ec2.internal	<none>	<none>
expense	backend-84b755f586-z5tvb	1/1	Running	1 (116s ago)	2m40s	192.168.18.222	ip-192-168-21-84.ec2.internal	<none>	<none>
expense	frontend-7c44f58b66-cb2vh	1/1	Running	0	2m40s	192.168.4.40	ip-192-168-29-251.ec2.internal	<none>	<none>
expense	frontend-7c44f58b66-f97q6	1/1	Running	0	3m4s	192.168.38.244	ip-192-168-38-23.ec2.internal	<none>	<none>
expense	mysql-0	1/1	Running	0	115s	192.168.58.243	ip-192-168-38-23.ec2.internal	<none>	<none>
expense	mysql-1	1/1	Running	0	2m36s	192.168.4.216	ip-192-168-29-251.ec2.internal	<none>	<none>
expense	mysql-2	1/1	Running	0	3m	192.168.32.152	ip-192-168-38-23.ec2.internal	<none>	<none>

After checking all the objects are running perfectly, we can delete blue nodes.

7 Delete BLUE NodeGroup

After confirmation:

```
eksctl delete nodegroup \
--cluster expense \
--name blue-ng \
--region us-east-1
```

- ✓ Now only green nodes remain.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
expense-blue-ng-Node	i-0b778694fc605e5a2	Shutting-d...	c7i-flex.large	-	View alarms +	us-east-1b
expense-blue-ng-Node	i-0a08d4e640714cc11	Shutting-d...	c7i-flex.large	-	View alarms +	us-east-1a
expense-blue-ng-Node	i-012e459debc48b014	Shutting-d...	c7i-flex.large	-	View alarms +	us-east-1a
expense-green-ng-Node	i-0fe775adc5c4dcdb8a	Running	c7i-flex.large	3/3 checks passec	View alarms +	us-east-1b
expense-green-ng-Node	i-0ecf64d1ae45f2c98	Running	c7i-flex.large	3/3 checks passec	View alarms +	us-east-1b
expense-green-ng-Node	i-0071a109d24a496e6	Running	c7i-flex.large	3/3 checks passec	View alarms +	us-east-1a
MyEC2Instance	i-05b889d4e323c3bbb	Running	m7i-flex.large	3/3 checks passec	View alarms +	us-east-1b

★ Why This Approach is Best (Production Standard)

- ✓ No Deployment changes
- ✓ No Helm chart updates
- ✓ Smooth rolling migration
- ✓ Clean upgrade strategy
- ✓ Supports zero downtime (if replicas + PDB exist)

PDB (Most Common)

If your deployment has 3 replicas

You can allow only 1 pod disruption at a time.

```
apiVersion: policy/v1
kind: PodDisruptionBudget
metadata:
  name: expense-app-pdb
spec:
  minAvailable: 2
  selector:
    matchLabels:
      app: expense
```

Meaning:

- Total pods = 3
- At least **2 pods must always be running**
- Only **1 pod can be evicted at a time**