Kubernetes,Daemomset,taints and toleration (ASSIGNMENT-13)

1. AWS Setup for EKS Cluster

Step 1: Configure AWS CLI

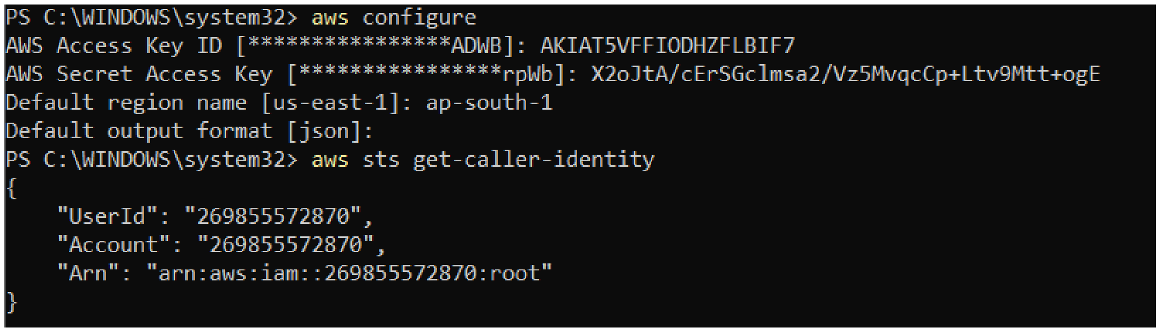
Command: aws configure

Inputs:
- AWS Access Key ID
- AWS Secret Access Key
- Region (e.g., ap-south-1)
- Output format (e.g., json)

Step 2: Verify Identity

Command: aws sts get-caller-identity

Returns your AWS Account ID, User ARN, and User ID.

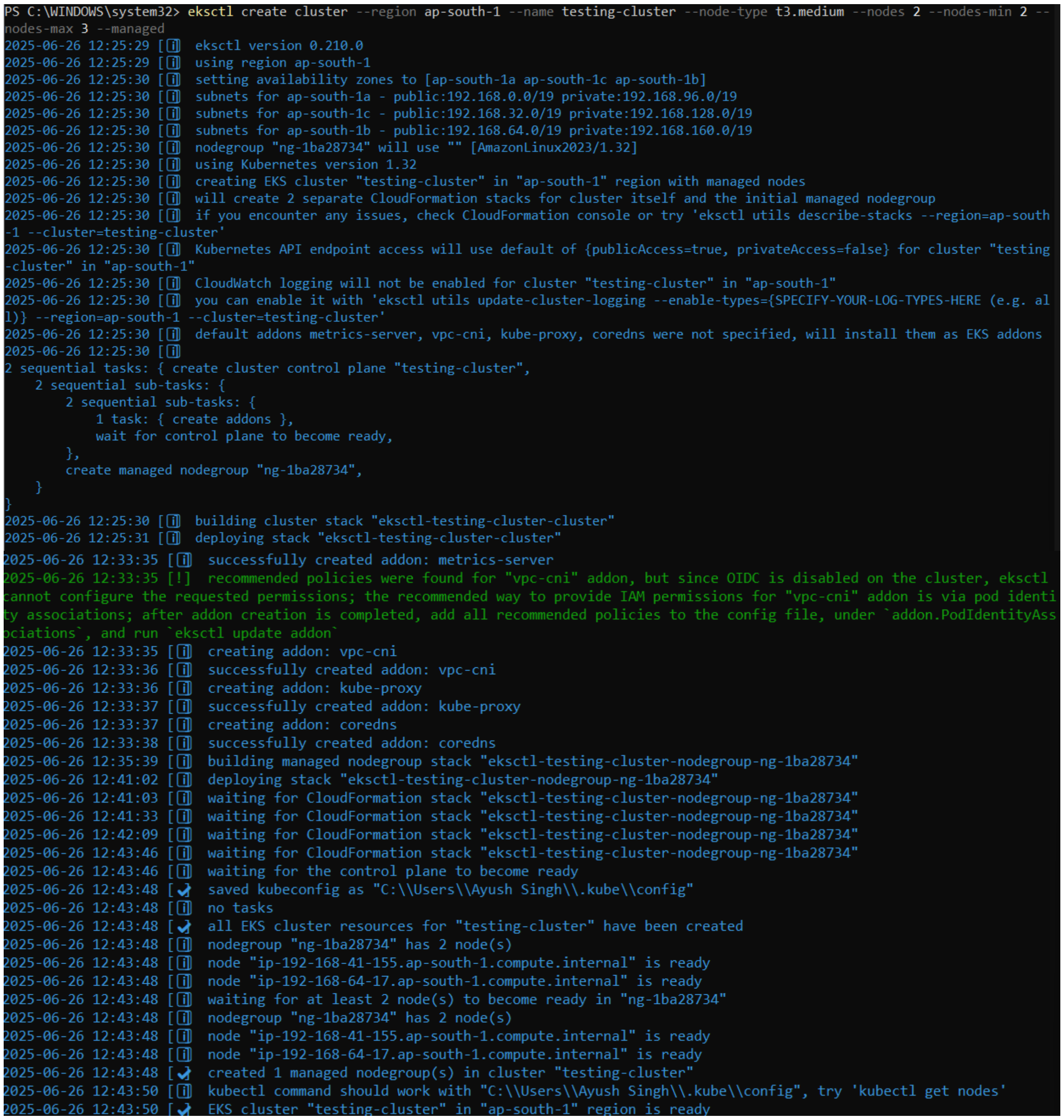


```
PS C:\WINDOWS\system32> aws configure
AWS Access Key ID [****************ADWB]: AKIAT5VFFIODHZFLBIF7
AWS Secret Access Key [****************rpWb]: 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
Default region name [us-east-1]: ap-south-1
Default output format [json]:
PS C:\WINDOWS\system32> aws sts get-caller-identity
{
    "UserId": "269855572870",
    "Account": "269855572870",
    "Arn": "arn:aws:iam::269855572870:root"
}
```

2. Create EKS Cluster Using eksctl

Command:

eksctl create cluster --name demo-cluster --region ap-south-1 --nodegroup-name standard-workers --node-type t3.medium --nodes 2 --nodes-min 1 --nodes-max 3 –managed

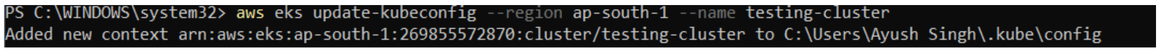


```
PS C:\WINDOWS\system32> eksctl create cluster --region ap-south-1 --name testing-cluster --node-type t3.medium --nodes 2 --nodes-min 2 --
nodes-max 3 --managed
2025-06-26 12:25:29 [ℹ]  eksctl version 0.210.0
2025-06-26 12:25:29 [ℹ]  using region ap-south-1
2025-06-26 12:25:30 [ℹ]  setting availability zones to [ap-south-1a ap-south-1c ap-south-1b]
2025-06-26 12:25:30 [ℹ]  subnets for ap-south-1a - public:192.168.0.0/19 private:192.168.96.0/19
2025-06-26 12:25:30 [ℹ]  subnets for ap-south-1c - public:192.168.32.0/19 private:192.168.128.0/19
2025-06-26 12:25:30 [ℹ]  subnets for ap-south-1b - public:192.168.64.0/19 private:192.168.160.0/19
2025-06-26 12:25:30 [ℹ]  nodegroup "ng-1ba28734" will use "" [AmazonLinux2023/1.32]
2025-06-26 12:25:30 [ℹ]  using Kubernetes version 1.32
2025-06-26 12:25:30 [ℹ]  creating EKS cluster "testing-cluster" in "ap-south-1" region with managed nodes
2025-06-26 12:25:30 [ℹ]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2025-06-26 12:25:30 [ℹ]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=ap-south
-1 --cluster=testing-cluster'
2025-06-26 12:25:30 [ℹ]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "testing
-cluster" in "ap-south-1"
2025-06-26 12:25:30 [ℹ]  CloudWatch logging will not be enabled for cluster "testing-cluster" in "ap-south-1"
2025-06-26 12:25:30 [ℹ]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. al
l)} --region=ap-south-1 --cluster=testing-cluster'
2025-06-26 12:25:30 [ℹ]  default addons metrics-server, vpc-cni, kube-proxy, coredns were not specified, will install them as EKS addons
2025-06-26 12:25:30 [ℹ]
2 sequential tasks: { create cluster control plane "testing-cluster",
    2 sequential sub-tasks: {
        2 sequential sub-tasks: {
            1 task: { create addons },
            wait for control plane to become ready,
        },
        create managed nodegroup "ng-1ba28734",
    }
}
2025-06-26 12:25:30 [ℹ]  building cluster stack "eksctl-testing-cluster-cluster"
2025-06-26 12:25:31 [ℹ]  deploying stack "eksctl-testing-cluster-cluster"
2025-06-26 12:33:35 [ℹ]  successfully created addon: metrics-server
2025-06-26 12:33:35 [!]  recommended policies were found for "vpc-cni" addon, but since OIDC is disabled on the cluster, eksctl
cannot configure the requested permissions; the recommended way to provide IAM permissions for "vpc-cni" addon is via pod identi
ty associations; after addon creation is completed, add all recommended policies to the config file, under `addon.PodIdentityAss
ociations`, and run `eksctl update addon`
2025-06-26 12:33:35 [ℹ]  creating addon: vpc-cni
2025-06-26 12:33:36 [ℹ]  successfully created addon: vpc-cni
2025-06-26 12:33:36 [ℹ]  creating addon: kube-proxy
2025-06-26 12:33:37 [ℹ]  successfully created addon: kube-proxy
2025-06-26 12:33:37 [ℹ]  creating addon: coredns
2025-06-26 12:33:38 [ℹ]  successfully created addon: coredns
2025-06-26 12:35:39 [ℹ]  building managed nodegroup stack "eksctl-testing-cluster-nodegroup-ng-1ba28734"
2025-06-26 12:41:02 [ℹ]  deploying stack "eksctl-testing-cluster-nodegroup-ng-1ba28734"
2025-06-26 12:41:03 [ℹ]  waiting for CloudFormation stack "eksctl-testing-cluster-nodegroup-ng-1ba28734"
2025-06-26 12:41:33 [ℹ]  waiting for CloudFormation stack "eksctl-testing-cluster-nodegroup-ng-1ba28734"
2025-06-26 12:42:09 [ℹ]  waiting for CloudFormation stack "eksctl-testing-cluster-nodegroup-ng-1ba28734"
2025-06-26 12:43:46 [ℹ]  waiting for CloudFormation stack "eksctl-testing-cluster-nodegroup-ng-1ba28734"
2025-06-26 12:43:46 [ℹ]  waiting for the control plane to become ready
2025-06-26 12:43:48 [✔]  saved kubeconfig as "C:\\Users\\Ayush Singh\\.kube\\config"
2025-06-26 12:43:48 [ℹ]  no tasks
2025-06-26 12:43:48 [✔]  all EKS cluster resources for "testing-cluster" have been created
2025-06-26 12:43:48 [ℹ]  nodegroup "ng-1ba28734" has 2 node(s)
2025-06-26 12:43:48 [ℹ]  node "ip-192-168-41-155.ap-south-1.compute.internal" is ready
2025-06-26 12:43:48 [ℹ]  node "ip-192-168-64-17.ap-south-1.compute.internal" is ready
2025-06-26 12:43:48 [ℹ]  waiting for at least 2 node(s) to become ready in "ng-1ba28734"
2025-06-26 12:43:48 [ℹ]  nodegroup "ng-1ba28734" has 2 node(s)
2025-06-26 12:43:48 [ℹ]  node "ip-192-168-41-155.ap-south-1.compute.internal" is ready
2025-06-26 12:43:48 [ℹ]  node "ip-192-168-64-17.ap-south-1.compute.internal" is ready
2025-06-26 12:43:48 [✔]  created 1 managed nodegroup(s) in cluster "testing-cluster"
2025-06-26 12:43:50 [ℹ]  kubectl command should work with "C:\\Users\\Ayush Singh\\.kube\\config", try 'kubectl get nodes'
2025-06-26 12:43:50 [✔]  EKS cluster "testing-cluster" in "ap-south-1" region is ready
```

3. Kubernetes Routing Structure

Manual EC2 Hosting:
EC2 → Application Host → Target Group → Load Balancer → DNS (Access)

Kubernetes Routing:
Service → Deployment → Pods



```
PS C:\WINDOWS\system32> aws eks update-kubeconfig --region ap-south-1 --name testing-cluster
Added new context arn:aws:eks:ap-south-1:269855572870:cluster/testing-cluster to C:\Users\Ayush Singh\.kube\config
```

Apply Commands:

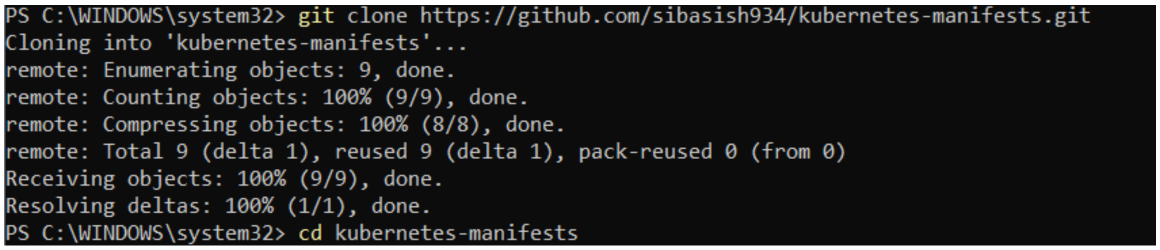
kubectl apply -f deployment.yaml
kubectl apply -f service.yaml

4. Clone GitHub Manifests Repository

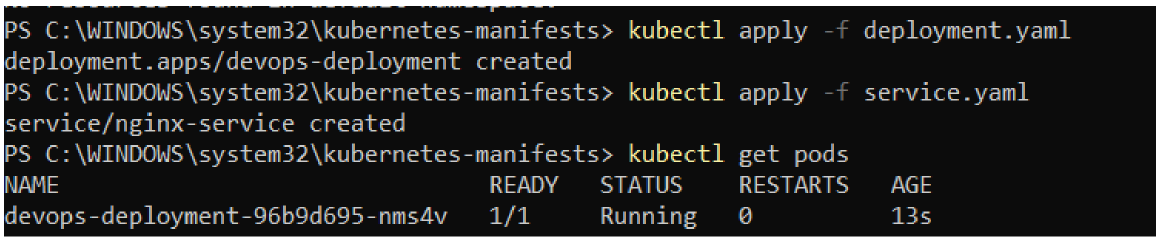
Repository: https://github.com/sibasish934/kubernetes-manifests

Commands:

git clone https://github.com/sibasish934/kubernetes-manifests.git
cd kubernetes-manifests



```
PS C:\WINDOWS\system32> git clone https://github.com/sibasish934/kubernetes-manifests.git
Cloning into 'kubernetes-manifests'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (8/8), done.
remote: Total 9 (delta 1), reused 9 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (1/1), done.
PS C:\WINDOWS\system32> cd kubernetes-manifests
```



```
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f deployment.yaml
deployment.apps/devops-deployment created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f service.yaml
service/nginx-service created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
devops-deployment-96b9d695-nms4v    1/1     Running   0          13s
```

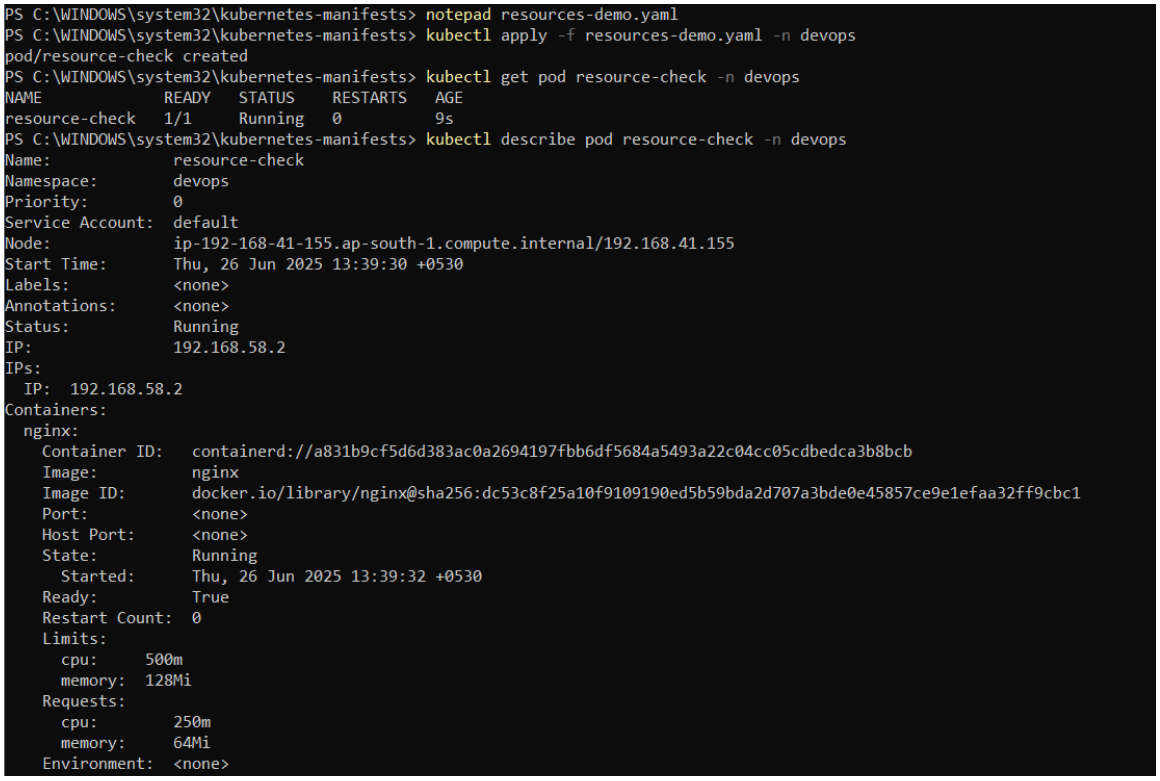
5. Resource Requests and Limits

Problem: One pod can consume too many resources and affect others.

Solution: Use resource requests and limits.

```
resources:
  requests:
    cpu: "250m"
    memory: "64Mi"
  limits:
    cpu: "500m"
    memory: "128Mi"
```

Apply: kubectl apply -f resources-demo.yaml



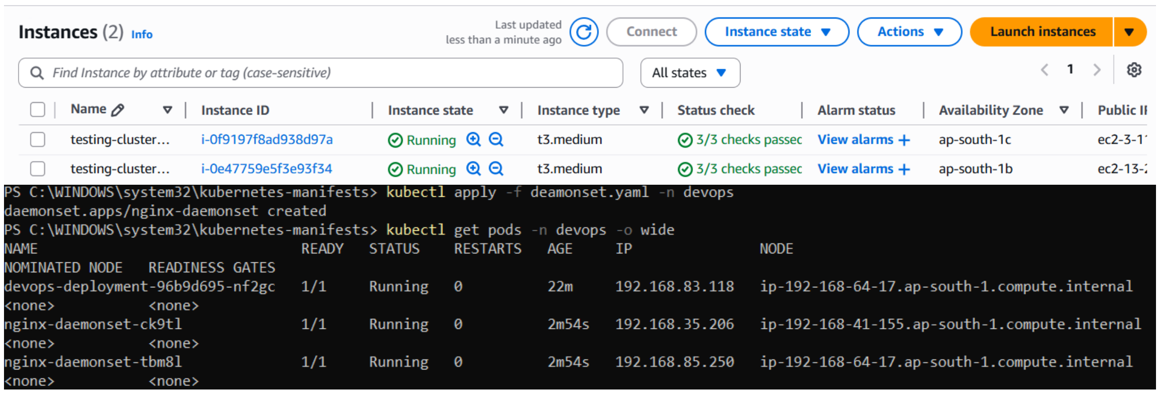
```
PS C:\WINDOWS\system32\kubernetes-manifests> notepad resources-demo.yaml
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f resources-demo.yaml -n devops
pod/resource-check created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get pod resource-check -n devops
NAME             READY   STATUS    RESTARTS   AGE
resource-check   1/1     Running   0          9s
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl describe pod resource-check -n devops
Name:             resource-check
Namespace:        devops
Priority:         0
Service Account:  default
Node:             ip-192-168-41-155.ap-south-1.compute.internal/192.168.41.155
Start Time:       Thu, 26 Jun 2025 13:39:30 +0530
Labels:           <none>
Annotations:      <none>
Status:           Running
IP:               192.168.58.2
IPs:
  IP:  192.168.58.2
Containers:
  nginx:
    Container ID:   containerd://a831b9cf5d6d383ac0a2694197fbb6df5684a5493a22c04cc05cdbedca3b8bcb
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:dc53c8f25a10f9109190ed5b59bda2d707a3bde0e45857ce9e1efaa32ff9cbc1
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Thu, 26 Jun 2025 13:39:32 +0530
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     500m
      memory:  128Mi
    Requests:
      cpu:        250m
      memory:     64Mi
    Environment:  <none>
```

6. Deploy a DaemonSet

Purpose: Ensure that a pod runs on every node.

Use-case: Logging agents, monitoring agents.

Command: kubectl apply -f daemonset.yaml


Instances (2) Info
Last updated less than a minute ago
Connect
Instance state ▼
Actions ▼
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states ▼
1

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public I
☐	testing-cluster…	i-0f9197f8ad938d97a	Running	t3.medium	3/3 checks passec	View alarms +	ap-south-1c	ec2-3-1
☐	testing-cluster…	i-0e47759e5f3e93f34	Running	t3.medium	3/3 checks passec	View alarms +	ap-south-1b	ec2-13-

```
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f deamonset.yaml -n devops
daemonset.apps/nginx-daemonset created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get pods -n devops -o wide
NAME                                READY   STATUS    RESTARTS   AGE    IP               NODE
NOMINATED NODE   READINESS GATES
devops-deployment-96b9d695-nf2gc    1/1     Running   0          22m    192.168.83.118   ip-192-168-64-17.ap-south-1.compute.internal
<none>           <none>
nginx-daemonset-ck9tl               1/1     Running   0          2m54s  192.168.35.206   ip-192-168-41-155.ap-south-1.compute.internal
<none>           <none>
nginx-daemonset-tbm8l               1/1     Running   0          2m54s  192.168.85.250   ip-192-168-64-17.ap-south-1.compute.internal
<none>           <none>
```

7. Taints and Tolerations

Use taints to prevent pods from being scheduled unless they have matching tolerations.

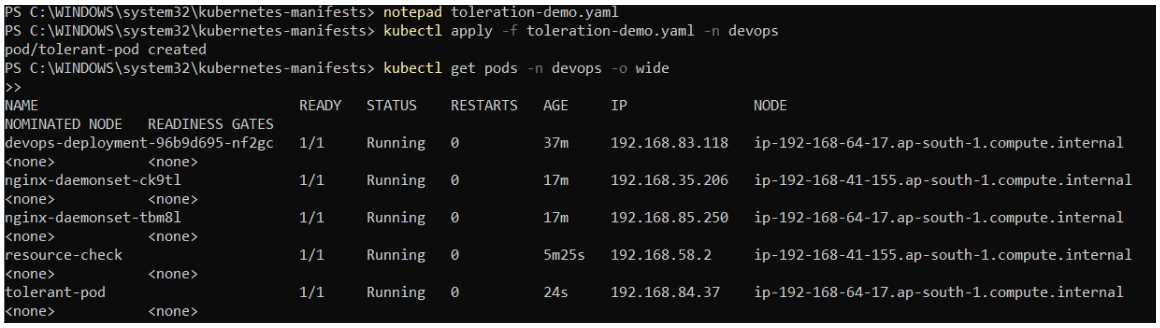
Taint a node:

kubectl taint nodes <node-name> key=value:NoSchedule

Toleration YAML:

```
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

Apply: kubectl apply -f toleration-demo.yaml



```
PS C:\WINDOWS\system32\kubernetes-manifests> notepad toleration-demo.yaml
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl apply -f toleration-demo.yaml -n devops
pod/tolerant-pod created
PS C:\WINDOWS\system32\kubernetes-manifests> kubectl get pods -n devops -o wide
>>
NAME                              READY   STATUS    RESTARTS   AGE     IP               NODE
NOMINATED NODE   READINESS GATES
devops-deployment-96b9d695-nf2gc   1/1     Running   0          37m     192.168.83.118   ip-192-168-64-17.ap-south-1.compute.internal
<none>           <none>
nginx-daemonset-ck9tl              1/1     Running   0          17m     192.168.35.206   ip-192-168-41-155.ap-south-1.compute.internal
<none>           <none>
nginx-daemonset-tbm8l              1/1     Running   0          17m     192.168.85.250   ip-192-168-64-17.ap-south-1.compute.internal
<none>           <none>
resource-check                     1/1     Running   0          5m25s   192.168.58.2     ip-192-168-41-155.ap-south-1.compute.internal
<none>           <none>
tolerant-pod                       1/1     Running   0          24s     192.168.84.37    ip-192-168-64-17.ap-south-1.compute.internal
<none>           <none>
```

8. Expose App to Internet (optional)

Expose deployment via LoadBalancer service:

kubectl expose deployment <deployment-name> --type=LoadBalancer --name=<service-name>

Check external IP:

kubectl get svc

9. Useful kubectl Commands

```
kubectl get all                  # View all resources
kubectl get nodes                # List cluster nodes
kubectl get pods -o wide         # View pod info with node IPs
kubectl describe pod <name>      # Describe pod in detail
kubectl logs <pod-name>          # View logs
kubectl exec -it <pod> -- sh     # Shell access to pod
kubectl delete -f <file.yaml>      # Delete a resource
```

10. Clean Up Resources

Delete all resources in 'devops' namespace:

```
kubectl delete all --all -n devops
```

Delete individual pods if needed:

```
kubectl delete pod resource-check -n devops
kubectl delete pod tolerant-pod -n devops
```

Delete the namespace (optional):

```
kubectl delete namespace devops
```

Delete the EKS cluster (IMPORTANT):

```
eksctl delete cluster --name demo-cluster --region ap-south-1
```