**SESSION - 65**

**--> cd /c/devops/daws-84s/repos/docker**

**--> terraform apply -auto-approve**

**--> sudo usermod -aG docker ec2-user**

**--> aws configure**

**--> git clone <https://github.com/Shan23-hash/docker.git>**

**--> eksctl create cluster --config-file=eks.yaml**

**--> kubectl get nodes**

Hostname

Username

Password

Deployment --> 10 pods

2 new pods create

2 new pods terminate

1st request -> new pod

2nd request --> old pod

for few sec, your application is serving 2 versions

**taints and tolerations**

**======================**

nodeSelector --> selects the node for us.. pod status is pending

--> In noe resources running fully what will happen the pod status

taints --> informing scheduler not to schedule pods onto this node.

PreferNoSchedule --> requesting scheduler not to schedule the pod. Scheduler still schedules the pod

NoSchedule --> ordering the scheduler not to schedule the pod

NoExecute --> NoSchedule + Evict the pods already running on that node

Why taints?

1. Special Hardware for some special projects

2. Networking requirements

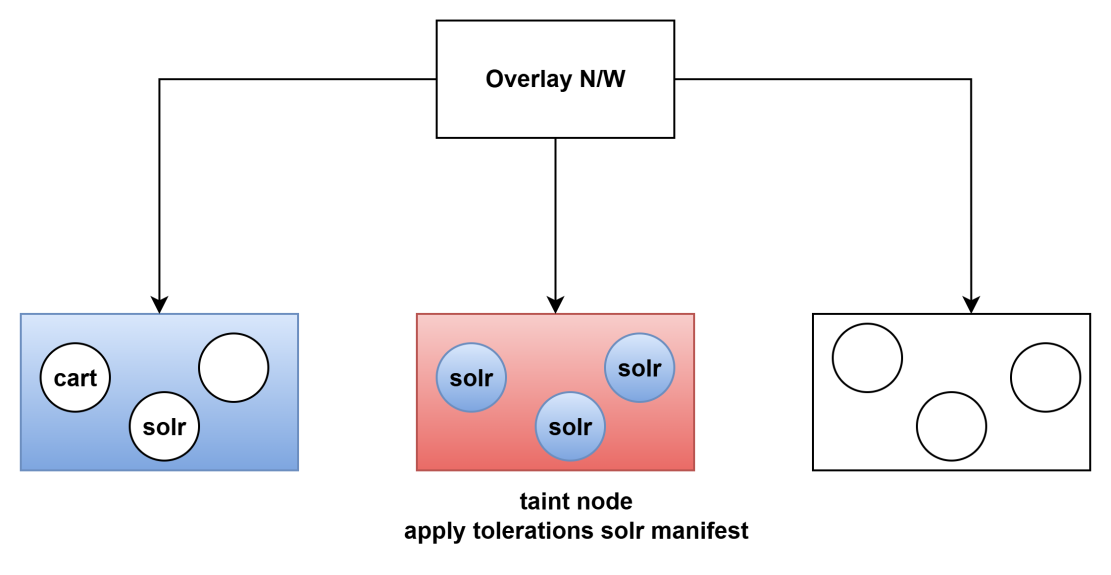
3. These nodes only may have DB connections

4. High priority applications

tolerations

===========

Solr pods responsible for search results



**Affinity and Anti-Affinity**

**=========================**

## Affinity and anti-affinity

nodeSelector is the simplest way to constrain Pods to nodes with specific labels. Affinity and anti-affinity expand the types of constraints you can define. Some of the benefits of affinity and anti-affinity include:

* The affinity/anti-affinity language is more expressive. nodeSelector only selects nodes with all the specified labels. Affinity/anti-affinity gives you more control over the selection logic.
* You can indicate that a rule is soft or preferred, so that the scheduler still schedules the Pod even if it can't find a matching node.
* You can constrain a Pod using labels on other Pods running on the node (or other topological domain), instead of just node labels, which allows you to define rules for which Pods can be co-located on a node.

The affinity feature consists of two types of affinity:

* Node affinity functions like the nodeSelector field but is more expressive and allows you to specify soft rules.
* Inter-pod affinity/anti-affinity allows you to constrain Pods against labels on other Pods.

### Node affinity

Node affinity is conceptually similar to nodeSelector, allowing you to constrain which nodes your Pod can be scheduled on based on node labels. There are two types of node affinity:

* requiredDuringSchedulingIgnoredDuringExecution: The scheduler can't schedule the Pod unless the rule is met. This functions like nodeSelector, but with a more expressive syntax.
* preferredDuringSchedulingIgnoredDuringExecution: The scheduler tries to find a node that meets the rule. If a matching node is not available, the scheduler still schedules the Pod.

Affinity = nodeSelector + more matching rules with operators like in, Exist, NotExist, gt, tlt etc

requiredDuringSchedulingIgnoredDuringExecution(hard rule) --> pod status pending if node not matches

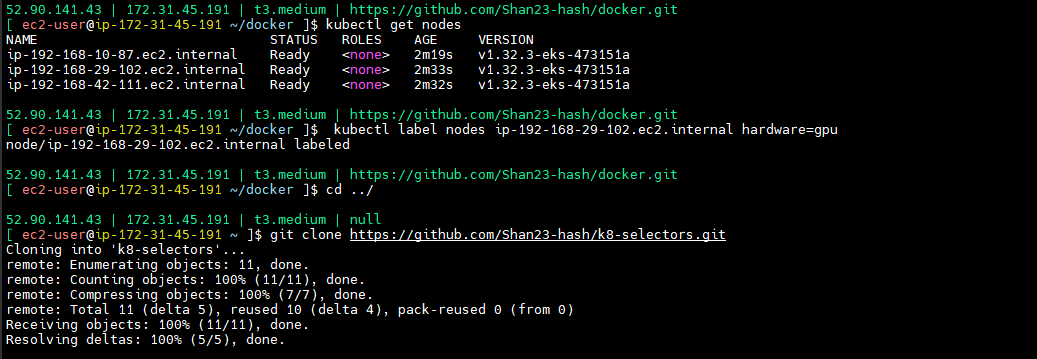
--> this is hard rule prefer during schedule, lables should be there some other place it will run

preferredDuringSchedulingIgnoredDuringExecution(soft rule) --> if labels are not matched, then scheduler schedules on to some other node

--> this is soft rule while executing no need

**--> kubectl get nodes**

**--> kubectl label nodes ip-192-168-29-102.ec2.internal hardware=gpu** (internal take from kubectl get nodes)



affinity = like = attraction

Not should attract to this pod

**03-node-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: with-node-affinity

spec:

affinity:

nodeAffinity:

# requiredDuringSchedulingIgnoredDuringExecution:

# nodeSelectorTerms:

# - matchExpressions:

# - key: hardware

# operator: In

# values:

# #- gpu

# - gpuuu

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 1

preference:

matchExpressions:

- key: another-node-label-key

operator: In

values:

- another-node-label-value

containers:

- name: with-node-affinity

image: nginx

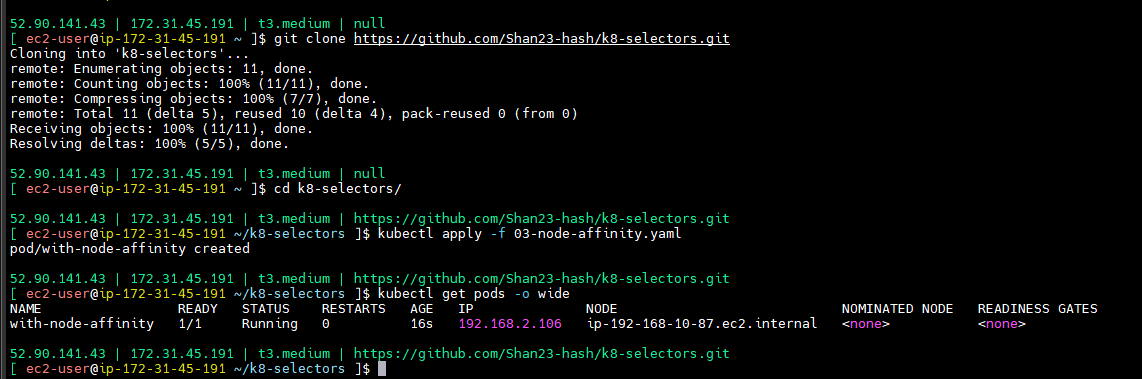
--> push and pull the code

--> this is all controlling.

**--> git clone https://github.com/Shan23-hash/k8-selectors.git**

**--> kubectl apply -f 03-node-affinity.yaml**

**--> kubectl get pods -o wide**



**-->** if not matched status it will come in pending

--> if you given preferred that one it will may success.

--> you can give many preferences.

--> this is soft rule, this one will 1 to 100 whatever it will match there it will schedule if not matched one also they their wish one place it will schedule

--> node-affinity is more flexible you can have soft rule and hard rule and you can have operations. So you can select multiple values.

--> anti-affinity

NOT in -- the label value is not contained in the supplied set of strings.

**04-node-anti-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: with-node-anti-affinity

spec:

affinity:

nodeAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

nodeSelectorTerms:

- matchExpressions:

- key: hardware

operator: NotIn

values:

- gpu

containers:

- name: with-node-anti-affinity

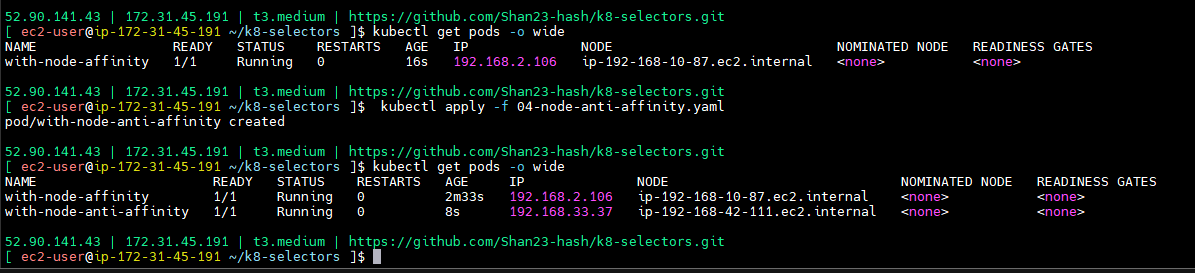
image: nginx

--> I given notin pod will go except 11.251

--> push and pull the code

**--> kubectl apply -f 04-node-anti-affinity.yaml**

**--> kubectl get pods -o wide**



--> for pods which node have to select that should be in our control.

In --> matching the labels

pod affinity

=============

pod-1 is in one node 11.252

pod-2 likes pod-1 it wants to run in the same node as pod-1

**05-pod-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: pod-1

spec:

containers:

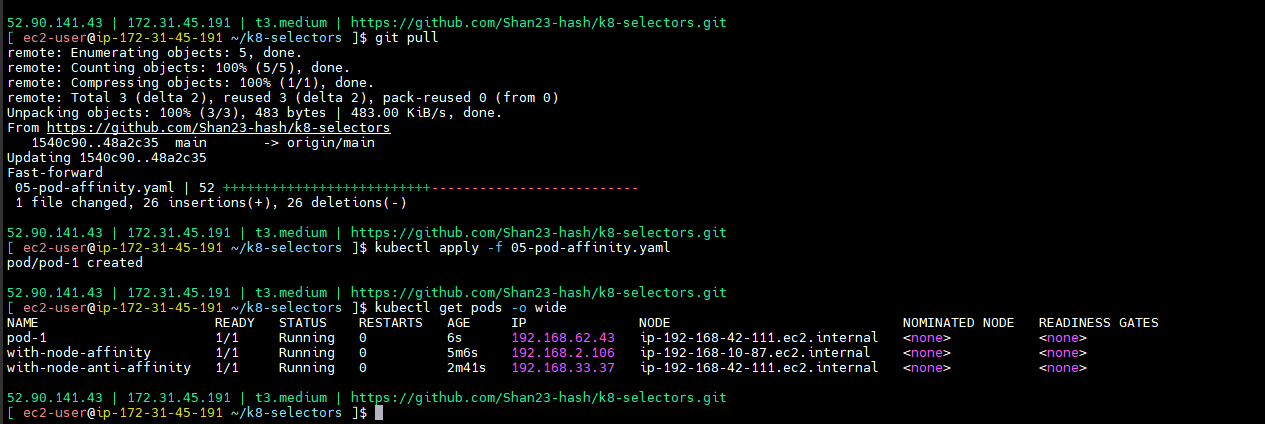
- name: pod-1

image: nginx

--> push and pull the code

**--> kubectl apply -f 05-pod-affinity.yaml**

**--> kubectl get pods -o wide**



**-->** pod running in 11.252

--> we are not given anything schedular it will select randomly

--> pod-1 likes pod-2 so it will go taat one where it is.

**05-pod-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: pod-1

labels:

app-name: pod-1

purpose: pod-affinity

spec:

containers:

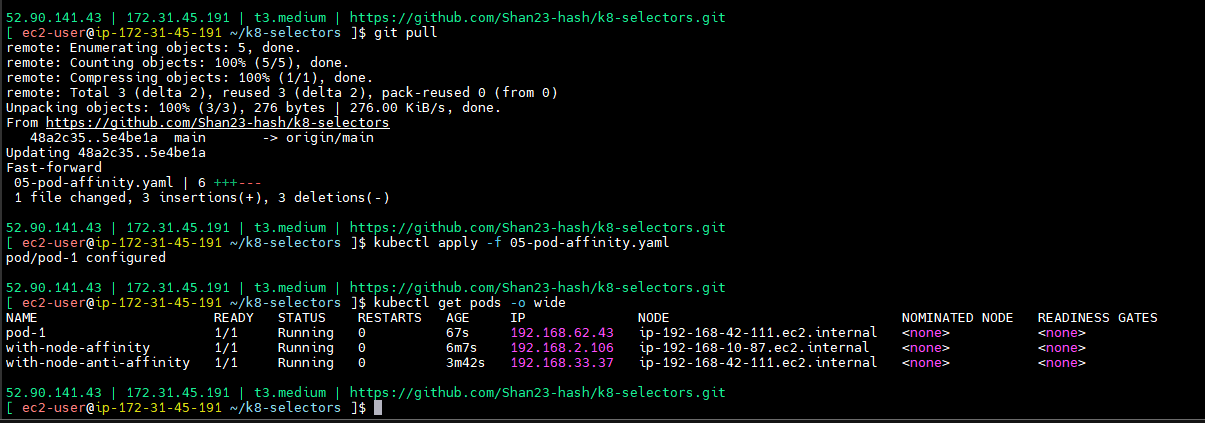
- name: pod-1

image: nginx

--> push and pull the code

**--> kubectl apply -f 05-pod-affinity.yaml**

**--> kubectl get pods -o wide**



--> I’m using label selectors

--> I want to use pod selectros

**05-pod-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: pod-1

labels:

app-name: pod-1

purpose: pod-affinity

spec:

containers:

- name: pod-1

image: nginx

---

apiVersion: v1

kind: Pod

metadata:

name: pod-2

labels:

app-name: pod-2

purpose: pod-affinity

spec:

affinity:

podAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app-nameee

operator: In

values:

- pod-1

topologyKey: "kubernetes.io/hostname"

containers:

- name: pod-2

image: nginx

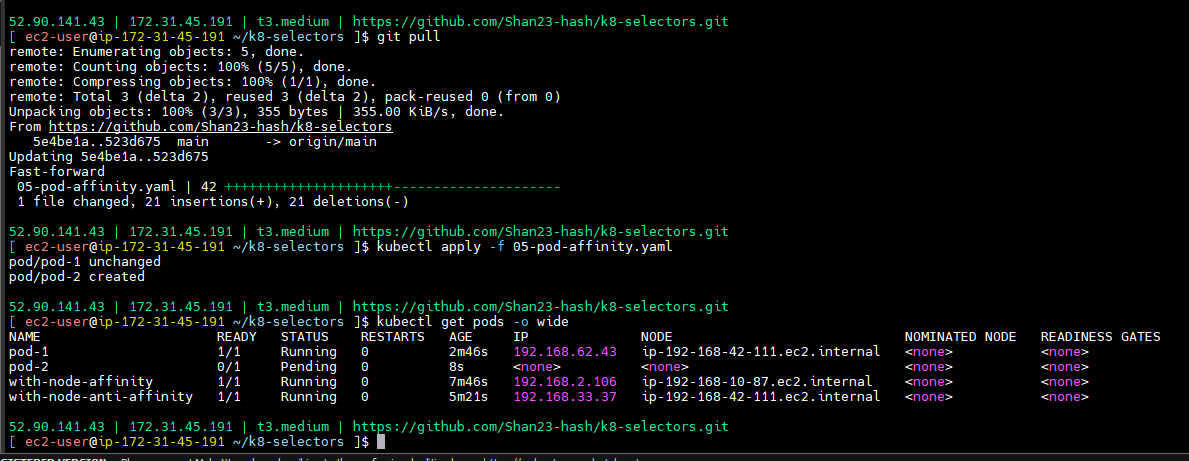
--> this will findout for hostname.

--> push and pull the code

**--> kubectl apply -f 05-pod-affinity.yaml**

**--> kubectl get pods -o wide**

--> pod-1 and pod-2 both are in same node



**05-pod-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: pod-1

labels:

app-name: pod-1

purpose: pod-affinity

spec:

containers:

- name: pod-1

image: nginx

---

apiVersion: v1

kind: Pod

metadata:

name: pod-2

labels:

app-name: pod-2

purpose: pod-affinity

spec:

affinity:

podAffinity:

# requiredDuringSchedulingIgnoredDuringExecution:

# - labelSelector:

# matchExpressions:

# - key: app-nameee

# operator: In

# values:

# - pod-1

# topologyKey: "kubernetes.io/hostname"

preferredDuringSchedulingIgnoredDuringExecution:

- weight: 1

podAffinityTerm:

labelSelector:

matchExpressions:

- key: app-nameee

operator: In

values:

- pod-1

topologyKey: "kubernetes.io/hostname"

containers:

- name: pod-2

image: nginx

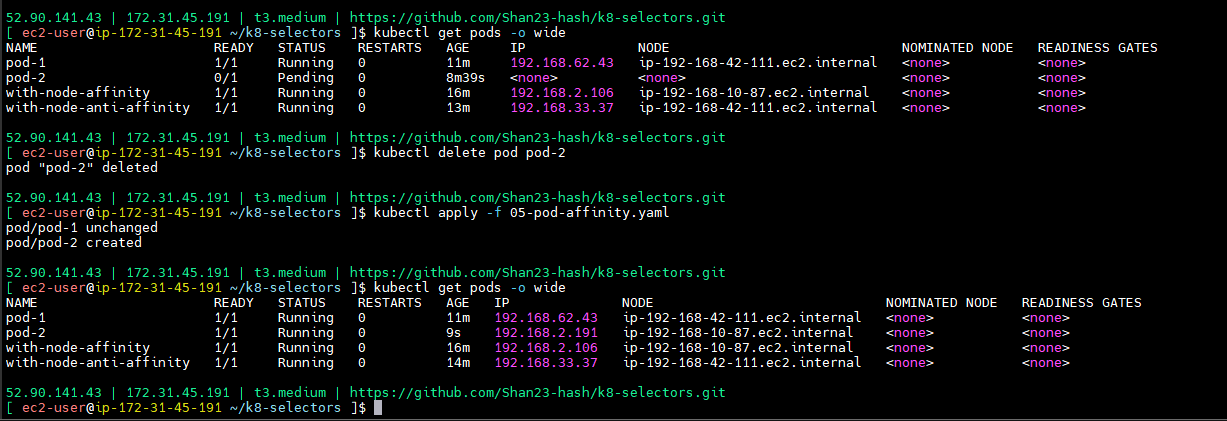
--> push and pull the code

**--> kubectl apply -f 05-pod-affinity.yaml**

**--> kubectl get pods -o wide**

**--> kubectl delete pod pod-2**

--> this is softrule



Application first will check in cache

application --> DB

application --> cache --> DB

**06-pod-anti-affinity.yaml**

apiVersion: v1

kind: Pod

metadata:

name: pod-1

labels:

app-name: pod-1

purpose: pod-affinity

spec:

containers:

- name: pod-1

image: nginx

---

apiVersion: v1

kind: Pod

metadata:

name: pod-2

labels:

app-name: pod-2

purpose: pod-anti-affinity

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app-name

operator: In

values:

- pod-1

topologyKey: "kubernetes.io/hostname"

containers:

- name: pod-2

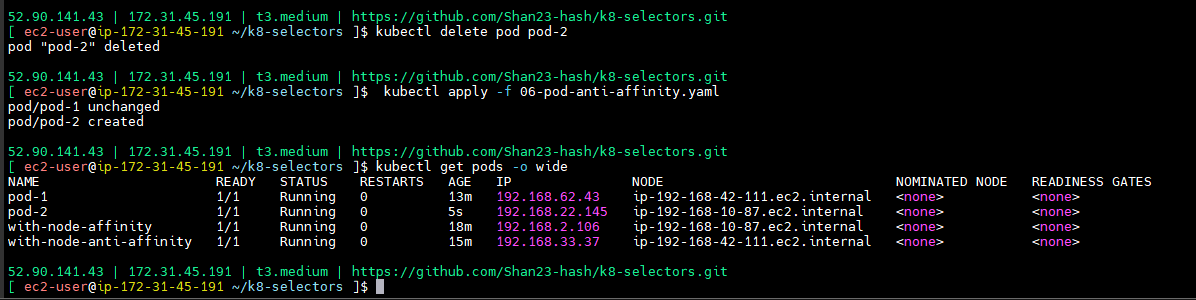
image: nginx

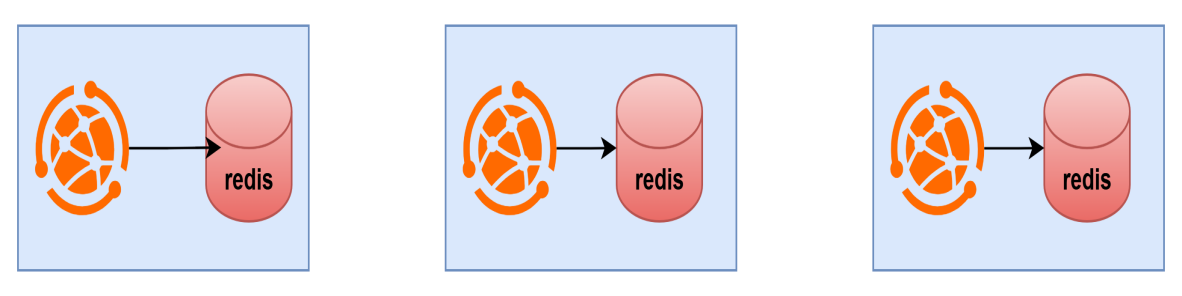
--> push and pull the code

**--> kubectl delete pod pod-2**

**--> kubectl apply -f 06-pod-anti-affinity.yaml**

**--> kubectl get pods -o wide**





**07-use-case.yaml**

apiVersion: apps/v1

kind: Deployment

metadata:

name: redis-cache

spec:

selector:

matchLabels:

app: store

replicas: 3

template:

metadata:

labels:

app: store

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- store

topologyKey: "kubernetes.io/hostname"

containers:

- name: redis-server

image: redis:3.2-alpine

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: web-server

spec:

selector:

matchLabels:

app: web-store

replicas: 3

template:

metadata:

labels:

app: web-store

spec:

affinity:

podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- web-store

topologyKey: "kubernetes.io/hostname"

podAffinity:

requiredDuringSchedulingIgnoredDuringExecution:

- labelSelector:

matchExpressions:

- key: app

operator: In

values:

- store

topologyKey: "kubernetes.io/hostname"

containers:

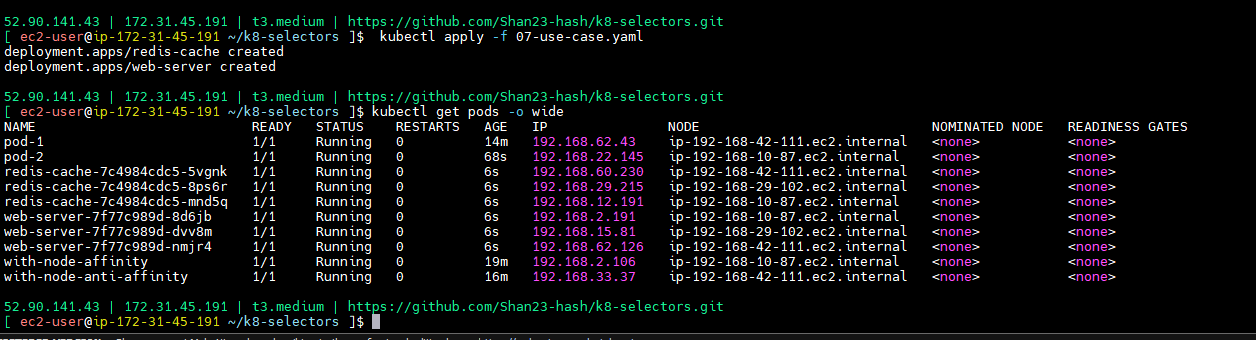
- name: web-app

image: nginx:1.16-alpine

--> push and pull the code

**--> kubectl apply -f 07-use-case.yaml**

**--> kubectl get pods -o wide**



--> all came to different nodes

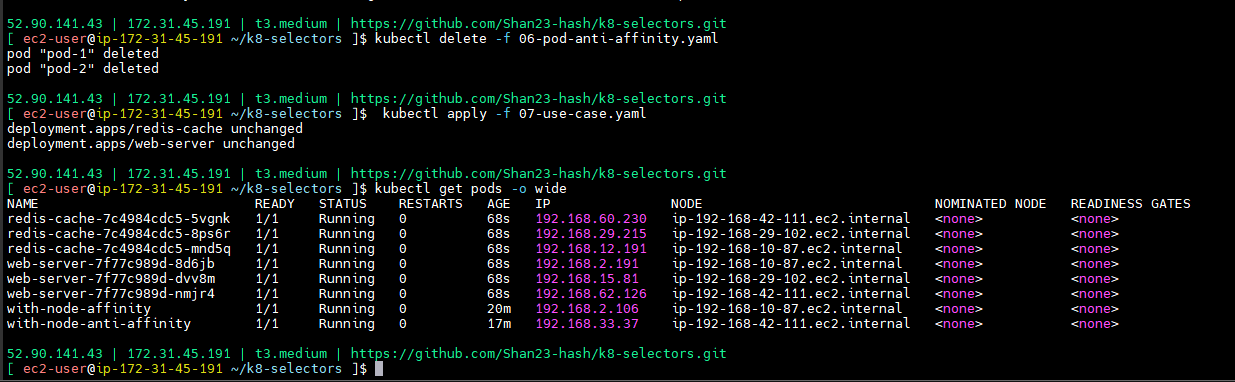
--> we are giving app in store

--> affinity and anti affinity both will use for application.

**--> kubectl delete -f 06-pod-anti-affinity.yaml**

**--> kubectl apply -f 07-use-case.yaml**

**--> kubectl get pods -o wide**



--> different nodes it will come.

--> use affinity and anti affinity will increase performance.

| **node-1** | **node-2** | **node-3** |
| --- | --- | --- |
| webserver-1 | webserver-2 | webserver-3 |
| cache-1 | cache-2 | cache-3 |

The overall effect is that each cache instance is likely to be accessed by a single client that is running on the same node. This approach aims to minimize both skew (imbalanced load) and latency.

You might have other reasons to use Pod anti-affinity. See the [ZooKeeper tutorial](https://kubernetes.io/docs/tutorials/stateful-application/zookeeper/" \l "tolerating-node-failure) for an example of a StatefulSet configured with anti-affinity for high availability, using the same technique as this example