1 Create ConfgMap or MongoDB EndPoint. (The MondoDB sevice name)

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: map
data:
  DB_URL: mongo-service
  clusterip: my-service
```

2 Create A secret or MongoDB User & PWD

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
data :
  USER_NAME: bW9uZ291c2Vy
  USER_PWD: bW9uZ29wYXNzd29yZA==
```

3 Create MongoDB Deployment Applicaton with Internal service (Clusterlp) Mongo DB needs username + password to operate Vars needed in mongoDB:
MONGO_INITDB_ROOT_USERNAME: root
MONGO_INITDB_ROOT_PASSWORD: example

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend
  labels:
    app: pod-mongo
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pod-mongo
  template:
    metadata:
      labels:
        app: pod-mongo
    spec:
      containers:
      - name: pod-mongo
        image: mongo:5.0
        ports:
        - containerPort: 8000
        env:
          - name: MONGO_INITDB_ROOT_USERNAME
            value: root
          - name: MONGO_INITDB_ROOT_PASSWORD
            value: example
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: pod-mongo
  ports:
    - protocol: TCP
      port: 8000
      targetPort: 8000
```

4 Create webApp Deployment(FrontEnd(with external service) and it needs to access MongoDb, so it needs username+ password + mongodb endpoint (mongodb service) container runs on 3000

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend
  labels:
    app: pod-webapp
spec:
  replicas: 1
  selector:
    matchLabels:
      app: pod-webapp
  template:
    metadata:
      labels:
        app: pod-webapp
    spec:
      containers:
      - name: pod-webapp
        image: nana˽anashia/k8s-demo-app:v1.0
        ports:
        - containerPort: 3000
        envFrom:
          - secretRef:
              name: mysecret
          - configMapRef:
              name: map
~
```

This is a picture of running all yaml files

```
controlplane $ vim secret-file
controlplane $ vim config-file
controlplane $ kubectl create -f secret-file
secret/mysecret created
controlplane $ kubectl create -f config-file
configmap/map created
controlplane $ vim deploy1
controlplane $ kubectl create -f deploy1
deployment.apps/backend created
controlplane $ vim clusterip
controlplane $ kubectl create -f clusterip
service/my-service created
controlplane $ vim deploy2
controlplane $ kubectl create -f deploy2
deployment.apps/frontend created
controlplane $
```

8- How many Nodes exist on the system?

two

```
controlplane $ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   8d    v1.26.0
node01         Ready    <none>          8d    v1.26.0
controlplane $
```

9- Do you see any taints on master ?

```
controlplane $ kubectl describe nodes controlplane , grep Taints
Taints:             node-role.kubernetes.io/control-plane:NoSchedule
controlplane $
```

10- Apply a label color=blue to the master node

```
controlplane $ kubectl taint nodes controlplane color=blue:NoSchedule
node/controlplane tainted
```

11- Create a new deployment named blue with the nginx image and 3 replicas Set Node Afnity to the deployment to place the pods on master only
NodeAfnity: requiredDuringSchedulingIgnoredDuringExecuton
Key: color
values: blue

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: blue
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
      affinity:
        nodeAffinity:
         requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: color
                operator: In
                values:
                - blue
```

12- Create a taint on node01 with key o spray, value o mortein and efect o NoSchedule

```
controlplane $ kubectl taint nodes node01 spray=mortein:NoSchedule
node/node01 tainted
controlplane $
```

13- Create a new pod with the NGINX image, and Pod name as mosquito

```
apiVersion: v1
kind: Pod
metadata:
  name: mosquito
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
~
```

14- What is the state o the mosquito POD?

```
controlplane $ kubectl get po mosquito
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          4m28s
controlplane $
```

15- Create another pod named bee with the NGINX image, which has a toleraton set to the taint Mortein Image name: nginx Key: spray Value: mortein Efect: NoSchedule Status: Running

```
apiVersion: v1
kind: Pod
metadata:
  name: bee
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
  tolerations:
  - key: spray
    value: mortein
    effect: "NoSchedule"
~
```