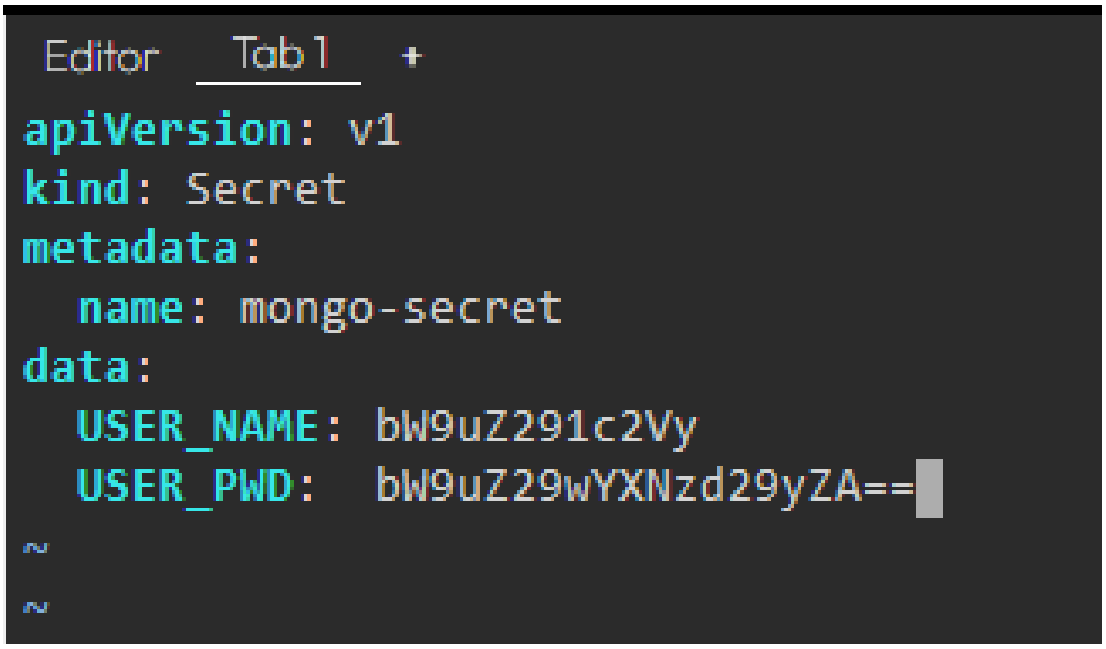
K8s_LAB03

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

2- Create A secret or MongoDB User & PWD:



3- Create MongoDB Deployment Applicaton with Internal service (ClusterIp) Mongo DB needs username + password to operate :

Editor Tab 1 +

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
      - name: mongodb_container
        image: mongo:5.0
      env:
      - name: MONGO_INITDB_ROOT_USERNAME
        valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: USER_NAME
      - name: MONGO_INITDB_ROOT_PASSWORD
        valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: USER_PWD
      envFrom:
        configMapRef: mongo-db
~
~
```

Editor Tab 1 +

```yaml
apiVersion: v1
kind: Service
metadata:
  name: mongo-service
spec:
  type: ClusterIp
  selector:
    name: mongo_db
  ports:
    - protocol: 80
      port: 80
      targetPort: 30022
~
~
```

Editor Tab 1 +

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongo-secret
data:
  USER_NAME: cm9vdA==
  USER_PWD: ZXhhbXBsZQ==
~
```

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

Editor Tab 1 +

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webApp-deployment
  labels:
    app: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: webApp_container
        image: nanajanashia/k8s-demo-app:v1.0
      env:
      - name: MONGO_INITDB_ROOT_USERNAME
        valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: USER_NAME
      - name: MONGO_INITDB_ROOT_PASSWORD
        valueFrom:
            secretKeyRef:
              name: mongo-secret
              key: USER_PWD
      envFrom:
        configMapRef: mongo-db
~
~
```

Editor Tab 1 +

```yaml
apiVersion: v1
kind: Service
metadata:
  name: webApp-NodePort
spec:
  type: NodePort
  ports:
  - port: 3000
  targetPort: 3000
  nodePort: 30022
~
~
~
~
```

8- How many Nodes exist on the system?

Answer: 2 Nodes

```
controlplane $ kubectl get Nodes
NAME           STATUS   ROLES           AGE    VERSION
controlplane   Ready    control-plane   5d2h   v1.26.0
node01         Ready    <none>          5d2h   v1.26.0
controlplane $
```

9- Do you see any taints on master ?

Answer: No

```
controlplane $ kubectl describe nodes controlplane | grep 'Taints'
Taints:             <none>
controlplane $
```

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

```
controlplane $ kubectl taint node controlplane color=blue:NoSchedule
node/controlplane tainted
controlplane $ kubectl describe nodes controlplane | grep 'Taints'
Taints:             color=blue:NoSchedule
controlplane $
```

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

Editor Tab 1 +

```
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
    spec:
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
              nodeSelectorTerms:
              - matchExpressions:
                - key: color
                  operator: In
                  values:
                  - blue
      containers:
      - name: nginx
        image: nginx
```

```
controlplane $ vim blue.yaml
controlplane $ kubectl apply -f blue.yaml
deployment.apps/blue created
controlplane $ kubectl get deployment
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
blue   0/3     3            0           25s
controlplane $
```

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

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

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

```
Editor    Tab 1    +
apiVersion: v1
kind: Pod
metadata:
  name: mosquito
spec:
  containers:
  - name: mosquito
    image: nginx
~
~
~
```

```
controlplane $ vim mosquito.yaml
controlplane $ kubectl apply -f mosquito.yaml
pod/mosquito created
controlplane $
```

14- What is the state o the mosquito POD?

Answer: Running

```
NAME                    READY   STATUS    RESTARTS   AGE
blue-86d5d8d6d7-2w9cn   0/1     Pending   0          16m
blue-86d5d8d6d7-c5ccp   0/1     Pending   0          16m
blue-86d5d8d6d7-kl4mr   0/1     Pending   0          16m
mosquito                1/1     Running   0          58s
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

```
controlplane $ vim bee.yaml
controlplane $ kubectl apply -f bee.yaml
pod/bee created
controlplane $ kubectl get pods
NAME                    READY   STATUS    RESTARTS   AGE
bee                     1/1     Running   0          7s
blue-86d5d8d6d7-2w9cn   0/1     Pending   0          26m
blue-86d5d8d6d7-c5ccp   0/1     Pending   0          26m
blue-86d5d8d6d7-kl4mr   0/1     Pending   0          26m
mosquito                1/1     Running   0          11m
controlplane $
```

```
Editor   Tab 1   +
apiVersion: v1
kind: Pod
metadata:
  name: bee
spec:
  containers:
  - name: bee
    image: nginx
  tolerations:
  - key: "spray"
    operator: "Equal"
    value: "mortein"
    effect: "NoSchedule"
~
~
~
```