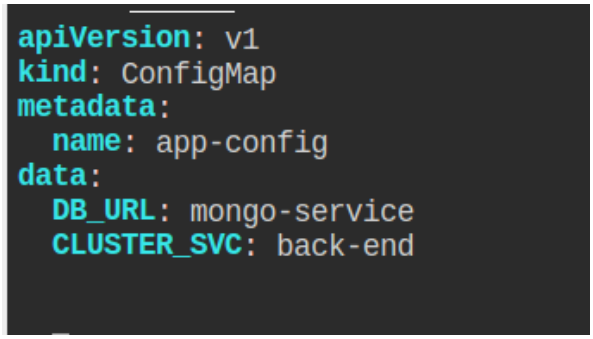
1 Create ConfigMap for MongoDB EndPoint. (The MondoDB sevice name)

`DB_URL:mongo-service`

`name of clusterIP service attached to db-deployment`

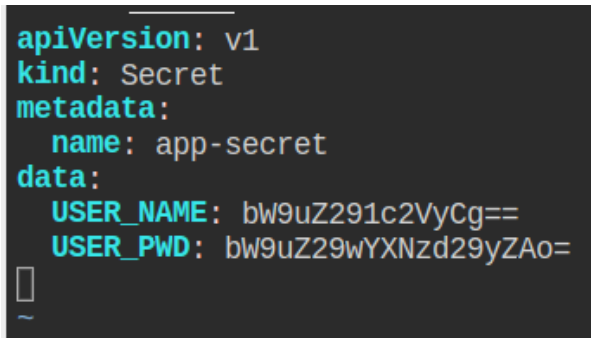


```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  DB_URL: mongo-service
  CLUSTER_SVC: back-end
```

2 Create A secret for MongoDB User & PWD

`USER_NAME:` mongouser

`USER_PWD: mongopassword`



```
apiVersion: v1
kind: Secret
metadata:
  name: app-secret
data:
  USER_NAME: bW9uZ291c2VyCg==
  USER_PWD: bW9uZ29wYXNzd29yZAo=
~
```

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

Vars needed in mongoDB:

```
MONGO_INITDB_ROOT_USERNAME: root

MONGO_INITDB_ROOT_PASSWORD: example
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ClusterIp
  selector:
    app: mongodb
  ports:
    - port: 27017
      targetPort: 27017
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
data:
  MONGO_INITDB_ROOT_USERNAME: cm9vdAo=
  MONGO_INITDB_ROOT_PASSWORD: ZXhhbXBsZQo=
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 3
  selector:
    matchLabels:
      app: mongodb-pod
  template:
    metadata:
      labels:
        app: mongodb-pod
    spec:
      containers:
      - name: mongodb
        image: mongo:5.0
        envFrom:
          - secretRef:
              name: mongodb-secret
```

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

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mongodb-data
data:
  DB_URL: mongo-service
```

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: frontend-secret
data:
  USER_NAME: c2FyYQo=
  PASSWORD: c2FyYTEyMzQK
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: frontend-svc
spec:
  selector:
    app: frontend
  type: NodePort
  ports:
    - port: 3000
      targetPort: 3000
      nodePort: 32000
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
  labels:
    app: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: frontend-pod
  template:
    metadata:
      labels:
        app: frontend-pod
    spec:
      containers:
      - name: frontend-pod
        image: nanajanashia/k8s-demo-app:v1.0
        envFrom:
          - secretRef:
              name: frontend-secret
          - configMapRef:
              name: mongodb-data
```

8- How many Nodes exist on the system?

```
controlplane $ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   33d   v1.26.0
node01         Ready    <none>          33d   v1.26.0
controlplane $
```

9- Do you see any taints on master ?

```
controlplane $ kubectl describe nodes controlplane | grep Taint
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 $
```

11- Create a new deployment named blue with the nginx image and 3 replicas

Set Node Affinity to the deployment to place the pods on master only

NodeAffinity: requiredDuringSchedulingIgnoredDuringExecution

Key: color

values: blue

```
controlplane $ kubectl label nodes node01 color=blue
node/node01 labeled
controlplane $
```

```yaml
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 of spray, value of mortein and efect of

NoSchedule

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

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

mosquito

```
controlplane $ kubectl run mosquito --image nginx
pod/mosquito created
```

14- What is the state of the mosquito POD?

```
controlplane $ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          92s
```

15- Create another pod named bee with the NGINX image, which has a toleration set to

the taint Mortein

Image name: nginx

Key: spray

Value: mortein

Efect: NoSchedule

Status: Running

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.14.2
  tolerations:
    - key: spray
      operator: "Equal"
      value: mortein
      effect: NoSchedule
```

```
controlplane $ vim pod.yml
controlplane $ kubectl apply -f pod.yml
pod/nginx created
controlplane $ kubectl get pods
NAME       READY   STATUS    RESTARTS   AGE
mosquito   0/1     Pending   0          7m53s
nginx      1/1     Running   0          5s
controlplane $
```